Université de Paris VII – Denis Diderot **UFR d'Informatique**

Théorie et Pratique de la Cryptologie sur Carte à Microprocesseur

MÉMOIRE

présenté et soutenu publiquement le 12 décembre 2003

pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches

par

Louis Goubin

Composition du jury

Rapporteurs :	Anca Muscholl Christof Paar Moti Yung
Examinateurs :	Marc Girault Jacques Patarin
Directeur de recherches :	Jacques Stern

Axalto (Schlumberger) – Département Recherche – Équipe Cryptographie et Sécurité Avancée

axalto

À mes parents,
dont le havre de paix
a permis à ce
mémoire d'exister.

Remerciements

Ce mémoire regroupe une partie des travaux que j'ai menés depuis sept ans au sein de l'équipe de recherche en cryptographie et sécurité avancée, qui a fait partie d'abord de Bull CP8 jusqu'en 2001, puis de Schlumberger, dont la division Cartes et Terminaux a pris le nom d'Axalto en septembre 2003.

Ces années ont été pour moi l'occasion d'essayer de maintenir un équilibre entre mathématiques et informatique, entre théorie et pratique, et entre recherche fondamentale et applications industrielles. Ceci n'aurait pas été possible sans l'aide de nombreuses personnes, auxquelles je tiens ici à exprimer toute ma gratitude.

Tout d'abord, c'est à Jacques Stern que j'adresse mes remerciements. C'est lui qui m'a prodigué ses conseils quand, après ma thèse de doctorat en mathématiques pures, j'ai souhaité m'orienter vers la cryptologie. Il m'a accueilli dans la communauté des cryptologues, et c'est lui qui m'a permis de rédiger ce mémoire d'habilitation, en tant que directeur de recherches.

Je dois beaucoup à Jacques Patarin, avec qui j'ai découvert le domaine nouveau de la cryptographie multivariable, et qui m'a accompagné dans mes travaux pendant près de cinq ans chez Bull CP8. Sa vision originale de la cryptographie et son enthousiasme communicatif m'ont beaucoup aidés, et je suis très heureux qu'il ait accepté de faire partie de ce jury.

Marc Girault a été aussi un interlocuteur de choix au cours de ces dernières années. Lors de projets communs, j'ai eu l'occasion d'apprécier ses grandes qualités scientifiques, pédagogiques et humaines. C'est toujours un grand plaisir de travailler avec lui, et je le remercie également d'avoir accepté de faire partie de ce jury.

Un nouveau projet, X-CRYPT, sera sans nul doute l'occasion de nouvelles collaborations fructueuses avec eux.

Je remercie aussi tout particulièrement Anca Muscholl, Christof Paar et Moti Yung, qui ont accepté la lourde tâche de rapporteur. Leur présence m'honore, et illustre le caractère résolument international de ce jury.

Je tiens au passage à rendre hommage au «Réseau National de Recherche en Télécommunications» (RNRT) qui a permis au projet Turbo-Signatures de voir le jour, et par là d'initier des échanges riches avec de nombreux autres cryptologues français : Olivier Baudron, Fabrice Boudot, Emmanuel Bresson, Laurent Frisch, Henri Gilbert, Jean-François Misarsky, Phong Nguyen, Guillaume Poupard et Jacques Traoré.

Je n'oublie pas non plus Eli Biham, Dario Catalano, Mathieu Ciet, Emmanuelle Dottax, Louis Granboulan, Gwenaëlle Martinet, David Pointcheval, Bart Preneel, Jean-Jacques Quisquater et Francesco Sica, pour d'éclairantes discussions lors notamment des projets européens Nessie et Stork, et aussi Jean-Sébastien Coron, Ronald Ferreira, Jean-Bernard Fischer, Helena Handschuh, Nick Howgrave-Graham, Pierre-Yvan Liardet, David Naccache, Pascal Paillier et William Whyte dans d'autres occasions.

Faire de la véritable recherche, avec «patience et longueur de temps», dans une entreprise privée n'a pas un caractère forcément évident, et je tiens à rendre hommage à ceux qui l'ont rendu possible. Tout d'abord – à tout seigneur tout honneur – Michel Ugon, inventeur de la carte à microprocesseur chez Bull CP8, qui m'a convaincu de tenter l'aventure, et a toujours suivi avec intérêt les travaux de notre équipe. J'ai toujours aimé ce mot d'esprit qu'il rapportait d'un ancien ingénieur en chef : «Vous l'avez fait parce que vous ne saviez pas que c'était impossible». Je remercie aussi Jean-Pierre Tual et Michel Thill, qui ont successivement supervisé

le fonctionnement de l'équipe cryptographie, en lui laissant toujours une liberté suffisante pour pouvoir se consacrer à des sujets de recherche à long terme.

Au sein de l'équipe cryptographie, dans une ambiance toujours agréable et motivante, je tiens à remercier tout particulièrement – outre Jacques Patarin – Mehdi-Laurent Akkar et Nicolas Courtois, toujours pleins d'inventivité et d'énergie, l'un pour les attaques physiques et leurs contre-mesures, l'autre pour la construction de schémas multivariables et la cryptanalyse d'algorithmes asymétriques ou même symétriques. J'associe à ces remerciements ceux qui ont fait partie plus brièvement de l'équipe : Guilhem Castagnos, Romain Duteuil, Yannick Leplard, Arnaud Leprince et Patrick Soquet.

Je tiens aussi à dire un grand merci aux collègues et amis qui m'ont aidé et soutenu pendant la rédaction de ce mémoire, et particulièrement Mehdi-Laurent Akkar, June Andronick, Jean-Sébastien Coron, Pascale Fine, Philippe Gateau, Béatrice et Guerric Giacomini, Magalie Lenoir, Olivier Ly, Christine et Laurent Nivert, David Pointcheval, Anne et Nicolas Vannieuwenhuyze. Que ceux que j'oublie veuillent bien me pardonner !

Ce mémoire d'habilitation est enfin l'occasion de citer l'ensemble de mes coauteurs, auxquels j'exprime toute ma reconnaissance : Mehdi-Laurent Akkar, Guilhem Castagnos, Jean-Sébastien Coron, Nicolas Courtois, Aviad Kipnis, Olivier Ly, Christian Mauduit, Willi Meier, Jacques Patarin, András Sárközy et Jean-Daniel Tacier.

Je voudrais conclure en soulignant une fois de plus la chance qu'a la communauté cryptographique de pouvoir travailler dans un climat de vraie coopération, que ce soit au plan national, européen ou international, où l'esprit de compétition n'est pas exclu, mais l'amitié non plus !

Louis Goubin, Mousterlin, juillet 2003.

Avant-propos

Ce document constitue le dossier en vue de l'obtention du diplôme d'habilitation à diriger des recherches, soumis à l'Université de Paris VII – Denis Diderot.

Il est composé de trois parties :

1. Une synthèse sur le thème principal de mes travaux effectués au cours de ces dernières années : *Théorie et pratique de la cryptologie sur carte à microprocesseur*. Ces travaux comprennent l'étude, la conception et l'implantation de nouveaux schémas cryptographiques particulièrement adaptés aux environnements contraints, notamment pour la signature électronique, la cryptanalyse de certains autres cryptosystèmes, et présentent une analyse des attaques physiques potentielles ainsi que des contre-mesures génériques pour garantir une sécurité réaliste dans les cartes à microprocesseur.
2. Un curriculum vitæ et une liste complète de mes publications ;
3. Une annexe donnant un résumé de la plupart de mes articles publiés. Ces articles illustrent les trois aspects principaux qui interviennent dans l'étude théorique et la réalisation pratique de mécanismes cryptographiques sur carte à microprocesseur :
   - de nouvelles méthodes de cryptanalyse (page 151) ;
   - la construction de nouveaux schémas cryptographiques (page 259) ;
   - la description d'attaques physiques et de contre-mesures (page 345).

Bien que ce document traite essentiellement de la cryptologie sur carte à microprocesseur, la liste complète de mes publications montre que mes contributions ne se limitent pas à ce seul domaine. J'ai en effet travaillé sur plusieurs autres sujets. J'ai notamment :

- publié plusieurs articles de mathématiques faisant suite à ma thèse de doctorat en théorie analytique des nombres (étude des sommes d'exponentielles à coefficients multiplicatifs, étude de suites binaires pseudo-aléatoires) ;
- étudié de nouvelles propriétés de l'algorithme DES ;
- proposé des méthodes génériques de sécurisation semi-automatique de code contre des attaques par injection de fautes.

Table des matières

I Théorie et pratique de la cryptologie sur carte à microprocesseur

Introduction **11**

Chapitre 1
La cryptologie **13**

1 Introduction à la cryptologie 13
1.1 Qu'est-ce que la cryptologie ? 13
1.2 Le chiffrement symétrique 14
2 La cryptographie asymétrique 15
2.1 Un nouveau paradigme 15
2.2 Protocoles d'authentification 16
2.3 Protocoles de signature 16
3 Le schéma RSA 17
3.1 Le problème RSA de base 18
3.2 Les protocoles construits autour du RSA 19
4 Signatures à base de logarithme discret 23
4.1 Notion de logarithme discret 23
4.2 Schémas génériques de signature 24
4.3 Transposition sur les courbes elliptiques 25

Chapitre 2
La carte à microprocesseur **27**

1 L'invention de la carte à puce . . . 27
2 Fonctionnement d'une carte à microprocesseur . . . 28
2.1 Description physique . . . 28
2.2 Communication avec la carte . . . 29
2.3 Format logique des commandes . . . 30
3 Performances pour la signature électronique . . . 30
3.1 Multiplications et exponentiations modulaires . . . 31
3.2 Temps de calcul sur une carte à microprocesseur . . . 31

Chapitre 3
Hypothèses calculatoires **33**

1 Introduction . . . 33
2 Systèmes d'équations quadratiques sur un corps fini . . . 34
2.1 Dans le corps $K = \mathbb{F}_2$ . . . 34
2.2 Dans un corps quelconque . . . 35
2.3 Systèmes de n équations quadratiques en $k \geq n$ variables . . . 36
3 Isomorphismes de polynômes . . . 37
3.1 Les problèmes IP et MP . . . 37
3.2 IP à un secret est au moins aussi difficile que les Isomorphismes de Graphes . . . 39
3.3 MP est NP-difficile . . . 41
3.4 Le problème de décision IP n'est pas NP-complet . . . 43
4 Le problème MinRank . . . 44
4.1 Définition du problème MinRank . . . 44
4.2 Complexité de MinRank . . . 45
5 Le problème de la décomposition fonctionnelle . . . 45
5.1 Motivation du problème . . . 45
5.2 L'algorithme de Dickerson . . . 46
5.3 Question de la NP-difficulté . . . 46

Chapitre 4
Cryptanalyse **49**

1 Introduction . . . 50
2 Attaques par polarisation . . . 50
2.1 Principe général . . . 50

2.2 L'algorithme de chiffrement asymétrique D^* . . . 50
2.3 Attaque de l'algorithme D^* par polarisation . . . 51
2.4 Généralisation . . . 52
2.5 Cryptanalyse de l'algorithme C^*_- par polarisation . . . 53
3 Dégénérescence des formes quadratiques . . . 54
3.1 Le problème de dégénérescence (DP) . . . 54
3.2 Trois algorithmes polynomiaux . . . 54
3.3 Application à la cryptanalyse de trois schémas asymétriques . . . 56
3.4 Extension aux schémas à deux tours . . . 57
4 Algorithmes pour le problème MinRank . . . 58
4.1 État de l'art . . . 58
4.2 La méthode du noyau . . . 59
4.3 Application au cryptosystème TTM . . . 59
5 Algorithmes pour les systèmes quadratiques sous-définis . . . 60
5.1 Schémas de signature et systèmes sous-définis . . . 60
5.2 Les méthodes générales . . . 61
5.3 Impact sur les schémas FLASH, SFLASH et UOV . . . 62
5.4 Le cas massivement sous-défini . . . 62
5.5 Un nouveau problème difficile ? . . . 64

Chapitre 5
Construction d'algorithmes dédiés **65**

1 Introduction . . . 65
2 L'algorithme QUARTZ . . . 66
2.1 Signatures courtes . . . 66
2.2 Description de l'algorithme QUARTZ . . . 67
2.3 Résolution d'une équation polynomiale dans un corps fini . . . 69
2.4 Preuves relatives de sécurité . . . 70
2.5 Difficulté du problème MQ pour QUARTZ . . . 70
3 L'algorithme SFLASH . . . 71
3.1 Un algorithme de signature rapide . . . 71
3.2 Description de l'algorithme SFLASH . . . 71
3.3 Preuves relatives de sécurité . . . 72
3.4 Implémentation optimisée sur carte à microprocesseur . . . 74
4 Bilan pour la signature sur carte à microprocesseur . . . 76

Chapitre 6
Attaques physiques **79**

1 Introduction . . . 79
2 Classification des attaques physiques . . . 80
2.1 Attaques invasives ou non-invasives . . . 80
2.2 Attaques actives ou passives . . . 81
3 Attaques par injection de fautes . . . 82
4 Attaques par analyse de consommation électrique . . . 83
4.1 Analyse élémentaire de la consommation . . . 84
4.2 Analyse différentielle de la consommation . . . 84
4.3 Exemple de l'algorithme DES . . . 85
5 Représentations booléennes et arithmétiques . . . 86
5.1 Le problème de conversion . . . 86
5.2 La méthode de Messerges . . . 86
5.3 Attaque par analyse de consommation . . . 87
6 Attaques physiques sur les courbes elliptiques . . . 88
6.1 État de l'art des protections . . . 88
6.2 Une attaque à message choisi . . . 89
6.3 Application à des courbes elliptiques standards . . . 91

Chapitre 7
Contre-mesures **93**

1 Introduction . . . 93
2 La méthode de duplication . . . 94
2.1 Contre-mesures usuelles . . . 94
2.2 Principe de la méthode de duplication . . . 94
2.3 Application à l'algorithme DES . . . 95
2.4 Application à l'algorithme RSA . . . 97
3 Conversions entre deux structures algébriques . . . 98
3.1 Implémentation sécurisée de l'algorithme SFLASH . . . 99
3.2 Le problème du zéro pour l'AES . . . 101
3.3 Une méthode de conversion universelle . . . 102
3.4 Application pratique . . . 105
4 Les attaques par analyse différentielle d'ordre supérieur . . . 105
4.1 Une généralisation de la DPA . . . 105
4.2 Insuffisance des contre-mesures «classiques» . . . 106

4.3 Une contre-mesure pour les attaques d'ordre supérieur 107

Conclusion et perspectives **111**

Termes et abréviations **113**

Liste des noms propres **117**

Références **121**

II Curriculum vitć et publications

Curriculum vitæ

1 État civil 141
2 Formation 141
3 Cursus professionnel 142
4 Valorisation de la recherche 143
5 Encadrement de la recherche 143
6 Enseignement 143

Liste de publications

1 Articles dans des journaux 145
2 Articles présentés à des congrès internationaux avec comité de lecture 145
3 Articles présentés à des colloques 146
4 Travaux de vulgarisation 147
5 Mémoires et rapports 147
6 Propositions à des organismes de normalisation 147
7 Brevets 148

III Annexe : articles joints

Nouvelles méthodes de cryptanalyse **151**

Trapdoor One-Way Permutations and Multivariate Polynomials

ICICS'97 (Version Complète) 153

Asymmetric Cryptography with S-Boxes

ICICS'97 (Version Complète) 177

Unbalanced Oil and Vinegar Signature Schemes

EUROCRYPT'99 (Version Complète) 207

Cryptanalysis of the TTM Cryptosystem

ASIACRYPT'2000 229

Solving Underdefined Systems of Multivariate Quadratic Equations

PKC'2002 243

Construction de nouveaux schémas cryptographiques **259**

Improved Algorithms for Isomorphisms of Polynomials

EUROCRYPT'98 (Version complète) 261

C^*_{-+} and *HM*: Variations around two schemes of T. Matsumoto and H. Imai

ASIACRYPT'98 (Version Complète) 287

QUARTZ, 128-bit long digital signatures

CT-RSA'2001 (Version Complète) 309

FLASH, a fast multivariate signature algorithm

CT-RSA'2001 (Version Complète) 323

A Fast and Secure Implementation of SFLASH

PKC'2003 333

Attaques physiques et contre-mesures **345**

DES and Differential Power Analysis

CHES'99 347

On Boolean and Arithmetic Masking against Differential Power Analysis

CHES'2000 361

A Sound Method for Switching between Boolean and Arithmetic Masking

CHES'2001 367

A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems

PKC'2003 379

A Generic Protection against High-Order Differential Power Analysis

FSE'2003 391

Première partie

Théorie et pratique de la cryptologie sur carte à microprocesseur

Introduction

Le système doit être matériellement, sinon mathématiquement, indéchiffrable. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi. [...] Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.

Auguste Kerckhoffs, *La cryptographie militaire* (1883)

On peut considérer que les principes édictés par Kerckhoffs à la fin du XIX^{e} siècle marquent la naissance de la cryptologie au sens moderne du terme. On peut en tout cas les apercevoir en filigrane tout au long de ce mémoire, qui présente les travaux de recherche que j'ai effectués au cours de ces dernières années sur la théorie et la pratique de la cryptologie sur carte à microprocesseur. Le texte qui suit se veut un itinéraire à travers toutes les étapes à parcourir, depuis les fondements mathématiques de la cryptographie jusqu'aux conséquences de l'architecture électronique de la carte à microprocesseur sur la sécurité.

Tout d'abord, la cryptologie. Nous la décrirons brièvement au chapitre 1. Il est maintenant naturel de postuler que le système «puisse sans inconvénient tomber entre les mains de l'ennemi». Et donc de considérer des algorithmes cryptographiques connus de tous, les informations secrètes étant concentrées dans une clé. Ce principe, poussé à l'extrême, fournit la notion de cryptographie asymétrique, ses avatars les plus célèbres étant RSA et les systèmes à base de courbes elliptiques.

«Il faut qu'il soit portatif» : on ne pourrait trouver meilleure illustration de ce principe que la carte à microprocesseur. Son invention, son fonctionnement et ses spécificités cryptographiques seront évoquées au chapitre 2. C'est encore le meilleur moyen qui ait été trouvé pour garder secrète une clé au sein d'un dispositif embarqué, tout en proposant des capacités de calcul cryptographique réelles.

«Le système doit être matériellement, sinon mathématiquement indéchiffrable». Dans le chapitre 3 seront résumés les travaux que j'ai effectués sur les hypothèses calculatoires. Dans le cadre de la théorie de la complexité, de nouveaux problèmes difficiles sont mis en évidence, notamment pour les systèmes d'équations polynomiales à plusieurs variables. Ce sont les briques de base, à partir desquelles on peut construire de nombreux nouveaux cryptosystèmes aux propriétés inédites, notamment pour tenir compte des contraintes de mémoire ou de temps propres aux cartes à puce.

La cryptographie multivariable, sujet d'un certain nombre de mes travaux, crée de multiples possibilités nouvelles, que ce soit pour le chiffrement, la signature ou l'authentification. Mais

des attaques nouvelles se font aussi jour, comme l'ont montré plusieurs exemples spectaculaires. Dans le chapitre 4, je présente une typologie des méthodes nouvelles de cryptanalyse que j'ai mises au point. L'objectif de ces attaques est d'éprouver la solidité conceptuelle des schémas cryptographiques multivariables que l'on propose.

Une fois posées les bases calculatoires, et ayant à disposition toute une batterie de techniques cryptanalytiques adaptées, manière de «banc d'essai», une construction saine de nouveaux cryptosystèmes multivariables est possible, comme nous l'illustrons au chapitre 5. La complexité des attaques, évaluée de façon précise, permet de choisir des paramètres concrets pour les différentes composantes des algorithmes, avec une sécurité pratique bien contrôlée.

Un certain nombre de mes travaux sont nés d'une rencontre entre la cryptologie et les cartes à microprocesseur, entre les attaques mathématiques et les analyses de comportement physique, entre les hypothèses conceptuelles et la réalité des implémentations. Ainsi sont abordées au chapitre 6 les attaques physiques, notamment par analyse de la consommation électrique. Leur modélisation et leur application pratique est illustrée sur différents cryptosystèmes, symétriques et asymétriques.

Pour terminer ce parcours vers une cryptographie sûre dans les cartes à microprocesseur, la dernière pièce du puzzle, au chapitre 7, concerne les contre-mesures que l'on peut mettre en place pour faire face aux attaques physiques. Dans plusieurs types de scénarios, j'ai décrit des méthodes générales qui permettent de sécuriser les implémentations d'algorithmes cryptographiques dans les cartes à microprocesseur, y compris contre les généralisations les plus récentes d'attaques par analyse de la consommation électrique ou du rayonnement électromagnétique.

Bonne lecture !

Chapitre 1

La cryptologie

Real mathematics has no effects on war. No one has yet discovered warlike purpose to be served by the theory of numbers [...], and it seems very unlikely that anyone will do so for many years.

Godfrey Harold Hardy, *A Mathematician's Apology* (1940)

The success achieved [by Allied cryptanalysts] merits the highest commendation, and all witnesses familiar with [it] have testified that it contributed enormously to the defeat of the enemy, greatly shortened the war, and saved many thousands of lives.

Rapport du Congrès américain sur Pearl Harbor (1945)

Sommaire

1 Introduction à la cryptologie 13
1.1 Qu'est-ce que la cryptologie? 13
1.2 Le chiffrement symétrique 14
2 La cryptographie asymétrique 15
2.1 Un nouveau paradigme 15
2.2 Protocoles d'authentification 16
2.3 Protocoles de signature 16
3 Le schéma RSA 17
3.1 Le problème RSA de base 18
3.2 Les protocoles construits autour du RSA 19
4 Signatures à base de logarithme discret 23
4.1 Notion de logarithme discret 23
4.2 Schémas génériques de signature 24
4.3 Transposition sur les courbes elliptiques 25

1 Introduction à la cryptologie

1.1 Qu'est-ce que la cryptologie?

Pour analyser le sens du mot *cryptologie*, il est utile de se référer à son étymologie, qui s'appuie sur les mots grecs $\chi\rho\upsilon\pi\tau\acute{o}\zeta$ (*kryptos* = caché) et $\lambda\acute{o}\gamma o\zeta$ (*logos* = discours, traité). On peut ainsi définir la cryptologie comme *science du secret* : elle s'appuie sur la possibilité de transmettre un message tout en dissimulant son contenu pour un observateur indiscret.

Plus précisément, dans sa conception moderne, la cryptologie a essentiellement pour objet l'étude de trois problématiques qui en constituent les véritables piliers (*cf* «la trilogie fondamentale de la cryptologie moderne» dans *La Science du Secret*, de Jacques Stern [75]) : la *confidentialité*, l'*authenticité* et l'*intégrité* de l'information. Pour répondre à ces trois exigences, trois concepts importants ont émergé au fil du temps, et sont aujourd'hui considérés aujourd'hui comme les fonctionnalités fondamentales de la cryptologie :

- Le *chiffrement*, qui permet de cacher l'information contenue dans un message ;
- La *signature électronique*, qui permet de prouver l'identité de l'auteur d'un message, et de garantir la non-répudiation.
- L'*authentification*, qui permet de prouver son identité, lors d'un contrôle d'accès.

Pour compléter cette typologie, remarquons qu'il est d'usage de distinguer deux facettes pour la cryptologie :

- La *cryptographie*, qui consiste à concevoir et à mettre au point des mécanismes cryptologiques adaptés afin d'assurer une ou plusieurs des trois notions de sécurités décrites précédemment [1]. Concrètement, ces mécanismes sont en général décrits de façon algorithmique, et font appel à des notions mathématiques, allant des probabilités à la théorie des nombres, en passant par la théorie de la complexité, la combinatoire, la théorie des codes correcteurs d'erreurs, la théorie de la réduction des réseaux, les corps finis, les polynômes multivariables, les courbes elliptiques et hyperelliptiques, la géométrie algébrique, etc ;
- La *cryptanalyse*, qui consiste à évaluer la résistance des méthodes mises au point par la cryptographie, contre les attaques. Une partie consiste à appliquer des scénarios d'attaque, et à les appliquer pour mettre à l'épreuve les algorithmes. Il peut s'agir d'attaques purement mathématiques, ou bien d'attaques faisant intervenir en outre la façon dont l'algorithme est implanté sur un système électronique (on parle alors d'«attaques physiques»). Cette façon de voir la cryptanalyse comme tentative d'attaque est le plus souvent heuristique : on n'a pas besoin de prouver rigoureusement que l'attaque est valable : il suffit qu'elle de constater empiriquement qu'elle fonctionne [2]. Toutefois, depuis quelques années, les «preuves de sécurité» sont également considérées comme partie intégrante de la cryptanalyse.

1.2 Le chiffrement symétrique

Dans la *cryptographie symétrique*, les clés de chiffrement et de déchiffrement sont identiques (*cf* Figure 1). Elles doivent donc toutes deux être gardées secrètes, ce qui fait qu'on parle également ici de *cryptographie à clé secrète*.

C'est le domaine le plus ancien de la cryptographie (il est aussi connu sous le nom de *cryptographie conventionnelle*) et de nombreux algorithmes appartiennent à cette catégorie. Citons les trois les plus importants actuellement.

En 1977, le gouvernement américain [3] a publié et standardisé l'algorithme DES (*Data Encryption Standard* [52, 53]). Celui-ci a fait l'objet d'un effort de recherche considérable pendant plus de 25 ans, et il est toujours considéré comme excellent. Même si la taille de la clé (56 bits) est devenue aujourd'hui trop courte, la variante appelée *Triple-DES* continue à être largement utilisée.

1. Ainsi que d'autres objectifs plus spécialisés, tels l'anonymat, le *broadcasting*, le *traitor tracing*, etc.

2. Notons toutefois que pour certaines attaques récentes, notamment celle proposée par Nicolas Courtois et Joseph Pieprzyk contre l'algorithme AES [18], il n'est pas possible de vérifier expérimentalement leur efficacité, ce qui crée une situation inédite et suscite un certain débat...

3. Plus précisément le NBS, National Bureau of standards, ancêtre du NIST, National Institute of Standards and Technology.

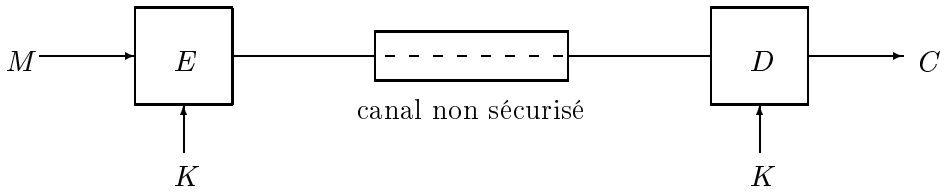


FIG. 1 – *Le chiffrement symétrique : schéma général (la clé secrète commune est notée K)*

L'algorithme AES (*Advanced Encryption Standard* [19, 58]) est appelé à remplacer le DES dans les prochaines années. Conçu pour être à la fois plus rapide et plus polyvalent que le Triple-DES, il fait actuellement l'objet d'une grande attention parmi les chercheurs. En particulier, de nouvelles attaques [18], exploitant certaines propriétés algébriques de cet algorithme, ont été récemment proposées et pourraient constituer une faiblesse.

Le projet européen NESSIE[4] [54], terminé en 2003, recommande en plus de l'AES, une autre solution de chiffrement avec une clé de 128 bits : Camellia [1]. Notons que cet algorithme utilise essentiellement les mêmes composantes potentiellement vulnérables aux attaques algébriques que l'AES.

2 La cryptographie asymétrique

2.1 Un nouveau paradigme

Le principe de la *cryptographie asymétrique* consiste, dans le cadre du chiffrement, à rendre publique la clé qui sert à la fonction de chiffrement. Naturellement, la clé de déchiffrement doit elle rester secrète sous peine de perdre complètement tout espoir d'assurer la confidentialité des messages. Les deux clés étant nécessairement différentes, ce qui justifie le qualificatif d'*asymétrique*. Cette branche de la cryptologie est souvent appelée également *cryptographie à clé publique* (ou même parfois cryptographie à clé publique-clé privée).

Remarquons qu'un attaquant disposant du message chiffré C connaît tout à la fois la fonction qui a servi à chiffrer, et la clé (publique) qui a été utilisée (*cf* Figure 2). C'est même encore plus inquiétant : il existe en général une relation connue entre la clé publique et la clé privée ! La situation semble donc paradoxale, et avec le recul, on s'aperçoit qu'une solution n'a pu émerger qu'avec le développement de la *théorie de la complexité*, qui rend plausible l'existence de problèmes mathématiques intrinsèquement difficiles. C'est ainsi qu'il n'est pas *en théorie* (*i.e.* en supposant que l'ennemi a une puissance de calcul infinie) impossible de retrouver le message clair M à partir de son chiffré C, mais *en pratique* cela demande une puissance de calcul que l'on espère non polynomiale (et même exponentielle dans certains cas) en la taille du message.

Aussi n'est-ce qu'en 1976 – dans leur célèbre article fondateur [23] – que Whitfield Diffie et Martin Hellman ont montré la possibilité théorique de la cryptographie à clé publique, en l'illustrant dans le cas particulier d'un protocole d'échange de clés[5]. L'algorithme RSA, inventé

4. New European Schemes for Signature, Integrity, and Encryption

5. Notons qu'un protocole avec les mêmes propriétés avait été décrit un peu auparavant par Ralph Merkle [48]. Les *puzzles de Merkle* avaient toutefois un moins bon rapport sécurité/performance.

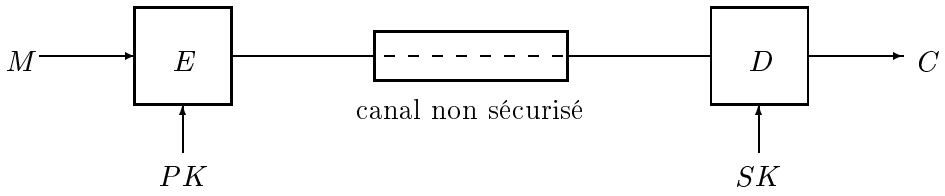


FIG. 2 – *Le chiffrement asymétrique : schéma général (la clé publique et la clé privée sont respectivement notées PK et SK)*

en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman [29, 67] est reconnu comme étant le premier algorithme publié[6] de chiffrement asymétrique réellement praticable[7].

Par rapport aux systèmes symétriques, le chiffrement à clé publique présente le grand avantage de ne pas nécessiter de mise en accord préalable entre les interlocuteurs qui souhaitent échanger des messages. Mais la plus grande nouveauté apportée par la cryptographie asymétrique est la possibilité de réaliser des protocoles d'authentification et de signature, répondant ainsi de manière spectaculaire aux besoins d'intégrité et d'authenticité.

2.2 Protocoles d'authentification

Pour s'authentifier, c'est-à-dire prouver son identité, la méthode générale consiste ici à prouver (de façon interactive) la connaissance de la clé privée associée à la clé publique.

Si l'on dispose d'une fonction de chiffrement asymétrique, on peut facilement en déduire un protocole d'authentification[8]. Cependant, il paraît souhaitable d'exiger du protocole d'authentification qu'il ne révèle aucune information sur le secret lui-même : c'est ce paradoxe (un de plus !) qu'ont résolu Shafi Goldwasser, Silvio Micali et Charles Rackoff en 1985 [33], en décrivant une technique générale de preuve «*à divulgation nulle de connaissance*» (ou *zero-knowledge* en anglais)[9].

Des exemples pratiques de preuve zero-knowledge ont été ensuite donnés par Amos Fiat et Adi Shamir en 1986 [26], puis Louis Guillou et Jean-Jacques Quisquater en 1988 [34] et 2000 [66], pour la connaissance de la factorisation d'un module, et par Claus Schnorr en 1989 [70, 71], ou encore Marc Girault, Guillaume Poupard et Jacques Stern en 1998 [31, 65], pour la connaissance d'un logarithme discret.

2.3 Protocoles de signature

Pour signer un message, il s'agit de prouver que l'on en est bien l'émetteur (ce que l'on ne pourra plus nier par la suite : c'est la *non-répudiation*). Là encore, il est possible de construire un

6. On sait maintenant [25] qu'au sein du service du chiffre britannique, James Ellis avait établi dès janvier 1970 la possibilité de la cryptographie asymétrique, et que par la suite Clifford Cocks a inventé en 1973 une variante de RSA, puis Malcolm Williamson a décrit en 1974 une variante du futur protocole d'échange de clés Diffie-Hellman.

7. Voir le paragraphe 2 pour une présentation plus détaillée de l'algorithme RSA pour le chiffrement et la signature.

8. Remarquons que le concept de cryptographie symétrique permet également de prouver sa connaissance de la clé secrète, mais uniquement à l'interlocuteur qui partage la même clé.

9. En fait, l'idée avait été proposée dès 1984 par Fischer, Micali et Rackoff lors de la "Rump Session" de la conférence Eurocrypt, mais leur article ne faut publié que 12 (!) ans après [27].

protocole de signature à partir d'une fonction de chiffrement asymétrique[10] (par exemple RSA, voir le paragraphe 2.2).

Par ailleurs, on peut montrer que toute preuve interactive de connaissance peut-être rendue non-interactive. C'est ainsi que le protocole de preuve de connaissance d'un logarithme discret de Schnorr a donné un schéma de signature, dont une variante est devenue le standard américain DSA [56, 57]. L'avantage de ce protocole est qu'il peut être transposé à n'importe quel groupe commutatif sur lequel le problème du logarithme discret est difficile : il peut en particulier s'adapter aux courbes elliptiques (voir paragraphe 3).

3 Le schéma RSA

L'algorithme RSA[11] , inventé par Ronald Rivest, Adi Shamir et Leonard Adleman [67], a été présenté publiquement pour la première fois dans le numéro d'août 1977 de la revue *Scientific American* [29]. C'est actuellement le cryptosystème à clé publique le plus utilisé au monde. On le trouve dans un nombre toujours croissant de produits commerciaux liés à la sécurisation des échanges de données sur Internet, à la protection de la confidentialité et de l'authenticité du courrier électronique, au paiement électronique au moyen de cartes à puce, etc.

Cela fait maintenant 25 ans que la communauté cryptographique mondiale étudie la solidité du RSA face à toute une panoplie d'attaques. En étudiant de plus près les résultats obtenus, on peut diviser ces attaques en trois catégories :

1. Les attaques sur le «problème RSA» lui-même, qui cherchent à trouver une faille dans les fondements mathématiques mêmes de l'algorithme. Il s'agit alors d'attaques qui parviennent par exemple à retrouver la clé privée de l'algorithme, uniquement à partir de données d'entrée/sortie de la fonction RSA, ou bien à être capable de retrouver la valeur d'entrée à partir de la valeur de sortie (sans avoir besoin de retrouver la clé privée).
2. Les attaques sur les protocoles construits autour du RSA : pour réaliser des applications de signature, d'authentification ou de chiffrement, il est en effet nécessaire de décrire avec précision la façon dont la fonction RSA de base est utilisée. Ce sont parfois ces protocoles qui sont mis à mal par des attaques, qui en revanche ne remettent pas en cause l'algorithme RSA lui-même.
3. Les attaques sur les implémentations du RSA : même si le RSA est correctement utilisé dans un protocole de chiffrement ou de signature, son implémentation concrète dans un composant électronique donné, par exemple dans une carte à puce, peut dans certains cas faire l'objet d'attaques d'un autre type, qui tirent parti d'informations physiques que le composant laisse «filtrer»: temps de calcul, consommation électrique, injection d'erreurs de calcul[12].

10. Là encore, dans le cadre de la cryptographie symétrique, on peut aussi prouver qu'on a bien émis un message donné, mais ceci uniquement auprès de l'interlocuteur qui partage la même clé, ou bien d'une autorité qui connaît cette clé (par exemple l'autorité qui l'a générée). On parle alors de MAC (*Message Authentication Code*). De plus, la non-répudiation est affaiblie : chacun des deux possesseurs de la clé secrète peut prétendre que c'est l'autre qui a produit le MAC.

11. Ce paragraphe est extrait de l'article *Stratégies d'attaques* paru dans la revue *Pour la Science*, numéro spécial *Cryptographie*, pages 58-61, juillet 2002. ©Pour la Science, 2002.

12. Voir Chapitre 6. Attaques physiques.

3.1 Le problème RSA de base

Mathématiquement, on peut décrire l'algorithme RSA de la manière suivante. On commence par choisir l'exposant public e (des exemples courants sont $e = 3$, $e = 17$, $e = 257$ ou $e = 65537$). On utilise ensuite un générateur de nombres aléatoires pour obtenir deux nombres premiers p et q tels que e soit premier avec $p - 1$ et avec $q - 1$. Si on pose $n = p \times q$, la clé publique est alors constituée de e et de n, alors que la clé secrète (ou privée) est constituée de p et q. Typiquement on prend souvent actuellement p et q de taille 512 bits, et donc n de taille 1024 bits. La fonction de chiffrement est alors définie par $f : \ x \mapsto y = x^e \bmod n$ et la fonction de déchiffrement par $f^{-1} : \ y \mapsto x = y^d \bmod n$, où d est l'inverse de e modulo ppcm($p - 1$,$q - 1$). Ici d est également une valeur qui doit rester secrète.

La fonction f est donc conçue pour être facilement inversible lorsqu'on connaît la «trappe» d. Casser la fonction RSA consiste à trouver un moyen de calculer $f^{-1}(y)$ sans avoir besoin de l'exposant secret d. Plus exactement, on s'intéresse donc au problème suivant : étant donnés la clé publique (e,n) et un entier y compris entre 0 et $n - 1$, est-il possible de calculer de façon efficace la racine e-ième de y modulo n ?

Notons tout d'abord que connaître d revient à connaître la factorisation $n = p \times q$. Dans un sens, c'est évident : si on connaît p et q, il est facile de calculer d (qui est égal à l'inverse de e modulo ppcm($p - 1$,$q - 1$)) au moyen de l'algorithme d'Euclide. La réciproque est moins évidente : on peut montrer que la connaissance de d permet de factoriser efficacement n [13].

On se place ici dans la situation où d est a priori inconnu, et donc la factorisation de n aussi. La première façon d'attaquer la fonction RSA consiste à essayer de retrouver malgré tout cette factorisation. De nombreux algorithmes spécialisés ont été inventés pour trouver p et q à partir de n (voir par exemple [40]). Dans notre cas, la meilleure méthode connue s'appelle le Crible Algébrique Général (*General Number Field Sieve* – ou GNFS – en anglais), inventé par John Pollard et Arjen Lenstra [43, 41]. Le nombre de calculs élémentaires pour factoriser n est proportionnel à $L(n)$, où L est la fonction définie par

$$L(x) = \exp(1.9229(\ln x)^{1/3}(\ln \ln x)^{2/3}).$$

Sachant que cet algorithme a été réellement utilisé avec succès en 1999 [12] pour factoriser un nombre n de 512 bits avec 10^4 Mips.an [14], on peut en déduire qu'il faut de l'ordre de $10^4 \cdot L(n)/L(2^{512})$ Mips.an pour factoriser un entier n de taille quelconque. Cela permet de savoir la puissance nécessaire pour casser la fonction RSA par ce moyen, en fonction de la taille du modulo n (voir figure 3). Empiriquement, on fait généralement l'hypothèse (connue sous le nom de «Loi de Moore») selon laquelle la puissance des ordinateurs double tous les 18 mois. On peut alors prédire (s'il n'y a pas de découverte théorique nouvelle concernant les techniques de factorisation [15]) que les clés RSA de 1024 bits seront cassées vers l'an 2034, et celles de 2048 bits vers l'an 2079...

On peut se demander s'il est vraiment nécessaire de factoriser n (ou, ce qui est équivalent, de connaître d) pour pouvoir calculer la racine e-ième d'un entier y modulo n. Il s'agit en fait d'un problème ouvert. Certains indices (voir [10]) laissent penser que la réponse à la question est non,

13. Voir par exemple [47] chapitre 8, page 287.

14. Mips signifie «million d'instructions élémentaires par seconde». 1 Mips.an représente le nombre d'instructions élémentaires exécutées par une machine qui en exécute 1 million par seconde, et que l'on fait tourner pendant 1 an. C'est-à-dire 1 Mips.an $\simeq 31{,}5 \cdot 10^{12}$ instructions élémentaires.

15. Signalons à ce propos les pistes nouvelles récemment proposées dans [72, 44, 4, 45, 30, 73].

Nombre de Mips.ans disponibles	Taille maximale des clés RSA «factorisables»
$4{,}8 \times 10^{-2}$	251
$4{,}9 \times 10^{-1}$	292
4,9	337
$5{,}0 \times 10^{1}$	385
$5{,}0 \times 10^{2}$	438
$5{,}0 \times 10^{3}$	494
$\mathbf{1{,}0 \times 10^{4}}$	512
$5{,}1 \times 10^{4}$	555
$5{,}1 \times 10^{5}$	620
10^{8}	784
10^{11}	1035
10^{16}	1551
10^{20}	2057

FIG. 3 – *Puissance nécessaire pour factoriser le modulo RSA*

ce qui signifie que casser la fonction RSA serait moins difficile que le problème de la factorisation. Néanmoins on ne connaît à l'heure actuelle aucune autre stratégie générale que celle consistant à factoriser n, pour en déduire ensuite l'exposant secret d.

D'autres attaques sur la fonction RSA sont possibles lorsqu'on fait certaines hypothèses supplémentaires sur l'exposant secret d. Les deux attaques qui suivent, que nous ne décrirons pas en détail, font appel au célèbre algorithme LLL (dû à Arjen Lenstra, Hendrik Lenstra et László Lovász [42]) qui permet de trouver un vecteur de «petite» norme dans un réseau.

Un premier exemple de ce type se présente lorsqu'on suppose que la taille de cet exposant d est relativement petite (ce qui présente l'avantage pratique de rendre plus rapide le calcul de la fonction f^{-1}). Ainsi Michael Wiener a montré en 1990 [79] qu'il était facile de retrouver la valeur d à partir de e et n, lorsque d est inférieur à $n^{1/4}$. Cet algorithme s'appuie sur l'approximation des nombres rationnels par des fractions continues. Cette attaque a été améliorée en 1998 par Dan Boneh et Glen Durfee [8], qui ont étendu l'attaque de Wiener à tous les exposants d inférieurs à $n^{0.292}$ (au lieu de $n^{0.25}$). La conjecture généralement admise est qu'une attaque devrait être possible pour tous les d inférieurs à $n^{0.5}$, mais cela reste un problème ouvert pour l'instant.

Un deuxième exemple apparaît lorsqu'on suppose que certains bits de la clé d sont déjà connus de l'attaquant. Dan Boneh, Glen Durfee et Yair Frankel ont prouvé en 1998 [9] que si d a une taille de k bits, la connaissance des $k/4$ bits de poids le plus faible de d est suffisante pour reconstituer l'intégralité de d. Dans le même ordre d'idée, Phong Nguyen a montré qu'une attaque similaire est possible si on suppose que l'on connaît les $k/4$ bits de d situés entre les positions $k/4$ et $k/2$. En revanche, on ne sait pas ce que l'on peut dire dans les cas intermédiaires où $k/4$ bits consécutifs sont connus, mais situés ailleurs entre les positions 1 et $k/2$.

3.2 Les protocoles construits autour du RSA

Même si la fonction RSA est solide, nous allons voir que la façon dont on l'utilise pour obtenir un cryptosystème capable d'effectuer du chiffrement ou de la signature n'est pas neutre. De manière générale, il doit non seulement être impossible en pratique de retrouver un message clair

1001110101101111111111010010101110011010 1	$n = 676188542773$
0110110011111010011011000000000010010110 1 1 0000000000000000000000000000000000?????????? ⇓ ⇓ 0000000000000000000000000000000001001110 1 1 Attaque de M. Wiener, lorsqu'on sait que les 3/4 des bits de poids le plus fort sont nuls	$e = 467964265051$ 3/4 des bits de d sont connus $d = 315$
0000000000000000000000000000000000000000000011 ????????????????????????????????0011110011 ⇓ ⇓ ⇓ ⇓ 0110100011110101010000001101000011110011 Attaque de D. Boneh *et al.* lorsqu'on connaît le quart des bits de poids le plus faible	$e = 3$ 1/4 des bits de d sont connus $d = 450791264499$
0000000000000000000000000000000000000000000011 ????????????????????0000110100?????????? ⇓ ⇓ ⇓ ⇓ ⇓ 0110100011110101010000001101000011110011 Attaque de P. Nguyen, lorsqu'on connaît le quart suivant	$e = 3$ 1/4 des bits de d sont connus $d = 450791264499$

FIG. 4 – *Si l'on suppose que certains bits de la clé secrète d sont connus, par exemple le quart des bits correspondant à ceux de poids le plus faible, on peut facilement retrouver d.*

à partir de son chiffré (ou de fabriquer une fausse signature), hormis pour l'utilisateur légitime du système bien entendu, mais il doit être également impossible de connaître la moindre information sur le message clair (ou sur la signature). C'est ce qu'on appelle la *sécurité sémantique*.

Il est facile de voir que la fonction RSA de base ne vérifie pas cette propriété. Elle jouit en effet d'une propriété de multiplicativité : $f(x \cdot y) = f(x) \cdot f(y)$, ce qui ouvre la voie à certaines attaques.

Attaques sur le RSA en mode chiffrement

Illustrons sur un exemple, dû à Johan Håstad en 1988 [35], comment cette propriété peut donner une attaque pour le chiffrement. Supposons qu'un utilisateur, que nous appellerons Bob, veuille envoyer le même message M, sous forme chiffrée, à trois interlocuteurs différents, dont les clés publiques RSA sont respectivement $(n_1,\ e = 3)$, $(n_2,\ e = 3)$ et $(n_3,\ e = 3)$. S'il utilise la fonction RSA de façon élémentaire, Bob envoie donc les valeurs $C_1 = M^3 \bmod n_1$, $C_2 = M^3 \bmod n_2$ et $C_3 = M^3 \bmod n_3$. Si un espion, disons Charlie, intercepte les trois chiffrés C_1, C_2 et C_3, il peut alors retrouver facilement le message clair M, de la manière suivante. En utilisant le théorème des restes chinois (*Chinese Remainder Theorem* – ou CRT – en anglais), Charlie peut calculer la valeur C' telle que $0 \leq C' < n_1 n_2 n_3$, et $C' \equiv M^3 \bmod n_1 n_2 n_3$. Or par hypothèse on a $0 \leq M < n_1$, $0 \leq M < n_2$ et $0 \leq M < n_3$. Donc $0 \leq M < n_1 n_2 n_3$. Par conséquent $C' = M^3$, ou encore $M = (C')^{1/3}$ (il s'agit ici de la racine cubique ordinaire, la réduction modulo n n'intervient pas).

L'attaque d'Håstad est en fait encore plus puissante : elle montre que, même si Bob applique une transformation polynomiale (connue, mais différente d'un destinataire à l'autre) au message M avant de lui appliquer la fonction RSA, l'attaquant peut toujours retrouver le message M lorsque celui-ci est envoyé à un nombre suffisant de destinataires. Cela montre qu'il est dangereux de chiffrer le même message M par plusieurs clés publiques RSA distinctes.

Dans une autre direction, D. Coppersmith, M. Franklin, J. Patarin et M. Reiter ont montré en 1996 [14], qu'il est également risqué de chiffrer plusieurs messages «liés», et ce au moyen de la

même clé publique RSA. Plus précisément, si Bob envoie les chiffrés C_1 et C_2 de deux messages M_1 et M_2 liés par une relation de dépendance polynomiale $M_2 = f(M_1) \bmod n$, alors il est facile pour un attaquant de retrouver M_1 et M_2 à partir de C_1 et C_2. En effet, on sait que M_1 est une racine commune aux deux polynômes $X^3 - C_1$ et $(f(X))^3 - C_2$ modulo n. Or, en utilisant l'algorithme d'Euclide, on peut aisément calculer le PGCD de ces deux polynômes, qui avec une bonne probabilité sera égal à $X - M_1$ et permettra à l'attaquant de retrouver les deux messages clairs.

Il peut sembler artificiel d'avoir à chiffrer deux messages ainsi «liés». Toutefois, un scénario assez proche peut se produire dans des applications réelles. Supposons que pour chiffrer un message M de longueur $(k - m)$ bits, on lui ajoute m bits aléatoires (à la fin) avant de lui appliquer la fonction RSA, dont le modulo fait k bits. Il peut arriver que Bob ait à envoyer deux fois le chiffré du même message M (par exemple, s'il y a eu un problème de transmission sur la ligne). Mathématiquement, cela signifie donc que Bob envoie successivement $C_1 = f(M_1)$ et $C_2 = f(M_2)$, avec $M_1 = 2^m.M + r_1$ et $M_2 = 2^m.M + r_2$, où r_1 et r_2 sont deux valeurs aléatoires de m bits. On peut avoir l'impression que la présence de ces m bits aléatoires est une protection suffisante. Néanmoins, Don Coppersmith a montré en 1997 [13] que l'on pouvait améliorer l'attaque précédente, et ainsi retrouver M à partir de C_1 et C_2 à condition que m ne soit pas trop grand : plus précisément l'attaque marche si m est inférieur à k/e^2 (ainsi pour $e = 3$, l'attaque est possible si la partie aléatoire ajoutée au message a une longueur inférieure à $1/9$ de celle du modulo).

Don Coppersmith a également montré dans [13] que si on chiffre un message M en appliquant la fonction RSA de façon élémentaire : $C = M^e \bmod n$, alors il est possible de retrouver rapidement M à partir de C, dans le cas où M est inférieur à $n^{1/e}$. Plus généralement il a montré que pour une équation de la forme $f(x) = 0 \bmod n$, où f est un polynôme de degré e, il est possible de calculer, si elles existent, les racines x_0 telles que $|x_0| < n^{1/e}$. Ce résultat très puissant est à nouveau une conséquence de l'algorithme LLL sur les réseaux.

Toutes ces attaques montrent qu'il faut faire très attention à la manière dont on applique la fonction RSA pour chiffrer un message M. Dans la pratique, on commence par «formater» le message M au moyen d'une transformation φ qui fait intervenir un nombre aléatoire. Ainsi le message chiffré est obtenu par $C = f(\varphi(M,r))$, où r est une valeur aléatoire. Pour déchiffrer, on calcule $\varphi(M,r) = f^{-1}(C)$, et φ est conçue pour qu'il soit facile de retrouver M à partir de $\varphi(M,r)$. La théorie des protocoles de chiffrement RSA est aujourd'hui bien maîtrisée, au point qu'on connaît maintenant des transformations φ qui sont prouvées «saines». Cela signifie que, pour ces transformations φ, on peut montrer que la sécurité sémantique du schéma global de chiffrement $C = f(\varphi(M,r))$ est équivalente à la solidité de la fonction RSA de base (*i.e.* de la fonction f).

En d'autres termes, si la fonction f est mathématiquement solide, et qu'on utilise l'une de ces «bonnes» transformations φ, alors il n'existe pas d'attaque mathématique plus simple que celle consistant à trouver un moyen d'inverser la fonction f. Notons toutefois que ces résultats sont vrais sous l'hypothèse que les *fonctions de hachage* qui sont utilisées dans la définition de φ, sont elle-mêmes «parfaites» en un certain sens (c'est ce que l'on appelle le modèle de l'*oracle aléatoire*). Parmi ces protocoles de chiffrement «prouvés sûrs», citons OAEP (qui signifie *Optimal Asymmetric Encryption Padding*), proposé par Mihir Bellare et Philip Rogaway en 1994 [2, 28], et maintenant inclus dans la norme de chiffrement PKCS#1 v2.0 [68] mise au point par la société américaine RSA Data Security[16]. Des variantes plus récentes sont : OAEP+ (Victor Shoup, 2001

16. Notons que la version 1.5 de PKCS#1 présentait quelques défauts de sécurité, mis en évidence par D.

[74]), SAEP+ (Dan Boneh, 2001 [7]) et REACT (Tatsuaki Okamoto et David Pointcheval, 2001 [61]).

Attaques sur le RSA en mode signature

La sécurité sémantique du RSA lorsqu'on l'utilise pour signer des messages ne va pas non plus de soi.

Considérons par exemple le cas où la signature S d'un message M soit obtenue par $S = M^d \bmod n$, où d est l'exposant secret. Supposons maintenant que Charlie veuille faire signer à Bob un message M qu'il a choisi lui-même (par exemple M peut signifier «Je soussigné, Bob, donne 1000 euros à Charlie.»). Évidemment, si Charlie envoie ce message M tel quel, il sera démasqué immédiatement par Bob. Mais il peut agir de manière plus subtile : il envoie plutôt le message $M' = M.R^e \bmod n$ à Bob, où R est une valeur aléatoire qui sert à «camoufler» le vrai message M. Si Bob n'est pas attentif et renvoie la signature $S' = M'^d \bmod n$, Charlie peut alors en déduire $S = S'.R^{-1} \bmod n$, qui est la signature du vrai message M. Cette attaque par «camouflage» (signalée par Judy Moore en 1984 [21]) montre qu'il est dangereux de signer n'importe quoi (même si le message semble inoffensif...).

Pour empêcher un éventuel attaquant d'utiliser la propriété de multiplicativité de la fonction RSA, une méthode naturelle consiste à «compléter» le message M par une valeur fixe P (pour «padding» en anglais), avant de lui appliquer la fonction f^{-1}. Cela donne $S = (P||M)^d \bmod n$. Plus généralement, on peut utiliser $S = (\omega.M + a)^d \bmod n$, où ω est appelé la redondance multiplicative et a la redondance additive (le «padding» avec P correspond à $\omega = 1$ et $a = P.2^m$, où m est la taille du message). Néanmoins de nombreuses attaques sont encore possibles, comme nous allons le voir.

Dans le cas $\omega = 1$, Wiebren De Jonge et David Chaum ont montré en 1985 [20] qu'il est possible de fabriquer une signature d'un message M, si la taille du message est supérieure à $2k/3$ bits (ou k est la taille du modulo n).

Cette attaque a été améliorée par Marc Girault et Jean-François Misarsky en 1997 [32], grâce à une extension de l'algorithme d'Euclide. Leur attaque s'applique quelle que soient les valeurs de ω et a et permet de produire une signature d'un message M dès que la taille du message est supérieure à $k/2$ bits.

Grâce à l'algorithme LLL, Misarsky, toujours en 1997, a étendu [50] l'attaque au cas plus complexe où la signature est de la forme $S = (\omega_1.M + \omega_2.(M \bmod b) + a)^d \bmod n$, et même au cas où le message M et sa réduction $(M \bmod b)$ sont divisés en plusieurs morceaux avant d'appliquer la fonction RSA.

En 2001, Eric Brier, Christophe Clavier, Jean-Sébastien Coron et David Naccache ont encore étendu le domaine de cette attaque [11], en montrant que l'on peut produire un couple (M,S), où S est la signature de M, en supposant seulement que la taille du message est supérieure à $k/3$.

Toutes ces attaques montrent qu'introduire de la redondance dans le message avant d'utiliser la fonction RSA n'est probablement pas suffisant pour cacher la structure multiplicative du RSA. Il est ainsi conjecturé que les attaques précédentes s'appliquent pour des tailles de messages aussi petites que l'on veut...

Bleinchenbacher dans une attaque à chiffé choisi adaptative [5], puis par J.-S. Coron, M. Joye, D. Naccache et P. Paillier dans une attaque à clair choisi [16].

Pour résoudre le problème, la technique prônée depuis quelques années consiste à appliquer successivement au message : une *fonction de hachage*, puis une fonction de *redondance*, et enfin la fonction f^{-1} du RSA. Certains standards internationaux, comme ISO 9796-1 [37], ISO 9796-2 [38], ou PKCS#1 [68] ont proposé de telles méthodes. Néanmoins, ces protocoles n'étaient jusqu'à présent pas prouvés sémantiquement sûrs. Et de fait[17], Jean-Sébastien Coron, David Naccache, Julien Stern, Don Coppersmith, Shai Halevi et Charanjit Jutla ont montré en 1999 [17, 15] que l'on pouvait forger[18] une signature S d'un message M dans le cas de la norme ISO 9796-1.

Heureusement, comme pour le chiffrement, la théorie des protocoles de signature RSA est aujourd'hui bien comprise : on connaît désormais des transformations φ pour lesquelles on peut montrer que la sécurité sémantique du schéma global de signature $S = f^{-1}(\varphi(M))$ est équivalente à la solidité de la fonction RSA de base (*i.e.* de la fonction f). À nouveau, ces résultats sont vrais si on se place dans le modèle de l'oracle aléatoire. Deux protocoles de signature ainsi «prouvés sûrs» sont connus : FDH (*Full Domain Hash*) et PSS (*Probabilistic Signature Scheme*), tous deux proposés en 1996 par Mihir Bellare et Philip Rogaway [3]. PSS doit prochainement devenir la nouvelle norme de signature dans PKCS#1 v2.1 [69].

4 Signatures à base de logarithme discret

4.1 Notion de logarithme discret

On considère un groupe fini $\mathcal{G}$, dont la loi de composition interne est notée multiplicativement. On peut dans ce contexte définir le «problème du logarithme discret» :

Problème du Logarithme Discret. Étant donnés $g \in \mathcal{G}$ et y appartenant au sous-groupe engendré par g, trouver $x \in \mathbb{Z}$ tel que

$$y = g^x.$$

Le groupe $\mathcal{G}$ n'est pas nécessairement commutatif. Néanmoins, on se place en fait dans le sous-groupe cyclique engendré par l'élément g, ce qui nous ramène au cas commutatif. Dans l'état actuel des connaissances en théorie de la complexité, la difficulté du problème du logarithme discret dépend fortement du type de groupe sur lequel on se place. Par exemple, il est évident que le logarithme discret est très facile à calculer sur les groupes additifs $(\mathbb{Z}_n,+)$. En revanche, le problème du logarithme discret est considéré comme «difficile» en pratique pour le groupe multiplicatif d'un corps fini, ou encore pour le groupe additif d'une courbe elliptique sur un corps fini quelconque.

Plus précisément, aucun algorithme sous-exponentiel n'est connu à l'heure actuelle pour le calcul d'un logarithme discret sur une courbe elliptique bien choisie. Cela crée un «saut de complexité» entre le cas des corps finis et le cas des courbes elliptiques, qui a poussé Miller [49] et Koblitz [39] à proposer des schémas cryptographiques à base de courbes elliptiques. L'intérêt de ces cryptosystèmes est d'autoriser des tailles de clés nettement inférieures à celles des schémas s'appuyant sur la factorisation, cela pour un même niveau de sécurité conjecturé.

À partir du problème du logarithme discret, plusieurs schémas de signature ont été successivement proposés. Dans ce qui suit, nous décrivons les principaux, ainsi que la manière de les appliquer au cas particulier du groupe additif d'une courbe elliptique.

17. en étendant une stratégie proposée par Yvo Desmedt et Andrew Odlyzko [22, 51]
18. c'est-à-dire produire une signature valide du message, sans connaître la clé privée.

4.2 Schémas génériques de signature

L'algorithme de signature de Schnorr

Cet algorithme a été décrit par Claus Schnorr dans [70, 71]. Il utilise, comme données publiques, un groupe $\mathcal{G}$ et un élément $g \in \mathcal{G}$ d'ordre q. De son côté, pour signer des messages, le signataire possède une clé secrète $x \in \mathbb{Z}_q^*$, de laquelle est déduite la clé publique $y = g^{-x} \in \mathcal{G}$.
Si $m \in \mathbb{Z}$ est le haché du message à signer, la signature est calculée de la manière suivante :

1. On tire un élément k aléatoire dans $\mathbb{Z}_q$.
2. La signature est alors :
$$(g^k, k + mx) \in \mathcal{G} \times \mathbb{Z}_q.$$

Pour s'assurer de la validité de la signature $(z,\alpha) \in \mathcal{G} \times \mathbb{Z}_q$ d'un message de haché m, le destinataire vérifie que l'égalité
$$z = g^\alpha y^m$$
est satisfaite dans $\mathcal{G}$.

On peut montrer que la sécurité de l'algorithme de signature de Schnorr est équivalente au problème du logarithme discret.

L'algorithme de signature d'ElGamal

Cet algorithme a été décrit par Tahar ElGamal dans [24]. Il utilise, comme données publiques, un groupe $\mathcal{G}$ et un élément $g \in \mathcal{G}$ d'ordre n cette fois. Plus précisément, on suppose connus l'ordre n de g dans $\mathcal{G}$, et un plongement $\varphi : \mathcal{G} \to \mathbb{Z}$. Quant au signataire, il possède une clé secrète $x \in \mathbb{Z}_n$, à laquelle est associée la clé publique $y = g^x \in \mathcal{G}$.
Si $m \in \mathbb{Z}$ est le haché du message à signer, la signature est calculée de la manière suivante :

1. On tire un élément k aléatoire dans $\mathbb{Z}_n$.
2. On calcule $\gamma = g^k \in \mathcal{G}$.
3. On calcule k^{-1}, l'inverse de k modulo n.
4. La signature est alors :
$$(\gamma, (m - x\varphi(\gamma))k^{-1} \bmod n) \in \mathcal{G} \times \mathbb{Z}_n.$$

Pour s'assurer de la validité de la signature $(\gamma,\delta) \in \mathcal{G} \times \mathbb{Z}_n$ d'un message de haché m, le destinataire vérifie que l'égalité
$$y^{\varphi(\gamma)}\gamma^\delta = g^m$$
est satisfaite dans $\mathcal{G}$.

Dans [62], David Pointcheval a défini une variante de ce schéma, dont la sécurité est prouvée équivalente au problème du logarithme discret, ceci dans le modèle de l'oracle aléatoire.

Signature DSA

Il s'agit d'une variante de la signature ElGamal, dont on peut trouver une description dans [56, 57]. Le schéma utilise toujours un groupe fini $\mathcal{G}$ supposé connu, ainsi qu'un élément $g \in \mathcal{G}$, dont l'ordre n est également public. De son côté, le signataire possède un exposant secret $x \in \mathbb{Z}$, auquel est associé la clé publique $y = g^x \in \mathcal{G}$. Comme précédemment, un plongement $\varphi : \mathcal{G} \to \mathbb{Z}$ est supposé connu[19].

19. Notons que, dans la signature DSA proprement dite, l'application φ choisie est la réduction modulo q, où $\mathcal{G}$ est $\mathbb{Z}_p$ et q un facteur de 160 bits de $p-1$. Elle n'est donc pas théoriquement injective, mais en pratique on peut la supposer comme telle.

Si $m \in \mathbb{Z}$ est le haché du message à signer. La signature est calculée de la manière suivante :

1. On choisit un élément k aléatoire dans $\mathbb{Z}_n$.
2. On calcule $\gamma = \varphi(g^k) \in \mathbb{Z}$.
3. On calcule k^{-1}, l'inverse de k modulo n.
4. La signature est alors :
$$(\gamma,(m + x\gamma)k^{-1} \bmod n) \in \mathbb{Z} \times \mathbb{Z}_n.$$

Pour s'assurer de la validité de la signature $(\gamma,\delta) \in \mathbb{Z} \times \mathbb{Z}_n$ d'un message de haché m, le destinataire vérifie que l'égalité
$$\varphi(g^{e_1} y^{e_2}) = \gamma$$
est satisfaite dans $\mathbb{Z}$, avec $e_1 = m\delta^{-1} \bmod n$ et $e_2 = \gamma\delta^{-1} \bmod n$.

Dans [64], David Pointcheval et Serge Vaudenay ont proposé une variante de DSA dont la sécurité équivalente au problème du logarithme discret, dans le modèle de l'oracle aléatoire.

4.3 Transposition sur les courbes elliptiques

Intéressons nous maintenant au cas où $\mathcal{G}$ est le groupe additif d'une courbe elliptique E sur le corps fini $\mathbb{F}_q$ de cardinal q (une puissance de nombre premier).

Lorsque la caractéristique du corps fini est égale 2 (ce qui est équivalent à dire que q est une puissance de 2), on peut décrire la courbe elliptique E comme l'ensemble des points $(x,y) \in \mathbb{F}_q^2$ vérifiant l'équation
$$y^2 + xy = x^3 + ax^2 + b$$
(où b est un élément non nul de $\mathbb{F}_q$), auquel on ajoute un «point à l'infini» noté $\mathcal{O}$.

Lorsque la caractéristique du corps fini est au moins 3, la courbe elliptique E se compose du point $\mathcal{O}$ à l'infini et des points $(x,y) \in \mathbb{F}_q^2$ vérifiant l'équation
$$y^2 = x^3 + ax + b,$$
(où a et b sont deux éléments de $\mathbb{F}_q$ tels que $4a^3 + 27b^2 \neq 0$).

Dans le cadre des courbes elliptiques, le logarithme discret a pour base un point G de la courbe E, dont l'ordre est un nombre premier r. Par aillers on choisit, conformément au paragraphe 4.2, un plongement de la courbe elliptique E dans les entiers[20].

Pour garantir la sécurité des schémas à base de logarithme discret sur les courbes elliptiques, il faut que le corps fini $\mathbb{F}_q$ et r soient suffisamment grands par rapport aux attaques connues. Pour une courbe elliptique convenablement choisie, les meilleurs algorithmes de logarithme discret sont du type «pas de géant-pas de bébé» (en anglais «baby step-giant step»), dont la figure 5 donne une estimation des temps de calcul.

La signature ECDSA

Il s'agit de l'algorithme de signature DSA transposé sur les courbes elliptiques. Dans ce schéma, proposé par Scott Vanstone en 1992 [114], et inclus dans la norme IEEE P1363, on considère une courbe elliptique E définie par le quintuplet (q,a,b,G,r). Le signataire possède une clé secrète s, à laquelle est associée la clé publique $W = sG$.

Si le haché du message à signer est un élément $f \in \mathbb{Z}_r$, la signature est le couple $(c,d) \in \mathbb{Z}_r^2$ obtenu par le procédé suivant :

1. On engendre un nombre aléatoire $u \in \mathbb{Z}_r$, et on calcule $V = uG$ dans E.

20. Par exemple, dans la norme IEEE P1363 [36], ce plongement est réalisé en prenant l'abscisse du point de la courbe, et en la plongeant dans les entiers au moyen de la représentation du corps fini adoptée.

Nombre de bits de r	Temps en Mips.ans
128	4×10^5
172	3×10^{12}
234	3×10^{21}
314	2×10^{33}

FIG. 5 – *Puissance nécessaire pour résoudre le logarithme discret*

2. On convertit l'abscisse de V en un entier i.
3. On calcule $c = i \bmod r$. Si $c \equiv 0$, on revient à la première étape.
4. On calcule $d = u^{-1}(f + sc) \bmod r$. Si $d \equiv 0$, on revient à la première étape.

Pour vérifier la validité d'une signature (c,d), on applique les étapes suivantes :

1. Si c ou d ne sont pas dans $[1,r-1]$, on rejette la signature.
2. On calcule $h = d^{-1} \bmod r$, $h_1 = fh \bmod r$ et $h_2 = ch \bmod r$.
3. On calcule le point $P = h_1 G + h_2 W$. Si P est nul, on rejette la signature.
4. On convertit l'abscisse de P en un entier i.
5. On calcule $c' = i \bmod r$.
6. Si $c' \equiv c$, on accepte la signature.

La signature ECNR de Nyberg et Rueppel

Cet algorithme signature fait également partie de la norme IEEE P1363 [36]. Il s'agit d'une variante de la signature ElGamal, proposée par Kaisa Nyberg et Rainer Rueppel en 1993 [59, 60]. Le schéma ENCR permet de recouvrer le message original. Soit E une courbe définie par le quintuplet (q,a,b,G,r). Le signataire possède une clé secrète s, à laquelle est associée la clé publique $W = sG$.

Si le message à signer est un entier f tel que $0 \leq f < r$, la signature est le couple $(c,d) \in \mathbb{Z}_r^2$ obtenu par le procédé suivant :

1. On engendre un nombre aléatoire $u \in \mathbb{Z}_r$, et l'on calcule $V = uG$ dans E.
2. On convertit l'abscisse de V en un entier i.
3. On calcule $c = i + f \bmod r$. Si $c \equiv 0$, on revient à la première étape.
4. On calcule $d = u - sc \bmod r$.

Le processus qui suit permet à la fois de vérifier la validité de la signature (c,d) et de recouvrer le message f lui-même :

1. Si $c \notin [1.r-1]$ ou si $d \notin [0,r-1]$, on rejette la signature.
2. On calcule le point $P = dG + cW$. Si P est nul, on rejette la signature.
3. On convertit l'abscisse de P en un entier i.
4. On renvoie $f = c - i \bmod r$.

Chapitre 2

La carte à microprocesseur

Chaque fois qu'un Gonda désirait quelque chose de nouveau, des vêtements, un voyage, des objets, il payait avec sa clé. Il pliait le majeur, enfonçait sa clé dans un emplacement prévu à cet effet et son compte, à l'ordinateur central, était aussitôt diminué de la valeur de la marchandise ou du service demandés.

René Barjavel, *La nuit des temps* (1967)

This application will put a sophisticated information-security device in the wallet or purse of practically every person in the industrialized world, and will therefore be the most extensive application ever made of cryptographic schemes.

Gustavus Simmons (1992)

Sommaire

1 L'invention de la carte à puce 27
2 Fonctionnement d'une carte à microprocesseur 28
2.1 Description physique 28
2.2 Communication avec la carte 29
2.3 Format logique des commandes 30
3 Performances pour la signature électronique 30
3.1 Multiplications et exponentiations modulaires 31
3.2 Temps de calcul sur une carte à microprocesseur 31

1 L'invention de la carte à puce

En 1967, dans *La nuit des temps*, titre d'un roman de science-fiction du romancier français René Barjavel, on trouve l'histoire d'un peuple mythique, les Gondas - une civilisation vieille de milliers d'années mais très avancée - qui utilise un anneau magique ayant des moyens de mémorisation et de communication. On peut y voir l'origine (au moins littéraire) de la carte à puce.

L'idée d'utiliser un composant électronique contenu dans une carte de crédit apparaît presque au même moment, et de nombreux brevets commencent à être publiés. Aux États-Unis, Pomeroy (1967), puis Jules Ellingboe (1970), qui décrit concrètement un moyen de paiement électronique sur une carte de crédit à contacts, et John Halpern (1972), avec son stylo électronique sécurisé

de paiement. Au Japon, Kunitaka Arimura (1970), qui propose une méthode d'authentification dynamique réalisée à l'aide d'un dispositif d'identification. En Allemagne, Jürgen Dethloff (1977). Et en France, Roland Moreno (1974), Michel Ugon (1977) et Louis Guillou (1979), ainsi que de nombreux autres.

La carte de Roland Moreno était une simple *carte à mémoire*, dite aussi *carte à logique câblée* et n'était pas programmable. Dès 1977, Michel Ugon – à qui on avait confié l'étude du problème chez CII-Honeywell Bull – se rend compte que seule la présence d'un microprocesseur peut donner à la carte suffisamment de fonctionnalités, notamment pour assurer la sécurité au moyen d'algorithmes cryptographiques. Il devient alors clair que la carte doit être intelligente.

C'est ainsi que prend naissance la *carte à microprocesseur*, dite aussi *carte à microcalculateur*, dont le premier exemplaire est baptisé CP8. C'est à l'époque une carte bi-puces, qui présente de ce fait des faiblesses sécuritaires évidentes, car un attaquant peut connaître le contenu des informations qui transitent entre la mémoire et le microprocesseur. Pour cette raison, en 1981, en collaboration avec Motorola, voit le jour le premier calculateur *monolithique* pour la carte à puce, appelé SPOM (*Self Programmable One-chip Microprocessor*).

En 1984, la carte à puce est choisie par le GIE Cartes Bancaires comme élément principal de la sécurité du réseau de paiement électronique français (plutôt que la logique câblée) qui se déploie à partir de 1986. En 1992, Gustavus Simmons prophétise dans [112] :

> «Cette application [la carte à microprocesseur] mettra un dispositif de sécurité de l'information dans le portefeuille ou le porte-monnaie de pratiquement tout le monde dans le monde industrialisé, et constituera de ce fait l'application la plus étendue jamais mise en œuvre pour les schémas cryptographiques.»

Et de fait, à partir de cette date, la carte à puce se généralise à de nombreux autres domaines: la télévision à péage, le porte-monnaie électronique, la téléphonie mobile, etc[1].

2 Fonctionnement d'une carte à microprocesseur

2.1 Description physique

Une carte à microprocesseur[2] est constituée d'un *micro-module* (appelé aussi *puce*) inséré dans rectangle de plastique au format «carte de visite» sur lequel sont en général inscrites des informations liées à l'identité du possesseur de la carte. Par ailleurs, la carte peut aussi comporter une piste magnétique (c'est le cas notamment des cartes bancaires actuelles). Conformément au standard ISO/IEC 7816-1 [95], les dimensions de la carte à microprocesseur doivent vérifier :

- $85.47 \text{ mm} \leq \text{Largeur} \leq 85.72 \text{ mm}$;
- $53.92 \text{ mm} \leq \text{Hauteur} \leq 54.03 \text{ mm}$;
- $\text{Épaisseur} = 0.76 \pm 0.08 \text{ mm}$

Comme spécifié dans le standard ISO/IEC 7816-2 [96], le micro-module comporte huit contacts, dont six sont effectivement connectés à la puce elle-même (qui est en général invisible). Les contacts sont utilisés pour l'alimentation (V_{cc} et V_{pp})[3], la masse (GND), l'horloge (CLK), le

1. Pour plus de détails sur les débuts de la carte à microprocesseur, on peut consulter [89, 90, 91, 113].

2. Ce paragraphe s'appuie en partie sur des données disponibles dans [101, 107, 106].

3. V_{cc} correspond à la tension d'alimentation en lecture, alors que V_{pp} correspond à la tension à appliquer, sur demande de la carte, pour programmer la mémoire de données (tension en écriture).

signal Reset (RST)[4] et le contact I/O d'entrée sortie par lequel transitent en mode *half-duplex* toutes les données échangées entre la carte et le monde extérieur.

Le plus souvent, la carte possède un processeur 8 bits, mais il existe des modèles à base de processeurs 16 et 32 bits. On dispose même de processeurs à architecture RISC [86]. Dans la plupart des cartes actuelles, le microprocesseur est construit autour d'un cœur Motorola 6805 ou Intel 8051, avec une horloge à 5 MHz. Même s'il est toujours possible de les augmenter, les fréquences d'horloge sont limitées par deux facteurs : la consommation du processeur, qui ne doit pas dépassser une certaine limite, et la conformité au standard ISO/IEC 7816-3 [97], qui oblige à respecter une certaine plage de fréquence, pour rester compatible avec les matériels déjà en place (notamment les lecteurs).

En revanche, dans les cartes à microprocesseur récentes, un système de multiplicateur de fréquence permet d'obtenir une horloge interne fonctionnant jusqu'à à 40 MHz, ce qui permet d'effectuer des calculs cryptographiques beaucoup plus rapidement. Par ailleurs, la carte contient souvent un *coprocesseur* cryptographique (ou *crypto-processeur*), généralement associé à un *générateur de nombres aléatoires* (RNG, *Random Number Generator*).

La carte à microprocesseur comporte en général trois types de mémoire :

- La ROM (*Read Only Memory*), qui n'a pas besoin d'être alimentée pour conserver l'information qu'elle contient. On l'utilise pour stocker le système d'exploitation de la carte, ainsi que les données permanentes. Ces éléments sont inscrits dans la carte dans la phase dite de «masquage», et ne peuvent plus être modifiés ensuite.
- L'E^{2}PROM (ou EEPROM, *Electrical Erasable Programmable Read Only Memory*) qui peut, comme la ROM, préserver son information même quand la carte n'est plus sous tension. La différence avec la ROM est qu'elle peut être modifiée par une application. Un problème de cette mémoire est sa durée de vie limitée en nombre de cycles d'écriture (typiquement de l'ordre de 100000 cycles) et en temps (10 ans), ce qui nécessite de changer régulièrement la carte pour éviter des dysfonctionnements. Un autre inconvénient est sa lenteur d'accès : l'E^{2}PROM est en effet en moyenne 1000 fois plus lente en écriture que la RAM.
- La RAM (*Random Access Memory*), qui est utilisée comme espace de stockage temporaire grâce à la rapidité des temps d'accès. Elle possède un caractère non persistent : dès que la carte n'est plus sous tension, la RAM perd son contenu. En revanche, elle peut être lue et écrite indéfiniment.

2.2 Communication avec la carte

La carte à microprocesseur contient un port de communication série (*via* une liaison asynchrone), pour échanger les données et les informations de contrôle avec le monde extérieur. La vitesse de transmission est généralement de 9600 bits par seconde, mais le standard ISO/IEC 7816-3 autorise jusqu'à 115200 bits par seconde.

Le protocole suivant lequel la carte communique avec l'extérieur est également spécifié par le standard ISO/IEC 7816-3, qui définit deux possibilités : la première (appelée $T = 0$) prend l'octet comme structure élémentaire, alors que la seconde (dite $T = 1$) s'appuie sur le bit comme élément d'information[5].

4. Une tension appliquée sur ce contact déclenche l'initialisation physique et logique du composant.

5. Remarquons que le standard prévoit en fait jusqu'à 14 protocoles, $T = 14$ correspondant à un protocole de communication propriétaire.

La tension électrique, le traitement des erreurs et la fréquence d'horloge imposent l'utilisation d'un dispositif matériel pour dialoguer avec la carte. Ce dispositif, appelé CAD (*Card Acceptance Device*), est l'équivalent d'un UART (*Universal Asynchronous Receiver/Transmitter*), avec des fonctionnalités plus sophistiquées. Il comporte typiquement :

- une interface mécanique : le *connecteur* ;
- une interface électronique : le *coupleur* ;
- un boîtier contenant ces deux éléments : le *lecteur de carte*.

Les lecteurs les plus simples sont comparables à des modems, et gèrent le protocole de communication de façon élémentaire, sans interagir avec le système d'exploitation de la carte. Ils peuvent en principe fonctionner avec n'importe quelle carte à microprocesseur compatible avec les standards ISO/IEC 7816.

Il existe des lecteurs plus complexes, qui peuvent être en partie reprogrammés et contenir des données (comme des clés), des fichiers et des programmes. Ils peuvent exécuter des algorithmes cryptographiques, disposer de clavier, d'écran, d'un langage de programmation spécifique. Ces lecteurs ne sont en général plus universels : ils sont dédiés à certaines cartes.

2.3 Format logique des commandes

Pour fonctionner avec une carte à microprocesseur, le lecteur doit pouvoir exécuter les fonctions suivantes :

- mettre la carte sous tension, ou hors tension ;
- initialiser (physiquement et logiquement) la carte ;
- lire des données de la carte (commande *get*) ;
- écrire des données dans la carte (commande *put*).

Chaque commande *get* et *put* contient un en-tête, qui indique à la carte comment traiter les données présentes dans le reste de la commande. Plus précisément, l'en-tête est constitué de cinq octets appelés CLA, INS, P1, P2 et LEN, qui contiennent respectivement la classe, l'instruction, le premier paramètre, un second paramètre, et la longueur des données à traiter. La carte renvoie un octet d'accusé de réception au début de la commande, ainsi que deux octets d'état SW1 et SW2 à la fin de la commande.

Le standard ISO/IEC 7816-4 [98] vise à assurer une inter-opérabilité. Il spécifie le contenu des messages entre la carte et le lecteur. Pour cela, il utilise le protocole APDU (*Application Protocol Data Units*) pour les commandes et les réponses. Le standard définit également les structures des fichiers et des données :

- l'accès à ces données ;
- l'architecture de sécurité ;
- la sécurisation des communications.

3 Performances pour la signature électronique

Pour les implémentations sur PC, une étude approfondie des performances des algorithmes de signature a eté menée par le projet européen NESSIE [55]. Nous évoquons ici le cas des cartes à microprocesseur.

3.1 Multiplications et exponentiations modulaires

La multiplication et l'exponentiation modulaire sont à la base de nombreux algorithmes à clé publique, et leur temps de calcul est crucial pour les schémas tels que RSA, Schnorr, ElGamal, DSA, ainsi que ECDSA ou ECNR si le corps fini choisi est $\mathbb{F}_p$ (avec p premier). De nombreuses méthodes ont été proposées pour effectuer ces calculs, notamment par Montgomery [100], Barrett [80], Sedlak [110], De Waleffe et Quisquater [85], ou encore Dhem et Quisquater [87].

Les performances de ces méthodes sont comparables, comme le montrent les chiffres donnés par David Naccache et David M'Raïhi dans [101] ou par Helena Handschuh et Pascal Paillier dans [92, 93]. En revanche, le choix de l'une ou l'autre méthode a un impact important sur l'architecture interne d'un éventuel coprocesseur cryptographique (voir [103], ou bien [99] pages 248 à 250).

3.2 Temps de calcul sur une carte à microprocesseur

Les figures 1 et 2 donnent une évaluation des temps de calcul pour générer et vérifier une signature sur une carte à microprocesseur. Les algorithmes considérés sont RSA [67], Rabin-Williams [105, 115], Fiat-Shamir [26] (dans sa version signature), GQ [34] et GQ2 [66] (dans leur version signature), DSA [56, 57], ElGamal [24], ACE-Sign [109], ESIGN [88], ECDSA [114, 36], QUARTZ [83, 84], SFLASH [81, 82], NTRUSign [94], SP (*Small Primes*) [102], PKP (*Permuted Kernel Problem*) [111] et IP (*Isomorphisms of Polynomials*) [104].

La fréquence d'horloge (interne) est 10 MHz. Dans la figure 1, on considère une carte «bas de gamme» sans coprocesseur cryptographique. La figure 2 montre l'impact de la présence d'un coprocesseur sur les performances de RSA et Rabin-Williams 1024 bits, ainsi que de l'algorithme ECDSA lorsque le corps fini est $\mathbb{F}_p$, où p est un nombre premier de 163 bits.

Algorithme de signature	Taille de la signature (en octets)	Taille de la clé publique (en octets)	Temps de signature (en ms)	Temps de vérification (en ms)
RSA 1 024 bits	128	128	73 200	285
Rabin-Williams 1 024 bits	128	128	73 200	143
Fiat-Shamir	500	512	71	71
GQ	64	128	7 140	3 570
GQ2	64	128	7 140	3 570
DSA	40	128	8 660	17 300
ElGamal	128	128	26 700	53 500
ACE-Sign	de 425 à 705	620	$\simeq$ 37 100	$\simeq$ 43 700
ESIGN	144	145	$\simeq$ 3 260	$\simeq$ 384
ECDSA	48 (peut être réduit à 41)	48	825	1 610
QUARTZ	16	71 000	$>$ 1 min	$\simeq$ 10
SFLASH	33	15 400	59	$\simeq$ 15
NTRUSign - 251	220	128	256	288
SP	5 000	1 000	866	17 300
PKP	5 000	128	1 430	714
IP	250	256	355	71

FIG. 1 – *Temps de calcul pour une carte à microprocesseur à 10 MHz (sans coprocesseur)*

Algorithme de signature	Taille de la signature (en octets)	Taille de la clé publique (en octets)	Temps de signature (en ms)	Temps de vérification (en ms)
RSA 1 024 bits	128	128	119	7
Rabin-Williams 1 024 bits	128	128	119	4
ECDSA	48 (peut être réduit à 41)	48	51	99

FIG. 2 – *Temps de calcul pour une carte à microprocesseur à 10 MHz (avec coprocesseur)*

Chapitre 3

Hypothèses calculatoires

Almost all of modern cryptography rises or fall with the question of whether functions exist [...] which are easy to evaluate but hard (on the average) to invert.

Oded Goldreich (1997)

Tous les mathématiciens savent que le passage de une à plusieurs variables est un «saut» brusque, qui s'accompagne de grandes difficultés et nécessite des méthodes toutes nouvelles.

Jean Dieudonné (1980)

Sommaire

1 Introduction 33
2 Systèmes d'équations quadratiques sur un corps fini 34
2.1 Dans le corps $K = \mathbb{F}_2$ 34
2.2 Dans un corps quelconque 35
2.3 Systèmes de n équations quadratiques en $k \geq n$ variables 36
3 Isomorphismes de polynômes 37
3.1 Les problèmes IP et MP 37
3.2 IP à un secret est au moins aussi difficile que les Isomorphismes de Graphes 39
3.3 MP est NP-difficile 41
3.4 Le problème de décision IP n'est pas NP-complet 43
4 Le problème MinRank 44
4.1 Définition du problème MinRank 44
4.2 Complexité de MinRank 45
5 Le problème de la décomposition fonctionnelle 45
5.1 Motivation du problème 45
5.2 L'algorithme de Dickerson 46
5.3 Question de la NP-difficulté 46

1 Introduction

La *théorie de la complexité* est un des domaines pour lesquels la maxime de Jean Dieudonné est sûrement vraie. Les résultats de complexité peuvent en effet différer complètement lorsqu'on

passe d'une à plusieurs variables. Dans ce chapitre sont exposées plusieurs hypothèses calculatoires que j'ai étudiées dans [168, 141, 142, 160]. Ce sont les briques de base, à partir desquelles on peut construire de nombreux nouveaux cryptosystèmes aux propriétés inédites, notamment pour tenir compte des contraintes de mémoire ou de temps propres aux cartes à puce.

Tout d'abord, résoudre une équation polynomiale monovariable de petit degré est faisable (la complexité est polynomiale en d), mais résoudre un système d'équations polynomiales à plusieurs variable de degré d sur un corps fini K est NP-difficile, même si $K = \mathbb{F}_2$ et $d = 2$. La décomposition fonctionnelle fournit un deuxième exemple : si elle est souvent facile pour un polynôme monovariable (voir [149, 150]), la complexité de la décomposition fonctionnelle pour les polynômes multivariables semble devenir exponentielle en le nombre de variables, même avec les meilleurs algorithmes. En outre, le problème général de la décomposition des polynômes multivariables est NP-difficile.

2 Systèmes d'équations quadratiques sur un corps fini

Le premier problème que nous considérons est celui qui consiste à essayer de résoudre le système suivant d'équations quadratiques :

$$\begin{cases} P_1(x_1,...,x_n) = y_1 \\ ... \\ P_m(x_1,...,x_n) = y_n \end{cases}$$

où $y = (y_1,...,y_n)$ est connu et où $x = (x_1,...,x_n)$ est l'inconnue. Les polynômes P_1, ..., P_n sont à coefficients dans un corps fini K.

Dans [141], j'ai prouvé que – quel que soit le corps K – le problème *général* de la résolution d'un système aléatoire d'équations multivariables quadratiques sur K est NP-complet. Notons que ce résultat était déjà connu pour $K = \mathbb{F}_2$ (voir [131] page 251).

2.1 Dans le corps $K = \mathbb{F}_2$

Considérons une instance du problème de 3-Satisfaisabilité (également appelé 3-SAT), donné par un ensemble fini $U = \{u_1,...,u_n\}$ de variables booléennes, et une collection $C = \{c_1,...,c_m\}$ de *clauses* sur U. Par définition, chaque clause est la *disjonction* d'au plus trois *littéraux* sur U (un littéral est de la forme u ou $\bar{u}$, avec $u \in U$).

Chaque variable booléenne peut être considérée, de façon évidente, comme un élément de $\mathbb{F}_2$. De plus[1] :

- Si $u \in U$ correspond à $x \in \mathbb{F}_2$, alors $\bar{u}$ correspond à $1 - x$;
- Si $u \in U$ (resp. $v \in U$) correspond à $x \in \mathbb{F}_2$ (resp. $y \in \mathbb{F}_2$), alors $(u \vee v)$ correspond à $(xy + x + y)$ dans $\mathbb{F}_2$.

Par conséquent, trouver un choix de U qui satisfasse toutes les clauses de C est équivalent à résoudre un système de m équations *cubiques* en n indéterminées sur $\mathbb{F}_2$. De plus, chaque équation de ce système contient au plus trois des indéterminées x_i. Si on ajoute les $\frac{n(n-1)}{2}$ nouvelles variables $z_{ij} = x_i x_j$ $(i < j)$, le système peut être réécrit comme un système de $m + \frac{n(n-1)}{2}$ équations *quadratiques* à $\frac{n(n+1)}{2}$ variables sur $\mathbb{F}_2$.

1. en notant $\vee$ l'opérateur booléen «OU»

En conclusion, on peut réduire, en temps polynomial, le problème consistant à résoudre une instance choisie aléatoirement du problème 3-SAT – qui est NP-complet – au problème de la résolution d'un système aléatoire d'équations multivariables quadratiques sur $\mathbb{F}_2$. Ce dernier problème est donc également NP-complet.

2.2 Dans un corps quelconque

Soit K un corps *quelconque*, et soit $\mathcal{S}$ le système suivant de n équations quadratiques en n indéterminées sur $\mathbb{F}_2$:

$$\sum_{1\leq i<j\leq n} \mu_{ijk} x_i x_j + \sum_{i=1}^{n} \nu_{ik} x_i = y_k \qquad (1 \leq k \leq n),$$

où les y_k sont des éléments fixés de $\mathbb{F}_2$. Le problème de la résolution d'un tel système aléatoire $\mathcal{S}$ peut se réduire – en temps polynomial – au problème de la résolution d'un système aléatoire d'équations quadratiques sur le corps K.

Transformation du système

Sur $\mathbb{F}_2$, résoudre $(\mathcal{S})$ est équivalent à résoudre le système suivant :

$$(\mathcal{S}')\begin{cases} \mu_{12k}x_1x_2 = z_{12k} & (1\leq k\leq n)\\ \mu_{13k}x_1x_3 = z_{12k} + z_{13k} & (1\leq k\leq n)\\ \quad \cdots\cdots\cdots\cdots & \\ \mu_{(n-1)nk}x_{n-1}x_n = z_{(n-2)nk} + z_{(n-1)nk} & (1\leq k\leq n)\\ \nu_{1k}x_1 = z_{(n-1)nk} + w_{1k} & (1\leq k\leq n)\\ \nu_{2k}x_2 = w_{1k} + w_{2k} & (1\leq k\leq n)\\ \quad \cdots\cdots\cdots\cdots & \\ \nu_{(n-1)k}x_{n-1} = w_{(n-2)k} + w_{(n-1)k} & (1\leq k\leq n)\\ \nu_{nk}x_n = w_{(n-1)k} + y_k & (1\leq k\leq n) \end{cases}$$

On obtient donc un système de $\frac{n^2(n+1)}{2}$ équations sur $\mathbb{F}_2$, avec $\frac{n^2(n+1)}{2}$ indéterminées : les x_i $(1 \leq i \leq n)$, les z_{ijk} $(1 \leq i < j \leq n,\ 1 \leq k \leq n)$, et les w_{ik} $(1 \leq i \leq n-1,\ 1 \leq k \leq n)$.

Transcription de $\mathbb{F}_2$ à K

Chaque élément $x \in \{0,1\}$ de $\mathbb{F}_2$ peut être considéré comme un élément x de K (où 0 désigne l'élément neutre de l'addition dans K, et 1 désigne l'élément neutre de la multiplication dans K) tel que $x(x-1) = 0$. La loi de multiplication $(x,y) \mapsto xy$ sur $\mathbb{F}_2$ peut se transposer en $(x,y) \mapsto xy$ sur K, alors qu'on peut traduire la loi d'addition $(x,y) \mapsto x+y$ sur $\mathbb{F}_2$ en $(x,y) \mapsto (x+y)(2-(x+y))$ sur K (ici 2 désigne l'élément $1+1$ de K).

Par conséquent, résoudre $(\mathcal{S}')$ sur $\mathbb{F}_2$ est équivalent à résoudre le système suivant sur K :

$$(\mathcal{S}'')\begin{cases}
\mu_{12k}x_1x_2 = z_{12k} & (1\leq k\leq n)\\
\mu_{13k}x_1x_3 = (z_{12k}+z_{13k})(2-z_{12k}-z_{13k}) & (1\leq k\leq n)\\
\quad \ldots\ldots\ldots\ldots & \\
\mu_{(n-1)nk}x_{n-1}x_n = (z_{(n-2)nk}+z_{(n-1)nk})(2-z_{(n-2)nk}-z_{(n-1)nk}) & (1\leq k\leq n)\\
\nu_{1k}x_1 = (z_{(n-1)nk}+w_{1k})(2-z_{(n-1)nk}-w_{1k}) & (1\leq k\leq n)\\
\nu_{2k}x_2 = (w_{1k}+w_{2k})(2-w_{1k}-w_{2k}) & (1\leq k\leq n)\\
\quad \ldots\ldots\ldots\ldots & \\
\nu_{(n-1)k}x_{n-1} = (w_{(n-2)k}+w_{(n-1)k})(2-w_{(n-2)k}-w_{(n-1)k}) & (1\leq k\leq n)\\
\nu_{nk}x_n = (w_{(n-1)k}+y_k)(2-w_{(n-1)k}-y_k) & (1\leq k\leq n)\\
x_i(x_i-1)=0 & (1\leq i\leq n)\\
z_{ijk}(z_{ijk}-1)=0 & (1\leq i<j\leq n,\ 1\leq k\leq n)\\
w_{ik}(w_{ik}-1)=0 & (1\leq i\leq n-1,\ 1\leq k\leq n)
\end{cases}$$

Il s'agit d'un système de $n^2(n+1)$ équations quadratiques en $\frac{n^2(n+1)}{2}$ indéterminées sur K.

Si on résume, le problème de résoudre un système aléatoire d'équations multivariables quadratiques sur $\mathbb{F}_2$ – qui est NP-complet – peut se réduire, en temps polynomial, au problème de résoudre un système aléatoire d'équations multivariables quadratiques sur K. Ce dernier problème est donc aussi NP-complet. Remarquons que ce qui précède est encore valable si le corps K n'est pas commutatif, car le raisonnement n'utilise pas la commutativité de la loi de multiplication sur K.

2.3 Systèmes de n équations quadratiques en $k \geq n$ variables

Dans [137], nous montrons que le problème de la résolution des systèmes de n équations quadratiques en $k \geq n$ variables est NP-difficile en pire cas.

Pour s'en convaincre, supposons que l'on dispose d'un oracle (*boîte noire*), qui prend en entrée un ensemble quelconque de n équations quadratiques en k variables, et donne en sortie une solution quand il en existe au moins une. Alors on peut utiliser cet oracle pour trouver une solution à n'importe quel système de n équations quadratiques en n variables (problème qui est NP-complet).

On peut procéder par exemple comme ceci. Soit $(\mathcal{A})$ un ensemble de $(n-1)$ équations quadratiques en $(n-1)$ variables x_1, x_2, ..., x_{n-1}. On définit alors α variables supplémentaires y_1, ..., y_α. Soit $(\mathcal{B})$ l'ensemble constitué des $(\mathcal{A})$ équations, plus une équation quadratique en y_1, ..., y_α (par exemple l'équation $(y_1+...+y_\alpha)^2=1$). On voit alors que $(\mathcal{B})$ est un système d'exactement n équations quadratiques en $(n+1+\alpha)$ variables. De la solution de $(\mathcal{B})$ on tire immédiatement un solution pour $(\mathcal{A})$.

Notons que $(\mathcal{B})$ est un système très particulier. Le raisonnement précédent indique qu'il existe certains systèmes de n équations à $k \geq n$ inconnues, dont la résolution est un problème NP-difficile. En revanche, cela n'exclut pas la possibilité qu'*en général* de tels systèmes soient résolubles en temps polynomial. C'est d'ailleurs ce que nous verrons pour le cas particulier des systèmes de n équations avec $k \geq n^2$ variables sur un corps fini de caractéristique 2 (voir chapitre 4).

3 Isomorphismes de polynômes

3.1 Les problèmes IP et MP

Rappelons ce qu'est le problème IP [104], dans le cas particulier des formes quadratiques (le problème peut être généralisé sans difficulté aux formes cubiques, et aux formes de plus grand degré).

Soient u et n deux entiers. Soit $\mathbb{F}_q$ un corps fini. Soit $(\mathcal{A})$ un ensemble public de u équations quadratiques en n variables x_1, ..., x_n sur le corps $\mathbb{F}_q$. On peut écrire ces équations de la manière suivante :

$$y_k = \sum_i \sum_j \gamma_{ijk} x_i x_j + \sum_i \mu_{ik} x_i + \delta_k \qquad (1 \leq k \leq u). \qquad (\mathcal{A})$$

Maintenant, soit s une transformation affine et bijective des variables x_i, $1 \leq i \leq n$, et soit t une transformation affine et bijective des variables y_k, $1 \leq k \leq u$.

On note $s(x_1,...,x_n) = (x'_1,...,x'_n)$ et $t(y_1,...,y_n) = (y'_1,...,y'_n)$.

À partir de $(\mathcal{A})$ on obtient un autre ensemble $(\mathcal{B})$ de k équations qui donne les y'_k en fonction des x'_i :

$$y'_k = \sum_i \sum_j \gamma'_{ijk} x'_i x'_j + \sum_i \mu'_{ik} x'_i + \delta'_k \qquad (1 \leq k \leq u). \qquad (\mathcal{B})$$

On dira que (s,t) est un *isomorphisme* entre $(\mathcal{A})$ et $(\mathcal{B})$, et on dira que $(\mathcal{A})$ et $(\mathcal{B})$ sont *isomorphes*.

Le problème IP est le suivant : si $(\mathcal{A})$ et $(\mathcal{B})$ sont deux ensembles publics de u équations quadratiques, et si $(\mathcal{A})$ et $(\mathcal{B})$ sont isomorphes, trouver un isomorphisme (s,t) entre $(\mathcal{A})$ et $(\mathcal{B})$.

Lorsque s et t ne sont pas supposés bijectifs, le problème correspondant sera appelé le problème des «Morphismes de Polynômes» (MP).

Exemples

Dans ce paragraphe, nous allons voir que les problèmes IP et MP sont liés de façon étroite à la cryptographie, à la théorie des graphes, et aux problèmes matriciels.

Exemple de IP à deux secrets

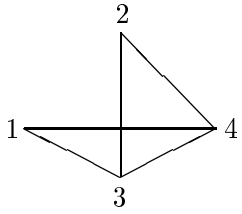
Soit $K = \mathbb{F}_2$ le corps contenant toutes les variables x_0, ..., x_4, y_0, ..., y_4, a_0, ..., a_4, b_0, ..., b_4. Soit :

$$(\mathcal{A}) \begin{cases} b_1 = a_1 + a_1a_5 + a_2a_3 + a_2a_4 + a_3a_4 \\ b_2 = a_3 + a_4 + a_5 + a_1a_2 + a_1a_4 + a_4a_5 \\ b_3 = a_5 + a_1a_2 + a_1a_3 + a_1a_5 + a_2a_3 + a_3a_5 + a_4a_5 \\ b_4 = a_2 + a_3 + a_4 + a_5 + a_1a_5 + a_3a_4 + a_3a_5 \\ b_5 = a_4 + a_1a_3 + a_1a_5 + a_2a_3 + a_2a_4 + a_2a_5 + a_3a_4 + a_3a_5 + a_4a_5 \end{cases}$$

et soit :

$$(\mathcal{B}) \begin{cases} y_1 = x_1 + x_3 + x_4 + x_5 + x_1x_2 + x_1x_3 + x_1x_5 + x_2x_4 + x_3x_5 + x_4x_5 \\ y_2 = x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_1x_5 + x_4x_5 \\ y_3 = x_1 + x_3 + x_4 + x_1x_2 + x_1x_5 + x_2x_3 + x_3x_4 \\ y_4 = x_3 + x_4 + x_1x_2 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 \\ y_5 = x_2 + x_4 + x_5 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2x_5 \end{cases}$$

Le problème consiste à trouver deux transformations s et t, bijectives et linéaires, telles que $(x_1,...,x_5) = s(a_1,...,a_5)$, $(y_1,...,y_5) = t(b_1,...,b_5)$ et qui transforment $(\mathcal{A})$ en $(\mathcal{B})$. (C'est un «problème IP à deux secrets» car il y a ici *deux* transformations affines secrètes à trouver : s et t).



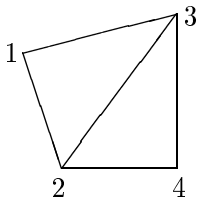


FIG. 1 – *Graphe* (I) *Graphe* (II)

Remarque 1 : Cet exemple est tiré d'un exemple donné par Tsutomu Matsumoto et Hideki Imai dans [139] pour décrire C^*, où les 8 variables sont séparées en $3+5$ variables. Le système $(\mathcal{A})$ vient de l'équation $b = a^3$ dans $\mathbb{F}_{2^5}$.

Remarque 2 : On peut montrer que – une fois déterminé s – il est facile d'en déduire t. Néanmoins, une recherche exhaustive sur s nécessiterait $2^{n^2} = 2^{25}$ opérations dans cet exemple.

Exemple de IP à un secret

Considérons le problème de trouver un isomorphisme entre les graphes (I) et (II) de la figure 1.

Soit K le corps fini $\mathbb{F}_2$.

Soit $(\mathcal{A})$ et $(\mathcal{B})$ les deux systèmes d'équations suivants :

$$(\mathcal{A}) \begin{cases} y_1 = x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_2 = x_1^2 + x_2^2 + x_3^2 + x_4^2 \end{cases} \text{ et } (\mathcal{B}) \begin{cases} y_1 = a_1a_2 + a_1a_3 + a_2a_3 + a_2a_4 + a_3a_4 \\ y_2 = a_1^2 + a_2^2 + a_3^2 + a_4^2 \end{cases}$$

Le problème est de trouver une transformation bijective et linéaire s telle que $(x_1,...,x_4) = s(a_1,...,a_4)$, et telle que – avec ce changement de variables – le système $(\mathcal{A})$ devienne le système $(\mathcal{B})$.

Si on peut trouver toutes les solutions de ce problème IP (appelé «problème IP à un secret» car il y a ici *une* transformation affine s à trouver), certaines de ces solutions donneront des isomorphismes de graphes entre le graphe (I) et le graphe (II). Ceci vient du fait que – dans $(\mathcal{A})$ – y_1 contient le monôme x_ix_j si et seulement si le point i est relié au point j dans le graphe (I). De même – dans $(\mathcal{B})$ – y_1 contient le monôme a_ia_j si et seulement si le point i est relié au point j dans le graphe (II). De plus, y_2 a été choisi comme polynôme symétrique des variables, si bien que tout isomorphisme de graphes entre le graphe (I) et le graphe (II) sera solution de ce problème IP à un secret particulier.

Exemple de MP

Les variables appartiennent à un anneau ou à un corps K. Soit $(\mathcal{A})$ et $(\mathcal{B})$ les deux systèmes d'équations suivants :

$$(\mathcal{A}) \begin{cases} b_1 = a_1a'_1 \\ b_2 = a_2a'_2 \\ b_3 = a_3a'_3 \\ b_4 = a_4a'_4 \\ b_5 = a_5a'_5 \\ b_6 = a_6a'_6 \\ b_7 = a_7a'_7 \end{cases} \quad \text{et} \quad (\mathcal{B}) \begin{cases} y_1 = x_1x'_1 + x_3x'_2 \\ y_2 = x_2x'_1 + x_4x'_2 \\ y_3 = x_1x'_3 + x_3x'_4 \\ y_4 = x_2x'_3 + x_4x'_4 \end{cases}$$

Le problème consiste à trouver deux transformations linéaires (non bijectives) s et t telles que $(a_1,...,a_7,a'_1,...,a'_7) = s(x_1,...,x_4,x'_1,...,x'_4)$, $(y_1,...,y_4) = t(b_1,...,b_7)$, et qui transforment $(\mathcal{A})$ en $(\mathcal{B})$.

On peut noter que le système $(\mathcal{B})$ correspond au produit de deux matrices 2×2 :

$$\begin{pmatrix} y_1 & y_3 \\ y_2 & y_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} \cdot \begin{pmatrix} x'_1 & x'_3 \\ x'_2 & x'_4 \end{pmatrix}$$

Dans le système $(\mathcal{A})$, on a exactement sept multiplications, si bien que résoudre le problème MP revient à répondre à la question suivante : «Comment multiplier deux matrices 2×2 avec 7 (au lieu de 8) multiplications?». (Le nombre d'additions, soustractions et multiplications par des constantes de K peut être élevé, mais le nombre de multiplications entre termes des matrices est au plus 7). Volker Strassen a trouvé en 1969 un algorithme pour multiplier deux matrices 2×2 avec 7 multiplications. Sa solution [147] est la suivante :

$$\begin{cases} a_1 = x_3 - x_4 \\ a_2 = x_1 + x_4 \\ a_3 = x_1 - x_2 \\ a_4 = x_1 + x_3 \\ a_5 = x_1 \\ a_6 = x_4 \\ a_7 = x_2 + x_4 \end{cases} \qquad \begin{cases} a'_1 = x'_2 + x'_4 \\ a'_2 = x'_1 + x'_4 \\ a'_3 = x'_1 + x'_3 \\ a'_4 = x'_4 \\ a'_5 = x'_3 - x'_4 \\ a'_6 = x'_2 - x'_1 \\ a'_7 = x'_1 \end{cases} \qquad \text{et} \qquad \begin{cases} y_1 = b_1 + b_2 - b_4 + b_6 \\ y_2 = b_4 + b_5 \\ y_3 = b_6 + b_7 \\ y_4 = b_2 - b_3 + b_5 - b_7 \end{cases}$$

3.2 IP à un secret est au moins aussi difficile que les Isomorphismes de Graphes

Dans [142], j'ai donné une réduction du problème d'*Isomorphisme de Graphes* (GI) au problème IP à un secret.

La preuve s'appuie sur un résultat général sur les permutations :

Theorème 1 *Toute permutation σ de $\{1,...,n\}$ peut être écrite de manière unique de la façon suivante :*

$$\sigma = \tau_{i_n,n} \circ \tau_{i_{n-1},n-1} \circ ... \circ \tau_{i_1,1},$$

où $i_k \in \{1,...,k\}$ pour tout k, $1 \leq k \leq n$, et où $\tau_{i,j}$ est la permutation qui échange i et j (par convention, $\tau_{i,i} = Id$).

Démonstration : On procède par récurrence sur n.

Il est facile de vérifier le résultat pour $n = 1$ et $n = 2$. Supposons que le résultat soit vrai pour un entier n donné, et prenons une permutation σ de $\{1,...,n+1\}$.

Soit $i_{n+1} = \sigma(n+1)$, et soit $\sigma' = \tau_{i_{n+1},n+1} \circ \sigma$. Le fait que $\sigma'(n+1) = n+1$ montre que σ' induit une permutation de $\{1,...,n\}$ (que l'on notera aussi σ').

D'après l'hypothèse de récurrence, il existe des indices i_1, ..., i_n satisfaisant $i_k \in \{1,...,k\}$ pour tout k, $1 \leq k \leq n$, et tels que :

$$\sigma' = \tau_{i_n,n} \circ ... \circ \tau_{i_1,1}.$$

Par conséquent, on a :

$$\sigma = \tau_{i_{n+1},n+1} \circ ... \circ \tau_{i_1,1},$$

avec $i_k \in \{1,...,k\}$ pour tout k, $1 \leq k \leq n+1$.

On a donc prouvé l'existence de la décomposition de σ.

L'unicité résulte de l'argument combinatoire suivant. Notons S_n l'ensemble de toutes les permutations de $\{1,...,n\}$ et T l'ensemble de toutes les permutations qui peuvent s'écrire $\tau_{i_n,n} \circ ... \circ \tau_{i_1,1}$. Il y a au plus n possibilités pour i_n, $n-1$ possibilités pour i_{n-1}, ..., et 1 possibilité pour i_1. Par conséquent, T contient $\leq n!$ éléments. De plus, on vient de prouver que la fonction

$$F : \begin{cases} T \to S_n \\ (\tau_{i_n,n},...,\tau_{i_1,1}) \mapsto \tau_{i_n,n} \circ ... \circ \tau_{i_1,1} \end{cases}$$

est surjective. Comme $|T| \leq |S_n|$, on peut en conclure que F est bijective. L'unicité est ainsi démontrée.

Voyons à présent comment utiliser ce théorème pour «traduire» le fait que la transformation s mise en jeu dans le problème IP correspondant à une instance de l'Isomorphisme de Graphes est caractérisée par $a_i = x_{\varphi(i)}$ pour une certaine permutation $\varphi \in S_n$. D'après le théorème 4.1, une telle permutation peut s'écrire comme le produit de (au plus) n permutations $\tau_{i,j}$.

Pour simplifier, donnons tout d'abord une «traduction» du fait qu'une transformation affine $s : x \mapsto a$ est caractérisée soit par s=Id, soit par $a_i = x_{\varphi(i)}$ (pour tout i), avec $\varphi = \tau_{ij}$ pour deux indices i et j fixés. Cela revient à dire que s est un isomorphisme de polynômes entre les deux systèmes suivants :

$$(\mathcal{A}) \begin{cases} y_0 = (X - x_i)(X - x_j) \\ y_k = x_k \quad (1 \leq k \leq n,\ k \neq i,j) \\ y_{n+1} = X \end{cases} \text{ et } (\mathcal{B}) \begin{cases} y_0 = (A - a_i)(A - a_j) \\ y_k = a_k \quad (1 \leq k \leq n,\ k \neq i,j) \\ y_{n+1} = A \end{cases}$$

En utilisant cet argument plusieurs fois, on obtient la «traduction» suivante d'un problème d'isomorphisme de graphes en un problème IP : s correspond à un isomorphisme de graphes (*i.e.* est en particulier de la forme $a_i = x_{\varphi(i)}$ pour un certain $\varphi \in S_n$) si et seulement si c'est un isomorphisme de polynômes entre les deux systèmes suivants :

$$(\mathcal{A}) \begin{cases} y_0 = \sum_{i,j} \gamma_{ij} x_i x_j \\ \forall i,j,\ 1 \leq i < j \leq n,\ (i,j) \neq (n-1,n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(X - x_i^{(\nu_{ij})}\right)\left(X - x_j^{(\nu_{ij})}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = x_k^{(\nu_{ij})} \quad (1 \leq k < i) \\ y_{(2n-1)\nu_{ij}+k} = x_k^{(\nu_{ij})} \quad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = x_k^{(\nu_{ij})} \quad (j < k \leq n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = x_k^{(\nu_{ij+1})} \quad (1 \leq k \leq n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = \left(X - x_i^{((n-2)(n+1)/2)}\right)\left(X - x_j^{((n-2)(n+1)/2)}\right) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = x_k^{((n-2)(n+1)/2)} \quad (1 \leq k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = x_k^{((n-2)(n+1)/2)} \quad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = x_k^{((n-2)(n+1)/2)} \quad (j < k \leq n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = X \end{cases}$$

et

$$(\mathcal{B})\begin{cases} y_0 = \sum_{i,j} \mu_{ij} x_i x_j \\ \forall i,j,\ 1 \le i < j \le n,\ (i,j) \ne (n-1,n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(A - a_i^{(\nu_{ij}+1)}\right)\left(A - a_j^{(\nu_{ij}+1)}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = a_k^{(\nu_{ij}+1)} \quad (1 \le k < i) \\ y_{(2n-1)\nu_{ij}+k} = a_k^{(\nu_{ij}+1)} \quad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = a_k^{(\nu_{ij}+1)} \quad (j < k \le n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = a_k^{(\nu_{ij+1})} \quad (1 \le k \le n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = (A-a_i)(A-a_j) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = a_k \quad (1 \le k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = a_k \quad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = a_k \quad (j < k \le n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = A \end{cases}$$

où $\nu_{ij} = i - 1 + \frac{(j-1)(j-2)}{2}$ et X, A, ainsi que les $x_k^{(\nu)}$ et $a_k^{(\nu)}$, sont des variables intermédiaires. Chacun de ces deux systèmes comporte $\frac{2n^3-3n^2-n+4}{2}$ équations en $\frac{(n+1)(n^2-2n+2)}{2}$ variables.

Conclusion

En résolvant IP à un secret sur un ensemble de $\mathcal{O}(n^3)$ équations quadratiques, on peut résoudre un problème d'Isomorphisme de Graphes à n sommets. Par conséquent, IP est au moins aussi difficile que GI.

Remarques

1. Dans le cas particulier où $(\mathcal{A})$ et $(\mathcal{B})$ contiennent seulement *une* équation quadratique en n variables, il existe un algorithme polynomial pour trouver l'isomorphisme [140].
2. Cela ne signifie pas que cette construction donne une nouvelle méthode plus efficace pour résoudre le problème des Isomorphismes de Graphes. En fait, bien qu'aucun algorithme polynomial ne soit connu pour le problème d'Isomorphisme de Graphes, en pratique on connaît des algorithmes très efficaces : par exemple, il est possible de trouver un isomorphisme entre des graphes de 1000 sommets – y compris pour des instances «difficiles» – en moins de 10 minutes sur un PC (voir par exemple [129] page 22). L'intérêt de la construction est donc plutôt de montrer que le problème IP à un secret n'est probablement pas résoluble par un algorithme probabiliste de complexité polynomiale (GI a été soigneusement étudié, et on pense généralement que GI n'est pas résoluble avec une complexité polynomiale).

3.3 MP est NP-difficile

Dans [142], j'ai démontré avec Jacques Patarin que le problème de Morphisme de Polynômes (MP) est NP-difficile pour tout corps fini, et pour les nombres rationnels.

La preuve utilise des propriétés des tenseurs tridimensionnels. Rappelons d'abord quelques définitions de base :

Définitions :

1. Un *tenseur tridimensionnel* est un tableau à trois dimensions $T = (t_{ijk})$ de nombres.

2. Il est dit de *rang 1* si et seulement si on peut l'écrire comme produit extérieur de trois vecteurs (*i.e.* si et seulement s'il existe trois vecteurs x, y et z tels que $m_{ijk} = x_i y_j z_k$ pour tous les indices i, j, k).
3. Le rang d'un tenseur quelconque T est le nombre minimal de tenseurs T_ν de rang 1 tels que $T = \sum_\nu T_\nu$.

Le résultat suivant sur la complexité du rang d'un tenseur tridimensionnel est dû à Johan Håstad [135] :

Theorème 2 (Håstad) *Le problème de la détermination du rang d'un tenseur est NP-complet sur tout corps fini, et NP-difficile sur les nombres rationnels.*

Voyons à présent comment cette propriété fondamentale peut s'appliquer à notre problème MP.

Supposons que l'on dispose d'un algorithme Φ pour résoudre le problème MP en temps polynomial (en particulier, cet algorithme peut être utilisé pour savoir si une instance donnée du problème MP possède ou non une solution).

Soit $T = (t_{ijk})$ un tenseur (tridimensionnel) $m \times n \times \ell$. Il est bien connu (voir par exemple [148]) que le rang de T est exactement égal au nombre minimal de multiplications nécessaires pour calculer – par un algorithme bilinéaire non commutatif – l'ensemble suivant de formes bilinéaires :

$$(\mathcal{B}) \begin{cases} y_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij1} x_i x'_j \\ \quad \vdots \\ y_\ell = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij\ell} x_i x'_j. \end{cases}$$

Soit r $(= rg(T))$ cette valeur minimale. Par définition, r peut être vu comme le plus petit entier u tel qu'il existe deux transformations linéaires s : $(x_1,...,x_m,x'_1,...,x'_n) \mapsto (a_1,...,a_u,a'_1,...,a'_u)$ et t : $(b_1,...,b_u) \mapsto (y_1,...,y_\ell)$, qui transforment

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

en $(\mathcal{B})$.

C'est une instance particulière du problème MP. Pour $u = mn$, trouver une solution (s,t) est assez facile. Il suffit de définir s et t par les formules suivantes :

$$\forall i,\ 1 \leq i \leq m,\ \forall j,\ 1 \leq j \leq n,\ \begin{cases} a_{(i-1)n+j} = x_i \\ a'_{(i-1)n+j} = x'_j. \end{cases}$$

$$\forall k,\ 1 \leq k \leq \ell,\ y_k = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ijk} b_{(i-1)n+j}.$$

Remarque : De plus, toute symétrie laisse inchangé le rang d'un tenseur tridimensionnel, si bien que l'on a en fait :

$$r \leq \min(mn,nk,km),$$

qui est l'analogue de l'inégalité classique $rg(M) \leq \min(m,n)$ pour une matrice M (bidimensionnelle) de taille $m \times n$.

L'algorithme Φ peut maintenant être utilisé pour construire un algorithme polynomial qui calcule la valeur exacte de r :

1. Poser $u = mn$.
2. Avec l'aide de Φ, essayer de trouver deux transformations linéaires $s : (x_1,...,x_m,x'_1,...,x'_n) \mapsto (a_1,...,a_u, a'_1,...,a'_u)$ et $t : (b_1,...,b_u) \mapsto (y_1,...,y_\ell)$ qui transforment

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

   en $(\mathcal{B})$.
3. Si Φ trouve une solution, remplacer u par $u-1$ et revenir à l'étape 2, sinon sortir «$r = u+1$».

Il est facile de voir que cet algorithme donne la bonne valeur de r après au plus mn appels à l'algorithme (polynomial) Φ, si bien que l'on a construit un algorithme qui calcule le rang d'un tenseur tridimensionnel en temps polynomial. D'après le résultat de Johan Håstad mentionné plus haut, on peut donc en conclure que le problème MP est NP-difficile sur tout corps fini, et sur les nombres rationnels.

Remarques :

1. Plus précisément, on a prouvé que le problème de *décision* MP (*i.e.* le problème de trouver s'il existe un morphisme entre deux ensembles donnés d'équations polynomiales multivariables) est NP-complet.
2. MP est à l'évidence un problème important en mathématiques : un algorithme efficace donnerait le nombre minimal de multiplications standard pour calculer le produit de deux matrices 3×3, 4×4 ou $k \times k$, pour k petit, et – partant – des algorithmes améliorés pour les réductions de Gauss et pour des problèmes connexes. Le meilleur algorithme connu actuellement, dû à Don Coppersmith et Shmuel Winograd [125], est asymptotiquement en $\mathcal{O}(n^\omega)$, où $\omega \simeq 2.3755$. Il est également intéressant de voir quels types de calculs intensifs ont été essayés pour résoudre ces problèmes [134].
3. Le fait que MP soit NP-difficile, et plus encore le fait que MP soit un problème important qui semble difficile même avec de petits paramètres, sont de fortes motivations pour construire des schémas cryptographiques à partir de ce problème. On sait [118, 132] que tout problème de NP peut mener à des algorithme d'authentification ou de signature asymétriques (en utilisant de «bonnes» fonctions de hachage). Comme MP est dans NP, on peut appliquer ces résultats généraux. Toutefois, ces constructions ne sont pas très pratiques, et il se peut que construire des schémas efficaces à partir de MP soit plus difficile qu'à partir d'IP.

3.4 Le problème de décision IP n'est pas NP-complet

On appelle «problème de décision IP» le problème consistant à trouver s'il existe un isomorphisme entre deux systèmes d'équations polynomiales multivariables. Dans [142], j'ai démontré avec Jacques Patarin que le problème de décision IP n'est pas NP-complet, sous l'hypothèse classique que la hiérarchie polynomiale ne s'effondre pas.

La preuve s'appuie sur les résultats généraux suivants[2] (voir [133, 119] pour les démonstra-

2. Les protocoles d'Arthur-Merlin ont été étudiés par László Babai et Shlomo Moran dans [116], mais nous n'avons pas besoin de plus de détails ici.

tions) :

Theorème 3 (Goldwasser, Sipser) *Si un problème possède une preuve interactive à nombre d'étapes constant, alors il possède aussi un protocole d'Arthur-Merlin à nombre d'étapes constant.*

Theorème 4 (Boppana, Håstad, Zachos) *Si le complément d'un problème* Π *possède un protocole d'Arthur-Merlin à nombre d'étapes constant, et si* Π *est NP-complet, alors la hiérarchie polynomiale s'effondre.*

Il reste donc à construire une *preuve interactive* à nombre d'étapes constant pour le problème de «Non Isomorphisme de Polynômes» (Non-IP). À cette fin, on utilise des techniques découvertes par Goldwasser, Micali et Rackoff [33] pour le problème analogue de «Non-Résiduosité Quadratique», et aussi utilisées par Goldreich, Micali et Wigderson [132] pour le «Non-Isomorphisme de Graphes», et par Petrank et Roth [143] pour la «Non-Équivalence de Codes».

Comme d'habitude, on suppose que deux systèmes (U_0) et (U_1) d'équations polynomiales sont publics, et qu'un prouveur, de puissance infinie (appelé Merlin), veut convaincre un vérifieur, de puissance polynomiale (appelé Arthur), que (U_0) et (U_1) ne sont pas isomorphes. Ils peuvent procéder de la manière suivante :

Première étape : Arthur choisit K nombres α_1, ..., α_K au hasard dans $\{0,1\}$. Pour chaque k, $1 \leq k \leq K$, Arthur construit un ensemble (V_k) isomorphe à (U_{α_k}), en choisissant deux permutation affines bijectives s_k et t_k et en calculant $(V_k) = t_k \circ (U_{\alpha_k}) \circ s_k$. Il envoie (V_1),...,(V_K) à Merlin.

Deuxième étape : Pout chaque k, $1 \leq k \leq K$, Merlin essaie de deviner α_k, *i.e.* il essaie de trouver si (V_k) est isomorphe à (U_0) ou (U_1). Il envoie ses réponses $(\beta_1,...,\beta_K)$ à Arthur.

Vérification finale : Arthur accepte la preuve si et seulement si $\beta_k = \alpha_k$ pour tout k, $1 \leq k \leq K$.

Il est facile de se convaincre que :

1. Si (U_0) et (U_1) sont non isomorphes, Merlin parvient toujours à convaincre Arthur.
2. Si (U_0) and (U_1) sont isomorphes, la probabilité qu'Arthur soit convaincu que (U_0) et (U_1) sont non isomorphes est au plus 2^{-K}.

4 Le problème MinRank

J'ai introduit le problème *MinRank* avec Nicolas Courtois dans [160]. Ce problème est au cœur de la sécurité de certains cryptosystèmes multivariables, notamment [3] l'analyse du schéma HFE [104], la cryptanalyse du schéma de Shamir [144] à base de *permutations birationnelles*, et ma cryptanalyse [160] du schéma TTM [165, 166]. Par ailleurs, il peut être utilisé pour bâtir des schémas asymétriques d'authentification [4].

4.1 Définition du problème MinRank

Soit r un entier et K un corps. On appelle Minrank(r) le problème suivant :

3. Voir chapitre 4, §4.
4. Voir par exemple le schéma d'authentification zero-knowledge proposé par N. Courtois dans [126].

Problème MinRank(r) : Étant donné un ensemble $\{M_1,\ldots,M_m\}$ de matrices $n \times n$ dont les coefficients sont dans K, trouver au moins un m-uplet $(\lambda_1,\ldots,\lambda_m) \in K^m$ tel que

$$\mathrm{Rang}\Big(\sum_{i=1}^{m} \lambda_i M_i\Big) \leq r.$$

Ce problème a en fait été mentionné et étudié pour la première fois par Buss, Frandsen et Shallit [120]. MinRank généralise le problème de *Rank Distance Coding* (introduit par Gabidulin [130] et étudié dans [146, 122]), qui lui-même généralise le problème de *poids minimal* (*Minimal Weight*) pour les codes correcteurs d'erreur (voir à ce sujet [117, 145, 121, 136]).

Dans [138], A. Kipnis et A. Shamir décrivent une stratégie d'attaque sur le cryptosystème HFE (inventé par J. Patarin [104]). Cela les amène à résoudre une instance de MinRank(r) (avec $r = \lceil \log_q n \rceil + 1$). À cet effet, ils introduisent la *technique de relinéarisation*. Néanmoins, l'attaque obtenue n'est pas polynomiale.

Il est à noter que l'idée de trouver des petits rangs a aussi été utilisée par D. Coppersmith, J. Stern et S. Vaudenay [123, 124] dans leur cryptanalyse du schéma de Shamir [144] à base de *permutations birationnelles*.

4.2 Complexité de MinRank

Il a été établi par Buss, Frandsen et Shallit que le problème *général* MinRank est NP-complet. Plus précisément, ils démontrent dans [120] que MinRank(r) est NP-complet quand $r = n - 1$. Cela correspond au problème consistant à trouver une combinaison linéaire des matrices $M_1,\ldots,M_m$ qui soit non inversible.

Le principe de la démonstration réside dans l'interprétation d'un système quelconque d'équations multivariables comme une instance de MinRank. On peut utiliser le même procédé pour étendre le résultat de NP-complétude aux cas $r = n-2$, $r = n-3$, ..., et même $r = n^\alpha$ (pour un exposant $\alpha > 0$ fixé). Toutefois, on n'a pas de résultat analogue pour les valeurs plus petites de r. Nous verrons d'ailleurs dans le chapitre 4 que des algorithmes polynomiaux existent lorsque r est fixé.

5 Le problème de la décomposition fonctionnelle

5.1 Motivation du problème

Lors de la construction de cryptosystèmes asymétriques à base de polynômes multivariables sur un corps K, une question naturelle se pose : est-il possible d'obtenir un algorithme plus solide en composant deux transformations, chacune d'entre elles étant donnée par un système de polynômes multivariables de degré deux ? On est ainsi conduit à examiner le problème suivant :

Problème de décomposition : Soient g et h deux fonctions allant de K^n vers K^n et qui sont décrites sous la forme de polynômes de degré total deux en n variables sur K. Alors $f = g \circ h$ est aussi une fonction de K^n vers K^n, qui est décrite par n polynômes de degré *quatre* en n variables sur K. Supposons que f soit donnée. Retrouver g et h est-il faisable (au sens de la théorie de la complexité) ?

Une réponse positive à ce problème impliquerait que les deux fonctions (dont la composition constitue la clé publique du cryptosystème) peuvent être facilement séparées l'une de l'autre. Par

conséquent, casser le cryptosystème deviendrait équivalent à casser deux sous-systèmes indépendants donnés par des polynômes quadratiques en n variables. Cela rendrait donc complètement inutile l'idée de composer plusieurs systèmes multivariables.

5.2 L'algorithme de Dickerson

Toutefois, le problème de la décomposition des polynômes multivariables a été étudié par Matthew Dickerson [127], qui donne un algorithme pour le problème suivant :

Décomposition multivariable «à gauche» : Étant donnés des polynômes f et h_1, ..., h_n dans $K[X_1,...,X_n]$, et un entier r, déterminer s'il existe un polynôme $g(x_1,...,x_n)$ de degré total au plus r qui se compose avec les h_i pour donner f. Dit autrement, existe-t-il un polynôme $g(x_1,...,x_n)$ tel que

$$f(x_1,...,x_n) = g(h_1(x_1,...,x_n),...,h_n(x_1,...,x_n))$$

et $\deg(g) \leq r$? Si oui, trouver les coefficients de g.

Dans [127, 128], Dickerson décrit le meilleur algorithme connu pour ce problème, dont la complexité est polynomiale en le degré de f, h_1, ..., h_n, mais *exponentielle* en le nombre n de variables (remarquons que notre problème de décomposition est encore plus difficile, puisque les h_i sont aussi des inconnues).

5.3 Question de la NP-difficulté

Dickerson montre aussi que le problème *général* de la décomposition des polynômes multivariables est difficile car le problème suivant est NP-difficile :

Problème de décomposition s-1 : Étant donné un polynôme unitaire monovariable $f(x)$ et un entier s, déterminer s'il existe une «décomposition s-1» de f, *i.e.* un polynôme unitaire monovariable h de degré s, et un polynôme à deux variables $g(y,x) \in K[Y,X]$ de la forme $g(y,x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$ avec α_i, β_i dans une extension algébrique $\hat{K}$ du corps K, telle que $f(x) = g(h(x),x)$. Si oui, trouver les coefficients de g et h.

Certains indices laissent penser que le problème suivant est également NP-complet (*cf* [127], problème 14, page 74) :

Décomposition multivariable de degré fixé : Étant donné un polynôme f de $K[X_1,...,X_n]$ et une partie des éléments suivants : des entiers k, r, s_1, ..., s_k, un polynôme $g(x_1,...,x_k) \in K[X_1,...,X_k]$, et des polynômes $h_1(x_1,..,x_n)$, ..., $h_k(x_1,...,x_n)$, déterminer s'il existe une décomposition fonctionnelle g, h_1, ..., h_k de f (*i.e.* $f = g(h_1,...,h_k)$) tel que $\deg(g) = r$, et $\deg(h_i) = s_i$ pour $1 \leq i \leq n$. Si oui, calculer ceux des coefficients de g et des h_i qui étaient inconnus.

Comme le remarque Dickerson (voir [127] page 75) :

> «The s-1-decomposition problem seems intuitively easier than problem 14. In problem 14, f, g and h are general multivariate polynomials of arbitrary dimension. Furthermore polynomial g takes the polynomial h_i as arguments, and we know nothing about the form of g other than its degree. In the s-1-decomposition problem, on the other hand, f and h are both univariate polynomials and g is only bivariate. Furthermore, g takes x and not another polynomial as its second argument. We also know a great deal about the structure of the polynomial g, namely that it factors as:

$g(y,x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$. However, we have tried without success to reduce the s-1-decomposition problem to problem 14.»

Le problème est toujours ouvert.

Chapitre 4

Cryptanalyse

Few persons can be made to believe that it is not quite an easy thing to invent a method of secret writing that shall baffle investigation. Yet it may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.

Edgar Allan Poe (1841)

The open cryptographic literature contains very few examples of universal methods of cryptanalysis, which can be successfully applied to a wide variety of encryption and hash functions.

Eli Biham et Adi Shamir (1993)

Sommaire

1 Introduction 50
2 Attaques par polarisation 50
2.1 Principe général 50
2.2 L'algorithme de chiffrement asymétrique D^* 50
2.3 Attaque de l'algorithme D^* par polarisation 51
2.4 Généralisation 52
2.5 Cryptanalyse de l'algorithme C^*_- par polarisation 53
3 Dégénérescence des formes quadratiques 54
3.1 Le problème de dégénérescence (DP) 54
3.2 Trois algorithmes polynomiaux 54
3.3 Application à la cryptanalyse de trois schémas asymétriques 56
3.4 Extension aux schémas à deux tours 57
4 Algorithmes pour le problème MinRank 58
4.1 État de l'art 58
4.2 La méthode du noyau 59
4.3 Application au cryptosystème TTM 59
5 Algorithmes pour les systèmes quadratiques sous-définis 60
5.1 Schémas de signature et systèmes sous-définis 60
5.2 Les méthodes générales 61
5.3 Impact sur les schémas FLASH, SFLASH et UOV 62
5.4 Le cas massivement sous-défini 62
5.5 Un nouveau problème difficile? 64

1 Introduction

Edgar Allan Poe se targuait d'être un cryptanalyste averti. Néanmoins on peut considérer que son opinion est exagérément pessimiste. Dans la cryptologie moderne, la cryptanalyse ne consiste pas tant à casser de manière astucieuse les algorithmes existants qu'à mettre en place une batterie de tests pour détecter les faiblesses potentielles des nouveaux cryptosystèmes proposés. En ce sens, la cryptographie multivariable présente des points communs avec la cryptographie symétrique[1].

Dans ce chapitre, je présente quatre méthodes génériques de cryptanalyse que j'ai développées dans [168, 141, 169, 137, 160, 155]. Elles ont eu de nombreuses répercussions, notamment sur les algorithmes C^* [162, 139], C^*_- [169], D^* [168], E^* [168], TTM [165, 166] et UOV [137].

2 Attaques par polarisation

2.1 Principe général

On considère ici des cryptosystèmes multivariables, dont la clé publique est donnée par un ensemble de polynômes quadratiques en plusieurs variables. L'idée générale de l'attaque, que j'ai proposée dans [168], est de considérer plutôt la *forme polaire* des formes quadratiques qui interviennent.

L'avantage est de pouvoir se ramener à des équations de degré un en les nouvelles variables «polarisées». Lorsque le cryptosystème est obtenu à l'aide du principe de *représentation obscure* (*obscure representation*, dû à Imai et Matsumoto [162]), la clé publique résulte de la transformation d'un polynôme multivariable par deux transformations linéaires ou affines secrètes. La polarisation peut alors mener à une attaque lorsque le polynôme interne est un *monôme*.

2.2 L'algorithme de chiffrement asymétrique D^*

Dans [168], j'ai montré comment utiliser le principe de polarisation pour obtenir une attaque efficace contre l'algorithme D^*. Cet algorithme est le cas le plus simple d'une famille d'algorithmes multivariables particulièrement intéressante puisqu'elle fournit des *permutations à sens unique avec trappe* (*trapdoor one-way permutations*). On peut trouver, toujours dans [168], un recensement exhaustif de tous les cryptosystèmes asymétriques existants qui vérifient cette propriété : il y en a peu[2].

Paramètres du système

On considère un corps $K = \mathbb{F}_q$, où $q = p^m$, m est impair et p est un nombre premier tel que $p \equiv 3 \bmod 4$. On choisit un entier n impair.

L'espace des messages sera alors:

$$\mathcal{M} = (K^n \setminus \{0\})/\{\pm 1\}$$

1. Jacques Stern écrit dans [75]: «En chiffrement conventionnel, aucun mécanisme n'est enfoui au plus profond de l'algorithme, et c'est chaque pièce de l'assemblage qui doit être vérifiée : les techniques de cryptanalyse différentielle et linéaire forment une sorte de banc d'essai, qui permet de s'assurer de l'absence de défaut visible.»

2. Notons que l'algorithme B, inventé par Imai et Matsumoto [162], et mentionné dans [168], a depuis été cassé dans [171].

Remarquons que les hypothèses faites sur q et n impliquent que -1 n'est pas un carré dans $\mathbb{F}_{q^n}$. Voir [168] pour des descriptions plus concrètes de $\mathcal{M}$.

Le schéma utilise également :

1. Une extension algébrique $\mathcal{L}_n$ de degré n sur K ;
2. Deux transformations linéaires bijectives et *secrètes* $s : K^n \to \mathcal{L}_n$ et $t : \mathcal{L}_n \to K^n$. Si on les représente par rapport à une base de $\mathcal{L}_n$ (en tant que K-espace vectoriel de dimension n), on peut voir chacune de ces deux transformations comme un système de n polynômes à n variables sur K.

Chiffrement d'un message $x \in \mathcal{M}$

Avec les notations précédentes, on définit le chiffré y de x comme l'élément de $\{+t(s(x)^2), -t(s(x)^2)\}$ qui appartient à $\mathcal{M}$ (cet élément existe et est unique par construction de $\mathcal{M}$).

Comme s et t sont de degré total 1 sur le corps K, on voit que $t(s(x)^2)$ peut se représenter comme un système de n polynômes *quadratiques* P_1, ..., P_n à n variables, dont les coefficients sont dans K. Ces polynômes constituent ainsi la clé publique, de sorte que tout le monde peut chiffrer un message : le chiffré de $x = (x_1,...,x_n) \in \mathcal{M}$ est l'élément $y = (y_1,...,y_n) = F(x)$ défini par

$$\begin{cases} y = (y_1,...,y_n) \in \mathcal{M} \\ y_1 = \pm P_1(x_1,...,x_n) \\ ... \\ y_n = \pm P_n(x_1,...,x_n) \end{cases}$$

Déchiffrement de $y \in \mathcal{M}$

On peut montrer [168] que la fonction F est une permutation de $\mathcal{M}$ dans $\mathcal{M}$, dont l'inverse F^{-1} est facile à calculer si on connaît les bijections linéaires secrètes s et t. Pour tout $y \in \mathcal{M}$, le message clair $x = F^{-1}(y)$ est donné par :

$$\begin{cases} x \in \mathcal{M} \\ x = \pm s^{-1}\Big((t^{-1}(y))^{\frac{q^n+1}{4}}\Big). \end{cases}$$

2.3 Attaque de l'algorithme D^* par polarisation

Pour illustrer le principe de polarisation, je détaille un peu plus la cryptanalyse de D^*. Celle-ci s'appuie sur l'identité

$$uv = \frac{(u+v)^2 - (u-v)^2}{2},$$

qui est vraie ici car le corps K a une caractéristique différente de 2. Par conséquent

$$\frac{F(x+x') - F(x-x')}{2} = \pm t(s(x) \cdot s(x')).$$

Et donc $\varphi(x,x') = t(s(x) \cdot s(x'))$ est donné par n formes bilinéaires *publiques*, à coefficients dans K.

On calcule l'espace vectoriel de toutes les transformations linéaires C et D de K^n dans K^n telles que

$$\forall x,x' \in K^n,\ C(\varphi(x,x')) = \varphi(D(x),x').$$

Ce K-espace vectoriel est de dimension au moins n, puisqu'on peut choisir, pour *tout* $\lambda \in \mathcal{L}_n$:

$$\begin{cases} D(x) = s^{-1}(\lambda \cdot s(x)) \\ C(y) = t(\lambda \cdot t^{-1}(y)). \end{cases}$$

On peut montrer [168] que la dimension est en fait *exactement* n. Comme l'ensemble des solutions pour C dépend de n variable libres, que l'on peut appeler Λ_1, ..., Λ_n, et ainsi noter C_Λ la solution de paramètre $\Lambda = (\Lambda_1,...,\Lambda_n)$.

On calcule alors l'espace vectoriel de toutes les transformations linéaires E de K^n dans K^n telles que

$$C_{E(\Lambda)}(\tilde{y}) = C_{E(\tilde{y})}(\Lambda).$$

On trouve à nouveau un K-espace vectoriel de dimension n.

Soit E_0 une telle solution, et $*$ la loi définie par

$$\Lambda * \tilde{y} = \tilde{y} * \Lambda = C_{E_0(\Lambda)}(\tilde{y}).$$

On peut alors calculer, en utilisant le principe *square and multiply* (appliqué à la loi $*$)

$$\tilde{y}^{*(\frac{q^n+1}{4})} = \underbrace{\tilde{y} * ... * \tilde{y}}_{\frac{q^n+1}{4} \text{ fois}} = t\Big(\mu^{\frac{q^n+1}{4}-1} \cdot t^{-1}(\tilde{y})^{\frac{q^n+1}{4}}\Big) = \pm t(\mu^{\frac{q^n+1}{4}-1} \cdot s(x)).$$

Par conséquent, il existe une transformation linéaire W de K^n dans K^n telle que tout couple clair/chiffré (x,y) vérifie

$$y^{*(\frac{q^n+1}{4})} = \pm W(x).$$

En outre, W peut facilement être trouvé par réduction de Gauss à partir d'un nombre suffisant de couples clair/chiffré [3].

Une fois que $*$ et W ont été trouvés, déchiffrer un chiffre y quelconque est facile :

$$x = \pm W^{-1}(y^{*(\frac{q^n+1}{4})}).$$

2.4 Généralisation

Toujours dans [168], j'ai montré que le même principe de polarisation fournit une attaque contre les systèmes de la famille D^* où la transformation monomiale $a \mapsto a^2$ est remplacée par un polynôme Q quelconque de degré $d < p$, où p est la caractéristique du corps K. Cela s'applique en particulier à l'algorithme de chiffrement asymétrique E^* (en degré *trois*) qui avait également été décrit dans [168].

L'idée cette fois est d'utiliser l'identité de polarisation généralisée suivante :

$$\frac{1}{2^k \cdot k!} \sum_{\{\varepsilon_1,...,\varepsilon_k\} \in \{-1,1\}^k} \Big(\prod_{j=1}^{k} \varepsilon_j\Big) \cdot \Big(\sum_{j=1}^{k} \varepsilon_j u_j\Big)^\delta = \begin{cases} 0 & \text{si } k > \delta \\ \prod_{i=1}^{k} u_i & \text{si } k = \delta \end{cases}$$

Ainsi, si le polynôme interne du cryptosystème est $Q(X) = \sum_{i=1}^{d} \alpha_i X^i$, on peut retrouver la valeur de d en appliquant l'identité de polarisation avec des valeurs décroissantes de k, jusqu'à

3. Plus précisément, on montre dans [168] qu'il suffit d'en avoir $\frac{n^2}{n-1}$.

ce que le résultat observé soit non nul. Cela permet à l'attaquant de déterminer le degré d de Q, même s'il ne le connaissait pas *a priori*.

L'application de l'identité de polarisation avec $k = d$ ne fait intervenir que le monôme dominant $\alpha_d X_d$ de Q. Comme dans le cas de D^*, on peut en déduire une loi $*$ sur d variables (et par conséquent une loi $*$ sur deux variables : $X'_1 * (X'_2 * ... * X'_n)$). À partir de cette loi $*$, on peut trouver des valeurs β_1, ..., β_d et un polynôme $R(X) = \sum_{i=1}^{d} \beta_i X_i$ qui engendre les équations publiques. On peut alors déchiffrer n'importe quel chiffré sans avoir à connaître s ni t. Il suffit d'utiliser R et $*$ au lieu de Q et de la multiplication ordinaire.

2.5 Cryptanalyse de l'algorithme C^*_- par polarisation

Il est tentant d'essayer d'appliquer cette technique à l'algorithme HFE [104]. Néanmoins, dans ce cas, le degré d du polynôme interne Q est toujours plus grand que la caractéristique p. Il semble donc que la méthode de polarisation échoue avec HFE. Toutefois dans [169], avec Nicolas Courtois, j'ai présenté une cryptanalyse de l'algorithme C^*_- qui s'appuie également sur l'idée de polarisation (avec cette fois le degré du polynôme interne au moins égal à la caractéristique du corps). Cette cryptanalyse a permis de mieux comprendre le comportement des cryptosystèmes multivariables vis à vis de la «suppression» de polynômes dans la clé publique (opération appelée aujourd'hui «$-$» [197]), et ainsi de pouvoir fixer des paramètres convenables pour l'algorithme SFLASH [81, 82] soumis au projet européen NESSIE (voir chapitre 5).

L'algorithme C^*_-, proposé dans [169], est une variante de l'algorithme C^* de Matsumoto et Imai [162, 139], consistant à ne pas publier r des n polynômes quadratiques qui constituent la clé publique P. La stratégie de notre attaque consiste à retrouver les r équations manquantes, et ensuite à appliquer l'attaque classique de [140] sur le système ainsi complété[4].

L'attaque décrite dans [169] a une complexité $\mathcal{O}(q^r)$. L'idée de départ consiste à nouveau à considérer la *forme polaire* de la clé publique P :

$$Q(x,t) := P(x+t) - P(x) - P(t).$$

Donnons ici le début de l'attaque qui en découle. On note $P_{(r+1)...n}$ la partie «publiée» de la clé publique (les r premiers polynômes restant secrets).

1. On choisit aléatoirement $t \neq 0$ et $x^{(0)}$.
2. On calcule $z_{(r+1)...n} := Q_{(r+1)...n}(x^{(0)},t)$.
3. On résout alors l'équation
$$Q_{(r+1)...n}(x,t) = z_{(r+1)...n}$$
où x est l'inconnue. Il y a au moins deux solutions ($x^{(0)}$ and $x^{(0)} + t$) et au plus 2^{r+1} solutions. Cela vient du fait que – pour une valeur donnée $z_{1...r}$ – (parmi 2^r possibles), l'équation $Q(x,t) = z$ a 0 ou 2 solutions (*cf* [169]).

On exécute les étapes 1 à 3 jusqu'à obtenir le nombre maximal 2^{r+1} de solutions (en moyenne il faut répéter le processus environ 2^r fois). On obtient ainsi $t \neq 0$ and $x^{(0)}$ tels que l'équation

$$Q_{(r+1)...n}(x,t) = z_{(r+1)...n}$$

4. En ce qui concerne l'attaque [140] de J. Patarin sur le cryptosystème C^* originel de Matsumoto et Imai, Hans Dobbertin a indiqué récemment qu'il avait découvert indépendamment la même attaque en 1993-94 lorsqu'il travaillait pour le BSI-Institute.

a exactement 2^{r+1} solutions:

$$\left\{x^{(0)},x^{(0)}+t,x^{(1)},x^{(1)}+t,...,x^{(2^r-1)},x^{(2^r-1)}+t\right\}.$$

Soit maintenant un entier k tel que $1 \leq k \leq r$. Pour la moitié des solutions, on a $Q_k(x,t) = 0$, et pour l'autre moitié on a $Q_k(x,t) = 1$. Et cela reste vrai si on se restreint au sous-ensemble

$$\left\{x^{(0)},...,x^{(2^r-1)}\right\}$$

de l'ensemble des solutions. En sommant, on en déduit

$$\sum_{\nu=0}^{2^r-1} Q_k(x^{(\nu)},t) = 2^{r-1},$$

ce qui donne une équation de degré 1 sur les $\frac{n(n-1)}{2}+1$ coefficients de Q_k.

L'attaque complète [169] consiste à répéter le processus précédent suffisamment de fois pour pouvoir retrouver les r équations manquantes (correspondant à $1 \leq k \leq r$) par réduction de Gauss, puis – une fois ces équations récupérées – à appliquer l'attaque classique de l'algorithme C^* décrite dans [140].

3 Dégénérescence des formes quadratiques

3.1 Le problème de dégénérescence (DP)

La cryptanalyse d'un certain nombre de cryptosystèmes multivariables repose sur l'étude du problème de dégérescence suivant[5] :

Problème de dégénérescence :
Étant donnée une équation à plusieurs variables $y = P(x_1,...,x_n)$, où P est un polynôme de degré deux en n variables x_1, ..., x_n, trouver une transformation affine des n variables x_i en (au plus) $n-1$ variables x'_1, ..., x'_{n-1} (*i.e.* $\forall i$, $1 \leq i \leq n$, $x_i = P'_i(x'_1,...,x'_{n-1})$, où P'_i est de degré total un), et un polynôme Q de degré total deux en les (au plus $n-1$) nouvelles variables, de telle sorte que $y = Q(x'_1,...,x'_{n-1})$.

J'ai proposé dans [141] plusieurs méthodes efficaces pour résoudre ce problème. Dans ce paragraphe, nous verrons que ces algorithmes ont des conséquences sur la cryptanalyse des algorithmes C^* [162, 139], «One Round of S-Boxes» [141] et TTM [165, 166].

3.2 Trois algorithmes polynomiaux

Plusieurs algorithmes sont décrits dans [141] pour résoudre en temps polynomial le problème de dégénérescence.

En utilisant la forme canonique des formes quadratiques
Cette première méthode s'appuie sur un résultat classique[6] de représentation d'une forme quadratique sur un corps de caractéristique *paire*.

5. Ce problème a été mentionné par Hironaka [161] dans le cadre de la résolution des singularités d'une variété algébrique sur un corps de caractéristique zéro.

6. Voir par exemple l'ouvrage classique de Lidl et Niederreiter [164].

Définitions :

1. Une *forme quadratique* sur K est un polynôme homogène de $K[X_1,...,X_n]$, de degré deux, ou bien le polynôme nul.
2. Deux polynômes f et g de degré ≤ 2 sur K sont dits *équivalents* si f peut être transformé en g au moyen d'un changement de variables linéaire bijective sur les indéterminées.
3. Une forme quadratique f en n indéterminées est dite *non dégénérée* si elle n'est pas équivalent à une forme quadratique en moins de n indéterminées.

Theorème 5 *Soit $f \in K[X_1,...,X_n]$ une forme quadratique non dégénérée sur $K = \mathbb{F}_q$, où q est pair. Si n est impair, alors f est équivalente à*

$$x_1x_2 + x_3x_4 + ... + x_{n-2}x_{n-1} + x_n^2.$$

Si n est pair, alors f est équivalente soit à

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n,$$

soit à une forme quadratique du type

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n + x_{n-1}^2 + ax_n^2 \qquad (a \in K^*).$$

La preuve classique donnée dans [164] donne en fait un algorithme constructif et de complexité polynomiale pour transformer la forme quadratique initiale en sa forme canonique. En outre, le même algorithme peut s'adapter sans problème au cas d'une forme quadratique *dégénérée*. Pour une telle forme f, on peut définir le «nombre de variables indépendantes de f» comme le plus petit entier k tel que f est équivalente à une forme quadratique en k indéterminées. L'algorithme de [164] montre que ce nombre k peut être calculé en temps polynomial.

En utilisant la linéarité dans certaines directions

Cet algorithme est probablement le plus simple pour les formes quadratiques [7].

Pour une forme quadratique Q dégénérée, il existe certaines valeurs $d = (d_1,...,d_n)$, $d \neq 0$, telles que

$$\forall x \in K^n,\ Q(x+d) = Q(x) \tag{1}$$

On peut déterminer ces valeurs d de la manière suivante. Chaque monôme x_ix_j de Q donne une contribution $(x_i+d_i)(x_j+d_j) - x_ix_j$, c'est-à-dire $x_id_j + x_jd_i + d_id_j$ dans l'expression $Q(x+d) = Q(x)$. Comme l'identité (1) est vraie pour tout $x \in K^n$, on en déduit que pour tout k, $1 \leq k \leq n$, le coefficient de x_k (qui est une combinaison linéaire des d_i) est nul. On peut alors déterminer le K-espace vectoriel des solutions d par réduction de Gauss.

En utilisant le calcul différentiel

Dans le cas de polynômes quadratiques, nous avons vu que la dégénérescence peut être détectée grâce à l'existence d'une forme canonique (ou bien par réduction de Gauss). Toutefois, dès que le degré devient ≥ 3, il n'existe pas de telle forme canonique [8]. C'est pourquoi j'ai proposé dans [141] une nouvelle méthode qui utilise une analogie avec la géométrie différentielle.

7. Cette idée a été suggérée en 1996 par Don Coppersmith, dans une communication personnelle à Jacques Patarin [153]. Il s'agissait pour lui d'un autre contexte : la cryptanalyse du schéma «Oil and Vinegar de degré 3», voir notre article [137] §9.3.

8. Ce thème a mobilisé beaucoup de mathématiciens à la fin du XIXe siècle, notamment Dickson et Hilbert (c'est une partie d'un programme bien plus vaste : la théorie des invariants). Mais aucune classification générale n'est connnue, même en se limitant au cas des formes cubiques.

Pour déterminer si une forme (correspondant à un polynôme Q) est dégénérée, l'idée est d'utiliser la notion de *gradient*. Pour tout polynôme Q à coefficients dans K, le gradient de Q au point $x = (x_1,...,x_n)$ est défini par :

$$\textbf{grad } Q(x) = \Big(\frac{\partial Q}{\partial x_1}(x_1,...,x_n),...,\frac{\partial Q}{\partial x_n}(x_1,...,x_n)\Big).$$

Le théorème suivant montre le lien entre les gradients de Q et le «nombre de variables indépendantes» de Q.

Theorème 6 *Soit Q une forme quadratique en n variables en n variables sur le corps K. Deux cas se présentent :*
1) Si la caractéristique de K est > 2, alors le nombre k de variables indépendantes de Q est égal à la dimension du sous-K-espace vectoriel $\mathcal{A}$ engendré par tous les **grad** $Q(x)$ *($x \in K^n$).*
2) Si la caractéristique de K est 2, alors il est facile de trouver un changement de variables linéaire bijectif sur les indéterminées qui transforme Q en une forme quadratique

$$R(x'_1,...,x'_n) = \Phi(x'_1,\ldots,x'_{\dim(\mathcal{A})}) + \sum_{i=\dim(\mathcal{A})+1}^{n} \lambda_i x_i'^2,$$

où Φ est une forme quadratique sur K. De plus, le nombre de variables indépendantes de Q est

$$k = \dim(\mathcal{A}) + \begin{cases} 0 & si\ \forall i,\ \lambda_i = 0. \\ 1 & si\ \exists i,\ \lambda_i \neq 0. \end{cases}$$

La détermination du «nombre de variables indépendantes» de Q revient donc essentiellement à trouver la description du sous-espace vectoriel $\mathcal{A}$ de K^n, ce qui peut se faire en temps polynomial [141].

Le grand avantage de cette méthode est que l'on peut obtenir un théorème analogue au théorème 6 pour les formes cubiques, et plus généralement pour les formes de degré quelconque sur K.

3.3 Application à la cryptanalyse de trois schémas asymétriques

Une première application du *problème de dégénérescence* est la cryptanalyse du schéma «One-Round of S-Boxes», proposé à titre d'exemple dans [141]. Cette cryptanalyse s'appuie sur le fait qu'avec une probabilité importante, une certaine combinaison linéaire $Q = \sum_{i=1}^{n} \alpha_i P_i(x_1,...,x_n)$ des polynômes publics qui constituent la clé publique est dégénérée. En utilisant l'un des algorithmes décrits précédemment, on peut alors identifier, dans le système des polynômes publics, la contribution de chacune des boîtes-S. La cryptanalyse complète est décrite dans [141].

Une deuxième application de la détection des dégénérescences est la cryptanalyse de certains cas particuliers du schéma C^* de Matsumoto et Imai [162, 139]. En effet, pour les versions qui possèdent plusieurs branches, chacune faisant intervenir n_i variables [9], les algorithmes précédents mènent à une nouvelle cryptanalyse dans le cas où les valeurs n_i ne sont pas trop grandes (voir [141] pour l'étude complète). Remarquons que cette cryptanalyse s'appuie uniquement sur la petite taille des branches, et non sur les propriété algébriques des transformations internes (comme dans la cryptanalyse générale de C^* [140]).

9. Voir [139] pour la définition du concept de branche et des nombres n_i.

Une troisième application du *problème de dégérescence* est la cryptanalyse du schéma TTM dans sa version signature. Ce schéma, proposé par T.-T. Moh [165, 166] (voir aussi [151]), utilise le principe de *représentation obscure* d'Imai et Matsumoto [162], pour cacher une transformation interne par deux bijections linéaires secrètes. La transformation interne elle-même est ici obtenue comme composée de deux *fonctions de De Jonquières*, ce qui lui confère une structure quasi-triangulaire (la «pointe» du triangle étant tronquée). Dans l'attaque du schéma de signature TTM que j'ai décrite dans [160] (publié avec Nicolas Courtois), on peut retrouver la structure du quasi-triangle en appliquant à nouveau les méthodes de détection des dégénérescences à la clé publique du cryptosystème : c'est l'attaque «par le bas» [10].

3.4 Extension aux schémas à deux tours

Dans le schéma 2R («Two Rounds of S-Boxes») que j'ai proposé avec Jacques Patarin dans [141], la clé publique P est un système de n polynômes à n variables, qui correspond à la transformation $P : K^n \to K^n$ définie par

$$P = t \circ \psi \circ s \circ \varphi \circ r,$$

où r, s, t sont des transformations linéaires bijectives secrètes, et φ, ψ sont construits à l'aide de boîtes-S.

Il est instructif de voir ce que donne la technique du gradient dans ce contexte. Voici l'idée dans ses grandes lignes.

Posons $h = s \circ \varphi \circ r$, que l'on peut aussi représenter sous la forme de n polynômes quadratiques h_1, ..., h_n en n indéterminées sur K. Soit $f = \sum_i \alpha_i P_i$, où les α_i sont choisis aléatoirement dans K. Soit enfin $g = \sum_i \alpha_i (t \circ \psi)_i$, de telle sorte que $f = g \circ h$.

Par simplicité, on suppose que les tailles des boîtes-S de ψ sont $n_1 = ... = n_8 = 8$. Alors, avec une probabilité $\frac{1}{2^8} = \frac{1}{256}$, g n'a pas de composante provenant de la première boîte-S.

Un calcul différentiel donne

$$\mathbf{grad}\ f(x) = \sum_{r=1}^{n} \frac{\partial g}{\partial c_r}(h(x))\ \mathbf{grad}\ h_r(x) = \sum_{r=9}^{n} \frac{\partial g}{\partial c_r}(h(x))\ \mathbf{grad}\ h_r(x).$$

Par conséquent $\mathbf{grad}\ f(x)$ vit dans le sous-espace vectoriel engendré par ($\mathbf{grad}\ h_9(x)$,...,$\mathbf{grad}\ h_n(x)$), et cela est vrai pour tout $x = (x_1,...,x_n) \in K^n$. On en déduit les égalités suivantes :

$$\begin{cases} det(\mathbf{grad}\ f(x), \mathbf{grad}\ h_2(x), ..., \mathbf{grad}\ h_n(x)) = 0 \\ det(\mathbf{grad}\ h_1(x), \mathbf{grad}\ f(x), \mathbf{grad}\ h_3(x), ..., \mathbf{grad}\ h_n(x)) = 0 \\ ... \\ det(\mathbf{grad}\ h_1(x), ..., \mathbf{grad}\ h_7(x), \mathbf{grad}\ f(x), \mathbf{grad}\ h_9(x), ..., \mathbf{grad}\ h_n(x)) = 0 \end{cases}$$

Soit maintenant $(-1)^{i_0+j_0}\Delta_{i_0 j_0}(x)$ la valeur du déterminant de la matrice

$$\Big(\frac{\partial h_i}{\partial x_j}(x)\Big)_{\substack{1 \le i \ne i_0 \le n \\ 1 \le j \ne j_0 \le n}}.$$

Les équations précédentes deviennent alors :

$$\sum_{j=1}^{n} \Delta_{ij}(x) \frac{\partial f}{\partial x_j}(x) = 0 \qquad (1 \le i \le 8),$$

10. Voir notre article [160] pour une description complète de l'attaque sur TTM en mode signature.

où $\Delta_{ij}(x)$ est inconnu et est polynomial de degré total $(n-1)$ en x.

On peut imaginer d'utiliser un grand nombre de valeurs de x pour obtenir des relations entre les coefficients des polynômes $\Delta_{ij}(x)$, mais ce projet ne semble pas réalisable en pratique, à cause du degré très élevé de ces polynômes[11].

Remarquons que pour un schéma avec *un seul tour*, l'attaque réussit parce que les Δ_{ij} deviennent des polynômes *constants*. On peut donc alors trouver les Δ_{ij} en essayant environ n valeurs différentes de x, puis en appliquant une réduction de Gauss.

4 Algorithmes pour le problème MinRank

Rappelons la définition du problème *MinRank*, que j'ai introduit avec Nicolas Courtois dans [160]. Soit r un entier et K un corps. On appelle Minrank(r) le problème suivant :

Problème MinRank(r) : Étant donné un ensemble $\{M_1,\ldots,M_m\}$ de matrices $n \times n$ dont les coefficients sont dans K, trouver au moins un m-uplet $(\lambda_1,\ldots,\lambda_m) \in K^m$ tel que

$$\mathrm{Rang}\Big(\sum_{i=1}^{m} \lambda_i M_i\Big) \leq r.$$

Si le problème général est NP-complet (voir chapitre 3), des algorithmes polynomiaux existent quand r est fixé.

4.1 État de l'art

On peut trouver une liste complète des algorithmes mis au point pour résoudre MinRank dans la thèse de Nicolas Courtois [154]. Dans ce paragraphe, ω désigne l'exposant de la complexité de la réduction de Gauss[12].

Les attaques par énumération

- La recherche exhaustive peut s'avérer utile dans certains cas. Sa complexité est
$$\mathcal{O}(q^m \cdot r^\omega).$$
- Lorsque $r \approx n$, une méthode a été suggérée par Claus Schnorr [154]. Si on pose $s = n - r$, sa complexité est
$$\mathcal{O}(q^{s^2} \cdot n^3).$$

Cas où $m \gg n$

- L'algorithme «Big m», proposé par Nicolas Courtois [154] est en
$$\mathcal{O}\Big(q^{\max(0,n(n-r)-m+1)} \cdot (n(n-r))^\omega\Big).$$
- L'algorithme du syndrome, issu d'une remarque de Gabidulin adaptée par N. Courtois [154], donne une complexité
$$\mathcal{O}\Big(q^{\max(\frac{n^2-m-1}{2}, nr-m-\frac{r^2}{4})} \cdot rn^2\Big).$$

11. Notons qu'une attaque plus prometteuse contre le schéma 2R, basée également sur le calcul différentiel, a été proposée par Ding-Feng Ye, Kwok-Yan Lam et Zong-Duo Dai dans [170].

12. Le meilleur exposant connu, dû à D. Coppersmith et S. Winograd [125], est $\omega \simeq 2.3755...$, mais la constante impliquée est très élevée, ce qui fait qu'on choisit le plus souvent $\omega = 3$.

Cas où $r \ll n$

- La méthode du *système surdéfini* a été proposée initialement par A. Kipnis et A. Shamir dans leur analyse [138] du schéma HFE [104]. L'algorithme nécessite la condition

$$n(n-r) \gg r(n-r) + m$$

et sa complexité est

$$\mathcal{O}(n^{\omega r}).$$

- La méthode des sous-matrices est issue de la cryptanalyse [123, 124] que D. Coppersmith, J. Stern et S. Vaudenay ont faite du schéma de Shamir [144] à base de *permutations birationnelles*. L'idée est de remarquer que – pour une matrice de rang r – toutes les sous-matrices de taille $(r+1) \times (r+1)$ sont singulières. L'algorithme fonctionne à condition que

$$\sum_{i=0}^{r+1} \binom{m}{i} \leq \binom{n}{r+1}^2$$

et donne une complexité

$$\mathcal{O}\left(\frac{m^{\omega(r+1)}}{(r+1)!}\right).$$

4.2 La méthode du noyau

Cette méthode, que j'ai proposée dans [160], est la plus puissante actuellement connue pour le problème MinRank. Elle est conçue à l'origine pour le cas où r est petit, mais son domaine d'applicabilité s'étend bien au-delà.

L'idée de départ consiste à deviner un certain nombre de vecteurs qui appartiennent au noyau de la matrice recherchée M. Puisque le rang de M est r, la probabilité qu'un vecteur soit dans le *noyau* est $\frac{1}{q^r}$.

On essaye alors de trouver $\lceil \frac{m}{n} \rceil$ vecteurs appartenant au noyau de M. Comme la probabilité de succès égale à $\frac{1}{q^r}$ pour chacun d'entre eux, on arrive à deviner $\lceil \frac{m}{n} \rceil$ vecteurs X_1, ..., $X_{\lceil \frac{m}{n} \rceil}$ du noyau avec une complexité de

$$\mathcal{O}\left(q^{\lceil \frac{m}{n} \rceil}\right).$$

De chacun de ces vecteurs X_i $(1 \leq i \leq \lceil \frac{m}{n} \rceil)$, on tire n équations linéaires sur les λ_j $(1 \leq j \leq m)$:.

$$0 = M \cdot X_i = \sum_{j=1}^{m} \lambda_j (M_j \cdot X_i).$$

En tout on a donc $\lceil \frac{m}{n} \rceil \cdot n \geq m$ équations linéaires à résoudre, et m inconnues λ_j $(1 \leq j \leq m)$. La complexité de l'algorithme est

$$\mathcal{O}\left(q^{\lceil \frac{m}{n} \rceil} \cdot m^{\omega}\right).$$

4.3 Application au cryptosystème TTM

J'ai utilisé la méthode du noyau pour cryptanalyser l'algorithme de chiffrement TTM, dans un article écrit avec N. Courtois [160].

Le cryptosystème TTM[13], déjà évoqué dans sa version «signature» au paragraphe 3.3, a été proposé par T.-T. Moh [165, 166]. Comme on l'a dit, le schéma s'appuie sur le principe de *représentation obscure* d'Imai et Matsumoto [162] afin de cacher une transformation interne au moyen de deux bijections linéaires secrètes. En mode chiffrement, la transformation interne elle-même est encore obtenue comme composée de deux *fonctions de De Jonquières*.

On a donc toujours une structure quasi-triangulaire (la «pointe» du triangle étant tronquée). Contrairement au cas de la signature, où l'attaque «par le bas» décrite au paragraphe 3.3 visait la «base» du triangle, l'attaque que j'ai proposée dans [160] cherche à identifier la pointe (tronquée) du triangle : c'est l'attaque «par le haut».

Dans la version de TTM décrite dans [165, 166], on est conduit à résoudre un problème MinRank avec $r = 2$. La méthode du noyau peut alors s'appliquer[14]. Avec les paramètres donnés dans [165, 166], la complexité de l'attaque obtenue est en 2^{52}. Avec N. Courtois, j'ai pu casser le challenge «TTM 2.1» proposé par l'auteur du schéma[15].

5 Algorithmes pour les systèmes quadratiques sous-définis

5.1 Schémas de signature et systèmes sous-définis

La plupart des schémas multivariables s'appuient sur le problème de la résolution de systèmes d'équations polynomiales multivariables sur un corps fini K. Dans la cas quadratique, le problème général – appelé MQ – est NP-complet[16].

Dans le cas des schémas de signature, la clé publique est un système de m équations quadratiques $G_i(x_1,...,x_n)$ $(1 \leq i \leq m)$ à n variables x_j $(1 \leq j \leq n)$ sur un corps fini $\mathbb{F}_q$ de (petit) cardinal q, où m et n sont des entiers suffisamment grands. Les messages sont représentés (*via* une fonction de hachage) comme des éléments du $\mathbb{F}_q$-espace vectoriel $\mathbb{F}_q^m$, et les signatures correspondantes sont des éléments de $\mathbb{F}_q^n$. Un élément $x = (x_1,...,x_n)$ est une signature valide d'un message M si on a

$$\forall i,\ 1 \leq i \leq m,\ G_i(x_1, \ldots, x_n) = y_i,$$

où $y = (y_1, \ldots, y_m)$ est la représentation du message M.

En pratique, une trappe est cachée dans le schéma de telle façon que la clé publique reste indistinguable d'un système d'équations quadratique aléatoire. La connaissance de la trappe permet au signataire légitime de trouver une solution x du système multivariable pour un message donné M (et donc une valeur donnée y).

Plusieurs schémas multivariables de signature utilisent ce problème MQ (avec m équations quadratiques à n variables) dans le cas particulier $n > m$. Il en est ainsi par exemple des cryptosystèmes FLASH [81] et SFLASH [81, 82], étudiés dans le cadre du projet européen NESSIE[17],

13. TTM signifie «Tame Transformation Method». Ce sont aussi les initiales de Tzuong-Tsieng Moh.

14. Voir notre article [160] pour une description complète de l'attaque sur TTM en mode chiffrement.

15. Il est à noter que T.-T. Moh a ensuite proposé des modifications de son schéma, notamment dans [167] (voir aussi [152]). Malheureusement on peut encore appliquer la méthode du noyau avec $r = 2$. De plus, même si on parvenait à rendre r suffisamment grand pour que l'attaque ne soit plus applicable, une nouvelle cryptanalyse proposée par Ding, Hodges et Schmidt [158, 159] semble encore plus radicale. Elle s'attaque en effet – en une variante ingénieuse de la cryptanalyse de C^* [140] – à la structure algébrique interne des fonctions de De Jonquières, ce qui laisse peu d'espoir de pouvoir réparer le schéma.

16. Voir le chapitre 3, §2.

17. Voir le chapitre 5 pour les caractéristiques de SFLASH.

ou bien du schéma UOV [137]. Le problème considéré est donc dans ce cas celui d'un système quadratique *sous-défini* [18].

5.2 Les méthodes générales

On peut trouver une liste complète des algorithmes généraux mis au point pour résoudre les systèmes sous-définis dans l'article que j'ai publié sur ce sujet avec N. Courtois, W. Meier et J.-D. Tacier [155].

On peut montrer qu'en moyenne il y a q^{n-m} solutions au problème MQ. L'objectif est d'obtenir un algorithme qui en trouve au moins une, ceci plus rapidement que la recherche exhaustive (qui est en q^m). Dans [155] nous avons proposé trois algorithmes pour y parvenir.

Les algorithmes valables en toute caractéristique

- L'*algorithme A* utilise le paradoxe des anniversaires et sa complexité est q^{m-k}, où le paramètre k est défini par
$$k = \min\left(\frac{m}{2}, \lfloor\sqrt{\frac{n}{2} - \sqrt{\frac{n}{2}}}\rfloor\right).$$
Par exemple, pour un système de $m = 20$ équations quadratiques à $n = 40$ variables, on trouve $k = 4$. Ainsi, si $K = \mathbb{F}_{16}$, cela donne une complexité de 2^{64}, au lieu des 2^{80} de la recherche exhaustive.
- L'*algorithme B* utilise la relinéarisation pour réduire la complexité de la résolution de MQ. Sa complexité est de la forme $K \cdot q^{m-k}$, où
$$k = \lfloor\sqrt{2m+2} - \frac{3}{2}\rfloor$$
et où le facteur K a une croissance polynomiale de petit degré.

Un algorithme valable lorsque K est de caractéristique 2

- L'*algorithme C* utilise la *forme canonique* des formes quadratiques. Sur le système ainsi transformé, le principe de relinéarisation est appliqué (ou bien la méthode XL). La complexité de l'algorithme dépend de celle d'XL. Par exemple, si $m = 16$, $q = 2^s$ et si on fait l'hypothèse $n \geq 3m$, on obtient une complexité comprise entre 2^{4s+26} et 2^{5s+25}.

Comparaison des algorithmes A, B et C

Quand $n \to m$, la complexité des trois algorithme tend vers q^m. Les résultats obtenus suggèrent que, même dans ce cas sous-défini ($n \gg m$), le problème MQ reste exponentiellement difficile, tant que $n < m^2$.

Selon le choix des paramètres q, m ou n, l'un ou l'autre des trois algorithmes peut être le plus adapté. Par exemple, si on prend $q = 2^s$, $n = 48$, l'algorithme A a une complexité 2^{12s} et l'algorithme C une complexité 2^{5s+26}. C'est donc l'algorithme C qu'il faut privilégier dès que $s \geq 4$.

18. Notons que, dans le cas – très différent – des systèmes *sur-définis*, plusieurs méthodes prometteuses ont été proposées, notamment la *relinéarisation* par Kipnis et Shamir [138], puis sa généralisation XL par Courtois, Klimov, Patarin et Shamir [156] (voir aussi [157] pour le cas où $K = \mathbb{F}_2$).

5.3 Impact sur les schémas FLASH, SFLASH et UOV

Les résultats précédents ont pour application immédiate de fournir de nouveaux critères pour le choix des paramètres dans certains schémas de signature multivariable. C'est le cas des cryptosystèmes FLASH [81] et SFLASH [81, 82], soumis au projet européen NESSIE, ainsi que du schéma UOV (*Unbalanced Oil and Vinegar*) [137].

La sécurité des schémas FLASH et SFLASH n'est pas mise en danger par les algorithmes décrits précédemment. Par exemple SFLASH utilise les paramètres $m = 26$, $n = 37$ et $q = 128$, et la meilleure attaque que l'on puisse dériver des algorithmes A, B ou C est en 2^{156}...

En revanche, pour le schéma UOV [137], ces algorithmes montrent que certains choix de paramètre sont à éviter. Par exemple, pour les valeurs $m = 16$, $n = 64$, $q = 16$, qui assurent une bonne sécurité face aux attaques utilisant les bases de Gröbner ou bien les techniques matricielles (notamment celle proposée par Kipnis et Shamir dans [163]), l'algorithme A donne une complexité 2^{44} ($k = 5$) et l'algorithme C une complexité comprise entre 2^{42} et 2^{45}. Il faut donc bien tenir compte de cette nouvelle classe d'attaques pour la mise au point d'une version pratique de UOV.

5.4 Le cas massivement sous-défini

Si les algorithmes A, B, C, conçus pour résoudre les systèmes quadratiques sous-définis ont une complexité exponentielle, on peut se demander si cela reste vrai lorsque le nombre n d'indéterminées devient *beaucoup* plus grand que le nombre m d'équations. C'est ce problème que j'ai résolu avec A. Kipnis dans [137], dans le cas où la caractéristique du corps est égale à 2, en construisant un algorithme pour MQ qui est polynomial dès que $n \geq m(m+1)$.

Donnons les grandes lignes de l'algorithme. Soit $(\mathcal{S})$ le système suivant :

$$(\mathcal{S}) \quad \begin{cases} \sum\limits_{1 \leq i \leq j \leq n} a_{ij1} x_i x_j + \sum\limits_{1 \leq i \leq n} b_{i1} x_i + \delta_1 = 0 \\ \qquad \vdots \\ \sum\limits_{1 \leq i \leq j \leq n} a_{ijm} x_i x_j + \sum\limits_{1 \leq i \leq n} b_{im} x_i + \delta_m = 0 \end{cases}$$

L'idée principale consiste à appliquer un changement de variable de la forme

$$\begin{cases} x_1 = \alpha_{1,1} y_1 + \alpha_{2,1} y_2 + ... + \alpha_{t,1} y_t + \alpha_{t+1,1} y_{t+1} + ... + \alpha_{n,1} y_n \\ \quad \vdots \\ x_n = \alpha_{1,n} y_1 + \alpha_{2,n} y_2 + ... + \alpha_{t,n} y_t + \alpha_{t+1,n} y_{t+1} + ... + \alpha_{n,n} y_n \end{cases}$$

dont les coefficients $\alpha_{i,j}$ (pour $1 \leq i \leq t$, $1 \leq j \leq n$) seront trouvés pas à pas, de façon à obtenir un système transformé $(\mathcal{S}')$ (écrit avec les nouvelles variables y_1, ..., y_n) facile à résoudre.

On commence par choisir aléatoirement $\alpha_{1,1}$, ..., $\alpha_{1,n}$. Puis on calcule $\alpha_{2,1}$, ..., $\alpha_{2,n}$ tels que le système $(\mathcal{S}')$ ne contienne pas de terme $y_1 y_2$. Cette condition est équivalente à un système de m équations *linéaires* sur les n variables $\alpha_{2,j}$ $(1 \leq j \leq n)$:

$$\sum_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{1,i} \alpha_{2,j} = 0 \qquad (1 \leq k \leq m).$$

On détermine ensuite des $\alpha_{3,1}$, ..., $\alpha_{3,n}$ tels que $(\mathcal{S}')$ ne contienne ni terme $y_1 y_3$, ni terme $y_2 y_3$. Cela revient à nouveau à résoudre un système linéaire, cette fois de $2m$ équations à n inconnues

$\alpha_{3,j}$ $(1 \leq j \leq n)$:

$$\begin{cases} \sum\limits_{1\leq i\leq j\leq n} a_{ijk}\alpha_{1,i}\alpha_{3,j} = 0 & (1 \leq k \leq m) \\ \sum\limits_{1\leq i\leq j\leq n} a_{ijk}\alpha_{2,i}\alpha_{3,j} = 0 & (1 \leq k \leq m) \end{cases}$$

On itère le processus, jusqu'à la dernière étape consistant à trouver $\alpha_{t,1}$, ..., $\alpha_{t,n}$ tels que $(\mathcal{S}')$ ne contienne aucun terme y_1y_t, ni y_2y_t, ..., ni $y_{t-1}y_t$. Cette dernière condition se traduit par un système linéaire de $(t-1)m$ équations à n inconnues $\alpha_{t,j}$ $(1 \leq j \leq n)$:

$$\begin{cases} \sum\limits_{1\leq i\leq j\leq n} a_{ijk}\alpha_{1,i}\alpha_{t,j} = 0 & (1 \leq k \leq m) \\ \quad\vdots \\ \sum\limits_{1\leq i\leq j\leq n} a_{ijk}\alpha_{t-1,i}\alpha_{t,j} = 0 & (1 \leq k \leq m) \end{cases}$$

En général, la réduction de Gauss fournit au moins une solution pour chacun de ces systèmes (en particulier pour le dernier) à condition d'avoir $n > m(t-1)$. Par ailleurs, les t vecteurs $\begin{pmatrix} \alpha_{1,1} \\ \vdots \\ \alpha_{1,n} \end{pmatrix}$, ..., $\begin{pmatrix} \alpha_{t,1} \\ \vdots \\ \alpha_{t,n} \end{pmatrix}$ ont une très forte probabilité d'être linéairement indépendants pour un système aléatoire $(\mathcal{S})$. Les constantes $\alpha_{i,j}$ restant à déterminer (*i.e.* celles correspondant à $t+1 \leq i \leq n$ et $1 \leq j \leq n$) sont choisies aléatoirement, de manière à obtenir un changement de variable *bijectif*.

En réécrivant le système $(\mathcal{S})$ par rapport à ces nouvelles variables y_i, on obtient :

$$(\mathcal{S}') \quad \begin{cases} \sum\limits_{i=1}^{t} \beta_{i,1}y_i^2 + \sum\limits_{i=1}^{t} y_iL_{i,1}(y_{t+1},...,y_n) + Q_1(y_{t+1},...,y_n) = 0 \\ \quad\vdots \\ \sum\limits_{i=1}^{t} \beta_{i,m}y_i^2 + \sum\limits_{i=1}^{t} y_iL_{i,m}(y_{t+1},...,y_n) + Q_m(y_{t+1},...,y_n) = 0 \end{cases}$$

où chaque $L_{i,j}$ est une fonction affine et chaque Q_i une fonction quadratique. On calcule alors y_{t+1}, ..., y_n tels que :

$$\forall i,\ 1 \leq i \leq t,\ \forall j,\ 1 \leq j \leq m,\ L_{i,j}(y_{t+1},...,y_n) = 0.$$

Cela est faisable parce qu'on se trouve face à un système linéaire de mt équations à $n-t$ inconnues, qui donne en général une solution à partir du moment où $n \geq (m+1)t$. On choisit l'une de ces solutions. Si on pose $\lambda_k = -Q_k(y_{t+1},...,y_n)$ $(1 \leq k \leq m)$, il reste alors à résoudre un système de m équations en les t variables y_1, ..., y_t :

$$(\mathcal{S}'') \quad \begin{cases} \sum\limits_{i=1}^{t} \beta_{i1}y_i^2 = \lambda_1 \\ \quad\vdots \\ \sum\limits_{i=1}^{t} \beta_{im}y_i^2 = \lambda_m \end{cases}$$

J'appelle «MQ^2 à t variables et m équations» le problème général de la résolution d'un tel système.

5.5 Un nouveau problème difficile ?

Lorsque le corps K est de caractéristique 2, et $n \geq m(m+1)$, il est facile de voir que l'algorithme décrit ci-dessus réussit (il suffit de choisir $t = m$). Le problème MQ^2 est facile à résoudre : on applique une réduction de Gauss pour trouver les y_i^2, puis on utilise le fait que la transformation $z \mapsto z^2$ est bijective sur tout corps de caractéristique 2.

Comme pour les algorithmes A, B, C, on en tire la conséquence qu'un algorithme de signature (par exemple UOV) ne doit pas être trop sous-défini, ce qui donne un nouveau critère pour le choix des paramètres pour les algorithmes de signature multivariables.

Toutefois, le raisonnement ne semble pas s'appliquer lorsque la caractéristique du corps K est *impaire*. En effet, pour chaque i, $1 \leq i \leq t$, la probabilité que la valeur trouvée pour y_i^2 soit effectivement un *résidu quadratique* dans K est égale à $\frac{1}{2}$. On obtient donc (toujours avec $t = m$) un algorithme de complexité 2^m, qui est meilleur que la recherche exhaustive en q^m, mais néanmoins exponentiel[19].

C'est pourquoi j'ai proposé MQ^2 comme nouveau problème difficile, sur lequel on peut bâtir de nouveaux cryptosystèmes (notamment un schéma d'authentification *zero-knowledge* non encore publié).

Problème $\mathbf{MQ}^2$: Soient deux entiers t et m tels que $t \geq 2m$. Soient β_{ij} $(1 \leq i \leq t, 1 \leq j \leq m)$ $t \times m$ éléments d'un corps K de caractéristique impaire, et λ_j $(1 \leq j \leq m)$ m éléments de ce même corps. Trouver au moins une solution $(y_1, \ldots, y_t) \in K^t$ au système

$$\begin{cases} \sum_{i=1}^{t} \beta_{i1} y_i^2 = \lambda_1 \\ \vdots \\ \sum_{i=1}^{t} \beta_{im} y_i^2 = \lambda_m \end{cases}$$

19. Notons que N. Courtois a établi une variante de ma méthode, où il montre [155] que le problème MQ sur un corps de caractéristique impair peut encore être résolu en temps polynomial (en n) à condition d'avoir $n \geq 2^{m/7}(m+1)$ et $\log_2 q > 4$. La complexité croît donc exponentiellement en m, mais relativement lentement.

Chapitre 5

Construction d'algorithmes dédiés

Current encryption technology is based on mathematics developed in the 17th and 18th centuries. Elliptic curve cryptography brings it forward into the mathematics of the 19th century. There is still a long way to go...

Whitfield Diffie (2002)

While it might appear that the evolution from RC5 to RC6 was straightforward, it in fact involved the design and analysis of literally dozens of alternatives. RC6 is the design that captures the spirit of our three goals of security, simplicity and performance the most effectively.

Ronald Rivest (1998)

Sommaire

1 Introduction 65
2 L'algorithme QUARTZ 66
2.1 Signatures courtes 66
2.2 Description de l'algorithme QUARTZ 67
2.3 Résolution d'une équation polynomiale dans un corps fini 69
2.4 Preuves relatives de sécurité 70
2.5 Difficulté du problème MQ pour QUARTZ 70
3 L'algorithme SFLASH 71
3.1 Un algorithme de signature rapide 71
3.2 Description de l'algorithme SFLASH 71
3.3 Preuves relatives de sécurité 72
3.4 Implémentation optimisée sur carte à microprocesseur 74
4 Bilan pour la signature sur carte à microprocesseur 76

1 Introduction

La phrase de Ronald Rivest ci-dessus, même si elle concerne un algorithme symétrique candidat pour l'AES (et qui a fait partie des cinq finalistes), montre bien le fossé qui existe entre un principe général de construction, et la spécification effective d'un algorithme répondant à des contraintes particulières. On pourrait dire la même chose du passage du principe de HFE à l'algorithme QUARTZ, ou bien du principe de C^*_- à l'algorithme SFLASH.

Dans ce chapitre, je présente la construction concrète d'algorithmes de signature multivariables, tels que SFLASH [81, 82] et QUARTZ [83, 84], qui ont été soumis au projet européen NESSIE. Les preuves partielles de sécurité s'appuient sur les hypothèses calculatoires du chapitre 3. Les cryptanalyses génériques présentées au chapitre 4 servent à guider le choix des paramètres permettant d'atteindre une sécurité pratique. Pour SFLASH, les contraintes propres aux cartes à microprocesseur, exposées dans le chapitre 2, ont rendu nécessaire la mise au point d'optimisations dans l'implémentation de ces algorithmes [173].

2 L'algorithme QUARTZ

2.1 Signatures courtes

Les signatures courtes ont leur importance dans certains environnements, où un être humain doit manipuler lui-même la signature. C'est le cas par exemple de certains logiciels fournis sur un CD-ROM : à l'installation du programme, l'utilisateur doit taper une signature sur le clavier pour s'enregistrer. Une autre application signalée dans [195, 198] est l'impression d'une signature sur un timbre-poste (ou une carte d'identité). Plus généralement, les signatures courtes sont utiles lorsque les canaux de transmission ont une faible bande passante.

Les signatures produites par les algorithmes de signature traditionnels sont relativement longues : 1024 bits ou plus pour le RSA, 320 bits pour DSA ou ECDSA. Notons que dans [195], D. Naccache et J. Stern ont proposé une variante de DSA qui réduit la taille de la signature à 240 bits[1]. Une autre technique classique consiste à produire des signatures avec recouvrement (partiel) du message : une partie du message est contenue dans la signature, ce qui réduit la taille du couple (message, signature). En pratique, pour de longs messages, l'accroissement de taille dû à la signature peut ainsi atteindre 160 bits pour l'algorithme DSA[2]. Néanmoins, pour des messages courts (par exemple 64 bits), la taille de la signature est toujours de 320 bits.

Plusieurs alternatives ont été proposées pour obtenir des signatures plus courtes. Ainsi D. Boneh, B. Lynn et H. Shacham ont décrit dans [176] un schéma qui engendre des signatures de l'ordre de 160 bits, quelle que soit la taille du message. Ce schéma est conçu pour des groupes dans lesquels le problème CDH (*Computational Diffie-Hellman*) est difficile alors que le problème de *décision* Diffie-Hellman (DDH) est facile[3]. Un autre exemple a été donné par N. Courtois, M. Finiasz et N. Sendrier, qui ont montré [184] que l'on pouvait utiliser le cryptosystème de McEliece (qui s'appuie sur le problème du *décodage de syndrome*) en mode signature, contrairement à ce qu'on croyait depuis longtemps. Pour une sécurité de 2^{83}, la taille des signatures obtenues peut aller jusqu'à 81 bits[4].

J'ai proposé en septembre 2000, avec N. Courtois et J. Patarin, l'algorithme QUARTZ [83, 84]. Il s'agit, à l'heure actuelle, de l'algorithme de signature réaliste qui produit les signatures les plus courtes : 128 bits (qui peuvent même être ramenés très facilement à 121 bits). L'idée principale est contenue dans l'algorithme HFE [104], mais la construction concrète a necessité une recherche plus approfondie dans deux directions. D'une part une étude détaillée de l'impact de chaque paramètre sur la sécurité de l'algorithme, afin de se protéger contre un certain nombre de méthodes génériques de cryptanalyse (voir chapitre 4). D'autre part la mise au point d'un

1. Dans [194], I. Mironov étudie une variante similaire dont il donne une preuve de sécurité concrète dans le modèle de l'oracle aléatoire.

2. En revanche, si on ne transmet que la signature elle-même, on a toujours 320 bits à transmettre

3. Le premier exemple de tels groupes a été donné par A. Joux et K. Nguyen dans [192].

4. Mais la taille de la clé publique est très élevée : de l'ordre d'1 Mo.

design adapté qui permette de mettre au point des preuves de sécurité relatives pour QUARTZ (voir paragraphe 2.3 ci-dessous).

2.2 Description de l'algorithme QUARTZ

QUARTZ est un cas particulier du schéma général HFEV$^-$ décrit dans notre article [137]. Les paramètres sont choisis de façon à obtenir des signatures de 128 bits, avec une sécurité en 2^{80} pour les meilleures attaques connues.

L'algorithme utilise l'extension algébrique $\mathcal{L} = \mathbb{F}_{2^{103}}$ du corps fini $\mathbb{F}_2$. Plus précisément $\mathcal{L} = \mathbb{F}_2[X]/(X^{103} + X^9 + 1)$, ce qui permet d'identifier $\mathcal{L}$ à $(\mathbb{F}_2)^{103}$.

La clé secrète : Elle est constituée de :

- Deux bijections affines $s : (\mathbb{F}_2)^{107} \to (\mathbb{F}_2)^{107}$ et $t : (\mathbb{F}_2)^{103} \to (\mathbb{F}_2)^{103}$ (que l'on peut donner par leur représentations matricielles).
- Une famille (F_V) de fonctions secrètes. À chaque $V \in (\mathbb{F}_2)^4$ correspond une fonction $F_V : \mathcal{L} \to \mathcal{L}$ de la forme

$$F_V(Z) = \sum_{\substack{0 \le i < j < 103 \\ 2^i + 2^j \le 129}} \alpha_{i,j} \cdot Z^{2^i + 2^j} + \sum_{\substack{0 \le i < 103 \\ 2^i \le 129}} \beta_i(V) \cdot Z^{2^i} + \gamma(V).$$

  Dans cette formule, chaque $\alpha_{i,j}$ appartient à $\mathcal{L}$, chaque β_i est une transformation affine de $(\mathbb{F}_2)^4$ dans $\mathcal{L}$, et γ est une transformation quadratique de $(\mathbb{F}_2)^4$ dans $\mathcal{L}$.
- Une valeur secrète Δ de 80 bits.

La clé publique : C'est la fonction $G : (\mathbb{F}_2)^{107} \to (\mathbb{F}_2)^{100}$ définie par

$$G(X) = \Big[t\Big(F_{[s(X)]_{103 \to 106}}([s(X)]_{0 \to 102}) \Big) \Big]_{0 \to 99}.$$

Par construction, G est une transformation quadratique sur $\mathbb{F}_2$.

$$(Y_0, \ldots, Y_{99}) = G(X_0, \ldots, X_{106}) \Leftrightarrow \begin{cases} Y_0 = P_0(X_0, \ldots, X_{106}) \\ \quad \vdots \\ Y_{99} = P_{99}(X_0, \ldots, X_{106}) \end{cases}$$

où chaque polynôme P_i est quadratique sur $\mathbb{F}_2$.

Calcul d'une signature : Il est résumé [5] dans la figure 1, où l'on a $Y \in (\mathbb{F}_2)^{100}$, $R \in (\mathbb{F}_2)^3$, $V \in (\mathbb{F}_2)^4$, $B \in \mathcal{L}$, $A \in \mathcal{L}$, $X \in (\mathbb{F}_2)^{107}$, $X_i \in (\mathbb{F}_2)^7$ et $\tilde{S} \in (\mathbb{F}_2)^{100}$.

Vérification d'une signature : Elle est résumée [6] dans la figure 2.

5. Voir [84] pour une spécification détaillée du calcul d'une signature par QUARTZ.
6. Voir [84] pour une spécification détaillée de la vérification d'une signature par QUARTZ.

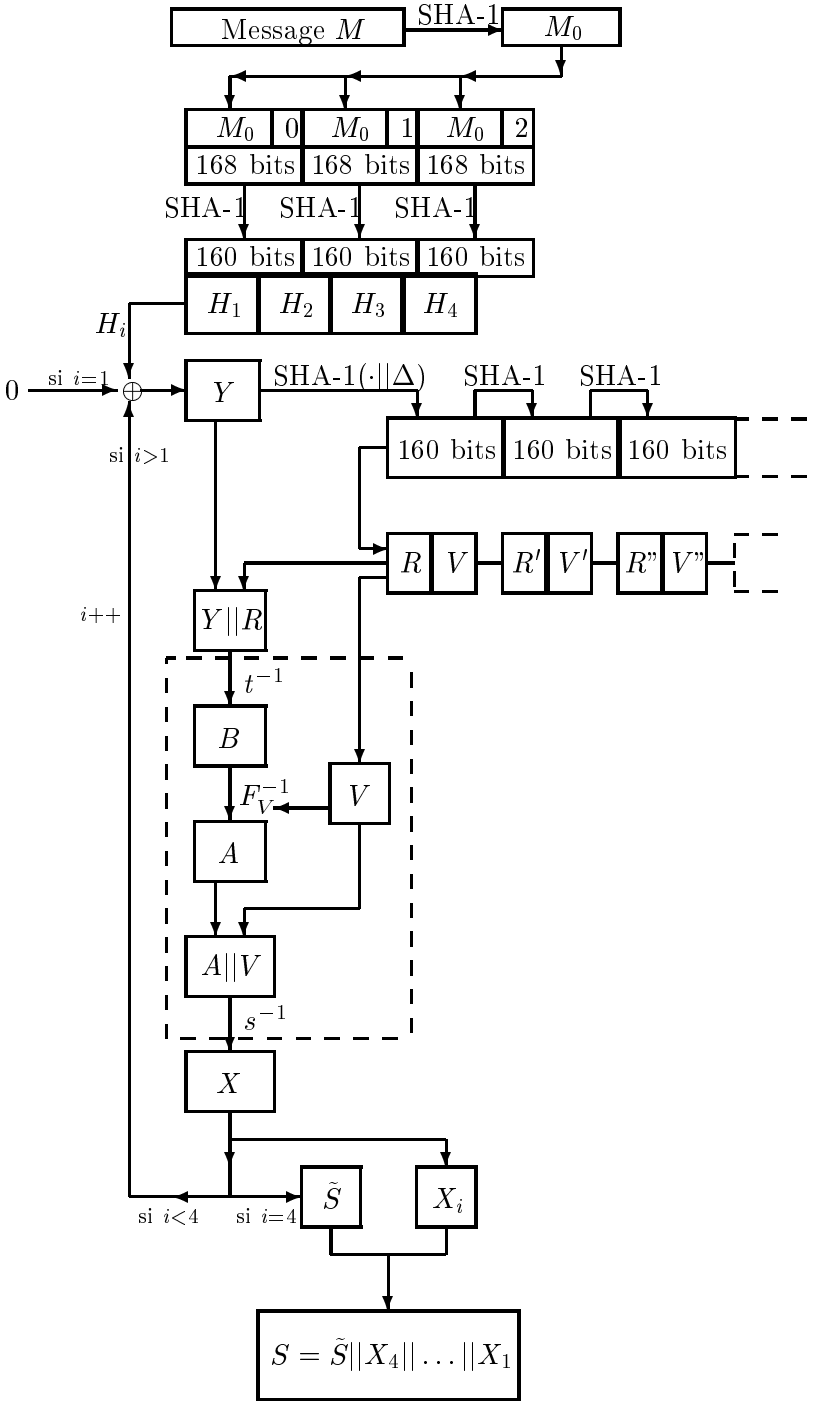


FIG. 1 – *Calcul d'une signature avec l'algorithme QUARTZ (au début $i = 1$)*

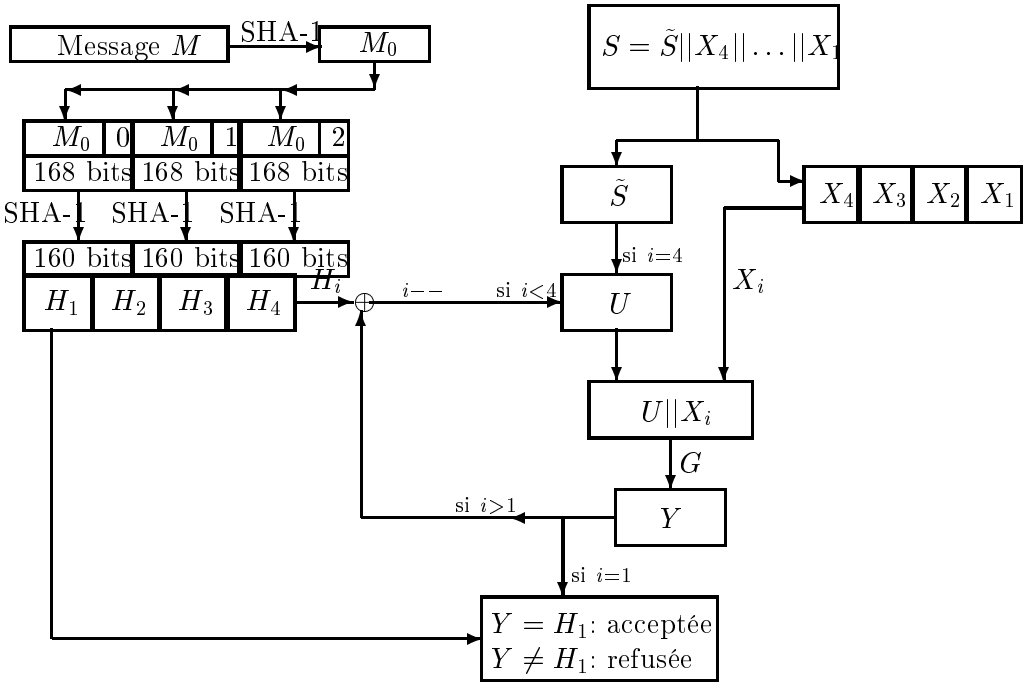


FIG. 2 – *Vérification d'une signature avec l'algorithme QUARTZ (au début $i = 4$)*

2.3 Résolution d'une équation polynomiale dans un corps fini

Pour signer un message avec l'algorithme QUARTZ, on a besoin de résoudre une équation de la forme $F_V(Z) = B$, où B est un élément de $\mathcal{L}$ et Z est l'inconnue. Dans QUARTZ, la résolution est faite de telle sorte que s'il y a plusieurs solutions, on en choisit une de manière déterministe, mais dépendant de la valeur Δ (qui joue le rôle de la graine d'un générateur pseudo-aléatoire).

En ce qui concerne l'implémentation, la première étape pour trouver les racines consiste à calculer

$$\Psi(Z) = \mathrm{PGCD}\Big(F_V(Z) - B, Z^{2^{103}} - Z\Big).$$

Pour obtenir ce PGCD, on peut calculer de façon récursive $Z^{2^i} \mod (F_V(Z) - B)$ pour $i = 0$, 1, ..., 103 puis calculer $\Theta(Z) = Z^{2^{103}} - Z \mod (F_V(Z) - B)$. Finalement, $\Psi(Z)$ est obtenu par

$$\Psi(Z) = \gcd\Big(F_V(Z) - B, \Theta(Z)\Big).$$

Avec cette méthode[7], le degré des polynômes apparaissant au cours du calcul ne dépasse jamais $2 \times 129 = 258$.

Le nombre de solutions de $F_V(Z) = B$ dans $\mathcal{L}$ est égal au degré de Ψ sur $\mathcal{L}$. On a alors trois possibilités :

- Si le degré de Ψ vaut 0, il n'y a pas de solution et on recommence avec de nouvelles valeurs pour V et B, comme spécifié dans [84].
- Si le degré de Ψ vaut 1, c'est-à-dire que Ψ est de la forme $\Psi(Z) = \kappa \cdot (Z - A)$ (avec $\kappa \in \mathcal{L}$), alors A est l'unique solution de l'équation $F_V(Z) = B$.

7. Des méthodes plus fines existent pour calculer $\Psi(Z)$, notamment celle de Kaltofen et Shoup [193].

– Si le degré de Ψ est ≥ 2, on applique par exemple l'algorithme de Berlekamp-Rabin pour déterminer toutes les racines, puis on en choisit une de façon déterministe.

2.4 Preuves relatives de sécurité

L'algorithme QUARTZ applique le paradigme CPC [54] (*Chained Patarin Construction*) (avec $t = 4$ tours) au schéma général HFEV$^-$ de [137]. Cette construction permet d'éviter le *paradoxe des anniversaires*.

En effet, dans tout schéma de signature pour lequel la vérification d'une signature S du message M consiste à vérifier une équation $H(S) = G(M)$, où H et M sont deux fonctions publiques, on peut en général trouver un couple (M,S) valide, en temps et mémoire de l'ordre de $2^{n/2}$ (où n est la taille des sorties de H et G). Cette attaque de *forge existentielle* est une application immédiate du paradoxe des anniversaires[8].

Or, dans QUARTZ, la vérification d'une signature S pour un message M consiste à vérifier une équation du type $\varphi(S,M) = 0$, où φ est une fonction publique. Cela rend impossible l'application directe du paradoxe des anniversaire. Une attaque généralisée est possible, mais sa complexité devient $\mathcal{O}(2^{\frac{tn}{t+1}})$ (où r est le nombre de tours de CPC). Pour QUARTZ (avec $n = 100$ et $t = 4$), cela donne donc une attaque de forge existentielle en 2^{80}, ce qui reste compatible avec les contraintes de sécurité actuelles (notamment celles de NESSIE). En outre, N. Courtois [182] a montré que, si on se place dans le modèle des *attaques à message connu*, cette attaque générique est la meilleure possible contre la construction CPC.

Il est à noter qu'un rapport (non publié) d'A. Joux et G. Martinet [191] se place dans un scénario différent : en faisant 2^{50} appels à l'oracle de signature et 2^{50} calculs de la fonction G, ils montrent qu'un attaquant peut trouver une seconde signature valide d'un message M qui a déjà été signé. Cela montre que le schéma QUARTZ ne vérifie pas la propriété de *non-forgeabilité forte*[9] (appelée également *non-malléabilité*). Néanmoins, comme nous l'avons mentionné dans [83], ce n'est pas véritablement gênant en pratique, car il faut *déjà* connaître la clé secrète pour pouvoir mettre en œuvre l'attaque[10].

Un autre type de scénario consiste à faire signer plusieurs fois le même message par l'oracle de signature. Dans le cas de l'algorithme ESIGN [88], cette idée a été appliquée avec succès par D. Pointcheval et J. Stern pour monter une attaque à messages choisis [199]. Dans le cas de QUARTZ (et aussi de SFLASH), ce type d'attaque a été intentionnellement rendu impossible en rendant *déterministe* (mais dépendant d'une valeur secrète Δ, qui joue le rôle de la graine d'un générateur pseudo-aléatoire) le calcul de la signature d'un message M. Cette idée a été reprise par L. Granboulan [190] pour réparer le schéma ESIGN. La technique a pris le nom de FDH-D (*Full-Domain Hash Deterministic*) [54].

2.5 Difficulté du problème MQ pour QUARTZ

Les preuves relatives de sécurité décrites ci-dessus ont pour objet de montrer que la sécurité globale du schéma QUARTZ se ramène à la difficulté du problème MQ sous-jacent, c'est-à-dire

8. Il suffit de stocker $\approx 2^{n/2}$ valeurs $G(M)$ et $\approx 2^{n/2}$ valeurs de $H(S)$, puis de trouver une collision.

9. Introduite par E. Brickell, D. Pointcheval, S. Vaudenay et M. Yung dans [179] (voir aussi [199]).

10. Le rapport de sécurité de NESSIE [54] va dans le même sens : «While the existence of duplicate signatures may be surprising, it is not a weakness of a digital signature scheme. If the scheme is existentially unforgeable, it proves that both signed messages were produced by a secret key holder» (page 214).

à la difficulté de trouver d'inverser la fonction G qui constitue la clé publique :

$$(Y_0,\ldots,Y_{99}) = G(X_0,\ldots,X_{106}) \Leftrightarrow \begin{cases} Y_0 = P_0(X_0,\ldots,X_{106}) \\ \quad \vdots \\ Y_{99} = P_{99}(X_0,\ldots,X_{106}) \end{cases}$$

En se plaçant dans le cas limite, si le degré des polynômes F_V n'était pas borné, le système quadratique à résoudre deviendrait aléatoire. Néanmoins, le schéma ne serait plus utilisable en pratique, car la résolution des équations du type $F_V(Z) = B$ (voir paragraphe 2.3) ne peut plus s'effectuer en temps polynomial. C'est pourquoi le degré des polynômes F_V a été choisi égal à $d = 129$ dans QUARTZ.

Des attaques expérimentales récentes [185] tentent de résoudre directement le problème MQ pour QUARTZ. La dépendance en d de la difficulté de ce problème n'est pas encore complètement comprise, mais les analyses faites par J-C. Faugère et A. Joux [187], ainsi que les évaluations de N. Courtois, M. Daum et P. Felke [183] suggèrent de modifier QUARTZ en prenant $d = 257$, afin de conserver une sécurité de 2^{80}.

3 L'algorithme SFLASH

3.1 Un algorithme de signature rapide

Si l'algorithme QUARTZ produit des signatures très courtes (128 bits), ses performances ne sont pas spectaculaires (il faut de l'ordre de 10 secondes pour calculer une signature sur un Pentium II à 500 MHz). Il n'est même pas envisageable de l'utiliser dans une carte à microprocesseur, sauf si la carte ne fait que vérifier des signatures. C'est pourquoi, j'ai proposé en septembre 2000, avec N. Courtois et J. Patarin, l'algorithme SFLASH [81, 82]. Même s'il est nettement plus rapide, les premières implémentations sur carte à puce calculaient une signature SFLASH en près de 12 secondes [55] (sur un processeur de type 8051 cadencé à 10 MHz). J'ai donc été amené, avec M.-L. Akkar, N. Courtois et R. Duteuil à mettre au point une implémentation extrêmement optimisée sur carte à microprocesseur [173].

3.2 Description de l'algorithme SFLASH

SFLASH est un cas particulier du schéma général C^*_- décrit dans [169]. Les paramètres sont choisis de façon à obtenir des signatures très rapides sur carte à microprocesseur, avec une sécurité en 2^{80} pour les meilleures attaques connues.

L'algorithme utilise l'extension algébrique $\mathcal{L} = \mathbb{F}_{128^{37}}$ du corps fini $K = \mathbb{F}_{128}$. Plus précisément $\mathcal{L} = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$, ce qui permet d'identifier $\mathcal{L}$ à K^{37}.

La clé secrète : Elle est constituée de :

- Deux bijections affines $s : K^{37} \to K^{37}$ et $t : K^{37} \to K^{37}$ (que l'on peut donner par leur représentations matricielles)[11].
- Une valeur secrète Δ de 80 bits.

11. En fait, la partie constante de s et t peut être retrouvée facilement comme l'ont montré Geiselmann, Steinwandt et Beth [188]. Elle doit donc faire partie des paramètres du système, et non de la clé secrète.

La clé publique : C'est la fonction $G : K^{37} \to K^{26}$ définie par

$$G(X) = \Big[t\Big(F(s(X))\Big)\Big]_{0\to 25}.$$

Ici F est la fonction monomiale de $\mathcal{L}$ dans $\mathcal{L}$ définie par

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{11}+1}.$$

Par construction, G est une transformation quadratique sur K.

$$(Y_0,\ldots,Y_{25}) = G(X_0,\ldots,X_{36}) \Leftrightarrow \begin{cases} Y_0 = P_0(X_0,\ldots,X_{36}) \\ \quad\vdots \\ Y_{25} = P_{25}(X_0,\ldots,X_{36}) \end{cases}$$

où chaque polynôme P_i est quadratique sur K.

Calcul d'une signature : Il est résumé[12] dans la figure 3, où l'on a $Y \in K^{26}$, $R \in K^{11}$, $B \in \mathcal{L}$, $A \in \mathcal{L}$ et $S \in K^{37}$.

Vérification d'une signature : Elle est résumée[13] dans la figure 4.

3.3 Preuves relatives de sécurité

Comme SFLASH est un cas particulier du schéma général C^*_- [169], les attaques que nous avons décrites au chapitre 4 (voir le paragraphe 2.5) sont potentiellement applicables. Elles ont une complexité en $\mathcal{O}(q^r)$ (où r est le nombre de polynômes non divulgués dans la clé publique G, et q est le cardinal du corps K), ce qui fait que nous avons choisi $r = 37 - 26 = 11$, afin d'assurer une sécurité pratique[14] de 2^{80} avec $K = \mathbb{F}_{128}$ comme corps de base[15].

SFLASH est bâti sur le même principe FDH-D (*Full-Domain Hash Deterministic*) [54] que QUARTZ : la calcul de la signature d'un message M est *déterministe*, mais dépend d'une valeur secrète Δ, qui joue le rôle de la graine d'un générateur pseudo-aléatoire. Cela permet notamment d'éviter les attaques consistant à faire signer plusieurs fois le même message par l'oracle de signature [199].

Comme dans le cas de QUARTZ, on peut montrer [182] qu'il n'y a pas d'attaque générique de forge existentielle meilleure que 2^{80} sur le schéma FDH-H utilisé dans SFLASH (si on se place dans le modèle des *attaques à message connu*).

Il semble donc que la seule voie d'attaque consiste à tenter de résoudre directement le problème MQ sous-jacent, c'est-à-dire d'inverser la fonction G qui constitue la clé publique :

$$(Y_0,\ldots,Y_{25}) = G(X_0,\ldots,X_{36}) \Leftrightarrow \begin{cases} Y_0 = P_0(X_0,\ldots,X_{36}) \\ \quad\vdots \\ Y_{25} = P_{25}(X_0,\ldots,X_{36}) \end{cases}$$

Les paramètres de SFLASH ont été choisis de manière à résister aux algorithmes classiques de Buchberger pour les bases de Gröbner [180, 181], à XL ou FXL [156], ainsi qu'à des versions plus sophistiquées dues à J-C. Faugère : F_5 et $F_5/2$ [186].

12. Voir [82] pour une spécification détaillée du calcul d'une signature par SFLASH.
13. Voir [82] pour une spécification détaillée de la vérification d'une signature par SFLASH.
14. la constante devant q^r est ≥ 8 dans la complexité des attaques sur C^*_-.
15. Notons qu'une première version de SFLASH recommandait de prendre les éléments de s et t dans le sous-corps $\mathbb{F}_2$ de K, ceci afin de réduire la taille de la clé publique. Mais H. Gilbert et M. Minier ont montré [189] que cela menait à une attaque dévastatrice.

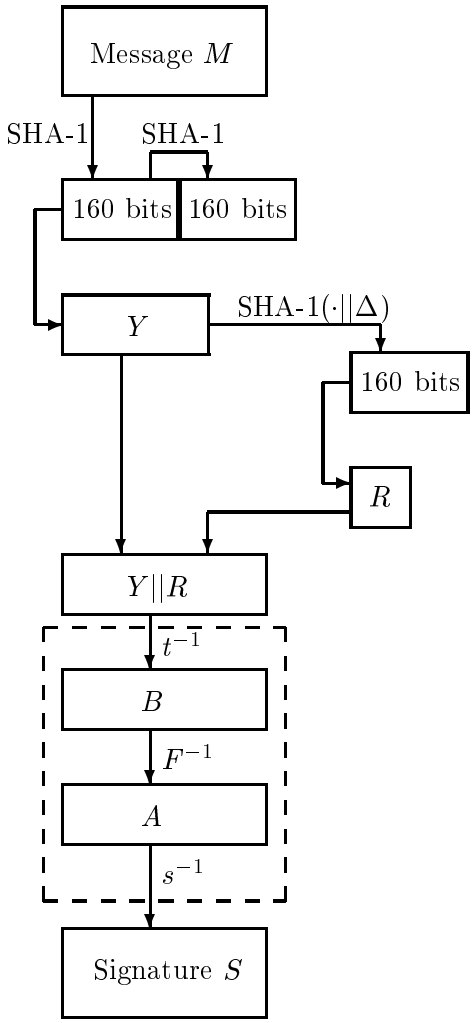


FIG. 3 – *Calcul d'une signature avec l'algorithme SFLASH*

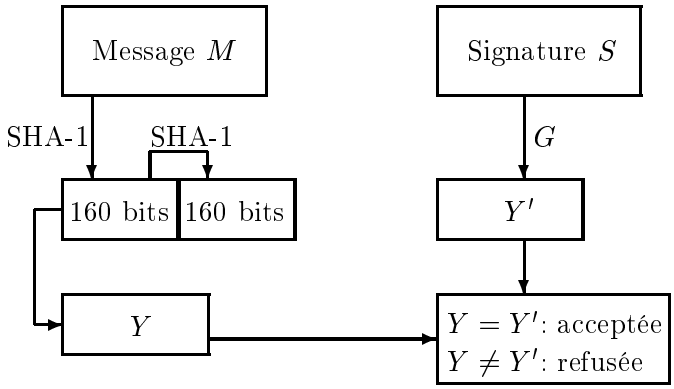


FIG. 4 – *Vérification d'une signature avec l'algorithme SFLASH*

3.4 Implémentation optimisée sur carte à microprocesseur

L'implémentation optimisée [173] que j'ai mise au point, avec M.-L. Akkar, N. Courtois et R. Duteuil, s'appuie sur un cœur de microprocesseur 8 bits Intel 8051.

Dans la RAM, chaque élément de $K = \mathbb{F}_{128}$ est stocké dans un octet. Chaque élément de $\mathcal{L} = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$ est représenté comme un tableau de 37 octets. Dans l'E^{2}PROM sont stockés s^{-1}, t^{-1} et Δ.

Opérations sur le corps $K = \mathbb{F}_{128}$:

1. Addition : comme K est un corps de caractéristique 2, la somme de deux éléments de K est obtenue par un *ou-exclusif* des octets qui les représentent.
2. Multiplication : comme le groupe multiplicatif K^* est cyclique (de générateur α), on peut ramener les multiplications à des additions au moyen de deux tables stockées en ROM : l'une pour les logarithmes en base α, l'autre pour les puissances de α. De plus la table d'exponentielle est stockée en double (pour éviter la réduction modulo 128 de la somme des logarithmes) et on attribue à la valeur *zéro* un «logarithme» artificiel, ce qui évite d'avoir à programmer un traitement spécial lorsqu'un des facteurs est nul.

Opérations sur le corps $\mathcal{L} = \mathbb{F}_{128^{37}}$:

1. Addition : étant donnés deux éléments de $\mathcal{L}$ représentés par deux tableaux de 37 octets, leur somme est appliquant un *ou-exclusif* aux octets pris deux à deux.
2. Multiplication : c'est l'opération la plus coûteuse. Notons que le polynôme irréductible sur K qui sert à définir $\mathcal{L}$ est un pentanôme (on peut vérifier qu'il n'existe pas de trinôme). Le produit de $a = (a_{36}, \ldots, a_0)$ par $b = (b_{36}, \ldots, b_0)$ s'effectue de la façon suivante :
   - Calculer $c' = \sum_{i=0}^{72} c'_i X^i$, où $c_i = \sum_{l+k=i} a_l \cdot b_k$.
   - Réduire c' modulo $X^{37} + X^{12} + X^{10} + X^2 + 1$. Le résultat c est le produit $a \cdot b$ dans $\mathcal{L}$.
3. Carré : le carré d'un élément a de $\mathcal{L}$ peut être calculé environ 5 fois plus rapidement que si on multipliait a par lui-même. En effet, le corps étant de caractéristique 2, il suffit de calculer $a' = \sum_{i=0}^{36} a_i^2 X^{2i}$ puis de réduire a' modulo $X^{37} + X^{12} + X^{10} + X^2 + 1$.

Calcul des transformations affines t^{-1} et s^{-1} : On utilise la multiplication matricielle classique, avec l'astuce suivante : comme chaque élément de la matrice intervient uniquement dans des multiplications sur K, on suppose que c'est son logarithme en base α (au lieu de sa véritable valeur) qui a été stocké dans l'E^{2}PROM.

Calcul de $A = F^{-1}(B)$: On veut calculer $A = B^h$ dans $\mathcal{L}$, avec

$$\begin{aligned} h &= (128^{11} + 1)^{-1} \mod (128^{37} - 1) \\ &= 1000000\ 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000 \\ &\quad 1000000\ 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000 \\ &\quad 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000 \\ &\quad 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000\ 1000000 \\ &\quad 1000000\ 0111111\ 0111111\ 0111111\ 1000000 \end{aligned}$$

Le coût de l'algorithme classique *square and multiply* serait au moins de 259 carrés et 145 multiplications dans $\mathcal{L}$. Sachant que notre meilleure implémentation de la multiplication dans $\mathcal{L}$

utilise 10000 cycles CPU (*i.e.* 2,4 ms à 10 MHz sur une carte à microprocesseur), cela n'est pas satisfaisant.

Les q-chaînes d'addition

Sur un corps $K = \mathbb{F}_q$, l'élévation à la puissance q est essentiellement gratuite, et l'optimisation consiste à réduire le nombre des (autres) multiplications. On peut modéliser ce problème à l'aide de la notion de q-chaîne d'addition, due à von zur Gathen [202].

Définition : Soit q un entier ≥ 2. Une q-chaîne d'addition est une suite

$$\gamma = ((j(1),k(1)),...,(j(\ell),k(\ell)))$$

de couples d'entiers (ou bien $-\infty$), avec $0 \leq j(i) < i$ et $k(i) = -\infty$, ou bien $0 \leq k(i) \leq j(i) < i$ pour tout $1 \leq i \leq \ell$. Le nombre ℓ de couples est la *longueur* $L(\gamma)$ de γ. On définit la *sémantique* $S(\gamma) = \{a_0,...,a_\ell\}$ par $a_0 = 1$ et

$$a_i = \begin{cases} a_{j(i)} + a_{k(i)} & \text{si } k(i) \neq -\infty \\ a_i = q \cdot a_{j(i)} & \text{si } k(i) = -\infty \end{cases}$$

pour $1 \leq i \leq \ell$. On dit que γ *calcule* les éléments de $S(\gamma)$.

Il s'agit d'une généralisation des chaînes d'addition ordinaires. On peut montrer[16] que la longueur d'une chaîne d'addition ordinaire calculant e est toujours au moins $\lfloor \log_2 e \rfloor$. Pour une q-chaîne d'addition, on peut descendre jusqu'à $c \cdot \frac{\log_2 e}{\log_2 \log_2 e}$ pour une certaine constante $c > 0$. On définit

$$\ell_q(e) = \min\Big\{ L(\gamma),\ \gamma \text{ est une } q\text{-chaîne d'addition calculant } e \Big\}$$

la longueur de la q-chaîne d'addition la plus courte calculant e, avec $\ell_q(1) = 0$. Notons que $\ell_2(e)$ correspond aux chaînes d'addition ordinaires et s'appelle parfois la *complexité additive* de e.

Parmi les algorithmes classiques pour trouver une q-chaîne d'addition la plus courte possible, on trouve la méthode q^r-aire de Brauer [177], formalisée par Knuth [99] et généralisée par von zur Gathen [202], et la méthode de Yao [204], modifiée par Brickell, Gordon, McCurley, Wilson [178] puis par Agnew, Mullin, Vanstone [172] et Stinson [200]. Un algorithme efficace a également été proposé par Bleichenbacher dans sa thèse [174]. Par ailleurs, le calcul de q-chaînes d'addition a tiré parti du concept de *compression de données*, notamment par Yacobi [203], qui a adapté l'algorithme de compression de Ziv et Lempel [206, 205], et par Bocharova et Kudryashov [175], qui ont adapté l'algorithme de Tunstall [201].

Dans le cas de la fonction monomiale $A = F^{-1}(B)$ de SFLASH, une recherche exhaustive est hors de portée. Néanmoins, j'ai mis en évidence [173] la q-chaîne d'addition (décrite dans l'algorithme 1), qui présente l'avantage de ne nécessiter que 12 multiplications (au lieu de 145 pour la méthode *square and multiply*).

Les opérations $x \mapsto x^{128}$ et $x \mapsto x^{128^7}$ sur $\mathcal{L} = \mathbb{F}_{128^{37}}$

Ces deux opérations sont K-linéaires sur $\mathcal{L}$ et peuvent donc être réalisées par un simple calcul matriciel. En outre, comme le polynôme irréductible $(X^{37} + X^{12} + X^{10} + X^2 + 1)$ qui définit $\mathcal{L}$ n'a que des coefficients 0 et 1, les matrices à utiliser sont elles aussi constituées uniquement de 0 et de 1 : la multiplication matricielle se ramène donc à de simples additions (*i.e* des ou-exclusifs).

16. Voir par exemple [99], section 4.6.3.

Algorithme 1 : q-chaîne d'addition pour SFLASH

Input: B

Output: $A = B^h$ avec $h = (128^{11} + 1)^{-1} \bmod (128^{37} - 1)$

$\alpha := (B^2)^2$;
$\beta := B \times \alpha$;
$\gamma := (\alpha^2)^2$;
$\delta := \beta \times \gamma$;
$u := \delta^2 \times \delta$;
$v := (\gamma^2)^2$;
$t := ((v^{128})^{128})^{128} \times u$;
$w := ((t^{128} \times t)^{128} \times t)^{128}$;
$x := ((w \times u)^{128})^{128^7}$;
$z := v^{128^7} \times w \times v \times x$;
$A := (((z^{128^7})^{128^7})^{128} \times x)^{128^7} \times z$.
Return A

Pour $x \mapsto x^{128^7}$, le fait que la matrice soit fixe permet d'accélérer encore le calcul. On cherche ici à trouver un chemin optimal dans la matrice, afin de minimiser le nombre d'additions (ou-exclusifs) à effectuer. L'idée est d'utiliser un principe analogue à celui du *code de Gray* [196], qui donne une manière d'ordonner tous les mots de n bits de telle sorte que deux mots consécutifs de diffèrent que d'un bit. Cela permet de calculer toutes les combinaisons linéaires sur $\mathbb{F}_2$ d'un ensemble de n vecteurs, avec un seul ou-exclusif par combinaison (au lieu de $n/2$ en moyenne par une méthode classique). Dans [173], nous avons mis au point une solution spécifique encore plus efficace. Les matrices de taille 37×37 utilisées sont divisées en sous-matrices $37 \times k$, puis une recherche exhaustive est effectuée pour chaque sous-matrice. Le choix $k = 7$ permet ainsi d'accélérer d'un facteur 40 le calcul de $x \mapsto x^{128^7}$, pour une taille de code (ROM) de 700 octets environ.

Pour $x \mapsto x^{128}$, les techniques matricielles ne sont pas nécessaires : effectuer 7 carrés successifs est plus rapide. De plus on peut ainsi réutiliser des parties du code déjà écrites, ce qui réduit la taille ROM nécessaire. Enfin, une astuce permet encore de gagner du temps. On effectue en effet en principe – pour chacun des 7 carrés – deux opérations : la mise au carré «en place» des 37 octets représentant x, puis la transformation linéaire comprenant l'expansion et la réduction modulo $X^{37} + X^{12} + X^{10} + X^2 + 1$. Or il est facile de voir que ces deux opérations commutent. On peut donc reporter tous les carrés à la fin, et on s'aperçoit alors qu'ils ne sont plus nécessaires (à cause de l'identité $a^{128} = a$ dans $\mathbb{F}_{128}$).

4 Bilan pour la signature sur carte à microprocesseur

En utilisant les méthodes ci-dessus, deux implémentations extrêmement performantes de SFLASH ont été obtenues [173].

Une première implémentation utilise la méthode traditionnelle des carrés pour les calcul des puissances 128^7. Elle est rapide et utilise assez peu de mémoire ROM :

– Mémoire RAM: 334 octets (112 octets à accès direct et 222 octets à accès indirect).

Processeur	8051	8051	SLE66	SLE66	8051	8051	SLE66	SLE66
Fréquence [MHz]	3,57	10	3,57	10	3,57	10	3,57	10
Version du code	v2	v2	v2	v2	v1	v1	v1	v1
Taille ROM [Ko]	3,1	3,1	3,1	3,1	2,5	2,5	2,5	2,5
Temps de calcul [ms]	750	268	164	**59**	1 075	384	230	82

FIG. 5 – *Génération d'une signature SFLASH (hors calcul de SHA-1)*

Cryptosystème	SFLASH	NTRU-251	RSA-1024	RSA-1024	ECDSA-191
Plate-forme	SLE-66	Philips 8051	SLE-66	ST-19XL	SLE-66
Mots élémentaires	8 bits	8 bits	8 bits	8 bits	8 bits
Taille de la ROM	3,1	5	?	?	?
Fréquence [MHz]	10	16	10	10	10
Crypto-processeur	non	non	non	oui	oui
Taille de la signature	259	1 757	1 024	1 024	382
Temps de calcul	**59 ms**	160 ms	plusieurs s.	111 ms	180 ms

FIG. 6 – *Comparaison des performances d'algorithmes courants de signature*

– Taille du code (ROM) : 2,5 Ko.

Une seconde impléntation utilise la technique apparentée au code de Gray pour le calcul des puissances 128^7. Elle est encore plus rapide, mais nécessite un peu plus de ROM :

– Mémoire RAM: 334 octets (112 octets à accès direct et 222 octets à accès indirect).

– Taille du code (ROM) : 3,1 Ko.

Les performances de ces deux solutions, implémentées respectivement sur le processeur Intel 8051 d'origine et sur la version SLE66 déclinée par Infineon, sont indiquées dans la figure 5.

On voit ainsi qu'une carte à microprocesseur «ordinaire» (notamment sans microprocesseur), cadencée à la fréquence usuelle de 10 MHz, est capable avec l'algorithme SFLASH de calculer une signature en 60 millisecondes environ. Cela fait de SFLASH une alternative sérieuse aux algorithmes RSA [67] ou ECDSA [114, 36] (dont les temps de calcul, même en présence d'un crypto-processeur, sont largement supérieurs), en particulier dans le cas où la carte n'a pas à transmettre sa clé publique au lecteur.

À titre de comparaison, un tableau des performances de cryptosystèmes classiques de signature pour les cartes à microprocesseur est donnée dans la figure 6. Le seul algorithme dont les performances soient comparables à SFLASH semble être NTRUSign [94]. Celui-ci est un peu plus lent que SFLASH. De plus, les signatures produites sont 6 fois plus longues, ce qui peut constituer un autre handicap[17].

17. Pour une vitesse de transmission sur le port série de 9 600 bits/s, il faut ajouter 200 ms pour transmettre la signature NTRUSign au terminal. En revanche la clé publique est plus petite que celle de SFLASH.

Chapitre 6

Attaques physiques

The vast majority of security failures occur at the level of implementation detail.

Ross Anderson (1993)

The great practicality and the inherent availability of physical attacks threaten the very relevance of complexity-theoretic security. Why erect majestic walls if comfortable underpasses will always remain wide open ?

Silvio Micali et Leonid Reyzin (2003)

Sommaire

1 Introduction 79
2 Classification des attaques physiques 80
2.1 Attaques invasives ou non-invasives 80
2.2 Attaques actives ou passives 81
3 Attaques par injection de fautes 82
4 Attaques par analyse de consommation électrique 83
4.1 Analyse élémentaire de la consommation 84
4.2 Analyse différentielle de la consommation 84
4.3 Exemple de l'algorithme DES 85
5 Représentations booléennes et arithmétiques 86
5.1 Le problème de conversion 86
5.2 La méthode de Messerges 86
5.3 Attaque par analyse de consommation 87
6 Attaques physiques sur les courbes elliptiques 88
6.1 État de l'art des protections 88
6.2 Une attaque à message choisi 89
6.3 Application à des courbes elliptiques standards 91

1 Introduction

Dans la conception traditionnelle de la *cryptologie* (voir chapitre 1), la sécurité est vue de manière abstraite : pour attaquer un cryptosystème, l'attaquant se borne à échanger des messages avec celui-ci, et espère en les utilisant pouvoir mettre en défaut les objectifs visés (confidentialité, intégrité, authenticité, ...) par différentes techniques de *cryptanalyse* (voir chapitre 4). La

cryptographie classique s'efforce donc de construire des schémas (voir chapitre 5) avec si possible des preuves relatives de sécurité contre ce type d'attaque, en admettant la difficulté de certains problèmes mathématiques au sens de la *théorie de la complexité* (voir chapitre 3).

Dans un modèle de sécurité plus étendu, on tient compte également, depuis quelques années, des *attaques physiques*. Ce nouveau concept prend en considération non seulement la sécurité des cryptosystèmes au sens *mathématique*, mais aussi les aspects liés à la nature *physique* des calculs. Ces attaques nouvelles sont particulièrement menaçantes pour les *systèmes embarqués* tels que les *cartes à microprocesseur* (voir chapitre 2), contre lesquels l'adversaire peut mobiliser des moyens d'analyse de plus en plus sophistiqués.

Dans ce domaine, j'ai effectué des travaux soit pour mettre en évidence de nouvelles stratégies d'attaques physiques [313, 237, 244] (voir le présent chapitre), soit pour proposer des *contre-mesures* [313, 311, 302, 173] avec dans certains cas des *preuves de sécurité* en établissant un modèle convenable de l'adversaire (voir chapitre 7).

Dans tout ce chapitre, les attaques physiques sont considérées en tant que menaces sur la sécurité des protocoles et algorithmes cryptographiques dans les cartes à microprocesseur. Néanmoins, tous les systèmes embarqués utilisant la cryptographie en sont aussi potentiellement la cible [1].

2 Classification des attaques physiques

Afin d'évaluer le niveau de résistance de ses produits, IBM a proposé en 1991 une taxonomie des attaquants potentiels [207] :

- Les «amateurs éclairés» (classe I) : ils sont souvent très intelligents, mais ont une connaissance imparfaite du système. Ils ont accès uniquement à du matériel de sophistication moyenne. Ils essaient souvent de tirer avantage de faiblesses existantes du système, plutôt que d'en créer de nouvelles.
- Les «attaquants experts» (classe II) : ils ont reçu une solide formation technique et ont de l'expérience. Ils ont une compréhension variable des parties du système, mais ont un accès potentiel à toutes ces parties. Ils possèdent souvent des outils et des instruments d'analyse hautement sophistiqués.
- Les «organisations financées» (classe III) : elles sont capables de rassembler des équipes de spécialistes ayant des capacités complémentaires, soutenues par des financements importants. Elles sont capables d'analyser en profondeur le système, de mettre au point des attaques complexes, et d'utiliser les outils d'analyse les plus modernes. Des attaquants de la classe II peuvent faire partie de ces équipes.

De même, on a pris l'habitude de classer les attaques physiques selon deux critères : les attaques *invasives* ou *non-invasives*, et les attaques *actives* ou *passives*.

2.1 Attaques invasives ou non-invasives

Une attaque *invasive* nécessite la suppression de l'enveloppe du micro-module, afin de pouvoir accéder directement à sa structure interne. Un exemple typique consiste à brancher une dérivation sur un *bus de données* afin d'intercepter les transferts de données.

1. Ainsi l'analyse de la vulnérabilité spécifique des FPGA (Field Programmable Gate Arrays) a été récemment abordée par T. Wollinger, C. Paar [297] et S.B. Örs, E. Oswald, B. Preneel [278].

Au contraire, dans une attaque *non-invasive*, l'adversaire n'utilise que les informations disponibles à l'extérieur du système, comme les temps d'exécution, la puissance électrique consommée, le rayonnement électromagnétique, *etc.*

On parle aussi parfois d'attaque *semi-invasive* [293], lorsque l'attaquant enlève l'enveloppe de la puce pour accéder à sa surface, mais garde intacte la couche de protection du micro-module (ces attaques ne nécessitant pas l'établissement d'un contact électrique avec la surface de métal).

Les cartes à microprocesseur sont munies de mécanismes de protection pour contrecarrer les attaques invasives. La technologie actuelle utilisée met ainsi en jeu (entre autres) plusieurs couches métalliques de protection, des détecteurs d'intrusion, ou encore un stockage des données sous des formats particuliers qui rendent très difficile leur interprétation. En revanche, les attaques non-invasives sont par définition indétectables. Comme elles sont en général également les moins onéreuses, ce sont elles qui constituent le principal danger pour la sécurité des cartes à puce.

2.2 Attaques actives ou passives

Dans une attaque *active*, on tente de perturber le fonctionnement normal du système. Ainsi les attaques par *injection de fautes* ont pour objectif de provoquer des erreurs au cours de certains calculs effectués par le système (voir le paragraphe 3).

Une attaque active implique une modification de l'environnement physique de la carte pour la placer dans des conditions anormales de fonctionnement. Plusieurs moyens sont à la disposition de l'attaquant[2] :

- L'*alimentation* : selon le standard ISO/IEC 7816-2 [96] (voir chapitre 2), le micro-module doit pouvoir supporter une tension d'alimentation V_{cc} comprise entre 4,25 et 5,25 volts. Pour ces valeurs, la carte doit fonctionner normalement. En revanche, si une variation brusque de l'alimentation (appelée *spike*) fait sortir V_{cc} de l'intervalle de tolérance, cela peut provoquer un résultat faux, à supposer que la carte soit capable de finir complètement le calcul.
- L'*horloge* : de façon analogue, le standard ISO/IEC 7816-2 définit une fréquence de référence pour l'horloge externe ainsi qu'un intervalle de tolérance. L'utilisation d'une fréquence anormalement haute ou basse peut également résulter en des erreurs[3].
- La *température* : placer la carte dans des conditions de température extrêmes est un moyen potentiel de provoquer des fautes, même s'il est assez peu utilisé aujourd'hui dans la pratique.
- Les *rayonnements* : le folklore présente souvent les attaques par injection de faute comme les «attaques au micro-ondes» (l'attaquant plaçant la carte à puce dans un four à micro-ondes pour lui faire calculer des résultats erronés). Au delà de cette vision un peu caricaturale, il est reconnu que des rayonnements correctement dirigés peuvent influencer le comportement de la carte.
- La *lumière* : Skorobogatov et Anderson [293] ont récemment observé que l'illumination d'un transistor peut le faire basculer temporairement dans son état conducteur, provoquant ainsi une erreur. En appliquant une source de lumière intense (produite par une lampe flash d'appareil photographique, amplifiée par un microscope), ils ont pu changer la valeur de

2. J'emprunte certains éléments de ce paragraphe à l'étude [282] de J.-J. Quisquater et F. Koeune, ainsi qu'à celle coordonnée par E. Oswald dans [54].

3. Blömer et Seifert [223] remarquent à ce sujet : «a finely tuned clock glitch is able to completely change a CPU's execution behavior including the omitting of instructions during the executions of programs».

bits individuels dans une mémoire SRAM (*Static* RAM). Par la même technique, ils ont pu également interférer avec les instructions `jump`, perturbant ainsi des sauts conditionnels.

- Les *courants de Foucault* (*Eddy currents*) : Quisquater et Samyde [284] ont également montré récemment que les courants de Foucault induits par un champ magnétique dans une bobine peuvent produire par exemple provoquer des erreurs dans une cellule de mémoire (qu'elle soit de type RAM, EPROM, EEPROM ou Flash).

À l'inverse, dans une attaque *passive*, l'adversaire se contente d'observer le comportement de la carte dans son fonctionnement normal. Là encore, plusieurs types d'informations de nature physique sont potentiellement utilisables par l'attaquant :

- Le *temps d'exécution* : le temps pris par un système pour exécuter un algorithme est parfois variable. Cela peut être dû à certaines instructions dont le temps d'exécution dépend des données, ou bien à des optimisations du compilateur, ou encore à l'existence de plusieurs branches dans l'algorithme. L'idée des *attaques temporelles* (*timing attacks* qui en découlent a été publiée par Paul Kocher en 1996 [259], avant d'être appliquée à de nombreux cryptosystèmes tels que DES [249], AES [288], RSA [239, 287] ou les schémas à base de courbes elliptiques [266, 265, 274].
- La *consommation électrique* : la plupart des micro-modules actuels s'appuient sur une logique CMOS (*Complementary Metal-Oxyde Semiconductor*), dont on peut caractériser la consommation ainsi : à chaque coup d'horloge, les portes logiques changent d'état simultanément, provoquant le chargement ou le déchargement des capacités internes, ce qui se traduit par une variation de l'intensité du courant, mesurable de l'extérieur. En pratique, l'attaquant utilise soit une carte d'acquisition de données, soit un oscilloscope numérique pour collecter les données. L'intensité du courant peut être mesurée soit directement avec une sonde, soit en branchant une résistance en série avec la masse ou l'entrée de l'alimentation de la carte.
- Le *rayonnement électromagnétique* : le chargement et le déchargement des capacités des portes logiques a également pour conséquence la création d'un champ électromagnétique. On distingue les émanations directes, dues au courant qui circule lors de l'exécution de l'algorithme, et les émanations indirectes, provoquées par des effets de couplage entre des composants très proches[4]. Des mises en œuvre concrètes d'attaques électromagnétiques sont décrites dans [242, 283, 209].

3 Attaques par injection de fautes

Si les *timing attacks* sont les attaques passives les plus faciles à mettre en œuvre[5], c'est en essayant de provoquer des erreurs de calcul que l'on obtient les attaques actives les plus simples à mener (et les moins coûteuses). Ces attaques constituent d'ailleurs une menace non seulement pour les algorithmes cryptographiques, mais aussi pour d'autres composantes logicielles, comme les *machines virtuelles* Java[6], ou plus globalement le *système d'exploitation* dans son ensemble, pour lequel j'ai récemment proposé une méthode générique de protection semi-automatique avec M.-L. Akkar et O. Ly [212].

4. La miniaturisation et la complexification des technologies CMOS ne font qu'accentuer ce phénomène.

5. D. Boney et D. Brumley [224] ont même montré récemment qu'elles peuvent s'appliquer «à distance», en attaquant (au travers d'un réseau local) un serveur utilisant le protocole SSL.

6. Voir à ce sujet l'article récent de S. Govindavajhala et A.W. Appel [245].

En ce qui concerne les cryptosystèmes, c'est en septembre 1996 que trois chercheurs de Bellcore, Boneh, DeMillo et Lipton proposent un nouveau modèle d'attaque physique sur les cartes à microprocesseur, qu'ils baptisent «cryptanalysis in the presence of hardware faults» [225, 226, 227]. Ce modèle d'attaque est dirigé contre plusieurs algorithmes cryptographiques à clé publique : le schéma de signature RSA et les schémas d'authentification de Fiat-Shamir ou de Schnorr.

Dans le cas de la signature RSA, les auteurs montrent que :

- si l'implémentation utilise le théorème des restes chinois (CRT), une signature erronée et la signature correcte correspondante suffisent à factoriser le module ;
- si l'implémentation n'utilise pas les restes chinois, l'attaque peut fonctionner avec un nombre de signature erronées de l'ordre du nombre de bits du module.

Très peu de temps après, Lenstra, puis Joye et Quisquater [263, 253] remarquent que, pour une implémentation du RSA avec CRT, il suffit d'avoir un message et une signature erronée de ce message pour retrouver la factorisation du modulo. Le cas du RSA, ainsi que des algorithmes de signature ElGamal, Schnorr et DSA, est également étudié dans [218, 255]. Pour RSA, des contre-mesures ont été proposées dans [291, 292, 299, 217].

Biham et Shamir s'intéressent ensuite au cas des algorithmes cryptographiques à clé secrète. Ils montrent [221], que l'algorithme DES est aussi potentiellement vulnérable aux attaques «à la Bellcore», qu'ils rebaptisent *Differential Fault Analysis* (DFA). Il faut pour cela réussir, pour 200 exécutions de l'algorithme, à perturber à chaque fois un bit, et à récupérer les 200 messages chiffrés erronés. Pour un modèle d'attaque légèrement différent, Anderson et Kuhn [213, 214] réduisent à 10 le nombre de messages erronés nécessaires pour retrouver la clé secrète du DES. Ils supposent que l'on peut perturber la carte de manière à ce qu'une instruction assembleur bien choisie ne soit pas exécutée par le microprocesseur.

Tous les algorithmes cryptographiques sont potentiellement vulnérables aux attaques par injection de fautes. Ainsi Biehl, Meyer, Müller [220] et Ciet, Joye [233] ont montré que les attaques du type *Differential Fault Analysis* peuvent également être efficaces contre les cryptosystèmes à base de courbes elliptiques. Le cas de l'AES a quant à lui été étudié par Blömer, Seifert [223], Giraud [243], Dusart, Letourneux, Vivolo [240] et Piret, Quisquater [281].

Par ailleurs, les attaques DFA ont été généralisées dans plusieurs directions. Ainsi Biham et Shamir [221] montrent que dans certains cas on peut retrouver la clé secrète, ceci même sans connaître la spécification de l'algorithme implémenté dans la carte. Paillier a poursuivi l'étude de ce modèle d'attaque dans [280]. Par ailleurs Joye, Quisquater, Yen et Yung montrent que vérifier la justesse du résultat des calculs cryptographiques n'est pas toujours suffisant [298], et que cela peut même parfois aider l'attaquant [256].

4 Attaques par analyse de consommation électrique

Les attaques passives, définies au paragraphe 2.2, présentent l'avantage (pour l'attaquant) de ne pas perturber le fonctionnement du système. Certaines d'entre elles ont connu depuis quelques années un retentissement tout particulier. Il s'agit des attaques par *analyse de consommation électrique* (*power analysis* en anglais).

Dans [313], écrit avec J. Patarin, j'ai étudié en détail les attaques utilisant le principe d'*analyse différentielle de consommation* (*Differential Power Analysis* – ou DPA – en anglais).

4.1 Analyse élémentaire de la consommation

On appelle généralement *trace* la courbe de consommation électrique (en fonction du temps) obtenue pour une exécution d'un algorithme cryptographique. Le principe de l'*analyse élémentaire de consommation* (*Simple Power Analysis* – ou SPA – en anglais) consiste à essayer d'obtenir des informations sur la clé secrète de l'algorithme à partir d'une seule trace. Ceci n'est possible que si les variations de consommation électrique sont suffisamment importantes pour être décelées visuellement de manière directe. En outre, cela suppose qu'il y ait un lien simple et exploitable entre les informations observées et la clé secrète elle-même. C'est pourquoi ce type d'attaque vise particulièrement les implémentations qui utilisent des branchements dépendant de la clé.

Pour mener à bien une attaque SPA, on utilise souvent le fait que les valeurs de la consommation électrique sont fortement corrélées au *poids de Hamming* des données manipulées par les instructions assembleur. Cette idée a été développée par Biham et Shamir [222] pour le DES, par Mangard [267] pour l'AES, ou encore par Rao, Rohatgi et Scherzer [285] pour l'algorithme COMP128-1 [246, 229] utilisé dans le standard GSM de téléphonie mobile[7]. Par ailleurs Schramm, Wollinger et Paar ont récemment montré que le principe de la SPA pouvait également être utilisé pour détecter des collisions internes, notamment dans l'algorithme DES : ce sont les *attaques par collision* (*collision attacks*) [290].

Les algorithmes asymétriques sont aussi potentiellement vulnérables face aux attaques de type SPA. En particulier pour les algorithmes (tels RSA) faisant intervenir l'*exponentiation* d'une valeur connue, par un exposant secret, un scénario d'attaque utilisant à nouveau la notion de *poids de Hamming* est décrit par Klíma et Rosa dans [258]. De même, la *multiplication scalaire* d'un point connu d'une courbe elliptique, par un scalaire secret, donne lieu à des attaques : ainsi Oswald [279] utilise le *modèle de Markov* pour retrouver la clé dans le cas d'une implémentation avec des *chaînes d'addition-soustraction*. Clavier et Joye [234] ont établi un modèle général de ces attaques SPA, ouvrant la voie à des protections génériques et prouvées sûres pour ce type d'algorithmes asymétriques.

4.2 Analyse différentielle de la consommation

L'*analyse différentielle de consommation* (*Differential Power Analysis* – ou DPA – en anglais) a été introduite par Kocher, Jaffe et Jun en 1998 [260] et publiée en 1999 [261]. L'idée est d'exploiter les *corrélations* éventuelles entre les données manipulées par le microprocesseur et les valeurs instantanées de consommation électrique. Comme ces corrélations sont souvent très faibles, il faut avoir recours à des méthodes *statistiques* pour en tirer le maximum d'information.

Dans une attaque de type DPA, le principe consiste à comparer des valeurs mesurées lors du fonctionnement du *véritable* dispositif physique (par exemple la carte à microprocesseur) avec des valeurs calculées grâce à un modèle *hypothétique* de ce dispositif (les hypothèses portant notamment sur la nature de l'implémentation, et surtout sur une partie de la clé secrète). En comparant ces deux ensembles de valeurs, on s'efforce ensuite de retrouver tout ou partie de la clé secrète.

Les cibles initiales des attaques DPA étaient les algorithmes symétriques. La vulnérabilité du DES – mise en évidence par Kocher, Jaffe, Jun [260, 261] – a été étudiée plus en détail dans mon article [313] écrit avec Patarin. Sur ce thème on peut citer aussi Messerges, Dabbish, Sloan [269]

7. L'algorithme COMP128-1 avait de toute façon déjà succombé à la cryptanalyse de Briceno, Goldberg et Wagner [228, 247] en 1998.

et Akkar, Bevan, Dischamp, Moyart [211]. L'application de ces attaques a été aussi largement prise en compte au cours du processus de sélection de l'AES, notamment par Biham, Shamir [222], Chari, Jutla, Rao, Rohatgi [232] et Daemen, Rijmen [238].

Mais les algorithmes asymétriques ne sont pas à l'abri non plus : j'ai ainsi montré dans [313] (tout comme Messerges, Dabbish, Sloan dans [270]) comment appliquer la DPA sur l'algorithme RSA, et le cas des courbes elliptiques a été analysé par Coron [236], Okeya, Sakurai [276], puis par beaucoup d'autres (voir paragraphe 6).

4.3 Exemple de l'algorithme DES

Considérons, pour illustrer les attaques par analyse différentielle de consommation, l'exemple de l'algorithme de chiffrement DES. Celui-ci s'exécute en 16 étapes, appelées *tours* (ou *rounds* en anglais). Lors de chacun de ces tours, une transformation F est appliquée sur 32 bits. Cette fonction F utilise elle-même huit transformations non-linéaires de 6 bits sur 4 bits, chacune d'entre elles étant stockée dans une table appelée *boîte*-S (S-*box*). On peut alors mettre en œuvre une attaque DPA de la manière suivante, telle que je l'ai décrite dans [313] (le nombre 1000 est donné à titre d'exemple).

Première étape : On mesure la trace de consommation électrique du premier tour, ceci pour 1000 exécutions du DES. On désigne par E_1, ..., E_{1000} les valeurs d'entrée de ces 1000 calculs, et par C_1, ..., C_{1000} les 1000 courbes de consommation mesurées au cours de ces calculs. À partir de là, on calcule la *courbe moyenne* MC de ces 1000 courbes de consommation.

Deuxième étape : On se focalise par exemple sur le premier bit de sortie de la première boîte-S, lors du premier tour. Soit b la valeur de ce bit. Il est facile de voir que b ne dépend que de 6 bits de la clé secrète. L'attaquant fait alors une hypothèse sur les 6 bits impliqués. Il peut calculer – à partir de ces 6 bits et des E_i – la valeur (théorique) attendue pour b, et ainsi séparer les entrées E_1, ..., E_{1000} en deux catégories : celles qui donnent $b = 0$ et celles qui donnent $b = 1$.

Troisième étape : On calcule maintenant la courbe moyenne MC' des courbes correspondant aux entrées de la première catégorie (*i.e.* celles pour lesquelles $b = 0$). Si MC et MC' présentent, en un certain point, une différence appréciable (au sens statistique, *i.e.* une différence nettement plus grande que l'écart-type du bruit mesuré), l'attaquant en déduit que la valeur choisie pour les 6 bits de clé étaient corrects. En revanche, si MC et MC' ne présentent pas de différence notable, on recommence la deuxième étape avec un autre choix pour les 6 bits[8].

Quatrième étape : On répète les étapes 2 et 3, avec un bit «cible» b dans la deuxième boîte-S, puis dans la troisième boîte-S, ..., jusqu'à la huitième boîte-S. En tout, cela permet de retrouver 48 bits de la clé secrète.

Cinquième étape : Les 8 bits manquants peuvent être trouvés par recherche exhaustive (ou bien en considérant le deuxième tour, avec exactement la même méthodologie d'attaque)

Cette attaque ne nécessite aucune connaissance *a priori* sur les valeurs de consommation individuelles de chaque instruction, ni sur la position dans le temps de ces instructions. La

8. En pratique, pour chaque choix des 6 bits, on trace la courbe représentant la différence entre MC et MC'. On obtient ainsi 64 courbes, parmi lesquelles une est supposée présenter des «pics» de consommation, aisément reconnaissables par rapport aux autres courbes.

DPA peut s'appliquer à partir du moment où l'attaquant connaît les entrées (ou les sorties) de l'algorithme, ainsi que les courbes de consommation correspondantes. Elle s'appuie uniquement sur l'hypothèse suivante, que j'ai mise en évidence dans [313] :

Hypothèse fondamentale : *Il existe une variable intermédiaire, qui apparaît au cours de l'exécution de l'algorithme, telle que la connaissance d'un petit nombre de bits de clé (en pratique moins de 32 bits) permet de décider si deux entrées (respectivement deux sorties) de l'algorithme donnent ou non la même valeur pour cette variable.*

5 Représentations booléennes et arithmétiques

5.1 Le problème de conversion

Dans [268], T. Messerges étudie l'application des attaques DPA aux candidats à l'AES. Il propose une méthode générale de sécurisation des implémentations, consistant à masquer systématiquement l'entrée et la sortie de chaque opération élémentaire du microprocesseur. Cette technique générique présente l'avantage de faciliter l'évaluation de l'impact de la protection sur les performances de l'algorithme considéré.

Cette technique de masquage est réalisable à condition que toutes les opérations fondamentales utilisées par un algorithme donné puissent être réécrites de façon à prendre leurs entrées sous forme masquée pour donner les sorties masquées également. On peut se persuader facilement que c'est bien le cas pour l'algorithme DES, car un masquage unique (utilisant le *ou-exclusif*) peut être utilisé tout au long des 16 tours du calcul. De même, pour RSA, il suffit de prendre un masque utilisant la multiplication dans le groupe multiplicatif modulo n.

En revanche, pour les algorithmes qui combinent les opérations *booléennes* et les opérations *arithmétiques*, on doit utiliser deux masquages différents. On a donc besoin d'une méthode de conversion d'un masquage booléen en un masquage arithmétique, et *vice versa*. Le problème se pose notamment pour les algorithmes IDEA [262] et pour trois des candidats à l'AES : MARS [231], RC6 [286] et TwoFish [289].

5.2 La méthode de Messerges

Aux opérations booléennes et arithmétiques sont associés deux masquages différents, utilisant une valeur aléatoire r :

- Un *masquage booléen*, consistant à remplacer chaque entrée (ou sortie) x d'une opération booléenne par la valeur $x' = x \oplus r$;
- Un *masquage arithmétique*, consistant à remplacer chaque entrée (ou sortie) x d'une opération booléenne par la valeur $A = (x - r) \bmod 2^k$.

Dans les deux cas, on dit que la valeur x est *masquée* par la valeur aléatoire r (le *masque*) pour donner la *valeur masquée* x' (resp. A).

Pour transformer une valeur masquée A (en *représentation arithmétique*) en la valeur masquée équivalente x' (en *représentation booléenne*), la méthode de conversion proposée par Messerges [268] consiste à appliquer successivement les opérations suivantes :

- $C \longleftarrow$ 0 ou 1 (avec probabilité $\frac{1}{2}$)
- $B \longleftarrow C \oplus r$ /* $B = r$ ou $B = \bar{r}$ */
- $A \longleftarrow B \oplus x'$ /* $A = x$ ou $A = \bar{x}$ */

- $A \longleftarrow A - B$ /* $A = x - r$ ou $A = \bar{x} - \bar{r}$ */
- $A \longleftarrow A + C$ /* $A = x - r$ ou $A = \bar{x} - \bar{r} - 1$ */
- $A \longleftarrow A \oplus C$ /* $A = x - r$ */

La conversion inverse (de la représentation booléenne vers la représentation arithmétique) est effectuée par un algorithme similaire.

5.3 Attaque par analyse de consommation

Dans [237], avec J.-S. Coron, j'ai montré que la méthode de Messerges est insuffisante pour protéger un algorithme cryptographique contre les attaques par analyse de consommation.

En effet, l'algorithme qui réalise la conversion d'une type de masquage à l'autre doit lui-même résister aux attaques DPA. Celui proposé par Messerges prend en entrée un couple (x',r) tel que $x = x' \oplus r$. La donnée non masquée est x et la donnée masquée est x'. L'algorithme fonctionne en «démasquant» x' au moyen de l'opération *ou-exclusif*, puis en le «remasquant» au moyen de l'addition.

La faiblesse réside dans le fait que l'algorithme recalcule la valeur x ou la valeur $\bar{x}$ au cours de son exécution. Certes l'attaquant ne sait pas de laquelle des deux valeurs il s'agit, ce qui empêche les attaques DPA sélectionnant un seul bit : comme chacune des deux valeurs x et $\bar{x}$ a une probabilité $\frac{1}{2}$ d'apparaître, chacun des bits est décorrélé de la valeur que l'on cherche à cacher. Néanmoins, ce raisonnement n'est plus valable pour une attaque DPA un peu plus générale, qui considère *deux* bits simultanément.

L'attaque s'appuie sur le fait que, si 2 bits de x sont égaux, alors les bits analogues dans $\bar{x}$ sont aussi égaux. Par conséquent, en modifiant le scénario d'attaque décrit au paragraphe 4.3, on sélectionne deux bits «cibles» qui permettent de séparer les traces de consommation en deux groupes : celles pour lesquelles les deux bits sont égaux, et celles pour lesquelles ces deux bits diffèrent. Cette classification n'est pas modifiée pas la transformation de x en $\bar{x}$. Ainsi, si la consommation électrique pour deux bits égaux est distinguable de la consommation pour deux bits différents, l'hypothèse correcte sur la clé secrète engendre un «pic» de consommation, alors que les hypothèses erronées produisent une courbe quasiment «plate». On peut dès lors retrouver tous les bits de la clé secrète.

Plus précisément, considérons les quatre distributions de consommation possible, et notons $\mu_{00}, \mu_{01}, \mu_{10}, \mu_{11}$ les valeurs moyennes respectives. Pour l'hypothèse correcte sur la clé, la consommation moyenne pour le premier groupe est égale à

$$\frac{\mu_{00} + \mu_{11}}{2}$$

alors qu'elle vaut, pour le second groupe,

$$\frac{\mu_{01} + \mu_{10}}{2}.$$

La différence entre les moyennes des deux groupes est donc égale à

$$\Delta = \frac{\mu_{00} + \mu_{11} - \mu_{01} - \mu_{10}}{2}$$

et l'attaque DPA sur deux bits réussit à partir du moment où $\Delta \neq 0$.

Cette attaque, qui ne considère qu'un point de chaque courbe de consommation, requiert un nombre d'échantillons comparable à l'attaque de base décrite au paragraphe 4.3. Elle montre que la méthode proposée dans [268] n'est pas suffisante pour garantir une protection contre les analyses de consommation. Nous verrons au chapitre 7 comment on peut résoudre le problème, avec en outre une preuve de sécurité contre la DPA.

6 Attaques physiques sur les courbes elliptiques

6.1 État de l'art des protections

Comme nous l'avons mentionné au paragraphe 4, les cryptosystèmes à base de courbes elliptiques ne sont pas à l'abri des attaques par analyse de consommation électrique.

Pour contrecarrer les attaques de type SPA, deux stratégies ont été élaborées. La première consiste à cacher le fait que – pendant le calcul de la *multiplication scalaire* $d \cdot P$ d'un point P de la courbe elliptique par l'entier d – la nature des opérations élémentaires exécutées successivement (*e.g.* addition ou doublement) dépend de la valeur de l'exposant secret d. Poursuivant cette idée, Coron [236] a proposé la méthode dite «toujours-additionner-et-doubler» («*double-and-add-always*» en anglais, voir l'algorithme 2), optimisée dans [275]. La méthode de Montgomery [272] (voir l'algorithme 3) s'est révélée capable elle aussi d'éviter à la fois les *timing attacks* et les attaques SPA [274, 276]. En outre, pour les corps $\mathbb{F}_{2^m}$, une astuce permet d'obtenir la multiplication scalaire en évitant le calcul des ordonnées y [208, 266]. Cette propriété a été étendue au cas des corps premiers $\mathbb{F}_p$ pour les courbes elliptiques ayant une *forme de Montgomery* [277, 271] puis pour toutes les courbes elliptiques sur $\mathbb{F}_p$ [251, 230, 241].

Algorithme 2 : «Toujours-additionner-et-doubler» (à partir du bit de poids fort)

Input: P, $d = d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + ... + d_1 2 + d_0$ (avec $d_{n-1} = 1$)
Output: $Q_0 = d \cdot P$
  $Q_0 := P$
  for $i = n - 2$ down to 0 **do**
    $Q_0 := 2 \cdot Q_0$
    $Q_1 := Q_0 + P$
    $Q_0 := Q_{d_i}$
  end for
  Return Q_0

La seconde stratégie consiste à rendre indistinguables l'addition et le doublement lors de la multiplication scalaire [234]. Cela s'est révélé faisable pour plusieurs classes de courbes sur un corps premier F_p : les courbes de type Hesse [294, 254] et celles de type Jacobi [264] donnent lieu à une formule unifiée pour le calcul des additions et des doublements. Une formule unifiée a été proposée[9] par Brier et Joye [230] pour obtenir la même indistinguabilité pour toutes les courbes sur $\mathbb{F}_{2^m}$ ou $\mathbb{F}_p$. Dans le même but, et pour la caractéristique 2, l'insertion d'opérations «leurres» est également possible [219].

9. Cette méthode a été toutefois remise en cause récemment par Izu et Takagi [252].

Algorithme 3 : Méthode de Montgomery

Input: P, $d = d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + ... + d_1 2 + d_0$ (avec $d_{n-1} = 1$)
Output: $Q_0 = d \cdot P$

```
Q_0 := P
Q_1 := 2 · P
for i = n − 2 down to 0 do
    Q_{1−d_i} := Q_0 + Q_1
    Q_{d_i} := 2 · Q_{d_i}
end for
Return Q_0
```

Comme cela a été noté dans [236, 276, 257], ces méthodes anti-SPA ne sont pas suffisantes pour empêcher les attaques DPA. Néanmoins, plusieurs contre-mesures ont été proposées pour transformer une implémentation SPA-résistant en une implémentation DPA-résistante.

Dans [236], Coron a suggéré trois méthodes anti-DPA : la *randomisation de l'exposant* secret d, l'*addition d'un point aléatoire* R à P, et l'utilisation de *coordonnées projectives* homogènes randomisées. La troisième méthode notamment est largement répandue [274, 276, 264].

Dans le même esprit, Joye et Tymen [257] ont proposé deux autres méthodes génériques : effectuer les calculs sur une autre courbe elliptique, déduite de la courbe usuelle par un *isomorphisme aléatoire de courbes elliptiques*, et effectuer les opérations élémentaires en utilisant une autre représentation du corps de base, déduite de la représentation usuelle par un *isomorphisme aléatoire de corps*. Ils donnent également une méthode spécifique pour les *courbes ABC* (voir aussi [248]).

6.2 Une attaque à message choisi

Dans [244], j'ai présenté une attaque par analyse de consommation, capable de retrouver la clé secrète pour un grand nombre de courbes elliptiques, même dans le cas où une contre-mesure SPA (telle que *toujours-additionner-et-doubler* ou la méthode de Montgomery) est utilisée conjointement avec l'une des trois protections anti-DPA mentionnées au paragraphe 6.1.

Dans mon scénario, l'attaquant peut choisir le message, ou plus précisément le point P de la courbe intervenant dans la multiplication scalaire $d \cdot P$. Le seul masquage appliqué aux données provient de l'utilisation, soit des coordonnées projectives randomisées (pour le point P), soit de l'isomorphisme aléatoire de courbes elliptiques (pour la courbe elle-même), soit de l'isomorphisme aléatoire de corps (pour la structure algébrique sous-jacente).

Dans ce paragraphe, je décris la stratégie générale de l'attaque, dans le cas particulier de l'algorithme «toujours-additionner-et-doubler». On peut consulter mon article [244] pour le cas de la méthode de Montgomery, qui procède de la même philosophie. Notons à ce sujet que l'attaque ne se limite pas au cas des méthodes binaires pour la multiplication scalaire (comme les algorithmes 2 et 3) et peut s'étendre sans difficulté à toutes les chaînes d'addition. Par ailleurs, elle s'applique aussi bien à la représentation de Weierstraβ des courbes elliptiques qu'aux différentes sortes de représentations projectives [10].

10. Notamment les coordonnées projectives homogènes ou jacobiennes, parmi les nombreuses possibilités analysées par Cohen, Miyaji et Ono dans [235].

Supposons que l'attaquant ait déjà réussi à déterminer les bits de poids fort d_{n-1}, ..., d_{i+1} de l'exposant secret d. L'objectif est maintenant de trouver le bit suivant d_i.

Supposons que la courbe elliptique $E(K)$ contienne un point «spécial» P_0, distinct du *point à l'infini* $\mathcal{O}$, mais ayant une de ses coordonnées (affines ou projectives) nulle dans le corps K. Il est facile de voir que, pour toutes les protections DPA mentionnées précédemment, la randomisation ne change pas le caractère «spécial» de ce point P_0.

Cas de l'algorithme «toujours-additionner-et-doubler»

Dans l'algorithme 2, pour tout point P initial, la valeur Q_0 obtenue après le i-ème tour de la boucle est

$$Q_0 = \Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i} + d_i \Big) \cdot P.$$

Il y a alors deux possibilités :

- Si $d_i = 0$, les valeurs apparaissant au $(i+1)$-ème tour de la boucle sont $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} \Big) \cdot P$ et $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big) \cdot P$.
- Si $d_i = 1$, ces valeurs sont $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 2 \Big) \cdot P$ et $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3 \Big) \cdot P$.

On considère alors le point P_1 de la courbe elliptique, défini par

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big)^{-1} \bmod |E(K)| \Big] \cdot P_0$$

si $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big)$ est premier avec $|E(K)|$ (cela correspond à l'hypothèse $d_i = 0$), ou bien

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3 \Big)^{-1} \bmod |E(K)| \Big] \cdot P_0$$

si $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3 \Big)$ est premier avec $|E(K)|$ (cela correspond à l'hypothèse $d_i = 1$). Très souvent, on a le choix entre les deux options.

Appelons maintenant $\mathcal{C}_r$, pour $1 \leq r \leq R$, les courbes de consommation électrique associées à r calculs distincts de $d \cdot P_1$. À cause de la randomisation effectuée au début de chaque calcul, la même valeur initiale P_1 peut engendrer des traces de consommation différentes les unes des autres. On considère alors la *courbe moyenne*

$$\mathcal{M}_{P_1} = \frac{1}{R} \sum_{r=1}^{R} \mathcal{C}_r.$$

Si l'hypothèse faite sur d_i (correspondant au choix fait pour le point P_1) est fausse, la courbe moyenne $\mathcal{M}_{P_1}$ est indistinguable en pratique de la courbe moyenne $\mathcal{M}_{P_2}$ qu'on obtiendrait avec un point quelconque P_2 de la courbe elliptique. En effet les valeurs apparaissant au $(i+1)$-ème tour de la boucle (dans l'algorithme 2) sont correctement randomisées.

En revanche, si l'hypothèse faite sur d_i est juste, la courbe moyenne $\mathcal{M}_{P_1}$ présente des «pics» de consommation appréciables par rapport à la courbe moyenne $\mathcal{M}_{P_2}$ qu'on obtiendrait avec un point quelconque P_2 de $E(K)$. Cela correspond à la consommation spécifique de la manipulation de la valeur zéro par les instructions assembleur au $(i+1)$-ème tour de la boucle.

Une fois déterminée la valeur de d_i, les bits restants d_{i-1}, ..., d_0 sont trouvés de façon récursive, en appliquant le même principe d'attaque.

6.3 Application à des courbes elliptiques standards

La stratégie d'attaque décrite au paragraphe 6.2 est applicable contre les méthodes de protection DPA reposant sur l'utilisation des coordonnées projectives randomisées, des isomorphismes aléatoires de courbes elliptiques ou encore des isomorphismes aléatoires de corps. Pour que l'attaque puisse s'appliquer concrètement, il faut vérifier l'existence d'un «point spécial» sur la courbe elliptique sur laquelle est construit le cryptosystème.

Points spéciaux de la forme $(0,y)$

Pour une courbe elliptique non-singulière, dont la forme réduite de Weierstraβ est $E : y^2+xy = x^3 + ax^2 + b$ sur $K = \mathbb{F}_{2^m}$, on peut prendre $P_0 = (0,b^{2^{m-1}})$ comme point «spécial».

Pour une courbe elliptique $E : y^2 = x^3 + ax + b$ sur un corps premier $K = \mathbb{F}_p$ $(p > 3)$, un point spécial de la forme $(0,y)$ existe si et seulement si b est un *résidu quadratique* modulo p, c'est-à-dire $\left(\frac{b}{p}\right) = 1$ où $\left(\frac{\cdot}{\cdot}\right)$ désigne le *symbole de Legendre*.

Parmi les courbes standards sur un corps premier satisfaisant cette condition, on en dénombre quatre dans FIPS 186-2 [273], une dans WTLS[11] [296] (la courbe usuelle), sept dans ANSI X9.62 [215] et deux dans ISO/IEC 15946-4 [250]. Seules une courbe FIPS 186-2 (P224) et quatre courbes ISO/IEC 15946-4 (Annexes A1.1[12], A4.1, A5.1 and A6.1) ne possèdent pas de point spécial $(0,y)$.

Points spéciaux de la forme $(x,0)$

Pour une courbe elliptique $E : y^2 = x^3 + ax + b$ sur un corps premier $K = \mathbb{F}_p$ $(p > 3)$, un point spécial de la forme $(x,0)$ existe si et seulement si l'équation $x^3 + ax + b = 0$ a au moins une racine α dans K.

Remarquons que $P_0 = (\alpha,0)$ est alors un point d'ordre 2 dans $E(K)$. Il semble alors que la stratégie du paragraphe 6.2 échoue, car P_1 ne dépend plus de l'hypothèse faite sur d_i (P_1 est toujours égal à P_0). Néanmoins, les valeurs successives de Q qui apparaissent pendant l'algorithme 2, pour $i = n-2$, ..., 0 sont soit $\mathcal{O}$ (si $d_i = 0$) soit P (si $d_i = 1$). La courbe moyenne $\mathcal{M}_{P_1}$ présente donc en fait beaucoup de «pics» : par exemple si on applique l'algorithme 2 avec des coordonnées projectives homogènes aléatoires, les instructions assembleur manipulant $\mathcal{O} = (0,\theta,0)$ sont susceptibles de provoquer deux tels pics (un pour chaque zéro), alors que celles qui manipulent $(\theta x,0,\theta)$ n'en créeront qu'un. Cela permet à l'attaquant de retrouver tous les bits d_i de l'exposant secret d en une seule étape.

Notons que certaines classes de courbes ont automatiquement de tels points d'ordre 2. Ainsi pour les courbes elliptiques sous *forme de Montgomery* [272], (0,0) est d'ordre 2 : son double est $\mathcal{O} = (0,1,0)$. Pour les courbes de type Hesse [294], tous les (x,x) sont d'ordre 2 : leur double est $\mathcal{O} = (-1,1,0)$.

11. *Wireless Transport Layer Security*, qui traite des aspects de sécurité dans le protocole WAP.
12. Toutefois cette courbe a un point d'ordre deux, et donc un point spécial $(x,0)$.

Cette attaque montre donc que l'implémentation des cryptosystèmes à base de courbes elliptiques demande un grand soin, et que certaines protections ne sont pas suffisantes en elles-mêmes[13]. Néanmoins, elles vont dans la bonne direction, et des améliorations sont possibles pour éviter ce type d'attaque[14].

13. Notons que T. Akishita et T. Takagi ont proposé tout récemment [210] une variante de mon attaque, appelée *Zero-Value Point Attack*, qui considère les valeurs nulles pouvant apparaître au sein même de l'opération d'addition sur la courbe elliptique. Par ailleurs, une variante de mon attaque a été décrite par R. Avanzi [216] pour les courbes *hyperelliptiques*.

14. Voir notamment l'article récent de N. Smart [295], qui propose le recours à la notion d'*isogénie* afin de se protéger contre mon attaque.

Chapitre 7

Contre-mesures

Two methods [...] suggest themselves for frustrating a statistical analysis. [...] The effect here is that the enemy must intercept a tremendous amount of material to tie down this structure.

Claude Shannon (1949)

A good disguise should not reveal the person's height. [...] A good disguise should not allow a mother to distinguish her own children.

Shafi Goldwasser et Silvio Micali (1982)

Sommaire

1 Introduction 93
2 La méthode de duplication 94
2.1 Contre-mesures usuelles 94
2.2 Principe de la méthode de duplication 94
2.3 Application à l'algorithme DES 95
2.4 Application à l'algorithme RSA 97
3 Conversions entre deux structures algébriques 98
3.1 Implémentation sécurisée de l'algorithme SFLASH 99
3.2 Le problème du zéro pour l'AES 101
3.3 Une méthode de conversion universelle 102
3.4 Application pratique 105
4 Les attaques par analyse différentielle d'ordre supérieur 105
4.1 Une généralisation de la DPA 105
4.2 Insuffisance des contre-mesures «classiques» 106
4.3 Une contre-mesure pour les attaques d'ordre supérieur 107

1 Introduction

Dans ce chapitre sont présentés plusieurs types de contre-mesures que j'ai développées pour protéger les implémentations d'algorithmes cryptographiques contre les attaques par analyse de consommation électrique. La première classe de contre-mesures génériques contre la DPA découle de la *méthode de duplication*, que j'ai publiée [313] avec J. Patarin. Le problème de la

protection des algorithmes qui utilisent deux structures algébriques différentes est abordé dans quatre de mes articles [237, 311, 173, 302], soit pour des algorithmes spécifiques (AES [302], SFLASH [173]), soit pour donner une *méthode de conversion* universelle [311] résistant aux attaques DPA, y compris celle décrite [237] au chapitre 6, §5.3. Enfin, une généralisation des attaques par analyse différentielle de consommation est étudiée dans [302], où j'ai proposé avec M.-L. Akkar une contre-mesure générique contre les attaques par *analyse différentielle d'ordre supérieur* (*High-Order Differential Power Analysis* – ou HO-DPA – en anglais).

2 La méthode de duplication

Pour protéger les implémentations de l'algorithme DES contre les attaques par analyse différentielle de consommation électrique (DPA), j'ai mis au point avec J. Patarin la méthode dite de «duplication», décrite pour la première fois dans [312], et publiée dans [313]. La même idée a été développée très peu de temps après par Chari, Jutla, Rao et Rohatgi [232, 305]. La version la plus générale de la méthode de duplication accroît d'un facteur important la mémoire nécessaire, d'autant plus qu'en général *tous* les tours de l'algorithme doivent être protégés [1]. C'est pourquoi [313] décrit également des variantes *ad hoc* pour les cartes à microprocesseur.

2.1 Contre-mesures usuelles

Plusieurs idées naturelles ont été proposées pour contrecarrer les attaques de type DPA :

- Introduire des *décalages temporels* aléatoires, ou faire varier aléatoirement la fréquence de l'horloge interne, de telle sorte que la DPA soit conduite à moyenner des valeurs de consommation associées à des instructions assembleur variables. Le point délicat de ce genre de défense consiste à réaliser les décalages de manière à rendre difficile la *resynchronisation* des traces de consommation (par des techniques d'*analyse du signal*).
- Remplacer certaines des instructions critiques (en particulier les instructions assembleur qui impliquent la modification du `carry`, la lecture de données dans un tableau, *etc*) par des instructions assembleur dont la *signature de consommation* soit difficile à analyser.
- Pour un algorithme donné, trouver une manière explicite d'effectuer les calculs, de telle sorte que la DPA soit inefficace sur l'implémentation obtenue, et ceci de façon prouvée. C'est la troisième stratégie qui est utilisée dans la méthode de duplication.

2.2 Principe de la méthode de duplication

La méthode de duplication, que j'ai proposée dans [313], repose sur l'idée suivante : si on sépare les données sensibles de l'algorithme en plusieurs parties, selon un schéma de *partage de secret* (*secret sharing*), alors toutes les variables manipulées au cours du calcul seront indistinguables de variables aléatoires. Le fait qu'elles soient alors complètement décorrélées de la clé secrète rend inopérante la stratégie d'attaque DPA décrite au chapitre 6, §4.3.

Plus précisément, chaque variable intermédiaire V, intervenant au cours du calcul et dépendant de l'entrée (ou de la sortie) de l'algorithme, est remplacée par k variables V_1, ..., V_k vérifiant l'équation $V = f(V_1,...,V_k)$, où f est une fonction publique ou secrète des k variables. La sécurité repose alors sur les deux conditions suivantes :

1. Voir notamment à ce sujet l'attaque IPA (*Inferential Power Analysis*) imaginée par Fahn et Pearson [308].

Condition 1 : Étant donnée une valeur v et un indice i ($1 \leq i \leq k$), il doit être impossible d'en déduire la moindre information sur l'ensemble des valeurs v_i telles qu'il existe un $(k-1)$-uplet $(v_1, ..., v_{i-1}, v_{i+1}, ..., v_k)$ satisfaisant l'équation $f(v_1,...,v_k) = v$.

Condition 2 : La fonction f est choisie de façon à ce que les transformations à effectuer sur V_1, V_2, ..., ou V_k pendant le calcul (à la place des transformations usuelles sur V dans une implémentation de base) puisse être implémentées sans recalculer V.

Il est facile de voir que la fonction $f(v_1,...,v_k) = v_1 \oplus v_2 \oplus ... \oplus v_k$, vérifie la condition 1, car pour tout indice i entre 1 et k, la variable v_i peut prendre toutes les valeurs possibles. De même, si la variable V appartient au groupe multiplicatif de $\mathbb{Z}_n$, on peut choisir la fonction f définie par $f(v_1,...,v_k) = v_1 \cdot v_2 \cdot ... \cdot v_k \bmod n$.

Une fois choisi un schéma de partage de secret (*i.e.* une fonction f), on «traduit» l'implémentation initiale de l'algorithme, en remplaçant chaque variable intermédiaire V dépendant de l'entrée (ou de la sortie) par les k variables V_1, ..., V_k.

2.3 Application à l'algorithme DES

Pour protéger l'algorithme DES contre les attaques DPA, on peut choisir de séparer les variables sensibles V en *deux* parties V_1 et V_2, au moyen de la fonction $f(v_1,v_2) = v = v_1 \oplus v_2$ (qui satisfait la condition 1). Une analyse de l'algorithme DES montre que les transformations subies par les variables v sont toujours de l'une des cinq catégories suivantes :

- *Permutation* des bits de v ;
- *Expansion* des bits de v ;
- *Ou-exclusif* de v avec une autre variable v' du même type ;
- *Ou-exclusif* de v avec une variable dépendant uniquement de la clé secrète ;
- Transformation de v par une *boîte-S*.

Les deux premières transformations sont *linéaires* sur $\mathbb{F}_2$. Elles *commutent* donc avec la fonction f, et par conséquent laissent invariante la relation $f(v_1,v_2) = v$. Par conséquent, il suffit de calculer – au lieu de la transformation habituellement appliquée à v – la permutation (resp. l'expansion) sur v_1, puis sur v_2.

De même, dans le troisième cas, il suffit de remplacer le calcul de $v'' = v \oplus v'$ par celui de $v''_1 = v_1 \oplus v'_1$ et de $v''_2 = v_2 \oplus v'_2$. Les identités $f(v_1,v_2) = v$ et $f(v'_1,v'_2) = v'$ impliquent en effet $f(v''_1,v''_2) = v''$. La condition 2 est également facile à satisfaire pour le *ou-exclusif* entre v et une variable c ne dépendant que de la clé secrète : il suffit de remplacer le calcul de $v \oplus c$ par $v_1 \oplus c$ (ou $v_2 \oplus c$).

Le point le plus délicat est la transformation $v' = S(v)$, où S est généralement donnée sous la forme d'une table (avec 6 bits en entrée et 4 bits en sortie, voir figure 1). Ici on calcule $(v'_1,v'_2) = S'(v_1,v_2)$ au moyen de *deux* nouvelles boîtes-S (chacune envoyant 12 bits sur 4 bits) :

$$(v'_1,v'_2) = S'(v_1,v_2) = (A(v_1,v_2), S(v_1 \oplus v_2) \oplus A(v_1,v_2)),$$

où A est une transformation secrète de 12 bits sur 4 bits choisie aléatoirement (voir figure 2). La première nouvelle boîte-S correspond à $(v_1,v_2) \mapsto A(v_1,v_2)$, et la seconde à $(v_1,v_2) \mapsto S(v_1 \oplus v_2) \oplus A(v_1,v_2)$. La condition 1 est satisfaite grâce au caractère aléatoire de A, et la condition 2 est obtenue par l'utilisation de tables pour les nouvelles boîtes-S.

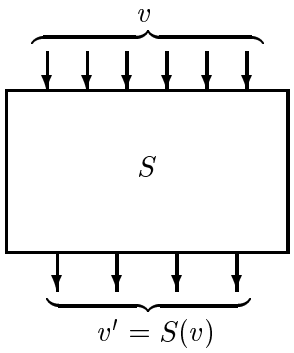


FIG. 1 – *Implémentation initiale : les valeurs prédictibles v et v' apparaissent en RAM*

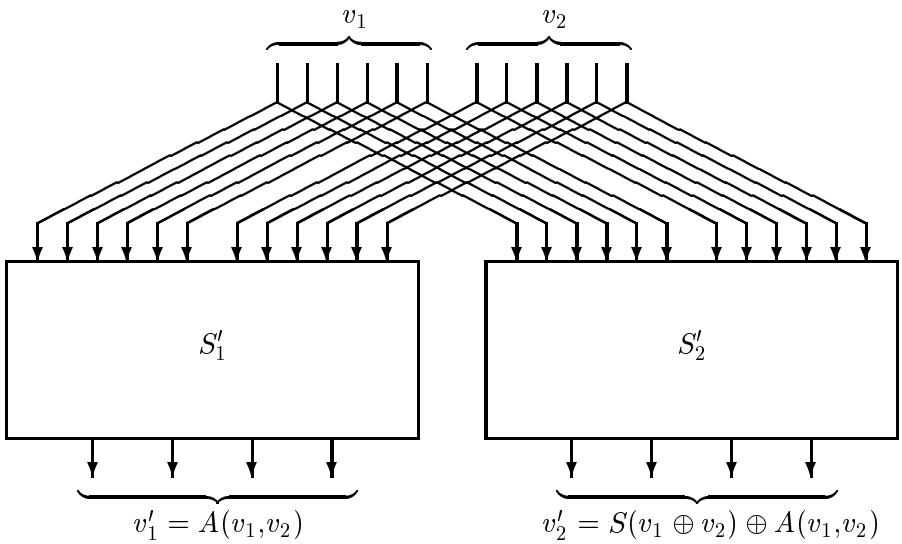


FIG. 2 – *Implémentation modifiée : les valeurs $v = v_1 \oplus v_2$ et $v' = v'_1 \oplus v'_2$ n'apparaissent jamais explicitement en RAM*

L'implémentation ainsi obtenue n'est réaliste que pour des cartes à microprocesseur où le DES est câblé, ou bien pour des PCs. En effet, la taille mémoire (RAM) nécessaire est de 32 Ko pour stocker les nouvelles boîtes-S. Pour une implémentation logicielle sur une carte à puce, j'ai décrit plusieurs variantes dans [313] qui réduisent l'encombrement mémoire à des valeurs acceptables. L'idée est de calculer $(v'_1,v'_2) = S'(v_1,v_2)$ de la façon suivante :

- $v_0 = \varphi(v_1 \oplus v_2)$
- $(v'_1,v'_2) = S'(v_1,v_2) = (A(v_0),S(\varphi^{-1}(v_0)) \oplus A(v_0))$

où φ est une fonction bijective et secrète de 6 bits sur 6 bits (et A une transformation secrète aléatoire de 6 bits sur 4 bits). Les deux nouvelles boîte-S (correspondant aux transformations $v_0 \mapsto A(v_0)$ et $v_0 \mapsto S(\varphi^{-1}(v_0)) \oplus A(v_0)$) n'occupent plus alors que 512 bits.

La fonction φ doit par ailleurs vérifier la condition 2, et permettre le calcul de $v_0 = \varphi(v_1 \oplus v_2)$ sans avoir à calculer $v_1 \oplus v_2$. On peut par exemple prendre pour φ une fonction bijective secrète et *linéaire* sur $\mathbb{F}_2$, ce qui donne $v_0 = \varphi(v_1) \oplus \varphi(v_2)$. Une autre possibilité, inspirée du principe de *représentation obscure* dû à Imai et Matsumoto [162], consiste à choisir $\varphi(x) = t(s(x)^5)$ où s est une bijection secrète linéaire de $(\mathbb{F}_2)^6$ sur $\mathbb{F}_{2^6}$ et t une bijection linéaire secrète de $\mathbb{F}_{2^6}$ sur $(\mathbb{F}_2)^6$. La fonction φ est alors bijective (car $a \mapsto a^5$ est bijective sur $\mathbb{F}_{2^6}$, d'inverse $b \mapsto b^{38}$) et *quadratique* sur $\mathbb{F}_2$. En outre, on peut satisfaire la condition 2 en utilisant l'identité

$$\varphi(v_1 \oplus v_2) = \psi(v_1,v_1) \oplus \psi(v_1,v_2) \oplus \psi(v_2,v_1) \oplus \psi(v_2,v_2),$$

où la fonction ψ est définie par $\psi(x,y) = t(s(x)^4 \cdot s(y))$.

2.4 Application à l'algorithme RSA

La méthode de duplication [313] permet également de sécuriser les implémentations de l'algorithme RSA contre les attaques DPA. Considérons par exemple une implémentation du type «*toujours-multiplier-et-mettre-au-carré*» [2] («*square-and-multiply-always*»), illustrée par l'algorithme 4.

Algorithme 4 : «Toujours-multiplier-et-mettre-au-carré» (à partir du bit de poids fort)

Input: x, n, $d = d_{m-1}2^{m-1} + d_{m-2}2^{m-2} + ... + d_1 2 + d_0$ (avec $d_{m-1} = 1$)
Output: $y_0 = x^d \bmod n$

```
y0 := x
for i = m − 2 down to 0 do
    y0 := y0^2 mod n
    y1 := y0 · x mod n
    y0 := y_{d_i}
end for
Return y0
```

Cette implémentation est sûre contre les attaques SPA. En revanche on constate que les premières valeurs prises par la variable y_0 ne dépendent que de quelques bits de l'exposant secret d. L'*hypothèse fondamentale* (voir chapitre 6, §4.3) est donc vérifiée, rendant possible une attaque

2. C'est l'analogue pour RSA de la méthode «*toujours-additionner-et-doubler*» («*double-and-add-always*») pour les courbes elliptiques (voir chapitre 6, §6.1).

DPA. On peut de fait deviner par exemple les 8 bits de poids fort de d, en déduire la valeur de d_0 qui en résulte (en théorie) à la fin du *neuvième* tour de boucle, et vérifier que l'hypothèse est correcte à l'aide des enregistrements de consommation électrique. Une fois ces 8 premiers bits obtenus, on itère le procédé pour retrouver tous les bits de d par blocs de 8.

La méthode de duplication permet de se prémunir contre cette attaque. En utilisant la fonction $f(v_1,v_2) = v = v_1 \cdot v_2 \bmod n$ pour notre partage de secret, on voit qu'il suffit pour calculer $x^d \bmod n$, de séparer aléatoirement x en (x_1,x_2) tels que $x = x_1 \cdot x_2 \bmod n$, de calculer $y_1 = x_1^d \bmod n$, puis $y_2 = x_2^d \bmod n$, et enfin $y = y_1 \cdot y_2 \bmod n$. Les deux calculs d'*exponentiation modulaire* étant complètement indépendants, on peut les effectuer en parallèle, ou successivement, ou encore de façon entremêlée[3].

3 Conversions entre deux structures algébriques

L'exemple de la méthode de duplication fournit une solution pratique pour protéger contre les attaques DPA les algorithmes qui utilisent essentiellement un seul type d'opération élémentaire (le *ou-exclusif* pour le DES, ou la *multiplication modulaire* pour RSA). Dans ce cas, un seul type de *partage de secret* est nécessaire. En revanche, les algorithmes «hybrides» qui mélangent deux structures algébriques différentes créent une difficulté supplémentaire.

Dans le cas où ces deux opérations correspondent à l'addition et à la multiplication d'un même *corps*, l'utilisation d'un masque additif et d'un masque multiplicatif est possible, et l'existence d'une même structure algébrique sous-jacente assure la compatibilité entre les deux masques (notamment *via* la *distributivité*). C'est notamment ce principe que j'ai utilisé dans [173] pour sécuriser les implémentations de l'algorithme de signature SFLASH.

La taille du corps ne doit toutefois pas être trop petite. En effet, si la probabilité d'obtenir la valeur *zéro* devient non négligeable, une attaque par analyse de consommation électrique redevient possible, ceci malgré l'utilisation des deux masques. Une illustration de ce danger est donnée par l'algorithme AES (qui se place sur le corps $\mathbb{F}_{256}$), où une protection proposée par Akkar et Giraud [301] s'est révélée insuffisante, comme je l'ai montré dans [302] avec M.-L. Akkar.

Par ailleurs dans certains cryptosystèmes coexistent des opérations *booléennes* et des opérations *arithmétiques*. C'est notamment le cas pour les algorithmes IDEA [262] et pour trois des candidats à l'AES : MARS [231], RC6 [286] et TwoFish [289]. Les *fonctions de hachage* comme SHA-1 [317] ou MD5 [319] ont également besoin d'être protégées dans certaines applications, par exemple lorsqu'elles sont utilisées[4] pour construire un MAC (*Message Authentication Code*), ou encore un *générateur pseudo-aléatoire*[5].

C'est pour ces situations que j'ai développé dans [311] une *méthode de conversion* universelle résistant aux attaques DPA (y compris celle décrite [237] au chapitre 6, §5.3).

3. Notons qu'en pratique, le partage de secret initial peut se faire avec $x_1 = r^e \cdot x \bmod n$ et $x_2 = r^{-e} \bmod n$, où r est choisi aléatoirement. De cette façon, $y_2 = x_2^d \bmod n$ est égal à $r^{-1} \bmod n$. On économise donc une exponentiation modulaire, mais en contrepartie on a une *division modulaire* à calculer.

4. Par le biais notamment de la construction HMAC [303] de Bellare, Canetti et Krawczyk, ou de la construction MDx-MAC [318] de Preneel et van Oorschot.

5. L'utilisation d'une fonction de hachage pour construire un générateur pseudo-aléatoire figure par exemple dans la norme ANSI X9.42 [300] Annexe C. On la trouve également comme composante des algorithmes QUARTZ [83, 84], SFLASH [81, 82] et ESIGN [88] (voir chapitre 5, §2.4).

3.1 Implémentation sécurisée de l'algorithme SFLASH

Dans l'algorithme SFLASH (voir chapitre 5, §3.2), la génération d'une signature S repose essentiellement sur le calcul de la fonction

$$S = s^{-1} \circ F^{-1} \circ t^{-1}(Y||R),$$

où Y, R proviennent du message à signer, s et t sont des fonctions affines bijectives et secrètes, et où F^{-1} est définie sur l'extension algébrique $\mathcal{L} = \mathbb{F}_{128^{37}}$ par

$$F^{-1}(B) = B^h, \text{ avec } h = (128^{11}+1)^{-1} \bmod (128^{37}-1).$$

Comme l'ont remarqué Steinwandt, Geiselmann, et Beth [320], une implémentation naïve de SFLASH est potentiellement vulnérable aux attaques par analyse différentielle de consommation électrique. Toutefois, j'ai montré dans [310, 173] que l'on peut obtenir une implémentation DPA-résistante en utilisant deux masques correspondant aux deux opérations sous-jacentes : l'addition et la multiplication sur le corps $\mathcal{L} = \mathbb{F}_{128^{37}}$. La compatibilité entre les deux masques aléatoires est assurée par la propriété de *distributivité* dans le corps.

L'implémentation sécurisée de SFLASH que j'ai obtenue dans [173] est décrite dans la figure 3, où $r \in \mathcal{L}$ désigne le masque *additif*, $\lambda \in \mathcal{L}^*$ le masques *multiplicatif*, S_m et T_m sont les matrices secrètes correspondant aux parties *linéaires* de s^{-1} et t^{-1}, et y est égal à la variable $Y||R$ (déduite du message à signer au moyen de la fonction de hachage SHA-1).

D'un point de vue algorithmique, le calcul de $F(\lambda)$ nécessite *une* multiplication, *quatre* élévations à la puissance 128, et *une* élévation à la puissance 128^7, grâce à l'identité

$$F(\lambda) = \lambda^{128^{11}+1} = \left((((\lambda^{128^7})^{128})^{128})^{128})^{128}\right) \cdot \lambda.$$

Quant au calcul de λ^{-1}, il peut être effectué au moyen d'une q-chaîne d'addition (voir chapitre 5, §3.4) décrite dans l'algorithme 5.

Algorithme 5 : q-chaîne d'addition pour le calcul de λ^{-1} $(= \lambda^{128^{37}-2})$

Input: $\lambda \in \mathcal{L}^*$
Output: $z = \lambda^{-1}$

$z := \lambda$
$z := z^2 \cdot z$
$z := z^{2^2} \cdot z$
$z := z^{2^4} \cdot z$
$z := (z^{128})^2 \cdot z$
$z := ((z^{128})^{128})^4 \cdot z$
$z := ((((z^{128})^{128})^{128})^{128})^{2^4} \cdot z$
$z := z^{2^{64}} \cdot z = (((z^{128^7})^{128})^{128})^2 \cdot z$
$z := z^{2^{128}} \cdot z = ((((((z^{128^7})^{128^7})^{128})^{128})^{128})^{128})^4 \cdot z$
$z := z^{2^2} \cdot \lambda^3$
$z := z^2$
Return z

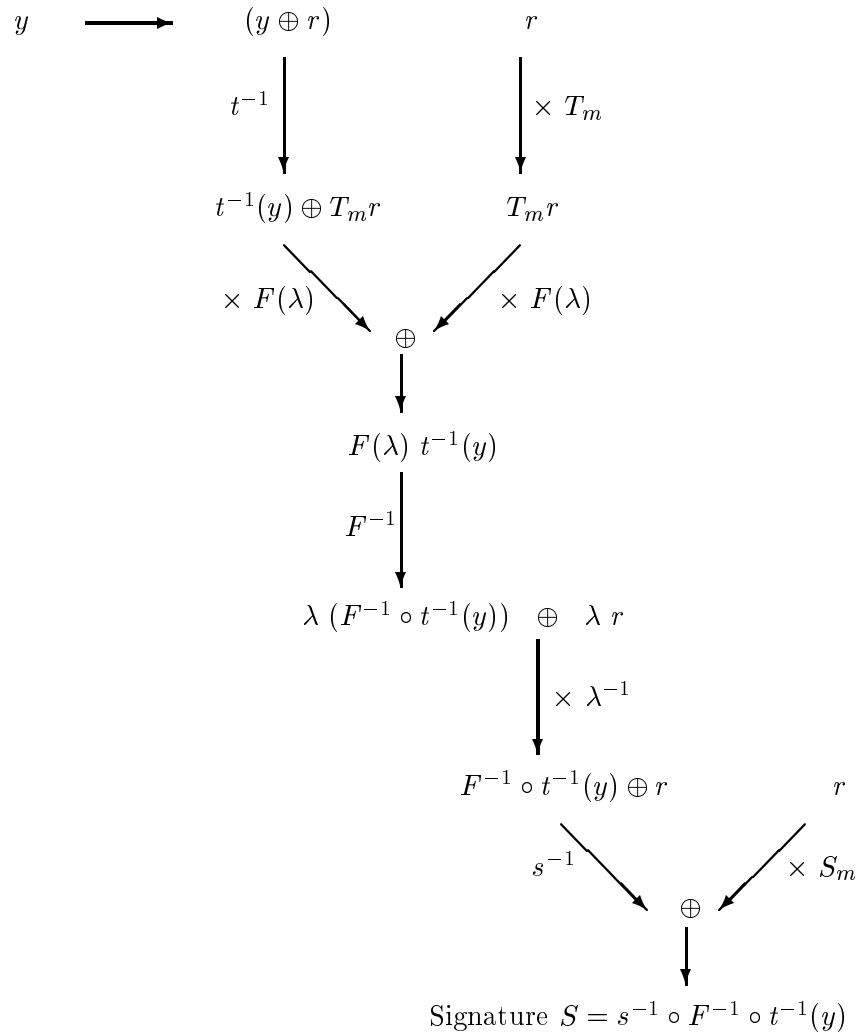


FIG. 3 – *Méthode de double masquage pour protéger SFLASH contre la DPA*

On obtient ainsi une implémentation DPA-résistante de SFLASH, où le temps de calcul d'une signature est approximativement *doublé* par rapport à l'implémentation non protégée (cf chapitre 5). Ce facteur 2 est comparable à ce que l'on obtient pour le DES ou RSA.

3.2 Le problème du zéro pour l'AES

L'algorithme AES, de manière comparable à SFLASH, met en jeu l'addition et la multiplication sur le corps $\mathbb{F}_{256}$. Il est donc naturel d'appliquer la même méthode de *double masquage* que pour SFLASH. C'est ce qu'ont proposé Akkar et Giraud dans [301]. Malheureusement, une attaque par analyse de consommation est encore possible malgré cette contre-mesure, comme je l'ai montré [302] avec M.-L. Akkar[6].

Chaque tour de l'AES comprend 16 *boîtes-S* identiques (de 8 bits sur 8 bits), qui transforment l'entrée A en son inverse $B = A^{-1}$ dans le corps $\mathbb{F}_{256}$ (plus une transformation affine sur $(\mathbb{F}_2)^8$ que l'on peut omettre ici). Si l'entrée A est donnée sous forme masquée (par un masque additif r), la méthode proposée dans [301] pour obtenir la sortie $B = A^{-1}$ (également masquée par r) est décrite dans la figure 4. Cette technique utilise un second[7] masque multiplicatif λ.

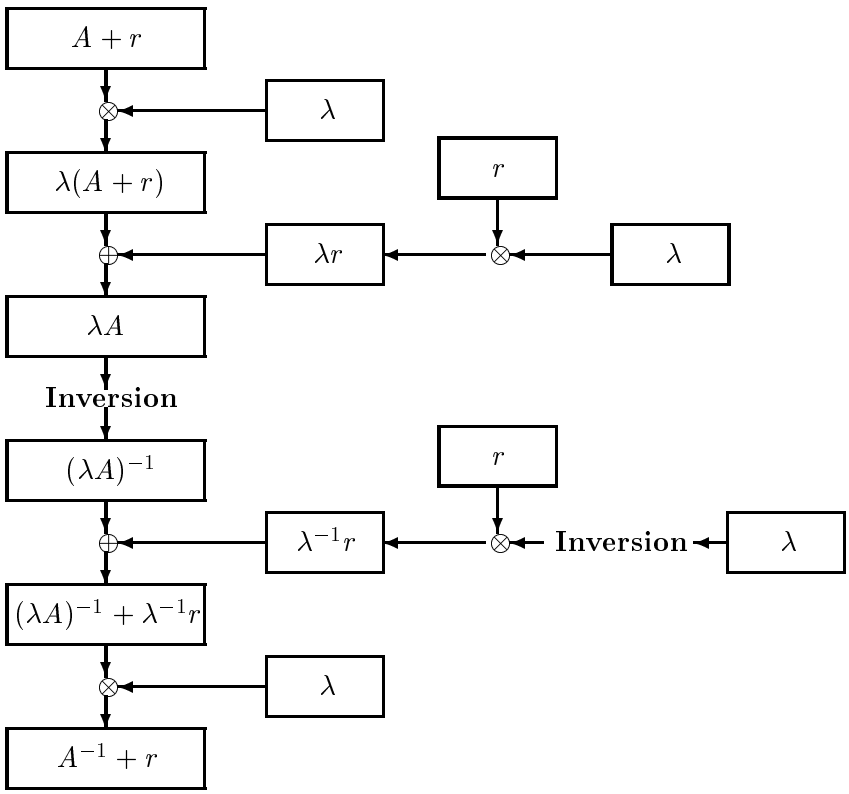


FIG. 4 – *Boîte-S de l'AES : inversion dans le corps* $\mathbb{F}_{256}$ *avec double masquage*

L'idée de l'attaque DPA que j'ai proposée dans [302] est la suivante. L'inversion dans $\mathbb{F}_{256}$ opère sur la valeur A masquée par λ. Or si A est nul, λA l'est aussi (quelle que soit la valeur

6. Cette attaque a été aussi trouvée indépendamment par Golić et Tymen [309].

7. Notons que Trichina, De Seta et Germani ont proposé dans [321] d'utiliser le *même masque* pour les deux opérations. Mais cette idée est insuffisante, même contre la DPA usuelle.

du masque multiplicatif). Par ailleurs, par construction de l'AES, la valeur A est obtenue par un *ou-exclusif* entre un octet du message et 8 bits de la clé secrète. On peut alors chiffrer plusieurs fois le même message M (ou plusieurs messages différents, mais ayant le i-ème octet en commun) et vérifier si, lors de la i-ème inversion, les traces de consommation présentent une variance importante. Si c'est le cas, on peut en déduire que la valeur λA est non nulle, et par conséquent que les 8 bits de clé sont différents de ceux de la clé. On recommence avec un autre choix d'octet pour le message, jusqu'à obtenir une variance anormalement faible dans les traces de consommation, qui indique que l'on a bien détecté les 8 bits de clé.

En réitérant le procédé, on peut retrouver les 128 bits de la clé de l'AES. Cela montre que le principe de double masquage n'est pas suffisant pour protéger l'algorithme contre la DPA [8].

3.3 Une méthode de conversion universelle

Pour les algorithmes qui mélangent l'addition et la multiplication d'un même *corps*, nous avons vu dans les deux paragraphes précédents qu'une méthode de *double masquage* est envisageable en tant que contre-mesure contre la DPA. En revanche, pour les cryptosystèmes utilisant à la fois des opérations *booléennes* et *arithmétiques*, on ne peut plus tirer parti d'une même structure algébrique sous-jacente. Il reste néanmoins souhaitable de conserver si possible un seul type de masque, commun aux deux opérations.

Comme nous l'avons vu au chapitre 6, le problème de conversion (entre représentations booléennes et arithmétiques) qui en résulte est délicat, et beaucoup de méthodes apparemment solides laissent malgré tout «filtrer» des informations partielles, qui permettent encore des attaques par analyse de consommation. C'est pourquoi j'ai mis au point dans [311] une méthode générale de conversion, avec la preuve formelle qu'elle est immunisée contre la DPA.

Les deux algorithmes «Booléen-vers-Arithmétique» et «Arithmétique-vers-Booléen» s'appuient sur un nombre très restreint d'opérations élémentaires : le *ou-exclusif* (XOR), le *et-logique* (AND), des *soustractions* et le *décalage logique à gauche* (*logical shift left*). Le premier algorithme en utilise un nombre *constant* (égal à 7), et le second un nombre proportionnel à la taille K des registres du microprocesseur (précisément $5K + 5$).

L'algorithme «Booléen-vers-Arithmétique»

Soit $I = (\mathbb{F}_2)^K$ avec $K \geq 1$. On considère la fonction $\Phi_{x'} : I \to I$, définie par

$$\Phi_{x'}(r) \equiv (x' \oplus r) - r \bmod 2^K.$$

On peut alors montrer le théorème suivant (voir [311] pour une démonstration) :

Theorème 7

$$\Phi_{x'}(r) = x' \oplus \bigoplus_{i=1}^{K-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \overline{x'}) \Big) \wedge (2^i x') \wedge (2^i r) \Big],$$

où $\overline{x'}$ désigne le complément à un de x', et $\wedge$ l'opérateur et-logique (AND).

On en déduit facilement le résultat suivant, assez surprenant :

Corollaire 7.1 *La fonction $\Phi_{x'}$ est affine sur $\mathbb{F}_2$.*

8. Notons que Golić et Tymen [309] ont proposé une solution à ce problème, qui consiste à se placer dans une extension algébrique de $\mathbb{F}_{256}$, suffisamment grande pour que la probabilité de tomber sur la valeur *zéro* au cours du calcul devienne négligeable.

Par conséquent, la fonction $\Psi_{x'} = \Phi_{x'} \oplus \Phi_{x'}(0)$ est *linéaire* sur $\mathbb{F}_2$, d'où l'identité

$$\Psi_{x'}(r) = \Psi_{x'}(\gamma \oplus (r \oplus \gamma)) = \Psi_{x'}(\gamma) \oplus \Psi_{x'}(r \oplus \gamma),$$

valable quels que soient r, x' et γ dans I.

Corollaire 7.2 *Pour toute valeur $\gamma \in I$, si on note $A = (x' \oplus r) - r$, on a aussi*

$$A = [(x' \oplus \gamma) - \gamma] \oplus x' \oplus [(x' \oplus (r \oplus \gamma)) - (r \oplus \gamma)].$$

Cela permet de convertir un masquage booléen en masquage arithmétique. L'algorithme 6 requiert deux variables auxiliaires (T et Γ), a besoin d'une valeur aléatoire, et utilise 7 opérations élémentaires (5 *ou-exclusifs* et deux soustractions).

Algorithme 6 : «Booléen-vers-Arithmétique»

Input: (x',r) tels que $x = x' \oplus r$

Output: A tel que $x = A + r$

Initialiser Γ avec une valeur aléatoire γ
$T := x' \oplus \Gamma$
$T := T - \Gamma$
$T := T \oplus x'$
$\Gamma := \Gamma \oplus r$
$A := x' \oplus \Gamma$
$A := A - \Gamma$
$A := A \oplus T$
Return A

Par ailleurs, si on suppose que γ est choisi de façon aléatoire avec une distribution uniforme sur I, il est facile de vérifier que toutes les variables intermédiaires de l'algorithme sont aussi uniformément distribuées sur I. L'algorithme 6 résiste donc aux attaques DPA de manière prouvée.

L'algorithme « Arithmétique-vers-Booléen»

La conversion dans ce sens est plus complexe. On utilise la formule de récurrence suivante (voir [311] pour une démonstration) :

Theorème 8 *Si on note $x' = (A + r) \oplus r$, on a aussi $x' = A \oplus u_{K-1}$, où u_{K-1} est donné par la formule de récurrence suivante :*

$$\begin{cases} u_0 = 0 \\ \forall k \geq 0,\ u_{k+1} = 2[u_k \wedge (A \oplus r) \oplus (A \wedge r)]. \end{cases}$$

Grâce au changement de variable $t_k = 2\gamma \oplus u_k$, on en tire la conséquence suivante :

Corollaire 8.1 *Pour toute valeur $\gamma \in I$, si on note $x' = (A+r) \oplus r$, on a aussi $x' = A \oplus 2\gamma \oplus t_{K-1}$, où t_{K-1} est donné par la formule de récurrence suivante :*

$$\begin{cases} t_0 = 2\gamma \\ \forall k \geq 0,\ t_{k+1} = 2[t_k \wedge (A \oplus r) \oplus \omega], \end{cases}$$

dans laquelle $\omega = \gamma \oplus (2\gamma) \wedge (A \oplus r) \oplus A \wedge r$.

Cela permet de convertir un masquage arithmétique en masquage booléen. L'algorithme 7 requiert trois variables auxiliaires (T, Ω et Γ), une valeur aléatoire, et utilise $(5K+5)$ opérations élémentaires ($(2K+4)$ *ou-exclusifs*, $(2K+1)$ *et-logiques* et K *décalages logiques à gauche*).

Algorithme 7 : «Arithmétique-vers-Booléen»

Input: (A,r) tels que $x = A + r$
Output: x' tel que $x = x' \oplus r$

Initialiser Γ avec une valeur aléatoire γ
$T := 2\Gamma$
$x' := \Gamma \oplus r$
$\Omega := \Gamma \wedge x'$
$x' := T \oplus A$
$\Gamma := \Gamma \oplus x'$
$\Gamma := \Gamma \wedge r$
$\Omega := \Omega \oplus \Gamma$
$\Gamma := T \wedge A$
$\Omega := \Omega \oplus \Gamma$
for $k = 1$ to $K - 1$ **do**
  $\Gamma := T \wedge r$
  $\Gamma := \Gamma \oplus \Omega$
  $T := T \wedge A$
  $\Gamma := \Gamma \oplus T$
  $T := 2\Gamma$
end for
$x' := x' \oplus T$
Return x'

Comme pour l'algorithme 6, on peut établir une preuve de sécurité de la conversion, face aux attaques DPA. Si on suppose que γ est choisi de façon aléatoire avec une distribution uniforme sur I, il est facile de voir que certaines variables intermédiaires (comme γ, $\gamma \oplus r$ et $\gamma \oplus u_{k-1}$) sont uniformément distribuées sur I, ou bien (telles 2γ et $2\gamma \oplus u_k$) sur un sous-ensemble de I. D'autres ont une distribution qui dépend de r mais pas de A (ainsi $\gamma \oplus \gamma \wedge r$ et $(2\gamma \oplus u_{k-1}) \wedge r$) ou bien de A mais pas de r (comme $2\gamma \oplus A$ et $(2\gamma \oplus u_{k-1}) \wedge A$).

Le cas des autres variables intermédiaires est plus délicat, et nécessite le théorème suivant :

Theorème 9 *Pour toute valeur $\delta \in I$, la fonction suivante est bijective :*

$$\Theta_\delta : \begin{cases} I \to I \\ \gamma \mapsto \gamma \oplus (2\gamma) \wedge \delta. \end{cases}$$

On peut en déduire (voir [311]) que les variables intermédiaires $\gamma \oplus 2\gamma \oplus A$, $\gamma \oplus (2\gamma) \wedge r \oplus A \wedge r$, $\omega = \gamma \oplus (2\gamma) \wedge (A \oplus r) \oplus A \wedge r$ et $\gamma \oplus (2\gamma) \wedge A \oplus u_{k-1} \wedge r \oplus A \wedge r$ sont uniformément distribuées sur I, alors que la distribution de $(\gamma \oplus 2\gamma \oplus A) \wedge r$ dépend de r mais pas de A.

3.4 Application pratique

Les algorithmes de conversion ainsi établis fournissent une contre-mesure générique pour tous les algorithmes utilisant des opérations booléennes et arithmétiques. Un des avantages de la méthode est qu'elle traite le problème des attaques par analyse de consommation électrique au niveau le plus bas des instructions élémentaires du microprocesseur. On peut ainsi imaginer un procédé (analogue à un *compilateur*) qui intègre automatiquement le masquage de toutes les variables sensibles dans l'implémentation d'un algorithme cryptographique. On va ainsi dans le sens d'une formalisation des attaques physiques [9].

La méthode que j'ai obtenue dans [311] est très efficace pour transformer un masquage booléen en masquage arithmétique. En revanche, la complexité de la transformation réciproque est linéaire en la taille des registres du microprocesseur. Cela limite son usage dans le cas d'une implémentation logicielle [10]. En revanche, cela ouvre la voie à des contre-mesures génériques pouvant être prises en compte dès la mise au point de l'architecture du microprocesseur lui-même, les algorithmes de conversion étant alors insérés de manière câblée.

4 Les attaques par analyse différentielle d'ordre supérieur

4.1 Une généralisation de la DPA

Comme nous l'avons vu au chapitre 6, le principe de l'*analyse différentielle de consommation* consiste à comparer des valeurs mesurées lors du fonctionnement du *véritable* dispositif physique (par exemple la carte à microprocesseur) avec des valeurs calculées grâce à un modèle *hypothétique* de ce dispositif (les hypothèses portant entre autres sur la nature de l'implémentation, et sur une partie de la clé secrète). En comparant ces deux ensembles de valeurs, on s'efforce ensuite de retrouver tout ou partie de la clé secrète.

Si le modèle théorique prédit *une seule* valeur, par exemple la consommation électrique du dispositif pour un seul instant t, on dit que l'attaque par analyse différentielle est du *premier ordre*. En revanche, si le modèle est capable de prédire *plusieurs* valeurs, on parle d'attaque par analyse différentielle d'*ordre supérieur*. Si on ne précise pas (ce qui est notre cas jusqu'ici), le terme DPA désigne implicitement une attaque du premier ordre.

Les attaques par *analyse différentielle d'ordre supérieur* (*High-Order Differential Power Analysis* – ou HO-DPA – en anglais), déjà suggérées par Kocher, Jaffe et Jun [260, 261] (voir aussi [307]), ont été formalisées par Messerges dans [315]. De manière générale, et dans le même esprit qu'au chapitre 6 (paragraphe 4.3), j'ai formulé dans [302] une condition nécessaire et suffisante pour qu'une attaque DPA d'ordre n s'applique :

Hypothèse fondamentale à l'ordre n : *Il existe un n-uplet de variables intermédiaires, qui apparaissent au cours de l'exécution de l'algorithme, tel que la connaissance d'un petit nombre de bits de clé (en pratique moins de 32 bits) permet de décider si deux entrées (respectivement deux sorties) de l'algorithme donnent ou non la même valeur pour une fonction connue de ces n variables.*

Dans l'article [302], écrit avec M.-L. Akkar, j'ai étudié les impacts des attaques HO-DPA sur les implémentations d'algorithmes cryptographiques, notamment en montrant que les contre-mesures habituelles pour la DPA ne sont pas suffisantes contre ce nouveau type de menace. Par

9. À ce sujet, on peut consulter l'article récent [316] de Micali et Reyzin.
10. Toutefois, quelques améliorations ont été récemment apportées par Coron et Tchulkine dans [306].

ailleurs, nous avons proposé une contre-mesure générique pour contrecarrer les attaques d'ordre supérieur, en l'illustrant plus particulièrement sur l'exemple du DES.

4.2 Insuffisance des contre-mesures «classiques»

La méthode de duplication

Comme nous l'avions remarqué dans [313], une méthode s'appuyant sur un *partage de secret* en n parties est par nature vulnérable face à une analyse différentielle d'ordre n.

Considérons par exemple le cas $n = 2$. Chaque variable V de l'implémentation initiale devient un couple de variables (V_1,V_2) dans la nouvelle implémentation. Par construction, chacune de ces deux variables se comporte de façon aléatoire, si bien qu'aucune corrélation ne peut être détectée entre la consommation électrique qu'elle engendre et la clé secrète. En revanche, une corrélation existe entre le *couple* (V_1,V_2) et la variable V, et par conséquent entre (V_1,V_2) et la clé secrète. En tenant alors compte de la consommation électrique *conjointe* des instructions manipulant ces deux variables, il devient possible d'attaquer l'implémentation [11] : il s'agit ici d'une attaque DPA d'ordre 2.

En pratique, la difficulté réside dans la localisation des n instants à considérer dans les traces de consommation. Si la méthode de duplication (ou son extension à l'ordre n) n'est pas appliquée avec suffisamment de soin, il peut être facile de détecter les instants t_1, ..., t_n correspondant aux n variables du partage de secret (notamment si on repère n parties identiques dans le calcul). Par ailleurs, on suppose même dans certains cas que l'attaquant a accès aux moindres détails de l'implémentation : c'est notamment l'hypothèse faite lors de *certifications sécuritaires* de haut niveau dans les schémas ITSEC [12] ou Critères Communs [314]. Dans ces situations, la partie analytique de l'attaque (consistant à rechercher des corrélations parmi toutes les traces de consommation obtenues) peut se faire en temps constant. En revanche, face à une implémentation plus sophistiquée, l'attaquant doit déterminer les instants t_1, ..., t_n par *recherche exhaustive*, de complexité *exponentielle* en n (voir [313, 304]).

La méthode de «transformation dépendant du masque»

Introduite par Akkar et Giraud [301] sous le nom de «*Transformed Masking Method*», cette technique consiste à masquer toutes les variables intermédiaires par la *même* valeur aléatoire. Pour les schémas faisant appel à des boîtes-S (comme le DES), cela nécessite de recalculer – à chaque exécution de l'algorithme – de nouvelles tables qui tiennent compte du masque. On a donc deux phases : la première calcule les boîtes-S en fonction d'un masque aléatoire, la seconde exécute l'algorithme de façon classique, mais avec les nouvelles tables (l'entrée étant préalablement masquée, et la sortie étant démasquée à la fin).

À nouveau, la possibilité d'une attaque DPA d'ordre deux avait été envisagée par les auteurs de la méthode [301]. Il suffit de remarquer que le *ou-exclusif* de l'entrée et de la sortie d'une boîte-S (dans le cas du DES par exemple) ne dépend pas de la valeur aléatoire du masque, et est prédictible si on connaît quelques bits de la clé secrète.

Les mêmes remarques que pour la méthode de duplication s'appliquent au sujet de la complexité de l'attaque en pratique. Notons que pour le cas du DES, nous avons décrit dans [302] une attaque plus simple, ne nécessitant pas de connaître la position précise dans le temps de l'entrée et de la sortie d'une boîte-S à l'intérieur d'un tour, mais uniquement la position *globale* des tours.

11. Voir notre article [302] pour une attaque d'ordre 2 détaillée sur le DES.
12. Information Technology Security Evaluation Criteria

Dans l'*attaque par superposition* (*Superposition Attack* en anglais), l'attaquant superpose les traces du premier et du 16-ième tours du DES. La courbe obtenue cumule les consommations dues à la sortie d'une des boîte-S du premier tour, et à la sortie de la boîte-S analogue du 16ème tour. Or le *ou-exclusif* de ces deux valeurs ne dépend que des entrée et sortie du DES (supposées connues), et de 12 bits de clé (6 pour chacune des deux boîtes-S). Cette attaque d'ordre 2 «simplifiée» est donc applicable, à condition qu'il y ait une corrélation entre le *ou-exclusif* de deux variables et la somme de leurs consommations associées[13].

4.3 Une contre-mesure pour les attaques d'ordre supérieur

Dans [302], j'ai proposé avec M.-L. Akkar la méthode dite du «masque unique» (*Unique Masking Method*), qui permet d'obtenir une implémentation protégée contre les attaques par analyse différentielle d'ordre n, quelle que soit la valeur de n.

Décrivons l'idée générale dans le cas du DES. Le masque unique est une valeur aléatoire α de 32 bits. De cette valeur sont déduites deux séries de 8 boîtes-S, notées $\tilde{S}_1$ et $\tilde{S}_2$, définies par

$$\begin{cases} \forall x \in (\mathbb{F}_2)^{48}, & \tilde{S}_1(x) = S(x \oplus E(\alpha)) \\ \forall x \in (\mathbb{F}_2)^{48}, & \tilde{S}_2(x) = S(x) \oplus P^{-1}(\alpha) \end{cases}$$

où S désigne les 8 boîtes-S usuelles du DES, E est la fonction d'expansion, et P la permutation intervenant classiquement après les boîtes-S.

Désignons par f_{K_i} $(1 \leq i \leq 16)$ les fonctions du schéma de Feistel du DES (avec les boîtes-S usuelles), et $\tilde{f}_{1,K_i}$ (respectivement $\tilde{f}_{2,K_i}$) la fonction analogue avec les boîtes $\tilde{S}_1$ (respectivement $\tilde{S}_2$). Pour les tours construits avec ces fonctions, on dénombre alors cinq configurations possibles, décrites dans la figure 5. Les traits pleins correspondent à des données non masquées, les pointillés à des valeurs masquées.

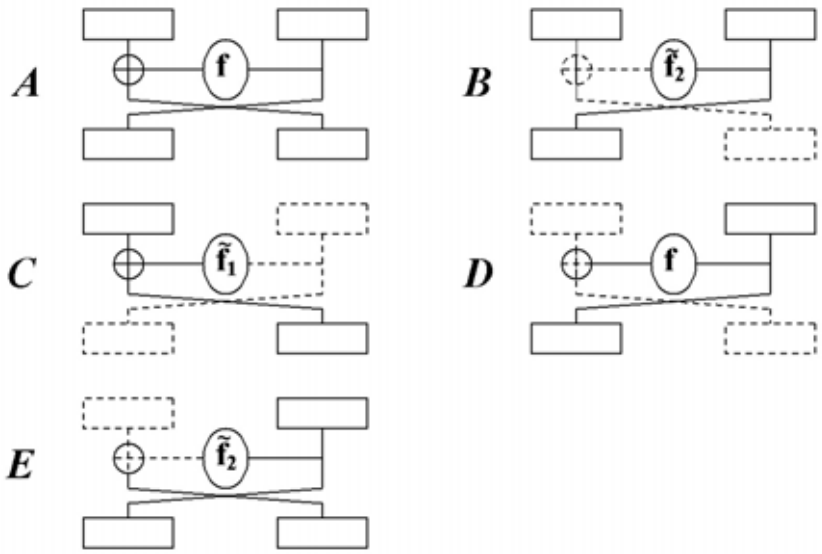


FIG. 5 – *Les cinq possibilités pour les tours du DES*

13. Voir notre article [302] pour une description détaillée de l'attaque par superposition.

Pour rester fonctionnellement compatible avec le calcul du DES, l'enchaînement des tours doit respecter certaines règles, que l'on peut synthétiser par un *automate fini*, représenté dans la figure 6. Les états initiaux correspondent à des entrées non masquées (A ou B), les états finaux à des sorties non masquées (A ou E). Par exemple $BCDCDCEBCDCDCDCE$ est un enchaînement valide.

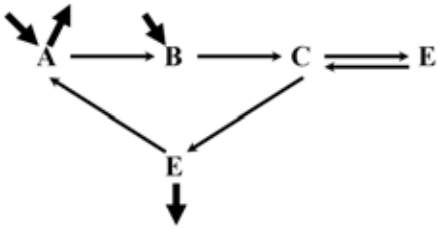


FIG. 6 – *Enchaînements valides pour les tours*

Pour se protéger contre les attaques par analyse différentielle, les valeurs dépendant de moins de 36 bits de clé doivent être masquées par α. Cela rajoute des contraintes sur les tours : les trois premiers doivent être de la forme BCD ou BCE, et symétriquement les trois derniers de la forme BCE ou DCE. L'enchaînement $BCDCDCEBCDCDCDCE$ satisfait ces conditions.

Il est facile de voir que la méthode du masque unique garantit une résistance en pratique aux attaques DPA du premier ordre[14]. Si on se place dans le cadre d'une attaque à clair et chiffré connus, il faut tenir compte en outre de l'*attaque par superposition*, ce qui oblige à choisir deux masques α_1 et α_2 et une implémentation correspondant à l'enchaînement

$$B_{\alpha_1} C_{\alpha_1} D_{\alpha_1} C_{\alpha_1} D_{\alpha_1} C_{\alpha_1} E_{\alpha_1} B_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} E_{\alpha_2}$$

On vérifie alors que, parmi les variables dépendant d'au plus 36 bits de la clé secrète, on n'en trouve jamais deux qui soient masquées par la même valeur. Par conséquent, l'hypothèse fondamentale à l'ordre n n'est pas vérifiée. Cela garantit une résistance aux attaques par analyse différentielle d'ordre quelconque.

La méthode est très flexible et plusieurs variantes sont possibles. Ainsi si on se restreint aux attaques à clair connu (ou à chiffré connu, mais pas les deux), on peut se contenter d'un seul masque. Si en revanche on souhaite interdire la réapparition du masque (même sur les valeurs dépendant de plus de 36 bits de clé), on peut utiliser la combinaison suivante :

$$B_{\alpha_1} C_{\alpha_1} E_{\alpha_1} AAAAAAAAAB_{\alpha_2} C_{\alpha_2} E_{\alpha_2}$$

Par ailleurs, pour accroître encore la résistance de l'implémentation, on peut ajouter deux masques supplémentaires, ce qui permet de masquer toutes les variables, y compris celles qui dépendent de 56 bits de clé.

La méthode du masque unique est actuellement la seule méthode de protection publiée contre les attaques par analyse différentielle d'ordre supérieur. Elle présente l'avantage de ne pas modifier la structure des implémentations classiques, la différence essentielle est la phase de génération

14. Si chaque trace est stockée sur 128 octets, la recherche exhaustive sur 36 bits nécessite une mémoire de 2 Tera-octets.

des tables[15]. Par ailleurs, les performances restent acceptables. Une implémentation du DES, incluant la méthode du masque unique (ainsi que des protections contre les attaques SPA et DFA), a été réalisée sur une carte à microprocesseur 8-bits[16], donnant un temps de 38 ms pour une fréquence d'horloge de 10 MHz.

15. Nous décrivons dans [302] plusieurs méthodes pour la génération des boîtes-S à partir du masque α.
16. Sur une puce ST19. Le code occupe 3 Ko en ROM, et nécessite 749 octets de RAM.

Conclusion et perspectives

Pas de plus ample chant que le chant qui finit.
Pas de plus douce main que la main qui s'enfuit.

Pierre Jean Jouve, *Diadème* (1949)

La carte à microprocesseur est un cadre d'application de la cryptologie qui met en lumière le caractère véritablement interdisciplinaire de ce domaine. En effet, si une approche puriste tend à réserver aux théoriciens la partie spécifiquement cryptographique et mathématique, pour laisser aux ingénieurs la tâche d'implémenter les algorithmes obtenus, la réalité montre qu'une véritable sécurité ne peut être atteinte qu'en tenant compte de tous ces aspects dès la spécification des protocoles.

En particulier, deux considérations s'imposent aujourd'hui. Tout d'abord, de nombreux schémas cryptographiques ne sont utilisables que s'il mènent à des implémentations efficaces. Cela a motivé la proposition et l'analyse de nouvelles hypothèses calculatoires, utilisant la notion de polynômes multivariables, avec l'idée de mettre en évidence des problèmes qui deviennent très rapidement difficiles quand la taille des paramètres augmente [168, 141, 142, 160].

Comme application de ces résultats, plusieurs cryptosystèmes nouveaux, particulièrement bien adaptés aux contraintes spécifiques des cartes à microprocesseur [168, 141, 142, 169, 137]. La mise au point de méthodes générales de cryptanalyse [168, 141, 142, 137, 160, 155] a permis de mettre à l'épreuve les nombreuses possibilités, pour obtenir une confiance accrue dans les algorithmes qui y résistent.

En particulier, on a obtenu deux algorithmes de signature électronique très intéressants [81, 83, 173], l'un produisant des signatures extrêmement courtes, l'autre fournissant les calculs de signature les plus rapides connus actuellement sur une carte à microprocesseur, ceci sans avoir recours à un co-processeur cryptographique. On ouvre ainsi la voie à des applications nouvelles, qui étaient jusque là très coûteuses, voire impossibles. Une voie de recherche ouverte consiste à voir dans quelle mesure les techniques de cryptographie multivariables peuvent mener également à des cryptosystèmes basés sur l'identité, ou encore à des protocoles de *traitor tracing*.

Certains problèmes restent non résolus. En particulier, dans la recherche d'alternatives à l'algorithme RSA, on ne connaît pas de permutation à sens unique avec trappe s'appuyant sur autre chose que l'arithmétique modulaire. Un tel objet aurait de nombreuses applications, notamment pour le chiffrement asymétrique. Dans une autre direction, l'analyse des protocoles

d'infrastructure à clé publique montre la nécessité de mettre au point des algorithmes de signature efficaces dans les cartes à puce tout en ayant des clés publiques de taille raisonnable.

Une deuxième spécificité des cartes à microprocesseur réside dans leur vulnérabilité face à des attaques physiques. La possibilité d'analyser les modifications physiques provoquées par le fonctionnement du microprocesseur, ou bien d'agir sur ces paramètres physiques pour influencer la carte, est inhérente au statut de dispositif embarqué qu'a la carte.

Dans [313, 237, 244], j'ai contribué au développement de ce domaine, en analysant les conditions de possibilité des attaques par analyse physique de la consommation électrique (ou des rayonnements électromagnétique), notamment dans le cas d'algorithmes symétriques (DES, AES) ou asymétriques (RSA, ECDSA) classiques. Une compréhension des principes fondamentaux de ces attaques est en effet nécessaire pour pouvoir espérer les contrer de façon théorique et prouvée.

Cette formalisation m'a permis de mettre en place des contre-mesures [313, 311, 173, 302] qui s'appliquent à une large classe d'algorithmes cryptographiques. La méthode de duplication, la méthode du masque unique, ainsi qu'une technique générale de conversion entre des représentations booléennes et arithmétiques des données dans le microprocesseur, ouvrent la voie à un traitement systématique, et même automatisable dans une certaine mesure, des attaques par analyse physique.

Une des retombées potentielles réside également dans la conception d'architectures matérielles pour les circuits électroniques, qui intègrent ces méthodes de protection de façon coordonnée avec le logiciel. Les attaques par analyse physique sont nées de l'analyse de corrélations imprévues entre les propriétés mathématiques des algorithmes et le comportement physique des portes logiques sous-jacentes, et les contre-mesures efficaces pourraient bien résulter de la même approche pluridisciplinaire.

Termes et abréviations

– Symboles –

2R 57
3-SAT 34

– A –

ACE-Sign 31
Advanced Encryption Standard 15
AES 15, 82, 84, 85, 94, 98, 101, 102
algorithme d'Euclide 18, 21, 22
algorithme de Berlekamp-Rabin 70
algorithme du syndrome 58
analyse du signal 94
AND 102
anonymat 14
APDU 30
attaque active 81
attaque invasive 80
attaque passive 82
attaque semi-invasive 81
attaques à message connu 70, 72
attaques par collision 84
attaques physiques 14, 80
authenticité 14
authentification 14, 16
automate fini 108

– B –

baby-step giant-step 25
bases de Gröbner 62, 72
boîte-S 85, 95, 101, 106
broadcasting 14
bus de données 80

– C –

C^* 50, 53, 54, 56, 60
C^*_- 50, 53, 65, 71, 72
CAD 30
Camellia 15
carte à logique câblée 28
carte à mémoire 28
carte à microcalculateur 28
carte à microprocesseur 28, 80, 81
CDH 66
certification sécuritaire 106
chaîne d'addition 75, 89
chaîne d'addition-soustraction 84
Chained Patarin Construction 70
chiffrement 14
clause 34
CLK 28
CMOS 82
code de Gray 76, 77
collision attacks 84
COMP128-1 84
compilateur 82, 105
complexité additive 75
compression de données 75
confidentialité 14
connecteur 30
contre-mesures 80
contrôle d'accès 14
coprocesseur 29
corps fini 25
courants de Foucault 82
courbes ABC 89
courbes elliptiques 17, 25, 88
courbes hyperelliptiques 92
CPC 70
crible algébrique général 18
Critères Communs 106
CRT 20, 83
cryptanalyse 14, 79
crypto-processeur 29, 77
cryptographie 14, 80
cryptographie à clé publique 15
cryptographie à clé secrète 14
cryptographie asymétrique 15
cryptographie conventionnelle 14
cryptographie symétrique 14
cryptologie 13, 79
cryptosystème de McEliece 66

– D –

D^* 50–53
décalage logique à gauche 102
Data Encryption Standard 14
DDH 66
décalages temporels 94
décodage de syndrome 66
DES 14, 82–86, 94, 95, 98, 101, 106, 107
DFA 83, 109
Differential Fault Analysis 83
Differential Power Analysis 83, 84
Digital Signature Algorithm 24
distributivité 98
division modulaire 98
DP 54
DPA 83–89, 93–95, 97–99, 101, 102, 105
DSA 17, 24, 25, 31, 66, 83

– E –

E^* 50, 52
ECDSA 25, 31, 66, 77
échange de clés 15
ECNR 26
EEPROM 29, 82
EPROM 82
ESIGN 31, 70, 98
exponentiation modulaire 98

– F –

F_5 72
$F_5/2$ 72
FDH 23
FDH-D 70, 72
Field Programmable Gate Arrays 80
FLASH 60, 62
fonction de hachage 21, 23, 98
fonctions de De Jonquières 57, 60
forge existentielle 70, 72
forme canonique 54, 61
forme de Montgomery 88
forme de Weierstraβ 89, 91
forme polaire 50, 53
FPGA 80
Full Domain Hash 23
FXL 72

– G –

General Number Field Sieve 18
générateur pseudo-aléatoire 69, 70, 72, 98
GI 39, 41
glitch 81
GND 28
GNFS 18
GQ 31
GQ2 31
gradient 56
GSM 84

– H –

half-duplex 29
HFE 44, 53, 59, 65
HFEV$^-$ 67, 70
High-Order Differential Power Analysis 94, 105
HMAC 98
HO-DPA 94, 105

– I –

I/O 29
IEEE P1363 ..... 25, 26
Inferential Power Analysis 94
injection de fautes 81
intégrité 14
IP 31, 37–41, 43
IPA 94
isogénie 92
Isomorphisme de Graphes 39, 41
Isomorphisms of Polynomials 31
ITSEC 106

– L –

lecteur de carte 30
littéral 34
LLL 19, 21, 22
logarithme discret (problème du) 23, 24
loi de Moore 18

– M –

MAC 17, 98
machine virtuelle 82
malléabilité 70
MARS 86, 98
MD5 98
MDx-MAC 98
mémoire Flash 82
Message Authentication Code 17, 98
méthode de duplication 93
méthode des sous-matrices 59
méthode du noyau 59, 60

micro-module 28
Minimal Weight 45
MinRank 44, 45, 58, 60
Mips 18
modèle de Markov 84
MP 37, 39, 41–43
MQ 60
MQ^2 ..... 63, 64
multiplicateur de fréquence 29
multiplication modulaire 98
multiplication scalaire 84, 88

– N –

non-équivalence de code 44
non-forgeabilité forte 70
non-isomorphisme de graphes 44
non-malléabilité 70
non-répudiation 14, 16
non-résiduosité quadratique 44
NTRUSign 31, 77

– O –

OAEP 21
OAEP+ 21
Oil and Vinegar 55
Optimal Asymmetric Encryption Padding . 21
oracle aléatoire (modèle de l') ... 21, 23–25, 66

– P –

padding 22
paradoxe des anniversaires 61, 70
partage de secret 94, 106
permutations à sens unique avec trappe ... 50
permutations birationnelles 44, 45, 59
PKP 31
poids de Hamming 84
point à l'infini 90
polarisation 50
preuve interactive 44
preuves de sécurité 14, 80
Probabilistic Signature Scheme 23
problème de dégénérescence 54
protocole d'Arthur-Merlin 43
PSS 23
puce 28
puzzles de Merkle 15

– Q –

q-chaîne d'addition 75, 99
QUARTZ 31, 65–67, 70, 72, 98

– R –

RAM 29, 82, 97
Rank Distance Coding 45
RC5 65
RC6 ..... 65, 86, 98
REACT 22
recherche exhaustive 106
redondance additive 22
redondance multiplicative 22
réduction de Gauss 52, 54, 55, 58, 63, 64
relinéarisation 61
représentation obscure (principe de) ... 50, 57, 60, 97
résidu quadratique 64
RNG 29
ROM 29
RSA 15, 17, 31, 66, 77, 82–86, 97, 98, 101
RST 29

– S –

SAEP+ 22
S-box 85
schéma de Feistel 107
sécurité sémantique 20–22
Self Programmable One-chip Microprocessor 28
SFLASH .31, 53, 60, 62, 65, 66, 70–72, 77, 94, 98, 99, 101
SHA-1 ..... 98, 99
signature 16
signature électronique 14
signature de consommation 94
Simple Power Analysis 84
Small Primes 31
SP 31
SPA 84, 88, 89, 97, 109
spike 81
SPOM 28
square and multiply 52, 74, 75
SRAM 82
SSL 82
symbole de Legendre 91
système d'exploitation 29
systèmes embarqués 80
systèmes sous-définis 61
systèmes sur-définis 61
système d'exploitation 82

– **T** –
tenseur tridimensionnel 41
théorème des restes chinois 20
théorie de la complexité 15, 33, 80
théorie des invariants 55
timing attacks 82, 88
trace 84
traitor tracing 14
Transformed Masking Method 106
trappe 60
Triple-DES 14
TTM 44, 50, 54, 57, 59, 60
TwoFish 86, 98

– **U** –
UART 30
Unbalanced Oil and Vinegar 62
Unique Masking Method 107
UOV 50, 61, 62

– **W** –
WAP 91
WTLS 91

– **X** –
XL 61, 72
XOR 102

– **Z** –
zero-knowledge 16, 44, 64

Liste des noms propres

– **A** –

Adleman, Leonard M. 16, 17
Agnew, Gordon B. 75
Akishita, Toru 92
Akkar, Mehdi-Laurent ..71, 74, 82, 85, 94, 98, 101, 105–107
Anderson, Ross J. 79, 81, 83
Appel, Andrew W. 82
Arimura, Kunitaka 28
Avanzi, Roberto M. 92

– **B** –

Babai, László 43
Barjavel, René 27
Barrett, Paul 31
Bellare, Mihir 21, 98
Berlekamp, Elwyn 70
Beth, Thomas 71, 99
Bevan, Régis 85
Biehl, Ingrid 83
Biham, Eli 49, 83–85
Bleichenbacher, Daniel 21, 75
Blömer, Johannes 81, 83
Bocharova, Irina E. 75
Boneh, Dan 19, 20, 22, 66, 82, 83
Boppana, Ravi B. 44
Brauer, Alfred Theodore 75
Briceno, Marc 84
Brickell, Ernest F. 70, 75
Brier, Éric 22, 88
Brumley, David 82
Buchberger, Bruno 72
Buss, Jonathan F. 45

– **C** –

Canetti, Ran 98
Chari, Suresh 85, 94
Chaum, David 22
Ciet, Mathieu 83
Clavier, Christophe 22, 84
Cocks, Clifford 16
Cohen, Henri 89
Coppersmith, Don 20, 21, 23, 43, 45, 55, 58, 59
Coron, Jean-Sébastien ..21–23, 85, 87–89, 105
Courtois, Nicolas T. .14, 44, 53, 57–61, 64, 66, 71, 74

– **D** –

Dabbish, Ezzy A. 84, 85
Daemen, Joan 85
Dai, Zong-Duo 58
Daum, Magnus 71
De Jonge, Wiebren 22
De Jonquières, Ernest 57, 60
De Seta, Domenico 101
De Waleffe, Dominique 31
DeMillo, Richard A. 83
Desmedt, Yvo 23
Dethloff, Jürgen 28
Dhem, Jean-François 31
Dickson, Leonard Eugene 55
Dieudonné, Jean 33
Diffie, Whitfield 15, 65, 66
Ding, Jintai 60
Dischamp, Paul 85
Dobbertin, Hans 53
Durfee, Glen 19
Dusart, Pierre 83
Duteuil, Romain 71, 74

– **E** –

ElGamal, Tahar 24, 26, 83
Ellingboe, Jules K. 27
Ellis, James 16
Euclide 18, 21, 22

– **F** –

Fahn, Paul N. 94
Faugère, Jean-Charles 71, 72
Feistel, Horst 107

Felke, Patrick 71
Fiat, Amos 16, 83
Finiasz, Matthieu 66
Fischer, Michael J. 16
Foucault, Léon 82
Frandsen, Gudmund S. 45
Frankel, Yair 19
Franklin, Matt 20

– **G** –

Gabidulin, Ernst M. 45
Gauss, Carl Friedrich ... 52, 54, 55, 58, 63, 64
Geiselmann, Willi 71, 99
Germani Lucia 101
Gilbert, Henri 72
Giraud, Christophe 83, 98, 101, 106
Girault, Marc 16, 22
Goldberg, Ian 84
Goldreich, Oded 33, 44
Goldwasser, Shafi 16, 44, 93
Golić, Jovan D. 101, 102
Gordon, Daniel M. 75
Govindavajhala, Sudhakar 82
Granboulan, Louis 70
Gray, Frank 76, 77
Gröbner, Wolfgang 62, 72
Guillou, Louis 16, 28

– **H** –

Halevi, Shai 23
Halpern, John 27
Hamming, Richard Wesley 84
Handschuh, Helena 31
Hardy, Godfrey Harold 13
Håstad, Johan 20, 42, 44
Hellman, Martin E. 15, 66
Hesse, Ludwig Otto 88, 91
Hilbert, David 55
Hironaka, Heisuke 54
Hodges, Timothy 60

– **I** –

Imai, Hideki 38, 50, 53, 57, 60, 97
Izu, Tetsuya 88

– **J** –

Jacobi, Carl Gustav Jacob 88
Jaffe, Joshua 84, 105
Jouve, Pierre Jean 111
Joux, Antoine 66, 70, 71
Joye, Marc 21, 83, 84, 88, 89
Jun, Benjamin 84, 105
Jutla, Charanjit S. 23, 85, 94

– **K** –

Kaltofen, Erich 69
Kerckhoffs, Auguste 11
Kipnis, Aviad 45, 61, 62
Klíma, Vastimil 84
Klimov, Alexander 61
Knuth, Donald E. 75
Koblitz, Neal 23
Kocher, Paul C. 82, 84, 105
Koeune, François 81
Krawczyk, Hugo 98
Kudryashov, Boris D. 75
Kuhn, Markus G. 83

– **L** –

Lam, Kwok-Yan 58
Legendre, Adrien-Marie 91
Lempel, Abraham 75
Lenstra, Arjen K. 18, 19, 83
Lenstra, Hendrik W., Jr 19
Letourneux, Gilles 83
Lidl, Rudolf 54
Lipton, Richard J. 83
Lovász, László 19
Ly, Olivier 82
Lynn, Ben 66

– **M** –

Mangard, Stefan 84
Markov, Andrei Andreyevich 84
Martinet, Gwenaëlle 70
Matsumoto, Tsutomu ... 38, 50, 53, 57, 60, 97
McCurley, Kevin 75
McEliece, Robert J. 66
Meier, Willi 61
Merkle, Ralph C. 15
Messerges, Thomas S. 84–87, 105
Meyer, Bernd 83
Micali, Silvio 16, 44, 79, 93, 105
Miller, Victor 23
Minier, Marine 72
Mironov, Ilya 66
Misarsky, Jean-François 22
Miyaki, Atsuko 89

Moh, Tzuong-Tsieng 57, 60
Montgomery, Peter L. 31, 88, 89, 91
Moore, Gordon 18
Moore, Judy 22
Moran, Shlomo 43
Moreno, Roland 28
Moyart, Didier 85
M'Raïhi, David 31
Müller, Volker 83
Mullin, Ronald C. 75

– N –

Naccache, David 21–23, 31, 66
Nguyen, Kim 66
Nguyen, Phong 19, 20
Niederreiter, Harald 54
Nyberg, Kaisa 26

– O –

Odlyzko, Andrew 23
Okamoto, Tatsuaki 22
Okeya, Katsuyuki 85
Ono, Takatoshi 89
Örs, Sıddıka Berna 80
Oswald, Elisabeth 80, 81, 84

– P –

Paar, Christof 80, 84
Paillier, Pascal 21, 31, 83
Patarin, Jacques 20, 41, 43, 45, 53, 55, 57, 61, 66, 70, 71, 83, 84, 93, 94
Pearson, Peter K. 94
Petrank, Erev 44
Pieprzyk, Joseph 14
Piret, Gilles 83
Poe, Edgar Allan 49
Pointcheval, David 22, 24, 25, 70
Pollard, John 18
Pomeroy 27
Poupard, Guillaume 16
Preneel, Bart 80, 98

– Q –

Quisquater, Jean-Jacques 16, 31, 81–83

– R –

Rabin, Michael O. 70
Rackoff, Charles 16, 44
Rao, Josyula R. 84, 85, 94
Reiter, Michael 20
Reyzin, Leonid 79, 105
Rijmen, Vincent 85
Rivest, Ronald L. 16, 17, 65
Rogaway, Philip 21
Rohatgi, Pankaj 84, 85, 94
Rosa, Tomáš 84
Roth, Ron M. 44
Rueppel, Rainer A. 26

– S –

Sakurai, Kouichi 85
Samyde, David 82
Scherzer, Helmut 84
Schmidt, Dieter 60
Schnorr, Claus Peter 16, 17, 24, 58, 83
Schramm, Kai 84
Sedlak, Holger 31
Seifert, Jean-Pierre 81, 83
Sendrier, Nicolas 66
Shacham, Hovav 66
Shallit, Jeffrey O. 45
Shamir, Adi 16, 17, 44, 45, 49, 59, 61, 62, 83–85
Shannon, Claude E. 93
Shoup, Victor 21, 69
Simmons, Gustavus J. 27, 28
Sipser, Michael 44
Skorobogatov, Sergei 81
Sloan, Robert H. 84, 85
Smart, Nigel P. 92
Steinwandt, Rainer 71, 99
Stern, Jacques 14, 16, 45, 50, 59, 66, 70
Stern, Julien P. 23
Stinson, Douglas R. 75
Strassen, Volker 39

– T –

Tacier, Jean-Daniel 61
Takagi, Tsuyoshi 88, 92
Tchulkine, Alexei 105
Trichina, Elena 101
Tunstall, Brian Parker 75
Tymen, Christophe 89, 101, 102

– U –

Ugon, Michel 28

– **V** –
Van Oorschot, Paul C. 98
Vanstone, Scott A. 25, 75
Vaudenay, Serge 25, 45, 59, 70
Vivolo, Olivier 83
Von zur Gathen, Joachim 75

– **W** –
Wagner, David 84
Weierstraβ, Karl Theodor Wilhelm 89, 91
Wiener, Michael 19, 20
Wigderson, Avi 44
Williamson, Malcolm 16
Wilson, David B. 75
Winograd, Shmuel 43, 58
Wollinger, Thomas 80, 84

– **Y** –
Yacobi, Yacov 75
Yao, Andrew Chi-Chih 75
Ye, Ding-Feng 58
Yen, Sung-Ming 83
Yung, Moti 70, 83

– **Z** –
Zachos, Stathis 44
Ziv, Jacob 75

Références

Chapitre 1 : La cryptologie

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, *Camellia: A 128-bit block cipher suitable for multiple platforms – design and analysis*. In Proceedings of Selected Areas in Cryptography – SAC'00, LNCS 2012, pp. 39-56, Springer-Verlag, 2001.

[2] M. Bellare, P. Rogaway, *Optimal Asymmetric Encryption – How to Encrypt with RSA*. In Proceedings of EUROCRYPT'94, LNCS 950, pp. 92-111, Springer-Verlag, 1994.

[3] M. Bellare, P. Rogaway, *The Exact Security of RSA Signatures – How to Sign with RSA and Rabin*. In Proceedings of EUROCRYPT'96, LNCS 1070, pp. 399-416, Springer-Verlag, 1996.

[4] D.J. Bernstein, *Circuits for Integer Factorization: A Proposal*. Manuscrit, 2001. Disponible sur `http://cr.yp.to/papers/nfscircuits.pdf`

[5] D. Bleichenbacher, *Chosen ciphertexts attacks against protocols based on the RSA encryption standard PKCS#1*. In Proceedings of CRYPTO'98, LNCS 1462, pp. 1-12, Springer-Verlag, 1998.

[6] D. Boneh, *Twenty Years of Attacks on the RSA Cryptosystem*. Notices of the American Mathematical Society, vol. 46, n.2, pp. 203-213, 1999.

[7] D. Boneh, *Simplified OAEP for the RSA and Rabin Functions*. In CRYPTO'2001, LNCS 2139, pp. 275-291, Springer-Verlag, 2001.

[8] D. Boneh, G. Durfee, *Cryptanalysis of RSA with Private Key d Less than* $n^{0.292}$. IEEE Transactions on Information Theory, vol. 46, n°4, pp. 1339-1349, 2000.

[9] D. Boneh, G. Durfee, Y. Frankel, *An attack on RSA given a fraction of the private key bits*. In Proceedings of ASIACRYPT'98, LNCS 1514, pp. 25-34, Springer-Verlag, 1998.

[10] D. Boneh, R. Venkatesan, *Breaking RSA may not be equivalent to factoring*. In Proceedings of EUROCRYPT'98, LNCS 1403, pp. 59-71, Springer-Verlag, 1998.

[11] E. Brier, C. Clavier, J.-S. Coron, D. Naccache, *Cryptanalysis of RSA Signatures with Fixed-Pattern Padding*. In Proceedings of CRYPTO'2001, LNCS 2139, pp. 433-439, Springer-Verlag, 2001.

[12] S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffet, C. Putnam, C. Putnam, P. Zimmermann, *Factorization of a 512-bit RSA Modulus*. In Proceedings of EUROCRYPT'2000, LNCS 1807, pp. 1-18, Springer-Verlag, 2000.

[13] D. Coppersmith, *Small solutions to polynomial equations, and low exponent RSA vulnerabilities*. In Journal of Cryptology, vol. 10 (1997), pp. 233-260.

[14] D. Coppersmith, M. Franklin, J. Patarin, M. Reiter, *Low-exponent RSA with related messages*. In Proceedings of EUROCRYPT'96, LNCS 1070, pp. 1-9, Springer-Verlag, 1996.

[15] D. Coppersmith, S. Halevi, C.S. Jutla, *ISO 9796-1 and the new forgery attack*. Présenté à la Rump Session de CRYPTO'99, 1999.

[16] J.-S. Coron, M. Joye, D. Naccache, P. Paillier, *New Attacks on PKCS#1 v1.5 Encryption*. In Proceedings of EUROCRYPT'2000, LNCS 1807, pp. 369-379, Springer-Verlag, 2000.

[17] J.-S. Coron, D. Naccache, J.P. Stern, *On the Security of RSA Padding*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 1-18, Springer-Verlag, 1999.

[18] N.T. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*. In Proceedings of ASIACRYPT'2002, LNCS 2501, pp. 267-287, Springer-Verlag, 2002.

[19] J. Daemen, V. Rijmen, *AES proposal: Rijndael*. La version la plus récente est disponible sur `http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf`

[20] W. De Jonge, D. Chaum, *Attacks on Some RSA Signatures*. In Proceedings of Crypto'85, LNCS 218, pp. 18-27, Springer-Verlag, 1986.

[21] D.E. Denning, *Digital Signatures with RSA and other Public-key cryptosystems*. Communications of the ACM, 27(4), pp. 388-392, 1984.

[22] Y. Desmedt, A. Odlyzko, *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes*. In Proceedings of CRYPTO'85, LNCS 218, pp. 516-522, Springer-Verlag, 1986.

[23] W. Diffie, M.E. Hellman, *New Directions in Cryptography*. IEEE Transactions on Information Theory, vol. IT-22, pp. 644-654, 1976.

[24] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. IEEE Transactions on Information Theory, Vol. 31, pp. 469-472, 1985.

[25] J. Ellis, *The story of non-secret encryption*. Article écrit en 1977 et publié après la mort de l'auteur en 1997. Disponible sur `http://www.cesg.gov.uk/about/nsecret`

[26] A. Fiat, A. Shamir, *How to Prove Yourself: Practical Solutions of Identification and Signature Problems*. In Proceedings of CRYPTO'86, LNCS 263, pp. 186-194, Springer-Verlag, 1987.

[27] M.J. Fischer, S. Micali, C. Rackoff, *A secure protocol for the oblivious transfer*. Journal of Cryptology, Vol. 9, No 3, pp. 191-195, 1996.

[28] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, *RSA-OAEP is Secure under the RSA Assumption*. In Proceedings of CRYPTO'2001, LNCS 2139, pp. 260-274, Springer-Verlag, 2001.

[29] M. Gardner, *A new kind of cipher that would take millions of years to break*. Scientific American, rubrique «Mathematical Games», pp. 120-124, août 1977.

[30] W. Geiselmann, R. Steinwandt, *A dedicated sieving hardware*. In Proceedings of PKC'2003, LNCS 2567, pp. 254-266, Springer-Verlag, 2003.

[31] M. Girault, *Self-Certified Public Keys*. In Proceedings of Eurocrypt'91, LNCS 547, pp. 490-497, Springer-Verlag, 1992.

[32] M. Girault, J.-F. Misarsky, *Selective Forgery of RSA Signatures Using Redundancy*. In Proceedings of EUROCRYPT'97, LNCS 1233, pp. 495-507, Springer-Verlag, 1997.

[33] S. Goldwasser, S. Micali, C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*. In Proceedings of the 17th ACM Symposium on the Theory of Computing (STOC'85), pp. 291-304, ACM Press, 1985.

[34] L.C. Guillou, J.-J. Quisquater, *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory*. In Proceedings of EUROCRYPT'88, LNCS 330, pp. 123-128, Springer-Verlag, 1988.

[35] J. Håstad, *Solving Simultaneous modular equations of low degree*. SIAM Journal of Computing, vol. 17, pp. 336-341, 1988.

[36] IEEE P1363-2000, *Standard Specifications for Public-Key Cryptography*, août 2000. Disponible sur `http://standards.ieee.org/catalog/olis/busarch.html`

[37] ISO/IEC 9796-1, *Information Technology – Security Techniques – Digital signature schemes giving message recovery – Part 1: Mechanisms using redundancy*, 1999.

[38] ISO/IEC 9796-2, *Information Technology – Security Techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*, 2002.

[39] N. Koblitz, *Elliptic curve cryptosystems*. Mathematics of Computation, Vol. 48, pp. 203-209, 1987.

[40] A.K. Lenstra, *Integer Factoring*. Designs, Codes and Cryptography, vol. 19, pp. 101-128, Kluwer Academic Publishers, Boston, 2000.

[41] A.K. Lenstra, H.W. Lenstra, *The development of the number field sieve*. Lecture Notes in Math., Vol. 1554, Springer-Verlag, Berlin, 1993.

[42] A.K. Lenstra, H.W. Lenstra, L. Lovász., *Factoring polynomials with rational coefficients*. Math. Ann., 261, pp.:513-534, 1982.

[43] A.K. Lenstra, H.W. Lenstra, M.S. Manasse, J.M. Pollard, *The number field sieve*. Proc. 22nd ACM Symposium on Theory of Computing, pp. 564-572, 1990.

[44] A.K. Lenstra, A. Shamir, *Analysis and Optimization of the TWINKLE Factoring Device*. In Proceedings of EUROCRYPT'2000, LNCS 1807, pp. 35-52, Springer-Verlag, 2000.

[45] A.K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer, *Analysis of Bernstein's Factorization Circuit*. In Proceedings of ASIACRYPT'2002, LNCS 2501, pp. 1-26, Springer-Verlag, 2002.

[46] A.K. Lenstra, E. Verheul, *Selecting cryptographic key sizes*. In Proceedings of PKC'2000, LNCS 1751, pp. 446-465, Springer-Verlag, 2000.

[47] A. Menezes, P. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996. Disponible en version électronique sur `http://www.cacr.math.uwaterloo.ca/hac`

[48] R.C. Merkle, *Secure Communication over Insecure Channels*. CACM, Vol. 21, No. 4, pp. 294-299, 1978.

[49] V. Miller, *Uses of elliptic curves in cryptography*. In Proceedings of CRYPTO'85, LNCS 218, pp. 417-426, Springer-Verlag, 1986.

[50] J.-F. Misarsky, *A Multiplicative Attack Using LLL Algorithm on RSA Signatures with Redundancy*. In Proceedings of CRYPTO'97, LNCS 1294, pp. 221-234, Springer-Verlag, 1997.

[51] J.-F. Misarsky, *How (not) to design RSA signature schemes*. In Proceedings of PKC'98, LNCS 1431, pp. 14-28, Springer-Verlag, 1998.

[52] NBS, *Data Encryption Standard*. Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC, 1977.

[53] NBS, *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication (FIPS PUB) 46-3, National Bureau of Standards, Gaithersburg, MD, 1999. Disponible sur `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

[54] NESSIE, *NESSIE Security Report*. Rapport NES/DOC/ENS/WP5/D20, version 2.0, février 2003. Disponible sur `http://www.cryptonessie.org`

[55] NESSIE, *Performance of Optimized Implementations of the NESSIE Primitives*. Rapport NES/TOC/TEC/WP6/D21, version 2.0, février 2003. Disponible sur `http://www.cryptonessie.org`

[56] NIST, *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication (FIPS PUB) 186, National Institute of Standards and Technology, novembre 1994.

[57] NIST, *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication (FIPS PUB) 186-2, National Institute of Standards and Technology, janvier 2000. Disponible sur `http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf`

[58] NIST, *Advanced Encryption Standard*. Federal Information Processing Standards Publication (FIPS PUB) 197, National Institute of Standards and Technology, décembre 2001. Disponible sur `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

[59] K. Nyberg, R.A. Rueppel, *A new signature scheme based on the DSA giving message recovery*. First ACCM Conference and Communications Security, ACM Press, pp. 58-61, 1993.

[60] K. Nyberg, R.A. Rueppel, *Message recovery for signature schemes based on the discrete logarithm problem*. Designs Codes and Cryptography, vol. 7, pp. 61-81, 1996.

[61] T. Okamoto, D. Pointcheval, *REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform*. In Proceedings of CT-RSA'2001, LNCS 2020, pp. 159-175, Springer-Verlag, 2001.

[62] D. Pointcheval, *Les Preuves de Connaissance et leurs Preuves de Sécurité*. Thèse de Doctorat, Université de Caen, 1996.

[63] D. Pointcheval, *How to Encrypt Properly with RSA*. Cryptobytes, vol. 5, n°1, RSA Laboratories, 2002.

[64] D. Pointcheval, S. Vaudenay, *On Provable Security for Digital Signature Algorithms*. Rapport interne, Laboratoire d'Informatique de l'École normale supérieure (LIENS), 1996.

[65] G. Poupard, J. Stern, *Security Analysis of a Practical "on the fly" Authentication and Signature Generation*. In Proceedings of EUROCRYPT'98, LNCS 1403, pp. 422-436, Springer-Verlag, 1998.

[66] J.-J. Quisquater, L.C. Guillou, *The new Guillou-Quisquater scheme*. In Proceedings of RSA'2000.

[67] R.L. Rivest, A. Shamir, L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, vol. 21, n°2, 1978, pp. 120-126.

[68] RSA Laboratories, *Public Key Cryptography Standards, PKCS #1*. Version 2.0. Disponible à l'adresse `http://www.rsalabs.com/pkcs/pkcs-1/index.html`

[69] RSA Laboratories, *Public Key Cryptography Standards, PKCS #1*. Version 2.1. À paraître sur `http://www.rsalabs.com/pkcs/pkcs-1/index.html`

[70] C.P. Schnorr, *Efficient Identification and Signatures for Smart Cards*. In Proceedings of CRYPTO'89, LNCS 435, pp. 235-251, Springer-Verlag, 1990.

[71] C.P. Schnorr, *Efficient Signature Generation by Smart Cards*. In Journal of Cryptology, Vol. 4, pp. 161-174, 1991.

[72] A. Shamir, *Factoring large numbers with the TWINKLE device*. In Proceedings of CHES'99, LNCS 1717, pp. 2-12, Springer-Verlag, 1999.

[73] A. Shamir, E. Tromer, *Factoring Large Numbers with the TWIRL Device*. In Proceedings of CRYPTO'2003, LNCS 2729, pp. 1-26, Springer-Verlag, 2003.

[74] V. Shoup, *OAEP Reconsidered*. In Proceedings of CRYPTO'2001, LNCS 2139, pp. 239-259, Springer-Verlag, 2001.

[75] J. Stern, *La Science du Secret*. Éditions Odile Jacob, Paris, 1997.

[76] STORK, *New Trends in Cryptology*. Rapport ENS-D4, version 1.1, édité par Phong Nguyen, février 2003. Disponible sur `http://www.stork.eu.org`

[77] STORK, *Open Problems in Cryptology*. Rapport RUB-D6, version 1.2, édité par Tanja Lange, février 2003. Disponible sur `http://www.stork.eu.org`

[78] STORK, *Research Agenda for the Future of Cryptology*. Rapport RUB-D5, version 1.2, édité par Tanja Lange, février 2003. Disponible sur `http://www.stork.eu.org`

[79] M.J. Wiener, *Cryptanalysis of Short RSA Secret Exponents*. IEEE Transactions on Information Theory, vol. 36, n°3, pp. 553-558, 1990.

Chapitre 2 : La carte à microprocesseur

[80] P. Barrett, *Implementing the Rivest, Shamir and Adleman Public-Key Encryption Algorithm on a Standard Digital Signal Processor*. In Proceedings of CRYPTO'86, LNCS 263, pp. 311-323, Springer-Verlag, 1987.

[81] N.T. Courtois, L. Goubin, J. Patarin, *FLASH, a fast multivariate signature algorithm*. In Proceedings of CT-RSA'2001, LNCS 2020, pp. 298-307, Springer-Verlag, 2001.
Note: Le schéma de signature SFLASH a été révisé depuis, voir [82].

[82] N.T. Courtois, L. Goubin, J. Patarin, *SFLASH, a fast asymmetric signature scheme for low-cost smartcards*. Version révisée, octobre 2001. Disponible sur `http://www.cryptonessie.org`

[83] N.T. Courtois, L. Goubin, J. Patarin, *QUARTZ, 128-bit long digital signatures*. In Proceedings of CT-RSA'2001, LNCS 2020, pp. 282-297, Springer-Verlag, 2001.
Note: Le schéma de signature QUARTZ a été révisé depuis, voir [84].

[84] N.T. Courtois, L. Goubin, J. Patarin, *QUARTZ, an asymmetric signature scheme for shot signatures on PC*. Version révisée, octobre 2001. Disponible sur `http://www.cryptonessie.org`

[85] D. de Waleffe, J.-J. Quisquater, *CORSAIR, A Smart Card for Public-Key Cryptosystems*. In Proceedings of CRYPTO'90, LNCS 537, pp. 503-513, Springer-Verlag, 1990.

[86] J.-F. Dhem, *Design of an Efficient Public-Key Cryptographic Library for RISC-based Smart Cards*. Ph.D Thesis, UCL, 1998.

[87] J.-F. Dhem, J.-J. Quisquater, *Recent results on modular multiplications for smart cards*. In Proceedings of CARDIS'98, LNCS 1820, pp. 336-352, Springer-Verlag, 1998. Disponible sur `http://users.belgacom.net/dhem/papers/mulmod_cardis98.pdf`

[88] E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, *ESIGN: Efficient digital signature scheme (submission to NESSIE)*. Primitive submitted to NESSIE by NTT Corp., septembre 2000. Disponible sur `http://www.cryptonessie.org`

[89] L.C. Guillou, M. Ugon, *Smart Card, a Highly Reliable and Portable Security Device*. In Proceedings of CRYPTO'86, LNCS 263, pp. 464-479, Springer-Verlag, 1987.

[90] L.C. Guillou, M. Ugon, J.-J. Quisquater, *The smart card, a standardized security device dedicated to public cryptography*. In [112], chapitre 13, pp. 561-613, IEEE Press, Piscataway, 1992.

[91] L.C. Guillou, M. Ugon, J.-J. Quisquater, *Cryptographic authentication protocols for smart cards*. Computer Networks 36(4), pp. 437-451, 2001.

[92] H. Handschuh, P. Paillier, *Smart Card Crypto-Coprocessors for Public-Key Cryptography*. In Cryptobytes, vol. 4, n°1, RSA Laboratories, 1998. Disponible sur `ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto4n1.pdf`

[93] H. Handschuh, P. Paillier, *Smart Card Crypto-Coprocessors for Public-Key Cryptography*. In Proceedings of CARDIS'98, LNCS 1820, pp. 386-394, Springer-Verlag, 2000.

[94] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte, *NTRUSign: Digital Signatures Using the NTRU Latice*. In Proceedings of CT-RSA'2003, LNCS 2612, Springer-Verlag, 2003.

[95] ISO/IEC 7816-1, *Information Technology – Security Techniques – Integrated circuit(s) cards with contacts – Part 1: Physical characteristics*, 1998.

[96] ISO/IEC 7816-2, *Information Technology – Security Techniques – Integrated circuit(s) cards with contacts – Part 2: Contact locations and minimum size*, 1998.

[97] ISO/IEC 7816-3, *Information Technology – Security Techniques – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols*, 1989.

[98] ISO/IEC 7816-4, *Information Technology – Security Techniques – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange*, 1995.

[99] D. Knuth, *The Art of Computer Programming*, vol. 2, Seminumerical Algorithms, Addison-Wesley, Reading MA, 1969.

[100] P.L. Montgomery, *Modular Multiplication without Trial Division*. Mathematics of Computations, vol. 44 (170), pp. 519-521, 1985.

[101] D. Naccache, D. M'Raïhi, *Cryptographic Smart Cards*. IEEE Micro, pp. 14-24, juin 1996.

[102] D. Naccache, D. M'Raïhi, W. Wolfowicz, A. di Porto, *Are Crypto-Accelerators Really Inevitable?* In Proceedings of EUROCRYPT'95, LNCS 1440, pp.404-409, Springer-Verlag, 1995.

[103] J. Omura, *A Public Key Cell Design for Smart Card Chips*. IT Workshop, Hawaii, USA, pp. 983-985, novembre 1990.

[104] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*. In Proceedings of EUROCRYPT'96, LNCS 1070, pp. 33-48, Springer-Verlag, 1996.

[105] M.O. Rabin, *Digitized Signatures and Public-Key Functions as Intractable as Factorization*. Technical Report LCS/TR-212, M.I.T. Laboratory for Computer Science, 1979.

[106] W. Rankl, W. Effing, *Smart card handbook*. John Wiley & Sons, 1997.

[107] D. Sauveron, *La Technology Java Card^{TM} – Présentation de la carte à puce – La Java Card*. Rapport Interne RR-1259-01, LaBRI, Université de Bordeaux I, 2001.

[108] B. Schneier, A. Shostack, *Breaking Up is Hard to Do: Modeling Security Threats for Smart Cards*. USENIX Workshop on Smart Card Technology, pp. 175-185, USENIX Press, octobre 1999. Disponible sur `http://www.counterpane.com/smartcard-threats.pdf`

[109] T. Schweinberger, V. Shoup, *ACE: The advanced cryptographic engine*. Primitive submitted to NESSIE, septembre 2000. Disponible sur `http://www.cryptonessie.org`

[110] H. Sedlak, *The RSA Cryptographic Processor: The First High Speed One-Chip Solution*. In Proceedings of EUROCRYPT'87, LNCS 293, pp. 95-105, Springer-Verlag, 1988.

[111] A. Shamir, *An Efficient Identification Scheme Based on Permuted Kernels*. In Proceedings of CRYPTO'89, LNCS 435, pp. 606-609, Springer-Verlag Verlag, 1990.

[112] G.J. Simmons, *Contemporary Cryptology, The Science of Information Integrity*. IEEE Press, Piscataway, 1992.

[113] M. Ugon, *L'Odyssée de la carte à puce*. Article paru dans *Le Monde*. Disponible sur `http://perso.wanadoo.fr/f1my/axtcp/cartapuc.htm`

[114] S.A. Vanstone, *Responses to NIST's proposal*. Communications of the ACM, vol. 35, pp. 50-52, juillet 1992.

[115] H.C. Williams, *A Modification of the RSA Public-Key Encryption Procedure*. IEEE Transactions on Information Theory, v.IT-26, n.6, 1980, pp. 726-729.

Chapitre 3 : Hypothèses calculatoires

[116] L. Babai, S. Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes*. JCSS, vol. 36, 1988, pp. 254-276.

[117] E.R. Berlekamp, R.J. McEliece, H.C.A. Van Tilborg, *On the inherent intractability of certain coding problems*. IEEE Transactions on Information Theory, IT-24(3), pp. 384-386, May 1978.

[118] M. Blum, *How to prove a theorem so no one else can claim it*. In Proceeedings of the International Congress of Mathematics, Berkeley CA, pp. 1444-1451, 1986.

[119] R.B. Boppana, J. Håstad, S. Zachos, *Does co-NP have short interactive proofs*. Information Proc. Letters, vol. 25, pp. 127-132, 1987.

[120] J.F. Buss, G.S. Frandsen, J.O. Shallit, *The computational complexity of some problems of linear algebra*. BRICS series report, Aarhus, Denmark, RS-96-33. Disponible sur `http://www.brics.dk/RS/96/33`

[121] F. Chabaud, *Asymptotic analysis of probabilistic algorithms for finding short codewords*. In Proceedings of EUROCODE'92, Udine, Italy, CISM Courses and lectures n° 339, pp. 217-228, Springer-Verlag, 1993.

[122] K. Chen, *A new identification algorithm*. Cryptography Policy and Algorithms Conference, LNCS 1029, Springer-Verlag, 1996.

[123] D. Coppersmith, J. Stern, S. Vaudenay, *Attacks on the Birational Permutation Signature Schemes*. In Proceedings of CRYPTO'93, LNCS 773, pp. 435-443, Springer-Verlag, 1993.

[124] D. Coppersmith, J. Stern, S. Vaudenay, *The Security of the Birational Permutation Signature Schemes*. In Journal of Cryptology, 10(3), pp. 207-221, 1997.

[125] D. Coppersmith, S. Winograd, *Matrix multiplication via arithmetic progressions*. J. Symbolic Computation, vol. 9 (1990), pp. 251-280.

[126] N.T. Courtois, *Efficient Zero-knowledge authentication based on a linear algebra problem MinRank*. In Proceedings of ASIACRYPT'2001, LNCS 2248, pp. 402-421, Springer-Verlag, 2001.

[127] M. Dickerson, *The functional Decomposition of Polynomials*. Ph.D Thesis, TR 89-1023, Department of Computer Science, Cornell University, Ithaca, NY, juillet 1989.

[128] M. Dickerson, *The Inverse of an Automorphism in polynomial Time*. IEEE 30^{th} annual symposium on Foundations of Computer Science (FOCS), 1989, pp. 82-87.

[129] S. Fortin, *The Graph Isomorphism Problem*. Technical Report 93-20, University of Alberta, Edmonton, Alberta, Canada, juillet 1996. Disponible sur `ftp://ftp.cs.ualberta.ca/pub/TechReports/1996/TR96-20/TR96-20.ps.gz`

[130] E.M. Gabidulin, *Theory of codes with maximum rank distance*. Problems of Information Transmission, 21:1-12, 1985.

[131] M.R. Garey, D.S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[132] O. Goldreich, S. Micali, A. Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*. Journal of the ACM, v. 38, n. 1, pp. 691-729, juillet 1991.

[133] S. Goldwasser, M. Sipser, *Private coins vs. public coins in interactive proof systems*. In Advances in Computing Research, S. Micali (Ed.), vol. 5, pp. 73-90, 1989.

[134] J. Gustafson, S. Aluru, *Massively Parallel Searching for Better Algorithms or, How to Do a Cross Product with Five Multiplications*. Ames Laboratory, Department of Energy, ISU, Ames, Iowa. Disponible sur `http://www.scl.ameslab.gov/Publications/FiveMultiplications/Five.html`

[135] J. Håstad, *Tensor Rank is NP-Complete*. Journal of Algorithms, vol. 11, pp. 644-654, 1990.

[136] S. Harari, *A new authentication algorithm*. In Coding Theory and Applications, LNCS 388, pp. 204-211, Springer-Verlag, 1989.

[137] A. Kipnis, J. Patarin, L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*. In Proceedings of EUROCRYPT'99, LNCS 1592, pp. 206-222, Springer-Verlag, 1999.

[138] A. Kipnis, A. Shamir, *Cryptanalysis of the HFE public key cryptosystem*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 19-30, Springer-Verlag, 1999.

[139] T. Matsumoto, H. Imai, *Public quadratic polynomial-Tuples for efficient Signature-Verification and Message-Encryption*. In Proceedings of EUROCRYPT'88, LNCS 330, pp. 419-453, Springer-Verlag, 1988.

[140] J. Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*. Proceedings of CRYPTO'95, LNCS 963, pp. 248-261, Springer-Verlag, 1995.

[141] J. Patarin, L. Goubin, *Asymmetric Cryptography with S-Boxes*. In Proceedings of ICICS'97, LNCS 1334, pp. 369-380, Springer-Verlag, 1997.

[142] J. Patarin, L. Goubin, N.T. Courtois, *Improved Algorithms for Isomorphisms of Polynomials*. In Proceedings of EUROCRYPT'98, LNCS 1403, pp. 184-200, Springer-Verlag, 1998.

[143] E. Petrank, R.M. Roth, *Is Code Equivalence Easy to Decide?* IEEE Transactions on Information Theory, 1997.

[144] A. Shamir, *Efficient Signature Schemes based on Birational Permutations*. In Proceedings of CRYPTO'93, LNCS 773, pp. 1-12, Springer-Verlag, 1993.

[145] J. Stern, *A new identification scheme based on syndrome decoding*. In Proceedings of CRYPTO'93, LNCS 773, pp. 13-21, Springer-Verlag, 1993.

[146] J. Stern, F. Chabaud, *The cryptographic security of the Syndrome Decoding problem for rank distance codes*. In Proceedings of ASIACRYPT'96, LNCS 1163, pp. 368-381, Springer-Verlag, 1985.

[147] V. Strassen, *Gaussian elimination is not optimal*. Numerische Mathematik, vol. 13, pp. 354-356, 1969.

[148] V. Strassen, *The asymptotic spectrum of tensors*. J. Reine Angew. Math., vol. 384, pp. 102-152, 1988.

[149] J. von zur Gathen, *Functional Decomposition of Polynomials: the tame Case*. Journal of Symbolic Computation, vol. 9, pp. 281-299, 1990.

[150] J. von zur Gathen, *Functional Decomposition of Polynomials: the wild Case*. Journal of Symbolic Computation, vol. 10, pp. 437-452, 1990.

Chapitre 4 : Cryptanalyse

[151] J.M. Chen, B.Y. Yang, B.Y. Peng, *Tame Transformation Signatures with Topsy-Turvy Hashes*. International Workshop for Asia Public Key Infrastructure, IWAP'2002. Disponible sur `http://www.usdsi.com/TTS.pdf`

[152] G. Chou, J. Guan, J.M. Chen, *A systematic Construction of a Q_{2^k}-model in TTM*. Comm. Algebra, 30(2), 551-562, 2002. Disponible sur `http://www.usdsi.com/chou.ps`

[153] D. Coppersmith, Communication personnelle à Jacques Patarin, 3 juin 1996.

[154] N.T. Courtois, *La sécurité des primitives cryptographiques basées sur les problèmes algébriques multivariables MQ, IP, MinRank, et HFE*. Thèse de Doctorat, Université de Paris VI, 2001. Disponible sur `http://www.minrank.org/phd.pdf`

[155] N.T. Courtois, L. Goubin, W. Meier, J.D. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*. In Proceedings of PKC'2002, LNCS 2274, pp. 211-227, Springer-Verlag, 2002.

[156] N.T. Courtois, A. Klimov, J. Patarin, A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*. In Proceedings of EUROCRYPT'2000, LNCS 1807, pp. 392-407, Springer-Verlag 2000.

[157] N.T. Courtois, J. Patarin, *About the XL Algorithm over $GF(2)$*. In Proceedings of CT-RSA'2003, LNCS 2612, pp. 141-157, Springer-Verlag, 2003.

[158] J. Ding, T. Hodges, *Cryptanalysis of an Implementation Scheme of the Tamed Transformation Method Cryptosystem*. IACR ePrint Archive, 2003/084. Disponible sur `http://eprint.iacr.org/2003/084/`

[159] J. Ding, D. Schmidt, *A Defect of the Implementation Schemes of the TTM Cryptosystem*. IACR ePrint Archive, 2003/085. Disponible sur `http://eprint.iacr.org/2003/085/`

[160] L. Goubin, N.T. Courtois, *Cryptanalysis of TTM*. In Proceedings of ASIACRYPT'2000, LNCS 1976, pp. 44-57, Springer-Verlag, 2000.

[161] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero*. Ann. Math, vol. 79, 1964.

[162] H. Imai, T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*. Algebraic Algorithms and Error Correcting Codes (AAECC-3), LNCS 229, pp. 108-119, Springer-Verlag, 1985.

[163] A. Kipnis, A. Shamir, *Cryptanalysis of the Oil and Vinegar Signature Scheme*. In Proceedings of CRYPTO'98, LNCS 1462, pp. 257-266, Springer-Verlag, 1998.

[164] R. Lidl, H. Niederreiter, *Finite Fields*. Encyclopedia of Mathematics and its applications, volume 20, Cambridge University Press, 1983.

[165] T.-T. Moh, *A public key system with signature and master key functions*. Communications in Algebra, 27(5), pp. 2207-2222, 1999. Disponible sur `http://www.usdsi.com/public.ps`

[166] T.-T. Moh, *A fast public key system with signature and master key functions*. In Proceedings of CrypTEC'99, International Workshop on Cryptographic Techniques and E-commerce, Hong-Kong City University Press, pp. 63-69, juillet 1999. Disponible sur `http://www.usdsi.com/cryptec.ps`

[167] T.-T. Moh, J.M. Chen, *On the Goubin-Courtois Attack on TTM*. IACR ePrint Archive, 2001/072. Disponible sur `http://eprint.iacr.org/2001/072/`

[168] J. Patarin, L. Goubin, *Trapdoor one-way permutations and multivariate polynomials*. In Proceedings of ICICS'97, LNCS 1334, pp. 356-368, Springer-Verlag, 1997.

[169] J. Patarin, L. Goubin, N.T. Courtois, *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*. In Proceedings of ASIACRYPT'98, LNCS 1514, pp. 35-49, Springer-Verlag, 1997.

[170] D.-F. Ye, K.-Y. Lam, Z.-D. Dai, *Cryptanalysis of the "2R" schemes*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 315-325, Springer-Verlag, 1999.

[171] A. Youssef, G. Gong, *Cryptanalysis of Imai and Matsumoto Scheme B Asymmetric Cryptosystem*. In Proceedings of INDOCRYPT'2001, LNCS 2247, pp. 215-222, Springer-Verlag 2001.

Chapitre 5 : Construction d'algorithmes dédiés

[172] G.B. Agnew, R.C. Mullin, S.A. Vanstone, *Fast Exponentiation in GF*(2^n). In Proceedings of EUROCRYPT'88, LNCS 330, pp. 251-255, Springer-Verlag, 1988.

[173] M.-L. Akkar, N.T. Courtois, R. Duteuil, L. Goubin, *A Fast and Secure Implementation of Sflash*. In Proceedings of PKC'2003, LNCS 2567, pp. 267-278, Springer-Verlag, 2003.

[174] D. Bleichenbacher, *Efficiency and Security of Cryptosystems based on Number Theory*. Ph.D dissertation, ETH, Zürich, 1996. Chapitre 4 : *Addition Chains*.

[175] I. Bocharova, B. Kudryashov, *Fast Exponentiation in Cryptography*. In Algrebraic Algorithms and Error-Correcting Codes: AAECC-11, LNCS 948, pp. 146-157, Springer-Verlag, 1995

[176] D. Boneh, B. Lynn, H. Shacham, *Short Signatures from the Weil Pairing*. In Proceedings of ASIACRYPT'2001, LNCS 2248, pp. 514-532, Springer-Verlag, 2001.

[177] A. Brauer, *On Addition Chains*. Bulletin of the American Mathematical Society, vol. 45, pp. 736-739, 1939.

[178] E.F. Brickell, D.M. Gordon, K. McCurley, D. Wilson, *Fast exponentiation with precomputation*. In Proceedings of EUROCRYPT'92, LNCS 658, pp. 200-207, Springer-Verlag, 1993.

[179] E.F. Brickell, D. Pointcheval, S. Vaudenay, M. Yung, *Design validations for discrete logarithm based signature schemes*. In Proceedings of PKC'2000, LNCS 1751, pp. 276-292, Springer-Verlag, 2000. Disponible sur `http://www.di.ens.fr/~pointche/pub.php?reference=BrPoVaYu00`

[180] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. Thèse de Doctorat, Université d'Innsbruck, Autriche, 1965.

[181] B. Buchberger, *Gröbner bases : an algorithmic method in polynomial ideal theory*. Multidimensional systems theory, No 16 in Mathematics and its Applications, ch. 6, pp. 184-232, D. Reidel Pub. Co., 1985.

[182] N.T. Courtois, *Generic Attacks and the Security of Quartz*. In Proceedings of PKC'2003, LNCS 2567, pp. 351-364, Springer-Verlag, 2003.

[183] N.T. Courtois, M. Daum, P. Felke, *On the Security of HFE, HFEv- and Quartz*. In Proceedings of PKC'2003, LNCS 2567, pp. 337-350, Springer-Verlag, 2003.

[184] N.T. Courtois, M. Finiasz, N. Sendrier, *How to Achieve a McEliece-Based Digital Signature Scheme*. In Proceedings of ASIACRYPT'2001, LNCS 2248, pp. 157-174, Springer-Verlag, 2001.

[185] J.-C. Faugère, *Report on a successful attack of HFE challenge 1 with Gröbner bases algorithm* $F_5/2$. Annonce sur `sci.crypt`, 19 avril 2002.

[186] J.-C. Faugère, *A new efficient algorithm for computing Gröbner bases without reduction to zero* (F_5). In Proceedings of International Symposium on Symbolic and Algebraic Computation – ISSAC'2002, pp. 75-83, ACM Press, 2002. Disponible sur `http://www-calfor.lip6.fr/~jcf/Papers/papers/f5/pdf`

[187] J.-C. Faugère, A. Joux, *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*. In Proceedings of CRYPTO'2003, LNCS 2729, pp. 44-60, Springer-Verlag, 2003.

[188] W. Geiselmann, R. Steinwandt, T. Beth, *Revealing 441 key bits of SFLASH^{v2}*. In Proceedings of the Third NESSIE Workshop, Munich, 6-7 novembre 2002.

[189] H. Gilbert, M. Minier, *Cryptanalysis of SFLASH*. In Proceedings of EUROCRYPT'2002, LNCS 2332, pp. 288-298, Springer-Verlag, 2002.

[190] L. Granboulan, *How to repair ESIGN*. In Proceedings of SCN'02, LNCS 2576, pp. 234-240, Springer-Verlag, 2002. Disponible sur `http://eprint.iacr.org/2002/0074/`.

[191] A. Joux, G. Martinet, *Weaknesses in Quartz signature scheme*. Rapport public, NESSIE, NES/DOC/ENS/WP5/026, 2002.

[192] A. Joux, K. Nguyen, *Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups*. IACR ePrint Archive, 2001/003. Disponible sur `http://eprint.iacr.org/2001/003/`.

[193] E. Kaltofen, V. Shoup, *Fast polynomial factorization over high algebraic extensions of finite fields*. In Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, 1997.

[194] I. Mironov, *A Short Signature as Secure as DSA*. Preprint, 2001.

[195] D. Naccache, J. Stern, *Signing on a Postcard*. In Proceedings of Financial Cryptography, FC'00, LNCS 1962, pp. 121-135, Springer-Verlag, 2000. Disponible sur `http://grouper.ieee.org/groups/1363/Research/contributions/Postcard.ps`

[196] NIST, *Dictionary of Algorithms, Data Structures, and Problems. Gray code – A definition and the recursive algorithm*. Disponible sur `http://www.nist.gov/dads/HTML/graycode.html`

[197] J. Patarin, *La Cryptographie Multivariable*. Diplôme d'habilitation à diriger des recherches, Université de Paris VII, 2000.

[198] L.A. Pintsov, S.A. Vanstone, *Postal Revenue Collection in the Digital Age*. In Proceedings of Financial Cryptography, FC'00, LNCS 1962, pp. 105-120, Springer-Verlag, 2000. Disponible sur `http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-43.ps`

[199] J. Stern, D. Pointcheval, J. Malone-Lee, N.P. Smart, *Flaws in applying proof methodologies to signature schemes*. In Proceedings of CRYPTO'2002, LNCS 2442, pp. 93-110, Springer-Verlag, 2002. Disponible sur `http://www.di.ens.fr/~pointche/pub.php?reference=MaPoSmSt02`

[200] D.R. Stinson, *Some Observations on Parallel Algorithms for Fast Exponentiation in GF*(2^n). SIAM Journal on Computing, 19(4), pp. 711-717, 1990.

[201] B.P. Tunstall, *Synthesis of noiseless compression codes*. Ph.D dissertation, Georgia Inst. Technol, 1968.

[202] J. von zur Gathen, *Efficient and Optimal Exponentiation in Finite Fields*. Computational Complexity, vol. 1, pp. 360-394, 1991.

[203] Y. Yacobi, *Exponentiating Faster with Addition Chains*. In Proceedings of EUROCRYPT'90, LNCS 473, pp. 222-229, Springer-Verlag, 1991.

[204] A.C. Yao, *On the Evaluation of Powers*. SIAM Journal on Computing, 5(1), pp. 100-103, 1976

[205] J. Ziv, A. Lempel, *Compression of Individual Sequences via Variable-Rate Coding*. IEEE Transactions on Information Theory, IT-24(5), 530-536, 1978.

[206] J. Ziv, A. Lempel, *A Universal Algorithm for Sequential Data Compression.* IEEE Transactions on Information Theory, IT-23(3), 337-343, 1977.

Chapitre 6 : Attaques physiques

[207] D.G. Abraham, G.M. Dolan, G.P. Double, J.V. Stevens, *Transaction Security System.* IBM Systems Journal, Vol. 30, No 2, pp 206-229, 1991.

[208] G.B. Agnew, R.C. Mullin, S.A. Vanstone, *An Implementation of Elliptic Curve Cryptosystems over* $\mathbb{F}_{2^{155}}$. IEEE Journal on Selected Areas in Communications, vol. 11, n. 5, pp 804-813, 1993.

[209] D. Agrawal, B. Archambeault, J.R. Rao, P. Rohatgi, *The EM Side-Channel(s).* In Proceedings of CHES'2002, LNCS 2523, pp. 29-45, Springer-Verlag, 2002.

[210] T. Akishita, T. Takagi, *Zero-Value Point Attacks on Elliptic Curve Cryptosystem.* À paraître in Procedings of the 6th Information Security Conference, ISC'2003, LNCS, 2003.

[211] M.-L. Akkar, R. Bevan, P. Dischamp, D. Moyart, *Power Analysis: What is now Possible.* In Proceedings of ASIACRYPT'2000, LNCS 1976, pp. 489-502, Springer-Verlag, 2000.

[212] M.-L. Akkar, L. Goubin, O. Ly, *About an Automatic Fault Injection Protection System.* In Proceedings of E-Smart'2003, Nice, 2003.

[213] R.J. Anderson, M.G. Kuhn, *Improved Differential Fault Analysis.* Manuscrit, 20 novembre 1996. Disponible sur `http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/dfa`

[214] R.J. Anderson, M.G. Kuhn, *Low Cost Attacks on Tamper Resistant Devices.* In Proceedings of Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, LNCS 1361, pp. 125-136, Springer-Verlag, 1997.

[215] ANSI X9.62, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).* 1999.

[216] R.M. Avanzi, *Countermeasures against Differential Power Analysis for Hyperelliptic Curve Cryptosystems.* In Proceedings of CHES'2003, LNCS 2779, pp. 366-381, Springer-Verlag, 2003.

[217] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, J.-P. Seifert, *Fault Attacks on RSA with CRT: concrete results and practical countermeasures.* In Proceedings of CHES'2002, LNCS 2523, pp. 260-275, Springer-Verlag, 2002.

[218] F. Bao, R.H. Deng, Y. Han, A. Jeng, A. Narasimhalu, T. Ngair, *Breaking Public-Key Cryptosystems on Tamper-Resistant Devices in the Presence of Transient Faults.* In Proceedings of Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, LNCS 1361, pp. 115-124, Springer-Verlag, 1997.

[219] A. Bellezza, *Countermeasures against Side-Channel Attacks for Elliptic Curve Cryptosystems.* IACR ePrint Archive, 2001/103. Disponible sur `http://eprint.iacr.org/2001/103/`

[220] I. Biehl, B. Meyer, V. Müller, *Differential Fault Attacks on Elliptic Curve Cryptosystems.* In Proceedings of CRYPTO'2000, LNCS 1880, pp. 131-146, Springer-Verlag, 2000.

[221] E. Biham, A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems.* In Proceedings of CRYPTO'97, LNCS 1294, pp. 513-528, Springer-Verlag, 1997.

[222] E. Biham, A. Shamir, *Power Analysis of the key scheduling of the AES candidates.* In Proceedings of the Second Advanced Encryption Standard Conference, NIST, 1999.

[223] J. Blömer, J.-P. Seifert, *Fault based cryptanalysis of the advanced en-*

cryption standard (AES). IACR ePrint Archive, 2002/075. Disponible sur `http://eprint.iacr.org/2002/075/`.

[224] D. Boneh, D. Brumley, *Remote timing attacks are practical*. À paraître in Proceedings of the 14th USENIX Security Symposium. Disponible sur `http://crypto.stanford.edu/~dabo/abstracts/ssl-timing.html`

[225] D. Boneh, R.A. DeMillo, R.J. Lipton, *New Threat Model Breaks Crypto Codes*. Bellcore Press Release, September 25th, 1996.

[226] D. Boneh, R.A. DeMillo, R.J. Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*. In Proceedings of EUROCRYPT'97, LNCS 1233, pp. 37-51, Springer-Verlag, 1997.

[227] D. Boneh, R.A. DeMillo, R.J. Lipton, *On the Importance of Eliminating Errors in Cryptographic Computations*. Journal of Cryptology, Vol. 14, n°2, pp. 101-119, 2001.

[228] M. Briceno, I. Goldberg, D. Wagner, *GSM Cloning*. Disponible sur `http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html`

[229] M. Briceno, I. Goldberg, D. Wagner, *An implementation of the GSM A3A8 algorithm (specifically COMP128)*. Disponible sur `http://www.mirrors.wiretapped.net/security/cryptography/algorithms/gsm/a3a8.txt`

[230] E. Brier, M. Joye, *Weierstraß Elliptic Curves and Side-Channel Attacks*. In Proceedings of PKC'2002, LNCS 2274, pp. 335-345, Springer-Verlag, 2002.

[231] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S.M. Matyas, L. O'Connor, M. Peyravian, D. Safford, N. Zunic, *MARS - A Candidate Cipher for AES*. Soumission AES. Disponible sur `http://www.research.ibm.com/security/mars.pdf`

[232] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, NIST, 1999.

[233] M. Ciet, M. Joye, *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults*. IACR ePrint Archive, 2003/028. Disponible sur `http://eprint.iacr.org/2003/028/`

[234] C. Clavier, M. Joye, *Universal Exponentiation Algorithm – A First Step towards Provable SPA-Resistance*. In Proceedings of CHES'2001, LNCS 2162, pp. 300-308, Springer-Verlag, 2001.

[235] H. Cohen, A. Miyaji, T. Ono, *Efficient Elliptic Curve Exponentiation Using Mixed Coordinates*. In Proceedings of ASIACRYPT'98, LNCS 1514, pp. 51-65, Springer-Verlag, 1998.

[236] J.-S. Coron, *Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems*. In Proceedings of CHES'99, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.

[237] J.-S. Coron, L. Goubin, *On Boolean and Arithmetic Masking against Differential Power Analysis*. In Proceedings of CHES'2000, LNCS 1965, pp. 231-237, Springer-Verlag, 2000.

[238] J. Daemen, V. Rijmen, *Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, NIST, 1999.

[239] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, J.-L. Willems, *A practical implementation of the timing attack*. In Proceedings of CARDIS'98, LNCS 1820, pp. 167-182, Springer-Verlag, 1998. Disponible sur `http://www.dice.ucl.ac.be/crypto/techreports.html`

[240] P. Dusart, G. Letourneux, O. Vivolo, *Differential Fault Analysis on A.E.S.*. IACR ePrint Archive, 2003/010. Disponible sur `http://eprint.iacr.org/2003/010/`

[241] W. Fischer, C. Giraud, E.W. Knudsen, J.-P. Seifert, *Parallel Scalar Multiplication on General Elliptic Curves over $\mathbb{F}_p$ hedged against Non-Differential Side-Channel Attacks*. IACR ePrint Archive, 2002/007. Disponible sur `http://eprint.iacr.org/2002/007/`

[242] K. Gandolfi, C. Mourtel, F. Olivier, *Electromagnetic Attacks: Concrete Results*. In Proceedings of CHES'2001, LNCS 2162, pp. 251-261, Springer-Verlag, 2001.

[243] C. Giraud, *DFA on AES*. IACR ePrint Archive, 2003/008. Disponible sur `http://eprint.iacr.org/2003/008/`

[244] L. Goubin, *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*. In Proceedings of PKC'2003, LNCS 2567, pp. 199-211, Springer-Verlag, 2003.

[245] S. Govindavajhala, A.W. Appel, *Using Memory Errors to Attack a Virtual Machine*. IEEE Symposium on Security and Privacy, Oakland, 2003. Disponible sur `http://www.cs.princeton.edu/~sudhakar/papers/memerr.pdf`

[246] GSM Association, *Functional Description of the One Way Function COMP128 for Subscriber Authentication and Key Generation in the Paneuropean Mobile Communication System*. Document classifié, mais dont le contenu a été divulgué dans [229].

[247] H. Handschuh, P. Paillier, *Reducing the collision probability of Alleged Comp128*. In proceedings of CARDIS'98, LNCS 1820, pp. 366-371, Springer-Verlag, 1998.

[248] M.A. Hasan, *Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems*. In Proceedings of CHES'2000, LNCS 1965, pp. 93-108, Springer-Verlag, 2000.

[249] A. Hevia, M.A. Kiwi, *Strength of two data encryption standard implementations under timing attacks*. ACM Transactions on Information and System Security (TISSEC), Vol. 2, pp. 416-437, 1999.

[250] ISO/IEC 15946-4, *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 4: Digital signatures giving message recovery*. Document de travail, JTC 1/SC 27, 28 décembre 2001.

[251] T. Izu, T. Takagi, *A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks*. In Proceedings of PKC'2002, LNCS 2274, pp. 280-296, Springer-Verlag, 2002.

[252] T. Izu, T. Takagi, *Exceptional procedure attack on elliptic curve cryptosystems*. In Proceedings of PKC'2003, LNCS 2567, pp. 224-239, Springer-Verlag, 2003.

[253] M. Joye, A.K. Lenstra, J.-J. Quisquater, *Chinese Remaindering Based Cryptosystems in the Presence of Faults*. Journal of Cryptology, Vol. 12, n. 4, pp. 241-245, 1999.

[254] M. Joye, J.-J. Quisquater, *Hessian Elliptic Curves and Side-Channel Attacks*. In Proceedings of CHES'2001, LNCS 2162, pp. 412-420, Springer-Verlag, 2001.

[255] M. Joye, J.-J. Quisquater, F. Bao, R.H. Deng, *RSA-type Signatures in the Presence of Transient Faults*. In Proceedings of the 6th IMA International Conference on Cryptography and Coding, Cirencester (U.K.), 17-19th December 1997, LNCS 1355, pp. 155-160, Springer-Verlag, 1997.

[256] M. Joye, J.-J. Quisquater, S.-M. Yen, M. Yung, *Observability analysis: Detecting when improved cryptosystems fail*. In Proceedings of CT-RSA'2002, LNCS 2271, pp. 17-29, Springer-Verlag, 2002.

[257] M. Joye, C. Tymen, *Protections against Differential Analysis for Elliptic Curve Cryptography – An Algebraic Approach*. In Proceedings of CHES'2001, LNCS 2162, pp. 377-390, Springer-Verlag, 2001.

[258] V. Klíma, T. Rosa, *Further Results and Considerations on Side Channel Attacks on RSA*. In Proceedings of CHES'2002, LNCS 2523, pp. 244-259, Springer-Verlag, 2002.

[259] P.C. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*. In Proceedings of CRYPTO'96, LNCS 1109, pp. 104-113, Springer-Verlag, 1996.

[260] P. Kocher, J. Jaffe, B. Jun, *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Disponible sur `http://www.cryptography.com/dpa/technical/index.html`

[261] P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.

[262] X. Lai, J. Massey, *A Proposal for a New Block Encryption Standard*. In Proceedings of EUROCRYPT'90, LNCS 473, pp. 389-404, Springer-Verlag, 1991.

[263] A.K. Lenstra, *Memo on RSA Signature Generation in the Presence of Faults*. Manuscrit, 28 septembre 1996. Disponible auprès de l'auteur.

[264] P.-Y. Liardet, N.P. Smart, *Preventing SPA/DPA in ECC system using the Jacobi Form*. In Proceedings of CHES'2001, LNCS 2162, pp. 401-411, Springer-Verlag, 2001.

[265] C.H. Lim, H.S. Hwang, *Fast Implementation of Elliptic Curve Arithmetic in GF(p^m)*. In Proceedings of PKC'2000, LNCS 1751, pp. 405-421, Springer-Verlag, 2000.

[266] J. López, R. Dahab, *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*. In Proceedings of CHES'99, LNCS 1717, pp. 316-327, Springer-Verlag, 1999.

[267] S. Mangard, *A simple power-analysis (SPA) attack on implementations of the AES key expansion*. In Proceedings of ICISC'2002, LNCS 2587, pp. 343-358, Springer-Verlag, 2002.

[268] T.S. Messerges, *Securing the AES Finalists Against Power Analysis Attacks*. In Proceedings of FSE'2000, LNCS 1978, pp. 150-164, Springer-Verlag, 2000.

[269] T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Investigations of Power Analysis Attacks on Smartcards*. In Proceedings of the USENIX Workshop on Smartcard Technology, pp. 151-161, mai 1999. Disponible sur `http://www.eecs.uic.edu/~tmesserg/papers.html`

[270] T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*. In Proceedings of CHES'99, LNCS 1717, pp. 144-157, Springer-Verlag, 1999.

[271] B. Möller, *Securing Elliptic Curve Point Multiplication against Side-Channel Attacks*. In Proceedings of ISC'2001, LNCS 2200, pp. 324-334, Springer-Verlag, 2001.

[272] P.L. Montgomery, *Speeding the Pollard and Elliptic Curve Methods for Factorizations*. Mathematics of Computation, vol. 48, pp. 243-264, 1987.

[273] NIST, *Recommended Elliptic Curves for Federal Government Use*. Appendice de FIPS 186-2 [57], National Institute of Standards and Technology, janvier 2000.

[274] K. Okeya, H. Kurumatani, K. Sakurai, *Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications*. In Proceedings of PKC'2000, LNCS 1751, pp. 238-257, Springer-Verlag, 2000.

[275] K. Okeya, K. Miyazaki, K. Sakurai, *A Fast Scalar Multiplication Method with Randomized Projective Coordinates on a Montgomery-form Elliptic Curve Secure against Side Channel Attacks*. In Pre-proceedings of ICICS'2001, pp. 475-486, 2001.

[276] K. Okeya, K. Sakurai, *Power Analysis Breaks Elliptic Curve Cryptosystem even Secure against the Timing Attack*. In Proceedings of INDOCRYPT'2000, LNCS 1977, pp. 178-190, Springer-Verlag, 2000.

[277] K. Okeya, K. Sakurai, *Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-coordinate on a Montgomery-form Elliptic Curve*. In Proceedings of CHES'2001, LNCS 2162, pp. 126-141, Springer-Verlag, 2001.

[278] S.B. Örs, E. Oswald, B. Preneel, *Power-Analysis Attacks on an FPGA – First Experimental Results*. In Proceedings of CHES'2003, LNCS 2779, pp. 35-50, Springer-Verlag, 2003.

[279] E. Oswald, *Enhancing simple power-analysis attacks on elliptic curve cryptosystems*. In Proceedings of CHES'2002, LNCS 2523, pp. 82-97, Springer-Verlag, 2002.

[280] P. Paillier, *Evaluating Differential Fault Analysis of Unknown Cryptosystems*. In Proceedings of PKC'99, LNCS 1560, pp. 235-244, Springer-Verlag, 1999.

[281] G. Piret, J.-J. Quisquater, *A Differential Fault Attack Technique Against SPN Structures, with Application to the AES and KHAZAD*. In Proceedings of CHES'2003, LNCS 2779, pp. 77-88, Springer-Verlag, 2003.

[282] J.-J. Quisquater, F. Koeune, *Survey of side channel attacks*. Rapport CRYPTREC numéro 1047, 2002. Disponible sur `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf`

[283] J.-J. Quisquater, D. Samyde, *ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards*. In Proceedings of E-Smart'2001, LNCS 2140, pp. 200-210, Springer-Verlag, 2001.

[284] J.-J. Quisquater, D. Samyde, *Eddy current for magnetic analysis with active sensor*. In Proceedings of E-Smart'2002, Nice, 2002.

[285] J.R. Rao, P. Rohatgi, H. Scherzer, *Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards*. IEEE Symposium on Security and Privacy, Oakland, 2002.

[286] R.L. Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin, *The RC6 Block Cipher*. Version 1.1, 20 août 1998. Disponible sur `ftp://ftp.rsasecurity.com/pub/rsalabs/aes/rc6v11.pdf`

[287] W. Schindler, *A combined timing and power attack*. In Proceedings of PKC'2002, LNCS 2274, pp. 263-279, Springer-Verlag, 2002.

[288] W. Schindler, J.-J. Quisquater, F. Koeune, *Improving divide and conquer attacks against cryptosystems by better error detection correction strategies*. In Proceedings of the 8th IMA International Conference on Cryptography and Coding, LNCS 2260, pp. 245-267, Springer-Verlag, 2001.

[289] B. Schneier, J. Kemsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, *Twofish: A 128-Bit Block Cipher*. Soumission AES. Disponible sur `http://www.counterpane.com/twofish.pdf`

[290] K. Schramm, T. Wollinger, C. Paar, *A new class of collision attacks and its application to DES*. In Proceedings of FSE'2003, LNCS 2887, Springer-Verlag, 2003.

[291] A. Shamir, *How to check modular exponentiation*. Présenté à la Rump Session d'EUROCRYPT'97, Konstanz, Allemagne. Cette méthode est brevetée [292].

[292] A. Shamir, *Method and apparatus for protecting public key schemes from timing and fault attacks*. United States Patent 5991415, 23 novembre 1999.

[293] S. Skorobogatov, R.J. Anderson, *Optical fault induction attacks*. In Proceedings of CHES'2002, LNCS 2523, pp. 2-12, Springer-Verlag, 2002.

[294] N.P. Smart, *The Hessian Form of an Elliptic Curve*. In Proceedings of CHES'2001, LNCS 2162, pp. 118-125, Springer-Verlag, 2001.

[295] N.P. Smart, *An Analysis of Goubin's Refined Power Analysis Attack*. In Proceedings of CHES'2003, LNCS 2779, pp. 281-290, Springer-Verlag, 2003.

[296] WAP, Wireless Application Protocol Forum, *Wireless Transport Layer Security (WTLS) Specification*. Disponible sur `http://www.wapforum.org`

[297] T. Wollinger, C. Paar, *How Secure are FPGAs in Cryptographic Applications ?* IACR ePrint Archive, 2003/119. Disponible sur `http://eprint.iacr.org/2003/119/`

[298] S.-M. Yen, M. Joye, *Checking before output may not be enough against fault-based cryptanalysis*. IEEE Transactions on Computers, Vol. 49, No. 9, pp. 967-970, septembre 2000.

[299] S.-M. Yen, S. Kim, S. Lim, S. Moon, *RSA speedup with residue number system immune against hardware fault cryptanalysis*. In Proceedings of ICISC'2001, LNCS 2288, pp. 397-413, Springer-Verlag, 2001.

Chapitre 7 : Contre-mesures

[300] ANSI X9.42, *Public Key Cryptography for The Financial Service Industry: Agreement of Symmetric Keys on Using Diffie-Hellman and MQV Algorithms*, 1998.

[301] M.-L. Akkar, C. Giraud, *An Implementation of DES and AES Secure against Some Attacks*. In Proceedings of CHES'2001, LNCS 2162, pp. 309-318, Springer-Verlag, 2001.

[302] M.-L. Akkar, L. Goubin, *A Generic Protection against High-Order Differential Power Analysis*. In Proceedings of FSE'2003, LNCS 2887, Springer-Verlag, 2003.

[303] M. Bellare, R. Canetti, H. Krawczyk, *Keying Hash Functions for Message Authentication*. In Proceedings of CRYPTO'96, LNCS 1109, pp. 1-15, Springer-Verlag, 1996. Disponible sur `http://www-cse.ucsd.edu/users/mihir/papers/hmac.html`

[304] J. Borst, *Block ciphers: Design, analysis and side-channel analysis*, Ph.D. thesis, K.U.Leuven, 2001.

[305] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power-Analysis Attacks*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 398-412, Springer-Verlag, 1999.

[306] J.-S. Coron, A. Tchulkine, *A new algorithm for switching from arithmetic to boolean masking*. In Proceedings of CHES'2003, LNCS 2779, pp. 89-97, Springer-Verlag, 2003.

[307] J. Daemen, M. Peters, G. Van Assche, *Bitslice Ciphers and Power Analysis Attacks*. In Proceedings of FSE'2000, LNCS 1978, pp. 134-149, Springer-Verlag, 2000.

[308] P.N. Fahn, P.K. Pearson, *IPA: A New Class of Power Attacks*. In Proceedings of CHES'99, LNCS 1717, pp. 173-186, Springer-Verlag, 1999.

[309] J.D. Golić, C. Tymen, *Multiplicative Masking and Power Analysis of AES*. In Proceedings of CHES'2002, LNCS 2523, pp. 198-212, Springer-Verlag, 2002.

[310] L. Goubin, *About DPA-resistant implementations of FLASH and SFLASH*. Présenté à la Rump Session du premier NESSIE Workshop, Leuven, Belgique, 13-14 novembre 2000.

[311] L. Goubin, *A Sound Method for Switching between Boolean and Arithmetic Masking*. In Proceedings of CHES'2001, LNCS 2162, pp. 3-15, Springer-Verlag, 2001.

[312] L. Goubin, J. Patarin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. Brevet européen, numéro de publication : 2789535, Bull CP8, 4 février 1999.

[313] L. Goubin, J. Patarin, *DES and Differential Power Analysis*. In Proceedings of CHES'99, LNCS 1717, pp. 158-172, Springer-Verlag, 1999.

[314] ISO/IEC 15408, *Information Technology – Security Techniques – Evaluation Criteria for IT Security*, 1999.

[315] T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant software*. In Proceedings of CHES'2000, LNCS 1965, pp. 238-251, Springer-Verlag, 2000.

[316] S. Micali, L. Reyzin, *Physically Observable Cryptography*. IACR ePrint Archive, 2003/120. Disponible sur `http://eprint.iacr.org/2003/120/`.

[317] NIST, *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication (FIPS PUB) 180-2, National Institute of Standards and Technology, août 2002. Disponible sur `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`

[318] B. Preneel, P.C. van Oorschot, *MDx-MAC and building fast MACs from hash functions*. In Proceedings of CRYPTO'95, LNCS 963, pp. 1-14, Springer-Verlag, 1995.

[319] R.L. Rivest, *The MD5 message-digest algorithm*. Request for Comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force, avril 1992.

[320] R. Steinwandt, W. Geiselmann, T. Beth, *A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates FLASH and SFLASH*. In Proceedings of the First NESSIE Workshop, Leuven, Belgique, 13-14 novembre 2000.

[321] E. Trichina, D. De Seta, L. Germani, *Simplified Adaptive Multiplicative Masking for AES*. In Proceedings of CHES'2002, LNCS 2523, pp. 187-197, Springer-Verlag, 2002.

Deuxième partie

Curriculum vitć et publications

Curriculum vitæ

Sommaire

1 État civil 141
2 Formation 141
3 Cursus professionnel 142
4 Valorisation de la recherche 143
5 Encadrement de la recherche 143
6 Enseignement 143

1 État civil

Louis Goubin, né le 2 avril 1970 à Valenciennes (Nord), célibataire.

2 Formation

- 1992-95 : **Université Paris XI, Orsay.**
  Thèse de Doctorat en mathématiques pures, soutenue le 10 janvier 1995 : «Sommes d'exponentielles à coefficients multiplicatifs» (Répartition des nombres premiers dans les progressions arithmétiques), dirigée par Hedi Daboussi.
- 1989-93 : **École normale supérieure, Paris.**
  - septembre 1992 : DEA «Modélisation et méthodes mathématiques appliquées à l'Économie» (Paris I Sorbonne – École Polytechnique), mention Très Bien.
  - octobre 1991 : DEA de mathématiques pures (Université Paris XI, Orsay), mention Bien.
  - juillet 1991 : Reçu 13^{e} à l'Agrégation de mathématiques.
  - juin 1990 : Licence et maîtrise de mathématiques.
- 1987-89 : **Lycée Louis Le Grand, Paris.**
  Admis à l'École Polytechnique et à l'École normale supérieure.
- 1987 : Troisième accessit au Concours Général de mathématiques. Mention régionale au Concours Général de physique. Baccalauréat série C, mention Très Bien.

3 Cursus professionnel

- Depuis septembre 2001 : Chef du service «Cryptographie et sécurité avancée» de Schlumberger Smart Cards.
  - Coordinateur du projet RNRT «X-Crypt» : outils cryptographiques adaptés aux réseaux de télécommunication à haut débit et aux réseaux sans fil émergents, combinant caractéristiques de sécurité avancées et faibles consommation de ressources.
  - Développement de méthodes pour protéger les algorithmes cryptographiques contre les attaques physiques (SPA, DPA, DFA, ...).
  - Implémentation optimisée et sécurisée de nouveaux algorithmes, en particulier AES et courbes elliptiques.
  - Preuves formelles de sécurité pour des protocoles cryptographiques.
  - Étude de la vérification de bytecode (pour la machine virtuelle Java sur les cartes à puce).
  - Nouveaux protocoles pour le «traitor tracing» (applications à la TV à péage et à la protection de logiciel).
  - Étude de nouveaux algorithmes de chiffrement rapide: NTRU, ECIES, ... (dans le cadre du projet européen Medea+ EsP@ss-IS project).
  - Recherche sur de nouveaux protocoles basés sur les courbes elliptiques (projet RNRT Turbosignatures).
- 1996-2001 : Expert cryptologue au département «Recherche et Développement» de Bull CP8.
  - Recherche : cryptographie asymétrique à base de polynômes multivariables sur des corps finis, preuves de sécurité pour des méthodes anti-DPA, générateurs pseudo-aléatoires.
  - Conception, analyse et implémentation de protocoles et d'algorithmes cryptographiques sur carte à puce et sur PC : téléphonie mobile (GSM, WAP, UMTS), cartes bancaires, porte-monnaie électronique, télévision à péage, transport.
  - Responsable Bull CP8 du projet RNRT «Turbo-Signatures» (France Telecom, Bull CP8 et École Normale Supérieure) : optimisation des ressources dans les protocoles de signature et d'authentification. Responsable scientifique des «Mécanismes à base de courbes elliptiques».
  - Spécification de trois nouveaux algorithmes de signature à clé publique pour le projet européen NESSIE : QUARTZ, FLASH et SFLASH.
  - Sécurisation des implémentations sur carte à puce contre les attaques physiques (Timing attacks, Differential Fault Analysis, SPA, DPA).
  - Optimisation (taille, mémoire et temps de calcul) de codes sur carte à puce : DES sur 6805, RSA sur ST22 : projet européen MASSC (Multi-Application Secure Smart Card), AES.
- 1995-96 : Service militaire : Professeur de mathématiques à l'EMSST (École Militaire, Paris).

4 Valorisation de la recherche

- 2003 : Membre du comité de programme de CHES'2003.
- 2003 : Conférencier invité aux journées «Sécurité et Multimedia» de l'université de Grenoble (février 2003).
- 2002 : Conférencier invité au workshop STORK à Bruges, Belgique (26-27 novembre 2002).
- 2001 : Conférencier invité à l'Institut de Mathématiques de Luminy (juin 2002).
- 2001 : Conférencier invité au «Workshop on algebraic methods in cryptography» à Bochum, Allemagne (8-11 novembre 2001).
- 2000 : Conférencier invité au séminaire «Systèmes Dynamiques Discrets» de l'Université de Marseille – Luminy (avril 2000).
- 1998 : Conférencier invité aux journées «Mathématiques et Industrie» (Luminy, 16-20 mars 1998).

5 Encadrement de la recherche

- 2003 : Stage de DESS – Guilhem Castagnos, *Autour de l'attaque de Davies et Murphy sur le DES.*
- 2003 : Stage de DESS – Yannick Leplard, *Protection semi-automatique de code contre les attaques par injection de fautes.*
- 2002 : Stage de DESS – Romain Duteuil, *SFLASH, la signature électronique la plus rapide au monde.*
- 2001 : Stage de DEA – Arnaud Leprince, *Schémas de chiffrement combinant boîtes S et transformations affines.*

6 Enseignement

- 2003-2004 : Cours de cryptographie à l'université de Versailles - Saint-Quentin-en-Yvelines
- 2003-2004 : Cours «Sécurité des cartes à microprocesseur» au DESS CCSI (Codes, Cryptographie et Securité Informatique) de l'université de Bordeaux I.
- 2002-2003 : Cours «Cryptographie et cartes à puce» au DESS CCSI (Codes, Cryptographie et Securité Informatique) de l'université de Bordeaux I.
- 2002-2003 : Cours «Théorie et pratique de la cryptologie» pour le département Recherche et Développement de Schlumberger.
- 2001-2002 : Formation «Cryptographic Algorithms: Secure Implementations and New Designs» pour la division Recherche de Schlumberger (Austin).
- 2000-2001 : Cours «Attaques DPA et contre-mesures» pour le département Recherche et Développement de Bull CP8.

- 1998-1999 : Cours «Schémas à base de courbes elliptiques» pour le département Recherche et Développement de Bull CP8.
- 1997-1999 : Professeur de mathématiques au CNED (Centre national d'enseignement à distance) : préparation à l'Agrégation de mathématiques.
- 1997-1998 : Formation «Cryptographie asymétrique et cartes à microcircuit» pour le GIE Cartes bancaires.
- 1996-1997 : Cours «Cryptographie symétrique» pour le département Recherche et Développement de Bull CP8.
- 1995-1996 : Service militaire : Professeur de mathématiques à l'EMSST (École Militaire, Paris).
- 1993-1995 : Enseignement de mathématiques en DEUG (Université Paris XI, Orsay).

Liste de publications

Sommaire

1	**Articles dans des journaux**	**145**
2	**Articles présentés à des congrès internationaux avec comité de lecture**	**145**
3	**Articles présentés à des colloques**	**146**
4	**Travaux de vulgarisation**	**147**
5	**Mémoires et rapports**	**147**
6	**Propositions à des organismes de normalisation**	**147**
7	**Brevets**	**148**

1 Articles dans des journaux

- L. Goubin, C. Mauduit et A. Sárközy, *Construction of large families of pseudo-random binary sequences*. À paraître in Journal of Number Theory, 2003.
- L. Goubin, *Sommes d'exponentielles et principe de l'hyperbole*. Acta Arithmetica, Vol. 73, n° 4, pp. 303-324, 1995.
- L. Goubin, *Sommes d'exponentielles à coefficients multiplicatifs et entiers sans grand facteur premier*. Acta Mathematica Hungarica, Vol. 67, n° 1-2, pp. 37-67, 1995.

2 Articles présentés à des congrès internationaux avec comité de lecture

- G. Castagnos, N.T. Courtois, L. Goubin, *What do DES S-boxes Say to Each Other?* À paraître.
- M.-L. Akkar, L. Goubin et O. Ly, *About an Automatic Fault Injection Protection System*. In Proceedings of E-Smart'2003, Nice, 2003.
- M.-L. Akkar et L. Goubin, *A Generic Protection against High-Order Differential Power Analysis*. In Proceedings of FSE'2003, LNCS 2887, Springer-Verlag, 2003.
- M.-L. Akkar, N.T. Courtois, R. Duteuil et L. Goubin, *A Fast and Secure Implementation of Sflash*. In Proceedings of PKC'2003, LNCS 2567, pp. 267-278, Springer-Verlag, 2003.
- L. Goubin, *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*. In Proceedings of PKC'2003, LNCS 2567, pp. 199-211, Springer-Verlag, 2003.

- N.T. Courtois, L. Goubin, W. Meier et J.D. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*. In Proceedings of PKC'2002, LNCS 2274, pp. 211-227, Springer-Verlag, 2002.
- L. Goubin, *A Sound Method for Switching between Boolean and Arithmetic Masking*. In Proceedings of CHES'2001, LNCS 2162, pp. 3-15, Springer-Verlag, 2001.
- N.T. Courtois, L. Goubin et J. Patarin, *FLASH, a fast multivariate signature algorithm*. In Proceedings of CT-RSA'2001, LNCS 2020, pp. 298-307, Springer-Verlag, 2001.
- N.T. Courtois, L. Goubin et J. Patarin, *QUARTZ, 128-bit long digital signatures*. In Proceedings of CT-RSA'2001, LNCS 2020, pp. 282-297, Springer-Verlag, 2001.
- N.T. Courtois et L. Goubin, *Cryptanalysis of TTM*. In Proceedings of ASIACRYPT'2000, LNCS 1976, pp. 44-57, Springer-Verlag, 2000.
- J.-S. Coron et L. Goubin, *On Boolean and Arithmetic Masking against Differential Power Analysis*. In Proceedings of CHES'2000, LNCS 1965, pp. 231-237, Springer-Verlag, 2000.
- L. Goubin et J. Patarin, *DES and Differential Power Analysis*. In Proceedings of CHES'99, LNCS 1717, pp. 158-172, Springer-Verlag, 1999.
- L. Goubin, A. Kipnis et J. Patarin, *Unbalanced Oil and Vinegar Signature Schemes*. In Proceedings of EUROCRYPT'99, LNCS 1592, pp. 206-222, Springer-Verlag, 1999.
- N.T. Courtois, L. Goubin et J. Patarin, *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*. In Proceedings of ASIACRYPT'98, LNCS 1514, pp. 35-49, Springer-Verlag, 1997.
- N.T. Courtois, L. Goubin et J. Patarin, *Improved Algorithms for Isomorphisms of Polynomials*. In Proceedings of EUROCRYPT'98, LNCS 1403, pp. 184-200, Springer-Verlag, 1998.
- L. Goubin et J. Patarin, *Asymmetric Cryptography with S-Boxes*. In Proceedings of ICICS'97, LNCS 1334, pp. 369-380, Springer-Verlag, 1997.
- L. Goubin et J. Patarin, *Trapdoor one-way permutations and multivariate polynomials*. In Proceedings of ICICS'97, LNCS 1334, pp. 356-368, Springer-Verlag, 1997.

3 Articles présentés à des colloques

- N.T. Courtois, L. Goubin, *Open Problems in Multivariate Cryptanalysis*. First STORK Cryptography Workshop, Bruges, 26-27 novembre 2002.
- L. Goubin, *Multivariate systems*. First STORK Cryptography Workshop, Bruges, 26-27 novembre 2002.
- N.T. Courtois, L. Goubin et J. Patarin, *FLASH, a fast asymmetric signature scheme for low-cost smartcards*. In Proceedings of the First Open NESSIE Workshop, Leuven, 13-14 novembre 2000.
- N.T. Courtois, L. Goubin et J. Patarin, *QUARTZ, an asymmetric signature scheme for short signatures on PC*. In Proceedings of the First Open NESSIE Workshop, Leuven, 13-14 novembre 2000.

- N.T. Courtois, L. Goubin et J. Patarin, *SFLASH, a fast asymmetric signature scheme for low-cost smartcards*. In Proceedings of the First Open NESSIE Workshop, Leuven, 13-14 novembre 2000.

- L. Goubin et J. Patarin, *Asymmetric Cryptography with Multivariate Polynomials over Finite Fields*. Workshop on Cryptography, Dagstuhl, Allemagne, septembre 1997.

4 Travaux de vulgarisation

- L. Goubin, *Stratégies d'attaques*. Pour la Science, numéro spécial «cryptographie», juillet 2002.

- L. Goubin et J. Patarin, *La Génération d'Aléas sur Ordinateur*. Les Nouvelles d'Archimède, mars-avril 2001.

- L. Goubin et J. Patarin, *La Génération d'Aléas sur Ordinateur*. Quadrature, n° 30, pp. 27-36, Éditions du Choix, novembre 1997.

5 Mémoires et rapports

- N.T. Courtois, L. Goubin et J. Patarin, *Asymmetric Cryptography with Multivariate Polynomials over a Small Finite Field*. Rapport Bull Smart Cards and Terminals, deuxième édition, septembre 1999.

- L. Goubin, *Sommes d'exponentielles à coefficients multiplicatifs*. Thèse de doctorat, Université Paris XI (Orsay), France, Janvier 1995.

- F.-X. Dehon et L. Goubin, *À propos de l'existence d'un équilibre avec des ensembles de production non convexes*. Mémoire de DEA, Université Paris I & École Polytechnique, France, Septembre 1992.

- L. Goubin, *Majoration de sommes trigonométriques*. Mémoire de DEA, Université Paris XI (Orsay), France, Octobre 1991.

6 Propositions à des organismes de normalisation

- N.T. Courtois, L. Goubin, M.-L. Akkar, R. Duteuil, *Digital Signatures and Low-Cost Smart Cards*. Présentation à IEEE P1363, Miami, 9 janvier 2003. Disponible sur `http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/SFLASH-P1363-Miami.PDF`

- N.T. Courtois, L. Goubin, J. Patarin, *SFLASH, a fast asymmetric signature scheme for low-cost smartcards*. Soumission à l'AFNOR pour la norme ISO/IEC 14888-3, *Information technology – Security techniques – Digital signatures with appendix – Part 3: Certificate-based mechanisms*, 19 décembre 2002.

- L. Goubin, *SFLASH, a Fast Asymmetric Signature Scheme for Low-Cost Smartcards*. Proposition pour IEEE P1363, Santa-Barbara, 22-23 août 2002. Disponible sur `http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/SFLASH-IEEEP1363.pdf`

- N.T. Courtois, L. Goubin, J. Patarin, *FLASH, a fast asymmetric signature scheme for low-cost smartcards*. Soumission NESSIE, version révisée, octobre 2001. Disponible sur `http://www.cryptonessie.org`
- N.T. Courtois, L. Goubin, J. Patarin, *QUARTZ, an asymmetric signature scheme for shot signatures on PC*. Soumission NESSIE, version révisée, octobre 2001. Disponible sur `http://www.cryptonessie.org`
- N.T. Courtois, L. Goubin, J. Patarin, *SFLASH, a fast asymmetric signature scheme for low-cost smartcards*. Soumission NESSIE, version révisée, octobre 2001. Disponible sur `http://www.cryptonessie.org`

7 Brevets

- L. Goubin, M.-L. Akkar et O. Ly, *Procédé de sécurisation d'un ensemble électronique exécutant un algorithme quelconque contre les attaques par introduction d'erreur(s)*. Déposé en France le 19 mars 2003.
- M.-L. Akkar, N.T. Courtois et L. Goubin, *Procédé et système de génération de signature*. Déposé en France le 13 septembre 2002.
- M.-L. Akkar et L. Goubin, *Procédé de sécurisation d'un ensemble électronique contre des attaques par introduction d'erreurs*. Déposé en France le 9 juillet 2002.
- M.-L. Akkar et L. Goubin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique d'ordre supérieur*. Déposé en France le 7 mars 2002.
- L. Goubin, *Procédé de sécurisation d'un ensemble électronique mettant en œuvre un algorithme cryptographique utilisant des opérations booléennes et des opérations arithmétiques, et système embarqué correspondant*. Déposé en France le 15 février 2001, numéro 01/02091.
- L. Goubin et J. Patarin, *Procédé pour mettre en œuvre une technique renforçant la sécurité des signatures à clé publique à base de polynômes multivariables*. Déposé en France le 29 septembre 2000, numéro 00/12403.
- L. Goubin, A. Kipnis et J. Patarin, *Procédé et système de signature à clé publique*. Déposé en Israël le 13 avril 2000, numéro 135647 (puis dans 9 pays).
- L. Goubin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à base d'exponentiation modulaire contre les attaques par analyse physique*. Déposé en France le 20 octobre 1999, numéro 99/13507.
- L. Goubin et J. Patarin, *Procédé de vérification de signature ou d'authentification*. Numéro de publication : 2792789. Déposé en France le 20 avril 1999 (puis dans 8 pays).
- L. Goubin et J. Patarin, *Procédé de sécurisation d'un ou plusieurs ensembles électroniques mettant en œuvre un même algorithme cryptographique avec clé secrète, une utilisation du procédé et l'ensemble électronique*. Numéro de publication : 2792141. Déposé en France le 9 avril 1999 (puis dans 5 pays).
- L. Goubin et J. Patarin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. Numéro de publication : 2789535. Déposé en France le 4 février 1999 (puis dans 5 pays).

Troisième partie

Annexe : articles joints

Nouvelles méthodes de cryptanalyse

Les cinq articles ci-dessous mettent en place de nouvelles techniques cryptanalytiques ainsi que leurs applications aux systèmes multivariables. L'étude de ces méthodes d'attaque vise non pas tant à «casser» de manière astucieuse les algorithmes existants qu'à mettre en place une batterie de tests pour détecter les faiblesses potentielles lors de la mise au point de nouveaux cryptosystèmes.

Page 153 – ICICS'97 (Version complète)
Trapdoor One-Way Permutations and Multivariate Polynomials.
Louis Goubin et Jacques Patarin

Cet article étudie plusieurs exemples d'algorithmes candidats pour être des permutations à sens unique avec trappe. La méthode de polarisation est introduite pour cryptanalyser deux premiers exemples (D^* et E^*). Le cryptosystème D^{**} et ses variantes sont ensuite définis, avec une analyse de leur sécurité.

Page 177 – ICICS'97 (Version complète)
Asymmetric Cryptography with S-Boxes.
Louis Goubin et Jacques Patarin

Une autre méthode de construction d'algorithmes asymétriques est présentée dans cet article, combinant l'utilisation de tables de transformation comme en cryptographie symétrique, et le principe de «représentation cachée» propre à la cryptographie multivariable. Plusieurs méthodes de cryptanalyse sont mises au point pour le problème sous-jacent de dégénérescence des formes quadratiques. L'algorithme $2R^-$ échappe à ces attaques.

Page 207 – EUROCRYPT'99 (Version complète)
Unbalanced Oil and Vinegar Signature Schemes.
Louis Goubin, Aviad Kipnis et Jacques Patarin

Cet article analyse l'algorithme UOV, qui est une réparation de l'algorithme «Oil and Vinegar» cassé par Kipnis et Shamir à CRYPTO'98. Une étude de la résolution des systèmes quadratiques, en termes de théorie de la complexité, et dans le cas particulier des systèmes sous-définis, est menée. Dans le même esprit que UOV, cet article contient la description du système HFEV.

Page 229 – ASIACRYPT'2000
Cryptanalysis of the TTM Cryptosystem.
Nicolas Courtois et Louis Goubin

Dans cet article, une cryptanalyse du cryptosystème TTM, inventé par T.-T. Moh, est décrite. Un des problèmes sur lesquels reposent la sécurité de TTM est le problème MinRank, que nous mettons en évidence. Plusieurs algorithmes pour

le résoudre sont proposés, dont la «méthode du noyau» qui permet de «casser» les instances de TTM proposées jusqu'à maintenant.

Page 243 – PKC'2002

Solving Underdefined Systems of Multivariate Quadratic Equations.
Nicolas Courtois, Louis Goubin, Willi Meier et Jean-Daniel Tacier

On poursuit ici l'étude des systèmes quadratiques sous-définis, commencée dans l'article d'EUROCRYPT'99. Plusieurs nouvelles méthodes de résolution sont proposées, ainsi qu'une amélioration dans le cas «massivement» sous-défini. Cela donne des critères précis pour choisi les paramètres dans un certain nombre d'algorithmes multivariables, notamment FLASH, SFLASH et UOV.

Trapdoor One-Way Permutations and Multivariate Polynomials

ICICS'97 *(Version complète)*
Article avec Jacques Patarin (Bull PTS)

Abstract

This article is divided into two parts. The first part describes the known candidates of trapdoor one-way permutations. The second part presents a new candidate trapdoor one-way permutation.

This candidate is based on properties of multivariate polynomials on finite fields, and has similar characteristics to T. Matsumoto, H. Imai, and J. Patarin's schemes.

What makes trapdoor one-way permutations particularly interesting is the fact that they immediately provide ciphering, signature, and authentication asymmetric schemes.

Our candidate performs excellently in secret key, and secret key computations can be implemented in low-cost smart-cards, *i.e.* without co-processors.

Key words : Trapdoor one-way permutations, multivariate polynomials, research of new asymmetric bijective schemes.

Note: In this extended version, we have taken into account the recent results of [5].

Part I: Known candidates

1 Introduction

Nobody can deny that the idea of trapdoor one-way permutation plays a very important role in cryptography. Many theoretic schemes use this concept as an "elementary block". Moreover, any candidate trapdoor one-way permutation can easily be transformed into a scheme of asymmetric cryptography, for ciphering, signature, as well as authentication.

Amazingly enough, no widespread paper exists that describes all the known candidates at present. As a result, many people think for example that RSA is the only explicit and available candidate today. As we will quickly see in the first part of this paper, it is only *almost* true. In fact, one can obtain many variants of RSA: by taking even exponents and a modified message space to keep bijectivity (Rabin-Williams), by using polynomial permutations that are different from the modular exponentiations (Dickson polynomials for instance), or by performing the computations in other groups (such as elliptic curves). There are also much less well known candidates, very

different from RSA, that are based on public forms given by multivariate equations (the original idea was first presented by T. Matsumoto and H. Imai).

All the bijective candidates of the "RSA-like" family are related to the factorisation problem on the integers. This link between the factorisation of the integers and the concept of trapdoor one-way permutation may seem surprising. This is a strong motivation to look for other ways of designing candidate trapdoor one-way permutations.

In this paper, we present a new example of such a candidate, that we have called D^{**}. One of the main interests of D^{**} lies in the fact that secret key computations are easy to implement: they are about 100 times faster than in 512 bits-RSA, and they require about 5 times less RAM. Therefore, D^{**} can be implemented in a smartcard without arithmetic co-processor (on the contrary, public key computations are supposed to be performed on a personal computer).

The security of D^{**}, as well as the security of other algorithms of the same family, cannot be related to a difficult problem as easily as RSA-like cryptosystems. Nevertheless, one can hope that these algorithms show interesting ways to build new candidates, or to discover new ideas in asymmetric cryptography.

2 Trapdoor one-way permutations

Let us recall the definition of a one-way function:

Definition: Let $f : A \to B$ be a function. f is said to be *one-way* if:
(i) Given $x \in A$, it is computationally easy to compute $y = f(x)$.
(ii) Given $y \in_f B$, it is computationally hard to compute $x \in A$ such that $f(x) = y$.

Note: In this definition, $\in_f$ means that y is "randomly" chosen in $f(A)$, where "randomly" means here that the probability of obtaining a value y is exactly:

$$\frac{|\{x \in A,\ f(x) = y\}|}{|A|}$$

Although many functions are thought to verify these two properties (they are called "candidate one-way functions"), nobody has ever *proven* that one-way functions exist. Moreover, in this paper, we will focus on a special class of them, namely the *trapdoor* one-way functions, which we define as follows:

Definition : Let $f : A \to B$ be a function. f is said to be trapdoor one-way if:
(i) f is a one-way function.
(ii) There is a secret information s such that, given $y \in_f B$ and s, it is computationally easy to compute $x \in A$ such that $f(x) = y$.

More precisely, we will only consider trapdoor one-way *permutations*, *i.e.* trapdoor one-way functions which are also bijective.

As we mentioned above, no function has ever been proven to be a trapdoor one-way permutation. If many "candidate one-way functions" are known, on the opposite, few candidate "trapdoor one-way functions" are known (they give essentially all the known asymmetric cryptosystems), and very few candidate "trapdoor one-way permutations" are known. We will now describe quickly all the candidate trapdoor one-way permutations we are aware of.

2.1 RSA

This is the most famous trapdoor one-way permutation. It was designed by Rivest, Shamir and Adleman in 1978 (cf [24]).

Suppose that n is the product of two large primes p and q, and let e be an integer. We consider the following function:

$$f : \begin{cases} \mathbf{Z}/n\mathbf{Z} \to \mathbf{Z}/n\mathbf{Z} \\ x \mapsto x^e \end{cases}$$

If e is chosen so as to be coprime to $\lambda(n) = \text{lcm}(p-1, q-1)$, and so that $e \not\equiv \pm 1 \bmod \lambda(n)$, then f is expected to be a trapdoor one-way permutation, whose corresponding secret information is the factorisation $n = pq$.

With this secret information, it is very easy to invert f:

$$f^{-1}(y) \equiv y^d \bmod n,$$

where $ed \equiv 1 \bmod \lambda(n)$ (d can be easily computed with Euclide's algorithm).

Note : It is obvious that:

(i) f is a permutation.

(ii) f is a trapdoor function.

What remains unclear is whether f is one-way. It has not been proven that factoring n is computationally hard (or that finding the secret exponent d, which can be proven equivalent to factoring n, is computationally hard). Furthermore, even if it is true, it is not clear whether we need factoring n (or computing d) to be able to compute $f^{-1}(y)$. These are two famous open problems.

2.2 Rabin-Williams

As we saw in the previous section, breaking RSA with modulus n has not be proven to be as difficult as factoring n. In 1979, Rabin (cf [23]) introduced the following modification of the scheme: instead of choosing e coprime to $\lambda(n)$, one can take $e = 2$. It can be shown that computing square roots is as difficult as factoring n. Unfortunately, the obtained function is no longer a permutation, which may make the decrypted messages ambiguous. One classical way to solve this problem is adding redundancy in the cleartext.

However, in 1980, Williams (cf [27]) showed a more elegant way to eliminate this problem: p and q are chosen so that $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$. We take $n = pq$, and a small integer s such that $(\frac{s}{n}) = -1$ (where $(\frac{\cdot}{\cdot})$ is the Jacobi symbol). n and s are public. Let $d = \frac{1}{2}(\frac{1}{4}(p-1)(q-1)+1)$. We define:

$$f : \begin{cases} \mathbf{Z}/n\mathbf{Z} \to Q_n \times \{0,1\} \times \mathbf{Z}/2\mathbf{Z} \\ x \mapsto \begin{cases} (x^2, 0, x \bmod 2) & \text{if } (\frac{x}{n}) = 1 \\ ((sx)^2, 1, sx \bmod 2) & \text{if } (\frac{x}{n}) = -1 \end{cases} \end{cases}$$

where Q_n is the set of quadratic residues in $\mathbf{Z}/n\mathbf{Z}$.

This function is a trapdoor permutation and it can be proven that finding a cleartext from a random ciphertext is as difficult as factoring.

Note: In 1985, Williams (cf [28]) extended this idea to $e = 3$ and $\mathbf{Z}[\omega]$ for the message space instead of $\mathbf{Z}$ (where ω is a primitive cube root of unity). In this public-key scheme, computing cleartexts from random ciphertexts is also provably as intractable as factoring n. In 1992, Loxton, Khoo, Bird and Seberry ([10]) gave another variant, with another choice for the complete set of residues used in defining the message space.

2.3 The Kurosawa-Itoh-Takeuchi cryptosystem

In [9], Kurosawa, Itoh and Takeuchi proposed the following trapdoor one-way permutation. Let $n = pq$ stands for the product of two large primes p and q, and let c be an integer such that $(\frac{c}{p}) = (\frac{c}{q}) = -1$ (where $(\frac{\cdot}{\cdot})$ is the Legendre symbol). If $D_{n,c} = \{y \in \mathbf{Z}/n\mathbf{Z},\ y^2 - 4c \in Q_n^*\}$ (where Q_n^* is the set of non-zero quadratic residues in $\mathbf{Z}/n\mathbf{Z}$), we define:

$$f : \begin{cases} (\mathbf{Z}/n\mathbf{Z})^* \to D_{n,c} \times \{0,1\} \times \{0,1\} \\ x \mapsto (x + (c/x) \bmod n,\ s,\ t) \end{cases}$$

where

$$s = \begin{cases} 0 & \text{if } (\frac{x}{n}) = 1 \\ 1 & \text{if } (\frac{x}{n}) = -1 \end{cases} \quad \text{and} \quad t = \begin{cases} 0 & \text{if } (c/x \bmod n) > x \\ 1 & \text{if } (c/x \bmod n) < x. \end{cases}$$

This function is a trapdoor permutation and it can be proven that finding a cleartext from a random ciphertext is as difficult as factoring.

2.4 Paillier's trapdoor one-way permutation

In [18], Pascal Paillier developped a new one-way trapdoor permutation over $\mathbf{Z}_{n^2}^*$. More precisely, let $n = pq$ stands for the product of two large primes p and q, let $\lambda = \lambda(n) = \text{lcm}(p-1, q-1)$ and let L be the function defined over $\mathcal{S}_n = \{u < n^2,\ u = 1 \bmod n\}$ by:

$$\forall u \in \mathcal{S}_n,\ L(u) = \frac{u-1}{n}.$$

If g is chosen such that $\gcd(\text{L}(g^\lambda \bmod n^2),\ n) = 1$, we define:

$$f : \begin{cases} \mathbf{Z}_{n^2}^* \to \mathbf{Z}_{n^2}^* \\ x = x_1 + nx_2 \mapsto g^{x_1} {x_2}^n \bmod n^2 \end{cases}$$

The function f is a trapdoor permutation, and it can be proven that it is one-way if and only if the so-called RSA[n,n] problem (*i.e.* extracting n-th roots modulo n, where $n = pq$ is of unknown factorisation) is hard.

2.5 Dickson polynomials

In [16] and [17] (see also [11]), Winfried Müller and Rupert Nöbauer developed a variant of RSA that makes use of Dickson polynomials, instead of the x^e monomial. This idea was generalized by Rudolph Lidl (see [12]) and W. Müller (see [15]). Basically, the schemes use the Dickson polynomials g_k defined by:

$$g_k(X) = \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \frac{k}{k-i} \binom{k-i}{i} (-1)^i x^{k-2i}.$$

Let $n = \prod_{i=1}^{r} p_i^{\alpha_i}$ and $v(n) = \text{lcm}(p_i^{\alpha_i - 1}(p_i^2 - 1),\ 1 \le i \le r)$. It can be proven that g_k is a permutation of $\mathbf{Z}/n\mathbf{Z}$ if and only if $\gcd(k, v(n)) = 1$, and that – in that case – the inverse of g_k is g_t, where $kt \equiv 1 \bmod v(n)$. From this property, it is easy to derive an analogue of RSA. Moreover, the only known method to invert g_k needs the factorisation of n, so that we have another candidate trapdoor one-way permutation based on the factoring problem.

Notes :

1. The Dickson polynomials are also known as Chebyshev polynomials of the first kind.
2. In 1993, the scheme of Müller and Nöbauer was re-invented (with minor differences) by P.J. Smith, who called it LUC (see [25] and [26]). This cryptosystem is formulated in terms of Lucas sequences. Some variations of LUC were also developed as (non bijective) analogies to the ElGamal scheme. Daniel Bleichenbacher, Wieb Bosma and Arjen K. Lenstra (see [1]) showed that – because of the deep links between Lucas sequences and exponentiation – all these variations of LUC, as well as RSA, are vulnerable to subexponential time attacks.

2.6 The Gong-Harn cryptosystem

In [7], Gong and Harn proposed the following trapdoor one-way permutation. Let $n = pq$ stands for the product of two large primes p and q, and let e be an integer such that

$$\gcd(e,p^2-1) = \gcd(e,p^3-1) = \gcd(e,q^2-1) = \gcd(e,q^3-1) = 1.$$

If a and b are two elements of $\mathbf{Z}/n\mathbf{Z}$, we define $s_e(a,b)$ and $s_{-e}(a,b)$ by the following equation:

$$X^3 - aX^2 + bX - 1 = (X-\alpha_1)(X-\alpha_2)(X-\alpha_3)$$

$$\Rightarrow (X-\alpha_1^e)(X-\alpha_2^e)(X-\alpha_3^e) = X^3 - s_e(a,b)X^2 + s_{-e}(a,b)X - 1.$$

We then consider:

$$f : \begin{cases} (\mathbf{Z}/n\mathbf{Z})^2 \to (\mathbf{Z}/n\mathbf{Z})^2 \\ (x_1,x_2) \mapsto (s_e(x_1,x_2), s_{-e}(x_1,x_2)). \end{cases}$$

This function is a trapdoor permutation and it can be proven that finding a cleartext from a random ciphertext is as difficult as factoring.

2.7 Elliptic curves

Another way to obtain analogues of RSA is to use elliptic curves over the ring $\mathbf{Z}/n\mathbf{Z}$ instead of the ring $\mathbf{Z}/n\mathbf{Z}$ itself to perform the computations. For any integer n, we denote by $E_n(a,b)$ the following elliptic curve:

$$E_n(a,b) = \{(x,y) \in (\mathbf{Z}/n\mathbf{Z})^2,\ y^2 \equiv x^3 + ax + b \bmod n\}.$$

– In 1991, Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto and Scott A. Vanstone (see [8]) proposed the following scheme: they choose two prime numbers p and q such that $p \equiv q \equiv 2 \bmod 3$, and an integer e coprime to $(p+1)(q+1)$. As in RSA, e, n are public, and p, q are secret. Each message is represented by an element $(x,y) \in (\mathbf{Z}/n\mathbf{Z})^2$. The encryption function is defined by:

$$f : \begin{cases} (\mathbf{Z}/n\mathbf{Z})^2 \to (\mathbf{Z}/n\mathbf{Z})^2 \\ (x,y) \mapsto e.(x,y) \end{cases} \qquad \text{the calculus being performed in } E_n(0, y^2 - x^3 - ax).$$

With the secrets d, it is very easy to invert f:

$$f^{-1}(x',y') = d.(x',y') \qquad \text{the calculus being performed in } E_n(0, y'^2 - x'^3 - ax'),$$

where $ed \equiv 1 \bmod \operatorname{lcm}(p+1, q+1)$.

Notes :

1. A variant chooses $p \equiv q \equiv 3 \bmod 4$ and performs the encryptions and decryptions in $E_n(a,0)$ instead of $E_n(0,b)$.
2. In the same way, we obtain an elliptic curve based analogue of the Rabin scheme.

As a result, this gives new candidate trapdoor one-way permutations, whose security is again based on the difficulty of factoring n. Moreover, these schemes seem to be more secure than RSA (or Rabin) against attacks without factoring, such as low multiplier attacks.

– In 1993, N. Demytko (see [3]) proposed another elliptic curve cryptosystem, whose security is also based on the difficulty of factoring $n = pq$, where p and q are secret prime integers. In this scheme, a *fixed* elliptic curve $E_n(a,b)$, with $\gcd(4a^3 + 27b^2, n) = 1$, and an integer e are chosen and made public. Each message is represented by an element $x \in \mathbf{Z}/n\mathbf{Z}$. The ciphertext $x' \in \mathbf{Z}/n\mathbf{Z}$ is defined as the first coordinate of the point $e.P \in E_n(a,b)$, where P is a point of the elliptic curve $E_n(a,b)$ whose first coordinate is x. There are explicit formulas giving x' in terms of x (and requiring neither the second coordinate of P, nor the secret parameters p and q), so that the encryption function above is well defined and can be performed by anyone. Moreover, when e is suitably chosen, an integer d can be computed with the secret parameters p and q, so that the cleartext of $x' \in \mathbf{Z}/n\mathbf{Z}$ is the first coordinate of $d.Q \in E_n(a,b)$, where Q is a point of $E_n(a,b)$ whose first coordinate is x'.

  The security of these candidate trapdoor one-way permutations also relies on the difficulty of factoring large composite numbers.

2.8 The C^* scheme

In 1988, Hideki Imai and Tsutomu Matsumoto proposed a very different public key scheme, called C^* (see [14]), that is based on multivariate polynomials over a finite field. The basic idea is to represent a message by an element $x \in K^n$, where $K = GF(2^m)$ is a public finite field, and n is a public integer. An integer θ such that $\gcd(1 + 2^{m\theta}, 2^{mn} - 1) = 1$, and an extension $\mathcal{L}_n$ of degree n over K are also public. The encryption function is defined by:

$$f : \begin{cases} K^n \to K^n \\ x \mapsto t(s(x)^{1+2^{m\theta}}) \end{cases}$$

where $s : K^n \to \mathcal{L}_n$ and $t : \mathcal{L}_n \to K^n$ are secret affine permutations. As f can be given by n public polynomials in n indeterminates over K, anyone can encrypt a message. It is easy to see that f is a trapdoor permutation. However, the C^* scheme was broken in 1995 by Jacques Patarin (see [19]), and therefore is not one-way.

Note: In [20], a way to repair the C^* scheme was suggested, but the new schemes (called HFE) are no longer bijective.

2.9 ABC

The C^* scheme we described in the previous section is not the only attempt of Matsumoto and Imai to design trapdoor one-way permutations (also called ABC, for Asymmetric Bijective Cryptosystems). In 1985 (see [13]), they proposed three schemes – called A, B and C – based on multivariate polynomials over finite fields.

– The first one is the same as the C^* scheme described in the section above.

– In the B scheme, $K = GF(2)$ and an integer n are public. The encryption function is:

$$f : \begin{cases} K^n \to K^n \\ x \mapsto \begin{cases} t((s(x) + c - 1) \bmod (2^n - 1) + 1) & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases} \end{cases}$$

where $s : K^n \to E$ and $t : E \to K^n$ are secret linear bijections, $E = \{k,\ 0 \leq k < 2^n\} = \left\{ \sum_{i=0}^{n-1} \alpha_i 2^i \right\}$ is considered as a vector space of dimension n over K, and c is a positive integer whose binary expression has small Hamming weight. The public-key is an "and-exclusive or" array pattern for the n-uple of n-variate sparse polynomials over K that represent f. We do not know if any cryptanalytic work has been done against this B scheme. Its security is – as far as we know – an open problem.

– In the C scheme, $K = GF(2^m)$ is public, and the encryption function is:

$$f : \begin{cases} K^4 \to K^4 \\ x \mapsto t(s(x)^2) \end{cases}$$

where $s : K^4 \to GL_2(K)$ and $t : GL_2(K) \to K^4$ are secret linear bijections, the set $GL_2(K)$ of 2×2 matrices over K being considered as a vector space of dimension 4 over K. The public key is the 4-uple of 4-variate quadratic polynomials over K that represent f. Moreover, with some minor changes, the scheme can be made bijective. However, in [22], we have presented a way to break this C scheme (the idea is to use the fact that $AB = BA$ when $B = A^2$ and A and B are two matrices).

Part II: Presentation of D^*

In part III, we will describe a new candidate trapdoor one-way permutation. However, in order to describe this candidate, called D^{**}, we need first to describe a scheme, called D^*. As we will see, this scheme D^* is not secure, but will be useful in part III. D^* and D^{**} have many analogies with the C^* scheme of section 2.5: the aim is to avoid the attacks and – in the same time – to keep the bijective properties.

3 D^*: a new tool

This section is devoted to the description of the D^* scheme used in encryption mode.

3.1 Representation of the message

A finite field $K = GF(q)$ is *public*, where $q = p^m$, m is odd, and p is a prime number such that $p \equiv 3 \bmod 4$ and p is not too small (for example $p = 251$; this point will be explained in section 3.6). Each message M is represented by n elements of K, where n is an *odd* and *public* integer (for example $n = 9$).

Note: We will see below that, with such q and n, -1 is not a square in $GF(q^n)$, and this property will be useful.

Moreover, we choose the representation x of M in the following message space:

$$\mathcal{M} = \left\{ x = (x_1,...,x_n) \in K^n,\ \exists k,\ 1 \leq k \leq n,\ x_k \in K' \text{ and } (\forall i < k,\ x_i = 0) \right\}$$

where K' is a complete set of residues of $K^*/\{\pm 1\}$.

Notes:

1. If $m = 1$, *i.e.* $K = GF(p)$, we can choose for example $K' = \{x.1_K,\ 1 \leq x \leq \frac{p-1}{2}\}$.
2. More generally, if $(e_1,...,e_m)$ is an arbitrary base of K over $GF(p)$, we can choose:

$$K' = \Big\{x = \sum_{i=1}^{m} x_i e_i \in K,\ \exists k,\ 1 \leq k \leq m,\ \Big(1 \leq x_k \leq \frac{p-1}{2}\Big) \text{ and } (\forall i < k,\ x_i = 0)\Big\}.$$

3. It is easy to verify on the examples above that:

$$|K'| = \frac{p-1}{2}(p^{m-1} + p^{m-2} + ... + p + 1) = \frac{p^m - 1}{2} = \frac{q-1}{2}$$

$$|\mathcal{M}| = \frac{q-1}{2}(q^{n-1} + q^{n-2} + ... + q + 1) = \frac{q^n - 1}{2}.$$

4. Any complete set of residues of $(K^n \setminus \{0\})/\{\pm 1\}$ can be chosen as the message space $\mathcal{M}$.

3.2 Encryption of $x \in \mathcal{M}$

The scheme also uses:

1. An extension $\mathcal{L}_n$ of degree n over K.
2. Two linear *secret* bijections $s : K^n \to \mathcal{L}_n$ and $t : \mathcal{L}_n \to K^n$. In a basis, these two linear permutations can be written as n polynomials in n variables over K, and of total degree one.

Note: $\mathcal{L}_n$ can be made public without reducing the security of the scheme, because changing $\mathcal{L}_n$ is equivalent to making other choices for s and t, so that $\mathcal{L}_n$ can be considered as a fixed extension.

With the preceeding notations, the ciphering algorithm can be described as follows. $y = F(x)$ is defined as the only element of $\{+t(s(x)^2), -t(s(x)^2)\}$ that belongs to $\mathcal{M}$ (this element exists and is unique, by construction of $\mathcal{M}$). Since s and t are of total degree one over K, $t(s(x)^2)$ can be given in a basis by n quadratic polynomials P_1, ... ,P_n in n variables, whose coefficients belong to K.

These polynomials are made public, so that anyone can easily encrypt a message, by using the following equations to compute $y = (y_1,...,y_n) = F(x)$ from $x = (x_1,...,x_n) \in \mathcal{M}$:

$$\begin{cases} y = (y_1,...,y_n) \in \mathcal{M} \\ y_1 = \pm P_1(x_1,...,x_n) \\ ... \\ y_n = \pm P_n(x_1,...,x_n) \end{cases}$$

3.3 Decryption of $y \in \mathcal{M}$

Under the hypothesis we have made ($p \equiv -1 \bmod 4$, m odd and n odd), it is easy to prove that, for any $y \in \mathcal{M}$ and $x \in K^n$:

1. If $t^{-1}(y)$ is a quadratic residue in $\mathcal{L}_n$, then

$$F(x) = y \Leftrightarrow t(s(x)^2) = y \Leftrightarrow x = \pm s^{-1}\Big((t^{-1}(y))^{\frac{q^n+1}{4}}\Big)$$

and we can choose the sign so as to ensure $x \in \mathcal{M}$.

2. If $t^{-1}(y)$ is not a quadratic residue in $\mathcal{L}_n$, then $-t^{-1}(y)$ is a quadratic residue in $\mathcal{L}_n$ (because $q^n \equiv -1 \bmod 4 \Rightarrow (-1)^{\frac{q^n-1}{2}} = -1 \Rightarrow -1$ is not a quadratic residue in $\mathcal{L}_n$). As a result:

$$F(x) = y \Leftrightarrow t(s(x)^2) = -y \Leftrightarrow x = \pm s^{-1}\Big(\big(-t^{-1}(y)\big)^{\frac{q^n+1}{4}}\Big) \Leftrightarrow x = \pm s^{-1}\Big((t^{-1}(y))^{\frac{q^n+1}{4}}\Big)$$

and the sign can also be chosen so that $x \in \mathcal{M}$.

Therefore, the encryption function F is a permutation from $\mathcal{M}$ to $\mathcal{M}$, whose inverse F^{-1} is easy to compute for anyone who knows the secret linear permutations s and t: for any $y \in \mathcal{M}$, $x = F^{-1}(y)$ is characterized by the following formula:

$$\begin{cases} x \in \mathcal{M} \\ x = \pm s^{-1}\Big((t^{-1}(y))^{\frac{q^n+1}{4}}\Big). \end{cases}$$

3.4 Complexity of the encryption and decryption algorithms

Encryption Obviously, using the public polynomials P_1, ..., P_n to encrypt a message requires $\leq \frac{n^3}{2}$ multiplications in $K = GF(q)$ and $\leq \frac{n^3}{2}$ additions in K. Since the complexity of a multiplication in K is $\mathcal{O}((\log q)^2)$, the encryption algorithm has a complexity $\mathcal{O}(n^3 m^2 (\log p)^2)$.

Note: Asymptotically, the complexity of a multiplication in $K = GF(q)$ is $\mathcal{O}(\log q . \log\log q)$ for very large q, but for our practical values of q, the algorithms are in $\mathcal{O}(\log^2 q)$, since the size of q is reasonable.

Decryption The decryption function is given by the following formula:

$$x = \pm s^{-1}\Big((t^{-1}(y))^{\frac{q^n+1}{4}}\Big).$$

Each linear transformation requires $\leq n^2$ multiplications and additions in $K = GF(q)$.

For the exponentiation, we can use the following identity:

$$\frac{q^n+1}{4} = \frac{q+1}{4}\Big[q(q-1)\sum_{i=0}^{\frac{n-3}{2}} q^{2i} + 1\Big].$$

If we use a normal base for $\mathcal{L}_n$ (*i.e.* a base that can be written $(\beta,\beta^q,\beta^{q^2},...,\beta^{q^{n-1}})$ for some $\beta \in \mathcal{L}_n$), we see that the complexity of the evaluation of the q^k-th power of an element of $\mathcal{L}_n$ can be neglected as compared to the complexity of the multiplication of two elements in $\mathcal{L}_n$.

Moreover, we can use the following remark: if we write the binary representation of $\frac{n-3}{2}$ as:

$$\frac{n-3}{2} = \sum_{\nu=1}^{N} 2^{\beta_\nu} \qquad (\beta_1 < \beta_2 < ... < \beta_N)$$

we can obtain the following identity:

$$\sum_{i=0}^{\frac{n-3}{2}} q^{2i} = \Big(\Big(...\Big(\Big(\prod_{j=\beta_{N-1}+1}^{\beta_N}(q^{2^j}+1)+q^{2.2^{\beta_N}}\Big)\prod_{j=\beta_{N-2}+1}^{\beta_{N-1}}(q^{2^j}+1)+q^{2(2^{\beta_N}+2^{\beta_{N-1}})}\Big)\prod_{j=\beta_{N-3}+1}^{\beta_{N-2}}(q^{2^j}+1)$$

$$+ ...\Big)\prod_{j=\beta_1+1}^{\beta_2}(q^{2^j}+1)+q^{2(2^{\beta_N}+...+2^{\beta_2})}\Big)\prod_{j=1}^{\beta_1}(q^{2^j}+1)+q^{2(2^{\beta_N}+...+2^{\beta_1})},$$

so that evaluating the $\sum_{i=0}^{\frac{n-3}{2}} q^{2i}$-th power of an element of $\mathcal{L}_n$ requires $\leq N + \beta_N \leq 3 \log n$ multiplications in $\mathcal{L}_n$ and $\leq 3 \log n$ evaluations of q^k-th powers in $\mathcal{L}_n$.

Note: This identity is a generalisation of an idea of T. Matsumoto and H. Imai, who used it to estimate the running time of their C^* scheme (cf [14], page 428).

In conclusion, the decryption algorithm requires at most $3n^2(\log n + \log q)$ multiplications or q^k-th exponentiations in $\mathcal{L}_n$, so that the complexity of the decryption algorithm is $\mathcal{O}(3(mn)^2(m \log p + \log n)(\log p)^2)$.

3.5 Complexity of solving quadratic systems in a field

To attack the cryptosystem described in this section, one of the most obvious ideas is trying to solve the following quadratic system:

$$\begin{cases} P_1(x_1,...,x_n) = \pm y_1 \\ ... \\ P_n(x_1,...,x_n) = \pm y_n \end{cases}$$

to find the cleartext x of a given ciphertext y.

However, we have proven that – whatever the field K may be – the *general* problem of solving a randomly selected system of multivariate quadratic equations over K is NP complete. This result was already known for $K = GF(2)$ (cf [6] page 251). Our proof of the general case is given in the appendix.

Notes:

1. In our scheme, it is not possible to choose $p = 2$, because $P_1,...,P_n$ would be of total degree one, and thus F would be a simple linear transformation, and so could be very easily inverted by gaussian reductions.
2. The problem the cryptanalyst has to cope with is a *particular* instance of a quadratic system over K. Therefore, the argument above does not prove that breaking the system is a NP-complete problem. Moreover, a classical theoretical argument of G. Brassard shows that breaking an encryption scheme is never a NP-complete problem.

3.6 The affine multiple attack

Another attack, which is very general, was described in [20]. It can be used against schemes based on a univariate polynomial transformation hidden by secret affine bijective transformations.

This attack is based on the following fact : if f is a univariate polynomial over a finite extension L of a finite field K, then by using a general algorithm (see for example [2]), one can compute an "affine multiple" of the polynomial $f(a) - b$, *i.e.* a polynomial $A(a,b) \in L[X,Y]$ such that:

1. Each solution of $f(a) = b$ is also a solution of $A(a,b) = 0$.
2. $A(a,b)$ is an affine function of a when written in a basis over K.

In the case of the D^* scheme, $L = \mathcal{L}_n$ and $f(a) = a^2$. It can be proven that any non-zero affine multiple $A(a,b)$ of f is at least of degree $\frac{p-3}{2}$ with respect to b. We have taken it for granted that p is not too small (a typical example is $p = 251$). With this hypothesis, the affine multiple attack does not threat the D^* scheme, because there is no practical way to compute $A(a,b)$.

4 First cryptanalysis of D^*

In this section, we prove that D^* is not secure.

– The cryptanalysis is based on the following identity:

$$uv = \frac{(u+v)^2 - (u-v)^2}{2},$$

which is valid because the characteristic of the field K is not 2. As a result:

$$\frac{F(x+x') - F(x-x')}{2} = \pm t(s(x) \cdot s(x')).$$

Therefore, $\phi(x,x') = t(s(x) \cdot s(x'))$ is given by n *public* bilinear forms with coefficients in K.

– We then compute the vector space of all the linear transformations C and D from K^n to K^n such that:

$$\forall x,x' \in K^n,\ C(\phi(x,x')) = \phi(D(x),x').$$

This vector space is at least of dimension n, because we can choose, for *any* $\lambda \in \mathcal{L}_n$:

$$\begin{cases} D(x) = s^{-1}(\lambda \cdot s(x)) \\ C(y) = t(\lambda \cdot t^{-1}(y)). \end{cases}$$

For simplicity, let us assume that the dimension is *exactly* n (we have made some simulations that confirm this property).
Since the set of solutions for C depends on n free variables, we can call these variables Λ_1, ..., Λ_n, and denote by C_Λ the solution with parameter $\Lambda = (\Lambda_1,...,\Lambda_n)$.

– We then compute the vector space of all linear transformations E from K^n to K^n such that:

$$C_{E(\Lambda)}(\tilde{y}) = C_{E(\tilde{y})}(\Lambda).$$

Here again, we find a vector space of dimension at least n.

Note: This is due to the fact that, by definition, $C_\Lambda(\tilde{y}) = t(\theta(\Lambda) \cdot t^{-1}(\tilde{y}))$, where θ is an unknown linear transformation from K_n to $\mathcal{L}_n$. Therefore, for any $\mu \in \mathcal{L}_n$, we can choose $E = \theta^{-1}(\mu \cdot t^{-1})$, and so obtain a solution.

Let E_0 be such a solution, and let $*$ be the operation such that, by definition:

$$\Lambda * \tilde{y} = \tilde{y} * \Lambda = C_{E_0(\Lambda)}(\tilde{y}).$$

Notes:

1. t and μ are still unknown, but $*$ has been found out.
2. By construction, it is easy to see that:

$$\exists \mu \in \mathcal{L}_n,\ \mu \neq 0,\ \forall \Lambda \in K^n,\ \forall \tilde{y} \in K^n,\ \Lambda * \tilde{y} = t(\mu \cdot t^{-1}(\Lambda) \cdot t^{-1}(\tilde{y})).$$

– We now compute, with the square-and-multiply principle (applied to the $*$ law):

$$\tilde{y}^{*(\frac{q^n+1}{4})} = \underbrace{\tilde{y} * ... * \tilde{y}}_{\frac{q^n+1}{4} \text{ times}} = t\Big(\mu^{\frac{q^n+1}{4}-1} \cdot t^{-1}(\tilde{y})^{\frac{q^n+1}{4}}\Big) = \pm t(\mu^{\frac{q^n+1}{4}-1} \cdot s(x)).$$

As a result, a linear transformation W from K^n to K^n exists, such that any cleartext/ciphertext pair (x,y) satisfies the following equation:

$$y^{*(\frac{q^n+1}{4})} = \pm W(x).$$

Moreover, W can be easily found by gaussian reductions on a few cleartext/ciphertext pairs. More precisely, let $(x[1],y[1])$, ..., $(x[k],y[k])$ be k cleartext/ciphertext pairs. According to the previous remark, there exist k elements of $\{-1,+1\}$, denoted by ϵ_1, ..., ϵ_k, such that:

$$\forall j,\ 1 \leq j \leq k,\ \epsilon_j x[j] - W^{-1}(y[j]^{*(\frac{q^n+1}{4})}) = 0.$$

As a result, we have nk equations (n equations for each value of j) and n^2+k unknown values (the n^2 coefficients of W^{-1} and the k variables ϵ_j). Moreover, if we suppose that $k \geq \frac{n^2}{n-1}$, we have $nk \geq n^2+k$, so that the system can be solved.

Note: The set of solutions of the system is a vector space of dimension 1, but we find only two solutions (which are opposite from each other) if the conditions $\epsilon_j = \pm 1$ $(1 \leq j \leq k)$ are taken into account. Moreover, to have a unique solution, we can suppose $\epsilon_1 = 1$ for example.

– After $*$ and W have been found, decrypting any ciphertext is easy, since:

$$x = \pm W^{-1}(y^{*(\frac{q^n+1}{4})}).$$

5 E^* cryptosystem - Description and cryptanalysis

– We call E^* the algorithm similar to D^*, but with $b = a^3$ instead of $b = a^2$, so that the encryption function G is given by:

$$y = G(x) = t(s(x)^3).$$

– As for D^*, the public key consists of n polynomials in n variables over K. For E^*, these polynomials are *cubic*.

– If we choose $p \equiv 2 \bmod 3$, m odd and n odd, then $\gcd(3,q^n-1) = 1$, so that $b = a^3 \Leftrightarrow a = b^h$, where h is the inverse of 3 modulo q^n-1. Under these asumptions, G is a trapdoor permutation of K^n, and the decryption function is given by:

$$x = G^{-1}(y) = s^{-1}(t^{-1}(y)^h).$$

– However, we are going to prove that the permutation G is not one-way. Moreover, the cryptanalysis is similar to that of D^*, and is based on the following identity:

$$uvw = \frac{(u+v+w)^3 + (u-v-w)^3 - (u-v+w)^3 - (u+v-w)^3}{24},$$

which is valid as soon as the characteristic of K is greater than 3. Thus:

$$\frac{G(x+x'+x'')+G(x-x'-x'')-G(x-x'+x'')-G(x+x'-x'')}{24} = t(s(x)\cdot s(x')\cdot s(x'')).$$

Therefore, $\psi(x,x',x'') = t(s(x)\cdot s(x')\cdot s(x''))$ is given by n public cubic forms with coefficients in K.
A $*$ law can then be derived exactly as in section 4 – by writing $s(x)\cdot s(x')\cdot s(x'')$ as $s(x)\cdot[s(x')\cdot s(x'')]$ – and finally, with a few cleartext/ciphertext pairs, any message can be easily decrypted.

6 Cryptanalysis of more general polynomial transformations

The algorithm used for the first cryptanalysis of D^* (*i.e.* with the polynomial $f(X) = X^2$), or for E^* (*i.e.* with the polynomial $f(X) = X^3$), can also be extended to any polynomial transformation $f(X)$ as long as the degree D of f is less than the characterisctic p of the field K.

Note: Moreover, when the degree D of the hidden polynomial f is smaller than p, then this degree D should be very small because it will also be the total degree of the public equations and the size of the public key must be reasonable.

So, let us assume $D < p$, and let $f(X) = \sum_{i=1}^{D} \alpha_i X^i$. The cryptanalysis is as follows:

Step 1: When the polarisation is done with more than D variables, we will have 0, and when it is done with less than D variables, we will not find 0. So the value D is easy to find by a few polarisations.

Step 2: Moreover, the polarisation with exactly D variables depends only on the monomial $\alpha_d X_d$ (all the other monomials give 0), so from this polarisation, we will find a $*$ law (we find a $*$ law on d variables, and this gives of course also a $*$ law on 2 variables: $X'_1 * (X'_2 * ... * X'_n)$).

Step 3: From the $*$ law found in step 2, we can find some values β_1, ..., β_D and a polynomial $f'(X) = \sum_{i=1}^{D} \beta_i X_i$ such that the public equations come from f'.

Step 4: Finally, a cleartext can be found from a ciphertext without knowing the secret affine functions s and t: we will just use f' and the $*$ law instead of f and the standard multiplication.

Remark: In the HFE schemes of [20] (with public polynomials of degree 2), the degree D of the hidden polynomial f is always larger than p because we want not only the monomials x and x^2 in f, so we must have at least one $x^{q^i+q^j}$ with $q^i + q^j > p$. So it seems that this "polarisation attack" does not work against the HFE schemes.

7 Another cryptanalysis of D^*

A few months after our first cryptanalysis (given in section 4), Nicolas Courtois found a very different cryptanalysis of D^*. We explain his cryptanalysis below.

Step 1: In the description of the D^* scheme, $s : K^n \mapsto \mathcal{L}_n$ is a *linear* and bijective application. Let us replace x by $x' = \sigma^{-1}(x)$ in the public equations of D^*, where $\sigma : K^n \mapsto K^n$ is an affine bijection, which is *not linear*. We obtain a new set of n public polynomials Q_1, ..., Q_n of total degree two in x'_1, ..., x'_n.

The first step of this attack consists in writing the affine applications $s \circ \sigma$ and t^{-1} as follows:

$$a = s \circ \sigma(x') = S_0 + \sum_{i=1}^{n} x'_i S_i,$$

$$b = t^{-1}(y) = \sum_{i=1}^{n} y_i T_i,$$

where S_0, S_1, ..., S_n and T_1, ..., T_n are elements of $\mathcal{L}_n$. Since s and t are bijective, $(S_i)_{1\leq i\leq n}$ and $(T_i)_{1\leq i\leq n}$ are two bases of $\mathcal{L}_n$. Moreover, since σ is not linear, we have $S_0 \neq 0$, and we can even suppose that $S_0 = 1$ (because for any $\lambda \in \mathcal{L}_n \setminus \{0\}$, we can change (s,t) into $(\frac{1}{\lambda} \cdot s, \lambda^2 \cdot t)$ and obtain the same cryptosystem.)

The D^* cryptosystem can thus be rewritten as follows:

$$\Big(1 + \sum_{i=1}^{n} x'_i S_i\Big)^2 = \sum_{i=1}^{n} y_i T_i.$$

Step 2: In the previous equation, if we replace each y_i by its public expression $Q_i(x'_1,...,x'_n)$, we obtain an equation of total degree two in the x'_i variables, which holds for any x:

$$1 + 2\sum_{i=1}^{n} x'_i S_i + \sum_{i=1}^{n} {x'_i}^2 S_i^2 + 2 \sum_{1\leq i\neq j\leq n} x'_i x'_j S_i S_j$$

$$= \sum_{i=1}^{n} \alpha_i T_i + \sum_{i=1}^{n} x'_i \Big(\sum_{j=1}^{n} \beta_{ij} T_j\Big) + \sum_{i=1}^{n} {x'_i}^2 \Big(\sum_{j=1}^{n} \gamma_{ij} T_j\Big) + \sum_{1\leq i\neq j\leq n} x'_i x'_j \Big(\sum_{k=1}^{n} \delta_{ijk} T_k\Big),$$

where the coefficients α_i, β_{ij}, γ_{ij}, δ_{ijk} are known elements of K.

If we successively take x = (0,...,0), x = (1,0,...,0), x = (0,1,0,...,0), ..., x = (0,...,0,1), x = (2,0,...,0), x = (0,2,0,...,0), ..., x = (0,...,0,2), x = (1,1,0,...,0), x = (1,0,1,0,...,0), ..., x = (0,...,0,1,1), we obtain the following $\frac{(n+1)(n+2)}{2}$ equations:

$$\begin{cases} 1 = \sum_{i=1}^{n} \alpha_i T_i & (1) \\ 2S_i = \sum_{j=1}^{n} \beta_{ij} T_j \quad (1 \leq i \leq n) & (2) \\ S_i^2 = \sum_{j=1}^{n} \gamma_{ij} T_j \quad (1 \leq i \leq n) & (3) \\ 2S_i S_j = \sum_{k=1}^{n} \delta_{ijk} T_k \quad (1 \leq i \neq j \leq n) & (4) \end{cases}$$

Step 3: Gaussian reductions on the equations (2) give the T_i as linear combinations of the S_j (the inversion is certainly possible, because $(S_i)_{1\leq i\leq n}$ is a basis of $\mathcal{L}_n$):

$$\forall i,\ 1 \leq i \leq n,\ T_i = \sum_{j=1}^{n} \theta_{ij} S_j. \qquad (*)$$

In the same way, we also have, from (1) and $(*)$:

$$1 = \sum_{j=1}^{n} \varphi_j S_j. \qquad (**)$$

Step 4: By using the equations $(*)$ in (3) and (4), we obtain a "multiplication table" for the S_i:

$$S_i \cdot S_j = \sum_{k=1}^{n} \nu_{ijk} S_k \qquad (1 \leq i,j \leq n).$$

The complexity of this first part of the attack is $\mathcal{O}(n^3)$ (due to the gaussian reductions used in step 3 to obtain the T_i as linear combinations of the S_j).

Step 5: With the "multiplication table" above, it is possible to find a cleartext $x = (x_1,...,x_n)$ from a given ciphertext $y = (y_1,...,y_n)$, as follows. By definition,

$$s \circ \sigma(x') = \pm(t^{-1}(y))^{\frac{q^n+1}{4}}$$

if we let $x' = \sigma^{-1}(x)$. With the notations of this section, this gives:

$$1 + \sum_{i=1}^{n} x'_i S_i = \pm\Big(\sum_{i=1}^{n} y_i T_i\Big)^{\frac{q^n+1}{4}}$$

By using $(*)$, we obtain:

$$1 + \sum_{i=1}^{n} x'_i S_i = \pm\Big(\sum_{j=1}^{n} \eta_j S_j\Big)^{\frac{q^n+1}{4}}$$

Then, by using the "square and multiply" principle, together with the "multiplication table" of Step 5, and equation $(**)$, we obtain:

$$\sum_{i=1}^{n} x'_i S_i = \sum_{j=1}^{n} \psi_j S_j,$$

where the ψ_j are known coefficients of K (there are two possibilities for the ψ_j, because of the $\pm$ above).

Since $(S_i)_{1\leq i\leq n}$ is a basis of $\mathcal{L}_n$, we deduce: $(x'_1,...,x'_n) = (\psi_1,...,\psi_n)$. Finally, for each of the two solutions x', we compute $x = \sigma(x')$ (σ is known), and the cleartext is the solution x that satisfies $x \in \mathcal{M}$.

8 How to find s and t in the D^* cryptosystem?

Following his attack of section 7, Nicolas Courtois has also found how to find the secret linear bijections s and t. We describe his method below. The notations are the same as in section 7.

Step 1: We choose a random value $x \neq 0$ in K^n. We can suppose that $a = s(x)$ is a generator of the (cyclic) multiplicative group of the algebraic extension $\mathcal{L}_n$. Actually, there are $\varphi(q^n - 1)$ such generators, so that the probability of finding one is:

$$\frac{\varphi(q^n-1)}{q^n-1} = \prod_{p|q^n-1}\left(1-\frac{1}{p}\right) \geq \prod_{p\leq q^n}\left(1-\frac{1}{p}\right).$$

According to Mertens' formula, we have:

$$\prod_{p\leq N}\left(1-\frac{1}{p}\right) \sim \frac{e^{-\gamma}}{\log N},$$

where γ is Euler's constant. Therefore, we can obtain a generator in $\mathcal{O}(n \log q)$ tries in average.

Step 2: With the notations above, we can write:

$$a = 1 + \sum_{i=1}^{n} x'_i S_i,$$

where $x' = \sigma^{-1}(x)$.

By using $(**)$, the "square and multiply" principle, and the "multiplication table" of section 7, we can write:

$$a^k = \sum_{i=1}^{n} \zeta_{ki} S_i \qquad (0 \leq k \leq n),$$

where the ζ_{ki} are known coefficients of K that can be computed in polynomial time.

Step 3: Since $\mathcal{L}_n$ is a vector space of dimension n over K, the vectors 1, a, a^2, ..., a^n are linked, *i.e.* we can find coefficients ξ_0, ..., ξ_n of K such that:

$$\sum_{k=0}^{n} \xi_k a^k = 0,$$

i.e. $\Phi(a) = 0$, where $\Phi(X) = \sum_{k=0}^{n} \xi_k X^k$ is a polynomial of degree n with coefficients in K.

Step 4: We then compute all the roots of Φ in $\mathcal{L}_n$. For example, with the Berlekamp-Rabin algorithm, they can be found in $\mathcal{O}(n^3 \log n \log q)$ in average. We obtain at most n roots. It remains to find which one is the correct value of $a = s(x)$.

Step 5: Let a_0 be one of the roots found in Step 4. We can notice that the elements 1, a_0, ..., a_0^{n-1} of $\mathcal{L}_n$ are linearly independent over K. Actually, if they were not, there would be a polynomial Π of degree $\leq n-1$ such that $\Pi(a_0) = 0$. The group generated by a_0 in $\mathcal{L}_n$ would therefore be contained in the set:

$$\{\rho(a_0),\ \rho \in K[X] \setminus \{0\},\ d^o\rho \leq n-2\},$$

whose cardinality is $q^{n-1} - 1 < q^n - 1$. As a result, a_0 could not be a generator of the multiplicative group of $\mathcal{L}_n$, a contradiction.

We thus obtain a system of n linearly independent equations on S_1, ..., S_n:

$$\forall k,\, 0 \leq k \leq n-1,\ \sum_{i=1}^{n} \zeta_{ki} S_i = a_0^k.$$

This system has a unique solution $(S_1,...,S_n)$.

When S is known, T can be deduced with equations $(*)$, and we can check if those S and T are correct.

We can repeat Step 5 with each root of Φ, until we find the correct one: that gives an algorithm in $\mathcal{O}(n^4 \log n \log q)$ to find S and T.

Note: A more practical method consists in generating another polynomial $\tilde{\Phi}$ of degree n over K such that $\tilde{\Phi}(a) = 0$ (by considering other powers of a in Step 3) and computing $\gcd(\Phi,\tilde{\Phi})$, whose degree is low with a high probability. The computation of the correct value of a is then likely to be much easier (we can expect the degree of $\gcd(\Phi,\tilde{\Phi})$ to be one with a rather high probability).

Part III: Our new candidate

9 The D^{**} algorithm

As we saw in the previous part, an encryption scheme based on the use of a monomial transformation of small degree (more precisely, of smaller degree than the characteristic of the field K) is insecure.

A natural idea is then to design a cryptosystem that uses *two* rounds of D^*-like transformations.

9.1 Representation of the message

We choose the same field K and the same message space $\mathcal{M}$ as in the description of D^* (see section 3).

9.2 Encryption of $x \in \mathcal{M}$

The scheme also makes use of an extension $\mathcal{L}_n$ of degree n over K (which can be fixed, as we mentioned before), and three linear secret bijections $s : K^n \to \mathcal{L}_n$, $t : \mathcal{L}_n \to \mathcal{L}_n$ and $u : \mathcal{L}_n \to K^n$ (each of them can be given by n polynomials in n variables over K, and of total degree one).

With these notations, the ciphering algorithm can be described as follows. $y = H(x)$ is defined as the only element of $\{+u(t(s(x)^2)^2), -u(t(s(x)^2)^2)\}$ that belongs to $\mathcal{M}$ (this element exists and is unique, by construction of $\mathcal{M}$). Since s, t and u are of total degree one over K, $u(t(s(x)^2)^2)$ can be given in a basis by n polynomials P_1, ..., P_n of total degree 4 in n variables, whose coefficients belong to K.

These polynomials are made public, so that anyone can easily encrypt a message, by using the following equations to compute $y = (y_1,...,y_n) = H(x)$ from $x = (x_1,...,x_n) \in \mathcal{M}$:

$$\begin{cases} y = (y_1,...,y_n) \in \mathcal{M} \\ y_1 = \pm P_1(x_1,...,x_n) \\ ... \\ y_n = \pm P_n(x_1,...,x_n) \end{cases}$$

9.3 Decryption of $y \in \mathcal{M}$

As for D^*, H is a permutation from $\mathcal{M}$ to $\mathcal{M}$, and the decryption of $y \in \mathcal{M}$ is also very easy, when the secret linear permutations s, t and u are known. For any $y \in \mathcal{M}$, $x = H^{-1}(y)$ is given by the following formula:

$$\begin{cases} x \in \mathcal{M} \\ x = \pm s^{-1}\Big((t^{-1}(u^{-1}(y))^{\frac{q^n+1}{4}})^{\frac{q^n+1}{4}}\Big). \end{cases}$$

10 Complexity of functional decomposition

In this section, we consider a natural attack on the D^{**} scheme. This attack consists in trying to "separate" the two rounds of D^{**}. This leads to the following problem:

Decomposition problem: Let g and h be two functions which map K^n into K^n and which are given by polynomials of total degree two in n variables over K. Then $f = g \circ h$ is also a function from K^n to K^n, and it is given by n polynomials of degree *four* in n variables over K. Suppose that f is given. Is it computationally feasible to recover g and h?

A positive answer to that problem would imply that the two rounds of D^{**} can be easily separated from each other. As a result, to break the scheme, we would only have to break two independent D^* schemes, and that is feasible as we saw in section 4.

That would of course make the general idea of using two rounds uninteresting. However, the decomposition of multivariate polynomials was studied by Matthew Dickerson, who gave an algorithm for the following problem:

Multivariate left decomposition: Given polynomials f and $h_1,...,h_n$ in $K[X_1,...,X_n]$, and an integer r, decide if there exists a polynomial $g(x_1,...,x_n)$ of total degree at most r that composes with the h_i's to give f. That is, does there exist a polynomial $g(x_1,...,x_n)$ such that

$$f(x_1,...,x_n) = g(h_1(x_1,...,x_n),...,h_n(x_1,...,x_n))$$

and $\deg(g) \leq r$? If so, determine the coefficients of g.

In [4], Dickerson presents the best-known algorithm for this problem, which is polynomial in the degree of $f,h_1,...,h_n$, but *exponential* in the number n of variables (note that our decomposition problem is even harder, because the h_i's are not known).

He also shows that the *general* problem of decomposition of multivariate polynomials is difficult, because the following one is NP-hard:

s-1-decomposition problem: Given a monic univariate polynomial $f(x)$ and an integer s, decide if there exists an s-1-decomposition of f, *i.e.* a monic univariate polynomial h of degree s, and a bivariate polynomial $g(y,x) \in K[Y,X]$ of the form $g(y,x) = \prod_{i=1}^{r}(y+\alpha_i x+\beta_i)$ with $\alpha_i,\beta_i \in \hat{K}$, an algebraic extension of K, such that $f(x) = g(h(x),x)$. If so, determine the coefficients of g and h.

There are some reasons to think that the following problem is also NP-hard (cf [4], problem 14, p. 74):

Multivariate decomposition of given degree: Given a polynomial f in $K[X_1,...,X_n]$ and some subset of the following: integers $k,r,s_1,...,s_k$, a polynomial $g(x_1,...,x_k) \in K[X_1,...,X_k]$, and polynomials $h_1(x_1,..,x_n),...,h_k(x_1,...,x_n)$, decide if there exists a functional decomposition $g,h_1,...,h_k$ of f such that $\deg g = r$, and $\deg h_i = s_i$ for $1 \leq i \leq n$. If so, compute those coefficients of g and the h_i's which were not given.

Dickerson (see [4], p. 75) notices that: "The s-1-decomposition problem seems intuitively easier than problem 14. In problem 14, f, g and h are general multivariate polynomials of arbitrary dimension. Furthermore polynomial g takes the polynomial h_i as arguments, and we know nothing about the form of g other than its degree. In the s-1-decomposition problem, on the other hand, f and h are both univariate polynomials and g is only bivariate. Furthermore, g takes x and not another polynomial as its second argument. We also know a great deal about the structure of the polynomial g, namely that it factors as: $g(y,x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$. However, we have tried without success to reduce the s-1-decomposition problem to problem 14."

Therefore, it is still an open problem... and – at the present – it does not lead to any practical attack on D^{**}.

11 Comparison with RSA in secret key computations

The aim of this section is to compare the speed of a realistic implementation of D^{**} with the speed of the standard 512 bits RSA cryptosystem.

We take $p = q = 251$, and $n = 9$, so that each message is about 72 bits large. By a careful study of the exponentiation $b \mapsto b^{\frac{q^n+1}{4}}$, it can be proved that D^{**} – in this example – requires less than 50 multiplications over $\mathcal{L}_n$ in secret key computations.

We can therefore summarize the complexity of secret key computations as follows:

$$\begin{cases} \leq 50 \text{ multiplications 72 bits} \times \text{72 bits} & \text{for } D^{**} \\ \simeq 768 \text{ multiplications 512 bits} \times \text{512 bits} & \text{for RSA.} \end{cases}$$

As a result, the secret key computations in D^{**} are expected to be at least 100 times faster than those of RSA.

12 TD^*: a D^{**} variation

The D^{**} algorithm is built with two rounds of D^* algorithms. We can also design a variation of this D^{**} scheme, called TD^*, where the first round will be a "triangular" or "mixed triangular/ D^*" scheme, and where the second round is still a D^*. By "triangular", we mean a transformation T of the following type:

$$T(a_1,\ldots,a_n) = (a_1^2, a_2^2 + q_2(a_1), a_3^2 + q_3(a_1,a_2), \ldots, a_n^2 + q_n(a_1,\ldots,a_{n-1})),$$

where $q_2,\ldots,q_n$ are homogeneous polynomials of total degree two.

By "mixed triangular/D^*" scheme, we mean a transformation f such that $f(A||B) = D^*(A)||T(B) + P(A)$, where $||$ is the concatenation function, where T is a "triangular" scheme, and where P is a homogeneous polynomial of total degree two. These TD^* schemes are also candidate trapdoor one-way permutations.

Note: The "triangular" or "mixed triangular/D^*" function must be in the first round. If the two rounds are put the other way round, the scheme can easily be broken.

13 Attacks agains 2R schemes

In 1999, in [5], an algorithm that is often able to find the decomposition of two quadratic multivariate polynomials has been published. This algorithm is expected to break the 2R schemes (among them D^{**}) when all the composition is given in the public key. In order to repair the 2R schemes (for example D^{**}), we can suggest:

- To not publish all the originally public equations (it gives a $2R^-$ scheme).
- Or to introduce a "perturbation" on these equations, for example by introducing some extra variables (it gives a 2RV scheme), by fixing some variables (ir gives a 2RF scheme), or by mixing the equations cith truly random equations (it gives a $2R^+$ scheme). Moreover, all these "perturbations" can be combined (it gives a $2R^{+-}$VF scheme). See [22] for more details (in [22], these "perturbations" are used and studied on the C^* scheme).

Note: When these perturbations are done on D^{**} or TD^*, no attacks are known against the resulting schemes (D^{**-}, D^{**+}, $D^{**}V$; $D^{**}F$, etc). However, unlike D^{**} or TD^*, these schemes are not bijective anymore...

14 Conclusion

From any trapdoor one-way permutation, it is easy to build asymmetric schemes for ciphering, signature, or authentication. However, very few candidate trapdoor one-way permutations are known at the present. Therefore, we think that all the candidates should be studied carefully, and that one should go on looking for new candidates.

In this paper, we have quickly described all the know candidates we are aware of. They can be split into two families: the "RSA-like" family and the "multivariate polynomial" family. The "RSA-like" family contains all the schemes that can be seen as generalizations of the RSA scheme (Rabin-Williams, Dickson, Elliptic curves analogues of RSA). In the "multivariate polynomial" family, the public key is given as a set of multivariate polynomials. The original C^* scheme of T. Matsumoto and H. Imai was a typical example of this family, but it is known to be insecure. However, it is possible to design some other schemes in this family, such as the B scheme [13], or such as the new schemes D^{**} and TD^* described in this paper.

Of course, the candidates of the "RSA-like" family look more "serious" than others. No candidates are known with an absolute proof of security, but the security of these candidates is related to a famous open problem, such as factoring, or computing e-th roots modulo n, whereas the security of the other candidates is just an open problem, not related to a famous problem.

After the publication of the 2R schemes at ICICS'97, a possible way to break these schemes was published in [5]. However, it is very easy to repair the schemes, *i.e.* to avoid the cryptanalysis of [5], for example by not publishing all the originally public equations. The resulting schemes, called $2R^-$, are not broken (but they are not bijective anymore...). For example for D^{**-} or TD^{*-}, no efficient attacks are known, and we keep the property that secret key computations are very easy. Secret key computations of D^{**-} or TD^{*-} are more than 100 times faster than RSA and they can be performed in low-cost smartcards (*i.e.* without co-processors). We hope that this paper will support the arising of new ideas in asymmetric cryptography.

References

[1] Daniel Bleichenbacher, Wieb Bosma, Arjen K. Lenstra, *Some Remarks on Lucas-Based Cryptosystems*, Advances in Cryptology, Proceedings of CRYPTO'95, Springer, pp. 386-396.

[2] Ian Blake, XuHong Gao, Ronald Mullin, Scott Vanstone, Tomik Yaghoobian, *Applications of finite Fields*, Kluwer Academic Publishers, p. 25.

[3] N. Demytko, *A New Elliptic Curve Based Analogue of RSA*, Advances in Cryptology, Proceedings of EUROCRYPT'93, Springer-Verlag, pp. 40-49.

[4] Matthew Dickerson, *The functional Decomposition of Polynomials*, Ph.D Thesis, TR 89-1023, Department of Computer Science, Cornell University, Ithaca, NY, July 1989.

[5] Y. Ding-Feng, L. Kwok-Yan, D. Zong-Duo, *Cryptanalysis of "2R" Schemes*, Proceedings of CRYPTO'99, Springer, pp. 315-325.

[6] Michael Garey, David Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman, p. 251.

[7] Guang Gong, Lein Harn, *Public-Key Cryptosystems Based on Cubic Finite Field Extensions*, to appear in IEEE Transactions on Information Theory.

[8] Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto, Scott A. Vanstone, *New Public-Key Schemes Based on Elliptic Curves over the Ring* $\mathbf{Z}_n$, Advances in Cryptology, Proceedings of CRYPTO'91, Springer-Verlag, pp. 252-266.

[9] Kaoru Kurosawa, Toshiya Itoh, Masashi Takeuchi, *Public key cryptosystem using a reciprocal number with the same intractactability as factoring a large number*, Cryptologia, vol. 12, n° 4, pp. 225-233, 1988.

[10] John H. Loxton, David S.P. Khoo, Gregory J. Bird, Jennifer Seberry, *A Cubic RSA Code Equivalent to Factorization*, Journal of Cryptology, v.5, n.2, 1992, pp. 139-150.

[11] Rudolf Lidl, G.L. Mullen, G. Turwald, *Pitman Monographs and Surveys in Pure and Applied Mathematics 65: Dickson Polynomials*, London, Longman Scientific and Technical, 1993.

[12] Rudolf Lidl, Winfried B. Müller, *Permutation Polynomials in RSA-Cryptosystems*, Advances in Cryptology, Proceedings of CRYPTO'83, Plenum Press, 1984, pp. 293-301.

[13] Tsutomu Matsumoto, Hideki Imai, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, AAECC-3, Grenoble, 1985.

[14] Tsutomu Matsumoto, Hideki Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Advances in Cryptology, Proceedings of EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[15] Winfried B. Müller, *Polynomial Functions in Modern Cryptology*, Contributions to General Algebra 3: Proceedings of the Vienna Conference, Vienna: Verlag Hölder-Pichler-Tempsky, 1985, pp. 7-32.

[16] Winfried B. Müller, Rupert Nöbauer, *Some Remarks on Public-Key Cryptography*, Studia Scientiarum Mathematicarum Hungarica, v.16, 1981, pp. 71-76.

[17] Winfried B. Müller, Rupert Nöbauer, *Cryptanalysis of the Dickson-scheme*, Advances in Cryptology, Proceedings of EUROCRYPT'85, Springer-Verlag, pp. 50-61.

[18] Pascal Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology, Proceedings of EUROCRYPT'99, Springer, pp. 223-238.

[19] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Advances in Cryptology, Proceedings of CRYPTO'95, Springer, pp. 248-261.

[20] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Advances in Cryptology, Proceedings of EUROCRYPT'96, Springer, pp. 33-48.

[21] Jacques Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Advances in Cryptology, Proceedings of CRYPTO'96, Springer, pp. 45-60.

[22] Jacques Patarin, Louis Goubin, Nicolas Courtois, C^*_{-+} *and HM: Variations around two schemes of T. Matsumoto and H. Imai*, Advances in Cryptology, Proceedings of ASIACRYPT'98, Springer, pp. 45-60.

[23] M.O. Rabin, *Digitized Signatures and Public-Key Functions as Intractable as Factorization*, Technical Report LCS/TR-212, M.I.T. Laboratory for Computer Science, 1979.

[24] R.L. Rivest, A. Shamir, L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, v.21, n.2, 1978, pp. 120-126.

[25] P.J. Smith, *LUC Public-Key Encryption*, Dr. Dobb's Journal, January 1993, pp. 44-49.

[26] P.J. Smith, M.J.J. Lennon, *LUC: a New Public Key System*, Proceedings of the Ninth IFIP Int. Symp. on Computer Security, 1993, pp. 103-117.

[27] H.C. Williams, *A Modification of the RSA Public-Key Encryption Procedure*, IEEE Transactions on Information Theory, v.IT-26, n.6, 1980, pp. 726-729.

[28] H.C. Williams, *An* M^3 *Public-Key Encryption Scheme*, Advances in Cryptology, Proceedings of CRYPTO'85, Springer-Verlag, pp. 358-368.

Appendix

Solving a system of quadratic equations over any field is NP-complete

It is known that solving a randomly selected system of multivariate quadratic equations over the field $K = GF(2)$ is an NP-complete problem (see [6] page 251). In this appendix, we show that this is still the case when K is *any* field. Moreover, as concerns the case $K = GF(2)$, Garey and Johnson refer to a paper that was never published, so that we give a proof for this case too.

1. The case $K = GF(2)$

Let us consider an instance of the 3-Satisfiability problem (also called 3-SAT), given by a finite set $U = \{u_1,...,u_n\}$ of Boolean variables, and a collection $C = \{c_1,...,c_m\}$ of clauses on X. By definition, each clause is a disjonction of at most three litterals over U (a litteral is some u or some $\bar{u}$, with $u \in U$).

Each Boolean variable can be considered in an obvious way as an element of $GF(2)$. Moreover:

- If $u \in U$ corresponds to $x \in GF(2)$, then $\bar{u}$ correponds to $1 - x$.
- If $u \in U$ (resp. $v \in U$) corresponds to $x \in GF(2)$ (resp. $y \in GF(2)$), then (u or v) corresponds to $(xy + x + y)$ in $GF(2)$.

Therefore, finding a truth assigment for U that satisfies all the clauses in C is equivalent to solving some system of m *cubic* equations in n indeterminates over $GF(2)$. Moreover, each equation of this system contains at most three of the x_i indeterminates. If we add the $\frac{n(n-1)}{2}$ new variables $z_{ij} = x_i x_j$ $(i < j)$, the system can be rewritten into a system of $m + \frac{n(n-1)}{2}$ *quadratic* equations in $\frac{n(n+1)}{2}$ indeterminates over $GF(2)$.

Conclusion The problem of solving a randomly selected instance of the 3-SAT problem – which is NP-complete – can be reduced in polynomial time to the problem of solving a system of randomly selected multivariate quadratic equations over the $GF(2)$. As a result, this latter problem is also NP-complete.

2. The general case

Let K be *any* field, and let $\mathcal{S}$ be the following system of n quadratic equations in n indeterminates over $GF(2)$:

$$\sum_{1\leq i<j\leq n} \mu_{ijk}x_ix_j + \sum_{i=1}^{n} \nu_{ik}x_i = y_k \qquad (1 \leq k \leq n),$$

where the y_k are fixed elements of $GF(2)$. We are going to show that the problem of solving the system $\mathcal{S}$ can be reduced – in polynomial time – to the problem of solving some system of quadratic equations over K.

Transformation of the system Over $GF(2)$, solving $(\mathcal{S})$ is equivalent to solving the following system:

$$(\mathcal{S}')\begin{cases} \mu_{12k}x_1x_2 = z_{12k} & (1 \leq k \leq n) \\ \mu_{13k}x_1x_3 = z_{12k} + z_{13k} & (1 \leq k \leq n) \\ \quad \ldots\ldots\ldots\ldots & \\ \mu_{(n-1)nk}x_{n-1}x_n = z_{(n-2)nk} + z_{(n-1)nk} & (1 \leq k \leq n) \\ \nu_{1k}x_1 = z_{(n-1)nk} + w_{1k} & (1 \leq k \leq n) \\ \nu_{2k}x_2 = w_{1k} + w_{2k} & (1 \leq k \leq n) \\ \quad \ldots\ldots\ldots\ldots & \\ \nu_{(n-1)k}x_{n-1} = w_{(n-2)k} + w_{(n-1)k} & (1 \leq k \leq n) \\ \nu_{nk}x_n = w_{(n-1)k} + y_k & (1 \leq k \leq n) \end{cases}$$

It is a system of $\frac{n^2(n+1)}{2}$ equations over $GF(2)$, with $\frac{n^2(n+1)}{2}$ indeterminates: the x_i $(1 \leq i \leq n)$, the z_{ijk} $(1 \leq i < j \leq n,\ 1 \leq k \leq n)$, and the w_{ik} $(1 \leq i \leq n-1,\ 1 \leq k \leq n)$.

From $GF(2)$ to K Each element $x \in \{0,1\}$ of $GF(2)$ can be considered as an element x of K (where 0 denotes the identity element of the addition law of K, et 1 denotes the identity element of the multiplication law of K) such that $x(x-1) = 0$. The multiplication law $(x,y) \mapsto xy$ of $GF(2)$ can be translated into $(x,y) \mapsto xy$ over K, whereas one can translate the addition law $(x,y) \mapsto x+y$ of $GF(2)$ into $(x,y) \mapsto (x+y)(2-(x+y))$ over K (where 2 denotes the element $1+1$ of K).

As a result, solving $(\mathcal{S}')$ over $GF(2)$ is equivalent to solving the following system over K:

$$(\mathcal{S}'')\begin{cases}
\mu_{12k}x_1x_2 = z_{12k} & (1 \le k \le n)\\
\mu_{13k}x_1x_3 = (z_{12k} + z_{13k})(2 - z_{12k} - z_{13k}) & (1 \le k \le n)\\
\quad \ldots\ldots\ldots\ldots & \\
\mu_{(n-1)nk}x_{n-1}x_n = (z_{(n-2)nk} + z_{(n-1)nk})(2 - z_{(n-2)nk} - z_{(n-1)nk}) & (1 \le k \le n)\\
\nu_{1k}x_1 = (z_{(n-1)nk} + w_{1k})(2 - z_{(n-1)nk} - w_{1k}) & (1 \le k \le n)\\
\nu_{2k}x_2 = (w_{1k} + w_{2k})(2 - w_{1k} - w_{2k}) & (1 \le k \le n)\\
\quad \ldots\ldots\ldots\ldots & \\
\nu_{(n-1)k}x_{n-1} = (w_{(n-2)k} + w_{(n-1)k})(2 - w_{(n-2)k} - w_{(n-1)k}) & (1 \le k \le n)\\
\nu_{nk}x_n = (w_{(n-1)k} + y_k)(2 - w_{(n-1)k} - y_k) & (1 \le k \le n)\\
x_i(x_i - 1) = 0 & (1 \le i \le n)\\
z_{ijk}(z_{ijk} - 1) = 0 & (1 \le i < j \le n,\ 1 \le k \le n)\\
w_{ik}(w_{ik} - 1) = 0 & (1 \le i \le n-1,\ 1 \le k \le n)
\end{cases}$$

It is a system of $n^2(n+1)$ quadratic equations with $\frac{n^2(n+1)}{2}$ indeterminates over K.

Conclusion The problem of solving a randomly selected system of multivariate quadratic equations over $GF(2)$ – which is NP-complete – can be reduced in polynomial time to the problem of solving a system of randomly selected multivariate quadratic equations over the field K. Therefore, this latter problem is also NP-complete. Note that K can even be a division ring, because the proof does not use the commutativity of the multiplication law in K.

Asymmetric Cryptography with S-Boxes

Is it easier than expected to design efficient asymmetric cryptosystems?

ICICS'97 *(Version complète)*
Article avec Jacques Patarin (Bull PTS)

Abstract

In [12], T. Matsumoto and H. Imai designed an asymmetric cryptosystem, called C^*, for authentication, encryption and signature. This C^* scheme was broken in [13] due to unexpected algebraic properties.

In this paper, we study some new "candidate" asymmetric cryptosystems based on the idea of hiding one or two rounds of small S-box computations with secret functions of degree one or two. The public key is given by multivariate polynomials of small degree. The C^* scheme (when its n_i values are small) can be seen as a very special case of these schemes, but in the new schemes, the algebraic properties of [13] generally do not exist, so that completely different cryptanalytic ideas have to be found. We study the efficiency of classical cryptanalysis (such as differential cryptanalysis), and we also present completeley new cryptanalytic tools (such as "gradient cryptanalysis"). With these cryptanalysis, most of the "new" algorithms can be broken and we deduce some very different cryptanalysis of C^*. Moreover, our cryptanalysis and the cryptanalysis of [13] can also be combined in order to faster compute a cleartext from a ciphertext, and to find more informations on the secret key. Thus one of the interests of the paper is to improve the cryptanalysis of C^*.

However, we were not able to find the cryptanalysis of all the new schemes. More precisely, when one round of secret quadratic functions is combined with one round of S-boxes, or when two rounds of S-boxes are carefully hidden by affine functions, the security of these schemes is surprisingly still an open problem. Another interest of the paper lies therefore in the highlighting of these new schemes. The main practical advantage of these schemes is that secret computations are easy and can be performed in low-cost smartcards.

Key words: New asymmetric algorithms, multivariate polynomials, differential cryptanalysis, efficient asymmetric algorithms for smartcards.

Note: In this extended version, we have taken into account the recent results of [1] and [4].

1 Introduction

In [18], Bruce Schneier wrote : "Any algorithm that gets its security from the composition of polynomials over a finite field should be looked upon with skepticism, if not outright suspicion". Moreover, in the same page, he also wrote : "Prospects for creating radically new and different public-key cryptography algorithms seem dim". Maybe Bruce Schneier is right. Nevertheless we will try in this paper to design new public key cryptosystems that get their security... from the composition of polynomials over a finite field ! Moreover, the design of our schemes seems to be amazingly simple... However, we must say that we have no proof of security for these schemes and we will not be shocked if these schemes are looked upon with skepticism, if not outright suspicion.

We will see that, as expected, the easier schemes are not secure but for some more complex schemes ("two-round schemes"), the security is still an open problem.

It should be noticed that the polynomials in the schemes are multivariate polynomials and that Bruce Schneier's declarations were essentially motivated by results on univariate polynomials. The complexity results may be completely different for multivariate polynomials. For example, solving a univariate polynomial equation of small degree d in a finite field is feasible (the complexity is polynomial in d), but solving a multivariate set of polynomial equations of small degree d in a finite field K is NP-hard, even when $K = GF(2)$ and $d = 2$ (cf [8]). Or finding the functional decomposition of a univariate polynomial is often easy (see [19] and [20]) but finding the functional decomposition of multivariate polynomials seems to have a complexity exponential in the number of variables, even with the best known algorithms (see [5] or [6] p. 86). Moreover the general problem of computing a multivariate polynomial decomposition is NP-hard (see [5] or [6] p. 87).

In fact, these hard problems are well known motivations to try to design new asymmetric cryptosystems with multivariate polynomials. In [7] a first design idea was studied by Harriet Fell and Whitfield Diffie but, as they pointed out, their design was not efficient because their function F and its inverse F^{-1} were multivariate polynomials with the same degree d. In [12] Tsutomu Matsumoto and Hideki Imai designed a very efficient scheme (called C^*) with a function F of total degree two, such that the degree of F^{-1} was much larger than two. However this scheme was broken in [13] due to unexpected algebraic properties. In [14] another scheme, called HFE, was designed by Jacques Patarin to avoid these unexpected algebraic properties, but the secret key computations of HFE are not as efficient as in the original Matsumoto-Imai scheme C^*.

The aim of this paper is to introduce and to study "candidate" schemes, whose interest lies in their very simple design, and in the very good efficiency of the secret key computations. More precisely, the secret key computations will be easy in low-cost smartcards because very little RAM is needed and because the number of computations to be performed is very moderate. The main idea of those schemes is to use small S-boxes where some random multivariate functions of small degree are stored, and to combine such a function with some secret multivariate functions of small degree. The public key is sometimes large, but its length is still polynomial in the length of the messages (moreover, we can expect that secret key computations are performed on smartcards, and public key computations are performed on personal computers).

An important part of the paper deals with the efficiency of very different cryptanalytic ideas on these schemes, such as differential cryptanalysis, canonical representation of multivariate polynomials of degree two, or affine multiple attacks. Moreover we introduce some new cryptanalytic

ideas, such as for example what we have called "gradient cryptanalysis". A new attack on Matsumoto and Imai's C^* cryptosystem is also described: we will see that it is always possible to "separate" the "branches" of the scheme from each other, whatever their size may be. We will like this be able to recover more of the secret key than in [13], and to faster compute cleartexts from ciphertexts.

Although the easier variations of our schemes can be broken by using those ideas, some of our schemes seem to resist cryptanalysis so far, and may therefore be interesting candidates for new and efficient asymmetric cryptosystems.

Part I: Cryptanalysis of One Round Schemes

2 One round of S-boxes: description of the schemes

Aim of this section

In section 2, we introduce the basic principles of the new schemes we deal with in this paper. As we pointed out above, transformations which are based on algebraic structures can often be used by the cryptanalyst to break the scheme. That is the reason we will use S-boxes, which are supposed to be built with randomly chosen polynomials of small degree, and thus have no algebraic structure. Moreover, the design of the scheme will be such that the encryption function is polynomial with a small degree (typically 2 or 4), but its inverse is polynomial with a high degree, and so cannot be found by Gaussian reductions.

Section 2 is about the simplest way of using these ideas: there will be only one round of S-boxes, hidden by two affine bijective transformations. However, we will see (in section 3) that this first scheme is not secure, whatever the degree of the S-boxes may be.

Description of the schemes

A finite field K, with $q = p^m$ elements is public. For example, we may take $K = GF(2)$, the finite field with two elements.

Each message will be represented by n elements of $GF(q)$, where n is another public integer. n is split into d integers n_1,...,n_d, such that $n = n_1 + ... + n_d$. For each e, $1 \leq e \leq d$, we have a S-box S_e which maps K^{n_e} into K^{n_e}, and is given by quadratic polynomials of the n_e variables.

The secret items will be:

1. Two affine bijections s and t from K^n to K^n.
2. The separation of n into d integers: $n = n_1 + ... + n_d$.
3. The S-boxes: S_1,...,S_d.

Let $x = (x_1,...,x_n)$ be the cleartext, and $y = (y_1,...,y_n)$ be the corresponding ciphertext. If we have the secrets, then we can obtain y from x as follows (we will see below how to obtain y from x without the secrets):

1. $x = (x_1,...,x_n)$ is first transformed with the affine secret permutation s into $s(x) = a = (a_1,...,a_n)$.
2. Then we compute

$$\begin{cases} (b_1,...,b_{n_1}) = S_1(a_1,...,a_{n_1}) \\ \quad ..... \\ (b_{n_1+...+n_{d-1}+1},...,b_{n_1+...+n_d}) = S_d(a_{n_1+...+n_{d-1}+1},...,a_{n_1+...+n_d}) \end{cases}$$

where for each e, $1 \leq e \leq d$, S_e is the secret quadratic S-box which maps K^{n_e} into K^{n_e}.

3. Finally, $b = (b_1,...,b_n)$ is transformed with another affine secret permutation t, into $t(b) = y = (y_1,...,y_n)$.

It is easy to see that the composition of all these operations is still a quadratic function of the components of x, so that it can be given by n polynomials of degree two in $x_1,...,x_n$. Each S-box can be seen as a "branch" of the algorithm.

The public items are:

1. The field K and the length n of the messages.
2. The n polynomials $P_1,...,P_n$ in n variables over K.

So anyone can encipher a message (because from $P_1,...,P_n$, anyone can obtain y from x).

Moreover, if the secret items are known, a message can be deciphered like this:

1. From $y = (y_1,...,y_n)$, we compute $b = (b_1,...,b_n) = t^{-1}(y_1,...,y_n)$.
2. Then we can find $(a_1,...,a_n)$ by looking in a table in which all the values of the S-box functions have been precomputed and sorted in lexicographic order.
3. Finally, we compute $x = (x_1,...,x_n) = s^{-1}(a_1,...,a_n)$.

Note 1: The parameters must be chosen so that the tables for the S-boxes are not too large. For example, we can take $K = GF(2)$, $n = 64$ and $n_1 = ... = n_8 = 8$, so that each table contains $2^8 = 256$ values.

Note 2: The S-boxes are not necessarily one-to-one maps. So there may be several possible cleartexts for a given ciphertext. One solution to avoid this ambiguity is to put some redundancy in the representation of the messages, by making use of an error correcting code or a hash function (for details, see [14] p. 34, where a similar idea is used in a different scheme).

Note 3: The scheme can also be used in signature. To sign a message M, the basic idea is to compute x from $y = h(R||M)$ (as if we were deciphering a message), where h is a hash function and R is a small pad. If we succeed, (x,R) will be the signature of M. If we do not succeed (because the function is not a bijection), we try another pad R (for variants and details, see [14], where a similar idea is used). It could also be noticed that, with the variation described below (with some quadratic functions q_1, ..., q_d), we will have a larger probability to suceed (*i.e.* we will have to try fewer values R).

Note 4: We said that the S-box can be inverted by looking in a table. More precisely, we always have three different ways to invert a S-box:

1. Looking in a table.
2. Trying all the possible inputs in order to find the right ones ("exhaustive search").
3. Solving the polynomial equations of the S-box. Since the S-box uses only a very small number of variables, these equations can often be inverted by algebraic algorithms (which are not efficient with more variables).

Note 5: The C^* scheme of T. Matsumoto and H. Imai can be seen as a very special case of our algorithm above, when, in C^*, all the n_i variables are small (see [12] for the definition of C^* and of the n_i variables). Therefore, the attacks of the next three sections can be applied against C^* with small n_i parameters, so that we obtain a different cryptanalysis of that scheme. Moreover,

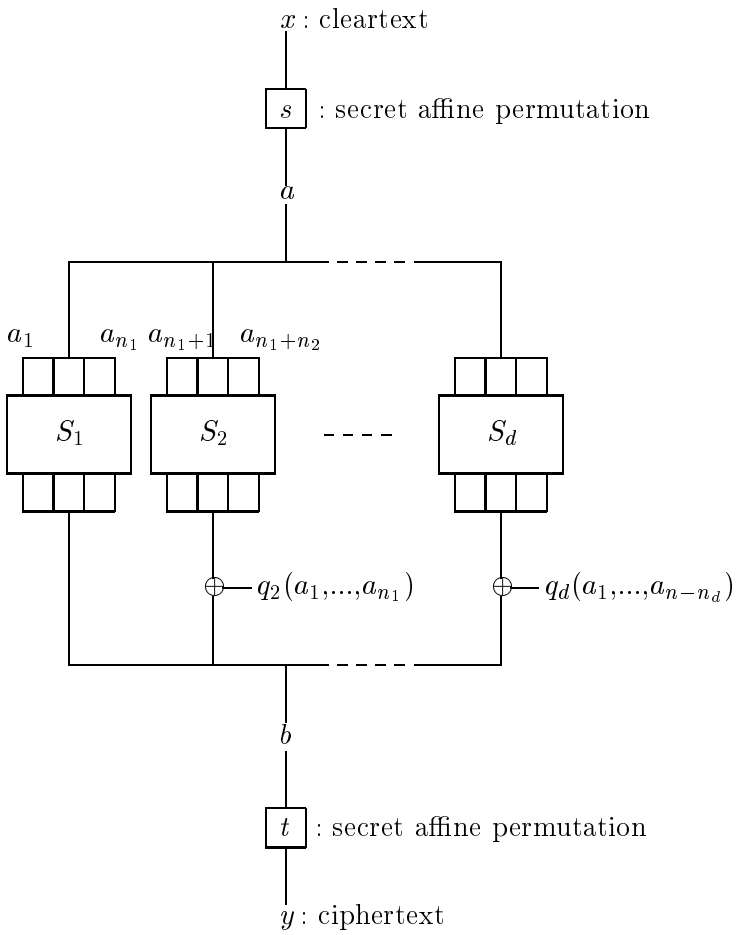


FIG. 1 – *One round of triangular S-boxes (it gives a weak scheme)*

this new cryptanalysis is based on the fact that the n_i's are small, and not on algebraic properties of the tranformations, such as in the general cryptanalysis of [13].

Variation 1 (General S-boxes scheme): The output of the S_i box ($2 \leq i \leq d$) can also be XORed with a quadratic function q_i of the variables $(a_1,a_2,...,a_{n_1+...+n_{i-1}})$ before performing the t permutation, as shown in figure 1 below. The public key is still a quadratic function of the x_j variables, $1 \leq j \leq n$, and the decryption will be done first on S_1 (it gives a_1, ..., a_{n-1}), then on S_2, ..., and finally on S_d (*i.e.* from the left to the right S-boxes). As mentioned in note 3, this variation may have practical advantages in signature. However, the cryptanalysis of this generalisation will be exactly the same as if $q_2 = ... = q_d = 0$.

Variation 2 (Triangular system except one S-box at the beginning): In figure 2, we have a "triangular system except one S-box at the beginning". This variation is "almost" a

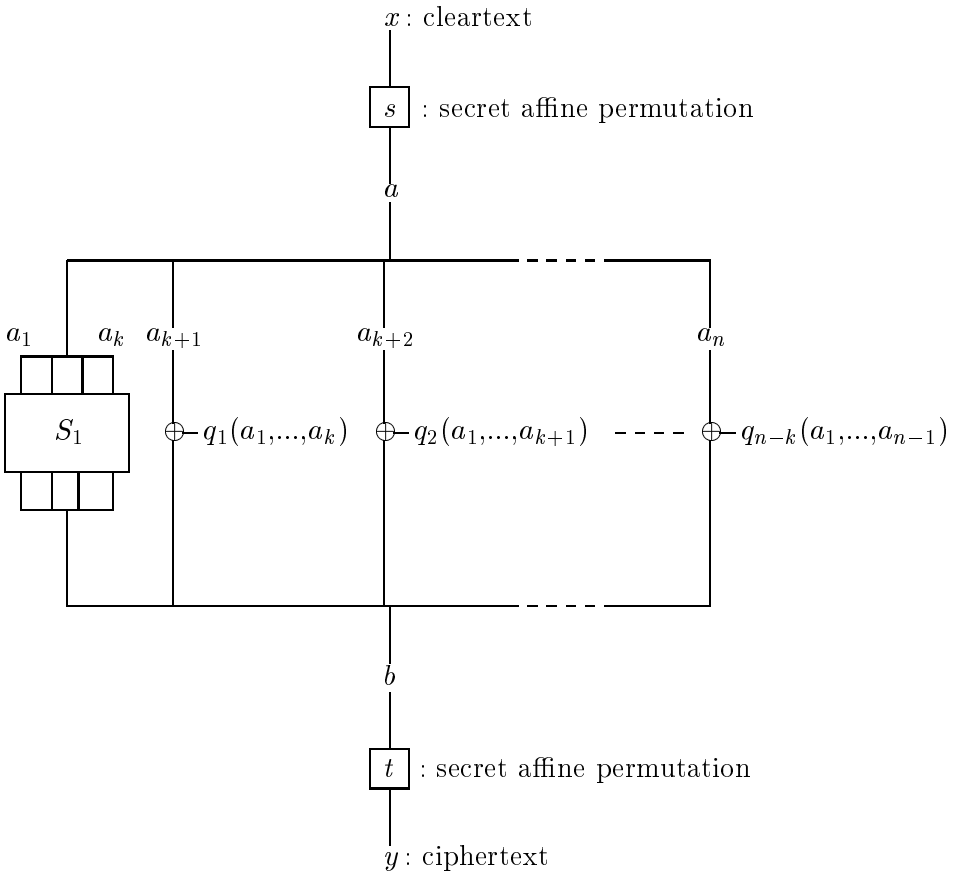


FIG. 2 – *One round of "one S-box followed by a triangular construction" (it gives a weak scheme)*

permutation, so it may have some practical advantages. However, this variation can be broken as the general one round scheme.

3 One round of S-boxes: Cryptanalysis of the schemes

3.1 The Degeneration Problem (DP)

As we will see below, the cryptanalysis of the one-round schemes is related to the following problem, that we call the "Degeneration Problem" (DP):

Given: A multivariate quadratic equation $y = P(x_1,...,x_n)$, where P is of degree two, with n variables x_1, ..., x_n.

Problem: Find a change of variables of degree one, from the n variables x_i to (at most) $n-1$ variables x'_1, ..., x'_{n-1} (*i.e.* $\forall i$, $1 \leq i \leq n$, $x_i = P'_i(x'_1,...,x'_{n-1})$, where P'_i is of total degree one), and find a polynomial Q of total degree two in (at most) $n-1$ variables, such that: $y = Q(x'_1,...,x'_{n-1})$.

As we will see, there are some algorithms with polynomial complexity for this DP problem, and from them a cryptanalysis of the one-round scheme can be found.

Remark: However, it can be noticed that we did not find an algorithm with polynomial complexity for the following more general problem, that we call the "Linear Combination Degeneration Problem (LCDP)":

Given: A set of k multivariate quadratic equations $y_i = P_i(x_1,...,x_n)$, where P_i is of degree two.

Problem: Find a linear combination $y = \sum_{i=1}^{k} \alpha_i y_i$ such that, for this linear combination y, the DP problem has a solution.

3.2 First attack: canonical cryptanalysis

In section 3.2, we present an attack based on the existence of a canonical representation of the quadratic polynomials involved in the scheme. Let us recall the classical theorem about the representation of quadratic forms over a field with even characteristic.

Definitions:

1. A *quadratic form* over K is a homogeneous polynomial in $K[X_1,...,X_n]$ of degree two, or the zero polynomial.
2. Two polynomials f and g of degree ≤ 2 over K, are said to be *equivalent* if f can be transformed into g by means of a nonsingular linear substitution of the indeterminates.
3. A quadratic form f in n indeterminates is called *nondegenerate* if it is not equivalent to a quadratic form in fewer than n indeterminates.

Theorem 1 *Let $f \in K[X_1,...,X_n]$ be a nondegenerate quadratic form over $K = GF(q)$, where q is even. If n is odd, then f is equivalent to:*

$$x_1x_2 + x_3x_4 + ... + x_{n-2}x_{n-1} + x_n^2. \tag{1}$$

If n is even, then f is either equivalent to:

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n \tag{2}$$

or to a quadratic form of the type:

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n + x_{n-1}^2 + ax_n^2 \tag{3}$$

where $a \in K$.

A proof of this theorem is given in [11]. Moreover, if f is given, then the transformation of f in (1), (2) or (3) is very easy.

The algorithm used in the proof can also be applied to a degenerate quadratic form, in the same way. For such a form f, we can define the "number of independent variables of f" as the

smallest integer k such that f is equivalent to a quadratic form in k indeterminates. Following the algorithm given in the proof of the theorem, we can see that this number k is also very easy to compute.

Using these tools, we can now derive a method to "separate" the S-boxes in the scheme. We use the notations of section 2.

Description of the algorithm:

1. Generate randomly n bits $\alpha_1,...,\alpha_n$, and compute:

$$y' = \sum_{i=1}^{n} \alpha_i P_i(X_1,...,X_n),$$

   which is a polynomial of degree two in $X_1,...,X_n$.
2. Compute the number k of independent variables of y', and write y' as a polynomial of degree two in $x'_1,...,x'_k$, where the x'_i have an affine expression in the x_j $(1 \leq j \leq n)$.

The probability that y' has no components in branch number one, *i.e.* the probability that y' has a linear expression in $b_{n_1+1},...,b_{n_1+...+n_d}$ is $\frac{1}{q^{n_1}}$.

So by repeating steps 1 and 2 about λq^{n_1} times, we will generate about λ equations $y'_1,...,y'_\lambda$ such that these equations have no components in branch number one. Moreover, when this occurs, then $k \leq n - n_1$, and when this does not occur, then the probability that $k \leq n - n_1$ is very small, except if there are different branches with about n_1 variables. However, it is very easy to see if two polynomials y'_1 and y'_2 with $k \leq n - n_1$ have no components from the same branch: we write y'_1 in $x'_1,...,x'_{k_1}$ and y'_2 in $x''_1,...,x''_{k_2}$, and we see how many terms are linearly independent in $x'_1,...,x'_{k_1},x''_1,...,x''_{k_2}$. If there are more than $n - n_1$ independent terms, then y'_1 or y'_2 has components from the first branch. And if there are less than $n - n_1$ independent terms, then we can write y'_1 and y'_2 as polynomials of degree two in $x'''_1,...,x'''_{n-n_1}$, where the x'''_i have affine expressions in the x_j $(1 \leq j \leq n)$.

Note: Another idea would be to compute the number k of independent variables of $y'_1 + y'_2$ and test if $k \leq n - n_1$.

Finally, after $O(q^{n_1})$ computations, we will find $n - n_1$ independent equations $y'_1,...,y'_{n-n_1}$ such that these equations have no components coming from the first S-box, and we will be able to write all these equations as polynomials of degree two in $x'_1,...,x'_{n-n_1}$, where the x'_i have affine expressions in the x_j $(1 \leq j \leq n)$.

So with these new variables x'_i and y'_i, we have eliminated the variables of the first branch. Applying this algorithm d times, we can "separate" the S-boxes from each other and then the scheme is very easy to break by simple exhaustive search on the cleartext for each S-box.

3.3 Second attack: differential cryptanalysis

We will now describe another very different attack, that also works very well against schemes with small branches. Let $F = (P_1,...,P_n)$ the function described in section 2. F is seen as a function from K^n to K^n. Let ψ be an element of K^n. Let $X_1,...,X_k$ be k random plaintexts ($k = n$ typically), and $Y_1,...,Y_k$ be the corresponding ciphertexts, *i.e.* $Y_i = F(X_i)$ for $i = 1,...,k$. Let Y'_i be the ciphertext of the plaintext $X_i + \psi$, *i.e.* $Y'_i = F(X_i + \psi)$ for $i = 1,...,k$. Finally, let $\psi'_i = Y'_i - Y_i$ (where the addition $+$ and the subtraction $-$ are of course done component by component in K).

Definition of test (A): We will say that "ψ satisfied the test (A)" if the vector space generated by all the values ψ'_i is not K^n itself by an affine subspace (the dimension of this subspace will be $n - \lambda$, where λ is one of the n_i values).

It is very easy to see if a value ψ satisfies this test (A) or not, since k is very small ($k = n$ typically). Moreover, we have this theorem:

Theorem 2 *The probability that ψ satisfies the test (A), when ψ is randomly chosen, is $\geq \frac{1}{q^{\max(n_i)}}$.*

Proof: The probability that the value $A_1 = (a_1,...,a_{n_1+1})$ is the same for X_i and $X_i + \psi$ is $\frac{1}{q^{n_1}}$ (because we have q^{n_1} possible values for A_1). We write this relation as $A_1(X_i) = A_1(X_i + \psi)$.

Moreover, since s is an affine function, if the A_1 value is the same for one pair $(X,X + \psi)$ for a specific value X, then for all values X', we will have $A_1(X') = A_1(X' + \psi)$. So, with a probability $\frac{1}{q^{n_1}}$, we have:

$$\forall X_i \in K^n,\ B_1(X_i) = B_1(X_i + \psi).$$

Now, since the transformation t is affine, we see that all the values $\psi'_i = Y'_i - Y_i = F(X_i + \psi) - F(X_i)$ belong to an affine subspace of dimension $n - n_1$. We can now describe our new attack. We proceed in five steps.

1. We find about n values ψ that satisfy the test (A), by randomly trying some ψ values.
2. We say that two such values ψ_1 and ψ_2 "belong to the same branch" if $\psi_1 + \psi_2$ (and more generally $\lambda_1\psi_1 + \lambda_2\psi_2$, $\lambda_1 \in K$, $\lambda_2 \in K$) also satisfies the test (A). We detect and group together the values ψ_i that belong to the same branches. In this way, we separate the ψ values in d different groups, that we call "branches". We will find about n_1 independent values ψ that come from the first branch, n_2 that come from the second branch, ..., n_d that come from the branch number d.
3. By an affine change of variables, we change the variables x_i into x'_i such that, with these new variables, the subspace generated by the values ψ of the first (resp. second,..., d^{th}) branch is characterized by: $x'_1 = ... = x'_{n_1} = 0$ (resp. $x'_{n_1+1} = ... = x'_{n_1+n_2} = 0$,..., $x'_{n_1+...+n_{d-1}+1} = ... = x'_{n_1+...+n_d} = 0$).
4. Similarly, by an affine change of variables, we change the variables y_i into y'_i such that, with these new variables, the subspace generated by the values ψ' of the first (resp. second,..., d^{th}) branch is characterized by: $y'_1 = ... = y'_{n_1} = 0$ (resp. $y'_{n_1+1} = ... = y'_{n_1+n_2} = 0$,..., $y'_{n_1+...+n_{d-1}+1} = ... = y'_{n_1+...+n_d} = 0$).
5. At this point, all the branches have been detected and isolated. To complete the attack, we can now perform an exhaustive search on each input of each of the d branches.

Note: In the second attack, we did not use the fact that the S-boxes are polynomial functions of degree two in a basis. So the attack will work in the same way whatever the functions in the S-boxes may be.

3.4 Third attack: gradient cryptanalysis

We now introduce a new tool to attack the scheme with S-boxes. The general principle of the algorithm is exactly the same as in the first attack that we called "canonical cryptanalysis". It is based on the fact that, with a rather high probability, a linear combination

$Q = \sum_{i=1}^{n} \alpha_i P_i(x_1,...,x_n)$ of the public polynomials has a "number of independent variables" less than n.

In the case of quadratic polynomials, we saw that this "lack" of variables can be detected because a canonical form exists, which enables us to read the "real" number of variables of such a polynomial.

But as soon as the degree is greater or equal than three, we don't have such a canonical representation.

The new idea is to use the so-called *gradient* of Q. For $x = (x_1,...,x_n)$, it is defined by:

$$\textbf{grad}\ Q(x) = \Big(\frac{\partial Q}{\partial x_1}(x_1,...,x_n),...,\frac{\partial Q}{\partial x_n}(x_1,...,x_n)\Big).$$

The following theorem shows the link between the gradients of Q and the number of independent variables of Q:

Theorem 3 *Let Q be a quadratic form in n variables over a field K, then:*

1) If the characteristic of K is > 2, then the number k of independent variables of Q is equal to the dimension of the subspace A generated by all the **grad** $Q(x)$ *($x \in K^n$).*

2) If the characteristic of K is 2, then it is easy to find a non singular linear substitution of the indeterminates that transforms Q into a quadratic form:

$$R(x'_1,...,x'_n) = \Phi(x'_1,...,x'_{\dim A}) + \sum_{i=\dim A+1}^{n} \lambda_i x'^2_i,$$

where Φ is a quadratic form over K. Moreover the number k of independent variables of Q is:

$$k = \dim A + \begin{cases} 0 & \text{if } \forall i,\ \lambda_i = 0. \\ 1 & \text{if } \exists i,\ \lambda_i \neq 0. \end{cases}$$

The rest of the attack is similar to that described in the canonical cryptanalysis. The great difference here is that we can derive the same kind of theorem for cubic forms over a field K, and more generally for forms of any degree over K.

3.5 Fourth attack: linearity in some directions

This attack is probably the simplest of all our attacks.

As above, the attack is based on the fact that, with a rather high probability, a linear combination $Q = \sum_{i=1}^{n} \alpha_i P_i(x_1,...,x_n)$ of the public polynomials has a "number of independent variables" less than n.

For such a Q, there are some values $d = (d_1,...,d_n)$, $d \neq 0$, such that:

$$\forall x \in K^n,\ Q(x+d) = Q(x). \qquad (\#)$$

From $(\#)$ we will show how to find these values d.

If $x_i x_j$ is a term of $Q(x)$, then in $(\#)$ this $x_i x_j$ gives the term: $(x_i + d_i)(x_j + d_j) - x_i x_j$, *i.e.* it gives: $x_i d_j + x_j d_i + d_i d_j$.

Therefore, by writing in $(\#)$ that all the terms in x_k, $1 \leq k \leq n$, are zero, we will have n equations of degree one in n variables (there variables are the d_λ variables). So by gaussian reductions, we will find the set of the solutions d, as wanted.

As a result, we have found a break-up into the spaces generated by the S-boxes. By iterating this, we will find the break-up into each of these spaces, and the scheme is very easy to break by simple exhaustive search on the cleartext from each S-box.

3.6 Linear approximations of the S-boxes

A natural idea is of course to also try linear cryptanalysis, by linear approximations of the S-boxes. However, we did not see how to use this idea for an efficient cryptanalysis, even when the S-boxes are public. The problem comes from the fact that s and t are very general affine bijections (much more general than the P transformation.of DES for example), and that they are secret.

3.7 Conclusion

In conclusion, one round of small S-boxes does not give secure algorithms. So, for security, we must have either a transformation that manipulates large values (such a transformation is thus not exhaustively storable in a S-box), or more than one round. We will specifically study the case of small S-boxes with two rounds later in this paper.

4 How to separate large branches in the C^* scheme

As we mentioned in section 2 (see note 5), each of the attacks described in section 3 gives a way to break the C^* scheme of Matsumoto and Imai, when the n_i parameters (which measure the size of the branches) are small. However, this cryptanalysis is deeply based on the fact that the branches are small, and does not give any result in the case of large branches.

Nevertheless, we have found another attack to separate large branches in the Matsumoto-Imai scheme, even if they are large. This attack is described in the extended version of [13] (available from the author).

Part II: Design of Two Round Schemes

5 Complexity of functional decomposition

As we saw in section 3, the scheme which uses only one round of S-boxes is insecure, and we can use several methods to "separate" the branches from each other.

Then a natural question arises: is it possible to design a more secure cryptosystem by using *two* rounds of transformations, each of which is given by polynomials of degree two? This leads to the following problem:

Decomposition problem: Let g and h be two functions which map K^n into K^n and which are given by polynomials of total degree two in n variables over K. Then $f = g \circ h$ is also a function from K^n to K^n, and it is given by n polynomials of degree *four* in n variables over K. Suppose that f is given. Is it computationally feasible to recover g and h?

A positive answer to that problem would imply that the two rounds can be easily separated from each other. So, to break the scheme, we would only have to break two independent schemes given by quadratic polynomials in n variables.

That would of course make the idea of using two rounds uninteresting. However, the decomposition of multivariate polynomials was studied by Matthew Dickerson, who gave an algorithm for the following problem:

Multivariate left decomposition: Given polynomials f and $h_1,...,h_n$ in $K[X_1,...,X_n]$, and an integer r, decide if there exists a polynomial $g(x_1,...,x_n)$ of total degree at most r that composes with the h_i's to give f. That is, does there exist a polynomial $g(x_1,...,x_n)$ such that

$$f(x_1,...,x_n) = g(h_1(x_1,...,x_n),...,h_n(x_1,...,x_n))$$

and $\deg(g) \leq r$? If so, determine the coefficients of g.

In [5], Dickerson presents the best-known algorithm for this problem, which is polynomial in the degree of $f,h_1,...,h_n$, but *exponential* in the number n of variables (note that our decomposition problem is even harder, because the h_i's are not known).

He also shows that the *general* problem of decomposition of multivariate polynomials is difficult, because the following one is NP-hard:

s-1-decomposition problem: Given a monic univariate polynomial $f(x)$ and an integer s, decide if there exists an s-1-decomposition of f, *i.e.* a monic univariate polynomial h of degree s, and a bivariate polynomial $g(y,x) \in K[Y,X]$ of the form $g(y,x) = \prod_{i=1}^{r}(y+\alpha_i x+\beta_i)$ with $\alpha_i,\beta_i \in \hat{K}$, an algebraic extension of K, such that $f(x) = g(h(x),x)$. If so, determine the coefficients of g and h.

There are some reasons to think that the following problem is also NP-hard (cf [5], problem 14, p. 74):

Multivariate decomposition of given degree: Given a polynomial f in $K[X_1,...,X_n]$ and some subset of the following: integers $k,r,s_1,...,s_k$, a polynomial $g(x_1,...,x_k) \in K[X_1,...,X_k]$, and polynomials $h_1(x_1,..,x_n),..., h_k(x_1,...,x_n)$, decide if there exists a functional decomposition $g,h_1,...,h_k$ of f (*i.e.* $f = g(h_1,...,h_k)$) such that $\deg g = r$, and $\deg h_i = s_i$ for $1 \leq i \leq n$. If so, compute those coefficients of g and the h_i's which were not given.

Dickerson (see [5], p. 75) notices that: "The s-1-decomposition problem seems intuitively easier than problem 14. In problem 14, f, g and h are general multivariate polynomials of arbitrary dimension. Furthermore polynomial g takes the polynomial h_i as arguments, and we know nothing about the form of g other than its degree. In the s-1-decomposition problem, on the other hand, f and h are both univariate polynomials and g is only bivariate. Furthermore, g takes x and not another polynomial as its second argument. We also know a great deal about the structure of the polynomial g, namely that it factors as: $g(y,x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$. However, we have tried without success to reduce the s-1-decomposition problem to problem 14."

So, it is still an open problem...

That is the reason why we propose to base a cryptosystem on two rounds of quadratic transformations.

Remark 1: The decomposition problem of this paper consists in finding the decomposition of a multivariate polynomial f (of total degree 4 in n variables x_1, ..., x_n) into two multivariate polynomials g and h, of total degree 2, such that $f = g \circ h$. In a basis, g (and h) are written with about $n \times \frac{n^2}{2}$ coefficients. When we write this equation $f = g \circ h$ in a basis, we obtain about $n \times \frac{n^4}{4!}$ equations in the $2 \times n \times \frac{n^2}{2}$ unknown coefficients. So we have $\mathcal{O}(n^5)$ equations of degree total two in the $\mathcal{O}(n^3)$ unknowns. There is maybe no polynomial time algorithm for this problem (unlike if we had $\mathcal{O}(n^6)$ equations), but since the number of quadratic equations is much larger than the number of unknowns, one can be dubitious about the chances to rely this

problem to an NP-complete problem. Moreover, this also shows that, most of the time, there exists "essentially" one decomposition of f (by "essentially", we do not take into account all the obvious solutions than can be deduced from one solution, *i.e.* all the $(g \circ \varphi) \circ (\varphi^{-1} \circ h)$, where φ is an affine permutation of the variables.

Remark 2: In 1999, in the paper [4], an algorithm to find the decomposition of two multivariate polynomials of degree two has been published. However, this algorithm needs all the description of the composition. So, despite the results of this paper [4], we will still be able to design schemes by giving only part of the composition, or by "perturbating" the composition as we will see below.

6 Two rounds: A general description of the schemes ("2R" and "2R$^-$" schemes)

The general scheme will be the following one:

As in section 2, a finite field K, with $q = p^m$ elements, is public, and we want to transform a cleartext $x = (x_1,...,x_n) \in K^n$ into a ciphertext $y = (y_1,...,y_n) \in K^n$.

The secret items will be:

1. Three affine bijections r, s and t from K^n to K^n.
2. An application $\phi : K^n \to K^n$ given by n quadratic equations over K.
3. An application $\psi : K^n \to K^n$ given by n quadratic equations over K.

If we have the secrets, then we can obtain y from x as follows:

1. $x = (x_1,...,x_n)$ is first transformed with the affine secret permutation r into $r(x) = a = (a_1,...,a_n)$.
2. Then we compute $b = \phi(a)$.
3. $b = (b_1,...,b_n)$ is then transformed with the affine secret permutation s into $s(b) = c = (c_1,...,c_n)$.
4. Then we compute $d = \psi(c)$.
5. Finally, $d = (d_1,...,d_n)$ is transformed with another permutation t, into $t(d) = y = (y_1,...,y_n)$.

The public items are:

1. The field K and the length n of the messages.
2. Some of the n polynomials $P_1,...,P_n$ of degree four in n variables over K, such that $y_i = P_i(x_1,...,x_n)$ $(1 \leq i \leq n)$. When all these polynomials are given, we call the scheme a "2R scheme". When only some of these polynomials are given, we call the scheme a "2R$^-$ scheme".

So anyone can encipher a message.

Moreover, if the secret items are known, we must be able to decipher a message. For that purpose, we need to invert the functions ϕ and ψ. In practice, ϕ and ψ can be chosen among the following classes of functions:

1. "C^*-functions": monomials over an extension of degree n over K, such as $a \mapsto a^{1+q^\theta}$ (they are the basic transformations in the C^* scheme of Matsumoto and Imai, cf [12]).
2. "Triangular-functions": $(a_1,...,a_n) \mapsto (a_1, a_2 + q_1(a_1), a_3 + q_2(a_1,a_2),..., a_n + q_{n-1}(a_1,...,a_{n-1}))$, where each q_i is a quadratic polynomial in i variables over K (in [7] Fell and Diffie used a particular case of these transformations).

Note: If $|K|$ is even, then $x \mapsto x^2$ is a bijection on K. So in this case, we can define as "triangular-functions" all the functions $(a_1,...,a_n) \mapsto (a_1^{\alpha_1},a_2^{\alpha_2} + q_1(a_1),...,a_n^{\alpha_n} + q_{n-1}(a_1,...,a_{n-1}))$, where each q_i is a quadratic polynomial, and where each α_i is either 1 or 2.

3. "S-boxes-functions": they have already been defined in section 2, in the case of a one-round scheme.
4. "One S-box followed by a triangular construction": $(a_1,...,a_n) \mapsto (b_1,...,b_n)$, with $(b_1,...,b_\lambda) = S(a_1,...,a_\lambda)$ and $b_i = a_i + q_{i-1}(a_1,...,a_{i-1})$ for $\lambda+1 \leq i \leq n$. Here, S, q_λ, ..., q_n are quadratic functions, and λ can be seen as the size of the S-box.
5. "Triangular with S-boxes functions", or "General S-boxes functions": $(a_1,...,a_n) \mapsto (b_1,...,b_n)$, with:
$$\begin{cases} (b_1,...,b_{n_1}) = S_1(a_1,...,a_{n_1}) \\ (b_{n_1+1},...,b_{n_1+n_2}) = S_2(a_{n_1+1},...,a_{n_1+n_2}) + (q_1(a_1,...,a_{n_1}),...,q_{n_2}(a_1,...,a_{n_1})) \\ \quad ..... \\ (b_{n_1+...+n_{d-1}+1},...,b_{n_1+...+n_d}) = S_d(a_{n_1+...n_{d-1}+1},...,a_{n_1+...+n_d}) \\ \qquad + (q_{n_2+...+n_{d-1}+1}(a_1,...,a_{n_1+...+n_{d-1}}),...,q_{n_2+...+n_d}(a_1,...,a_{n_1+...+n_{d-1}})). \end{cases}$$
where the functions S_i and q_i are all quadratic over K.
6. "D^* functions": the definitions and analysis of these D^* functions are the main thema of [16].

Note: It is easy to see that classes number 2, 3 and 4 are just particular cases of class number 5.

The first two have the advantage of being bijections (D^* functions will also be bijective). But, as we will see in sections 7 and 8, all the schemes where ψ is one of the functions of classes 1 and 2, are insecure, whatever we may choose for ϕ. On the contrary, if the second round function ψ is chosen in classes 3, 4, 5 or 6, and the first round function ϕ in classes 1, 2, 3, 4, 5 or 6, the corresponding schemes seem to resist cryptanalysis, so far. We call these schemes the "2R" schemes ("2R" stands fro "two rounds").

7 Cryptanalysis of a second round with C^*

In this section, we study the scheme described in section 6, when ψ is a C^*-function.

A typical example of this situation is the case where ϕ and ψ are both C^*-functions. As was pointed out by Jacques Patarin (cf [15]), this scheme is insecure. Moreover, the cryptanalysis of [15] can be applied even if ϕ is not a C^*-function. Whatever the function ϕ of the first round may be, the basic idea of the attack is to compute all the equations of the form:
$$\sum_{i,j,k} \mu_{ijk} y_i x_j x_k + \sum_{i,j} \nu_{ij} x_i x_j + \sum_{i,j} \xi_{ij} y_i x_j + \sum_i \omega_i x_i + \sum_i \theta_i y_i + \pi_0 = 0$$
and then to introduce some "transition" variables p_i such that these equations can be written:
$$\sum_{i,j} \gamma_{ij} p_i y_j + \sum_i \alpha_i p_i + \sum_i \beta_i y_i + \delta_0 = 0.$$
When the y_i variables are given, then from these equations we can compute the p_j variables by Gaussian reductions. Finally, we can deduce x from the p_j by using:

1. The cryptanalysis of the C^* scheme (cf [13]) if ϕ is a "C^*-function" (this is exactly the situation of [15]).

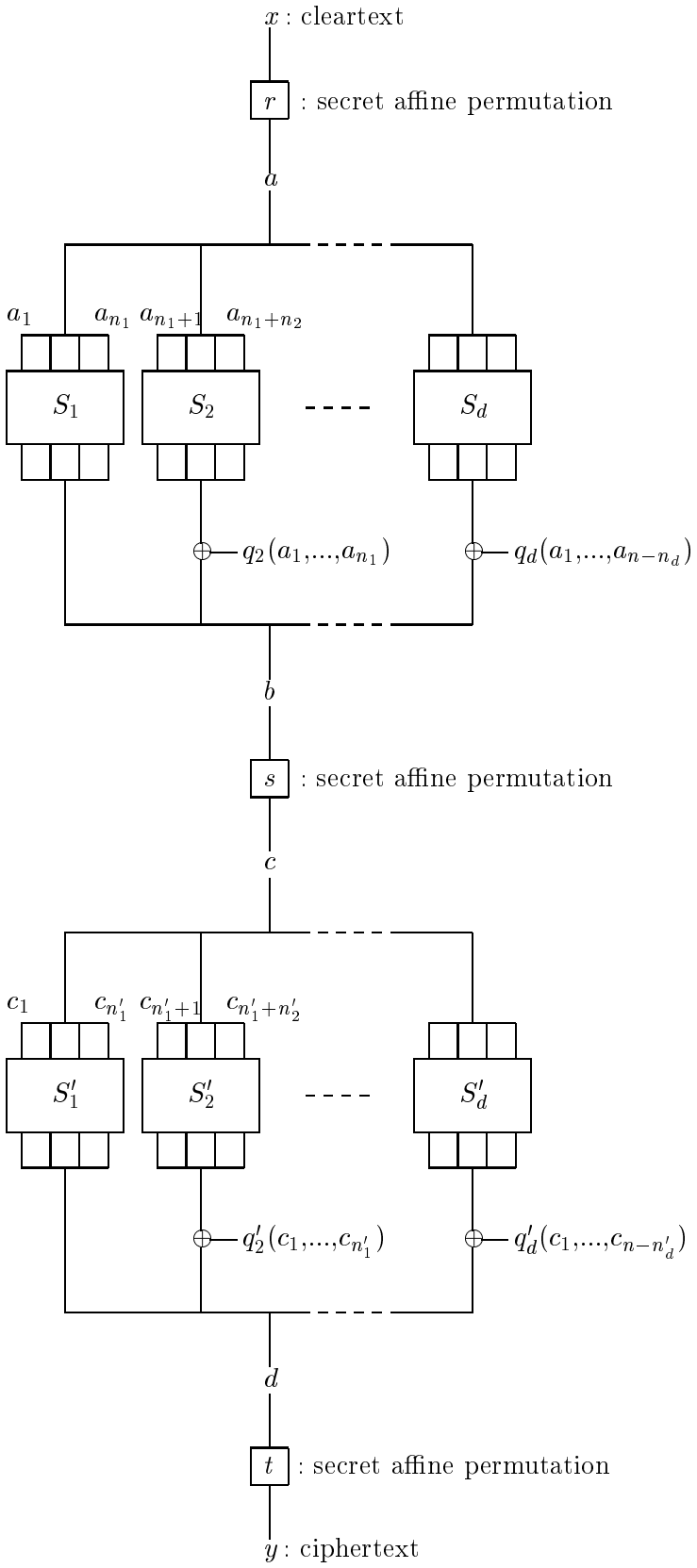


FIG. 3 – *Two rounds of triangular S-boxes (such a scheme may be secure...)*

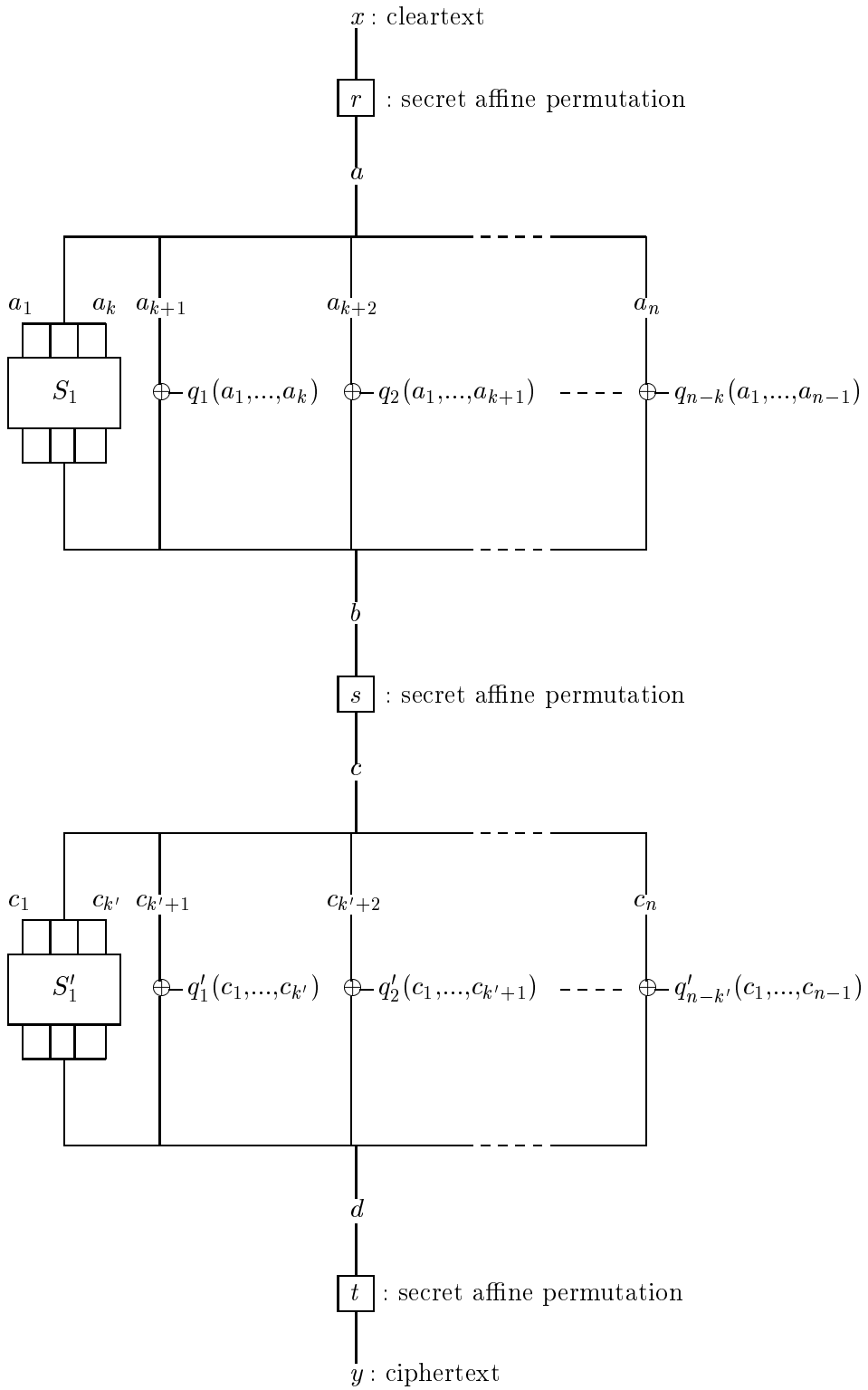


FIG. 4 – *Two rounds of "One S-box followed by a triangular construction" (such a scheme may be secure...)*

2. A cryptanalysis similar to the one described in section 8, if ϕ is a "triangular-function".
3. The cryptanalysis of one round of S-boxes (see section 3), if ϕ is a "S-boxes-function".

8 Cryptanalysis of a second round with a triangular-function

Here, we consider the scheme of section 6, when ψ is a "triangular-function". More precisely, we have:

$$(d_1,...,d_n) = \psi(c_1,...,c_n) = (c_1, c_2 + q_1(c_1),..., c_n + q_{n-1}(c_1,...,c_{n-1})),$$

where $q_1,...,q_{n-1}$ are quadratic polynomials.

Once more, that scheme is insecure. The key idea is to use the fact that the equation $d_1 = c_1$ gives an equation of degree only 2 in the x_i variables, and that, for each i, $1 \leq i \leq n$, c_i can be obtained from d_i and $c_1,...,c_{i-1}$. An attack can be derived as follows:

1. By Gaussian reductions on the α_i, β_{ij}, γ_i and δ variables below, find the equations of the form:
$$\sum_i \alpha_i y_i = \sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta.$$
The vector space of the solutions has dimension one (because the solutions come from $d_1 = c_1$), so that we obtain $d_1 = \sum_i \alpha_i y_i$ and $c_1 = \sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta$.
2. By Gaussian reductions on the α'_i, β'_{ij}, γ'_i and δ' and on the coefficients of q_1, find the equations of the form:
$$\sum_i \alpha'_i y_i = \sum_i \sum_j \beta'_{ij} x_i x_j + \sum_i \gamma'_i x_i + \delta' + q_1\Big(\sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta\Big).$$
The vector space of the solutions has again dimension one (because the solutions come from $d_2 = c_2 + q_1(c_1)$). We thus obtain $d_2 = \sum_i \alpha'_i y_i$ and $c_2 = \sum_i \sum_j \beta'_{ij} x_i x_j + \sum_i \gamma'_i x_i + \delta'$. We also obtain q_1.

By repeating this argument n times, we have finally found $c_1,...,c_n$ as quadratic polynomials in $x_1,...,x_n$. What remains is to cryptanalyse the first round, *i.e.* the function $c = s \circ \phi \circ r(x)$, and that is feasible for each of the possible choices for ϕ (see section 3).

9 Second round with S-boxes: description of the schemes

In this section, we present our candidate algorithms. They are all based on the general scheme described in section 6. As we said in section 6, four possibilities exist for the second round function ψ. We choose ψ as a "S-boxes-function", or as "one S-box followed by a triangular construction" or as "S-boxes combined with a triangular construction", or as a "D^* function" (D^* functions are treated in [16] and not in this paper). The algorithms are also differentiated from each other by the choice of the first round function ϕ.

As we said before, six possibilities exist for ϕ:

1. ϕ is a "C^*-function".
2. ϕ is a "triangular-function".
3. ϕ is a "S-boxes-function".
4. ϕ is "one S-box followed by a triangular construction".
5. ϕ is a "Triangular with S-boxes function".

6. ϕ is a "D^* function" (D^* functions are treated in [16] and not in this paper).

Figure 3 illustrates the example with two rounds of "S-boxes combined with a trangular construction". Similarly, figure 4 illustrates the example with two rounds of "one S-box followed by a triangular construction".

In the design of these schemes, the main idea is to hide the quadratic function ϕ (which is weak when used alone, as we saw in section 3) with the second round ψ. For instance, in the case of "C^*-S-box", the algebraic structure of the C^*-functions (which is not completely hidden in the scheme of Matsumoto and Imai, which led to the cryptanalysis of their scheme) seems difficult to recover if it has been mixed with S-boxes, which are supposed to have no special structure. This point will be detailed in section 10.3.

Moreover, as we pointed out, the problem of decomposition of multivariate polynomials seems to be difficult, so probably no general attack exists to "separate" the two rounds from each other.

In conclusion, the security of our schemes is an open problem...

Note: When the first round is a "C^*-function" and the second round is a "S-boxes-function", the scheme may be secure, but it becomes insecure if the two quadratic functions are put the other way round, as we saw in section 7. This shows that the analysis of the security closely depends on the order of the quadratic functions used in the schemes.

10 Remarks about the security of these 2R ("two round") schemes

10.1 The "Quadratic Degeneration Problem" (QDP)

As we have seen in section 3, the security of the one round schemes is related to the DP problem. Similarly, the security of the two round schemes seems to be related to the following problem, that we call the "Quadratic Degeneration Problem" (QDP).

Given: A multivariate equation of total degree four, $y = P(x_1,...,x_n)$, where P is of degree four, with n variables x_1, ..., x_n.

Problem: Find (at most) $n-1$ polynomials of total degree two, P'_1, ..., P'_{n-1}, and a polynomial Q of total degree two, such that:

$$\begin{cases} \forall i,\ 1 \leq i \leq n-1,\ x'_i = P'_i(x_1,...,x_n) \\ y = Q(x'_1,...,x'_n) \end{cases}$$

We do not know if an algorithm with polynomial complexity exists for this problem (when a solution exists). So we do not know how to cryptanalize the general two-round schemes.

Remark: In the problem above, we have to find $n-k$ polynomials P'_1, ..., P'_{n-k}, with $k = 1$. When $k = 0$, this is the decomposition problem (of a multivariate polynomial of total degree 4). If the problem is easy for $k = 1$, or for $k = 2$, or for $k = 3$ for example, then it would give a cryptanalysis of two-round schemes, and this problem may indeed be easier for $k = 1$, or $k = 3$, than for $k = 0$.

10.2 Effect of the gradient cryptanalysis

The method we described in section 3.4, in order to cryptanalyse one round of S-boxes, can be extended to the case of two rounds. But, fortunately or unfortunately, it does not seem to lead to a practical attack against our schemes. Let us describe the idea.

We use the notations of sections 6 and 10.1. Let us consider $h = s \circ \phi \circ r$, which is given by n quadratic polynomials h_1,...,h_n in n variables over K. Let $f = \sum_i \alpha_i P_i$, where the α_i's are randomly chosen in K, and let $g = \sum_i \alpha_i (t \circ \psi)_i$, so that $f = g \circ h$.

We suppose for simplicity that the sizes of the S-boxes of ψ are $n_1 = ... = n_8 = 8$. Then, with a probability $\frac{1}{2^8} = \frac{1}{256}$, g has no components from the first S-box. An easy calculation then gives:

$$\textbf{grad } f(x) = \sum_{r=1}^{n} \frac{\partial g}{\partial c_r}(h(x)) \textbf{ grad } h_r(x) = \sum_{r=9}^{n} \frac{\partial g}{\partial c_r}(h(x)) \textbf{ grad } h_r(x).$$

Therefore, **grad** $f(x)$ lies in the subspace generated by (**grad** $h_9(x)$,...,**grad** $h_n(x)$), and that is true for any $x = (x_1,...,x_n)$ in K^n. So we may deduce the following equalities:

$$\begin{cases} det(\textbf{grad } f(x), \textbf{grad } h_2(x), ..., \textbf{grad } h_n(x)) = 0 \\ det(\textbf{grad } h_1(x), \textbf{grad } f(x), \textbf{grad } h_3(x), ..., \textbf{grad } h_n(x)) = 0 \\ ... \\ det(\textbf{grad } h_1(x), ..., \textbf{grad } h_7(x), \textbf{grad } f(x), \textbf{grad } h_9(x), ..., \textbf{grad } h_n(x)) = 0 \end{cases}$$

Let $(-1)^{i_0+j_0}\Delta_{i_0 j_0}(x)$ be the value of the determinant of the matrix $\left(\frac{\partial h_i}{\partial x_j}(x)\right)_{\substack{1 \le i \ne i_0 \le n \\ 1 \le j \ne j_0 \le n}}$. Then the equations become the eight following:

$$\sum_{j=1}^{n} \Delta_{ij}(x) \frac{\partial f}{\partial x_j}(x) = 0 \qquad (1 \le i \le 8),$$

where $\Delta_{ij}(x)$ is unknown and is polynomial of total degree $n-1$ in x. We may imagine using a lot of different values of x to obtain relations between the coefficients of the polynomials $\Delta_{ij}(x)$, but this task seems to be impractical, due to the very high degree of these polynomials.

Note 1: In the case of *one* round of S-boxes, the attack works, because the corresponding Δ_{ij} are constant polynomials. Thus we obtain about $8n$ equations about the $8n$ unknown Δ_{ij}, by trying about n different values of x. The Δ_{ij} can then be found by Gaussian reductions.

Note 2: Once again, the scheme seems to resist this attack because the *inverse* of the quadratic functions that we use, has a high total degree in x.

10.3 Effect of the affine multiple attack

Another attack, which is very general, was described in [14]. It can be used against schemes based on a univariate polynomial transformation hidden by secret affine bijective transformations.

This attack is based on the following fact: if f is a univariate polynomial over a finite field K, then by using a general algorithm (see for example [3]), we can compute an "affine multiple" of the polynomial $f(x) - y$, *i.e.* a polynomial $A(x,y) \in K[X,Y]$ such that:

1. Each solution of $f(x) = y$ is also a solution of $A(x,y) = 0$.
2. $A(x,y)$ is an affine function of x.

We will now see that, because of the affine multiple attack, we cannot choose $n_1 = ... = n_d = 1$ in our schemes, *i.e.* each S-box must have at least two elements of K as input and output.

- Suppose first that K has $q = 2^m$ elements and that $n_1 = ... = n_d = 1$ (for example $m = 8$, $n = 8$ and $d = 8$). Each S-box S_e is given by a univariate quadratic polynomial over $K = GF(2^m)$.
  If $m = 1$, then the cryptanalysis is obvious: the quadratic polynomial is in fact an affine function (because $x^2 = x$ if $GF(2)$), and thus $t \circ \psi \circ s$ is itself a secret affine function, so that the scheme can be broken as a one-round scheme, and so can be easily broken.
  If $m > 1$, then the same attack can also work as follows. The public equations are given over $K = GF(2^m)$, but the cryptanalyst can rewrite them over $GF(2)$, so that the previous attack applies, with mn public equations of degree 2 over $GF(2)$, instead of n public equations of degree 4 over $GF(2^m)$.
- Suppose now that K has $q = p^m$ elements, where p is a small prime, $p \neq 2$. Since the characteristic is not 2, the S-boxes cannot be seen as affine functions any more. However, we can use here the affine multiple principle to attack the scheme.
  Let $f_e(x) = \alpha_e x^2 + \beta_e x + \gamma_e$ be the univariate quadratic polynomial stored in S_e ($1 \leq e \leq d$), and let $A_e(x,y)$ be an affine multiple of $f_e(x) - y$. If all the exponents in y are $\leq k$, then there will be an attack with a Gaussian reductions on $O(n^{1+k})$ terms. Moreover, since p is small, k is also small, so that this attack is efficient.

  Example: We take $p = 3$. An easy calculation gives $A_e(x,y) = \alpha_e^2 x^3 - (\alpha_e y + \beta_e^2 - \alpha_e \gamma_e)x + \beta_e(y - \gamma_e)$. Then, by Gaussian reductions, we compute all the equations of the form:
  $$\sum_{i,j,k} \mu_{ijk} y_i x_j x_k + \sum_{i,j} \nu_{ij} x_i x_j + \sum_{i,j} \xi_{ij} y_i x_j + \sum_i \omega_i x_i + \sum_i \theta_i y_i + \pi_0 = 0$$
  and we can then proceed as in section 7.

In conclusion, we do not recommend using S-boxes given by a univariate polynomial of degree 2 over K.

Note If we have for example $K = GF(2)$, $n = 64$ and $n_1 = ... = n_8 = 8$, we can imagine a similar attack, if we consider that each S-box S_e can be given as a univariate polynomial over $GF(2^8)$. But it is impractical, because this polynomial f_e is generally of very high degree (about 256), and most of the time, the degree of $A_e(x,y)$ in y becomes very high too. Therefore, the Gaussian reductions is not feasible any more.

11 How to choose the parameters and smartcard implementations

Let $K = GF(2^m)$ (K is not necessarily of characteristic 2, but for simplicity we will assume that it is so; moreover the computations are a little easier in characteristic 2). Let n be, as usual, the length of the cleartext x or of the ciphertext y (*i.e.* $x \in K^n$ and $y \in K^n$). Let F be one of our candidate algorithms (with public polynomials of degree four).

We recommend choosing m and n such that:

$$\begin{cases} mn \geq 128 & \text{(C1)} \\ n \geq 12 & \text{(C2)} \end{cases}$$

We also recommend to not publish in the public key all the equations of the composition (*i.e.* to have a "2R$^-$" scheme). (C3)

Condition (C1): To avoid exhaustive search on the cleartext x, we need $mn \geq 64$. Moreover, Eli Biham found an attack, based on the "birthday paradox" that shows that we need in fact $mn \geq 128$. This attack is described in Appendix 1.

Condition (C2): When n is very small ($n < 8$ typically), then to solve a set of n polynomial equations of small total degree d ($d \leq 4$ for example) with n variables in a finite field K is feasible with *ad hoc* techniques (for example with GCD of polynomials, Gröbner bases, or by exhaustive search on some of the variables). Moreover, this is often easy even when K is large (because here n is very small). So, in order to avoid these attacks, we must have $n \geq 8$. Moreover, for polynomial equations of total degree $d = 2$, we recommend for security to have $n \geq 16$, even if it is not yet clear if these attacks are efficient when $8 < n < 16$. Similarly, for polynomial equations of total degree $d = 4$, we recommend for security to have $n \geq 12$, even if it is not clear if these attacks are efficient when $8 < n < 12$. (This gives the condition (C2)).

Condition (C3): Condition (C3) is here to avoid a nice idea from [4] that may create an efficient decomposition algorithm. This idea is described in Appendix 2.

Note: Instead of (C3), another way to avoid the results of paper [4] is to introduce a "perturbation" in the originally public equations, as it was suggested in [17] for the C^* scheme. These "perturbations" can consist in introducing extra variables (it will give a 2RV scheme), fixing some variables (it will give a 2RF scheme), mixing the equations with truly random ones (it will five a 2R$^+$ scheme), etc. Moreover, all these "perturbations" can be combined (it gives a 2R^{-+}VF scheme). See [17] for more details. (In [17], these ideas are used and studied in the case of a C^* scheme).

Speed: Our schemes are very fast in secret key computations (more than 100 times faster than a 512 bits RSA for example). However, our schemes may be slower in public key computations compared with a 512 bits RSA with a small public exponent e.

Length of the public key:

- If $m = 1$ (*i.e.* $K = GF(2)$), then the length of the public key is huge: it is 162 Mbytes with $n = 128$.
- If $m = 4$ (*i.e.* $K = GF(16)$), then the length of the public key is more realistic: it is 920 Kbytes if $n = 32$.
- Moreover, if r, s and t are linear (not only affine), and if the S-boxes are homogeneous, then the public polynomials will be homogeneous. If $m = 4$, the length of the public key is then 818 Kbytes if $n = 32$.
- If $K = \mathbf{F}_{257}$ or $K = \mathbf{F}_{256}$ and $n = 16$, then the length of the public key is only $\simeq 20$ Kbytes.
- It might also be possible to choose r, s, t and the S-boxes as polynomials with values in a subfield K' of K. For example, if $K = GF(16)$ and $K' = GF(2)$, then the public key is divided by 4: its length is now 205 Kbytes if $n = 32$. Moreover, if $K = \mathbf{F}_{256}$, $K' = GF(2)$ and $n = 16$, then the length of the public key is only $\simeq 2.5$ Kbytes. It is not yet clear if this decreases the security of the scheme or not.

So, if $m \neq 1$, the schemes can have a reasonable length for the public key, despite the degree four of the public polynomials.

Smartcard implementations, secret key computations: The secret computations are very easy and very fast in a smartcard. The RAM needed when $mn \leq 128$ is about 32 bytes. The ROM needed for the program which computes F is also very moderate. The secret functions r, s and t can be stored in EEPROM or be computed from a secret seed of 64 bits stored in EEPROM. If the S-boxes have very small inputs, they can be stored in EEPROM, or even in ROM if all the cards have the same S-boxes. For example, if a S-box takes 8 bits in input and gives 8 bits in output, it will be stored in 256 bytes. We may also have the S-boxes all the same, or such that each S-box S_i can easily be computed from S_1 and/or a small secret seed (if the S-boxes have larger inputs, for example 16 bits in input and 16 bits in output, then the S-boxes might be recomputed and the inversion might require some small polynomial resolutions). As we can see, the parameters can be chosen in order to have very efficient secret key computations in smartcards.

Smartcard implementations, public key computations: A smartcard can compute F at least as easily as F^{-1} when F is its own function, *i.e.* when it uses its secret values to compute F and F^{-1}. The public key is then not needed to compute F and F^{-1}. In some applications, the smartcard doesn't have to make public key computations (which are then done in a PC for example), but only secret key computations. In this case, our schemes are very efficient. However, in some applications, the public key is required (for example we may ask for a stored and signed public key). As we have seen above, in some implementations, we may have a public key of 7.5 Kbytes (in 1996, some smartcards can store 8 Kbytes of EEPROM, or more), but for most of the schemes, the public key is larger than that, and it will then not be possible to store it in the smartcard. In this case, and if the public key must be given, we can imagine that the signed public key is stored in another device than the secure chip of the card, for example in an optic storage on the card... Or, if we have a lot of time to compute the public key, the card will compute and give some specific cleartext/ciphertext pairs (it will choose these pairs of course), and by Gaussian reductionss, the public key will be computed outside (however, this might take a long time !). So, as we can see, the public key can be stored only in very few cases, and most of the schemes are very efficient in smartcards, when only secret key computations are required in the smartcard.

12 Concrete examples of S-boxes

Attacks on Matsumoto-Imai like cryptosystems often rely on the existence of general relations between the inputs x_i and the outputs y_j, that we can classify as follows:

Type 1 relations: $\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

Type 2 relations: $\sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

Type 3 relations: $\sum \xi_{ijk} x_i y_j y_k + \sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

In order to avoid generalisations of these attacks against our two round schemes, we will select S-boxes such that no type 1, 2 or 3 relations exist (except $0 = 0$ or obvious relations such that $y_i = y_i^2$ in $\mathbf{F}_2$).

Nicolas Courtois did for us some simulations in order to see if there are some small S-boxes without any type 1, 2, 3 relations. The results of these simulations are given in Table 1, Table 2 and Table 3.

Note: In table 1, table 2 and table 3, λ is by definition the number such that the S-boxes take λ elements of the field $\mathbf{F}_q$ in input and give also λ elements of $\mathbf{F}_q$ in output.

λ	2	3	4	5	6	7	8	9	10	11
$\mathbf{F}_2$	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	7 9 91	0 0 112	0 0 114	0 0 78	0 0 0	
$\mathbf{F}_4$	$\neq 0$	$\neq 0$	$\neq 0$	0 0 84	0 0 0					
$\mathbf{F}_8$	$\neq 0$	0 6 105	0 0 11	0 0 0						
$\mathbf{F}_{16}$	$\neq 0$	0 8 50	0 0 0							
$\mathbf{F}_{32}$	26 60 250	0 10 40	0 0 0							
$\mathbf{F}_{64}$	$\neq 0$	0 12 48								
$\mathbf{F}_{128}$	$\neq 0$	0 14 56								

Table 1: some results obtained in characteristic 2

Legend:

type 1 type 2 type 3	: means that we found at least one S-box with those numbers of independent equations.
$\neq 0$	: means that we did not find a S-box with 0 equations of type 1.
	: we did no simulations but we expect no equations of type 1, 2, 3 for most of the S-boxes

Examples:

- For $\lambda = 10$ and $\mathbf{F}_2$, we found a S-box with 10 bits in input and 10 bits in output, where no type 1, 2 or 3 equations exist.
- For $\lambda = 4$ and $\mathbf{F}_{16}$, we found a S-box with 4 elements of $\mathbf{F}_{16}$ in input and 4 elements of $\mathbf{F}_{16}$ in output, where no type 1, 2 or 3 equations exist.

λ	2	3	4	5
$\mathbf{F}_3$	2 4 10	0 1 14	0 0 9	0 0 0
$\mathbf{F}_9$	2 4 14	0 0 1	0 0 0	
$\mathbf{F}_{27}$	3 3 21	0 0 0		

Table 2: some results in characteristic 3

λ	1	2	3
$\mathbf{F}_5$	0 1 1	0 0 0	0 0 0
$\mathbf{F}_7$	0 0 0		
$\mathbf{F}_{17}$	0 0 0		
$\mathbf{F}_{251}$	0 0 0		

Table 3: some results in characteristic ≥ 5

Conclusion: Even when λ is very small, we can find some S-boxes with no type 1, 2, 3 equations. This is nice, since large values of λ mean more complex S-boxes, and could imply larger public keys. The example given in section 11, with $n = 16$ and $K = \mathbf{F}_{16}$ (with a public key of 30 Kbytes) can be obtained with 4 S-boxes, each of which has 4 elements of $\mathbf{F}_{16}$ in input and output, and where no type 1, 2, 3 relations exist.

13 Is it possible to have bijective S-boxes?

A natural question is the following: is it possible to use *bijective* S-boxes in our schemes?

A first idea is using constructive methods to build bijective S-boxes. Unfortunately, such methods have always given the possibility of an attack of the scheme, so far (for example with C^* or triangular-functions).

If the probability of being bijective were not too small, we could randomly generate functions with a small degree, and thus find some of them which are bijective and have no specific structure (for example, this second idea works for functions of degree 1). When the degree is ≥ 2, obtaining an accurate evaluation of the probability of being bijective is difficult. But, if we suppose that this probability is about the same as for randoms functions, it may be shown (see below) that building bijective S-boxes with this method gives a huge public key for our schemes.

A new method for building efficient and strong bijective S-boxes with small degree may be discovered some day, but at the present, all known methods give not efficient or weak bijective S-boxes.

Rough evaluation of the number of "strong" bijective S-boxes

The problem : We would like to know if some "strong" bijective S-boxes exist, *i.e.* some functions f from $GF(2^n)$ to $GF(2^n)$ $(2 \leq n \leq 32)$, such that :

1. f is a bijection.
2. When we consider $GF(2^n)$ as a vector space of dimension n over $GF(2)$, then in a basis, f is given by n polynomials of degree $\leq d$ (for us, $d = 2$ or $d = 3$).
3. f is "strong", *i.e.* the "construction of f" does not give any obvious weakness for the security of our schemes (we will give more details about this point below).

For example, with a triangular construction, or with a Matsumoto-Imai C^* construction, we can very easily design a function f that satisfies 1 and 2. However, the construction of f gives a way to attack the schemes. This comes from the fact that, if $f(x) = y$, where $x = (x_1,...,x_n)$ and $y = (y_1,...,y_n)$, and $y_i = P_i(x_1,...,x_n)$ $(1 \leq i \leq n)$, then in these constructions, there are some polynomial equations $Q_i(x_1,...,x_n,y_1,...,y_n) = 0$ of small total degree, and of degree one in the x_i variables. More generally, we do not want either to have equations $Q_i(x_1,...,x_n,y_1,...,y_n) = 0$ of small total degree, except sums of products of the public equations by polynomials of small total degree (*i.e.* except the polynomials of the ideal of $K[x_1,...,x_n,y_1,...,y_n]$ generated by the public equations).

When this occurs, we will say that the "construction of f" gives a weakness, and if this does not occur, we will say that f is a "strong" bijection. So, do strong bijections exist (for $d = 2$ or $d = 3$ for example)?

First evaluation : Let α_n be the probability for a function from $GF(2^n)$ to $GF(2^n)$ to be a bijection. Let H_d be the number of functions from $GF(2^n)$ to itself, that are given by polynomial equations of degree $\leq d$ in a basis over $GF(2)$. Then, in a first evaluation, we may think that the number of "strong" bijections from $GF(2^n)$ to $GF(2^n)$ of degree $\leq d$ is about $\alpha_n H_d$. This comes from the fact that, in a first evaluation, we may suppose that most of the bijections are "strong" and that the density of strong bijections in H_d is perhaps about the same as the density of strong bijections in the set of *all* functions from $GF(2^n)$ to $GF(2^n)$.

An easy calculation gives :

$$\alpha_n = \frac{(2^n)!}{2^{n.2^n}}, \quad H_2 = 2^{n(1+n+\frac{n(n-1)}{2})}, \quad H_3 = 2^{n(1+n+\frac{n(n-1)}{2}+\frac{n(n-1)(n-2)}{6})}.$$

Let $N = 2^n$. The Stirling formula $N! \sim N^N e^{-N}\sqrt{2\pi N}$ gives :

$$\alpha_n = \frac{N!}{N^N} \simeq e^{-N}\sqrt{2\pi N}.$$

Example 1 : Let $n = 10$. Then $N = 1024$ and $\alpha_{10} \simeq \frac{1}{2^{1471}}$, $H_2 = 2^{560}$, $H_3 = 2^{1760}$. Since $\alpha_{10} H_2$ is much smaller than 1, in first approximation, we expect to have no "strong" bijections of degree two over $GF(2^{10})$. Moreover, since $\alpha_{10} H_3 \simeq 2^{289}$, in first approximation, we expect to have about 2^{289} "strong" bijections over $GF(2^{10})$, of degree three (in a basis).

Example 2 : Let $n = 6$. Then $N = 64$ and $\alpha_6 \simeq \frac{1}{2^{88}}$, $H_2 = 2^{132}$. Then, in this first evaluation, we may expect to have about $\frac{2^{132}}{2^{88}} = 2^{44}$ "strong" bijections over $GF(2^6)$, of degree two in a basis. However, we will see now that this is probably not true.

Second evaluation : If f is a "strong" bijection and if g and h are two affine bijective functions, then $f' = g \circ f \circ h$ is also a "strong" bijection. Moreover, there are exactly

$$q^{n(n+1)}\Big[\Big(1-\frac{1}{q}\Big)\Big(1-\frac{1}{q^2}\Big)...\Big(1-\frac{1}{q^n}\Big)\Big]$$

bijections from $GF(q^n)$ to $GF(q^n)$ that are affine over $GF(q)$. So, since we have exactly

$$C = 2^{2n(n+1)}\Big[\Big(1-\frac{1}{2}\Big)...\Big(1-\frac{1}{2^n}\Big)\Big]^2$$

possibilities for (g,h), we see that if the number B of "strong" bijections (obtained in the first evaluation) is much smaller than C, then is does not mean that there are "about B" strong bijections, but it means, in this second evaluation, that we expect to have *no* such bijection.

Examples :

- In our example 1 above, $C \simeq 2^{217}$ and since $289 > 217$, we expect indeed to have about 2^{289} "strong" bijections of degree three, as claimed. These bijections of degree three are expected to come from about 2^{72} bijections f_i of degree three, and to be of the form $g \circ f_i \circ h$, where g and h are two affine bijections.
- However, in our example 2 above, we have $C \simeq 2^{81}$, and since $44 < 81$, we expect to have no strong bijections of degree two with $n = 6$ (if we had one, then there would exist at least about 2^{81} such bijections).

Results : With our "second evaluation", the results are :

- No "strong" bijective functions of degree $d = 2$ are expected to exist.
- "Strong" bijective functions of degree $d = 3$ are expected to exist if and only if $n \leq 10$.
- "Strong" bijective functions of degree $d = 4$ are expected to exist if and only if $n \leq 13$.
- "Strong" bijective functions of degree $d = 5$ are expected to exist if and only if $n \leq 16$.

So, if we want to design a bijective scheme with composition of a quadratic function h and a bijective function g made with bijective S-boxes, g will have to be of degree $d \geq 3$, and therefore $F = g \circ h$ will be of degree ≥ 6. However, a function F of degree 6 from $GF(2^n)$ to $GF(2^n)$ will have a public key of 635 Mbytes if $n = 64$. This is too large for all practical applications (at least at the present !). So, if this second evaluation is valid, we can conclude that we will not have bijective S-boxes in our two-round schemes.

Note : More precisely, we can conclude that this "second evaluation" shows that there are probably no strong bijective S-boxes of small degree for "random reasons". However, there may be some for "structural reasons", *i.e.* the hypothesis that the strong permutations are almost randomly distributed in the subset of multivariate functions of total degree d, may be false. For example, the probability for a linear function to be a permutation is much higher than the probability for a random function to be a permutation. So, some strong bijective S-boxes may exist despite our evaluations. In this appendix, we have just studied two "first evaluations" based on a random distribution hypothesis, but this hypothesis may be wrong, and some constructions for strong permutations of small degree may still exist.

14 Comparison with symmetrical cryptosystems

The cryptosystems we study in this paper have a lot of similarities with classical symmetric cryptosystems (such as DES for instance), since they use for example S-boxes (*i.e.* local transformations of a small number of values), followed by linear transformations, and there are several rounds (two for our schemes). However, DES for instance (cf [2]), or Khufu (cf [9]) are not secure when only very few rounds are used. So, why do our schemes (with only two rounds) resist classical attacks? This can be explained by the following arguments:

1. In each round that uses S-boxes, *all* the input bits are transformed, and not only half of them, as happens in a Feistel scheme, such as the one used in DES.
2. The affine transformations are *secret*, and that is not the case for the P transformation of DES, which is public.
3. Moreover, the affine transformations are very general, *i.e.* every output bit is a linear combination of *all* the input bits, and not only of one input bit, such as in the P transformation of DES.
4. In our schemes, the S-boxes can be secret or public. In DES, they are public, and in Khufu they are secret.

Note 1: R. Rivest recently proposed XDES, which is a composition of DES, an initial simple affine secret transformation, and a final one (more precisely, these transformations consist in XORing the input with a secret value). Maybe this change does not strengthen DES against differential of linear cryptanalysis, but it seems to prevent other attacks (in particular, against exhaustive search on the key, cf [10]). So, XDES may illustrate the idea that composing an encryption algorithm with initial and final affine secret transformations, may lead to a significant strengthening of this algorithm.

Note 2: In our schemes, it is very important to have only two rounds, because the composition of three rounds, each round being quadratic, would lead to polynomials of degree eight, so that the length of the public key would be much too large for practical applications.

15 Conclusion

In this paper, we have studied new asymmetric algorithms which all rely on the idea of using one or two rounds of very simple quadratic transformations, which are hidden by secret affine tranformations. By "very simple" quadratic transformations, we mean quadratic transformations given by a triangular set of equations, or given by quadratic and small S-boxes. When there is only one round like this, or when the second round is built with functions whose algebraic structure is poorly hidden, we have proven that the corresponding schemes are insecure. From these ideas, we were able to design some new cryptanalysis of the Matsumoto and Imai scheme C^*.

However, when the parameters and the quadratic polynomials involved in each round are carefully chosen, we still have candidate algorithms that seem to resist the attacks. In this paper, the main characteristic of these schemes is that, in the second round, they use S-boxes, which are given by multivariate polynomials of small degree, and are randomly chosen in order to avoid any simple algebraic structure.

Since the publication of these algorithms at ICICS'97, two cryptanalysis papers ([1] and [4]) have been written on these algorithms. Due to [1], it is necessary that the input of the schemes has at least 128 bits. Due to [4], it is recommended to not publish all the originally public equations.

When all this is done, are these algorithms secure? If they are, it would be a surprising and easy way of designing asymmetric cryptosystems. If they are not, it would strengthen the idea that the messages cannot be split in small branches, but must be transformed in a global way, and therefore that we need algebra to build secure asymmetric cryptosystems.

So the question remains open...

References

[1] Eli Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S-Boxes*, not yet published.

[2] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the full 16-Round DES*, CRYPTO'92, Springer-Verlag, pp. 487-496.

[3] Ian Blake, XuHong Gao, Ronald Mullin, Scott Vanstone, Tomik Yaghoobian, *Applications of finite Fields*, Kluwer Academic Publishers, p. 25.

[4] Y. Ding-Feng, L. Kwok-Yan, D. Zong-Duo, *Cryptanalysis of the "2R" schemes*, Proceeding of CRYPTO'99, Springer, pp. 315-325.

[5] Matthew Dickerson, *The functional Decomposition of Polynomials*, Ph.D Thesis, TR 89-1023, Department of Computer Science, Cornell University, Ithaca, NY, July 1989.

[6] Matthew Dickerson, *The Inverse of an Automorphism in polynomial Time*, IEEE 30^{th} annual symposium on Foundations of Computer Science (FOCS), 1989, pp. 82-87.

[7] Harriet Fell and Whitfield Diffie, *Analysis of a public Key Approach based on polynomial Substitutions*, CRYPTO'85, Springer-Verlag, pp. 340-349.

[8] Michael Garey, David Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman, p. 251.

[9] Henri Gilbert, Pascal Chauvaud, *A chosen Plaintext Attack of the 16-Round Khufu Cryptosystem*, CRYPTO'94, Springer-Verlag, pp. 359-368.

[10] Joe Kilian, Phillip Rogaway, *How to protect DES against eshaustive Key Search*, CRYPTO'96, Springer-Verlag, pp. 252-267.

[11] Rudolf Lidl, Harald Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its applications, volume 20, Cambridge University Press.

[12] Tsutomu Matsumoto, Hideki Imai, *Public quadratic polynomial-Tuples for efficient Signature-Verification and Message-Encryption*, EUCROCRYPT'88, Springer-Verlag, pp. 419-453.

[13] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, CRYPTO'95, Springer-Verlag, pp. 248-261.

[14] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*, EUROCRYPT'96, Springer-Verlag, pp. 33-48.

[15] Jacques Patarin, *Asymmetric Cryptography with a Hidden Monomial*, CRYPTO'96, Springer-Verlag, pp. 45-60.

[16] Jacques Patarin, Louis Goubin, *Trapdoor one-way permutations and multivariate polynomials*, ICICS'97, Springer, November 1997, pp. 356-368.

[17] Jacques Patarin, Louis Goubin, Nicolas Courtois, *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, Advances in Cryptology, Proceedings of ASIACRYPT'98, Springer, pp. 45-60.

[18] Bruce Schneier, *Applied cryptography*, second edition, John Wiley and Sons, chapter 20, p. 501.

[19] Joachim von zur Gathen, *Functional Decomposition of Polynomials: the tame Case*, J. Symbolic Computation (1990), vol. 9, pp. 281-299.

[20] Joachim von zur Gathen, *Functional Decomposition of Polynomials: the wild Case*, J. Symbolic Computation (1990), vol. 10, pp. 437-452.

Appendix 1

An attack of Eli Biham based on the birthday paradox (cf [1])

Introduction: Eli Biham found an attack of the scheme based on two rounds of S-boxes with a complexity about $\mathcal{O}(\sqrt{q^n})$ (instead of $\mathcal{O}(q^n)$) for exhaustive search of one cleartext). Moreover, after his attack, it is possible to decipher very fast any ciphertext (not only one). At ICICS'97, we gave examples with $q^n \geq 2^{128}$, so that Eli Biham's attack is not very efficient against our published schemes, but however this attack is very interesting because it illustrates (one more time) the efficiency in cryptanalysis of the "birthday paradox" and it shows that for schemes based on two rounds of S-boxes we really need $q^n \geq 2^{128}$. (In the HFE schemes of [14], different ways of having a scheme with $q^n \simeq 2^{64}$ are explained. These ways of having $q^n \simeq 2^{64}$ should not be generalized in the case of two rounds of S-boxes.)

The idea: The idea is to use the fact that two inputs that differ only for one S-box will give the same output (*i.e.* a collision) if they have the same output for this S-box.

The starting point of the attack is to search for a collision $f(a) = f(b)$, and then to randomly choose values c_i's and search for (about 100) collisions $f(c_i \oplus a) = f(c_i \oplus b)$. From these collisions, we will have a way to detect the first secret linear transformation and the S-boxes of the first round.

Appendix 2

An attack (from [4]) for the decomposition problem

In [4], an algorithm is suggested for computing the decomposition of two quadratic multivariate polynomials. The efficiency of this algorithm relies on two hypotheses. No simulations have been done so far on this algorithm, so it is not easy to evaluate the probablity of the algorithm to succeed (*i.e.* when the two hypotheses are valid). However, this algorithm looks sufficiently dangerous to recommend to not publish <u>all</u> the equations of a composing $h = f \circ g$ in the public key of the 2R schemes described in this paper.

Notations: Let $h = f \circ g$, when f and g are two quadratic functions from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$. In a basis, g is given as $(g_1,...,g_n)$, where g_i, $1 \leq i \leq n$ is a function from $(\mathbf{F}_q)^n$ to $\mathbf{F}_q$.

Aim of the algorithm: The aim of the algorithm is to find the vector space G generated by g_1, ..., g_n, *i.e.* G =Vect(g_1,...,g_n). From G, it is then easy to find a decomposition of h. Since $h = f \circ g$, h is also equal to $(f \circ A^{-1}) \circ (A \circ g)$, where A is any linear and bijective function from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.

Remark: G is a vector space of dimension about n and G is included in the vector space of dimension about $\frac{n^2}{2}$ of all the quadratic polynomials from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.

How to compute G: Let $V = \Big\{ \sum_{i=1}^{n} x_i G \Big\}$. (Remark: V is a vector space of dimension about n^2 and V is included in the vector space of dimension about $\frac{n^3}{6}$ of all the cubic polynomials from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.)

To compute G, the algorithm uses two hypotheses:

Hypothesis 1:

$$V = \mathrm{Vect}\Big(\frac{\partial h_i}{\partial x_j}\Big).$$

When this hypothesis 1 is true, then we can compute V, since h is given.

Hypothesis 2:

$$G = \{\text{polynomials } r \text{ of degree 2 such that: } \forall i,\ 1 \leq i \leq n,\ x_i \cdot r \in V\}.$$

When this hypothesis 2 is true, then we can compute G since this hypothesis 2 will give relations of degree one on the coefficients of r.

Remark: To avoid problems such that $x_i^4 = x_i$ or $x_i^3 = x_i$ or $x_i^2 = x_i$, the authors of [4] make the hypothesis that $q \geq 5$.

Unbalanced Oil and Vinegar Signature Schemes

EUROCRYPT'99 *(Version complète)*
Article avec Aviad Kipnis (NDS Israël) et Jacques Patarin (Bull SC&T)

Abstract

In [16], J. Patarin designed a new scheme, called "Oil and Vinegar", for computing asymmetric signatures. It is very simple, can be computed very fast (both in secret and public key) and requires very little RAM in smartcard implementations. The idea consists in hiding quadratic equations in n unknowns called "oil" and $v = n$ unknowns called "vinegar" over a finite field K, with linear secret functions. This original scheme was broken in [9] by A. Kipnis and A. Shamir. In this paper, we study some very simple variations of the original scheme where $v > n$ (instead of $v = n$). These schemes are called "Unbalanced Oil and Vinegar" (UOV), since we have more "vinegar" unknowns than "oil" unknowns. We show that, when $v \simeq n$, the attack of [9] can be extended, but when $v \geq 2n$ for example, the security of the scheme is still an open problem. Moreover, when $v \simeq \frac{n^2}{2}$, the security of the scheme is exactly equivalent (if we accept a very natural but not proved property) to the problem of solving a random set of n quadratic equations in $\frac{n^2}{2}$ unknowns (with no trapdoor). However, we show that (in characteristic 2) when $v \geq n^2$, finding a solution is generally easy. In this paper, we also present some practical values of the parameters, for which no attacks are known. We also study schemes with public keys of degree three instead of two. We show that no significant advantages exist at the present to recommend schemes of degree three instead of two. However, we show that it is very easy to combine the Oil and Vinegar idea and the HFE schemes of [14]. The resulting scheme, called HFEV, looks at the present also very interesting both from a practical and theoretical point of view. In UOV, the number of vinegar variables must be $> n$, but in HFEV this number can be very small or very large. The length of a UOV signature can be as short as 192 bits and the length of a HFEV signature can be as short as 80 bits.

1 Introduction

Since 1985, various authors (see [6], [8], [12], [14], [16], [17], [18], [21] for example) have suggested some public key schemes where the public key is given as a set of multivariate quadratic (or higher degree) equations over a small finite field K.

The general problem of solving such a set of equations is NP-hard (cf [7]) (even in the quadratic case). Moreover, when the number of unknowns is, say, $n \geq 16$, the best known algorithms are often not significantly better than exhaustive search (when n is very small, Gröbner bases algorithms might be efficient, cf [5]).

The schemes are often very efficient in terms of speed or RAM required in a smartcard implementation. (However, the length of the public key is generally ≥ 1 Kbyte. Nevertheless it is sometimes useful to notice that secret key computations can be performed without the public key). The most serious problem is that, in order to introduce a trapdoor (to allow the computation of signatures or to allow the decryption of messages when a secret is known), the generated set of public equations generally becomes a small subset of all the possible equations and, in many cases, the algorithms have been broken. For example [6] was broken by their authors, and [12], [16], [21] were broken. However, many schemes are still not broken (for example [14], [17], [18], [20]), and also in many cases, some very simple variations have been suggested in order to repair the schemes. Therefore, at the present, we do not know whether this idea of designing public key algorithms with multivariate polynomials over small finite fields is a very powerful idea (where only some too simple schemes are insecure) or not.

In this paper, we will present two new schemes: UOV and HFEV. UOV is a very simple scheme: the original Oil and Vinegar signature scheme (of [16]) was broken (see [9]), but if we have significantly more "vinegar" unknowns than "oil" unknowns (a definition of the "oil" and "vinegar" unknowns can be found in section 2), then the attack of [9] does not work and the security of this more general scheme (called UOV) is still an open problem.

Moreover, we show that, when we have approximately $\frac{n^2}{2}$ vinegar unknowns for n oil unknowns, the security of UOV is exactly equivalent (if we accept a natural but not proved property) to the problem of solving a random set of n quadratic equations in $\frac{n^2}{2}$ unknowns (with no trapdoor). This result suggests that some partial proof of security (related to some simple to describe and supposed very difficult to solve problems) might be found for some schemes with multivariate polynomials over a finite field. However, we show that most of the systems of n quadratic equations in n^2 (or more) variables can be solved in polynomial complexity... As a result, at the present, we rather recommend $v \simeq 3n$ for example than $v \simeq \frac{n^2}{2}$ for security in UOV. We also study Oil and Vinegar schemes of degree three (instead of two). HFEV combines the ideas of HFE (of [14]) and of vinegar variables. HFEV looks more efficient than the original HFE scheme.

2 The (Original and Unbalanced) Oil and Vinegar of degree two

Let $K = \mathbf{F}_q$ be a small finite field (for example $K = \mathbf{F}_2$). Let n and v be two integers. The message to be signed (or its hash) is represented as an element of K^n, denoted by $y = (y_1,...,y_n)$. Typically, $q^n \simeq 2^{128}$. The signature x is represented as an element of K^{n+v} denoted by $x = (x_1,...,x_{n+v})$.

Secret key

The secret key is made of two parts:

1. A bijective and affine function $s : K^{n+v} \to K^{n+v}$. By "affine", we mean that each component of the output can be written as a polynomial of degree one in the $n+v$ input unknowns, and with coefficients in K.
2. A set $(\mathcal{S})$ of n equations of the following type:

$$\forall i,\ 1 \leq i \leq n,\ y_i = \sum \gamma_{ijk} a_j a'_k + \sum \lambda_{ijk} a'_j a'_k + \sum \xi_{ij} a_j + \sum \xi'_{ij} a'_j + \delta_i \qquad (\mathcal{S}).$$

The coefficients γ_{ijk}, λ_{ijk}, ξ_{ij}, ξ'_{ij} and δ_i are the secret coefficients of these n equations. The values a_1, ..., a_n (the "oil" unknowns) and a'_1, ..., a'_v (the "vinegar" unknowns) lie in K. Note that these equations $(\mathcal{S})$ contain no terms in $a_i a_j$.

Public key

Let A be the element of K^{n+v} defined by $A = (a_1,...,a_n,a'_1,...,a'_v)$. A is transformed into $x = s^{-1}(A)$, where s is the secret, bijective and affine function from K^{n+v} to K^{n+v}.

Each value y_i, $1 \leq i \leq n$, can be written as a polynomial P_i of total degree two in the x_j unknowns, $1 \leq j \leq n+v$. We denote by $(\mathcal{P})$ the set of these n equations:

$$\forall i,\ 1 \leq i \leq n,\ y_i = P_i(x_1,...,x_{n+v}) \qquad (\mathcal{P}).$$

These n quadratic equations $(\mathcal{P})$ (in the $n+v$ unknowns x_j) are the public key.

Computation of a signature (with the secret key)

The computation of a signature x of y is performed as follows:

Step 1: We find n unknowns a_1, ..., a_n of K and v unknowns a'_1, ..., a'_v of K such that the n equations $(\mathcal{S})$ are satisfied.

This can be done as follows: we randomly choose the v vinegar unknowns a'_i, and then we compute the a_i unknowns from $(\mathcal{S})$ by Gaussian reductions (because – since there are no a_ia_j terms – the $(\mathcal{S})$ equations are affine in the a_i unknowns when the a'_i are fixed).

Remark: If we find no solution, then we simply try again with new random vinegar unknowns. After very few tries, the probability of obtaining at least one solution is very high, because the probability for a $n \times n$ matrix over $\mathbf{F}_q$ to be invertible is not negligible. (It is exactly $(1-\frac{1}{q})(1-\frac{1}{q^2})...(1-\frac{1}{q^{n-1}})$. For $q = 2$, this gives approximately 30 %, and for $q > 2$, this probability is even larger.)

Step 2: We compute $x = s^{-1}(A)$, where $A = (a_1,..,a_n,a'_1,...,a'_v)$. x is a signature of y.

Public verification of a signature

A signature x of y is valid if and only if all the $(\mathcal{P})$ are satisfied. As a result, no secret is needed to check whether a signature is valid: this is an asymmetric signature scheme.

Note: The name "Oil and Vinegar" comes from the fact that – in the equations $(\mathcal{S})$ – the "oil unknowns" a_i and the "vinegar unknowns" a'_j are not all mixed together: there are no a_ia_j products. However, in $(\mathcal{P})$, this property is hidden by the "mixing" of the unknowns by the s transformation. Is this property "hidden enough" ? In fact, this question exactly means: "is the scheme secure ?". When $v = n$, we call the scheme "Original Oil and Vinegar", since this case was first presented in [16]. This case was broken in [9]. It is very easy to see that the cryptanalysis of [9] also works, exactly in the same way, when $v < n$. However, the cases $v > n$ are, as we will see, much more difficult. When $v > n$, we call the scheme "Unbalanced Oil and Vinegar".

3 A short description of the attack of [9]: cryptanalysis of the case $v = n$

The idea of the attack of [9] is essentially the following:

In order to separate the oil variables and the vinegar variables, we look at the quadratic forms of the n public equations of $(\mathcal{P})$, we omit for a while the linear terms. Let G_i for $1 \leq i \leq n$ be the respective matrix of the quadratic form of P_i of the public equations $(\mathcal{P})$.

The quadratic part of the equations in the set $(\mathcal{S})$ is represented as a quadratic form with a corresponding $2n \times 2n$ matrix of the form : $\begin{pmatrix} 0 & A \\ B & C \end{pmatrix}$, the upper left $n \times n$ zero submatrix is due to the fact that an oil variable is not multiplied by an oil variable.

After hiding the internal variables with the linear function s, we get a representation for the matrices $G_i = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t$, where S is an invertible $2n \times 2n$ matrix.

Definition 3.1: We define the oil subspace to be the linear subspace of all vectors in K^{2n} whose second half contains only zeros.

Definition 3.2: We define the vinegar subspace as the linear subspace of all vectors in K^{2n} whose first half contains only zeros.

Lemma 1 *Let E and F be a $2n \times 2n$ matrices with an upper left zero $n \times n$ submatrix. If F is invertible then the oil subspace is an invariant subspace of EF^{-1}.*

Proof: E and F map the oil subspace into the vinegar subspace. If F is invertible, then this mapping between the oil subspace and the vinegar subspace is one to one and onto (here we use the assumption that $v = n$). Therefore F^{-1} maps back the vinegar subspace into the oil subspace this argument explains why the oil subspace is transformed into itself by EF^{-1}.

Definition 3.4: For an invertible matrix G_j, define $G_{ij} = G_i G_j^{-1}$.

Definition 3.5: Let O be the image of the oil subspace by S^{-1}.

In order to find the oil subspace, we use the following theorem:

Theorem 4 *O is a common invariant subspace of all the matrices G_{ij}.*

Proof:

$$G_i G_j^{-1} = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t (S^t)^{-1} \begin{pmatrix} 0 & A_j \\ B_j & C_j \end{pmatrix}^{-1} S^{-1} = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} \begin{pmatrix} 0 & A_j \\ B_j & C_j \end{pmatrix}^{-1} S^{-1}$$

The two inner matrices have the form of E and F in lemma 1. Therefore, the oil subspace is an invariant subspace of the inner term and O is an invariant subspace of $G_i G_j^{-1}$.

The problem of finding common invariant subspace of set of matrices is studied in [9]. Applying the algorithms in [9] gives us O. We then pick V to be an arbitrary subspace of dimension n such that $V + O = K^{2n}$, and they give an equivalent oil and vinegar separation.

Once we have such a separation, we bring back the linear terms that were omitted, we pick random values for the vinegar variables and left with a set of n linear equations with n oil variables.

Note: Lemma 1 is not true any more when $v > n$. The oil subspace is still mapped by E and F into the vinegar subspace. However F^{-1} does not necessary maps the image by E of the oil subspace back into the oil subspace and this is why the cryptanalysis of the original oil and vinegar is not valid for the unbalanced case.

This corresponds to the fact that, if the submatrix of zeros in the top left corner of F is smaller than $n \times n$, then F^{-1} does not have (in general) a submatrix of zeros in the bottom right corner. For example:

$$\begin{pmatrix} 0 & 3 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{pmatrix}^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -5 & 4 \\ 2 & -2 & 1 \\ -3 & 6 & -3 \end{pmatrix}.$$

However, when $v - n$ is small, we see in the next section how to extend the attack.

4 Cryptanalysis when $v > n$ and $v \simeq n$

In this section, we discuss the case of Oil and Vinegar schemes where $v > n$, although a direct application of the attack described in [9] and in the previous section does not solve the problem, a modification of the attack exists, that is applicable as long as $v - n$ is small (more precisely the expected complexity of the attack is approximately $q^{(v-n)-1} \cdot n^4$).

Definition 4.1: We define in this section the oil subspace to be the linear subspace of all vectors in K^{n+v} whose last v coordinates are only zeros.

Definition 4.2: We define in this section the vinegar subspace to be the linear subspace of all vectors in K^{n+v} whose first n coordinates are only zeros.

Here in this section, we start with the homogeneous quadratic terms of the equations: we omit the linear terms for a while.

The matrices G_i have the representation

$$G_i = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t$$

where the upper left matrix is the $n \times n$ zero matrix, A_i is a $n \times v$ matrix, B_i is a $v \times n$ matrix, C_i is a $v \times v$ matrix and S is a $(n+v) \times (n+v)$ invertible linear matrix.

Definition 4.3: Define E_i to be $\begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix}$.

Lemma 2 *For any matrix E that has the form $\begin{pmatrix} 0 & A \\ B & C \end{pmatrix}$, the following holds:*

a) E transforms the oil subspace into the vinegar subspace.

b) If the matrix E^{-1} exists, then the image of the vinegar subspace by E^{-1} is a subspace of dimension v which contains the n-dimensional oil subspace in it.

Proof: a) follows directly from the definition of the oil and vinegar subspaces. When a) is given then b) is immediate.

The algorithm we propose is a probabilistic algorithm. It looks for an invariant subspace of the oil subspace after it is transformed by S. The probability for the algorithm to succeed on the first try is small. Therefore we need to repeat it with different inputs. We use the following property: any linear combination of the matrices E_1, ..., E_n is also of the form $\begin{pmatrix} 0 & A \\ B & C \end{pmatrix}$.

The following theorem explains why an invariant subspace may exist with a certain probability.

Theorem 5 *Let F be an invertible linear combination of the matrices E_1, ..., E_n. Then for any k such that E_k^{-1} exists, the matrix FE_k^{-1} has a non trivial invariant subspace which is also a subspace of the oil subspace, with probability not less than $\frac{q-1}{q^{2d}-1}$ for $d = v - n$.*

Proof: The matrix F maps the oil subspace into the vinegar subspace, the image by F of the oil subspace is mapped by E_k^{-1} into a subspace of dimension v that contains the oil subspace – these are due to lemma 1. We write $v = n + d$, where d is a small integer. The oil subspace and its image by FE_k^{-1} are two subspaces with dimension n that reside in a subspace of dimension $n + d$. Therefore, their intersection is a subspace of the oil subspace with dimension not less than $n - d$. We denote the oil subspace by I_0 and the intersection subspace by I_1. Now, we take the inverse images by FE_k^{-1} of I_1: this is a subspace of I_0 (the oil subspace) with dimension not less than $n - d$ and the intersection between this subspace and I_1 is a subspace of I_1 with dimension not less than $n - 2d$. We call this subspace I_2. We can continue this process and define I_ℓ to be the intersection of $I_{\ell-1}$ and its inverse image by $FE_k - 1$. These two subspaces have co-dimension not more than d in $I_{\ell-2}$. Therefore, I_ℓ has a co-dimension not more than $2d$ in $I_{\ell-2}$ or a co-dimension not more than d in $I_{\ell-1}$. We can carry on this process as long as we are sure that the inverse image by FE_k^{-1} of I_ℓ has a non trivial intersection with I_ℓ. This is ensured as long as the dimension of I_ℓ is greater than d, but when the dimension is d or less than d, there is no guaranty that these two subspaces – that reside in $I_{\ell-1}$ – have a non trivial intersection. We end the process with I_ℓ that has dimension $\leq d$ that resides in $I_{\ell-1}$ with dimension not more than $2d$.

We know that the transformation $(EG_k^{-1})^{-1}$ maps I_ℓ into $I_{\ell-1}$. With probability not less than $\frac{q-1}{q^{2d}-1}$, there is a non zero vector in I_ℓ that is mapped to a non zero mutiple of itself – and therefore there is a non trivial subspace of $FE_k - 1$ which is also a subspace of the oil subspace.

Note: It is possible to get a better result for the expected number of eigenvectors and with much less effort: I_1 is a subspace with dimension not less than $n - d$ and is mapped by FE_k^{-1} into a subspace with dimension n. The probability for a non zero vector to be mapped to a non zero multiple of itself is $\frac{q-1}{q^n-1}$. To get the expected value, we multiply it by the number of non zero vectors in I_1. It gives a value which is not less than $\frac{(q-1)(q^{n-d}-1)}{q^n-1}$. Since every eigenvector is counted $q - 1$ times, then the expected number of invariant subspcaes of dimension 1 is not less than $\frac{q^{n-d}-1}{q^n-1} \sim q^{-d}$.

We define O as in section 3 and we get the following result for O:

Theorem 6 *Let F be an invertible linear combination of the matrices G_1; ..., G_n. Then for any k such that G_k^{-1} exists, the matrix FG_k^{-1} has a non trivial invariant subspace, which is also a subspace of O with probability not less than $\frac{q-1}{q^{2d}-1}$ for $d = v - n$.*

Proof:

$$FG_k^{-1} = (\alpha_1 G_1 + ... + \alpha_n G_n) G_k^{-1}$$
$$= S(\alpha_1 E_1 + ... + \alpha_n E_n) S^t (S^t)^{-1} E_k^{-1} S^{-1} = S(\alpha_1 E_1 + ... + \alpha_n E_n) E_k^{-1} S^{-1}.$$

The inner term is an invariant subspace of the oil subspace with the required probability. Therefore, the same will hold for FG_k^{-1}, but instead of a subspace of the oil subspace, we get a subspace of O.

How to find O?

We take a random linear combination of G_1, ..., G_n and multiply it by an inverse of one of the G_k matrices. Then we calculate all the minimal invariant subspaces of this matrix (a minimal invariant subspace of a matrix A contains no non trivial invariant subspaces of the matrix A – these subspaces corresponds to irreducible factors of the characteristic polynomial of A). This can be done in probabilistic polynomial time using standard linear algebra techniques. This matrix may have an invariant subspace wich is a subspace of O.

The following lemma enables us to distinguish between subspaces that are contained in O and random subspaces.

Lemma 3 *If H is a linear subspace and $H \subset O$, then for every x, y in H and every i, $G_i(x,y) = 0$ (here we regard G_i as a bilinear form).*

Proof: There are x' and y' in the oil subspace such that $x' = xS^{-1}$ and $y' = yS^{-1}$.

$$Gi(x,y) = xS \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t y^t = (x'S^{-1})S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} ((y'S^{-1})S)^t = x' \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} (y')^t = 0.$$

The last term is zero because x' and y' are in the oil subspace.

This lemma gives a polynomial test to distinguish between subspaces of O and random subspaces.

If the matrix we used has no minimal subspace which is also a subspace of O, then we pick another linear combination of G_1, ..., G_n, multiply it by an inverse of one of the G_k matrices and try again.

After repeating this process approximately q^{d-1} times, we find with good probability at least one zero vector of O. We continue the process until we get n independent vectors of O. These vectors span O.

The expected complexity of the process is proportional to $q^{d-1} \cdot n^4$. We use here the expected number of tries until we find a non trivial invariant subspace and the term n^4 covers the computational linear algebra operations we need to perform for evey try.

5 The cases $v \simeq \frac{n^2}{2}$ (or $v \geq \frac{n^2}{2}$)

Property

Let $(\mathcal{A})$ be a random set of n quadratic equations in $(n+v)$ variables x_1, ..., x_{n+v}. (By "random" we mean that the coefficients of these equations are uniformly and randomly chosen). When $v \simeq \frac{n^2}{2}$ (and more generally when $v \geq \frac{n^2}{2}$), there is probably – for most of such $(\mathcal{A})$ – a linear change of variables $(x_1,...,x_{n+v}) \mapsto (x'_1,...,x'_{n+v})$ such that the set $(\mathcal{A}')$ of $(\mathcal{A})$ equations written in $(x'_1,...,x'_{n+v})$ is an "Oil and Vinegar" system (*i.e.* there are no terms in $x'_i \cdot x'_j$ with $i \leq n$ and $j \leq n$).

An argument to justify the property

Let

$$\begin{cases} x_1 = \alpha_{1,1}x'_1 + \alpha_{1,2}x'_2 + ... + \alpha_{1,n+v}x'_{n+v} \\ \quad \vdots \\ x_{n+v} = \alpha_{n+v,1}x'_1 + \alpha_{n+v,2}x'_2 + ... + \alpha_{n+v,n+v}x'_{n+v} \end{cases}$$

By writing that the coefficient in all the n equations of $(\mathcal{A})$ of all the $x'_i \cdot x'_j$ ($i \leq n$ and $j \leq n$) is zero, we obtain a system of $n \cdot n \cdot \frac{n+1}{2}$ quadratic equations in the $(n+v) \cdot n$ variables $\alpha_{i,j}$ ($1 \leq i \leq n+v$, $1 \leq j \leq n$). Therefore, when $v \geq$ approximately $\frac{n^2}{2}$, we may expect to have a solution for this system of equations for most of $(\mathcal{A})$.

Remarks:

1. This argument is very natural, but this is not a complete mathematical proof.
2. The system may have a solution, but finding the solution might be a difficult problem. This is why an Unbalanced Oil and Vinegar scheme might be secure (for well chosen parameters): there is always a linear change of variables that makes the problem easy to solve, but finding such a change of variables might be difficult.
3. In section 7, we will see that, despite the result of this section, it is not recommended to choose $v \geq n^2$.

6 Solving a set of n quadratic equations in k unknowns, $k > n$, is NP-hard

We present in section 7 an algorithm that solves in polynomial complexity more than 99% of the sets of n quadratic equations in n^2 (or more) variables (*i.e.* it will probably succeed in more than 99% of the cases when the coefficients are randomly chosen).

Roughly speaking, we can summarize this result by saying that solving a "random" set of n quadratic equations in n^2 (or more) variables is feasible in polynomial complexity (and thus is not NP-hard if $P \neq NP$). However, we see in the present section that the problem of solving **any** (*i.e.* 100%) set of n quadratic equations in $k \geq n$ variables (so for example in $k = n^2$ variables) is NP-hard !

To see this, let us assume that we have a black box that takes any set of n quadratic equations with k variables in input, and that gives one solution when at least one solution exists. Then we can use this black box to find a solution for any set of n quadratic equations in n variables (and this is NP-hard). We proceed (for example) as follows. Let $(\mathcal{A})$ be a set of $(n-1)$ quadratic equations with $(n-1)$ variables x_1, x_2, ..., x_{n-1}. Then let y_1, ..., y_α be α more variables.

Let $(\mathcal{B})$ be the set of $(\mathcal{A})$ equations plus one quadratic equation in y_1, ..., y_α (for example the equation: $(y_1 + ... + y_\alpha)^2 = 1$). Then $(\mathcal{B})$ is a set of exactly n quadratic equations in $(n+1+\alpha)$ variables. It is clear that from the solution of $(\mathcal{B})$ we will immediately find one solution for $(\mathcal{A})$.

Note 1: $(\mathcal{B})$ has a very special shape ! This is why there is a polynomial algorithm for 99% of the equations without contradicting the fact that solving these sets $(\mathcal{B})$ of equations is a NP-hard problem.

Note 2: For $(\mathcal{B})$, we can also add more than one quadratic equations in the y_i variables and we can linearly mix these equations with the equations of $(\mathcal{A})$. In this case, $(\mathcal{B})$ is still of very special form but this very special form is less obvious at first glance since all the variables x_i and y_j are in all the equations of $(\mathcal{B})$.

7 A generally (but not always) efficient algorithm for solving a random set of n quadratic equations in n^2 (or more) unknowns

In this section, we describe an algorithm that solves a system of n randomly chosen quadratic equations in $n+v$ variables, when $v \geq n^2$.

Let $(\mathcal{S})$ be the following system:

$$(\mathcal{S}) \quad \begin{cases} \sum\limits_{1\leq i\leq j\leq n+v} a_{ij1}x_ix_j + \sum\limits_{1\leq i\leq n+v} b_{i1}x_i + \delta_1 = 0 \\ \quad\vdots \\ \sum\limits_{1\leq i\leq j\leq n+v} a_{ijn}x_ix_j + \sum\limits_{1\leq i\leq n+v} b_{in}x_i + \delta_n = 0 \end{cases}$$

The main idea of the algorithm consists in using a change of variables such as:

$$\begin{cases} x_1 = \alpha_{1,1}y_1 + \alpha_{2,1}y_2 + ... + \alpha_{n,1}y_n + \alpha_{n+1,1}y_{n+1} + ... + \alpha_{n+v,1}y_{n+v} \\ \quad\vdots \\ x_{n+v} = \alpha_{1,n+v}y_1 + \alpha_{2,n+v}y_2 + ... + \alpha_{n,n+v}y_n + \alpha_{n+1,n+v}y_{n+1} + ... + \alpha_{n+v,n+v}y_{n+v} \end{cases}$$

whose $\alpha_{i,j}$ coefficients (for $1 \leq i \leq n$, $1 \leq j \leq n+v$) are found step by step, in order that the resulting system $(\mathcal{S}')$ (written with respect to these new variables y_1, ..., y_{n+v}) is easy to solve.

- We begin by choosing randomly $\alpha_{1,1}$, ..., $\alpha_{1,n+v}$.
- We then compute $\alpha_{2,1}$, ..., $\alpha_{2,n+v}$ such that $(\mathcal{S}')$ contains no y_1y_2 terms. This condition leads to a system of n *linear* equations on the $(n+v)$ unknowns $\alpha_{2,j}$ $(1 \leq j \leq n+v)$:

$$\sum_{1\leq i\leq j\leq n+v} a_{ijk}\alpha_{1,i}\alpha_{2,j} = 0 \qquad (1 \leq k \leq n).$$

- We then compute $\alpha_{3,1}$, ..., $\alpha_{3,n+v}$ such that $(\mathcal{S}')$ contains neither y_1y_3 terms, nor y_2y_3 terms. This condition is equivalent to the following system of $2n$ *linear* equations on the $(n+v)$ unknowns $\alpha_{3,j}$ $(1 \leq j \leq n+v)$:

$$\begin{cases} \sum\limits_{1\leq i\leq j\leq n+v} a_{ijk}\alpha_{1,i}\alpha_{3,j} = 0 \qquad (1 \leq k \leq n) \\ \sum\limits_{1\leq i\leq j\leq n+v} a_{ijk}\alpha_{2,i}\alpha_{3,j} = 0 \qquad (1 \leq k \leq n) \end{cases}$$

- ...
- Finally, we compute $\alpha_{n,1}$, ..., $\alpha_{n,n+v}$ such that $(\mathcal{S}')$ contains neither y_1y_n terms, nor y_2y_n terms, ..., nor $y_{n-1}y_n$ terms. This condition gives the following system of $(n-1)n$ *linear* equations on the $(n+v)$ unknowns $\alpha_{n,j}$ $(1 \leq j \leq n+v)$:

$$\begin{cases} \sum\limits_{1\leq i\leq j\leq n+v} a_{ijk}\alpha_{1,i}\alpha_{n,j} = 0 \qquad (1 \leq k \leq n) \\ \quad\vdots \\ \sum\limits_{1\leq i\leq j\leq n+v} a_{ijk}\alpha_{n-1,i}\alpha_{n,j} = 0 \qquad (1 \leq k \leq n) \end{cases}$$

In general, all these linear equations provide at least one solution (found by Gaussian reductions). In particular, the last system of $n(n-1)$ equations and $(n+v)$ unknowns generally gives a solution, as soon as $n+v > n(n-1)$, *i.e.* $v > n(n-2)$, which is true by hypothesis.

Moreover, the n vectors $\begin{pmatrix} \alpha_{1,1} \\ \vdots \\ \alpha_{1,n+v} \end{pmatrix}$, ..., $\begin{pmatrix} \alpha_{n,1} \\ \vdots \\ \alpha_{n,n+v} \end{pmatrix}$ are very likely to be linearly independent for a random quadratic system $(\mathcal{S})$.

The remaining $\alpha_{i,j}$ constants (*i.e.* those with $n+1 \leq i \leq n+v$ and $1 \leq j \leq n+1$) are randomly chosen, so as to obtain a *bijective* change of variables.

By rewriting the system $(\mathcal{S})$ with respect to these new variables y_i, we are led to the following system:

$$(\mathcal{S}') \quad \begin{cases} \sum_{i=1}^{n} \beta_{i,1} y_i^2 + y_1 L_{1,1}(y_{n+1},...,y_{n+v}) + ... + y_n L_{n,1}(y_{n+1},...,y_{n+v}) + Q_1(y_{n+1},...,y_{n+v}) = 0 \\ \vdots \\ \sum_{i=1}^{n} \beta_{i,n} y_i^2 + y_1 L_{1,n}(y_{n+1},...,y_{n+v}) + ... + y_n L_{n,n}(y_{n+1},...,y_{n+v}) + Q_n(y_{n+1},...,y_{n+v}) = 0 \end{cases}$$

where each $L_{i,j}$ is an affine function and each Q_i is a quadratic function.

We then compute y_{n+1}, ..., y_{n+v} such that:

$$\forall i,\ 1 \leq i \leq n,\ \forall j,\ 1 \leq j \leq n+v,\ L_{i,j}(y_{n+1},...,y_{n+v}) = 0.$$

This is possible because we have to solve a linear system of n^2 equations and v unknowns, which generally provides at least one solution, as long as $v \geq n^2$. We pick one of these solutions.

It remains to solve the following system of n equations on the n unknowns y_1, ..., y_n:

$$(\mathcal{S}'') \quad \begin{cases} \sum_{i=1}^{n} \beta_{i1} y_i^2 = \lambda_1 \\ \vdots \\ \sum_{i=1}^{n} \beta_{in} y_i^2 = \lambda_n \end{cases}$$

where $\lambda_k = -Q_k(y_{n+1},...,y_{n+v})$ $(1 \leq k \leq n)$.

In general, this gives the y_i^2 by Gaussian reduction.

Then, in characteristic 2, since $x \mapsto x^2$ is a bijection, we will then find a solution for the y_i from this expression of the y_i^2.

Note: In characteristic $\neq 2$, this algorithm will also succeed when 2^n is not too large (*i.e.* when $n \leq 40$ for example). (However, when $2^n \geq 2^{64}$ and when the characteristic is $\neq 2$, this algorithm requires too many computations.)

8 A variation with twice smaller signatures

In the UOV described in section 2, the public key is a set of n quadratic equations $y_i = P_i\ (x_1,\ ...,\ x_{n+v})$, for $1 \leq i \leq n$, where $y = (y_1,...,y_n)$ is the hash value of the message to be signed. If we use a collision-free hash function, the hash value must at least be 128 bits long. Therefore, q^n must be at least 2^{128}, so that the typical length of the signature, if $v = 2n$, is at least $3 \times 128 = 384$ bits.

As we see now, it is possible to make a small variation in the signature design in order to obtain twice smaller signatures. The idea is to keep the same polynomial P_i (with the same associated secret key), but now the public equations that we check are:

$$\forall i,\ P_i(x_1,...,x_{n+v}) + L_i(y_1,...,y_n,x_1,...,x_{n+v}) = 0,$$

where L_i is a linear function in $(x_1,...,x_{n+v})$ and where the coefficients of L_i are generated by a hash function in $(y_1,...,y_n)$.

For example $L_i(y_1,...,y_n,x_1,...,x_{n+v}) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_{n+v} x_{n+v}$, where $(\alpha_1,\alpha_2,...,\alpha_{n+v}) =$ Hash $(y_1, ..., y_n||i)$. Now, n can be chosen such that $q^n \geq 2^{64}$ (instead $q^n \geq 2^{128}$). (Note: q^n must be $\geq 2^{64}$ in order to avoid exhaustive search on a solution x). If $v = 2n$ and $q^n \simeq 2^{64}$, the length of the signature will be $3 \times 64 = 192$ bits.

9 Oil and Vinegar of degree three

9.1 The scheme

The quadratic Oil and Vinegar schemes described in section 2 can easily be extended to any higher degree. We now present the schemes in degree three.

Variables

Let K be a small finite field (for example $K = \mathbf{F}_2$). Let a_1, ..., a_n be n elements of K, called the "oil" unknowns. Let a'_1, ..., a'_v be v elements of K, called the "vinegar" unknowns.

Secret key.

The secret key is made of two parts:

1. A bijective and affine function $s : K^{n+v} \to K^{n+v}$.
2. A set $(\mathcal{S})$ of n equations of the following type: for all $i \leq n$,

$$y_i = \sum \gamma_{ijk\ell} a_j a'_k a'_\ell + \sum \mu_{ijk\ell} a'_j a'_k a'_\ell + \sum \lambda_{ijk} a_j a'_k + \sum \nu_{ijk} a'_j a'_k + \sum \xi_{ij} a_j + \sum \xi'_{ij} a'_j + \delta_i \qquad (\mathcal{S}).$$

The coefficients γ_{ijk}, $\mu_{ijk\ell}$, λ_{ijk}, ν_{ijk}, ξ_{ij}, ξ'_{ij} and δ_i are the secret coefficients of these n equations. Note that these equations $(\mathcal{S})$ contain no terms in $a_j a_k a_\ell$ or in $a_j a_k$: the equations are affine in the a_j unknowns when the a'_k unknowns are fixed.

Public key

Let A be the element of K^{n+v} defined by $A = (a_1,...,a_n,a'_1,...,a'_v)$. A is transformed into $x = s^{-1}(A)$, where s is the secret, bijective and affine function from K^{n+v} to K^{n+v}. Each value y_i, $1 \leq i \leq n$, can be written as a polynomial P_i of total degree three in the x_j unknowns, $1 \leq j \leq n+v$. We denote by $(\mathcal{P})$ the set of the following n equations:

$$\forall i,\ 1 \leq i \leq n, y_i = P_i(x_1,...,x_{n+v}) \qquad (\mathcal{P}).$$

These n equations $(\mathcal{P})$ are the public key.

Computation of a signature

Let y be the message to be signed (or its hash value).

Step 1: We randomly choose the v vinegar unknowns a'_i, and then we compute the a_i unknowns from $(\mathcal{S})$ by Gaussian reductions (because – since there are no $a_i a_j$ terms – the $(\mathcal{S})$ equations are affine in the a_i unknowns when the a'_i are fixed. (If we find no solution for this affine system of n equations and n "oil" unknowns, we just try again with new random "vinegar" unknowns.)

Step 2: We compute $x = s^{-1}(A)$, where $A = (a_1,...,a_n,a'_1,...,a'_v)$. x is a signature of y.

Public verification of a signature

A signature x of y is valid if and only if all the $(\mathcal{P})$ are satisfied.

9.2 First cryptanalysis of Oil and Vinegar of degree three when $v \leq n$

We can look at the quadratic part of the public key and attack it exactly as for an Oil and Vinegar of degree two. This is expected to work when $v \leq n$.

Note: If there is no quadratic part (*i.e.* is the public key is homogeneous of degree three), or if this attack does not work, then it is always possible to apply a random affine change of variables and to try again. Moreover, we will see in section 9.3 that, surprisingly, there is an even easier and more efficient attack in degree three than in degree two!

9.3 Cryptanalysis of Oil and Vinegar of degree three when $v \leq (1+\sqrt{3})n$ and K is of characteristic $\neq 2$ (from an idea of [2])

The key idea is to detect a "linearity" in some directions. We search the set V of the values $d = (d_1,...,d_{n+v})$ such that:

$$\forall x,\ \forall i,\ 1 \leq i \leq n,\ P_i(x+d) + P_i(x-d) = 2P_i(x) \qquad (\#).$$

By writing that each x_k indeterminate has a zero coefficient, we obtain $n \cdot (n+v)$ quadratic equations in the $(n+v)$ unknowns d_j.

(Each monomial $x_i x_j x_k$ gives $(x_j+d_j)(x_k+d_k)(x_\ell+d_\ell)+(x_j-d_j)(x_k-d_k)(x_\ell-d_\ell)-2x_j x_k x_\ell$, *i.e.* $2(x_j d_k d_\ell + x_k d_j d_\ell + x_\ell d_j d_k)$.)

Furthermore, the cryptanalyst can specify about $n-1$ of the coordinates d_k of d, since the vectorial space of the correct d is of dimension n. It remains thus to solve $n \cdot (n+v)$ quadratic equations in $(v+1)$ unknowns d_j. When v is not too large (typically when $\frac{(v+1)^2}{2} \leq n(n+v)$, *i.e.* when $v \leq (1+\sqrt{3})n$), this is expected to be easy.

As a result when $v \leq$ approximately $(1+\sqrt{3})n$ and $|K|$ is odd, this gives a simple way to break the scheme.

Note 1: When v is sensibly greater than $(1+\sqrt{3})n$ (this is a more unbalanced limit than what we had in the quadratic case), we do not know at the present how to break the scheme.

Note 2: Strangely enough, this cryptanalysis of degre three Oil and Vinegar schemes does not work on degree two Oil and Vinegar schemes. The reason is that – in degree two –writing

$$\forall x,\ \forall i,\ 1 \leq i \leq n,\ P_i(x+d) + P_i(x-d) = 2P_i(x)$$

only gives n equations of degree two on the $(n+v)$ d_j unknowns (that we do not know how to solve).

(Each monomial $x_j x_k$ gives $(x_j + d_j)(x_k + d_k) + (x_j - d_j)(x_k - d_k) - 2x_j x_k$, *i.e.* $2d_j d_k$.)

Note 3: In degree two, we have seen that Unbalanced Oil and Vinegar public keys are expected to cover almost all the set of n quadratic equations when $v \simeq \frac{n^2}{2}$. In degree three, we have a similar property: the public keys are expected to cover almost all the set of n cubic equations when $v \simeq \frac{n^3}{6}$ (the proof is similar).

10 Public key length

If we choose $K = \mathbf{F}_2$ then the public key is often large. So it is often more practical to choose a larger K and a smaller n: then the length of the public key can be reduced a lot (see the examples in section 14). However, even when K and n are fixed, it is always feasible to make some easy transformations on a public key in order to obtain the public key in a canonical way such that this canonical expression is slightly shorter than the original expression.

- First, it is always possible to publish only the homogeneous part of the quadratic equations (and not the linear part), because if we know the secret affine change of variables in an Oil and Vinegar scheme with a public key P, then we can solve $P(x) + L(x) = y$, where L is any linear expression with exactly the same affine change of variables. It is thus possible to publish only the homogeneous part of P and to choose a convention for computing the linear part L of the public key (instead of publishing L). For example, this convention can be that the linear terms of L in the equation number i $(1 \leq i \leq n)$ are computed from $\mathrm{Hash}(i||Id)$ (or from $\mathrm{Hash}(i||P)$), where Hash is a public hash function and where Id is the identity of the owner of the secret key.

  Remark: It is also possible to decide that the linear part is always zero. However, from a theoretical point of view, this may be less secure because we cannot exlude the possibility that some efficient attacks exist against the homogeneous Oil and Vinegar without finding the secret key (and without breaking the non-homogeneous case).
- On the equations, it is also possible to:
  1. Make linear and bijective changes of variable $x' = A(x)$.
  2. Compute a linear and bijective transformation on the equation: $\mathcal{P}' = t(\mathcal{P})$. (For example, the new first equation can be the old first plus the old third equation, etc).

  By combining easily these two transformations, it is always possible to decrease slightly the lenght of the public key.

Idea 1: It is possible to make a change of variables such that the first equation is in a canonical form (see [11], chapter 6). With this presentation of the public key, the length of the public key will be approximately $\frac{n-1}{n}$ times the initial length.

Idea 2: Another idea is to use the idea of section 7, *i.e.* to create a square of $\lambda \times \lambda$ zeros in the coefficients, where $\lambda \simeq \sqrt{n+v}$. With this presentation, the lenght of the public key is approximately $\frac{(n+v)^2-(n+v)}{(n+v)^2}$ times the initial length.

Remark: As we will see in section 13, the most efficient way of reducing the length of the public key is to choose carefully the values q and n.

11 Another variation of the schemes: Unbalanced Oil, Vinegar and Salt

The scheme

Let $(\mathcal{A})$ be a set of n quadratic "Oil and Vinegar" equations, as described above, with n oil variables and v vinegar variables. We denote by $(q_1,...,q_n)$ these equations. Let $(\mathcal{A}')$ be a set of r truly trandom quadratic equations in all the variables (*i.e.* we can have terms in $a_i a_j$ where a_i and a_j are oil variables in $(\mathcal{A}')$ but not in $(\mathcal{A})$). We denote by $(q'_1,...,q'_n)$ these equations. We will call these r equations the "salt" equations.

Let t be a secret affine permutation of $K^{n+r} \rightarrow K^{n+r}$. Let $(\mathcal{P})$ be the set of the equations $t(q_1, ..., q_n, q'_1, ..., q'_r)$. We denote by $P_1, ..., P_{n+r}$ these equations of $(\mathcal{P})$. $(\mathcal{P})$ will be the public key (*i.e.* we have "mixed" Oil and Vinegar quadratic equations and truly random quadratic equations with a secret affine permutation $(\mathcal{P})$.

Let $y \in K^{n+r}$ be the hash of a message M to be signed (or $y =$Hash$(M||0010||R)$) where R is a random value with no 0010 in base 2). Let $x \in K^{n+v}$. Then x is a valid signature of y if $P(x) = y$ (*i.e.* if $\forall i$, $1 \leq i \leq n+r$, $P_i(x) = y_i$).

When the secret affine functions s and t are known, it is feasible to compute a valid signature after approximately $\mathcal{O}(q^r)$ computations because we will easily compute a solution for the n equations $(\mathcal{A})$ as before, and the probability that this solution also satisfies the r equations $(\mathcal{A}')$ is $\frac{1}{q^r}$ (we will try again with another random R until we succeed). When q^r is small (for example if $q \leq 256$ and $r \leq 2$), this is clearly feasible. (The name "salt" comes from the fact that we cannot put a lot of salt equations since q^r must stay small for efficiency.)

Cryptanalysis when $v = n$

Here we assume that $v = n$.

Let G_i and G_j be random linear sums of the $n+r$ equations $(\mathcal{P})$. The probability that G_i and G_j are linear sums of only the n equations $(\mathcal{A})$ is $(1/q^r)^2$ (because it is $1/q^r$ for G_i and $1/q^r$ for G_j). If this occurs, then from G_i and G_j, we will attack the scheme exactly as described in [9]. Therefore, if $v = n$, the scheme can be attacked with a complexity approximately q^{2r} (and for the legitimate user, computing a signature has a complexity approximately $\mathcal{O}(q^r)$). As a result, we do not recommend to use this variation when $v = n$.

The case $v > n$

For well chosen parameters, we have seen that we do not know how to attack Unbalanced Oil and Vinegar schemes. Therefore, of course, we do not know either how to attack the schemes when the two ideas – $v > n$ and mixing the equations with truly random equations – are combined together. However, the idea of choosing $v > n$ seems at the present to be a stronger idea (both for security and for practical implementations) than the idea of mixing Oil and Vinegar with truly random equations.

12 Another scheme: HFEV

The Unbalanced Oil and Vinegar schemes and the HFE schemes of [14] can very easily be combined, as we will see in this section. Moreover, the combined scheme looks very efficient since (at the present) we are able to avoid all the known attacks with more efficient choices of the parameters. So this HFEV schemes look both more efficient (because a smaller degree d looks sufficient for security) and more secure compared to the original HFE scheme. HFEV is also more efficient (but more complex) compared to UOV, because very few vinegar variables are needed.

The scheme (HFEV)

In the "most simple" HFE scheme (we use the notations of [14]), we have $b = f(a)$, where:

$$f(a) = \sum_{i,j} \beta_{ij} a^{q^{\theta_{ij}}+q^{\varphi_{ij}}} + \sum_i \alpha_i a^{q^{\xi_i}} + \mu_0, \qquad (1)$$

where β_{ij}, α_i and μ_0 are elements of the field $\mathbf{F}_{q^n}$.

Let v be an integer (v will be the number of extra x_i variables, or the number of "vinegar" variables that we will add in the scheme).

Let $a' = (a'_1,...,a'_v)$ be a v-uple of variables of K. Let now each α_i of (1) be an element of $\mathbf{F}_{q^n}$ such that each of the n components of α_i in a basis is a secret random linear function of the vinegar variables a'_1, ..., a'_v.

And in (1), let now μ_0 be an element of $\mathbf{F}_{q^n}$ such that each one of the n components of μ_0 in a basis is a secret random quadratic function of the variables a'_1, ..., a'_v.

Then, the $n+v$ variables a_1, ..., a_n, a'_1, ..., a'_v will be mixed in the secret affine bijection s in order to obtain the variables x_1, ..., x_{n+v}.

And, as before, $t(b_1,...,b_n) = (y_1,...,y_n)$, where t is a secret affine bijection.

Then the public key is given as the n equations $y_i = P_i(x_1,...,x_{n+v})$.

To compute a signature, the vinegar values a'_1, ..., a'_v will simply be chosen at random. Then, the values μ_0 and α_i will be computed. Then, the monovariate equations (1) will be solved (in a) in $\mathbf{F}_{q^n}$.

Simulations

Nicolas Courtois did some simulations on HFEV and, in all his simulations, when the number of vinegar variables is ≥ 3, there is no affine multiple equations of small degree (which is very nice).

Example: Let $K = \mathbf{F}_2$. In HFEV, we can, for example, choose the hidden polynomial to be:

$$f(a) = a^{17} + \beta_{16}a^{16} + a^{12} + a^{10} + a^9 + \beta_8 a^8 + a^6 + a^5 + \beta_4 a^4 + a^3 + \beta_2 a^2 + \beta_1 a + \beta_0,$$

where:

- $a = (a_1,...,a_n)$, where a_1, ..., a_n are the "oil" variables.
- β_1, β_2, β_4, β_8 and β_{16} are given by n secret linear functions on the v vinegar variables.
- β_0 is given by n secret quadratic functions on the v vinegar variables.

In this example, we compute a signature as follows: the vinegar variables are chosen at random and the resulting equation of degree 17 is solved in a.

Note: Unlike UOV, in HFEV we have terms in oil×oil (such as a^{17}, a^{12}, a^{10}, etc), oil×vinegar (such as $\beta_{16}a^{16}$, $\beta_8 a^8$, etc) and vinegar×vinegar (in β_0).

13 The C^*V and C^*VV schemes

We can also introduce some vinegar variables in the C^* scheme of [12]. We call C^*V the resulting scheme. However, the "oil" and "vinegar" variables are not as well mixed in C^*V compared with HFEV, because in C^*V we will have terms in oil×oil and vinegar×vinegar, but not in oil×vinegar. So the C^*V scheme can seen as in figure 1.

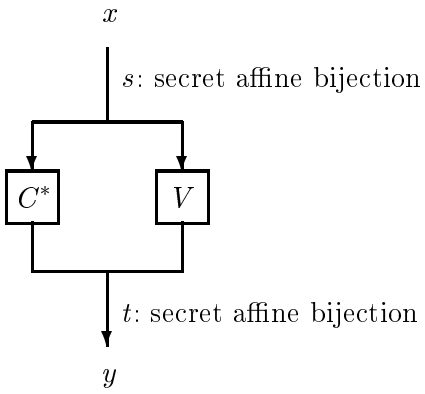


FIG. 1 – *The C^*V scheme*

- When the number v of vinegar variables is such that $q^v <$ about 2^{50}, then the scheme can be attacked with some of the cryptanalysis algorithms described in [18], since V can then be seen as a small S-box in v variables. So the C^*V scheme is insecure when $q^v <$ about 2^{50}.
- However, when $q^v \geq 2^{64}$, we do not know how to attack the scheme. In this case, we call the scheme the "C^*VV" scheme (in order to show that we have many vinegar variables). The C^*VV scheme cannot be used for encryption any more, but the scheme is still very efficient for signatures. (These properties of C^*VV look very similar to the C^*_{--} scheme described in [20].)

14 Summary of the results for UOV

The underlying field is $K = \mathbf{F}_q$ with $q = p^m$. Its characteristic is p.

"As difficult as random" means that the problem of breaking the scheme is expected to be as difficult as the problem of solving a system of equations in v variables when the coefficients are randomly chosen (*i.e.* with no trapdoor).

Degree	Broken	Not Broken	Not broken and as difficult as random	Broken (despite as difficult as random)
2 (for all p)	$v \leq n$ or $v \simeq n$	$2n \leq v \leq \frac{n^2}{2}$	$\frac{n^2}{2} \leq v \leq n^2$	$v \geq n^2$
3 (for $p = 2$)	$v \leq (1+\sqrt{3})n$	$(1+\sqrt{3})n \leq v \leq \frac{n^3}{6}$	$\frac{n^3}{6} \leq v \leq \frac{n^3}{2}$	$v \geq \frac{n^3}{6}$
3 (for $p \neq 2$)	$v \leq n$ or $v \simeq n$	$2n \leq v \leq \frac{n^3}{6}$	$\frac{n^3}{6} \leq v \leq n^4$	$v \geq n^4$

In this table, we have summarized our current results on the attacks on Unbalanced Oil and Vinegar schemes. The original paper ([9]) was only studying the case $v = n$ for quadratic equations.

15 Concrete examples of parameters for UOV

In all the examples below, we do not know how to break the scheme. We have arbitrary chosen $v = 2n$ (or $v = 3n$) in all these examples (since $v \leq n$ and $v \geq n^2$ are insecure).

Example 1: $K = \mathbf{F}_2$, $n = 128$, $v = 256$ (or $v = 384$). The signature scheme is the one of section 2. The length of the public key is approximately $n \cdot (\frac{(n+v)^2}{2})$ bits. This gives here a huge value: approximately 1.1 Mbytes (or 2 Mbytes)! The length of the secret key (the s matrix) is approximately $(n+v)^2$ bits, *i.e.* approximately 18 Kbytes. However, this secret key can always be generated from a small secret seed of, say, 64 bits.

Example 2: $K = \mathbf{F}_2$, $n = 64$, $v = 128$ (or $v = 192$). The signature scheme is the one section 8. The length of the public key is 144 Kbytes (or 256 Kbytes).

Example 3: $K = \mathbf{F}_{16}$, $n = 16$, $v = 32$ (or $v = 48$). s is a secret affine bijection of $\mathbf{F}_{16}$. The signature scheme is the one section 8. The length of the public key is 9 Kbytes (or 16 Kbytes).

Example 4: $K = \mathbf{F}_{16}$, $n = 16$, $v = 32$ (or $v = 48$). s is a secret affine bijection of $\mathbf{F}_{16}$ such that all its coefficients lie in $\mathbf{F}_2$. Moreover, the secret quadratic coefficients are also chosen in $\mathbf{F}_2$, so that the public functions P_i, $1 \leq i \leq n$, are n quadratic equations in $(n+v)$ unknowns of $\mathbf{F}_{16}$, with coefficients in $\mathbf{F}_2$. In this case (the signature scheme is still the one of section 8), the length of the public key is 2.2 Kbytes (or 4 Kbytes).

Note: In all these examples, $n \geq 16$ in order to avoid Gröbner bases algorithms to find a solution x (cf [5]), and $q^n \geq 2^{64}$ in order to avoid exhaustive search on x.

16 Concrete example of parameters for HFEV

At the present, it seems possible to choose a small value for v (for example $v = 3$) and a small value for d (for example $d = 17$ if $K = \mathbf{F}_2$).

17 State of the art (in May 1999) on Public-Key schemes with Multivariate Polynomials over a small finite field

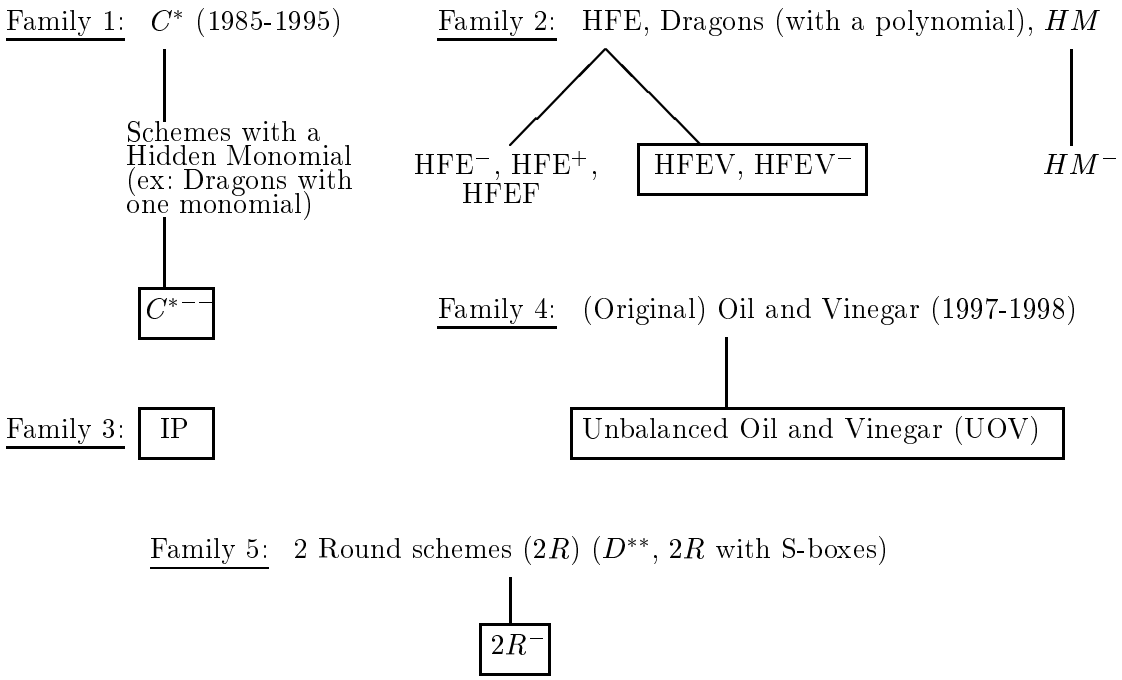
Recently, many new ideas have been introduced to improve the schemes, such as UOV or HFEV described in this paper. Another idea is to fix some variables to hide some algebraic

properties (see below). However, many new ideas have also been introduced to design better attacks on previous schemes, such as the – not yet published – papers [10], [3], [1], [4]. So the field is fast moving and it can look a bit confusing at first. Moreover, some authors use the word "cryptanalysis" for "breaking" and some authors use this word with the meaning "an analysis about the security" that does not necessary mean "breaking". In this section, we describe what we know at the present about the main schemes.

In the large families of the schemes with a public key based on multivariate polynomials over a small finite field, we can distinguish between 5 main families characterized by the way the trapdoor is introduced or on the difficult problem on which the security relies.

In the first family are the schemes "with a Hidden Monomial", *i.e.* the key idea is to compute an exponentiation $x \mapsto x^d$ in a finite field for secret key computation and to "hide" such a function in the public key. In the second family are the schemes where a polynomial function (with more than one monomial) is hidden. In the third family, the security relies on an isomorphism problem. In the fourth family, the secret key computations are based on Gaussian computations. Finally, in the fifth family, the security relies on the difficulty of finding the decomposition of two multivariate quadratic polynomials from all or part of their composition

The main schemes in these families are described in the figure below. The most interesting schemes in each family are in a rectangle.



- C^* was the first scheme of all, and it can be seen as the ancestor of all these schemes. It was designed in [12] and broken in [13].
- Schemes with a Hidden Monomial (such as some Dragon schemes) were studied in [15], where it is shown that most of the simplest variations of C^* are insecure. However, C^{*--} (studied in [20]) is (at the present) the most efficient signature scheme (in time and RAM) in a smartcard. The scheme is not broken (but it may seem too simple or too close to C^* to have a large confidence in its security ...).

- HFE was designed in [14]. The most recent results about its security are in [10] and [3]. In these papers, very clever attacks are described. However, at the present, the HFE scheme is still not broken since for well chosen and still reasonable parameters the computations required to break it are still too large (moreover, asymptotically, the cryptanalysis is not polynomial if d increases as $d = \mathcal{O}(n)$ for example). For example, the first challenge of US \$500 given in the extended version of [14] has not been claimed yet (it is a pure HFE with $n = 80$ and $d = 96$ over $\mathbf{F}_2$).
- HFE$^-$ is just an HFE where k of the originally public equations are not publish. Due to [10] and [3], it may be recommended to do this (despite the fact that original HFE may be secure without it). In the extended version of [14] a second challenge of US \$500 is described on a HFE$^-$. In an encryption scheme, k must be small, but in a signature scheme, k may be large.
- HFEV is described in this paper. HFEV and HFEV$^-$ look very hard to break. Moreover, HFEV is more efficient than the original HFE and it can give public key signatures of only 80 bits ! In a signature scheme, the number v of "vinegar variables" can be large, but in an encryption scheme, v must be small.
- HFE$^+$ is just an HFE scheme where the n originally public equations have been linearly mixed with k truly random equations. In a signature scheme, k must be small, but in an encryption scheme, k may be large.
- HFEF is just an HFE scheme where k of the variables x_i have been fixed. In a signature scheme, k must be small, but in an encryption scheme, k may be large.
- HFEVF^{+-} is just an HFE scheme where all these "perturbations" (V, F, $+$, $-$) have been applied on the public key.
- HM and HM^- were designed in [20]. Very few analysis have been done in these schemes (but maybe we can recommend to use HM^- instead of HM ?).
- IP was designed in [14]. IP schemes have the best proofs of security so far (see [19]). IP is very simple and can be seen as a nice generalization of Graph Isomorphism.
- Oil and Vinegar was presented in [16] and broken in [9].
- UOV is described in this paper. With IP, they are certainly the most simple schemes.
- $2R$ was designed in [17] and [18]. Due to [1], it is necessary to have at least 128 bits in input in the "$2R$ with S-boxes" scheme, and due to [4], it may be wise to not publish all the (originally) public equations in all the $2R$ schemes: this gives the $2R^-$ algorithms (the efficiency of the decomposition algorithms given in [4] on the $2R$ schemes is not yet completely clear).

Remark 1: When a new scheme is found in these families, we do not necessary have to explain how the trapdoor has been introduced. Then we have a "Secret-Public Key scheme" ! The scheme is clearly a Public Key scheme since anybody can verify a signature from the public key (or can encrypt from the public key) and the scheme is secret since the way to compute the secret key computations (*i.e.* the way the trapdoor has been introduced) has not been revealed. For example, we could have done this for HFEV (instead of publishing it).

Remark 2: These schemes are of theoretical interest but (at the exception of IP) their security is not directly relied to a clearly defined and considered to be difficult problem. So is it reasonable to implement them in real products ? We think indeed that it is a bit risky to rely all the security of sensitive applications on such scheme. However, at the present, most of the smartcard

applications use secret key algorithms because RSA smartcards are more expensive. So it can be reasonable to put in a low-cost smartcard one of the previous public key schemes in addition to (not instead of) the existing secret key schemes (such as Triple-DES). Then the security can only be increased, the price of the smartcard would still be low (no coprocessor needed). The security would then rely on a master secret key for the secret key algorithm (with the risk of depending on a master secret key) and on a new low-cost public-key scheme (with the risk that the scheme has no proof of security).

18 Conclusion

In this paper, we have presented two new public key schemes: UOV and HFEV. The study of such schemes has led us to analyze very general properties about the solutions of systems of general quadratic forms. Moreover, from the general view presented in section 15, we see that these two schemes are at the present among the most interesting schemes in two of the five main families of schemes based on multivariate polynomials over a small finite field. Will this still be true in a few years?

References

[1] E. Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes*, not yet published.

[2] D. Coppersmith, *personal communication*, e-mail.

[3] N. Courtois, *The Security of HFE*, not yet published.

[4] Y. Ding-Feng, L. Kwok-Yan, D. Zong-Duo, *Cryptanalysis of "2R" Schemes*, Proceedings of CRYPTO'99, Springer, pp. 315-325.

[5] J.-C. Faugere, *personal communication*.

[6] H. Fell, W. Diffie, *Analysis of a public key approach based on polynomial substitutions*, Proceedings of CRYPTO'85, Springer-Verlag, vol. 218, pp. 340-349

[7] M. Garey, D. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman, p. 251.

[8] H. Imai, T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, Algebraic Algorithms and Error Correcting Codes (AAECC-3), Grenoble, 1985, Springer-Verlag, LNCS n°229.

[9] A. Kipnis, A. Shamir, *Cryptanalysis of the Oil and Vinegar Signature Scheme*, Proceedings of CRYPTO'98, Springer, LNCS n°1462, pp. 257-266.

[10] A. Kipnis, A. Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Proceedings of CRYPTO'99, Springer, pp. 19-30.

[11] R. Lidl, H. Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its applications, volume 20, Cambridge University Press.

[12] T. Matsumoto, H. Imai, *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption*, Proceedings of EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[13] J. Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, Proceedings of CRYPTO'95, Springer-Verlag, pp. 248-261.

[14] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP) : Two New Families of Asymmetric Algorithms*, Proceedings of EUROCRYPT'96, Springer, pp. 33-48.

[15] J. Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Proceedins of CRYPTO'96, Springer, pp. 45-60.

[16] J. Patarin, *The Oil and Vinegar Signature Scheme*, presented at the Dagstuhl Workshop on Cryptography, september 1997 (transparencies).

[17] J. Patarin, L. Goubin, *Trapdoor One-way Permutations and Multivariate Polynomials*, Proceedings of ICICS'97, Springer, LNCS n°1334, pp. 356-368.

[18] J. Patarin, L. Goubin, *Asymmetric Cryptography with S-Boxes*, Proceedings of ICICS'97, Springer, LNCS n°1334, pp. 369-380.

[19] J. Patarin, L. Goubin, N. Courtois, *Improved Algorithms for Isomorphisms of Polynomials*, Proceedings of EUROCRYPT'98, Springer, pp. 184-200.

[20] J. Patarin, L. Goubin, N. Courtois, *C^*_{-+} and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai*, Proceedings of ASIACRYPT'98, Springer, pp. 35-49.

[21] A. Shamir, *A simple scheme for encryption and its cryptanalysis found by D. Coppersmith and J. Stern*, presented at the Luminy workshop on cryptography, september 1995.

Cryptanalysis of the TTM Cryptosystem

ASIACRYPT'2000
Article avec Nicolas Courtois (Bull CP8)

Abstract

In 1985 H. Fell and W. Diffie studied a paradigm for constructing public key cryptosystems based on sequentially solved multivariate equations [9], which can be seen as a triangular system. In the present paper, we study a more general family of TPM (for "Triangle Plus Minus") schemes. A TPM is a triangular construction, with added u final random polynomials and r beginning equations removed.

We go beyond all previous attacks proposed on such cryptosystems, that used the low degree of a component of the inverse function. We show that the the cryptanalysis of TPM reduces to solving a simple linear algebra problem called *MinRank(r)*: Find a linear combination of given matrices that has a small rank r.

We present a new algorithm for the MinRank(r) problem and the TPM cryptosystems. It is called 'Kernel Attack' and is polynomial for a fixed r. As an application of this technique, we present two different attacks on the TTM cryptosystem proposed by T.T. Moh at CrypTec'99 [13, 14] with $r = 2$.

Though the TTM cleartext is 512 bits long, we are able to completely break TTM (*i.e.* to recover the secret key) in $\mathcal{O}(2^{52})$. Moreover, the particular cryptosystem described in [13, 14] has additional weaknesses that allows an attack in $\mathcal{O}(2^{28})$.

The attacks we describe are both theoretical and pratical: as an example, we present the solution to the TTM 2.1 challenge proposed by the company US Data Security, currently selling implementations of TTM.

We conclude that no scheme in the TPM class is secure.

1 Introduction

The research effort to bring further the practical public key cryptography introduced by R. Rivest, A. Shamir and L. Adleman, with univariate polynomials over $\mathbf{Z}/N\mathbf{Z}$, is following two paths. The first is considering more complex groups, e.g. elliptic curves. The second is considering multivariate equations. Many proposed schemes are being broken, some of them remain unbroken even for the simplest groups like $\mathbf{Z}/2\mathbf{Z}$.

One of the paradigms for constructing multivariate trapdoor cryptosystems is the triangular construction, proposed initially in an iterated form by H. Fell and W. Diffie (1985). It uses equations that involve 1, 2, ..., n variables and are solved sequentially. The special form of the equations is hidden by two linear transformations on inputs (variables) and outputs (equations).

We call T this triangular construction. Let TPM (T Plus-Minus) be T with added final random polynomials and with some of the beginning equations removed.

The cryptosystem TTM has been proposed by T.T. Moh at CrypTec'99. TTM, in spite of apparent complexity, proved to be a subcase of TPM design. After showing a trivial attack using linearities of initially proposed TTM, we focus on breaking more general TPM schemes.

Recovering the secret key of TPM/TTM leads to the following linear algebra problem called MinRank. Let M be a $n \times n$ matrix with entries being linear combinations of variables $\lambda_1, \ldots, \lambda_t$ over GF(q). The MinRank problem consists in determining whether there is such a valuation for $\lambda_1, \ldots, \lambda_t$ that Rank(M) $\leq r$. The weakness of MinRank instances in TTM lies in the fact that r is small ($r = 2$ in T.T. Moh's paper), while in general MinRank is NP-complete.

First we present an attack that works when q^r is small, exploiting the small co-dimension of the kernel of the unknown matrix. Our attacks break in approximately 2^{52} a cryptosystem with 512 bit cleartexts. In section 6, we present the solution (plaintext) to the TTM 2.1 challenge proposed by the company US Data Security, which is currently selling implementations of TTM. Finally, in section 5, we present an attack that works on TPM signature schemes when q^u is not too large and breaks TTM signature proposals [13, 14]. As a result, we conclude that no scheme in the TPM class is secure.

2 The TPM Family of Cryptosystems

2.1 General Description of TPM

In the present section, we describe the general family TPM(n,u,r,K), with:

- n, u, r integers such that $r \leq n$. We also systematically put $m = n + u - r$.
- $K = \mathrm{GF}(q)$ a finite field.

We first consider a function $\Psi : \ K^n \mapsto K^{n+u-r}$ such that $(y_1, \ldots, y_{n+u-r}) = \Psi(x_1, \ldots, x_n)$ is defined by the following system of equations:

$$\begin{cases} y_1 = x_1 + g_1(x_{n-r+1}, \ldots, x_n) \\ y_2 = x_2 + g_2(x_1 \ ; x_{n-r+1}, \ldots, x_n) \\ y_3 = x_3 + g_3(x_1, x_2 \ ; x_{n-r+1}, \ldots, x_n) \\ \quad \vdots \\ y_{n-r} = x_{n-r} + g_{n-r}(x_1, \ldots, x_{n-r-1} \ ; x_{n-r+1}, \ldots, x_n) \\ y_{n-r+1} = g_{n-r+1}(x_1, \ldots, x_n) \\ \quad \vdots \\ y_{n-r+u} = g_{n-r+u}(x_1, \ldots, x_n) \end{cases}$$

with each g_i $(1 \leq i \leq n + u - r)$ being a randomly chosen quadratic polynomial.

The Public Key

The user selects a random invertible affine transformation $s : \ K^n \mapsto K^n$, and a random invertible affine transformation $t : \ K^{n+u-r} \mapsto K^{n+u-r}$. Let $F = t \circ \Psi \circ s$. By construction, if we denote $(y'_1, \ldots, y'_{n+u-r}) = F(x'_1, \ldots, x'_n)$, we obtain an explicit set $\{P_1, \ldots, P_{n+u-r}\}$ of $(n+u-r)$ quadratic polynomials in n variables, such that:

$$\begin{cases} y'_1 = P_1(x'_1, \ldots, x'_n) \\ \quad \vdots \\ y'_{n+u-r} = P_{n+u-r}(x'_1, \ldots, x'_n) \end{cases}$$

This set of $(n+u-r)$ quadratic polynomials constitute the public key of this TPM(n,u,r,K) cryptosystem. Its size is $\frac{1}{8}(n+u-r)(n+1)(\frac{n}{2}+1)\log_2(q)$ bytes.

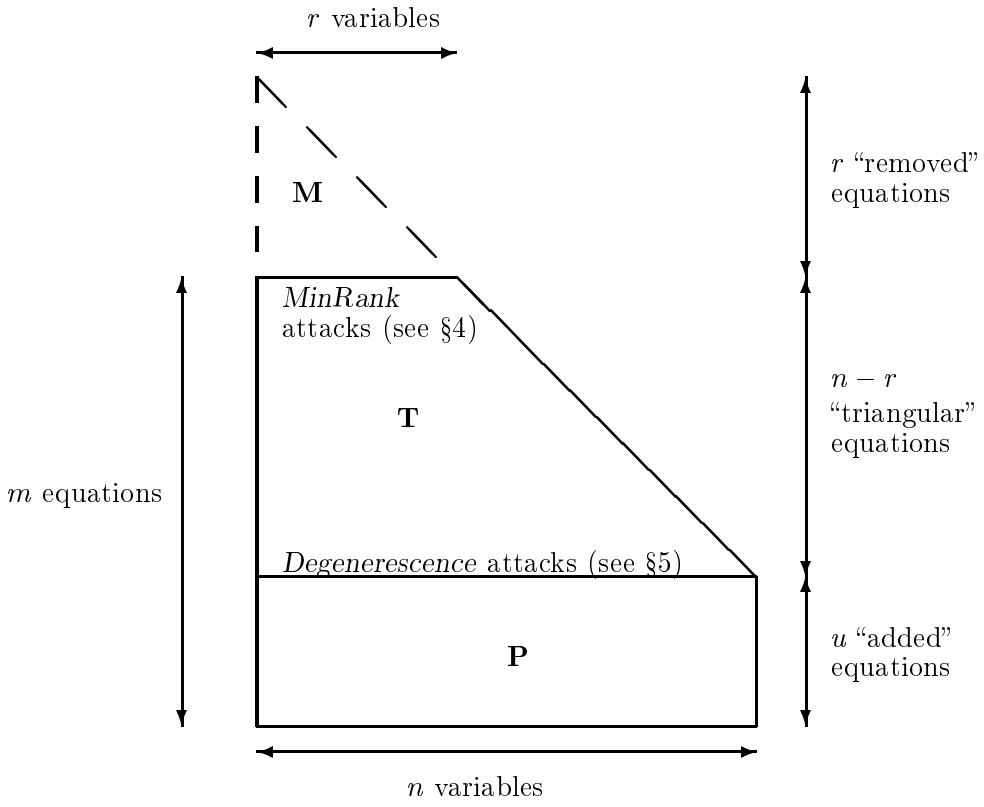


FIG. 1 – *General view of the TPM scheme – The two classes of attacks*

2.2 Encryption Protocol (when $u \geq r$)

Encrypting a message

Given a plaintext $(x'_1,\ldots,x'_n) \in K^n$, the sender computes $y'_i = P_i(x'_1,\ldots,x'_n)$ for $1 \leq i \leq n+u-r$ – thanks to the public key – and sends the ciphertext $(y'_1,\ldots,y'_{n+u-r}) \in K^{n+u-r}$.

Decrypting a message

Given a ciphertext $(y'_1,\ldots,y'_{n+u-r}) \in K^{n+u-r}$, the legitimate receiver recovers the plaintext by the following method.

- Compute $(y_1,\ldots,y_{n+u-r}) = t^{-1}(y'_1,\ldots,y'_{n+u-r})$;
- Make an exhaustive search on the r-tuple $(x_{n-r+1},\ldots,x_n) \in K^r$, until the n-tuple $(x_1,\ldots,x_n)$ obtained by $x_i = y_i - g_i(x_1,\ldots,x_{i-1};x_{n-r+1},\ldots,x_n)$ (for $1 \leq i \leq n-r$) satisfies the u following equations $g_i(x_1,\ldots,x_n) = y_i$ (for $n-r+1 \leq i \leq n-r+u$).
- For the obtained $(x_1,\ldots,x_n)$ n-tuple, get $(x'_1,\ldots,x'_n) = s^{-1}(x_1,\ldots,x_n)$.

This decryption algorithm thus has a complexity essentially $\mathcal{O}(q^r)$. As a result, a TPM(n,u,r,K) cryptosystem can be practically used in encryption mode only under the assumption that q^r is "small enough".

The condition $u \geq r$ insures that the probability of obtaining a collision is negligible, and thus that the ciphering function F can be viewed as an injection from K^n into K^{n+u-r}.

When $r = u = 0$, this kind of scheme was already considered and attacked by H. Fell and W. Diffie in [9] (in an iterated form) and by J. Patarin and the first author in [16]. All these authors heavily use the fact that the inverse function if of low degree on some of its variables. The goal of this paper is to extend these attacks to a much more general case, with r being non-zero (but q^r is not too large) and u is non-zero.

2.3 Signature Protocol (when $u \leq r$)

Signing a message

Given a message M, we suppose that $(y'_1,\ldots,y'_{n+u-r}) = h(M) \in K^{n+u-r}$, with h being a (collision-free) hash function. To sign the message M, the legitimate user:

- computes $(y_1,\ldots,y_{n+u-r}) = t^{-1}(y'_1,\ldots,y'_{n+u-r})$;
- chooses random r-tuples $(x_{n-r+1},\ldots,x_n)$, until the n-tuple $(x_1,\ldots,x_n)$ obtained by $x_i = y_i - g_i(x_1,\ldots,x_{i-1};x_{n-r+1},\ldots,x_n)$ (for $1 \leq i \leq n-r$) satisfies the u following equations $g_i(x_1,\ldots,x_n) = y_i$ (for $n-r+1 \leq i \leq n-r+u$).
- for the obtained $(x_1,\ldots,x_n)$ n-tuple, gets $(x'_1,\ldots,x'_n) = s^{-1}(x_1,\ldots,x_n)$.

This signature algorithm thus has a complexity essentially $\mathcal{O}(q^u)$. As a result, a TPM(n,u,r,K) cryptosystem can be practically used in signature mode only under the assumption that q^u is "small enough".

The condition $u \leq r$ insures that the probability of finding no solution for $(x_1,\ldots,x_n)$ for the equation $\Psi(x_1,\ldots,x_n) = (y_1,\ldots,y_{n+u-r})$ is negligible, and thus that the ciphering function F can be viewed as an surjection from K^n onto K^{n+u-r}.

We will describe in section 5 a general attack on this signature scheme, that is also applicable when u is non-zero, with q^u not too large. Therefore the signature proposed by T.T. Moh in [13, 14] is insecure.

2.4 The TTM encryption system

In the present section, we recall the original description of the TTM cryptosystem, given by T.T. Moh in [13, 14]. This definition of TTM is based on the concept of *tame automorphisms*. As we will see, TTM is a particular case of our general family TPM: it belongs to the family TPM(64,38,2,GF(256)).

General Principle

Let K be a finite field (which will be supposed "small" in real applications). We first consider two bijections Φ_2 and Φ_3 from K^{n+v} to K^{n+v}, with $(z_1,\ldots,z_{n+v}) = \Phi_2(x_1,\ \ldots,\ x_{n+v})$ and $(y_1,\ldots,y_{n+v}) = \Phi_3(z_1,\ldots,z_{n+v})$ defined by the two following systems of equations:

$$\Phi_2 : \begin{cases} z_1 = x_1 \\ z_2 = x_2 + f_2(x_1) \\ z_3 = x_3 + f_3(x_1,x_2) \\ \quad\vdots \\ z_n = x_n + f_n(x_1,\ldots,x_{n-1}) \\ z_{n+1} = x_{n+1} + f_{n+1}(x_1,\ldots,x_n) \\ \quad\vdots \\ z_{n+v} = x_{n+v} + f_{n+v}(x_1,\ldots,x_{n+v-1}) \end{cases} \qquad \Phi_3 : \begin{cases} y_1 = z_1 + P(z_{n+1},\ldots,z_{n+v}) \\ y_2 = z_2 + Q(z_{n+1},\ldots,z_{n+v}) \\ y_3 = z_3 \\ \quad\vdots \\ y_{n+v} = z_{n+v} \end{cases}$$

with $f_2,\ldots,f_{n+v}$ quadratic forms over K, and P, Q two polynomials of degree eight over K.

Φ_2 and Φ_3 are both "tame automorphisms" (see [13, 14] for a definition) and thus are one-to-one transformations. As a result, $(x_1,\ldots,x_{n+v}) \mapsto (y_1,\ldots,y_{n+v}) = \Phi_3 \circ \Phi_2(x_1,\ldots,x_{n+v})$ is also one-to-one and can be described by the following system of equations :

$$\begin{cases} y_1 = x_1 + P(x_{n+1} + f_{n+1}(x_1,\ldots,x_n),\ldots,x_{n+v} + f_{n+v}(x_1,\ldots,x_{n+v-1})) \\ y_2 = x_2 + f_2(x_1) + Q(x_{n+1} + f_{n+1}(x_1,..,x_n),..,x_{n+v} + f_{n+v}(x_1,..,x_{n+v-1})) \\ y_3 = x_3 + f_3(x_1,x_2) \\ \quad\vdots \\ y_n = x_n + f_n(x_1,\ldots,x_{n-1}) \\ y_{n+1} = x_{n+1} + f_{n+1}(x_1,\ldots,x_n) \\ \quad\vdots \\ y_{n+v} = x_{n+v} + f_{n+v}(x_1,\ldots,x_{n+v-1}) \end{cases}$$

T.T. Moh found a clever way of choosing P, Q and f_i such that y_1 and y_2 both become *quadratic* functions of $x_1,\ldots,x_n$ when we set $x_{n+1} = \ldots = x_{n+v} = 0$.

Actual Parameters

This paragraph is given in the appendix. T.T. Moh chooses $n = 64$, $v = 36$ and $K = \mathrm{GF}(256)$. As a result, TTM belongs to TPM(64,38,2,GF(256)). Applying the formula of section 2.1, the size of the public keys is 214.5 Ko.

3 General Strategy for an Attack on TPM

In the present section, we describe a general strategy to attack a cryptosystem of the TPM Family, when r is "small". As a result TTM, which belongs to TPM(64,38,2,GF(256)) is threatened by such attacks.

3.1 The MinRank Problem

Let r be an integer and K a field. We denote by MinRank(r) the following problem: given a set $\{M_1,\ldots,M_m\}$ of $n \times n$ matrices whose coefficients lie in K, find at least one m-tuple $(\lambda_1,\ldots,\lambda_m) \in K^m$ such that $\mathrm{Rank}\Big(\sum_{i=1}^{m} \lambda_i M_i\Big) \leq r$.

The MinRank problem was defined and studied in [17] by Shallit, Frandsen and Buss. MinRank generalizes the "Rank Distance Coding" problem (introduced by E. Gabidulin in [10], and considered in [3, 20]), which itself generalizes the "Minimal Weight" problem in error correcting codes (see [1, 19, 2, 11]). In [12], A. Kipnis and A. Shamir exposed a strategy to attack the HFE cryptosystem (invented by J. Patarin, see [15]). They had to face an instance of MinRank(r) with $r = \lceil \log_q n \rceil + 1$. For that purpose, they introduced the "relinearization technique". But their attack was still not polynomial, unlike here. Note that the idea of finding small ranks was also used by D. Coppersmith, J. Stern and S. Vaudenay (see [6, 7]) in their cryptanalysis of a scheme proposed by A. Shamir in [18].

3.2 Complexity of MinRank

The general MinRank problem has been proven to be NP-complete by Shallit, Frandsen and Buss (see [17]). More precisely, they prove that MinRank(r) NP-complete when $r = n - 1$ (this

corresponds to the problem of finding a linear combination of $M_1, \ldots, M_m$ that is singular). The principle of their proof consists in writing any set of multivariate equations as an instance of MinRank. It can be used in the same way to extend their result to the cases $r = n-2$, $r = n-3$, ... and even $r = n^\alpha$ (when $\alpha > 0$ is fixed). However, there is no such complexity result for smaller values of r. Indeed, as we will see in the following sections, polynomial algorithms can be described to solve the MinRank problem when r is fixed.

3.3 Strategy of attack

We recall that $m = n + u - r$. We suppose $m \leq 2n$, as an encryption function with expansion rate > 2 is unacceptable. Moreover, if $m > \mathcal{O}(n)$, the cryptosystem would be broken by Gröbner bases [8].

In each equation $y_i = x_i + g_i(x_1, \ldots, x_{i-1}\,; x_{n-r+1}, \ldots, x_n)$ $(1 \leq i \leq n-r)$, the homogeneous part is given by ${}^{\mathrm{t}}XA_iX$, with ${}^{\mathrm{t}}X = (x_1, \ldots, x_n)$, A_i being a (secret) matrix. Similarly, in each public equation $y'_i = P_i(x'_1, \ldots, x'_n)$ is given by ${}^{\mathrm{t}}X'M_iX'$, with ${}^{\mathrm{t}}X' = (x'_1, \ldots, x'_n)$, M_i being a (public) matrix.

The fact that $(x_1, \ldots, x_n) = s(x'_1, \ldots, x'_n)$ and $(y'_1, \ldots, y'_m) = t(y_1, \ldots, y_m)$ implies that there exist an invertible $n \times n$ matrix S and an invertible $m \times m$ matrix T such that:

$$\begin{pmatrix} {}^{\mathrm{t}}(SX')A_1(SX') \\ \vdots \\ {}^{\mathrm{t}}(SX')A_m(SX') \end{pmatrix} = T^{-1} \begin{pmatrix} {}^{\mathrm{t}}X'M_1X' \\ \vdots \\ {}^{\mathrm{t}}X'M_mX' \end{pmatrix}.$$

Let $T^{-1} = (t_{ij})_{1 \leq i,j \leq m}$. We thus have, for any X':

$${}^{\mathrm{t}}X'({}^{\mathrm{t}}SA_iS)X' = {}^{\mathrm{t}}X'\Big(\sum_{j=1}^{m} t_{ij}M_j\Big)X'$$

so that:

$$\forall i,\ 1 \leq i \leq m,\ \sum_{j=1}^{m} t_{ij}M_j = {}^{\mathrm{t}}SA_iS.$$

From the construction of TPM(n,u,r,K), we have $\mathrm{Rank}(A_1) \leq r$. Since S is an invertible matrix, we have $\mathrm{Rank}(A_1) = \mathrm{Rank}({}^{\mathrm{t}}SA_1S)$ and thus $\mathrm{Rank}\Big(\sum_{j=1}^{m} t_{1j}M_j\Big) \leq r$, that is precisely an instance of MinRank(r).

Suppose we are able to find (at least) one m-tuple $(\lambda_1, \ldots, \lambda_m)$ such that $\mathrm{Rank}\Big(\sum_{j=1}^{m} \lambda_jM_j\Big) \leq r$. With a good probability, we can suppose that:

$$\sum_{j=1}^{m} \lambda_jM_j = \mu\,{}^{\mathrm{t}}SA_1S \qquad (\mu \in K^*).$$

Then we deduce the vector spaces $V_0 = S^{-1}(K^{n-r} \times \{0\}^r)$ (corresponding to $x_{n-r+1} = \ldots = x_n = 0$) and $W_0 = S^{-1}(\{0\}^{n-r} \times K^r)$ (corresponding to $x_1 = \ldots = x_{n-r} = 0$) by simply noticing that $V_0 = \mathrm{Im}\Big(\sum_{j=1}^{m} \lambda_jM_jA_1\Big)$ and $W_0 = \mathrm{Ker}\Big(\sum_{j=1}^{m} \lambda_jM_jA_1\Big)$.

Once we have found V_0 and W_0, we can easily deduce the vector space $V_1 = S^{-1}(\{0\} \times K^{n-r-1} \times \{0\}^r)$ of dimension 1 (corresponding to $x_1 = x_{n-r+1} = \ldots = x_n = 0$) and $W_1 =$

$S^{-1}(K \times \{0\}^{n-r-1} \times K^r)$ (corresponding to $x_2 = \ldots = x_{n-r} = 0$): we just look for coefficients $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m$ such that the following equation:

$$\sum_{j=1}^{m} \beta_j y'_j = \sum_{i=1}^{n} \alpha_i x_i + \delta,$$

holds for any element of V_0. This can be obtained by simple Gaussian reduction. We also obtain the g_2 quadratic function by Gaussian reduction.

By repeating these steps, we obtain two sequences of vector spaces:

$$V_0 \supseteq V_1 \supseteq V_2 \supseteq \ldots \supseteq V_{n-r-1}$$

$$W_0 \subseteq W_1 \subseteq W_2 \subseteq \ldots \subseteq W_{n-r-1}.$$

At the end, we have completely determined the secret transformations s and t, together with the secret functions g_i. As a result, this algorithm completely breaks the TPM family of cryptosystems (we recovered the secret key).

4 Two attacks on MinRank and TPM

In the previous section, we proved that breaking the TPM family of cryptosystems is easy if we can solve the MinRank(r) problem. This is the point we are interested in, in the present section. Two algorithms will be described.

4.1 The 'Linearity Attack' on MinRank and TTM

In this paragraph, we study the particular case of TTM, as described by T.T. Moh in [13, 14]. In this case, we show that the MinRank(r) problem is easily solved, because of the particular structure of the Q_8 function used in Φ_3.

Description of the Attack

In section 3.3, we proved that an attack can be successfully performed on this cryptosystem, as soon as we can find out the vector spaces $V_0 = S^{-1}(\{0\}^2 \times K^{62})$ (corresponding to $x_1 = x_2 = 0$) and $W_0 = S^{-1}(K^2 \times \{0\}^{62})$ (corresponding to $x_3 = \ldots = x_{64} = 0$). At first sight, the equations giving y_1 and y_2 seem to be quadratic in $(x_1, \ldots, x_{64})$. This leads *a priori* to an instance of MinRank(2).

However, note that the function $x \mapsto x^2$ is linear on $K = \mathrm{GF}(256)$, considered as a vector space of dimension 8 over $F = \mathrm{GF}(2)$. Therefore, considering the equations describing the (secret) Ψ function of TTM[1], if we choose a basis $(\omega_1, \ldots, \omega_8)$ of K over F and write $x_i = x_{i,1}\omega_1 + \ldots + x_{i,8}\omega_8$ $(1 \leq i \leq 64)$, y_1 and y_2 become linear functions of $x_{1,1}, x_{1,2}, \ldots, x_{1,8}, \ldots, x_{64,1}, \ldots, x_{64,8}$. In terms of MinRank, this means that TTM leads to an instance of MinRank(0) for $8n \times 8n$ matrices (instead of an instance of MinRank(2) for $n \times n$ matrices). This leads to the following attack on TTM:

1. Let $x'_i = x'_{i,1}\omega_1 + \ldots + x'_{i,8}\omega_8$ $(1 \leq i \leq 64)$. Rewrite each public equation $y'_i = P_i(x'_1, \ldots, x'_{64})$ as $y'_i = \tilde{P}_i(x'_{1,1}, \ldots, x'_{64,8})$ (with $\tilde{P}_i$ a quadratic polynomial in $64 \times 8 = 512$ variables over $F = \mathrm{GF}(2)$).

1. See (E) in the appendix, in which t_{19} is a linear transformation.

2. Find the vector space of the 612-tuples $(\beta_1,\ldots,\beta_{100},\alpha_{1,1},\ldots,\alpha_{64,8}) \in K^{612}$ satisfying:

$$\sum_{i=1}^{100} \beta_i y'_i = \sum_{i=1}^{64}\sum_{j=1}^{8} \alpha_{i,j} x'_{i,j}.$$

   This can be done by Gaussian reduction. We thus obtain the vector spaces V_0 and W_0 defined above.
3. The remaining part of the attack is exactly the same as in section 3.3.

Complexity of the Attack
The main part of the algorithm consists in solving a system of linear equations on 612 variables, by Gaussian reduction. We thus obtain a complexity of approximately 2^{28} elementary operations to break TTM.

4.2 The 'Kernel Attack' on MinRank and TPM

We describe here a new attack on MinRank(r), which works when q^r is small enough.

Description of the Attack (with the same notations as in section 3.3)

1. Choose k random vectors $X'^{[1]},\ldots,X'^{[k]}$ (with k an integer depending on n and m, that we define below). Since $\dim \mathrm{Ker}({}^tSA_1S) = n - \mathrm{Rank}({}^tSA_1S) \geq n-r$, we have the simultaneous conditions $X'^{[i]} \in \mathrm{Ker}({}^tSA_iS)$ $(1 \leq i \leq k)$ with a probability $\geq q^{-kr}$.
2. We suppose we have chosen a "good" set $\{X'^{[1]},\ldots,X'^{[k]}\}$ of k vectors (*i.e.* such that they all belong to $\mathrm{Ker}({}^tSA_1S)$). Then we can find an m-tuple $(\lambda_1,\ldots,\lambda_m)$ such that, for all i, $1 \leq i \leq k$, $\Big(\sum_{j=1}^{m} \lambda_j M_j\Big)(X'^{[i]}) = 0$. They are solution of a system of kn linear equations in m indeterminates. As a result, if we let $k = \lceil \frac{m}{n} \rceil$, the solution is essentially unique and can be easily found by Gaussian reduction. We thus obtain the two vector spaces $V_0 = S^{-1}(K^{n-r} \times \{0\}^r)$ (corresponding to $x_{n-r+1} = \ldots = x_n = 0$) and $W_0 = S^{-1}(\{0\}^{n-r} \times K^r)$ (corresponding to $x_1 = \ldots = x_{n-r} = 0$).
3. The remaining part of the attack is exactly the same as in section 3.3.

Complexity of the Attack
From the description of the attack, its complexity is easily seen to be $\mathcal{O}(q^{\lceil \frac{m}{n} \rceil r} \cdot m^3)$.

Application to TTM
In the particular case of TTM, we have $q = 256$, $n = 64$, $m = 100$ and $r = 2$.
We thus obtain an attack on TTM in complexity $\mathcal{O}(2^{52})$.

Note: Compared to the 2^{28} of section 4.1, this attack is slower, but it does not make use of any linearity of y_1 and y_2, so that it can also be used to break possible generalizations of TTM, with more general "Q_8 components" (see [4] for examples of Q_8 which provide non linear expressions for y_1 and y_2 over GF(2)).

5 The 'Degenerescence Attack' on TPM signature schemes

We describe here a general attack on TMP signature schemes (recall that such schemes are possible only for $u \leq r$), when q^u is not too large. From the description of the attack, its complexity is easily seen to be $\mathcal{O}(q^u \cdot n^6)$. We use the same notations as in section 3.3. In particular, $m = n + u - r$.

1. We choose a random m-tuple $(\beta_1, \ldots, \beta_m) \in K^m$. With a probability q^{-u-1}, we can suppose that $\beta_i P_i$ is a degenerated quadratic polynomial (*i.e.* a quadratic polynomial which can be rewritten with fewer variables after a linear change of variables). The fact that a quadratic polynomial is degenerated can easily be detected: for instance by using its canonical form (see [16] for some other methods).
2. Suppose we have found a "good" m-tuple $(\beta_1, \ldots, \beta_m)$. Considering the new set of $(< n)$ variables for the quadratic form $\sum_{i=1}^{m} \beta_i P_i$, we deduce easily the vector space $W_{n-r} = S^{-1}(K^{n-r-1} \times \{0\} \times K^r)$.
3. Then we look for a n-tuple $(\alpha_1, \ldots, \alpha_n) \in K^n$ and a quadratic function g^{n-r}, such that:

$$\sum_{i=1}^{m} \beta_i y'_i = \sum_{i=1}^{n} \alpha_i x'_i + g_{n-r}(x'_1, \ldots, x'_n)$$

   is true for any $(x'_1, \ldots, x'_n) \in W_{n-r}$. This can be done by Gaussian reduction. We thus obtain the vector space $V_{n-r} = S^{-1}(\{0\}^{n-r-1} \times K \times \{0\}^r)$ and the quadratic polynomial g_{n-r}.
4. The same principle can be repeated $n - r$ times, so as to obtain two sequences of vector spaces:

$$V_{n-r} \subseteq V_{n-r-1} \subseteq \ldots \subseteq V_0$$

$$W_{n-r} \supseteq W_{n-r-1} \supseteq \ldots \supseteq W_0.$$

   At the end, as in the attack described in section 3.3, we have completely determined the secret transformations s and t, together with the secret functions g_i. As a result, this algorithm completely breaks the TPM family in signature mode (we recovered the secret key).

6 Solution to the TTM 2.1 Challenge of US Data Security

In 1997, US Data Security published challenges about TTM (see [21]). They are based on three different versions of TTM in encryption mode, corresponding to different choices of the parameters.

On May 2nd, 2000, we managed to break the second challenge, based on TTM 2.1. As mentioned in [21], "the public-key TTM 2.1 is a block cipher with plaintext block size 64 and ciphertext block size 100. It works on 8 bits finite field (*i.e.*, characters)". The public key can be obtained by approximately 2000 queries to the "encryption oracle". As mentioned in section 2.4, its size is 214.5 Kbytes. By using the general method described in section 4.1, we obtained the following plaintext, which can be easily checked to be the exact solution to this "Contest II" (note that the quotation marks are part of this plaintext):

"Tao TTP way BCKP of living hui mountain wen river moon love pt"

7 Conclusion

We cryptanalysed a large class of cryptosystems TPM, that includes TTM as it has been described by T.T. Moh [14]. They can be broken in polynomial time, as long as r is fixed. The proposed TTM cryptosystem [14] can be broken in 2^{28}. As an application of our general method, we broke the "TTM 2.1" challenge proposed by US Data Security in October 1997. Even if Q_8 is nonlinear, and since $r = 2$, it is still broken in 2^{52} elementary operations for a 512-bit cryptosystem. There is very little hope that a secure triangular system will ever be proposed.

References

[1] E.R. Berlekamp, R.J. McEliece, H.C.A. Van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Transactions on Information Theory, IT-24(3), pp. 384-386, May 1978.

[2] F. Chabaud, *Asymptotic analysis of probabilistic algorithms for finding short codewords*, in Proceedings of EUROCODE'92, Udine, Italy, CISM Courses and lectures n° 339, Springer-Verlag, 1993, pp. 217-228.

[3] K. Chen, *A new identification algorithm*, Cryptography Policy and Algorithms Conference, LNCS n° 1029, Springer-Verlag, 1996.

[4] C. Y. Chou, D. J. Guan, J. M. Chen, *A systematic construction of a Q_{2^k}-module in TTM*, Preprint, October 1999. Available at http://www.usdsi.com/chou.ps

[5] D. Coppersmith, S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Computation (1990), **9**, pp. 251-280.

[6] D. Coppersmith, J. Stern, S. Vaudenay, *Attacks on the Birational Permutation Signature Schemes*, in Advances in Cryptology, Proceedings of Crypto'93, LNCS n° 773, Springer-Verlag, 1993, pp. 435-443.

[7] D. Coppersmith, J. Stern, S. Vaudenay, *The Security of the Birational Permutation Signature Schemes*, in Journal of Cryptology, 10(3), pp. 207-221, 1997.

[8] N. Courtois, A. Shamir, J. Patarin, A. Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, in Advances in Cryptology, Proceedings of EUROCRYPT'2000, LNCS n° 1807, Springer, 2000, pp. 392-407.

[9] H. Fell, W. Diffie, *Analysis of a public key approach based on polynomial substitutions*, in Advances in Cryptology, Proceedings of CRYPTO'85, LNCS n° 218, Springer-Verlag, 1985, pp. 340-349.

[10] E.M. Gabidulin, *Theory of codes with maximum rank distance*, Problems of Information Transmission, 21:1-12, 1985.

[11] S. Harari, *A new authentication algorithm*, in Coding Theory and Applications, LNCS n° 388, Springer, 1989, pp. 204-211.

[12] A. Kipnis, A. Shamir, *Cryptanalysis of the HFE public key cryptosystem*, in Advances in Cryptology, Proceedings of Crypto'99, LNCS n° 1666, Springer, 1999, pp. 19-30.

[13] T.T. Moh, *A public key system with signature and master key functions*, Communications in Algebra, 27(5), pp. 2207-2222, 1999. Available at http://www.usdsi.com/public.ps

[14] T.T. Moh, *A fast public key system with signature and master key functions*, in Proceedings of CrypTEC'99, International Workshop on Cryptographic Techniques and E-commerce, Hong-Kong City University Press, pp. 63-69, July 1999. Available at http://www.usdsi.com/cryptec.ps

[15] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms*, in Advances in Cryptology, Proceedings of EUROCRYPT'96, LNCS n° 1070, Springer Verlag, 1996, pp. 33-48.

[16] J. Patarin, L. Goubin, *Asymmetric cryptography with S-Boxes*, in Proceedings of ICICS'97, LNCS n° 1334, Springer, 1997, pp. 369-380.

[17] J.O. Shallit, G.S. Frandsen, J.F. Buss, *The computational complexity of some problems of linear algebra*, BRICS series report, Aarhus, Denmark, RS-96-33. Available at http://www.brics.dk/RS/96/33

[18] A. Shamir, *Efficient Signature Schemes based on Birational Permutations*, in Advances in Cryptology, Proceedings of Crypto'93, LNCS n° 773, Springer-Verlag, 1993, pp. 1-12.

[19] J. Stern, *A new identification scheme based on syndrome decoding*, in Advances in Cryptology, Proceedings of CRYPTO'93, LNCS n° 773, Springer-Verlag, 1993, pp. 13-21.

[20] J. Stern, F. Chabaud, *The cryptographic security of the Syndrome Decoding problem for rank distance codes*, in Advances in Cryptology, Proceedings of ASIACRYPT'96, LNCS n° 1163, Springer-Verlag, 1985, pp. 368-381.

[21] *The US Data Security Public-Key Contest*, available at http://www.usdsi.com/contests.html

Appendix: Actual Parameters for the TTM Cryptosystem

Let Q_8 be the function defined by

$$Q_8(q_1,\ldots,q_{30}) = q_1^8 + q_{29}^4 + q_{30}^2 + [q_2^4 + q_3^2 q_8^2 + q_4^2 q_5^2 + q_6^2 q_{12}^2 + q_7^2 q_{13}^2]$$

$$\times[q_9^4 + (q_{10}^2 + q_{14}q_{15} + q_{18}q_{19} + q_{20}q_{21} + q_{22}q_{24})(q_{11}^2 + q_{16}q_{17} + q_{23}q_{28} + q_{25}q_{26} + q_{13}q_{27})].$$

A straightforward computation gives $Q_8(q_1,\ldots,q_{30}) = t_{19}^2$ as soon as we make the following choices for $q_1,\ldots,q_{30}$:

$q_1 = t_1 + t_2 t_6$	$q_2 = t_2^2 + t_3 t_7$	$q_3 = t_3^2 + t_4 t_{10}$	$q_4 = t_3 t_5$
$q_5 = t_3 t_{11}$	$q_6 = t_4 t_7$	$q_7 = t_4 t_5$	$q_8 = t_7^2 + t_5 t_{11}$
$q_9 = t_6^2 + t_8 t_9$	$q_{10} = t_8^2 + t_{12} t_{13}$	$q_{11} = t_9^2 + t_{14} t_{15}$	$q_{12} = t_7 t_{10}$
$q_{13} = t_{10} t_{11}$	$q_{14} = t_{12}^2 + t_7 t_8$	$q_{15} = t_{13}^2 + t_{11} t_{16}$	$q_{16} = t_{14}^2 + t_{10} t_{12}$
$q_{17} = t_{15}^2 + t_{11} t_{17}$	$q_{18} = t_{12} t_{16}$	$q_{19} = t_{11} t_{12}$	$q_{20} = t_8 t_{13}$
$q_{21} = t_7 t_{13}$	$q_{22} = t_8 t_{16}$	$q_{23} = t_{14} t_{17}$	$q_{24} = t_7 t_{11}$
$q_{25} = t_{12} t_{15}$	$q_{26} = t_{10} t_{15}$	$q_{27} = t_{12} t_{17}$	$q_{28} = t_{11} t_{14}$
$q_{29} = t_{18} + t_1^2$	$q_{30} = t_{19} + t_{18}^2$		

We choose $n = 64$, $v = 36$, and we consider the $t_i = t_i(u_1,\ldots,u_{19})$ $(1 \leq i \leq 19)$ as randomly chosen linear forms (*i.e.* homogeneous polynomials of degree one in $u_1,\ldots,u_{19}$), satisfying the following conditions:

- $t_1(u_1,\ldots,u_{19}) = u_1$;
- $t_{18}(u_1,\ldots,u_{19}) = u_{18}$;
- $t_{19}(u_1,\ldots,u_{19}) = u_{19}$;
- $t_6(u_1,\ldots,u_{19})$, $t_7(u_1,\ldots,u_{19})$, $t_{18}(u_1,\ldots,u_{19})$ and $t_{19}(u_1,\ldots,u_{19})$ depend only on the variables $u_6, u_7, \ldots, u_{17}$,

We thus obtain polynomials $q_i = q_i(u_1,\ldots,u_{19})$ $(1 \le i \le 30)$ of degree two in $u_1,\ldots,u_{19}$. Finally, we choose:

$$\begin{cases} P(z_{65},\ldots,z_{100}) = Q_8(z_{93},\ldots,z_{100},z_{73},\ldots,z_{92},z_{63},z_{64}) \\ Q(z_{65},\ldots,z_{100}) = Q_8(z_{65},\ldots,z_{92},z_{61},z_{62}) \\ f_{61}(x_1,\ldots,x_{60}) = q_{29}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) - x_{61} \\ f_{62}(x_1,\ldots,x_{61}) = q_{30}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) - x_{62} \\ f_{63}(x_1,\ldots,x_{62}) = q_{29}(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) - x_{63} \\ f_{64}(x_1,\ldots,x_{63}) = q_{30}(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) - x_{64} \\ f_{65}(x_1,\ldots,x_{64}) = q_1(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \\ \quad\vdots \\ f_{92}(x_1,\ldots,x_{91}) = q_{28}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \\ f_{93}(x_1,\ldots,x_{92}) = q_1(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \\ \quad\vdots \\ f_{100}(x_1,\ldots,x_{99}) = q_8(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \end{cases}$$

and randomly chosen quadratic forms for f_i $(2 \le i \le 60)$.

Let us denote $\theta : K^{64} \to K^{100}$ the function defined by

$$\theta(x_1,\ldots,x_{64}) = (x_1,\ldots,x_{64},0,\ldots,0).$$

Hence $(x_1,\ldots,x_{64}) \mapsto (y_1,\ldots,y_{100}) = \Phi_3 \circ \Phi_2 \circ \theta(x_1,\ldots,x_{64})$ is given by the following system:

$$(E)\begin{cases} y_1 = x_1 + [t_{19}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62})]^2 \qquad (= x_1 + x_{62}^2) \\ y_2 = x_2 + f_2(x_1) + [t_{19}(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64})]^2 \\ \qquad\qquad (= x_2 + f_2(x_1) + x_{64}^2) \\ y_3 = x_3 + f_3(x_1,x_2) \\ \quad\vdots \\ y_{60} = x_{60} + f_{60}(x_1,\ldots,x_{59}) \\ y_{61} = q_{29}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \qquad (= x_{61} + x_9^2) \\ y_{62} = q_{30}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \qquad (= x_{62} + x_{61}^2) \\ y_{63} = q_{29}(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \quad (= x_{63} + x_{10}^2) \\ y_{64} = q_{30}(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \quad (= x_{64} + x_{63}^2) \\ y_{65} = q_1(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \\ \quad\vdots \\ y_{92} = q_{28}(x_9,x_{11},\ldots,x_{16},x_{51},\ldots,x_{62}) \\ y_{93} = q_1(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \\ \quad\vdots \\ y_{100} = q_8(x_{10},x_{17},\ldots,x_{20},x_{15},x_{16},x_{51},\ldots,x_{60},x_{63},x_{64}) \end{cases}$$

The Public Key

The user selects a random invertible affine transformation $\Phi_1 : K^{64} \to K^{64}$, and a random invertible affine transformation $\Phi_4 : K^{100} \to K^{100}$, such that the function $F = \Phi_4 \circ \Phi_3 \circ \Phi_2 \circ \theta \circ \Phi_1$ satisfies

$$F(0,\ldots,0) = (0,\ldots,0).$$

By construction of F, if we denote $(y'_1,\ldots,y'_{100}) = F(x'_1,\ldots,x'_{64})$, then we have an explicit set $\{P_1,\ldots,P_{100}\}$ of 100 quadratic polynomials in 64 variables, such that:

$$\begin{cases} y'_1 = P_1(x'_1,\ldots,x'_{64}) \\ \quad\vdots \\ y'_{100} = P_{100}(x'_1,\ldots,x'_{64}) \end{cases}$$

This set of 100 polynomials constitutes the public key of the TTM cryptosystem.

Encrypting a message

Given a plaintext $(x'_1,\ldots,x'_{64}) \in K^{64}$, the sender computes $y'_i = P_i(x'_1,\ldots,x'_{64})$ for $1 \leq i \leq 100$ (thanks to the public key) and sends the ciphertext $(y'_1,\ldots,y'_{100})$.

Decrypting a message

Given a ciphertext $(y'_1,\ldots,y'_{100}) \in K^{100}$, the legitimate receiver recovers the plaintext by:

$$(x'_1,\ldots,x'_{64}) = {\Phi_1}^{-1} \circ \pi \circ {\Phi_2}^{-1} \circ {\Phi_3}^{-1} \circ {\Phi_3}^{-1} \circ {\Phi_4}^{-1}(y'_1,\ldots,y'_{100})$$

with $\pi : K^{100} \mapsto K^{64}$ defined by $\pi(x_1,\ldots,x_{100}) = (x_1,\ldots,x_{64})$ and thus satisfies $\pi \circ \theta = \mathrm{Id}$.

Solving Underdefined Systems of Multivariate Quadratic Equations

PKC'2002
Article avec Nicolas Courtois (Schlumberger),
Willi Meier et Jean-Daniel Tacier (FH Aargau, Suisse)

Abstract

The security of several recent digital signature schemes is based on the difficulty of solving large systems of quadratic multivariate polynomial equations over a finite field $\mathbf{F}$. This problem, sometimes called MQ, is known to be NP-hard. When the number m of equations is equal to the number n of variables, and if $n < 15$, Gröbner base algorithms have been applied to solve MQ. In the overdefined case $n \ll m$, the techniques of relinearization and XL, due to A. Shamir et. al., have shown to be successful for solving MQ. In signature schemes, we usually have $n \gg m$. For example signature schemes Flash and Sflash submitted to Nessie call for primitives or the UOV scheme published at Eurocrypt 1999. Little is known about the security of such underdefined systems.

In this paper, three new and different methods are presented for solving underdefined multivariate systems of quadratic equations. As already shown at Eurocrypt 1999, the problem MQ becomes polynomial when $n \geq m(m+1)$ for fields $\mathbf{F}$ of characteristic 2. We show that for any field, for about $n \geq 2^{m/7}(m+1)$, exponential but quite small in practice, the problem becomes polynomial in n.

When $n \to m$ the complexity of all our 3 algorithms tends to q^m. However for practical instances of cryptosystems with $n \approx \mathcal{O}(m)$, we show how to achieve complexities significantly lower than exhaustive search. For example we are able to break unbalanced Oil and Vinegar signature schemes for two practical sets of parameters.

1 Introduction

Since the 1970's many digital signature schemes have been proposed and the best are probably those based on factoring or discrete logarithms in well chosen groups. However in many specific applications the classical standardized schemes are either too slow, or give signatures that are too long. In order to fill the gap, several new digital signature schemes based on multivariate polynomials have been studied recently J. Patarin et. al. hoping to do better. These signature schemes can be very efficient in smart card implementations and are among the shortest signature schemes ever known [9]. They differ in the type of trapdoor structure embedded into the public polynomials, see [5] for a particular example of such a scheme and for an overview of various other schemes. Several of these schemes were broken soon after being proposed, for others the security is an open problem.

Most multivariate schemes rely on the problem of solving systems of multivariate polynomial modular equations over a small finite field $\mathbf{F}$. The general problem is called MQ and is known to be NP-complete, even for $q = 2$, cf. [2]. In practice, the public key will be a system of m quadratic polynomials $G_i(x_1,...,x_n)$, $i = 1,...,m$, with n variables x_j, $j = 1,...,n$ over a finite field $\mathbf{F}$ of (small) order q, where m and n are (large enough) integers. Messages are represented as elements of the vector space $\mathbf{F}^m$, whereas signatures are elements of $\mathbf{F}^n$. An element x is a signature of a message y if the public polynomials evaluated at x give the correct components of y, i.e., if $G_i(x_1,...,x_n) = y_i$ for $i = 1,...,m$. The trapdoor information enables a legitimate signer to find a solution x of the system for a given message y.

Several multivariate signature schemes use MQ systems of m quadratic equations in n variables with $n \gg m$. For example cryptosystems Flash and Sflash submitted to Nessie call for primitives [8] or the Unbalanced Oil and Vinegar scheme (UOV) [5, 6].

As opposed to the case $m \gg n$, where useful methods for solving MQ are known (see [4], [7]), only one result seems to be known for finding a solution if $n \gg m$: In the massively underdefined case $n \geq m(m+1)$, a polynomial algorithm for solving MQ is given in [5], provided the field $\mathbf{F}$ has characteristic 2.

In this paper, three different methods are developed for solving MQ in the underdefined case $n \gg m$ (algorithms A, B and C). We also generalize the known algorithm from [5] that solves in polynomial time massively underdefined systems of equations over fields of characteristic 2. We show that it can be extended to odd characteristics. The present version works for roughly about $n \geq 2^{m/7}(m+1)$, exponential but in practice not so big.

The three algorithms A, B and C can also be applied for practical systems when $n = \mathcal{O}(m)$. Depending on the choice of m, n and q, one or the other of the algorithms as presented turns out to be more efficient, and no one outperforms the others in general. Their complexity remain exponential, but each of the 3 algorithms enables solving MQ with a complexity significantly lower than exhaustive search for some practical parameter sets.

The three algorithms apply different but well known principles: The birthday paradox, a linearization technique, and reduced representations of quadratic forms. These principles are used however in a novel way. By studying and comparing the efficiency of different algorithms for the same problem, we attempt to get a better understanding of the difficulty of MQ. As a cryptographic application, we obtain new criteria for the security parameters of general multivariate signature schemes. The security of schemes like Flash and Sflash is not affected by the results of this paper. However our algorithms can be used to break unbalanced Oil and Vinegar signature schemes (cf. [5]) for two sets of practical parameters proposed in [6], for which no attacks were previously known.

In section 2 the problem MQ is described. In section 3, algorithms A, B and C for solving MQ in the underdefined case $n \gg m$ are presented, and their efficiency is studied. Section 4 presents efficient algorithms for solving massively underdefined MQ systems. In section 5, our methods are applied to the cryptanalysis of practical multivariate signature schemes. In Appendix A, a result on which algorithm C is based upon is derived.

2 The Problem MQ

Let $\mathbf{F}$ be a finite field of order q. Consider a (random) system of m simultaneous equations in n variables over $\mathbf{F}$,

$$G_l(x_1,x_2,...,x_n) = y_l, \quad l = 1,...,m,$$

where the G_l are (not necessarily homogeneous) polynomials of degree two, i.e., G_l is of the form

$$G_l(x_1,...,x_n) = \sum_{i=1}^{n}\sum_{j\geq i}^{n} \alpha_{ijl} x_i x_j + \sum_{k=1}^{n} \gamma_{kl} x_k,\ l = 1,...,m.$$

Hereby the coefficients of the polynomials as well as the components of the vectors $x = (x_1,x_2,...,x_n)$ and $y = (y_1,y_2,...,y_m)$ are elements in $\mathbf{F}$. Depending on the parameters m and n, several algorithms have been developed to solve MQ: If $m = n < 15$, Gröbner bases algorithms can be applied to solve MQ (see [1] for one of the most efficient variants amongst such algorithms). In the overdefined case $m \gg n$, the methods of relinearization and XL have shown to be useful (see [4] and [7]). In particular, in the overdefined case $m = \varepsilon n^2$, the XL algorithm is expected to be polynomial. In the massively underdefined case $n \geq m(m+1)$, a polynomial algorithm has been developed in [5] to solve MQ in case $\mathbf{F}$ has characteristic 2.

3 Solving MQ for underdefined systems of equations

Suppose a given system of quadratic equations is underdefined, i.e., $n \gg m$. Several algorithms are presented in this section for solving MQ faster than by exhaustive search when $n \gg m$. Our goal is to find one among about q^{n-m} expected solutions.

The complexity of our algorithms will be compared to the complexity of exhaustive search that is about $\mathcal{O}(q^m)$. Thus a natural choice for an elementary operation is a numeric evaluation of all m quadratic polynomials G_l for a single set of values of $x_1,...,x_n$. If we want otherwise measure the complexity in terms of numbers of operations in $GF(q)$, the complexity should be multiplied by about $m \cdot n^2$.

3.1 Algorithms for solving MQ over any finite field

Algorithm A

Choose k variables out of n and k' equations out of m. Write each equation G_l, $l = 1,...,k'$, in the following form:

$$g_l(x_1,..,x_k) + \sum_{i=1}^{k} x_i \cdot (\sum_{j=k+1}^{n} \beta_{lij} x_j) + g'_l(x_{k+1},...,x_n) = y_l$$

where g_l, $g_{l'}$ are multivariate quadratic polynomials. Our aim is to remove the part of G_l where $x_1,...,x_k$ and $x_{k+1},...,x_n$ are mixed. For each G_l this is done by imposing k linear relations on the variables $x_{k+1},...,x_n$ by

$$\sum_{j=k+1}^{n} \beta_{lij} x_j = c_{li},\ i = 1,...,k,$$

where the constants c_{li} are elements in $\mathbf{F}$. Let $k = \min(m/2, \lfloor\sqrt{n/2 - \sqrt{n/2}}\rfloor)$, and let $k' = 2k$. Then $k' \leq m$, and we have

$$kk' \leq 2(\sqrt{n/2 - \sqrt{n/2}})^2 \leq n - 2\sqrt{n/2} \leq n - 2k,$$

(i.e., $2k^2 \leq n - 2k$ is satisfied). Therefore $n - k - kk' \geq k$. Thus imposing kk' linear constraints on the $n-k$ variables $x_{k+1},...,x_n$, we can still express them by $\bar{k} \geq k$ independent new variables

$x'_1,...,x'_{\bar{k}}$. Restricting to the equations G_l, $l = 1,...,2k$, the system to solve becomes

$$g'_l(x_1,...,x_k) + h_l(x'_1,...,x'_{\bar{k}}) = y_l,\ l = 1,...,2k.$$

where g'_l differs from g_l by linear summands in $x_1,...,x_k$. This system can be solved in about q^k rather than q^{2k} trials: Generate the set of vectors obtained by evaluating g'_l, $l = 1,...,2k$, in all q^k arguments. Similarly generate a set of q^k result vectors for $y_l - h_l$, $l = 1,...,2k$. By the birthday paradox, with some probability the sets have an element in common (see e.g. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of applied cryptography, p. 53). Once this partial system is solved, with probability q^{2k-m} the remaining $m - 2k$ equations will be satisfied too. Otherwise repeat the whole attack as described, with a different choice of subsets of k variables and of k' equations.

Thereby some polynomial overhead arises by imposing each time a new set of kk' linear constraints on a different set of $n - k$ variables. However this can be mostly avoided if n is slightly larger than m: Suppose that k, m and n besides $2k^2 \leq n - 2k$ also satisfy $2k^2 > m - 2k$. The latter inequality guarantees that the search space is still large enough if only the constants c_{li} change in each trial, but the subsets of variables and equations are chosen only once, so that the linear constraints (except the constants) remain the same. This means that in a parameterized expression describing the new variables $x'_1,...,x'_{\bar{k}}$, only the constants change. Hence in substituting these variables in the quadratic expressions G_l each time, only the linear parts need to be changed. Thus if $2k^2 > m - 2k$ we have:

Complexity of Algorithm A: $\mathcal{O}(q^{m-k})$, where $k = \min(m/2, \lfloor\sqrt{n/2 - \sqrt{n/2}}\rfloor)$. Hence the complexity of algorithm A for increasing n tends to $\mathcal{O}(q^{m/2})$.

Example: Let $n = 40$. Then $k = 4$ satisfies $2k^2 \leq n - 2k$. As this holds by equality, $m = 39$ is the largest number of equations so that also $2k^2 > m - 2k$ is satisfied. Suppose furthermore that $q = 16$, $m = 20$. Then the complexity of algorithm A is of order 2^{64} instead of 2^{80} operations.

Algorithm B

This algorithm uses linearization to reduce the search complexity for solving MQ. Let k be an integer to be specified. In an initial step chose $m - k$ of the quadratic equations to eliminate the quadratic terms $x_i x_j$, $1 \leq i,j \leq k$. Hereby these terms are simply regarded as linear variables. The aim is to get k selected equations in which the variables $x_1,...,x_k$ occur only linearly. This is possible if

$$k(k+1)/2 \leq m - k.$$

Hence

$$k \approx \lfloor\sqrt{2m+2} - 1.5\rfloor,$$

which is at most $m/2$ for $m \geq 2$. After the initial step, which is done only once, the algorithm proceeds as follows:

1. Choose random values in $\mathbf{F}$ for the variables $x_{k+1},...,x_n$.
2. Substitute these values in the k selected quadratic equations to get a system of k linear equations in $x_1,...,x_k$.
3. Solve this system of k linear equations.
4. Go back to 1. as long as the values for $x_1,...,x_n$ thus determined do not satisfy the other $m - k$ quadratic equations.

Step 3 is of complexity $\mathcal{O}(k^3) = \mathcal{O}(m^{3/2})$. However this is not measured by numbers of numeric evaluations of systems of quadratic polynomials but by simple operations like addition and multiplication in the field $\mathbf{F}$. To estimate the complexity of algorithm B, first note that in step 2, a number of $n-k$ values are substituted in k quadratic equations of n variables. This also includes a step of formally simplifying these equations. whose complexity can be reduced by only varying $m-k$ out of $n-k$ variables in step 1 in each trial, and leaving the other variables constant throughout the search. Thus the complexity of step 2 is growing with $k(m-k)^2$. The complexity of the formal step may be some multiple of the unit complexity and can dominate the complexity of step 3. Let $K = \max(C_2,C_3)$, where C_2 and C_3 measure the complexity of steps 2 and 3, respectively. Then we get

Complexity of Algorithm B: $K \cdot q^{m-k}$, where $k = \lfloor \sqrt{2m+2} - 1.5 \rfloor$, and where the factor K grows polynomially in k or $m-k$, respectively, and is not explicitly quantified.

Remark There are possibilities to combine the principles applied in algorithms A and B, leading to algorithms with lower complexity of the exponential part compared to each of algorithms A and B, but at the cost of an increased polynomial overhead. Depending on the variant to be specified of such a combined algorithm, and depending on m, n and q, this algorithm may be more efficient than algorithms A and B alone.

3.2 An Algorithm for solving MQ over fields of characteristic 2

In this section, a general method for simplifying underdefined systems of multivariate quadratic equations over fields of characteristic 2 is described. Combined with a relinearization or XL algorithms [4, 7] it gives another algorithm to solve MQ in case $n \gg m$.

Preliminaries.

Let $\mathbf{F}$ denote the field $GF(2^s)$. Then a quadratic form Q with n variables over $\mathbf{F}$ is a homogeneous polynomial in these variables of total degree two. Two quadratic forms Q_1 and Q_2 are equivalent if Q_1 can be transformed into Q_2 by means of a nonsingular linear transformation of the variables. A quadratic form Q with n variables is nondegenerate if it is not equivalent to a quadratic form with fewer than n variables. A linear form f over $\mathbf{F}$ with n variables is a linear expression $\sum_{i=1}^{n} a_i x_i$, where the coefficients a_i as well as the variables x_i are in $\mathbf{F}$. In later use, $\underline{f}$ will denote an element of $\mathbf{F}^n$, i.e., a column vector with components $a_i \in \mathbf{F}$, whereas the linear form f will denote the scalar product of $\underline{f}$ with $\underline{x} \in \mathbf{F}^n$. A nondegenerate quadratic form over $\mathbf{F}$ can be transformed in a sum of $\lfloor \frac{n}{2} \rfloor$ products of pairs of linear forms $f_{2i-1}f_{2i}$, $i=1,\ldots,\lfloor \frac{n}{2} \rfloor$ plus at most two square terms. More precisely for n odd, $Q(\underline{x})$ can be transformed in:

$$f_1 f_2 + f_3 f_4 + \cdots + f_{n-2} f_{n-1} + f_n^2$$

and for n even in one of the two following forms:

$$f_1 f_2 + f_3 f_4 + \cdots + f_{n-1} f_n$$
$$f_1 f_2 + f_3 f_4 + \cdots + f_{n-1} f_n + f_{n-1}^2 + a f_n^2$$

In the last formula, a stands for an element whose trace over $\mathbf{F}$ has value 1. A constructive proof is given in ([10], p.286). The reduction of $Q(\underline{x})$, as described in[10] requires $O(n^3)$ operations in $GF(2^s)$. The derivation of this fact is straightforward but lengthy.

Simplifying underdefined systems of quadratic equations with more unknowns than equations.

Suppose t is a positive integer and $n \geq (t+1)m$. Then we show that it is possible to adaptively fix $t \cdot m$ linear relations between the unknowns, so that by eliminating unknowns using these relations, we get a simpler system of m equations with m unknowns, where a few equations simultaneously have become linear in the remaining m variables.
By a linear relation between the unknowns we mean a relation $\sum_{i=1}^{n} a_i x_i = b$, where a_i, $i = 1,...,n$ and b are elements of $\mathbf{F}$. The equations that have thus become linear can be used to eliminate further unknowns so that we get a simplified system with less unknowns and equations. If in a later step for solving this system it turns out that there exists no solution, we assign different values b to the relations.

The following Lemma is derived from results in [10] and states two basic facts on quadratic forms that will be useful later. Similar facts also hold for any polynomial of degree two.

Lemma 1 *Let $Q(\underline{x})$ be a nondegenerate quadratic form with n variables over $\mathbf{F}$. Suppose Q is written in reduced representation, $Q = f_1 f_2 + f_3 f_4 + ...$, where the f_i's, $i = 1,...,n$ denote appropriate linear forms. Then*

a) the coefficient vectors $\underline{f}_i$, $i = 1,...,n$, are linearly independent over $\mathbf{F}$

b) Q has $\lfloor \frac{n}{2} \rfloor$ product terms, and at most $\lfloor \frac{n}{2} \rfloor + 1$ linear relations in $x_1,...,x_n$ need to be fixed in order that Q becomes linear in the remaining variables

The next result gives a simple lower bound for the number of equations that can be made linear (depending on t) by fixing linear relations.

Proposition 1 *Let a system of m polynomial equations of degree two with n unknowns be given. Denote $u = \lfloor \log_2(t+1) \rfloor$. Suppose $n \geq (t+1)m$ and that $m \geq u - 1$. Then a number $\nu \leq t \cdot m$ of linear relations between the unknowns can be fixed so that at least $u - 1$ equations become linear in the remaining $n - \nu \geq m$ unknowns.*

Proof: As stated in Lemma 1, b), the polynomial G_1 can be made linear by fixing at most $\lfloor \frac{n}{2} \rfloor + 1$ linear relations. Using the linear relations thus fixed, $\lfloor \frac{n}{2} \rfloor + 1$ unknowns can be eliminated. Iterating this procedure from G_2 onwards, we can fix further linear forms and eliminate unknowns, while the number R of remaining unknowns is at least m. Suppose $t + 1$ is a power of 2, i.e $t + 1 = 2^u$. The general case can be easily reduced to this case. Thus check whether $R \geq m$ holds if the above procedure has been iterated $u - 1$ times: $R \geq 2^u m - (2^{u-1}m + 1) - (2^{u-2}m + 1) - ... - (2m + 1) = 2m - (u - 1) \geq m$, by assumption.

As $u - 1$ variables can be eliminated using the linear equations, Proposition 1 can immediately be used to (slightly) reduce the search for a solution of MQ: The complexity of this search is approximately $q^{m - \log_2(n/m)}$ instead of q^m. We show that depending on t one can generally do better than indicated in Proposition 1. In fact, for very large t, i.e., for $t \geq m$, solving systems of m polynomial equations of degree two with $n \geq (t + 1)m$ unknowns over $\mathbf{F} = GF(2^s)$ has been shown to be easy (cf. [5]). Our method does work for general t, but to get specific results, we focus here on small values of t, as these are of main interest for our cryptographic applications. The idea is to successively fix linear relations so that the number of product terms decreases *simultaneously* in two polynomials G_i. This can be applied to derive the following result (see Appendix A):

Theorem 1 *Let $G_i(x_1,x_2,...,x_n) = y_i$, $i = 1,...,m$, denote a system of m polynomial equations of degree two in n unknowns over $\mathbf{F}$, where $m > 10$ is even, and $n \geq (t + 1)m$. Then $t \cdot m$ linear*

relations between the unknowns can be fixed to get a system of m equations with m variables so that

- *if $t = 2$, G_1 is linear, and G_2 in reduced representation is the sum of at most one product of linear forms, a square of a linear form, and linear terms.*
- *if $t = 3$, G_1 and G_2 are linear, and G_3 in reduced representation is the sum of about $\frac{2m}{9} + 2$ products of linear forms, a square of a linear form, and linear terms.*

Remarks A similar result also holds for an odd number m of equations. Moreover $m > 10$ is just chosen to assure that certain steps in the proof are not void. The complexity of the procedure to get the modified system of m equations with m unknowns in Theorem 1 is of order $\mathcal{O}(n^3)$. By a similar technique, for larger t some more equations can be made linear, e,g, if $t = 8$, about 5 equations can be made simultaneously linear. The method equally applies if the number of variables is a not an integer multiple of the number of equations. Moreover it is possible to improve on Theorem 1, as is shown in Appendix A. Theorem 1 immediately shows that the complexity of solving MQ in case $n \geq 4m$ is at most of order $\mathcal{O}(q^{m-2})$ (without any polynomial overhead). However solving MQ can be significantly improved if Theorem 1 is combined with the methods of relinearization and XL as introduced in [4] and [7] for solving overdefined systems of multivariate quadratic equations.

In [7], the efficiency of relinearization is investigated, and another algorithm for the same purpose, XL (for extended relinearization), is introduced and discussed. Suppose the given system has m equations with n variables, $m > n$, such that $m = \varepsilon n^2$ for $0.1 < \varepsilon \leq 1/2$. Then in [7] it is stated that the algorithm XL is expected to succeed with work factor

$$WF \approx \frac{n^{(\omega \lceil \frac{1}{\sqrt{\varepsilon}} \rceil)}}{(\lceil \frac{1}{\sqrt{\varepsilon}} \rceil)!}, \tag{1}$$

where $2 \leq \omega < 3$ is the exponent of gaussian reduction.
This bound only holds asymptotically in the number of variables n. Therefore in [7] experimental results for various concrete values of m and n are given. In particular, an experiment with 11 equations (over $GF(2^7)$) and 9 variables is reported. In order to solve such a system, the XL algorithm leads to 3543 linear equations in the same number of variables. Thus the complexity of XL to solve a system of quadratic equations with $m = 11$ equations and $n = 9$ variables is of the order 2^{35}, which is much larger than the work factor given by (1). The complexity drops however, if $m - n$ is larger: In ([7], Table 1) results of an experimental analysis of relinearization are given. For our purpose, we quote that solving $m = 12$ equations with $n = 8$ variables leads to a linear system of only 336 equations with 324 variables. The complexity of solving this system is of the order 2^{25}, i.e. it is close to the asymptotic work factor in (1) evaluated for $n = 8$. We now proceed to solve MQ:

Algorithm C

Let $n = t \cdot m$, $t \geq 2$.

1. Suitably fix linear relations between the variables with the simultaneous condition that
   i) two (or three) equations become linear (Theorem 1)
   ii) the simplified system of equations gets sufficiently overdefined for XL to become efficient.
2. Apply XL to solve the simplified system of quadratic equations.

The complexity of algorithm C depends on the complexity of XL. To obtain precise estimates of the complexity of algorithm C, we restrict to cases as mentioned, where the exact complexity of XL (or of relinearization) has been determined experimentally. This is applied in the following examples.

Example 1 Let $t \geq 3$, $m = 16$, and $q = 2^s$. Then apply the techniques used to prove Theorem 1 to the initial system of equations, to get a system of m equations with m variables, where the first two equations are linear and the third equation is a sum of at most 4 products of linear forms, a square of a linear form, and linear terms. Thus we need 5 relations to be fixed in order to eliminate the product terms and the square, so that the third equation also becomes linear. Use these relations and the three linear equations to get a system of 13 equations with 8 unknowns. Apply the result in [7] on relinearization, as quoted, to solve a system of 12 (or 13) equations with 8 variables, so that we get an upper bound for the total complexity of the order of $2^{5s} \cdot 2^{25} = 2^{5s+25}$. Using a refinement of Theorem 1 as sketched in Appendix A, and using the approximation (1), we may even arrive at a complexity of 2^{4s+26}. Thus the complexity of solving MQ for $m = 16$ and $n \geq 64$, is of an order between 2^{4s+26} and 2^{5s+25}.

Example 2 Let $t \geq 2$, $m = 16$, and $q = 2^s$. By similar arguments as in Example 1 we estimate the complexity of algorithm C to solve MQ to be of an order between 2^{4s+28} and 2^{5s+26}.

The complexities as determined in the above examples are the product of the complexity of a (partial) search and the complexity of the XL algorithm, i.e., of solving (large) linear systems of equations. This complexity is measured in numbers of $GF(q)$-operations rather than in numbers of evaluations of m quadratic polynomials in n variables. Therefore the estimates as given in Examples 1 and 2 may be viewed as upper bounds for the complexity of algorithm C.

3.3 Comparing efficiency of the algorithms

Our results show that the problem MQ, even in the underdefined case $n \gg m$, remains exponential in general, as long as $n < m^2$. For different regions of a three dimensional space in q, m and n, one or the other of the algorithms we have presented for solving MQ will be more efficient. To illustrate this point, let, e.g., $m = 16$. Then if n is only slightly larger than m, we expect that algorithm B will outperform other algorithms for solving MQ. However, if n is getting larger, algorithm A will be more efficient than algorithm B. In the case that the order of $\mathbf{F}$ is a power of 2, $q = 2^s$, compare algorithm A with algorithm C: Let, e.g., $n = 48$. Then for algorithm A, $k = 4$ is a suitable value and thus the complexity of algorithm A is of order 2^{12s}. On the other hand the complexity of algorithm C is upper bounded by 2^{5s+26} (see Example 2). Thus this method outperforms algorithm A as soon as $s \geq 4$, e.g., if $s = 4$ the complexities are 2^{46} for algorithm C, compared to 2^{48} for algorithm A, and if $s = 8$ they are 2^{66} compared to 2^{96}. This is due to the fact that the complexity of algorithm A is dominated by a search part and the polynomial overhead can be practically ignored, whereas the complexity of algorithm C is the product of the complexities of a small search and a larger polynomial part.

4 Solving MQ for massively underdefined systems of equations

In the massively underdefined case $n \geq m(m+1)$, a polynomial algorithm was developed in [5] to solve MQ in case $\mathbf{F}$ has characteristic 2, leaving the case of odd characteristic open. In this section, we extend these results to odd characteristics and solve a random underdefined MQ in

polynomial time[1] as soon as:

$$\begin{cases} n \geq m(m+1) & \text{if } \mathbf{F} \text{ has characteristic 2;} \\ n \geq \text{roughly } 2^{\frac{m}{7}}(m+1) & \text{if } \mathbf{F} \text{ has an odd characteristic.} \end{cases}$$

Let $(\mathcal{S})$ be the following system:

$$(\mathcal{S}) \quad \begin{cases} \sum\limits_{1 \leq i \leq j \leq n} a_{ij1} x_i x_j + \sum\limits_{1 \leq i \leq n} b_{i1} x_i + \delta_1 = 0 \\ \vdots \\ \sum\limits_{1 \leq i \leq j \leq n} a_{ijm} x_i x_j + \sum\limits_{1 \leq i \leq n} b_{im} x_i + \delta_m = 0 \end{cases}$$

The main idea of the algorithm consists in using a change of variables such as:

$$\begin{cases} x_1 = \alpha_{1,1} y_1 + \alpha_{2,1} y_2 + ... + \alpha_{t,1} y_t + \alpha_{t+1,1} y_{t+1} + ... + \alpha_{n,1} y_n \\ \quad \vdots \\ x_n = \alpha_{1,n} y_1 + \alpha_{2,n} y_2 + ... + \alpha_{t,n} y_t + \alpha_{t+1,n} y_{t+1} + ... + \alpha_{n,n} y_n \end{cases}$$

whose $\alpha_{i,j}$ coefficients (for $1 \leq i \leq t$, $1 \leq j \leq n$) are found step by step, in order that the resulting system $(\mathcal{S}')$ (written with respect to these new variables y_1, ..., y_n) is easy to solve.

- We begin by choosing randomly $\alpha_{1,1}$, ..., $\alpha_{1,n}$.
- We then compute $\alpha_{2,1}$, ..., $\alpha_{2,n}$ such that $(\mathcal{S}')$ contains no $y_1 y_2$ terms. This condition leads to a system of m *linear* equations in the n unknowns $\alpha_{2,j}$ $(1 \leq j \leq n)$:

$$\sum_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{1,i} \alpha_{2,j} = 0 \qquad (1 \leq k \leq m).$$

- We then compute $\alpha_{3,1}$, ..., $\alpha_{3,n}$ such that $(\mathcal{S}')$ contains neither $y_1 y_3$ terms, nor $y_2 y_3$ terms. This condition is equivalent to the following system of $2m$ *linear* equations in the n unknowns $\alpha_{3,j}$ $(1 \leq j \leq n)$:

$$\begin{cases} \sum\limits_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{1,i} \alpha_{3,j} = 0 & (1 \leq k \leq m) \\ \sum\limits_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{2,i} \alpha_{3,j} = 0 & (1 \leq k \leq m) \end{cases}$$

- ...
- Finally, we compute $\alpha_{t,1}$, ..., $\alpha_{t,n}$ such that $(\mathcal{S}')$ contains neither $y_1 y_t$ terms, nor $y_2 y_t$ terms, ..., nor $y_{t-1} y_t$ terms. This condition gives the following system of $(t-1)m$ *linear* equations in the n unknowns $\alpha_{t,j}$ $(1 \leq j \leq n)$:

$$\begin{cases} \sum\limits_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{1,i} \alpha_{t,j} = 0 & (1 \leq k \leq m) \\ \vdots \\ \sum\limits_{1 \leq i \leq j \leq n} a_{ijk} \alpha_{t-1,i} \alpha_{t,j} = 0 & (1 \leq k \leq m) \end{cases}$$

1. In time polynomial in the size of the initial system that can be exponential.

In general, all these linear equations provide at least one solution (found by Gaussian reductions). In particular, the last system of $m(t-1)$ equations and n unknowns generally gives a solution, as soon as $n > m(t-1)$.

Moreover, the t vectors $\begin{pmatrix} \alpha_{1,1} \\ \vdots \\ \alpha_{1,n} \end{pmatrix}$, ..., $\begin{pmatrix} \alpha_{t,1} \\ \vdots \\ \alpha_{t,n} \end{pmatrix}$ are very likely to be linearly independent for a random system $(\mathcal{S})$. The remaining $\alpha_{i,j}$ constants (*i.e.* those with $t+1 \leq i \leq n$ and $1 \leq j \leq n$) are randomly chosen, so as to obtain a *bijective* change of variables. By rewriting the system $(\mathcal{S})$ with respect to these new variables y_i, we have the following system:

$$(\mathcal{S}') \quad \begin{cases} \sum_{i=1}^{t} \beta_{i,1} y_i^2 + \sum_{i=1}^{t} y_i L_{i,1}(y_{t+1},...,y_n) + Q_1(y_{t+1},...,y_n) = 0 \\ \vdots \\ \sum_{i=1}^{t} \beta_{i,m} y_i^2 + \sum_{i=1}^{t} y_i L_{i,m}(y_{t+1},...,y_n) + Q_m(y_{t+1},...,y_n) = 0 \end{cases}$$

where each $L_{i,j}$ is an affine function and each Q_i is a quadratic function.
Then we compute y_{t+1}, ..., y_n such that:

$$\forall i,\ 1 \leq i \leq t,\ \forall j,\ 1 \leq j \leq m,\ L_{i,j}(y_{t+1},...,y_n) = 0.$$

This is possible because we have to solve a linear system of mt equations and $n-t$ unknowns, which generally provides at least one solution, as long as $n \geq (m+1)t$. We pick one of these solutions. It remains to solve the following system of m equations in the t unknowns y_1, ..., y_t:

$$(\mathcal{S}'') \quad \begin{cases} \sum_{i=1}^{t} \beta_{i1} y_i^2 = \lambda_1 \\ \vdots \\ \sum_{i=1}^{t} \beta_{im} y_i^2 = \lambda_m \end{cases}$$

where $\lambda_k = -Q_k(y_{t+1},...,y_n)$ $(1 \leq k \leq m)$. We call this problem the MQ^2 problem with t variables and m equations. We have two cases:

4.1 When F has characteristic 2 and $n \geq m(m+1)$

In this case, it is enough to have $t = m$ or slightly bigger and MQ^2 is easy. Then the system $(\mathcal{S}'')$ gives the y_i^2 by Gaussian reduction and since $z \mapsto z^2$ is a bijection on any field of characteristic 2, we will then find y_i from the y_i^2.
Our algorithm works for $n \geq (m+1)t = m(m+1)$ as claimed.

4.2 When the characteristic of F is odd and $n = \mathcal{O}(m^2)$

This case is still not completely solved when $n \geq m(m+1)$. In what follows we show an algorithm in which n grows exponentially in m, but very slowly. More precisely when $n \geq$ about $2^{\frac{m}{7}} m(m+1)$, then the system will be solved in polynomial time in n. In practice for many systems with $n = \mathcal{O}(m^2)$ we will also have $n \geq$ about $2^{\frac{m}{7}} m(m+1)$ and our algorithm will solve these cases.

The starting point is the exponential algorithm already mentioned in [5, 6]. In order to solve the MQ^2 problem with $t = \mathcal{O}(m)$ we fix all with the exception of about m variables. The resulting

system is linear in the y_i^2, and is then solved by gaussian elimination. For each resulting solution obtained for y_i^2, the probability that it is indeed a square in $\mathbf{F}$ is 1/2. The probability that all the y_i^2 are squares is 2^{-m} and therefore we need to repeat the attack 2^m times. For this there must be at least about $\log_q(2^m) = \frac{m}{\log_2 q}$ additional variables. Therefore, the algorithm solves the MQ^2 problem in time 2^m when $t \geq m + \frac{m}{\log_2(q)}$.

Certainly, the exponential algorithm is efficient for $m \leq 40$. Now we will improve it. We will show a reduction from the MQ^2 problem with t variables and m equations to the MQ^2 problem with $\frac{t}{40+40/\log_2 q}$ variables and $m-40$ equations. For this we do the following:

1. We assume that the available computing power is greater than 2^{40} operations.
2. First we ignore most of the variables except the $40 + 40/\log_2 q$ variables $y_1, \ldots y_{40+40/\log_2 q}$.
3. With a multiple of 2^{40} operations we find a nonzero solution to the first 40 equations, provided that the contribution of the other variables is zero.
4. The remaining variables are divided in groups of at least $40 + 40/\log_2 q$ variables.
5. For each group of variables, in about 2^{40} operations, we find a nonzero solution, such that their contribution to the first 40 equations is zero.
6. Now we have found a solution $y_1, \ldots y_{40+40/\log_2 q}, \ldots, y_t$ such that the first 40 equations are satisfied.
7. This solution to the first 40 equations, gives in fact many solutions: for example if we have a nonzero solution, $y_{1+40+40/\log_2 q}, \ldots y_{2\cdot(40+40/\log_2 q)}$ such that their contribution to the first 40 equations is zero, such is also the case for the values $zy_{1+40+40/\log_2 q}, \ldots zy_{2\cdot(40+40/\log_2 q)}$ and for any value of z.
8. For each group starting from the second, we can add a new variable z_i, $i = 1..\frac{t}{40+40/\log_2 q} - 1$ as described in 7.
9. Whatever are the values of the z_i, the first 40 equations are satisfied.
10. Now we have another MQ^2 system: with $m-40$ equations and with $\frac{t}{40+40/\log_2 q} - 1$ variables z_i, such that if it is satisfied, the whole original system is satisfied.

The reduction from MQ^2 with (t,m) to the problem with $\left(\frac{t}{40+40/\log_2 q},\ m-40\right)$ can be iterated. Therefore, we see that we can solve MQ^2 for any m as long as

$$t \geq (40 + 40/\log_2 q)^{m/40}$$

The complexity of the algorithm is about $2^{40} \cdot t$, which is essentially linear in t: for each group of $(40 + 40/\log_2 q)$ variables we solve a small MQ^2 in 2^{40}, remaining parts can be neglected. Moreover, if $t > (40 + 40/\log_2 q)^{m/40}$, we ignore the remaining variables and the complexity will be only $2^{40}\,(40 + 40/\log_2 q)^{m/40}$.

Conclusion for odd characteristic.

Now we combine our result for MQ^2 with the reduction from MQ to MQ^2 that works for $n \geq (m+1)t = m(m+1)$ described in 7.4.
Therefore, a massively underdefined system MQ with

$$n \geq (40 + 40/\log_2 q)^{m/40}\,(m+1)$$

can be solved in time about

$$2^{40}\,(40 + 40/\log_2 q)^{m/40}\,.$$

In practice we have usually $\log_2 q > 4$ and therefore:

$$n \geq (50)^{m/40}(m+1) \geq 2^{m/7}(m+1)$$

Example 1: Let $q \approx 2^8$, $m = 40$. The exhaustive search for such MQ is in 2^{320}, whatever is n. Now, if there is enough variables, $n > 1845$, our new algorithm gives about 2^{46} instead of 2^{320}.

Example 2: Let $q = 127$, $m = 80$. The exhaustive search for such MQ is in 2^{559}, whatever is n. Now, if there is enough variables, $n > 136000$, our new algorithm gives about 2^{51} instead of 2^{559}.

5 Application: Cryptanalysis of certain multivariate signature schemes

For several recent signature schemes, the public key consists of a system of m quadratic equations in n variables and where $n \gg m$ (cf. e.g., [5], [6], [8]). For these systems, signatures can be forged if this system of quadratic equations can be solved. Therefore as an immediate cryptographic application, our results lead to new criteria for the choice of the security parameters of such systems.

Due to the parameters chosen for the signature schemes Quartz, Flash and Sflash submitted to Nessie call for primitives, the security of these schemes is not affected by our results. However we break the unbalanced Oil and Vinegar signature scheme (cf. [5]) for two concrete choices of parameters as proposed in [6]:

Let $\mathbf{F} = GF(2^4)$, and let $m = 16$ be the number of public equations. Furthermore let either $n = 48$ or $n = 64$. Then $q = 2^s = 16$, and t in the notation of subsection 7.3.2.0 is either 2 or 3. The public key is given in terms of a set of elements in $\mathbf{F}$, describing the coefficients of the public system of quadratic equations. The length of the public key is 9 Kbytes for $t = 2$ and 16 Kbytes for $t = 3$. The number $m = 16$ of equations has been chosen to defeat Gröbner bases algorithms to solve MQ, and $q^{16} = 2^{64}$ has been chosen in order to prevent from an exhaustive search. Moreover $t \geq 2$ was chosen to escape from an attack as given in [3] and [5], which exploits the trapdoor in Oil and Vinegar signature schemes, and which does hold for $n \approx m$.

Our attack does not rely on the fact that a trapdoor is hidden in the construction of the public polynomials. Rather we directly apply algorithms A and C to show that the complexity of solving MQ with these parameters is significantly lower than 2^{64} trials for exhaustive search. Interestingly, for the chosen parameter sizes the complexities of the two algorithms come quite close. Let $n = 48$. Then the complexity of algorithm A is of order 2^{48} whereas the complexity of algorithm C is about 2^{46} (see subsection 7.3.3). If $n = 64$, $k = 5$ satisfies $2k^2 \geq n - 2k$ and is thus a suitable parameter for algorithm A. Hence the complexity of algorithm A is 2^{44}. The complexity of algorithm C for $n = 64$ is upper bounded by a value between 2^{42} and 2^{45}. If for $m = 16$ and $n = 48$ one wants to increase the security at the cost of a moderate increase of the size of the public key, one could choose a larger subfield, say $q = 2^6$ instead of 2^4. Then algorithm A has complexity 2^{66} but the complexity of algorithm C is at most 2^{56}. Hence the security increase is insufficient. As a consequence of our algorithms, for a multivariate signature scheme with $n = t \cdot m$, $t \geq 2$, the number m has to be 24 or larger. However this number of quadratic equations needs a larger public (and private) key. In particular, this may render unbalanced Oil and Vinegar signature schemes less practical than previously believed.

References

[1] J.-Ch. Faugère, A new efficient algorithm for computing Gröbner bases (F_4), Journal of Pure and Applied Algebra 139 (1999), pp. 61-88. See www.elsevier.com/locate/jpaa.

[2] M. R. Garey, D. S. Johnson, Computers and Intractability, A Guide to the Theory of NP-completeness, W. H. Freeman and Company, New York, 1979.

[3] A. Kipnis, A. Shamir, Cryptanalysis of the Oil and Vinegar Signature Scheme, Advances in Cryptology – CRYPTO'98, Proceedings, H. Krawczyk (Ed.), Lecture Notes in Computer Science, Springer Verlag, vol. 1462, pp. 257 - 266.

[4] A. Kipnis, A. Shamir, Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization, Advances in Cryptology – CRYPTO'99, Proceedings, M. Wiener (Ed.), Lecture Notes in Computer Science, Springer Verlag, vol. 1666, pp. 19 - 30.

[5] A. Kipnis, J. Patarin, L. Goubin, Unbalanced Oil and Vinegar Signature Schemes, Advances in Cryptology – EUROCRYPT'99, Proceedings, J. Stern (Ed.), Lecture Notes in Computer Science, Springer Verlag, vol. 1592, pp. 206 - 222.

[6] A. Kipnis, J. Patarin, L. Goubin, Unbalanced Oil and Vinegar Signature Schemes, Extended Version. Available at http://www.cp8.com/sct/uk/partners/page/c_publication.html

[7] N. Courtois, A. Klimov, J. Patarin, A. Shamir, Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations, Advances in Cryptology – EUROCRYPT'2000, Proceedings, B. Preneel (Ed.), Lecture Notes in Computer Science, Springer Verlag, vol. 1807, pp. 392 - 407.

[8] J. Patarin, N. Courtois, L. Goubin, FLASH, a Fast Multivariate Signature Algorithm, in Progress in Cryptology–CT-RSA 2001, D. Nacchache, ed., vol 2020, Springer Lecture Notes in Computer Science, pp. 298-307.

[9] Jacques Patarin, Louis Goubin, Nicolas Courtois, Quartz, 128-bit long digital signatures; Cryptographers' Track RSA Conference 2001, San Francisco 8-12 Avril 2001, LNCS2020, Springer-Verlag. Also published in Proceedings of the First Open NESSIE Workshop, 13-14 November 2000, Leuven, Belgium.

[10] R. Lidl, R. Niederreiter, Finite fields, Encyclopedia of mathematics and its applications, vol. 20, 1997.

Appendix A: Deriving Theorem 1

In order to derive Theorem 1, a few preparatory steps are explained. For simultaneously reducing the number of product terms in two polynomials G_i, first ignore linear and constant parts and concentrate on homogeneous (degree two) parts. Let $n > 2$ be even (the case n odd is similar) and let $Q_1 = f_1 f_2 + f_3 f_4 + ... + q_1$ and $Q_2 = g_1 g_2 + g_3 g_4 + ... + q_2$ be reduced representations of the homogeneous parts of G_1 and G_2 (which are assumed to be nondegenerate). Depending on the case, q_i, $i = 1,2$, is an abbreviation for 0 or $f_{n-1}^2 + a' f_n^2$ (or for $g_{n-1}^2 + a'' g_n^2$ respectively) for some a',$a'' \in \mathbf{F}$.

Restrict first to reducing the number of product terms in Q_1 and Q_2, and deal with squares of linear terms later. To start with, both Q_1 and Q_2 have $\frac{n}{2}$ product terms.

Consider, e.g., the relation imposed by setting $f_1 = b, b \in \mathbf{F}$ arbitrary. This relation is applied to every polynomial G_i and obviously reduces the number of product terms in Q_1 by one, as $Q_1 = f_3 f_4 + ... + b f_2 + q_1$. For iterating our procedure in later steps, we need to see the explicit effect this linear relation has on Q_2. Recall (cf. Lemma 1, a)) that the coefficient vectors $\underline{g}_i$ of g_i, $i = 1,...,n$, are a basis in $\mathbf{F}^n$. Therefore the coefficient vector $\underline{f}_1$ can be written (use Gaussian elimination) as a linear combination $\underline{f}_1 = \sum_{i=1}^{n} \alpha_i \underline{g}_i$ for suitable $\alpha_i \in \mathbf{F}$, $i = 1,...,n$, where not all α_i's are 0. Thus we get the identity of linear forms $\sum_{i=1}^{n} \alpha_i g_i = f_1$. Suppose, e.g., that

$\alpha_1 \neq 0$. Use the relation $\sum_{i=1}^{n} \alpha_i g_i = f_1 = b$ to express g_1 as $g_1 = \sum_{i=2}^{n} \alpha_i' g_i + b'$, where $\alpha_i' = \frac{\alpha_i}{\alpha_1}$, $i = 1,...,n$ and $b' = \frac{b}{\alpha_1}$. Thus substituting g_1 in Q_2 we get

$$Q_2 = (\sum_{i=2}^{n} \alpha_i' g_i + b')g_2 + g_3 g_4 + ... + g_{n-1} g_n + q_2 = (g_3 + \alpha_4' g_2)(g_4 + \alpha_3' g_2) + \tag{2}$$

$$+... + (g_{n-1} + \alpha_n' g_2)(g_n + \alpha_{n-1}' g_2) + (\alpha_2' + \alpha_3'\alpha_4' + ...\alpha_{n-1}'\alpha_n')g_2^2 + b' g_2 + q_2,$$

where the last expression has $\frac{n}{2} - 1$ products of $n-2$ linear forms g_j', $j = 3,...,n$ (we still focus only on product terms and not on squares). Note that using (2) the simultaneous reduction of product terms in Q_1 and Q_2 with a given linear relation can be carried out efficiently. After eliminating one variable x_i using the relation $f_1 = b$, Q_1 has $n-1$ variables and $\frac{n}{2} - 1$ product terms and is now of the form (renaming f_i' by f_i) $Q_1 = f_3 f_4 + ... + f_{n-1} f_n +$ linear terms + squares of linear terms, and similarly for Q_2.

To simultaneously eliminate further product terms in Q_1 and Q_2, consider the system of $n-1$ linear equations with unknowns α_i,$\beta_j \in \mathbf{F}$, i,$j = 3,...,n$,

$$\sum_{i=3}^{n} \alpha_i \underline{f}_i + \sum_{j=3}^{n} \beta_j \underline{g}_j = 0, \tag{3}$$

We still have $2(n-2)$ unknowns, and thus many solutions, from which we choose a nontrivial one. Furthermore, the $\underline{f}_i$'s, as well as the $\underline{g}_i$'s, can be assumed to be linearly independent. Hence not all α_i's and not all β_j's are 0.

Then both sides of the identity of linear forms $\sum_{i=3}^{n} \alpha_i f_i = \sum_{j=3}^{n} \beta_j g_j$ are of the form $\sum_{i=1}^{n} a_i x_i$ for suitable $a_i \in \mathbf{F}$, $i = 1,...,n$. So let $\sum_{i=1}^{n} a_i x_i = b$, $b \in \mathbf{F}$ arbitrary, be the relation to be fixed. Then we can eliminate one product term in Q_1 and one in Q_2 as before, and in the same time eliminate one further variable x_i. This procedure can be repeated while the linear system (3) has a nontrivial solution. After we have fixed r relations, $n - 2r$ linear forms remain involved in products in each of Q_1 and Q_2, and the number of variables after elimination has decreased to $n - r$. Therefore system (3) has a solution as long as $(n-2r) + (n-2r) > n - r$. This simplifies to $r < \frac{n}{3}$. As soon as $r + 1 > \frac{n}{3}$ for some $r > 0$, consider, e.g., polynomials G_1 and G_3 and simultaneously eliminate product terms in G_1 and G_3 and so on.

Finally fix linear forms occurring in squares. As squaring is a linear bijective operation in characteristic 2, sums of squares simplify to a single square of a linear relation and we need only fix one linear relation in each polynomial G_i in which we have eliminated product terms.

Proof of Theorem 1 (Sketch): The proof proceeds in three steps. (In subsequent equalities between integers and fractions, either floors or ceilings should be taken. These operations depend on divisibility properties of n and can be ignored as far as their effects cancel out in book-keeping of terms.) Let Q_1, Q_2 and Q_3 denote the homogeneous parts of the polynomials G_1, G_2 and G_3.

Step 1: Simultaneously eliminate product terms in Q_1 and Q_2 by fixing appropriate linear relations between the variables as described. We can eliminate $r+1$ products, where the condition $r < \frac{n}{3}$ holds. Thus $r + 1 = \frac{n}{3}$, and in each of Q_1 and Q_2 there remain $\frac{n}{2} - \frac{n}{3} = \frac{n}{6}$ product terms with $\frac{n}{3}$ linear forms as factors involved. Moreover, using the fixed linear relations, eliminate $\frac{n}{3}$ unknowns in all polynomials G_i. Thus all G_i's are polynomials of $n - \frac{n}{3} = \frac{2n}{3}$ variables and for $i > 2$, G_i has at most $\frac{n}{3}$ product terms with at most $\frac{2n}{3}$ linear forms as factors.

In a similar way, in Steps 2 and 3 the numbers of product terms in Q_i, $i = 1,2,3$, are further reduced: In Step 2, product terms in Q_1 and Q_3 are simultaneously eliminated, whereas Step 3 deals with simultaneously eliminating product terms in Q_2 and Q_3. Book-keeping of the number

of remaining nonlinear summands in the Q_i's leads to the simplified system of m equations in m variables as stated in Theorem 1. Details are omitted here due to space limitation.

A refinement. In all three steps of the proof of Theorem 1, a number $r+1$ is computed, which is the number of products that can simultaneously be eliminated in reduced representations of two quadratic forms. The number r has been limited by the condition that the sum of the numbers of linear forms occurring in products in both quadratic forms exceeds the number of components of the coefficient vectors of the linear forms. This was to assure that a linear system of equations similar to (3) has nontrivial solutions. However, with a probability $p > 0$, these coefficient vectors are linearly dependent even if not enough of them are available to satisfy the above condition. It can be shown that this probability is about 0.63. By trying Step 1 a few times, each time choosing different linear relations to be fixed, one can increase this probability to close to 1. This applies also to the other steps and allows to eliminate a few more nonlinear terms than stated in Theorem 1.

Construction de nouveaux schémas cryptographiques

Dans les cinq articles ci-dessous sont abordées d'une part les hypothèses calculatoires sur lesquelles reposent les algorithmes de cryptographie multivariable, et d'autre part la construction concrète de plusieurs algorithmes de signature électronique, dont SFLASH et QUARTZ soumis au projet européen NESSIE. La problématique de l'optimisation des implémentations sur carte à microprocesseur est aussi analysée.

Page 261 – EUROCRYPT'98 (Version complète)
Improved Algorithms for Isomorphisms of Polynomials.
Nicolas Courtois, Louis Goubin et Jacques Patarin

Cet article présente une analyse détaillée des problèmes d'isomorphismes de systèmes d'équations polynomiales. Des résultats de théorie de la complexité sont prouvés pour les problèmes IP et MP, notamment leur lien avec la théorie des graphes. Par ailleurs, des algorithmes performants sont présentés pour résoudre le problème IP.

Page 287 – ASIACRYPT'98 (Version complète)
C^*_{-+} and *HM*: Variations around two schemes of T. Matsumoto and H. Imai.
Nicolas Courtois, Louis Goubin et Jacques Patarin

Dans cet article sont introduits plusieurs concepts nouveaux de la cryptographie multivariable. L'idée de ne pas publier toutes les équations de la clé publique, ou bien d'en ajouter, mène aux cryptosystèmes C^*_{-+}. Une cryptanalyse est présentée dans certains cas. Le cryptosystème HM est également décrit, il s'appuie sur des transformations matricielles multivariables.

Page 309 – CT-RSA'2001 (Version complète)
QUARTZ, 128-bit long digital signatures.
Nicolas Courtois, Louis Goubin et Jacques Patarin

L'algorithme QUARTZ est une application des systèmes multivariables pour obtenir des signatures extrêmement courtes. Dans cet article, une description détaillée en est donnée, avec une analyse de la sécurité pour les paramètres précis choisis, ainsi que des indications pour l'implémentation du schéma.

Page 323 – CT-RSA'2001 (Version complète)
FLASH, a fast multivariate signature algorithm.
Nicolas Courtois, Louis Goubin et Jacques Patarin

L'algorithme SFLASH, qui repose sur le schéma C^*_- d'ASIACRYPT'98, est particulièrement bien adapté pour la génération extrêmement rapide de signatures

digitales. Cet article montre comment les paramètres doivent être choisis pour atteindre un niveau de sécurité de 2^{80} en pratique, en tenant compte des cryptanalyses décrites au chapitre 4.

Page 333 – PKC'2003

A Fast and Secure Implementation of SFLASH.

Mehdi-Laurent Akkar, Nicolas Courtois, Romain Duteuil et Louis Goubin

Le cas de l'algorithme SFLASH est abordé ici sous l'angle de l'implémentation sur une carte à microprocesseur. Plusieurs méthodes d'optimisation sont ainsi décrites, soit algorithmiques, soit plus proches du langage utilisé. La question de la résistance aux attaques par analyse de consommation électrique est également étudiée.

Improved Algorithms for Isomorphisms of Polynomials

EUROCRYPT'98 *(Version complète)*
Article avec Nicolas Courtois (Université de Toulon) et Jacques Patarin (Bull SC&T)

Abstract

This paper is about the design of improved algorithms to solve Isomorphisms of Polynomials (IP) problems. These problems were first explicitly related to the problem of finding the secret key of some asymmetric cryptographic algorithms (such as Matsumoto and Imai's C^* scheme of [13], or some variations of Patarin's HFE scheme of [15]). Moreover, in [15], it was shown that IP can be used in order to design an asymmetric authentication or signature scheme in a straightforward way. We also introduce the more general Morphisms of Polynomials problem (MP). As we see in this paper, these problems IP and MP have deep links with famous problems such as the Isomorphism of Graphs problem or the problem of fast multiplication of $n \times n$ matrices.

The complexities of our algorithms for IP are still not polynomial, but they are much more efficient than the previously known algorithms. For example, for the IP problem of finding the two secret matrices of a Matsumoto-Imai C^* scheme over $K = \mathbf{F}_q$, the complexity of our algorithms is $\mathcal{O}(q^{n/2})$ instead of $\mathcal{O}(q^{(n^2)})$ for previous algorithms. (In [14], the C^* scheme was broken, but the secret key was not found). Moreover, we have algorithms to achieve a complexity $\mathcal{O}(q^{\frac{3}{2}n})$ on any system of n quadratic equations with n variables over $K = \mathbf{F}_q$ (not only equations from C^*). We also show that the problem of deciding whether a polynomial isomorphism exists between two sets of equations is not NP-complete (assuming the classical hypothesis about Arthur-Merlin games), but solving IP is at least as difficult as the Graph Isomorphism problem (GI) (and perhaps much more difficult), so that IP is unlikely to be solvable in polynomial time. Moreover, the more general Morphisms of Polynomials problem (MP) is NP-hard. Finally, we suggest some variations of the IP problem that may be particularly convenient for cryptographic use.

Key Words: Morphisms of Polynomials, Isomorphisms of Polynomials, Graph Isomorphisms, Zero-Knowledge Authentications, Asymmetric Signatures.

1 Introduction

This paper presents new algorithms for the Isomorphism of Polynomials (IP) problem. IP was explicitly described in [15], where it was shown that if some efficient algorithm exists for IP, then the secret key of some asymmetric cryptosystems (such as the C^* scheme of [13] or some variations of the HFE scheme of [15]) would be found. (The cryptanalysis of C^* given in [14]

breaks the scheme without finding the secret key). No polynomial algorithm is known for IP. On the other side, in [15], it was also shown that if no efficient algorithm exists for IP, then this IP problem can be used to design some zero-knowledge authentication schemes. These schemes can also be transformed in order to have an asymmetric signature scheme.

This paper is divided into two parts. In part I, we present different variations of IP and we study a closely related problem, called MP for "Morphisms of Polynomials". We then show that these problems are closely related to other and more famous problems such as the Graph Isomorphism problem, and Fast Matrix Multiplication. In part II, we present our improved algorithms for IP. These algorithms are not polynomial (so the schemes based on IP are not broken), but they are much more efficient that the previously known schemes.

Part I: Presentation and general properties of the IP and MP problems

2 The IP and MP problems

IP was presented in [15]. Let us recall what this problem is, in the particular case of quadratic forms (the problem can be generalized without difficulties to cubic forms, as well as forms of higher degree).

Let u and n be two integers. Let $\mathbf{F}_q$ be a finite field. Let $(\mathcal{A})$ be a public set of u quadratic equations with n variables a_1, ..., a_n over the field $\mathbf{F}_q$. We can write these equations as follows:

$$b_k = \sum_i \sum_j \gamma_{ijk} a_i a_j + \sum_i \mu_{ik} a_i + \delta_k \qquad (1 \leq k \leq u). \qquad (\mathcal{A})$$

Now let s be a bijective and affine transformation of the variables a_i, $1 \leq i \leq n$, and let t be a bijective and affine transformation of the variables b_k, $1 \leq k \leq u$.

We denote $s(a_1,...,a_n) = (x_1,...,x_n)$ and $t(b_1,...,b_n) = (y_1,...,y_n)$.

From $(\mathcal{A})$ we obtain another set $(\mathcal{B})$ of k equations that give the y_k values from the x_i values:

$$y_k = \sum_i \sum_j \gamma'_{ijk} x_i x_j + \sum_i \mu'_{ik} x_i + \delta'_k \qquad (1 \leq k \leq u). \qquad (\mathcal{B})$$

We say that (s,t) is an *isomorphism* from $(\mathcal{A})$ to $(\mathcal{B})$, and we say that $(\mathcal{A})$ and $(\mathcal{B})$ are *isomorphic*.

The IP problem is the following: if $(\mathcal{A})$ and $(\mathcal{B})$ are two public sets of u quadratic equations, and if $(\mathcal{A})$ and $(\mathcal{B})$ are isomorphic, find an isomorphism (s,t) from $(\mathcal{A})$ to $(\mathcal{B})$.

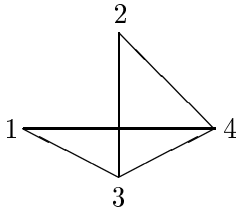
When s and t are not necessary bijective, we call the corresponding problem the "Morphism of Polynomials" problem (MP).

3 Examples

We now present three "toy examples" in order to become more familiar with the IP and MP problems.

Example of IP with two secrets

Let $K = \mathbf{F}_2$ be the field in which all the variables x_0, ..., x_4, y_0, ..., y_4, a_0, ..., a_4, b_0, ..., b_4 lie.



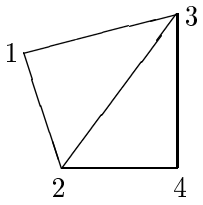


FIG. 1 – *Graph* (I) *Graph* (II)

Let:

$$(\mathcal{A})\begin{cases} b_1 = a_1 + a_1a_5 + a_2a_3 + a_2a_4 + a_3a_4 \\ b_2 = a_3 + a_4 + a_5 + a_1a_2 + a_1a_4 + a_4a_5 \\ b_3 = a_5 + a_1a_2 + a_1a_3 + a_1a_5 + a_2a_3 + a_3a_5 + a_4a_5 \\ b_4 = a_2 + a_3 + a_4 + a_5 + a_1a_5 + a_3a_4 + a_3a_5 \\ b_5 = a_4 + a_1a_3 + a_1a_5 + a_2a_3 + a_2a_4 + a_2a_5 + a_3a_4 + a_3a_5 + a_4a_5 \end{cases}$$

and let:

$$(\mathcal{B})\begin{cases} y_1 = x_1 + x_3 + x_4 + x_5 + x_1x_2 + x_1x_3 + x_1x_5 + x_2x_4 + x_3x_5 + x_4x_5 \\ y_2 = x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_1x_5 + x_4x_5 \\ y_3 = x_1 + x_3 + x_4 + x_1x_2 + x_1x_5 + x_2x_3 + x_3x_4 \\ y_4 = x_3 + x_4 + x_1x_2 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 \\ y_5 = x_2 + x_4 + x_5 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2x_5 \end{cases}$$

The problem is to find two bijective and linear transformations s and t such that $(x_1,...,x_5) = s(a_1,...,a_5)$, $(y_1,...,y_5) = t(b_1,...,b_5)$ and such that $(\mathcal{A})$ becomes $(\mathcal{B})$ with these transformations. (This is sometimes called an "IP problem with two secrets" since here there are two secret affine transformations to find: s and t).

Note 1: This "toy example" comes from the "toy example" given in [13] for C^*, when the 8 variables are separated in $3+5$ variables, as explained in [14]. The equation $(\mathcal{A})$ come from the equation $b = a^3$ in $\mathbf{F}_{2^5}$.

Note 2: It is possible to show that, when s is found, then t is easy to find. However, an exhaustive search on s would require $2^{n^2} = 2^{25}$ computations in this toy example. Our aim is to improve this complexity.

Example of IP with one secret

Let us consider the problem of finding an isomorphism between graphs (I) and (II) of figure 1.

Let K be the finite field $\mathbf{F}_2$.

Let $(\mathcal{A})$ and $(\mathcal{B})$ be the two following sets of equations:

$$(\mathcal{A})\begin{cases} y_1 = a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \\ y_2 = a_1^2 + a_2^2 + a_3^2 + a_4^2 \end{cases} \quad \text{and} \quad (\mathcal{B})\begin{cases} y_1 = x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_2 = x_1^2 + x_2^2 + x_3^2 + x_4^2 \end{cases}$$

The problem is to find a bijective and linear transformation s such that $(x_1,...,x_4) = s(a_1,...,a_4)$, and such that – with this change of variables – the system $(\mathcal{A})$ becomes the system $(\mathcal{B})$ (*i.e.* such that y_1 becomes as written in $(\mathcal{B})$, and y_2 also becomes as written in $(\mathcal{B})$).

If we are able to find all the solutions of this IP problem (this is sometimes called an "IP problem with one secret" since here there is one affine transformation s to find), then some of these solutions will be Graph Isomorphisms from Graph (I) to Graph (II). This comes from the fact that – in $(\mathcal{A})$ – y_1 was written such that the monomial $a_i a_j$ is in y_1 if and only if the point i is linked with the point j in graph (I). Similarly – in $(\mathcal{B})$ – y_1 was written such that the monomial $x_i x_j$ is in y_1 if and only if the point i is linked with the point j in graph (II). Moreover, y_2 was chosen to be a symmetrical polynomial of the variables, so that any graph isomorphism between graph (I) and graph (II) will be a solution to this particular IP problem with one secret. In section 4, we will generalize this construction to study more precisely the links between Graph Isomorphism and IP with one secret.

Example of MP

The variables belong to a ring or to a field K. Let $(\mathcal{A})$ and $(\mathcal{B})$ be the two following sets of equations:

$$(\mathcal{A}) \begin{cases} b_1 = a_1 a'_1 \\ b_2 = a_2 a'_2 \\ b_3 = a_3 a'_3 \\ b_4 = a_4 a'_4 \\ b_5 = a_5 a'_5 \\ b_6 = a_6 a'_6 \\ b_7 = a_7 a'_7 \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = x_1 x'_1 + x_3 x'_2 \\ y_2 = x_2 x'_1 + x_4 x'_2 \\ y_3 = x_1 x'_3 + x_3 x'_4 \\ y_4 = x_2 x'_3 + x_4 x'_4 \end{cases}$$

The problem is to find two (non bijective) linear transformations s and t such that $(a_1,...,a_7,a'_1,...,a'_7) = s(x_1,...,x_4,x'_1,...,x'_4)$, $(y_1,...,y_4) = t(b_1,...,b_7)$, s of rank 8, t of rank 4, and such that when $(\mathcal{A})$ is satisfied and when these two transformations s and t are done, then $(\mathcal{B})$ is satisfied.

We can notice that the system $(\mathcal{B})$ is the system of equations of a product of two 2×2 matrices:

$$\begin{pmatrix} y_1 & y_3 \\ y_2 & y_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} \cdot \begin{pmatrix} x'_1 & x'_3 \\ x'_2 & x'_4 \end{pmatrix}$$

In the system $(\mathcal{A})$, we have exactly seven multiplications, so that solving the MP problem is solving the problem: "How to multiply 2×2 matrices with 7 (instead of 8) multiplications". (The number of additions, subtractions and multiplications with constants of K can be high, but the number of multiplications of terms of the matrices is at most 7).

V. Strassen found in 1969 how to multiply 2×2 matrices with 7 multiplications (cf [17]). His solution is the following:

$$\begin{cases} a_1 = x_3 - x_4 \\ a_2 = x_1 + x_4 \\ a_3 = x_1 - x_2 \\ a_4 = x_1 + x_3 \\ a_5 = x_1 \\ a_6 = x_4 \\ a_7 = x_2 + x_4 \end{cases} \qquad \begin{cases} a'_1 = x'_2 + x'_4 \\ a'_2 = x'_1 + x'_4 \\ a'_3 = x'_1 + x'_3 \\ a'_4 = x'_4 \\ a'_5 = x'_3 - x'_4 \\ a'_6 = x'_2 - x'_1 \\ a'_7 = x'_1 \end{cases} \qquad \text{and} \qquad \begin{cases} y_1 = b_1 + b_2 - b_4 + b_6 \\ y_2 = b_4 + b_5 \\ y_3 = b_6 + b_7 \\ y_4 = b_2 - b_3 + b_5 - b_7 \end{cases}$$

4 IP with one secret is at least as difficult as Graph Isomorphism

First links between the two problems

We first generalize the construction of section 3 about IP with one secret. Let (I) and (II) be two n-vertices graphs, and let $(\mathcal{A})$ and $(\mathcal{B})$ the following systems:

$$(\mathcal{A}) \begin{cases} y_1 = \sum_{i,j} \gamma_{ij} x_i x_j \\ y_2 = \sum_{i=1}^{n} x_i^2 \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = \sum_{i,j} \mu_{ij} a_i a_j \\ y_2 = \sum_{i=1}^{n} a_i^2, \end{cases}$$

where all the γ_{ij} and μ_{ij} coefficients lie in $\{0,1\}$ and satisfy:

$$\gamma_{ij} = 1 \Leftrightarrow \textit{The } i \textit{ and } j \textit{ vertices of graph } (I) \textit{ are linked together}$$

$$\mu_{ij} = 1 \Leftrightarrow \textit{The } i \textit{ and } j \textit{ vertices of graph } (II) \textit{ are linked together.}$$

It is easy to see that each graph isomorphism between (I) and (II) corresponds to a morphism of polynomials between $(\mathcal{A})$ and $(\mathcal{B})$. This morphism of polynomials is a very special one, since it can be defined by $a_i = x_{\varphi(i)}$, where φ is some permutation of $\{1,...,n\}$.

Reciprocally, can *any* isomorphism of polynomials between $(\mathcal{A})$ and $(\mathcal{B})$ be characterized by $a_i = x_{\varphi(i)}$ for some permutation φ of $\{1,...,n\}$? Of course not, and explicit examples can be built to be convinced. Nevertheless, we can consider the following argument: there exist $q^{n(n+1)}$ possible affine transformations between the x_i and the a_i, among which $q^n(q^n-1)(q^n-q)(q^n-q^2)...(q^n-q^{n-1})$ are bijective. The probability that such a transformation leaves y_1 invariant is *a priori* $\leq \frac{1}{q^{\frac{n(n+1)}{2}+1}}$ (because there are $\frac{n(n+1)}{2}$ monomials $x_i x_j$ $(i \leq j)$, and a constant term). The same property is *a priori* true for y_2. Moreover, since

$$q^n(q^n-1)(q^n-q)(q^n-q^2)...(q^n-q^{n-1}) < \left(q^{\frac{n(n+1)}{2}+1}\right)^2,$$

we can think that – "most of the time" – there are very few solutions for the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ that do not correspond to a solution of Graph Isomorphism. Of course, it is a rough evaluation. Nevertheless, it suggests that if a method is found to solve the IP problem with one secret, then we should be able to use this method to also solve the Graph Isomorphism problem. We now study how to obtain a real proof of this property.

Possible improvements

Notice that several practical solutions can be thought of to better eliminate the "unwanted" solutions of the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ (*i.e.* those which do not correspond to a graph isomorphism).

For instance, we can add the equation $y_3 = x_1 + x_2 + ... + x_n$ to $(\mathcal{A})$, and similarly $y_3 = a_1 + a_2 + ... + a_n$ to $(\mathcal{B})$. More generally, any symmetrical polynomial can be added, for example $y_4 = x_1^3 + x_2^3 + ... + x_n^3$ to $(\mathcal{A})$ and $y_4 = a_1^3 + a_2^3 + ... + a_n^3$ to $(\mathcal{B})$. (In that case, we must suppose that we can solve IP not only for quadratic systems, but also for systems of higher degree).

Another *ad hoc* solution consists in adding the equation $y_3 = x_1^2$ to $(\mathcal{A})$ and $y_3 = a_k^2$ to $(\mathcal{B})$, where k is randomly chosen between 1 and n. This way, we can expect – for some of these k values – to eliminate the "unwanted" solutions, while keeping at least one solution of the Graph Isomorphism problem (but this is not a proof).

A better idea, in order to obtain real proofs, is to introduce two new variables, say X and X', and to consider the two following systems:

$$(\mathcal{A}) \begin{cases} y_1 = \sum_{i,j} \gamma_{ij} x_i x_j \\ y_2 = (X - x_1)(X - x_2)...(X - x_n) \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = \sum_{i,j} \mu_{ij} a_i a_j \\ y_2 = (X' - a_1)(X' - a_2)...(X' - a_n) \end{cases},$$

where the γ_{ij} and μ_{ij} values are defined as before.

Now, since $K[X,X',a_1,...,a_n,x_1,...,x_n]$ is a factorial ring, any solution of the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ is indeed a solution of the Graph Isomorphism problem (note that the two equations in $(\mathcal{A})$ and $(\mathcal{B})$ can also be combined in one equation like this: $y_1 = (X - x_1)...(X - x_n)(X - \sum_{i,j} \gamma_{ij} x_i x_j)$). However, there is still a problem with this construction: the equations y_2 are of degree n, and if an equation of degree n, with $n + 1$ variables, is given as a sum of monomials (expanded form), it has a non polynomial (in n) number of terms. We now present the real proof of the link between IP and GI, where this problem is solved, and moreover, we consider again IP on quadratic forms only.

The real proof

We first prove a general property about permutations:

Theorem 7 *Any permutation σ of $\{1,...,n\}$ can be written in a unique way as follows:*

$$\sigma = \tau_{i_n,n} \circ \tau_{i_{n-1},n-1} \circ ... \circ \tau_{i_1,1},$$

where $i_k \in \{1,...,k\}$ for all k, $1 \leq k \leq n$, and where $\tau_{i,j}$ is the permutation that exchanges i and j (by convention, $\tau_{i,i} = Id$).

Proof: We proceed by induction on n.

It is easy to check the result for $n = 1$ and $n = 2$. Let us suppose that the result is true for a given integer n, and let us consider a permutation σ of $\{1,...,n+1\}$.

Let $i_{n+1} = \sigma(n+1)$, and let $\sigma' = \tau_{i_{n+1},n+1} \circ \sigma$. The fact that $\sigma'(n+1) = n+1$ shows that σ' induces a permutation of $\{1,...,n\}$ (that we also denote by σ').

From the induction hypothesis, there exist indices i_1, ..., i_n satisfying $i_k \in \{1,...,k\}$ for all k, $1 \leq k \leq n$, and such that:

$$\sigma' = \tau_{i_n,n} \circ ... \circ \tau_{i_1,1}.$$

As a result, we have:

$$\sigma = \tau_{i_{n+1},n+1} \circ ... \circ \tau_{i_1,1},$$

with $i_k \in \{1,...,k\}$ for all k, $1 \leq k \leq n+1$.

We have thus proved the existence of the above decomposition of σ.

The unicity is a consequence of the following combinatoric argument. Let us denote by S_n the set of all permutations of $\{1,...,n\}$ and by T the set of all permutations that can be written as $\tau_{i_n,n} \circ ... \circ \tau_{i_1,1}$. There are at most n possibilities for i_n, $n-1$ possibilities for i_{n-1}, ..., and 1 possibility for i_1. Therefore, T contains $\leq n!$ elements. Moreover, we have just proven that the function

$$F : \begin{cases} T \to S_n \\ (\tau_{i_n,n},...,\tau_{i_1,1}) \mapsto \tau_{i_n,n} \circ ... \circ \tau_{i_1,1} \end{cases}$$

is surjective. Since $|T| \leq |S_n|$, we can conclude that F is bijective. The unicity is thus proven.

We now see how to use this theorem to "translate" the fact that the transformation s involved in the NP-problem corresponding to a Graph Isomorphism instance is characterized by $a_i = x_{\varphi(i)}$ for some permutation $\varphi \in S_n$. According to theorem 4.1, such a permutation can be written as the product of (at most) n permutations $\tau_{i,j}$.

To simplify, let us first give a "translation" of the fact that an affine transformation $s : x \mapsto a$ is characterized either by $s = \mathrm{Id}$, or by $a_i = x_{\varphi(i)}$ (for all i), with $\varphi = \tau_{ij}$ for two given indices i and j. It is equivalent to saying that s is an isomorphism of polynomials between the two following systems:

$$(\mathcal{A}) \begin{cases} y_0 = (X - x_i)(X - x_j) \\ y_k = x_k \quad (1 \leq k \leq n,\, k \neq i,j) \\ y_{n+1} = X \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_0 = (A - a_i)(A - a_j) \\ y_k = a_k \quad (1 \leq k \leq n,\, k \neq i,j) \\ y_{n+1} = A \end{cases}$$

By using this argument several times, we obtain the following "translation" of a Graph Isomorphism problem into an IP problem: s corresponds to an isomorphism of the graphs (*i.e.* in particular is of the form $a_i = x_{\varphi(i)}$ for some $\varphi \in S_n$) iff it is an isomorphism of polynomials between the two following systems:

$$(\mathcal{A}) \begin{cases} y_0 = \sum_{i,j} \gamma_{ij} x_i x_j \\ \forall i,j,\ 1 \leq i < j \leq n,\ (i,j) \neq (n-1,n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(X - x_i^{(\nu_{ij})}\right)\left(X - x_j^{(\nu_{ij})}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = x_k^{(\nu_{ij})} \quad (1 \leq k < i) \\ y_{(2n-1)\nu_{ij}+k} = x_k^{(\nu_{ij})} \quad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = x_k^{(\nu_{ij})} \quad (j < k \leq n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = x_k^{(\nu_{ij+1})} \quad (1 \leq k \leq n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = \left(X - x_i^{((n-2)(n+1)/2)}\right)\left(X - x_j^{((n-2)(n+1)/2)}\right) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = x_k^{((n-2)(n+1)/2)} \quad (1 \leq k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = x_k^{((n-2)(n+1)/2)} \quad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = x_k^{((n-2)(n+1)/2)} \quad (j < k \leq n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = X \end{cases}$$

and

$$(\mathcal{B}) \begin{cases} y_0 = \sum_{i,j} \mu_{ij} x_i x_j \\ \forall i,j,\ 1 \leq i < j \leq n,\ (i,j) \neq (n-1,n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(A - a_i^{(\nu_{ij}+1)}\right)\left(A - a_j^{(\nu_{ij}+1)}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = a_k^{(\nu_{ij}+1)} \quad (1 \leq k < i) \\ y_{(2n-1)\nu_{ij}+k} = a_k^{(\nu_{ij}+1)} \quad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = a_k^{(\nu_{ij}+1)} \quad (j < k \leq n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = a_k^{(\nu_{ij+1})} \quad (1 \leq k \leq n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = (A - a_i)(A - a_j) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = a_k \quad (1 \leq k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = a_k \quad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = a_k \quad (j < k \leq n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = A \end{cases}$$

where $\nu_{ij} = i - 1 + \frac{(j-1)(j-2)}{2}$ and X, A, as well as the $x_k^{(\nu)}$ and $a_k^{(\nu)}$, are intermediate variables. Each of these two systems contains $\frac{2n^3-3n^2-n+4}{2}$ equations over $\frac{(n+1)(n^2-2n+2)}{2}$ variables.

Conclusion: By solving IP with one secret on a set of $\mathcal{O}(n^3)$ quadratic equations we can solve any Graph Isomorphism problem with n vertices. Therefore, IP is at least as hard as GI.

Remark:

1. In the particular case where $(\mathcal{A})$ and $(\mathcal{B})$ contain only *one* quadratic equation with n variables, there exist a polynomial algorithm to find the isomorphism (see [15]).
2. We do not pretend that this construction gives a new and more efficient way to solve the Graph Isomorphism problem. In fact, although no polynomial algorithm is known for the Graph Isomorphism problem, in practice very efficient algorithms are known: for example it is feasible to find an isomorphism for graphs even for "hard" instances of 1000 vertices graphs in less than 10 minutes on a personal computer (see [6] p. 22). So the main interest of this construction is to show that the IP problem with one secret is probably not solvable with a probabilistic algorithm of polynomial complexity. (Because GI was carefully studied and many people think that GI is not solvable in polynomial complexity).

5 MP is NP-hard

In this section, we prove that the Morphism of Polynomials (MP) problem, defined in section 2, is NP-hard for any finite field, and for the rational numbers.

The proof uses some properties of three-dimensional tensors. Let us first recall some basic definitions:

Definitions:

1. A *three-dimensional tensor* is a three-dimensional array $T = (t_{ijk})$ of numbers.
2. It has *rank 1* iff it can be written as the outer product of three vectors (*i.e.* iff there exist three vectors x, y and z such that $m_{ijk} = x_i y_j z_k$ for all indices i, j, k).
3. The rank of a general tensor T is the minimal number of rank 1 tensors T_ν such that $T = \sum_\nu T_\nu$.

The following result about the complexity of finding the rank of a three-dimensional tensor was proved by Johan Håstad in [11]:

Theorem 8 (Håstad) *Tensor rank is NP-complete for any finite field and NP-hard for the rational numbers.*

Let us now see how this fundamental property can be applied to our MP problem.

Let us suppose we have an algorithm Φ that solves the MP-problem in polynomial time (in particular, this algorithm can be used to know whether there exist a solution or not for some given instance of the MP-problem).

Let $T = (t_{ijk})$ be a $m \times n \times \ell$ (three-dimensional) tensor. It is well known (see for instance [18]) that the rank of T is exactly equal to the minimal number of multiplications needed to compute the following corresponding set of bilinear forms by a bilinear non-commutative algorithm:

$$(\mathcal{B}) \begin{cases} y_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij1} x_i x'_j \\ \quad \vdots \\ y_\ell = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij\ell} x_i x'_j. \end{cases}$$

Let r $(= rank(T))$ be this minimal value. By definition, r can be thought as the smallest integer u such that two linear transformations $s:\ (x_1,...,x_m,x'_1,...,x'_n) \mapsto (a_1,...,a_u,a'_1,...,a'_u)$ and $t:\ (b_1,...,b_u) \mapsto (y_1,...,y_\ell)$ exist, that transform

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

into $(\mathcal{B})$.

This is a particular instance of the MP problem. For $u = mn$, finding a solution (s,t) is quite easy. Namely, we can define s and t by the following formulas:

$$\forall i,\ 1 \leq i \leq m,\ \forall j,\ 1 \leq j \leq n,\ \begin{cases} a_{(i-1)n+j} = x_i \\ a'_{(i-1)n+j} = x'_j. \end{cases}$$

$$\forall k,\ 1 \leq k \leq \ell,\ y_k = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ijk} b_{(i-1)n+j}.$$

Note: Moreover, any symmetry of a given three-dimensional tensor leaves its rank unchanged, so that we have in fact:

$$r \leq \min(mn,nk,km),$$

which is the analogue of the classical inequality $rank(M) \leq \min(m,n)$ for a (two-dimensional) $m \times n$ matrix M.

The Φ algorithm can now be used to build a polynomial algorithm that computes the exact value of r:

1. Let $u = mn$.
2. With the help of Φ, try to find two linear transformations $s:\ (x_1,...,x_m,x'_1,...,x'_n) \mapsto (a_1,...,a_u,\ a'_1,...,a'_u)$ and $t:\ (b_1,...,b_u) \mapsto (y_1,...,y_\ell)$ that transform

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

   into $(\mathcal{B})$.
3. If Φ gives a solution, then replace u by $u-1$ and go back to step 2, else output "$r = u+1$".

It is easy to see that this algorithm outputs the correct value of r after at most mn calls to the (polynomial) Φ algorithm, so that we have built an algorithm that computes the rank of a three-dimensional tensor in polynomial time. According to the result of Johan Håstad mentioned above, we can thus conclude that the MP-problem is NP-hard for any finite field, and for the rational numbers.

Remarks:

1. More precisely, we have just proven that the "Deciding MP" problem (*i.e.* the problem of finding whether there exist a morphism between to given sets of multivariate polynomial equations) is NP-complete. As we will see in section 6, this property becomes false if we replace "MP" by "IP".

2. MP is clearly a very important problem in mathematics: an efficient algorithm would give the computation of the minimum number of standard multiplications to compute the product of two 3×3, 4×4 or $k \times k$ matrices, for small k, and – from this – improved algorithms for Gaussian reductions and related problems may be found. (The best known asymptotic algorithms are at the present in $\mathcal{O}(n^c)$, where $c \simeq 2.3755$, see [4]. It is also interesting to see what kind of brute-force computations have been done so far to solve such problems, see [10].)
3. The fact that MP is NP-hard, and moreover the fact that MP is a very important problem that seems to be very difficult even with very small parameters, are strong motivations to design cryptographic algorithms based on this problem. From [2] and [7], it is known that any problem of NP can be used to design an asymmetric authentication scheme (with the help of "good" hash functions). (The proof is extendable to design asymmetric signature schemes. again with the help of "good" hash functions). Since MP is in NP, we can apply these general results. However, these very general constructions are not very practical, and it may be more difficult to design efficient schemes from MP than from IP.

6 Deciding IP is not NP-complete

We call "Deciding IP" the problem of finding whether there exist an isomorphism between two sets of multivariate polynomials equations. In this section, we prove that the Deciding IP problem is not NP-complete, under the classical hypothesis that the so-called "polynomial-time hierarchy" does not collapse.

The proof is based on the following general results (see [9] and [3] for proofs):

Theorem 9 (Goldwasser, Sipser) *If a problem has a constant-round interactive proof, then it also has a constant-round Arthur-Merlin protocol.*

Theorem 10 (Boppana, Håstad, Zachos) *If the complement of a problem* Π *has an Arthur-Merlin protocol with a constant number of rounds, and if* Π *is NP-complete, then the polynomial-time hierarchy collapses.*

Note: Arthur-Merlin protocols have been studied by Babai and Moran in [1], but we do not need to go into further details for our proof.

What remains to do is building a constant-round interactive proof for the "Non Isomorphism of Polynomials" (Non-IP) problem. For that purpose, we use techniques discovered by Goldwasser, Micali and Rackoff ([8]) for the analogous "Quadratic Non-Residuosity" problem, and also used by Goldreich, Micali and Wigderson ([7]) for "Graph Non-Isomorphism", and by Petrank and Roth ([16]) for "Code Non-Equivalence".

As usual, we suppose that two sets (U_0) and (U_1) of polynomial equations are public, and an infinitely-powerful prover (called Merlin) wants to convince a polynomial-time verifier (called Arthur) that (U_0) and (U_1) are not isomorphic. They can proceed as follows:

First round: Arthur takes K numbers α_1, ..., α_K randomly chosen in {0,1}. For each k, $1 \leq k \leq K$, Arthur builds a set (V_k) which is isomorphic to (U_{α_k}), by choosing two random s_k and t_k bijective affine permutations and computing $(V_k) = t_k \circ (U_{\alpha_k}) \circ s_k$. He sends (V_1),...,(V_K) to Merlin.

Second round: For each k, $1 \leq k \leq K$, Merlin tries to guess α_k, *i.e.* he tries to find whether (V_k) is isomorphic to (U_0) or (U_1). He sends his guesses $(\beta_1,...,\beta_K)$ to Arthur.

Final verification: Arthur accepts the proof iff $\beta_k = \alpha_k$ for all k, $1 \leq k \leq K$.

It is easy to see that:

1. If (U_0) and (U_1) are non isomorphic, Merlin never fails in convincing Arthur.
2. If (U_0) and (U_1) are isomorphic, the probability that Arthur is convinced that (U_0) and (U_1) are non-isomorphic is at most 2^{-K}.

Part II: New algorithms for IP

We use the same notations as in section 2 (for u, n, q, $(\mathcal{A})$ and $(\mathcal{B})$). In sections 7 and 8, we will design some improved algorithms for the IP problem on two sets of u quadratic equations with n variables. Then, in section 10, we will concentrate on the special case $u = n$.

7 A first algorithm for IP

Let $\alpha_1, \ldots, \alpha_u$ be u values randomly chosen in $K = \mathbf{F}_q$. There is a probability $1/q^u$ that $\sum_{i=1}^{u} \alpha_i b_i = y_1 \circ s \quad (*)$.
Moreover when $(*)$ occurs it is very easy to find an affine bijection S transforming a into x such that $(*)$ does occur (when $(*)$ is written in a and x instead of b and y).
The reason for this is that there is a "canonical" representation of each equation of degree two over a finite field (cf. [12], chapter 6 for example), and from the canonical representations of $\sum_{i=1}^{u} \alpha_i b_i$ and y_1, when these equations are written in a and x, it will be easy to find such a bijection S.
What is the probability that this S is indeed the right s?
If $u \neq 1$, in the matrix of the secret affine function s we have $n(n+1)$ coefficients and $(*)$ gives only $1 + n(n+1)/2$ quadratic equations on these coefficients if $K = \mathbf{F}_2$ and $n(n+1)/2 + n + 1$ is $K \neq \mathbf{F}_2$. So the probability that we will recover the right s this way is expected to be approximately $1/q^{u-1}.q^{\frac{n^2+n}{2}}$ if $K = \mathbf{F}_2$ and approximately $1/q^{u-1} \cdot q^{\frac{n^2-n}{2}}$ if $K \neq \mathbf{F}_2$. Then, when s is found it is then easy to recover t by gaussian reductions on the coefficients of t.
The complexity of this algorithm to recover the secrets is expected to be $\mathcal{O}\left(q^{\frac{n^2+n+2u}{2}}\right)$ if $K = \mathbf{F}_2$ and $\mathcal{O}\left(q^{\frac{n^2-n+2u}{2}}\right)$ if $K \neq \mathbf{F}_2$ (because we try again another S until we find a valid solution). This is better than the complexity $\mathcal{O}(q^{n^2+n})$ of exhaustive search on s.

Note 1. If $u = 1$ then the first part of the algorithm will easily find an s solution (not necessary the right s) so u must be $\neq 1$.

Note 2. It is logical to try to improve this algorithm, for example by using two equations $\sum_{i=1}^{u} \alpha_i b_i = y_1 \circ s$ and $\sum_{i=1}^{u} \alpha'_i b_i = y_2 \circ s$. However, it is not clear how to combine these equations in order to improve the algorithm. More precisely, when $u \geq 2$, we do not know any polynomial algorithm for the IP problem. Even for $u = 2$, finding such an algorithm appears to be as difficult as finding a probabilistic polynomial algorithm for the Graph Isomorphism problem, as we saw in section 4.

8 A second algorithm for IP (found by H. Gilbert and F. Cherbonnier)

As before, let $\mathcal{A}$ be the quadratic transformation that gives b from a, and $\mathcal{B}$ be the quadratic transformation that gives y from x:

$$b = \mathcal{A}(a) \; ; \quad y = \mathcal{B}(x).$$

Let s and t be the two secret affine transformations such that $a = s(x)$ and $y = t(b)$. Therefore:

$$b = t^{-1} \circ \mathcal{B}(x) = \mathcal{A} \circ s(x).$$

The idea is to use the fact that the components of $\mathcal{A} \circ s$ are always linear combinations of the components of $\mathcal{B}$. Moreover, this is of course also true if we consider $\mathcal{A} \circ s$ only on a small number of variables.

Example Let $(\mathcal{A})$ and $(\mathcal{B})$ be the following two sets of equations:

$$(\mathcal{A}) \; : \; \begin{cases} b_1 = a_1 + a_1a_5 + a_2a_3 + a_2a_4 + a_3a_4 \\ b_2 = a_3 + a_4 + a_5 + a_1a_2 + a_1a_4 + a_4a_5 \\ b_3 = a_5 + a_1a_2 + a_1a_3 + a_1a_5 + a_2a_3 + a_3a_5 + a_4a_5 \\ b_4 = a_2 + a_3 + a_4 + a_5 + a_1a_5 + a_3a_4 + a_3a_5 \\ b_5 = a_4 + a_1a_3 + a_1a_5 + a_2a_3 + a_2a_4 + a_2a_5 + a_3a_4 + a_3a_5 + a_4a_5 \end{cases}$$

($(\mathcal{A})$ comes from $b = a^3$).

$$(\mathcal{B}) \; : \; \begin{cases} y_1 = x_1 + x_3 + x_4 + x_5 + x_1x_2 + x_1x_3 + x_1x_5 + x_2x_4 + x_3x_5 + x_4x_5 \\ y_2 = x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_2x_5 + x_4x_5 \\ y_3 = x_1 + x_3 + x_4 + x_1x_2 + x_1x_5 + x_2x_3 + x_3x_4 \\ y_4 = x_3 + x_4 + x_1x_2 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 \\ y_5 = x_2 + x_4 + x_5 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2x_5 \end{cases}$$

The problem is to find two linear transformations s and t such that $a = s(x)$ and $y = t(b)$.

Remark This example comes from a transformation made in [14] of the "toy example" given in [13].

We will choose $x_4 = 0$ and $x_5 = 0$ (so that we obtain a vector space of dimension $6 > 5$ generated by the monomials x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3). $(\mathcal{B})$ becomes:

$$(\mathcal{B}) \; : \; \begin{cases} y_1 = x_1 + x_3 + x_1x_2 + x_1x_3 \\ y_2 = x_2x_3 \\ y_3 = x_1 + x_3 + x_1x_2 + x_2x_3 \\ y_4 = x_3 + x_1x_2 \\ y_5 = x_2 + x_1x_3 + x_2x_3 \end{cases}$$

So the vector space generated by the equations of $(\mathcal{B})$ is the vector space generated by the 5 vectors $\{x_1, x_2, x_1x_3, x_2x_3, x_3 + x_1x_2\}$.

Let s be:

$$(s) \; : \; \begin{cases} a_1 = \alpha_1x_1 + \alpha_2x_2 + \alpha_3x_3 \; (+\alpha_4x_4 + \alpha_5x_5) \\ a_2 = \beta_1x_1 + \beta_2x_2 + \beta_3x_3 \; (+\beta_4x_4 + \beta_5x_5) \\ a_3 = \gamma_1x_1 + \gamma_2x_2 + \gamma_3x_3 \; (+\gamma_4x_4 + \gamma_5x_5) \\ a_4 = \delta_1x_1 + \delta_2x_2 + \delta_3x_3 \; (+\delta_4x_4 + \delta_5x_5) \\ a_5 = \varepsilon_1x_1 + \varepsilon_2x_2 + \varepsilon_3x_3 \; (+\varepsilon_4x_4 + \varepsilon_5x_5) \end{cases}$$

We then perform an exhaustive search on the α_i, β_i, γ_i, δ_i, ε_i $(1 \leq i \leq 3)$ values, *i.e.* there are 2^{15} possibilities for these values.

Example of value: We try

$$\begin{cases} a_1 = x_1 \ (+\alpha_4 x_4 + \alpha_5 x_5) \\ a_2 = x_2 \ (+\beta_4 x_4 + \beta_5 x_5) \\ a_3 = x_3 \ (+\gamma_4 x_4 + \gamma_5 x_5) \\ a_4 = 0 \ (+\delta_4 x_4 + \delta_5 x_5) \\ a_5 = x_1 + x_2 \ (+\varepsilon_4 x_4 + \varepsilon_5 x_5) \end{cases}$$

Then:

$$b_1 = x_1 x_2 + x_2 x_3.$$

However, this b_1 is **not** in the vector space generated by $\{x_1,x_2,x_1x_3,x_2x_3,x_3+x_1x_2\}$. As a result, this choice for the values α_i, β_i, γ_i, δ_i, ε_i $(1 \leq i \leq 3)$ can be eliminated.

Complexity: For about one try among 32, the five values b_i $(1 \leq i \leq 5)$ will be in the vector space generated by $\{x_1,x_2,x_1x_3,x_2x_3,x_3 + x_1x_2\}$ (because each value b_i has a probability about $\frac{1}{2}$ to be in this vector space). Then for these cases, we will consider $x_4 \neq 0$ and we will try all the values for α_4, β_4, γ_4, δ_4, ε_4.

This way, we may thus hope to find the good s (and t) after about 2^{15} tried values.

General complexity of the algorithm In general, $(\mathcal{A})$ and $(\mathcal{B})$ are two sets of the following type:

$$(\mathcal{A}) \ : \ \begin{cases} b_1 = P_1(a_1,...,a_n) \\ \quad \vdots \\ b_u = P_u(a_1,...,a_n) \end{cases} \quad \text{and} \quad (\mathcal{B}) \ : \ \begin{cases} y_1 = P'_1(x_1,...,x_n) \\ \quad \vdots \\ y_u = P'_u(x_1,...,x_n) \end{cases}$$

We will choose all the values $x_i = 0$, $i > \lambda$, where λ is such that:

$$\lambda + \frac{\lambda(\lambda-1)}{2} > u \qquad (\textit{i.e. } \lambda \simeq \sqrt{2u})$$

Therefore, the vector space generated by x_1^2, x_2^2, ..., x_λ^2, x_1x_2, ..., x_ix_j $(1 \leq i < j \leq \lambda)$, ..., $x_{\lambda-1}x_\lambda$ will have a dimension $> u$.

We will then perform an exhaustive search on the part of s that gives a_1, ..., a_n from x_1, ..., x_λ, *i.e.* on λn values. The complexity of this part 1 is thus expected to be about $q^{n\lambda} = q^{n\sqrt{2}\sqrt{u}}$.

Then for about one try among $\left(q^{\frac{\lambda^2+\lambda}{2}-u}\right)^u$, the u values b_i $(1 \leq i \leq u)$ will be in the right vector space. Then for these cases we will continue by introducing the value $x_{\lambda+1}$. This introduces q^n new possibilities, so that the typical value λ for the complexity of this part 2 of the algorithm is such that $\frac{\lambda^2+\lambda}{2} - u \geq \frac{n}{u}$ (*i.e.* $\lambda \simeq \sqrt{2}\sqrt{\frac{n}{u}+u}$).

Conclusion: The complexity of this algorithm is thus expected to be at most $q^{n\lambda} = q^{n\sqrt{2}\sqrt{\frac{n}{u}+u}}$.

Remarks

- When $u = n$, this is $q^{\mathcal{O}(n^{\frac{3}{2}})}$, which is better than the bound $q^{\mathcal{O}(n^2)}$ of the previous algorithms.
- When $u = 2$, this is also $q^{\mathcal{O}(n^{\frac{3}{2}})}$.

9 Some easy cases of IP with one secret

As seen in section 4, IP with one secret is at least as difficult as Graph Isomorphism. However, some instances of IP with one secret are easy to solve, as we will see now.

Let us consider for example the case of IP on two quadratic equations over a field of characteristic $p \neq 2$. We can write these equations as:

$$\begin{cases} q_1 = {}^t a A_1 a \\ q_2 = {}^t a A_2 a \end{cases} \quad \text{and} \quad \begin{cases} q_1 = {}^t x B_1 x \\ q_2 = {}^t x B_2 x \end{cases}$$

These two systems of two equations are public. The secret invertible matrix is S such that $x = Sa$. The problem is to find S. Let us assume that A_1 or A_2 is invertible (say A_1 for example). Since

$$\begin{cases} A_1 = {}^t S B_1 S \\ A_2 = {}^t S B_2 S \end{cases}$$

we have (since A_1 is invertible):

$$B_2 S = B_1 S A_1^{-1} A_2. \qquad (*)$$

This equation $(*)$ is of degree one in the values of the secret matrix S! So, from $(*)$ we will obtain some informations on the matrix S.

However, it is not clear how to obtain similar informations on S when A_1 and A_2 are not invertible (the case $p = 2$ may also create some technical difficulties, but it may be also feasible to find S when $p = 2$ and A_1 or A_2 is invertible).

Toy example 1: Let $K = \mathbf{F}_3$ and let the public equations be:

$$\begin{cases} q_1 = 2a_1^2 + a_1a_2 + 2a_1a_3 + a_2^2 + 2a_2a_3 \\ q_2 = a_1^2 + a_1a_3 + a_2^2 + 2a_3^2 \end{cases} \quad \text{and} \quad \begin{cases} q_1 = 2x_1^2 + x_1x_2 + 2x_1x_3 + x_2^2 + 2x_2x_3 \\ q_2 = 2x_1^2 + x_1x_2 + x_1x_3 + x_2^2 + 2x_2x_3 + 2x_3^2. \end{cases}$$

So $A_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 2 \end{pmatrix}$, $B_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $B_2 = \begin{pmatrix} 2 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 2 \end{pmatrix}$.

Here A_1 and A_2 are invertible.

Let $S = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{pmatrix}$.

By writing that $B_2 S = B_1 S A_1^{-1} A_2$, we find (after some Gaussian reductions):

$$\begin{array}{lll} \alpha_1 = 0, & \alpha_2 = \ ?, & \alpha_3 = 2\alpha_2, \\ \beta_1 = 2\alpha_2, & \beta_2 = 2\alpha_2, & \beta_3 = 0, \\ \gamma_1 = 0, & \gamma_2 = \ ?, & \gamma_3 = \alpha_2. \end{array}$$

So after a few tries on α_2 and γ_2, a solution can easily be found for S:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}.$$

Toy example 2: Let $K = \mathbf{F}_3$ and let the public equations be:

$$\begin{cases} q_1 = 2a_1^2 + a_1a_2 + 2a_1a_3 + a_2^2 + 2a_2a_3 \\ q_2 = 2a_1^2 + a_1a_3 + a_2^2 + 2a_3^2 \end{cases} \quad \text{and} \quad \begin{cases} q_1 = 2x_1^2 + x_1x_2 + 2x_1x_3 + x_2^2 + 2x_2x_3 \\ q_2 = 2x_1x_2 + x_1x_3 + 2x_2^2 + 2x_2x_3 + 2x_3^2. \end{cases}$$

So $A_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 2 \end{pmatrix}$, $B_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $B_2 = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{pmatrix}$.

Here A_1 is invertible, and A_2 is not invertible.

Let $S = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{pmatrix}$.

By writing that $B_2S = B_1SA_1^{-1}A_2$, we find (after some Gaussian reductions):

$$\begin{array}{lll} \alpha_1 = \ ?, & \alpha_2 = \ ?, & \alpha_3 = \ ?, \\ \beta_1 = \alpha_1 + \alpha_2 + 2\alpha_3, & \beta_2 = \alpha_1 + \alpha_3 & \beta_3 = \alpha_1 + 2\alpha_2 + 2\alpha_3, \\ \gamma_1 = \alpha_1, & \gamma_2 = 2\alpha_2 + 2\alpha_3, & \gamma_3 = \alpha_2. \end{array}$$

So after a few tries on α_1, α_2, α_3, a solution can easily be found for S:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}.$$

10 Algorithms for IP with two secrets in $\mathcal{O}(q^{\mathcal{O}(n)})$

Further investigation of the IP problem requires to divide it into smaller subproblems. Thus, we separate the one and two secret cases, and more precisely we limit ourselves to the central case $u = n$. We call this case central because the case $u = 1$ is easily solved as most cases $u > (n+1)(n+2)/2$. As a matter of fact it becomes easy as soon as some $(n+1)(n+2)/2$ equations are independent (as a formal sum of x_ix_j coefficients), so that the u quadratic equations form a generating set. The probability of being so rapidly tends to 1 as u grows. In that case **any** invertible s allow to find at least one t by Gaussian reduction and **every** two generating sets of equations are isomorphic.

We also found interesting to separate the affine part of applications s and t, as the linear case seems a bit easier. However, since it is always possible to find the constant terms by exhaustive search in $\mathcal{O}(q^n)$, any algorithm for the IP problem with linear s and t with complexity $\mathcal{O}(q^{\alpha n})$ can obviously be transformed into a general algorithm for affine s and t with complexity $\mathcal{O}(q^{(\alpha+1)n})$.

So from now on and throughout the present section we exclusively consider the IP problem with two secret linear s and t transformations.

As a consequence of separation between affine and linear parts, we should suppose that $\mathcal{A}(0) = 0$ and $\mathcal{B}(0) = 0$. Otherwise we would have a completely artificial weakness, as we know that $t(\mathcal{A}(0)) = \mathcal{B}(0)$ and it gives an *a priori* knowledge on t.

The present section is divided into three parts (in these three parts, s and t are assumed to be linear). First, we will see a simple algorithm solving most cases of the IP in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and using $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ memory. Then we will explain a very different approach which uses only polynomial memory and runs in between $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ for all IP cases. Finally we will try to combine the ideas of these two attacks in a very powerful, 'birthday paradox'-based

attack which to the best of our knowledge runs for every IP in $n^{\mathcal{O}(1)}\mathcal{O}(q^{n/2})$ with $n^{\mathcal{O}(1)}\mathcal{O}(q^{n/2})$ memory. Only the last attack uses the fact that equations given in IP are quadratic forms. The first two can operate on any function of indifferent degree.

10.1 A simple attack in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ based on inversion

This attack operates in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and uses $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ of memory space used essentially to form a complete table of $\mathcal{A}$ and $\mathcal{B}$ functions.

Let $\mathcal{A}$ be a randomly chosen quadratic equations set. We studied a probability p_i of a random value b of the $\mathcal{A}$ form to have i possible corresponding entries a. We made a lot of computer simulations and we have found that the probability distribution of p_i for a randomly chosen set of quadratic equations is the same as for any randomly chosen function, which can be easily shown to be $p_i = \frac{1}{ei!}$.

It allows to divide entries into classes, following their image's inversion degree. The elements of each class can be summed up and the corresponding values for $\mathcal{A}$ and $\mathcal{B}$ must correspond through the linear transformation s. They do not necessarily sum up to 0, since as $p_i \to 0$ we have very small classes and for a random $\mathcal{A}$ they should not sum to 0.

It is easy to show that if $n \leq q$ we would get more than n equations and we could recover s by Gaussian reduction. With less than n equations we iterate the process as follows:

If we have found one equation $s(x^{(0)}) = a^{(0)}$ we can increase the number of classes in our partition of initial values space. We use the fact that if $s(x) = a$, we also have $s(x+x^{(0)}) = a+a^{(0)}$. Therefore we classify a not only by the inversion degree of its image $\mathcal{A}(a)$ as before, but we will also consider the class of $a + a^{(0)}$. The number of classes will systematically increase.

This attack will work for any sensible non-bijective (> 2 classes) quadratic equations set. Yet we cannot solve a toy example given at the beginning of the article, since it is bijective. In order to solve these cases we hit upon another approach.

10.2 The "to and fro" attack

We will describe the attack on the toy example from the chapter 2, showing directly how it works.

Let $q = 2$ and $n = 5$. We consider

$$\mathcal{A}: \begin{cases} b_0 = a_0 + a_0a_4 + a_1a_2 + a_1a_3 + a_2a_3 \\ b_1 = a_2 + a_3 + a_4 + a_0a_1 + a_0a_3 + a_3a_4 \\ b_2 = a_4 + a_0a_1 + a_0a_2 + a_0a_4 + a_1a_2 + a_2a_4 + a_3a_4 \\ b_3 = a_1 + a_2 + a_3 + a_4 + a_0a_4 + a_2a_3 + a_2a_4 \\ b_4 = a_3 + a_0a_2 + a_0a_4 + a_1a_2 + a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \end{cases}$$

and

$$\mathcal{B}: \begin{cases} y_0 = x_0 + x_2 + x_3 + x_4 + x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_2x_4 + x_3x_4 \\ y_1 = x_0x_3 + x_0x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 \\ y_2 = x_0 + x_2 + x_3 + x_0x_1 + x_0x_4 + x_1x_2 + x_2x_3 \\ y_3 = x_2 + x_3 + x_0x_1 + x_0x_4 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_4 = x_1 + x_3 + x_4 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_1x_4 \end{cases}$$

as functions $\mathbf{F}_{2^5} \to \mathbf{F}_{2^5}$.

We have made a table of functions $\mathcal{A}$ and $\mathcal{B}$, where for simplification purposes we denote $\mathbf{F}_{2^5}$ elements by integers, so that to $x = (x_4,x_3,x_2,x_1,x_0)$ corresponds $x = \sum_{i=0}^{n} x_i 2^i$. Therefore if we read in the table that $\mathcal{A}(5) = 31$ it means that if $a = (a_4,a_3,a_2,a_1,a_0) = (0,0,1,0,1)$ then $x = s(a)$ equals to (1,1,1,1,1).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	2 8	29	30	31
$\mathcal{A}$	0	1	8	15	10	31	23	4	26	25	3	6	9	30	5	20	14	18	22	12	24	16	21	27	2	28	11	19	13	7	17	29
$\mathcal{B}$	0	5	16	24	13	25	11	18	29	10	30	4	28	26	9	2	17	27	19	20	21	14	1	23	7	31	22	3	15	6	8	12

In the real application we will not construct any table, and we will proceed with exhaustive search at every time we need to compute $\mathcal{A}^{-1}$ or $\mathcal{B}^{-1}$ which happens $\mathcal{O}(n)$ times in this attack.

The main idea is to say that if we have k equations on s:

$$\begin{cases} s(x^{(1)}) &= a^{(1)} \\ & \vdots \\ s(x^{(k)}) &= a^{(k)} \end{cases}$$

that are linearly independent, we have $q^k - 1$ dependent equations

$$\begin{cases} s(\sum \sim x^{(i)}) &= \sum \sim a^{(i)} \\ & \vdots \end{cases}$$

– where $\sim$ denotes some coefficients. But since $\mathcal{A}$ is generally non-linear we can get from that as much as (or almost) $q^k - 1$ independent equations 'on the other side':

$$\begin{cases} \mathcal{A}\left(s(\sum \sim x^{(i)})\right) &= \mathcal{A}\left(\sum \sim a^{(i)}\right) \\ & \vdots \end{cases}$$

Then we apply t (formally), which changes nothing as to linear independence of these equations:

$$\begin{cases} t\left(\mathcal{A}(s(\sum \sim x^{(i)}))\right) &= t\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots \end{cases}$$

And finally, since $\mathcal{B} = t \circ \mathcal{A} \circ s$ we have about $q^k - 1$ implicit equations on t:

$$\begin{cases} \mathcal{B}(\sum \sim x^{(i)}) &= t\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots \end{cases}$$

We can do that the other direction, start from equations on t and get s in a very similar way, but it is less convenient since we need to compute $\mathcal{A}^{-1}$ or $\mathcal{B}^{-1}$ and sum up all possible solutions. In our example it was easy, since it is bijective.

The whole method is called "to and fro" as it proceeds toing and froing from one side to another and ends when we get n independent equations on s and another n on t.

If $q \neq 2$ we have $q^k - 1 >> k$ and the attack complexity is about $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ at most to find initial 1 or 2 equations.

If $q = 2$ it is $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ in most cases (no exceptions are known) since we can start with 2 equations.

Let us see how it works on our example:
We start with the two following equations:

$$\begin{cases} s(1) = 1 \\ s(2) = 7 \end{cases}$$

Our equation set $\mathcal{A}$ comes from $b = a^3$ in the $\mathbf{F}_{2^n}$, $n = 5$ field, and it has $n \cdot 2^n$ automorphisms of the form:

$$\begin{cases} s : x \mapsto \alpha x^{1+2^\beta} \\ t : x \mapsto \alpha^{-1-2^\theta} x^{2^{-\beta}} \end{cases} \quad \text{with } \alpha \neq 0,\ \beta \in \{0,1,\ldots,n-1\}.$$

Therefore there are at least $n2^n$ solutions s and t (in fact we have found there are no more). Thus the probability of our 2 equations being right is $\frac{n2^n}{2^{2n}} = \frac{n}{2^n}$ and the whole attack complexity is only $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$. Moreover, for $n = 5$ one gets $\frac{n}{2^n} \sim 0.15$ and we admit to have been lucky choosing two starting equations right. Then we get 3 dependent equations:

$$\begin{cases} s(1) = 1 \\ s(2) = 7 \\ s(3) = 6 \end{cases}$$

as – in $\mathbf{F}_{32}$ – $1 + 2 = 3$ and $1 + 7 = 6$.

Then we use the table to 'transfer' equations on the other side:

$$\begin{cases} t(1) = 5 \\ t(4) = 16 \\ t(23) = 24 \end{cases}$$

For example with $s(2) = 7$, we get $\mathcal{A}(7) = 4$ and $\mathcal{B}(2) = 16$ and all that implies that $t(4) = 16$.

Now we combine them again to get $2^3 - 1 = 7$ dependent equations:

$$\begin{cases} t(1) = 5 \\ t(4) = 16 \\ t(5) = 21 \\ t(18) = 13 \\ t(19) = 8 \\ t(22) = 29 \\ t(23) = 24 \end{cases}$$

And using $\mathcal{A}^{-1}$ and $\mathcal{B}^{-1}$ we get 7 equations on s:

$$\begin{cases} s(1) = 1 \\ s(2) = 7 \\ s(3) = 6 \\ s(4) = 17 \\ s(8) = 18 \\ s(20) = 14 \\ s(30) = 27 \end{cases}$$

6 from those 7 equation are actually independent, and yet it is enough to recover s by Gaussian reduction. Similarly we get t and verify the correctness of our solution.

$$(a_0,a_1,a_2,a_3,a_4) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} (x_0,x_1,x_2,x_3,x_4)$$

$$(y_0,y_1,y_2,y_3,y_4) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} (b_0,b_1,b_2,b_3,b_4)$$

If we look closely at the "to and fro" attack we can see that it can be transformed into an 'birthday paradox' or 'meet in the middle' attack with all the related speed-memory tradeoffs. This is possible because in our attack we can operate as follows:

1. First, we start with given 2 or 1 values on entries of $\mathcal{A}$ we write on the one side, and the corresponding entries for $\mathcal{B}$ we write on the other side.
2. Then on each side we compute new values as before by successive linear transformations, applying $\mathcal{A}$ or $\mathcal{A}^{-1}$ and $\mathcal{B}$ or $\mathcal{B}^{-1}$, again linear transformations etc. Thus we obtain new values which still match the respective values on the other side. We call this phenomenon "boosting" as the initial information is amplified.
3. At the end we look for a linear s that maps these two (ordered) sets.

Now in order to transform the attack into the 'birthday paradox'-based attack, we need only to modify part 3. Let us assume we have $n+\varepsilon$ related entries $a^{(i)}$ for $\mathcal{A}$ and on the other side $n+\varepsilon$ related entries $x^{(i)}$ for $\mathcal{B}$. We do not need to compute s to know they are really related by s. There is a different way: we find separately how the $n+\varepsilon$ related entries $a^{(i)}$ are linearly dependent between **themselves** and we do the same for the $n+\varepsilon$ entries $x^{(i)}$ of $\mathcal{B}$. Two sets which are related by s must have the same linear dependencies. Now we can easily detect a collision in two huge lists of sets looking for the same dependencies.

The probability of error can be made as small as we want since $n+\varepsilon$ have at least ε linear dependencies, and the probability of them being all true for a linearly non-related set of data grows very small with ε: $1/q^{\varepsilon n}$.

Our next attack explores this 'birthday' approach with several improvements, as we want to insure the possibility of starting all cases of "to and fro" with one equation and we do not want to do any exhaustive search for inversion of $\mathcal{A}$ or $\mathcal{B}$.

10.3 Combined power attack.

In order to further improve our attacks we found out that it would be nice to have a function which we call "boosting function" with the following properties: on an entry x we compute in

polynomial time $x' = F(x)$ such that F is preserved by the isomorphism of polynomials:

$$\textit{If } s(x) = a,\ F_{\mathcal{B}}(x) = x' \textit{ and } F_{\mathcal{A}}(a) = a',\ \textit{then } s(x') = a'.$$

However:

Theorem 11 *There is no non-trivial boosting function which works for all the quadratic equations sets.*

Proof: The "to and fro" attacks will almost always, apart from improbable cases, allow to recover in a unique, deterministic way, the entire s and t starting from 2 equations on s. Yet, a non-trivial boosting function would allow, with some non-zero probability, to do the same starting with only one equation.

Now, in the very precise equations set derived from $b = a^3$ it is not possible. From the structure of $b = a^3$ automorphisms shown in the last chapter we can easily see that there are at least n possible solutions s and t satisfying any given equation on s.

The situation is not that bad though, since in most $q \neq 2$ cases the very simple function:

$$F(z) = \mathcal{B}(rz), \text{ with } r \in \mathbf{F}_p$$

can give few equations on t with one equation on s.

Therefore we must look for less powerful functions and in some way relax its axioms. We have found few ways of doing so:

1. We may consider more general functions, dealing with different sets of information on entries and results of a function.
2. Moreover we may use different boosting functions depending on the particularities $\mathcal{A}$.
3. We ask the condition of $x' = F(x)$ being preserved by the isomorphism to be true only with a certain noticeable probability.
4. We can have a function F probabilistic.
5. It is enough to suppose that $x' = F(x)$ is preserved no more by **all**, but by at least by **one** possible isomorphism that satisfies the initial data.

The last two additions are important, since they are those which do demolish our proof of non-existence of boosting function.

We call a boosting function or "weak boosting function" a function which satisfies 1.-5., and the previous definition is referred to as a "strong boosting function".

In the following attack description we used 3 different boosting functions F, G and H to cover different cases of quadratic forms set $\mathcal{A}$. All these function are based on the same principle of 'differential solving' described as follows:

Since we have quadratic equations in $\mathcal{A}$, it is possible to perform a 'differential solving' of these equations, *i.e.* given a pair c and d one can find two entries a and a' such that $a - a' = c$ and $\mathcal{A}(a) - \mathcal{A}(a') = d$. Here is how we can proceed: we write the polar version of the $\mathcal{A}$ form set:

$$\mathcal{Q}(a,c) = \mathcal{A}(a + c) - \mathcal{A}(a) - \mathcal{A}(c)$$

which is bilinear in the x_i and the y_i. Then the equation we need to solve:

$$\mathcal{A}(a + c) - \mathcal{A}(a) = d$$

is the same as

$$\mathcal{Q}(a,c) = \mathcal{A}(c) + d$$

which – once c and d are fixed – contains only linear equations on the a_i, and allows us to compute by Gaussian reduction all its solutions.

The differential solving, as described above, leads to the following boosting function:

$$Function(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)}} x.$$

It works fine for $p \neq 2$, but it fails for $p = 2$. The problem is that if a is a solution, $a + c$ is a solution as well, and we cannot either easily tell them apart, either sum them up, since $a + (a + c)$ gives a, which is not a new data. We will take one of these values, at random, in every pair $(x,x + z)$. The sum will be known with possible $(+z)$, and exactly with probability $1/2$ which is not bad. We put:

$$F_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x) \\ \text{one only } \in\{x,x+z\}}} x.$$

The function is defined the same way for $\mathcal{A}$:

$$F_{\mathcal{A}}(c) = \sum_{\substack{a \text{ such that} \\ \mathcal{A}(a+c)=\mathcal{A}(a) \\ \text{one only } \in\{a,a+c\}}} a.$$

It is clear that the boosting function F works only for non-bijective forms. For bijective forms we use another boosting function based on the same principle:

$$G_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\mathcal{B}(z) \\ \text{one only } \in\{x,x+z\}}} x.$$

Finally, because of the theorem we showed, neither of F and G can work in the particular case of C^* cryptosystem quadratic forms. We use then:

$$H_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\alpha_{\mathcal{B}} \\ \text{one only } \in\{x,x+z\}}} x.$$

with $\alpha_{\mathcal{B}}$ chosen at random, but as a function of $\mathcal{B}$, so it is always the same for the same $\mathcal{B}$. We do the same for $\mathcal{A}$. The H works because it restricts the number of solutions s we are going to find to those for which $t(\alpha_{\mathcal{A}}) = \alpha_{\mathcal{B}}$ and there are still enough of them.

In [5] we describe another boosting function K which could be useful in case the present should fail on some particular quadratic form set.

We now describe the whole attack:

The combined power attack

This attack is designed to solve **all** the IP cases, $u = n$ and with linear s and t transformations, with the complexity $n^{\mathcal{O}(1)}q^{n/2}$ and using $n^{\mathcal{O}(1)}q^{n/2}$ of space. We do not know any particular function which would resist it.

1. *Initialisation.*
   We pick at random $(\log n)^{\mathcal{O}(1)}q^{n/2}$ entries $z^{(i)}$ and $(\log n)^{\mathcal{O}(1)}q^{n/2}$ entries $c^{(i)}$ (the same can be used). Thus by the well known 'birthday principle' there are at least $(\log n)^{\mathcal{O}(1)}$ collisions such that $s(z^{(i)}) = c^{(j)}$.

2. Starting (1 value $\mapsto \log\log n + \mathcal{O}(1)$ values):
We put $z^{(i,0)} = z^{(i)}$ et $c^{(i,0)} = c^{(i)}$. Then we apply $\log\log n + \mathcal{O}(1)$ times an appropriate boosting function:
$$z^{(i,j+1)} = F_{\mathcal{B}}(z^{(i,j)})$$
and
$$c^{(i,j+1)} = F_{\mathcal{A}}(c^{(i,j)})$$
(eventually replace F by G or H.)

   On this stage, if any of the $F_{\mathcal{B}}(z^{(i,j)})$ or $F_{\mathcal{A}}(c^{(i,j)})$ is zero, we throw away the initial value $z^{(i)}$ or $c^{(i)}$. Thus we keep 1 out of about $\mathcal{O}(1)^{\log\log n+\mathcal{O}(1)} = (\log n)^{\mathcal{O}(1)}$ initial entries which is still enough.

   If $p = 2$ our boosting functions values are randomly chosen from 2 possibilities and there are few possible lists $(z^{(i,j)}, j = 1..)$. Still, the number of possible lists is small: $2^{\log\log n+\mathcal{O}(1)} = \mathcal{O}(\log n)$.

   Having thrown away some values, we have still about $\mathcal{O}(\log n)$ collisions $s(z^{(i)}) = c^{(j)}$. And even if $p = 2$, we can expect about $\mathcal{O}(1)$ to have made the same choices in probabilistic functions F, G, H. Finally, there are at least $\mathcal{O}(1)$ pairs $(z^{(i)},c^{(j)})$ such that
$$\forall k = 1..(\log\log n + \mathcal{O}(1)), \quad S(z^{(i,k)}) = c^{(j,k)}$$

3. *Pre-Expansion* ($\log\log n + \mathcal{O}(1)$ *input values* $\mapsto \log n + \mathcal{O}(1)$ *output values):*
We apply simply $\mathcal{B}$ to all possible linear combinations of the $z^{(i)}$and we get about $q^{\log\log n+\mathcal{O}(1)} > \log n + \mathcal{O}(1)$ new values we note $t^{(i,j)}$, $j = 1..(\log n + \mathcal{O}(1))$.
We can suppose that at least $\log n + \mathcal{O}(1)$ are independent, and we take care to have them always written in the same fixed order.
We do the same for $\mathcal{A}$ and we get the $d^{(i,j)}$ list.

4. *Expansion* ($\log n + \mathcal{O}(1)$ *output values* $\mapsto n + \mathcal{O}(1)$ *input values):*
For every candidate i we apply again the 'differential solving' principle for $\mathcal{B}$. The input difference will always be the same $z^{(i,0)}$. For the output differences we will take all the possible linear combinations of the $t^{(i,j)}$, $j = 1..(\log n + \mathcal{O}(1))$.

   Thus we get about $q^{\log n+\mathcal{O}(1)} > n + \mathcal{O}(1)$ values defined as follows:
$$\sum_{\substack{\text{the } x \text{ such that} \\ \mathcal{B}(x+z)-\mathcal{B}(x)=\sum\limits_{j}\sim t^{(i,j)} \\ \text{one only } \in\{x,x+z\}}} x.$$
with $\sim$ being some coefficients $\in \mathbf{F}_q$.
From these values we keep $n + \varepsilon$, $\varepsilon \in \mathcal{O}(1)$ which again need to be taken always in the same ordering. We call them $z^{(i,j)}$, $j = 1..(n + \varepsilon)$.

   We do the same for $\mathcal{A}$ and it gives $c^{(i,j)}$, $j = 1..(n + \varepsilon)$.

5. *Label distribution.*
For each list $z^{(i,j)}$, $j = 1..(n + \varepsilon)$ we are going to detect by Gaussian reduction all the possible linear dependencies. In order to easily manage the set of dependencies we will proceed in a way to get only independent linear equations, as follows: for $j = 1,2,\ldots$ we seek all the linear dependencies between $z^{(i,j)}$ and those of the $z^{(i,k)}$, $k < j$ which are non-zero and have not been found linearly dependent before.

This list of linear dependencies, having between 2 and $n+\varepsilon$ members is a label, a signature we give to the list $z^{(i,j)}$, $j = 1..$, and actually to the sole value $z^{(i,0)}$.

As we explained before (5.2), if two lists, one for $\mathcal{A}$ and another for $\mathcal{B}$, are related by the linear application s, they will have the same label. If they are not, the error probability is as small as $1/q^{\varepsilon n}$. Furthermore $\varepsilon = 2$ is enough.

6. *Collision detection.*
   If $p \neq 2$ we simply look for two same labels.

   If $p = 2$, two lists $z^{(i,k)}$, $k = 1..$ and $c^{(j,k)}$, $k = 1..$ can be still presumed to correspond through s for some i, j to be found, but imperfectly known: with possible $+z^{(i,0)}$ for the first list and $+c^{(i,0)}$ for the second.

   The probability of a $n + \varepsilon + 1$ elements list to have the same label as if some randomly chosen elements would have been added the same value $z^{(i,0)}$ is quite big: about $\frac{1}{2^\varepsilon}$ as we expect to have ε linear relations in a list and it is $1/2$ for one relation. Therefore we expect that with the probability about $\frac{1}{2^{2\varepsilon}}$ we can detect if two $z^{(i,0)}$ and $c^{(j,0)}$ are related by s. Again $\varepsilon = 2$ is enough and we have $\frac{1}{2^{2\varepsilon}} = \frac{1}{16}$. It means that for $p = 2$ one in 16 s functions we find in the present attack is correct.

   Finally, in all cases we simply search for i and j such that $z^{(i,k)}$, $k = 0..(n + \varepsilon)$ and $c^{(j,k)}$, $k = 0..(n + \varepsilon)$ have the same label.
7. *Computing of s.*
   We can then compute s by Gaussian reduction on $> n$ equations on s: $s(z^{(i,k)}) = c^{(j,k)}$, $k = 0..(n + \varepsilon)$.
8. *Ending step.*
   From s we easily get t and we verify the solution. Eventually, if $p = 2$, we need to try about 16 collisions.

We do not know any example of quadratic equations for which all the three variations of the attack would fail altogether. We did not program the whole attack so far, but we did testing on particular points that seemed to be uncertain.

11 Suggestions of IP variations

IP with one secret

As we have seen above, some improved algorithms exist for IP with two secrets. Therefore, when IP is used for authentication or signature as explained in [15], it might be suggested to use IP with one secret instead of IP with two secrets in order to have a more efficient scheme. It may look surprising that IP with one secret might be a more difficult problem (with practical values of the parameters) than IP with two secrets. However, this is not so surprising: in IP with two secrets, we have about $2n^2$ unknown coefficients of K (the secret values of the s and t matrices) and about $\frac{n^3}{2}$ quadratic equations on these unknown values (when we formally identify the two sets of equations $(\mathcal{A})$ and $(\mathcal{B})$), *i.e.* much more equations than unknowns. However, in IP with one secret, the parameters can be chosen in order that the number of equations (given by a formal identification) will be about approximately equal to the number of unknowns. This occurs for instance when $(\mathcal{A})$ and $(\mathcal{B})$ are two sets of two quadratic equations. (However, in section 9, we have seen that some instances of this problem are easy to solve.) As a result, despite the fact that there is only one affine change of variables, such a problem might be more difficult than the IP with two secrets. (IP with one secret on a single cubic equation might also be of interest.)

Subgroups of $GL_n(K)$

Another possible idea would be to choose the secret transformations s and t of IP with two secrets in a subgroup G of $GL_n(K)$. ($GL_n(K)$ is the set of all linear bijective transformations from K^n to K^n). This way, n may be chosen rather large (in order to make the problem difficult) and the length of the asymmetric signature (when the problem is used as explained in [15] for asymmetric signatures) might still be of reasonable size. For instance, G might be the orthogonal group of some quadratic form q (*i.e.* the set of all $g \in GL_n(K)$ such that $\forall x \in K^n$, $q(g(x)) = q(x)$), or G might be the group of all matrices of the form $\begin{pmatrix} A & -B \\ B & A \end{pmatrix}$, where A and B are two $\frac{n}{2} \times \frac{n}{2}$ matrices. It is not clear whether choosing s and t in such a subgroup G makes it easier to solve the IP problem or not.

Remark: A similar idea is used in the DSS or in Schnorr's algorithm, where – in order to have shorter length for the signatures – a subgroup is chosen for the exponents used in these schemes. However, here, the schemes are very different.

12 Conclusion

In this paper, we have shown that the MP and IP problems are not only related to the problem of finding the secret key of esoteric cryptographic algorithms, but also to very general problems such as Graph Isomorphism and Fast Matrix Multiplication. From their definitions, MP and IP look very similar but, as we have seen in this paper, Deciding MP is NP-complete whereas Deciding IP is not NP-complete (assuming the usual hypothesis about Arthur-Merlin games).

If $\mathcal{O}(n^2)$ is the length of the secrets to be found, then our new algorithms for IP ("with two secrets") have a complexity in $\mathcal{O}(q^n)$ or $\mathcal{O}(q^{n/2})$ when s and t are linear (so at most $\mathcal{O}(q^{\frac{3}{2}n})$ when s and t are affine). This is not polynomial, but this is clearly much better than the complexity $\mathcal{O}(q^{(n^2)})$ of the previous algorithms. For example, for the toy example given in [13], we have found the secret key with only about 7 computations (!), instead of 2^{25} for previously known algorithms (or 2^{50} for exhaustive search on the two secret matrices s and t). This means that the schemes based on IP problems with two secrets (such as some the schemes described in [15]) should have larger values of the parameters than expected for security. However, there are a lot of possible variations for IP and the choice of the variations and the choice of the parameters can still be done in order to have both efficient asymmetric schemes and no practical attacks.

Acknowledgements: We thank Henri Gilbert and Frédéric Cherbonnier for giving their algorithm of section 8. We also want to thank Jean-Pierre Seifert for pointing out references [11] and [16] to us, and Don Coppersmith for pointing out a possible link between morphisms of polynomials and fast matrix multiplication.

References

[1] László Babai, Shlomo Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes*, JCSS, vol. 36, 1988, pp. 254-276.

[2] Manuel Blum, *How to prove a theorem so no one else can claim it*, Proceeedings of the International Congress of Mathematics, Berkeley CA, 1986, pp. 1444-1451.

[3] Ravi B. Boppana, Johan Håstad, Stathis Zachos, *Does co-NP have short interactive proofs*, Information Proc. Letters, vol. 25, 1987, pp. 127-132.

[4] Don Coppersmith, Shmuel Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Computation (1990), **9**, pp. 251-280.

[5] Nicolas Courtois, *Les cryptosystèmes asymétriques à représentation obscure*, Rapport de DEA, Paris 6 University, July 1997.

[6] Scott Fortin, *The Graph Isomorphism Problem*, Technical Report 93-20, University of Alberta, Edmonton, Alberta, Canada, July 1996. This paper is available at ftp://ftp.cs.ualberta.ca/pub/TechReports/1996/TR96-20/TR96-20.ps.gz

[7] Oded Goldreich, Silvio Micali, Avi Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, Journal of the ACM, v. 38, n. 1, Jul. 1991, pp. 691-729.

[8] Shafi Goldwasser, Silvio Micali, Charles Rackoff, *The knowledge complexity of interactive proofs*, SIAM J. Comput., vol. 18, 1989, pp. 186-208.

[9] Shafi Goldwasser, Michael Sipser, *Private coins vs. public coins in interactive proof systems*, Advances in Computing Research, S. Micali (Ed.), vol. 5, 1989, pp. 73-90.

[10] John Gustafson, Srinivas Aluru, *Massively Parallel Searching for Better Algorithms or, How to Do a Cross Product with Five Multiplications*, Ames Laboratory, Department of Energy, ISU, Ames, Iowa. This paper is available at http://www.scl.ameslab.gov/Publications/FiveMultiplications/Five.html

[11] Johan Håstad, *Tensor Rank is NP-Complete*, Journal of Algorithms, vol. 11, pp. 644-654, 1990.

[12] Rudolf Lidl, Harald Niederreiter, *"Finite Fields"*, *Encyclopedia of Mathematics and its applications*, Volume 20, Cambridge University Press.

[13] Tsutomu Matsumoto, Hideki Imai, *Public quadratic polynomial-Tuples for efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[14] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, CRYPTO'95, Springer-Verlag, pp. 248-261.

[15] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*, EUROCRYPT'96, Springer-Verlag, pp. 33-48.

[16] Erev Petrank, Ron M. Roth, *Is Code Equivalence Easy to Decide?*, IEEE Transactions on Information Theory, 1997.

[17] Volker Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik 13, 1969, pp. 354-356.

[18] Volker Strassen, *The asymptotic spectrum of tensors*, J. Reine Angew. Math., vol. 384, pp. 102-152, 1988.

C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai

ASIACRYPT'98 *(Version complète)*
Article avec Nicolas Courtois (Bull SC&T) et et Jacques Patarin (Bull SC&T)

Abstract

In [4], H. Imai and T. Matsumoto presented some new candidate trapdoor one-way permutations with a public key given as multivariate polynomials over a finite field. One of these schemes was later presented in [8] under the name C^*, and was based on the idea of hiding a monomial field equation. This scheme was broken in [9] by Jacques Patarin, due to unexpected algebraic properties. J. Patarin and L. Goubin then suggested ([10], [11], [12], [13]) some schemes to repair C^*, but this was done at the cost of slightly more complex public key or secret key computations. In part I of this paper, we will study some very simple variations of the C^* scheme, where the attack of [9] is avoided, and where the very simple secret key and public key computations are almost kept. The C^*_{-+} scheme will be one of these variations. We will design some new cryptanalysis that are efficient against some of – but not all – these variations.

Another scheme of [4], very different from C^* (despite the name), was called [C] and was based on the idea of hiding a monomial matrix equation. No cryptanalysis has been published so far for this scheme. In part II of this paper, we will show how to attack this scheme [C]. We will then study more general schemes, still using the idea of hiding matrix equations. The HM scheme will be one of these variations.

1 Introduction

What is – at the present – the asymmetric signature algorithm with the most simple smartcard implementation (in terms of speed and RAM needed), and not broken?

We think that it is one simple variation of the Matsumoto-Imai C^* algorithm that we will present in the part I of this paper. The C^* algorithm was presented in [4] and [8], and was broken in [9], due to unexpected algebraic properties. However, it is possible to imagine many ways of avoiding the cryptanalysis of [9].

In [10], J. Patarin suggested to use a "hidden polynomial" instead of a "hidden monomial". These "HFE" algorithms are still unbroken. However, the secret key computations in HFE schemes are sensibly more complex than in the original C^* scheme. In [11], [12] and [13], J. Patarin and L. Goubin also studied some variations, where the public equations are given in different forms (some of these schemes are also presented in [6]), but here again, in order to avoid the attacks, the secret key computations or the public key computations are generally slightly more complex than in the original C^* scheme.

In part I of this paper, we will design and study a few very simple variations of the original C^* scheme. We will keep a quadratic public key and the main secret key operations will still be the computation of a monomial function $f : x \mapsto x^h$ in a finite field. (The length of the elements of this finite field is much shorter than what we have in RSA, and this explains why the implementations are much more efficient.)

Some of the new variations will be broken in this paper. However, as we will see, there are still some very simple and efficient variations that we do not know how to break. These schemes are related to some problems of orthogonal polynomials (how to complete a set of orthogonal polynomials, how to eliminate some random polynomials linearly mixed with orthogonal polynomials, etc).

It can be noticed that all the very simple transformations that we will study in the case of the C^* scheme, can also be applied to the more general HFE scheme of [10] or to Dragon schemes of [11], or to the *HM* scheme. We concentrate on C^* because the secret computations of C^* are particularly efficient, and because we wanted to see if these simple ideas could be sufficient or not to enforce the security (in HFE, the analysis is more difficult since no efficient attacks are known at the present).

In part II of this paper, we will study a very different (despite the name) algorithm of [4], called $[C]$. It is based on the idea of hiding (with secret affine transformations) a monomial matrix equation. Since the multiplication of matrices is a non-commutative operation, it creates a scheme with very special features. However, as in C^* or HFE, the public key is still given as a set of multivariate polynomials on a finite field, and some of the ideas used in [9] will also be useful.

We will show how to break the original $[C]$ sheme (no cryptanalysis of this scheme was published before). We will then study some more general schemes, based on the same idea of hiding matrix equations.

Since all the unbroken algorithms presented in this paper are new and very similar to broken algorithms, we certainly do not recommend to use them for very sensible applications. However, we believe that it is nice to study them because they have very efficient implementations and because they provide a better understanding of the subtle links between the concept of asymmetric cryptosystem and the computations required for security.

Part I: Variations around C^*

2 A short description of HFE and C^*

We present a short description of the HFE and C^* schemes. See [8] (for C^*), or [10] (for HFE) for more details.

The quadratic function f

Let $K = \mathbf{F}_q$ be a finite field of cardinality q. Let $\mathbf{F}_{q^n}$ be an extension of degree n over $\mathbf{F}_q$. Let

$$f(a) = \sum_{i,j} \beta_{i,j} a^{q^{\theta_{ij}} + q^{\varphi_{ij}}} + \sum_k \alpha_k a^{q^{\xi_k}} + \mu \in \mathbf{F}_{q^n}[a]$$

be a polynomial in a over $\mathbf{F}_{q^n}$, of degree d, for integers θ_{ij}, φ_{ij} and $\xi_k \geq 0$.

Since $\mathbf{F}_{q^n}$ is isomorphic to $\mathbf{F}[x]/(g(x))$, if $g(x) \in \mathbf{F}_q[x]$ is irreducible of degre n, elements of $\mathbf{F}_{q^n}$ may be represented as n-uples over $\mathbf{F}_q$, and f may be represented by n polynomials in n

variables a_1, ..., a_n over $\mathbf{F}_q$:

$$f(a_1,...,a_n) = (f_1(a_1,...,a_n),...,f_n(a_1,...,a_n)).$$

The f_i are quadratic polynomials, due to the choice of f and the fact that $a \mapsto a^q$ is a linear transformation of $\mathbf{F}_{q^n}$.

Secret affine transformation of f

Let s and t be two secret affine bijections $(\mathbf{F}_q)^n \to (\mathbf{F}_q)^n$, where $(\mathbf{F}_q)^n$ is regarded as an n-dimensional vector space over $\mathbf{F}_q$.

Using the function f above and some representation of $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$, the function $(\mathbf{F}_q)^n \to (\mathbf{F}_q)^n$ that assigns $t(f(s(x)))$ to $x \in (\mathbf{F}_q)^n$ can be written as

$$t(f(s(x_1,...,x_n))) = (p_1(x_1,...,x_n),...,p_n(x_1,...,x_n)),$$

where the p_i are quadratic polynomials due to the choice of s, t and f.

The "basic" HFE (cf [10])

Public key: The polynomials P_i, for $i = 1,2,...,n$, as above.

Secret key: The function f and the two affine bijections s and t as above.

Encryption: To encrypt the n-uple $x = (x_1,...,x_n)$, compute the ciphertext $y = (P_1(x_1,...,x_n), ..., P_n(x_1,..., x_n))$ (x should have redundancy, or a hash of x should also be sent).

Decryption: To decrypt y, first find all the solutions z to the equation $f(z) = t^{-1}(y)$ by solving a **monovariate** polynomial equation of degree d. This is always feasible when d is not too large (say $d \leq 1000$ for example) or when f has a special shape (as we will see below in the case of the C^* scheme). Next, compute all the $s^{-1}(z)$, and use the redundancy (or the hash of x) to find M from these.

The "basic" HFE in signature

HFE can also be used in signature, as explained in [10] (essentially, the idea is that now x will be the signature and y the hash of the message to be signed. If the equation $f(z) = t^{-1}(y)$ has no solution z, we compute another hash).

The C^* algorithm (cf [8])

C^* can be seen as a special case of the more general HFE scheme, where the function f is $f(a) = a^{1+q^\theta}$.

Such a function f has some practical advantages: if K is of characteristic 2 and if $1 + q^\theta$ is coprime to $q^n - 1$, then f is a bijection, and the computation of $f^{-1}(b)$ is easy since $f^{-1}(b) = b^{h'}$, where h' is the inverse of $1 + q^\theta$ modulo $q^n - 1$.

However, the C^* scheme was broken in [9], essentially because – in the case of a C^* scheme – there always exist equations such as

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_j \beta_j y_j + \mu_0 = 0 \qquad (1)$$

from which it is possible to break the scheme (see [9]). (Here x is the cleartext (or the signature), y is the ciphertext (or the hash of the message), and γ_{ij}, α_i, β_i and μ_0 are elements of K.) Throughout this paper, we will call "equation of type (1)" any equation like (1).

In the case of HFE, no cryptanalysis has yet been found (when f is well chosen), but the secret key computations are more complex.

3 Four simple variations of C^* (and HFE)

3.1 Less public polynomials: C^*_-, HFE$_-$

The polynomials $(P_1, \ldots, P_n)$ of the "basic" HFE algorithm give y from x. However, it is possible to keep some of these polynomials secret. Let k be the number of these polynomials P_i that we do not give in the public key, so that only P_1, P_2, ..., P_{n-k} are public.

- In an encryption scheme, k must be small, because in order to recover x from y, we will compute the q^k possibilities for y, compute all the corresponding possible x, and find the good x thanks to the redundancy.
  When q is not too large, and when k is very small, for example with $k = 1$ or 2, this is clearly feasible.
- In a signature scheme, k may be much larger. However, we must still have enough polynomials P_i in order that the problem of finding a value x, whose images by P_1, ..., P_{n-k} are given values, is still intractable. A value $k = 1$, 2, or $k = \frac{n}{2}$ for example may be practical and efficient.

Note: This idea to keep some polynomials P_i secret may increase, or not, the security of some schemes. In this paper, we will study the cryptanalytic effects of this idea on the original C^* scheme. We will call C^*_- the obtained scheme ($-$ means that we have *less* public equations).

3.2 Introducing some random polynomials: C^*_+, HFE$^+$

Let P_i be the public polynomials in x_1, x_2, ..., x_n, of a "basic" HFE scheme.

We can imagine to introduce some random extra quadratic polynomials Q_i in x_1, ..., x_n, and to mix the polynomials Q_i and P_i with a secret affine bijection in the given public key. Let k be the number of these Q_i polynomials.

- In a signature scheme, k must be small, because for a given x, the probability to satisfy these extra Q_i equations is $\frac{1}{q^k}$. When m and k are small, the scheme is efficient: after about q^k tries, we will obtain a signature.
- In an encryption scheme, k may be much larger. However, the total number $k + n$ of quadratic public equations must be such that the problem of finding x from a given y is still intractable (hence $k + n$ must be $< \frac{n(n+1)}{2}$, because with $\frac{n(n+1)}{2}$ equations, the values $x_i x_j$ will be found by Gaussian reductions, and then the values x_i will be found). A value $k = 1$, 2 or $k = \frac{n}{2}$ for example may be practical and efficient.

Note 1: This idea of introducing some random polynomials may increase, or not, the security of some schemes. In this paper, we will study the cryptanalytic effects of this idea on the original C^* scheme. We will call C^*_+ the obtained scheme ($+$ means that we mixed the public equations with *additional* random equations).

Note 2: Of course, it is possible to combine the variations of sections 3.1 and 3.2. For example, it is possible to design a signature or an encryption scheme from a "basic" HFE with polynomials P_1, ..., P_n, by keeping P_n secret, introducing a random polynomial Q_n instead of P_n, and computing the public key as a secret affine transformation of P_1, ..., P_{n-1}, Q_n. In the case of a C^* scheme, we will call C^*_{-+} such algorithms.

Note 3: In this paper, we will study the cryptanalytic effects of these ideas on the original C^* scheme, but potentially, these general ideas (adding or/and eliminating some equations) can also be used in many other algorithms such as HFE of [10], or Dragon schemes of [11].

3.3 Introducing more x_i variables: C^*V, HFEV

In signature, it is easy to introduce more x_i variables. In a "basic" HFE scheme, we have $b = f(a)$, where:

$$f(a) = \sum_{i,j} \beta_{ij} a^{q^{\theta_{ij}}+q^{\varphi_{ij}}} + \sum_i \alpha_i a^{q^{\xi_i}} + \mu_0, \qquad (1)$$

where β_{ij}, α_i and μ_0 are elements of L_n.

Let $a' = (a'_1,...,a'_k)$ be a k-uple of variables of K.

In (1), let now α_i be an element of L_n such that each of the n components of α_i in a basis is a secret random linear function of the variables a'_1, ..., a'_k.

And in (1), let now μ_0 be an element of L_n such that each one of the n components of μ_0 in a basis is a secret random quadratic function of the variables a'_1, ..., a'_k.

Then, the $n+k$ variables a_1, ..., a_n, a'_1, ..., a'_k, will be mixed in the secret affine bijection s in order to obtain the variables x_1, ..., x_{n+k}.

And, as before, $t(b_1,...,b_n) = (y_1,...,y_n)$, where t is a secret affine bijection.

Then the public key is given as the n equations $y_i = P_i(x_1,...,x_{n+k})$.

To compute a signature, the values a'_1, ..., a'_k will simply be chosen at random. Then, the values μ_0 and α_i will be computed. Then, the monovariate equation (1) will be solved (in a) in L_n.

Note 1: The idea of this section 3.3, as before, may or may not increase the security of some schemes.

Note 2: This idea is easily applicable in a "basic" HFE scheme, because the complexity of computing f^{-1} depends mainly on the degree of f and not very much on the linear part. This idea is also applicable in a C^* scheme, but the "mixing" of the variables is less efficient in this case: it gives.a scheme that we call C^*V. These schemes with more variables are studied in the paper [5] (and not in the present paper).

3.4 Fixing some x_i variables: C^*F, HFEF

In the original public key of a C^* scheme, we can randomly fix a few values x_i. This can be done on a very small number of values x_i in signature, and it can be done on many variables in encryption.

3.5 Combining all these four ideas

All these four ideas can be combined: it gives the schemes C^*V^-, C^*F^+, C^*VF^{+-}, etc. However, the ideas 3.1 and 3.3 can be done many times in signature and only a few times in encryption. And ideas 3.2 and 3.4 can be done many times in encryption and only a few times in signature.

4 First remarks about C^*_-

First example of attack (with the birthday paradox)

For example, let us assume that we have a Matsumoto-Imai algorithm with $K = \mathbf{F}_2$,$n = 64$,$f(x) = x^{1+2^\theta}$.

So we will have 64 public polynomials P_1,P_2,...,P_{64}.

This algorithm can be attacked as shown in [9].

Now let us assume that P_{64} is kept secret. Decryption of a message (with the secret key) is still possible as follows: we will try the two possibilities for P_{64}, so we will find exactly two solutions for the cleartext x, and thanks to redundancy in x, the right x will be found.

Is this scheme secure? No.

To attack this scheme, we can proceed like this:

1. Find two values x and x', $x \neq x'$ such that $C(x) = C(x')$. By $C(x)$ we mean the encryption of x with the algorithm, *i.e.* the output of the polynomials P_1,P_2,... P_{63}.
2. Let $P_{64} = \sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}x_ix_j + \sum_{i=1}^{n}\mu_ix_i + \delta_0$.
   Since a Matsumoto-Imai algorithm is a permutation, we know that $P_{64}(x) \oplus P_{64}(x') = 1$. It gives us a linear relation in the coefficients γ_{ij},μ_i.
3. Go back to 1 until we have found sufficiently many linear relations in γ_{ij},μ_i in order to find a candidate P_{64} so that P_1,P_2,...,P_{64} are the components of a permutation of $\mathbf{F}_2^{64}$.
4. Attack P_1,P_2,... P_{64} as for a usual Matsumoto-Imai algorithm with the attack described in [9].

For Step 1, we will proceed like this:

1a. We will compute and store in a file F, say 2^{32} pairs $(x,C(x))$. (If this storage capacity is not available then some time/memory trade off are possible: we will need less storage but more time). The storage is done in a way to have easy access to x when $C(x)$ is given.

1b. Now we generate and compute 2^k new pairs $(x',C(x'))$, where k is an integer that we will fix afterwards. For each of these pairs, the probability that $\exists x \in F$ such that $C(x) = C(x')$ is about $1/2^{32}$. (This is because since the Matsumoto-Imai algorithm is a permutation, there is exactly one x, $x \neq x'$, so that $C(x) = C(x')$, it is the value x with the same P_1,...,P_{63} and the opposite P_{64}. And the probability that this x is in F is about $2^{32}/2^{64} = 1/2^{32}$ because there are about 2^{32} values in F and 2^{64} possible values for x).

So the number of x,x',$x \neq x'$, so that $C(x) = C(x')$ that we will found is about $2^k/2^{32} = 2^{k-32}$.

So if we only need one such pair (x,x'), we can have $k \simeq 32$.

Here we need a few of these pairs and k will be slightly bigger than 32, but the attack will be very efficient.

An attack that does not work

In order to avoid the attack given above, one can suggest to have messages of at least 128 bits (*i.e.* $n \geq 128$ if $K = \mathbf{F}_2$), and/or to keep secret more than one bit of the output (for example to keep secret at least two polynomials if $K = \mathbf{F}_2$).
However, even with these transformations, we do not recommend to use such a scheme, since we will see in section 5 how to attack such a scheme. But, before this, we will show here an idea of attack that does not work.
In the cryptanalysis of Matsumoto-Imai scheme given in [9], the key idea is to compute all the "equations of type (1)", *i.e.* such as:

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1).$$

However, as we will see in the simulations below, we may not find enough such equations for a cryptanalysis if some public polynomials of Matsumoto and Imai are kept secret.
Nevertheless one can suggest, instead of computing these equations (1) to compute all the equations such as:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \sum \gamma_{ij} x_i y_j + \delta_0 = 0. \qquad (1').$$

These equations (1') will be computed as usual, since each pair (cleartext/ciphertext) gives linear equations in the coefficients μ_{ijk},ν_{ij},α_i,β_i,γ_{ij},δ_0.
So after some Gaussian reductions all the valid equations (1') will be found.
Moreover all the now secret expressions in y_i are polynomials of degree two in the x_i variables, so for each equation (1) of the original Matsumoto-Imai scheme there is one equation (1') of the scheme with less public polynomials.
So a lot of these equation (1') always exist. From these equations (1') one can think that, maybe, a cryptanalyst will be able to recover the expression of the secret (originaly public) polynomials linear in y_i.

However, this attack does not work, and the existence of these equations (1') is not a problem for the scheme, as we will see below.

The vector space of the equations (1') is of dimension at least $n^2 + n$, since all y_i, $1 \leq i \leq n$, and all $x_i y_j$, $1 \leq i \leq n$, $1 \leq j \leq n$, can be written as polynomials of degree 2 or 3 in the x_k variables. Moreover, the dimension is **exactly** $n^2 + n$, since if it was more than $n^2 + n$, by Gaussian reduction, we would obtain an equation (1'), say P, with only terms in the x_k variables. Since $P(x_1,...,x_n) = 0$ for any $(x_1,...,x_n)$, we would thus have $P = 0$.

It is therefore easy to see what the equations (1') are: they can be seen as the expression as polynomials of degree 3 in x_k of the y_i and $x_i y_j$. As a result, they can be obtained immediately from the public key (*i.e.* from the y_j expressions). There is thus a great difference between equations (1) and (1'): equations (1) are true only for some very simple and very specific quadratic functions y (as in the C^* scheme), whereas equations (1') always exist, even for random quadratic functions y. Therefore, the existence of these equations (1') (unlike equations (1)) is not a dangerous threat for the schemes.

5 Toy simulations of C^*_{-+} with $n = 17$

We have made some toy simulations with $K = \mathbf{F}_2$ and $n = 17$ of the C^*_{-+} algorithm. (These are just "toy" simulations because here the value of n is very small. In real examples, n must be

≥ 64 if $K = \mathbf{F}_2$).

In all these simulations, we have computed the exact number of independent equations between the 16 bits of the input x_1, ..., x_{16}, and the 16 bits of the output y_1, ..., y_{16} of the following form:

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1)$$

or

$$\sum \gamma_{ijk} x_i y_j y_k + \sum \mu_{ij} x_i y_j + \sum \nu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (2)$$

or

$$\sum \gamma_{ijk} x_i x_j y_k + \sum \mu_{ij} x_i y_j + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (3)$$

The obtained results are given in the tables below.

Throughout this paper, we call "equations of type (1)" (resp. 2, 3) any equation like (1), (resp. (2), (3)).

Note 1: In these tables, we have subtracted the number of independent "trivial" equations, such as $x_i^2 = x_i$, or y_i·"y_j"="y_i"·y_j, where "y_i" and "y_j" are written with their expression in the x_k variables.

Note 2: The notation $[\alpha]$ means that, when the y_k variables are given explicit values, we obtain in average α independent equations in the x_k variables.

(In the tables below, we always have $K = \mathbf{F}_2$ and $n = 17$.)

$C^*: \quad f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	34 [16]	17 [16]	17 [16]	17 [16]	17 [16]	17 [16]	17 [16]
Equations (2)	612 [16]	340 [16]	323 [16]	340 [17]	323 [16]	374 [16]	323 [16]
Equations (3)	578 [153]	442 [153]	476 [153]	493 [153]	476 [153]	459 [153]	493 [153]

Table 1

$C^*_{-+1}: \quad f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	17 [15]	1 [1]	1 [1]	1 [1]	1 [1]	1 [1]	1 [1]
Equations (2)	340 [15]	52 [15]	36 [15]	36 [15]	36 [15]	87 [15]	36 [15]
Equations (3)	443 [153]	307 [152]	341 [153]	358 [153]	341 [152]	324 [152]	358 [153]

Table 2

$C^*_{-+2}: \quad f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	1 [1]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	54 [13]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	309 [151]	173 [135]	207 [151]	224 [152]	207 [150]	190 [152]	224 [153]

Table 3

$C^*_{-+3}: \quad f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	176 [153]	51 [68]	74 [91]	91 [108]	74 [91]	57 [74]	91 [108]

Table 4

C^*_{-+4} : $f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	44 [61]	0 [17]	0 [17]	0 [17]	0 [17]	0 [17]	0 [17]

Table 5

As shown in these tables, the attacks of [9] do not work directly against C^*_{-+} if we have less public polynomials, at least if $f(x) = x^3$ is avoided and if two or more polynomials are kept secret.

6 First cryptanalysis of C^*_-

Principle of the attack

We denote by P the complete public form of C^*. We suppose that the first r public equations have been removed. Let $P_{(r+1)...n}$ be the remaining part of P.

The aim of the attack is to recover the public equations $P_{1...r}$ and then to use the classical attack of [9]. Obviously, those equations can be found only modulo the vector space generated by all the public equations. In this section, we will describe an algorithm to recover these equations, with a complexity about $\mathcal{O}(q^n)$. (So this algorithm is expected to succeed when $q^r \leq 2^{40}$ for example.)

Description of the algorithm

Let Q be the *polar* form of P, defined by:

$$Q(x,t) := P(x+t) - P(x) - P(t).$$

1. We randomly choose $t \neq 0$ and $x^{(0)}$.
2. We compute $z_{(r+1)...n} := Q_{(r+1)...n}(x^{(0)},t)$.
3. We solve the equation:
$$Q_{(r+1)...n}(x,t) = z_{(r+1)...n}$$
where x is the indeterminate. There are at least two solutions ($x^{(0)}$ and $x^{(0)}+t$) and at most 2.2^r solutions. This comes from the fact that – for a given value $z_{1...r}$ – (among 2^r possible), the equation $Q(x,t) = z$ has 0 or 2 solutions.

Proof: Suppose that x and x' are two *distinct* solutions.

$$Q(x,t) = Q(x',t) \Rightarrow P(x+t) - P(x) = P(x'+t) - P(x').$$

By definition of P, this yields:

$$t(s(x+t)^{1+2^\theta} - s(x)^{1+2^\theta}) = t(s(x'+t)^{1+2^\theta} - s(x')^{1+2^\theta}).$$

Since t is bijective, we also have:

$$s(x+t)^{1+2^\theta} - s(x)^{1+2^\theta} = s(x'+t)^{1+2^\theta} - s(x')^{1+2^\theta}.$$

If we let $\lambda = s(0)$, we can write:

$$(s(x) + s(t) + \lambda)(s(x)^{2^\theta} + s(t)^{2^\theta} + \lambda^{2^\theta}) - s(x).s(x)^{2^\theta}$$

$$= (s(x') + s(t) + \lambda)(s(x')^{2^\theta} + s(t)^{2^\theta} + \lambda^{2^\theta}) - s(x').s(x')^{2^\theta}.$$

After a few computations, we obtain:

$$(s(x' - x) + \lambda).(s(t) + \lambda)^{2^\theta} = (s(x' - x) + \lambda)^{2^\theta}.(s(t) + \lambda).$$

Since $x \neq x'$, this implies:

$$(s(x' - x) + \lambda)^{2^\theta - 1} = (s(t) + \lambda)^{2^\theta - 1}.$$

Finally, $a \mapsto a^{2^\theta - 1}$ is bijective, because $gcd(2^\theta - 1, 2^n - 1) = 1$, so that

$$x' = x + t.$$

As a result, if x is a solution, there exists *exactly one* other solution: $x' = x + t$.

Steps 1, 2 and 3 are repeated until we obtain the maximum number of solutions: 2.2^r (we use each time a different choice for $t \neq 0$ and $x^{(0)}$). The average number of necessary tries is estimated to be about 2^r.

4. Suppose we have found $t \neq 0$ and $x^{(0)}$ such the equation

$$Q_{(r+1)\ldots n}(x,t) = z_{(r+1)\ldots n}$$

has exactly 2.2^r solutions:

$$\left\{x^{(0)}, x^{(0)} + t, x^{(1)}, x^{(1)} + t, \ldots, x^{(2^r-1)}, x^{(2^r-1)} + t\right\}.$$

Let k be an integer such that $1 \leq k \leq r$. For half of the solutions, we have $Q_k(x,t) = 0$, and for the other half, we have $Q_k(x,t) = 1$, and this remains true if we consider only the subset

$$\left\{x^{(0)}, \ldots, x^{(2^r-1)}\right\}$$

of the set of solutions. Therefore, a summation gives:

$$\sum_{\nu=0}^{2^r-1} Q_k(x^{(\nu)}, t) = 2^{r-1}.$$

This gives an equation of degree one on the $\frac{n(n-1)}{2} + 1$ coefficients of Q_k (this equation is the same for all the values k, $1 \leq k \leq r$).

5. By repeating steps 1-4 $\mathcal{O}(n^2)$ times, with different choices of $(x^{(0)}, t)$, we expect to find $\frac{n(n-1)}{2} + 1 - n$ equations on the coefficients of the Q_k $(1 \leq k \leq r)$. This will give Q_1, ..., Q_r modulo the vector space generated by all the public equations.
6. The public polynomials P_k $(1 \leq k \leq r)$ are not yet completely determined: Q_k only contains terms of the form $x_i t_j + x_j t_i$ $(i \neq j)$ (which come from terms $x_i x_j$ $(i \neq j)$ of P_k), and a constant term (which is the same as the one of P_k). We can suppose that $K = \mathbf{F}_2$, and thus we can write:

$$P_k = \tilde{P}_k + \sum_i \nu_{ik} x_i \qquad (1 \leq k \leq r)$$

where $\tilde{P}_k$ is now known, but where the coefficients ν_{ik} are still to be found. We also note $P_k = \tilde{P}_k$ when $r + 1 \leq k \leq n$, and $\tilde{y}_k = \tilde{P}_k(x_1, \ldots, x_n)$ for $1 \leq k \leq n$.

From the cryptanalysis of C^*, we know that there exist equations of the form

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_i \beta_i y_i + \delta_0 = 0.$$

i.e. (with the notations above):

$$\sum_{i,j} \gamma_{ij} x_i \tilde{y}_j + \sum_i \alpha_i x_i + \sum_i \beta_i \tilde{y}_i + \delta_0 + \sum_i \sum_{j=1}^{r} \gamma_{ij} x_i \Big(\sum_k \nu_{kj} x_k \Big) + \sum_{i=1}^{r} \beta_i \Big(\sum_k \nu_{ki} x_k \Big) = 0.$$

If we replace $\tilde{y}_j$ by $\tilde{P}_j(x_1,...,x_n)$ and if we consider the terms of total degree 3 in the x_i, we obtain $\frac{n(n-1)(n-2)}{6}$ equations on the n^2 indeterminates γ_{ij}, which can thus be determined by gaussian reductions.
We then consider the terms of total degree 2 in x_i, and that gives $\frac{n(n-1)}{2}$ equations on the $rn+n$ indeterminates ν_{kj} and β_i.

7. Once P_1, ..., P_n are completely known, the classical attack on C^* can be applied, so that C^*_- (when r is small) is also broken. The complexity of this cryptanalysis is in $\mathcal{O}(q^r)$ plus the complexity of the cryptanalysis of the original C^* scheme.

Remark: This cryptanalysis uses deeply the fact that C^* is a permutation polynomial. A general theory about permutation polynomials, and the related notion of orthogonal systems of equations, can be found in [7], chapter 7.

7 The C^*_{--} algorithm

When $q^r \geq 2^{64}$, then the cryptanalysis given in section 6 is not efficient. The scheme is then called C^*_{--}. The C^*_{--} scheme cannot be used for encryptions any more, but this scheme is still a very efficient scheme for signatures, and its security is an open problem.

8 Cryptanalysis of C^*_+

The cryptanalysis of C^*_+ is very simple: it just works exactly as the original cryptanalysis of C^*. We will first generate all the equations

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_j \beta_j y_j + \delta_0 = 0. \qquad (1)$$

Since in C^*_+, we just have **added** some equations (and eliminated none), we will find at least as much equations (1) as in the original C^*.

Then, as explained in [9], from such equations (1) we will be able to find x from y (and thus to break the system).

Remark 1: Moreover, we will be able to eliminate the random added equations and to recover an original C^*, because an equation (1) generally comes from only the y_i of C^* (and not from the added equations). Therefore, by generating an equation (1), by writing it as $x_1(P_1(y)) + x_2(P_2(y)) + \ldots + x_n(P_n(y))$ (where P_1, ..., P_n are polynomials of degree one in y_1, ..., y_{n+k}), and by making the change of variables $y'_1 = P_1(y)$, ..., $y'_n = P_n(y)$, the variables y'_1, ..., y'_n are the outputs of an original C^* scheme.

Remark 2: However, this idea of adding an equation may be much more efficient in a scheme where no equation (1) exist (as in some HFE schemes) (or when we add **and** eliminate some equations, as we will see in C^*_{-+}).

9 Cryptanalysis of C^*_{-+}, second cryptanalysis of C^*_-

The idea

Let us consider – as an example – (A) and (B) the two following equations of type (1) on $K = \mathbf{F}_2$:

$$x_1y_1 + x_3y_4 + x_4y_4 + x_5y_2 = 0, \qquad (A)$$

$$x_3y_1 + x_4y_2 + x_5y_2 = 1. \qquad (B)$$

Then $x_3 \cdot (A) + x_1 \cdot (B)$ gives:

$$x_3y_4 + x_3x_4y_4 + x_3x_5y_2 + x_1x_4y_2 + x_1x_5y_2 = x_1. \qquad (C)$$

This equation is an equation "of type (3)", and it has no term in y_1.

Cryptanalysis of C^*_{-+1}

We will first use this idea for the cryptanalysis of C^*_{-+} when the number r of removed equations is $r = 1$.

We know that from the variables of the original C^* we have at least n independent equations of type (1).

So by multiplying these equations by one x_k, $1 \leq k \leq n$, we generate n^2 independent equations of type (3).

By Gaussian reductions, we will obtain at least $n^2 - \frac{n(n+1)}{2}$ ($= \frac{n(n-1)}{2}$) equations of type (3) with no terms in y_1 (because we have at most $\frac{n(n+1)}{2}$ terms in $y_1x_ix_j$ or y_1x_i).

Now, when we give explicit values for y, we obtain (by Gaussian reductions on the $X_{ij} = x_i \cdot x_j$ variables) the x_i values. As a result, with the equations (3) we will be able to break C^*_{-+1}: *i.e.* to recover an x from a given y.

Remark: This attack works because (as shown in our simulations, see the tables of section 5) the number of independent equations does not decrease significantly when the y_k variables are given explicit values. Moreover, our simulations also show that in this attack (based on equations (3)), we will generally have more than $\frac{n(n-1)}{2}$ equations (3) for $r = 1$, so that the attack will work even better than expected.

Cryptanalysis of C^*_{-+r}, for $r = 2,3$

As shown in the tables, we generally have more than $\frac{n(n-1)}{2}$ equations of type (3), so that the attack also works very well when $r = 2$ or $r = 3$, since we have more equations (3) than expected. Of course, when – after Gaussian reductions – we still have a few variables to guess, we can guess them by exhaustive search (if this number is very small).

Cryptanalysis of C^*_{-+r}, for $r \geq 4$

When $r \geq 4$, the attack given above may not work, so that we may need to generalize this attack by generating more general equations such as equations of total degree $d \geq 4$ (instead of three), and of degree one in the y_i variables.

We know that from the variables of the original C^* we have at least n independent equations of type (1). So by multiplying these equations by $d-2$ variables x_k, $1 \leq k \leq n$, we generate about $n \cdot \frac{n^{d-2}}{(d-2)!}$ independent equations of the following type:

$$\sum \gamma_{i_1 i_2 ... i_d} x_{i_1} x_{i_2} ... x_{i_{d-1}} y_d + ... = 0. \qquad (*)$$

By Gaussian reductions, we will obtain at least $n \cdot \frac{n^{d-2}}{(d-2)!} - r \cdot \frac{n^{d-1}}{(d-1)!}$ equations $(*)$ with no terms in y_1, y_2, ..., y_r (because we have at most $\frac{n^{d-1}}{(d-1)!}$ terms in $y_\mu x_{i_1} x_{i_2} ... x_{i_{d-1}}$, and r values μ such that $1 \leq \mu \leq r$).

Now, when we give explicit values for y, we obtain (by Gaussian reductions on the $X_{i_1 ... i_{d-1}} = x_{i_1} ... x_{i_{d-1}}$ variables) the x_i values if

$$n \cdot \frac{n^{d-2}}{(d-2)!} - r \cdot \frac{n^{d-1}}{(d-1)!} \geq \frac{n^{d-1}}{(d-1)!}$$

(because as shown in our simulations the number of independent equations do not dramatically decrease when we give explicit values for y), *i.e.* when $r \leq d-2$.

Complexity: The complexity of this attack is essentially the complexity of Gaussian reductions on $\mathcal{O}(n^d)$ terms. This complexity is in $\mathcal{O}(n^{\omega d})$, with $\omega = 3$ in the usual Gaussian reduction algorithms, or $\omega = 2.3755$ in the best known general purpose Gaussian reduction algorithm (see [1]). As a result, this complexity increases in $\mathcal{O}(n^{\omega r})$, *i.e.* exponentially in r.

Since our simulations show that this attack works sensibly better than described above (because we have a few more equations $(*)$), we expect that – with this attack – it is feasible to attack C^*_{-+r} when $r \leq 10$ approximately. Therefore, we think that any $r \leq 10$ is insecure. However, the complexity of the attack increases a lot when r increases. Hence, at the present, for practical applications, it is an open problem to find efficient cryptanalysis of C^*_{-+r} when $r > 10$.

Can we recover the corresponding C^*_- from C^*_{-+}?

This is sometime feasible. For example, when we have equations of type (2) (this is generally the case only when r is very small: see the tables), then these equations generally come from y_k variables of the original C^*_-, and not from the added random quadratic equations. Therefore, by looking at the terms in factor of a monomial $x_i x_j$ in those equations (2), we will find the vector space generated by the public equations of the original C^*_- equations. (Then in such a case the C^*_{-+} algorithm can be attacked as a C^*_- algorithm.)

Remark: One can think to also use this idea on equations of type (3) instead of equations of type (2). However, there is a technical problem with the equations of type (3): these equations are "mixed" with "trivial" equations $y_i \cdot$"y_j"$=$"y_i"$\cdot y_j$, where "y_i" and "y_j" are written with their quadratic expression in the x_k variables. And it is not clear how these "trivial" equations can be removed. This is why we have used equations (2) instead.

Part II: Schemes with a hidden matrix

10 The $[C]$ scheme

In this section, we recall the description of the $[C]$ scheme, presented by H. Imai and T. Matsumoto in [4].

Let $K = GF(2^m)$ be a *public* finite field of cardinality $q = 2^m$. The basic idea is to use the transformation $A \mapsto A^2$ of the set $\mathcal{M}_2(K)$ of the 2×2 matrices over the field K.

This transformation is not one-to-one, but it can be made bijective if we restrict ourselves to matrices with a *non-zero trace*:

Lemma 4 *Let* $\mathcal{E} = \{M \in \mathcal{M}_2(K),\ tr(M) \neq 0\}$.

Then $\Phi : \begin{cases} \mathcal{E} \to \mathcal{E} \\ A \mapsto A^2 \end{cases}$ *is bijective. Moreover, for any* $B \in \mathcal{E}$, *we have:*

$$\Phi^{-1}(B) = \frac{1}{\sqrt{tr(B)}} \cdot \Big(B + \sqrt{det(B)} \cdot I\Big),$$

where $\sqrt{\ }$ *denotes the inverse of the* bijective *function* $\begin{cases} GF(2^m) \to GF(2^m) \\ \lambda \mapsto \lambda^2 \end{cases}$.

Proof: Let $B \in \mathcal{E}$.

- From Cayley-Hamilton theorem (applied to B), we have:

$$B^2 - (\mathrm{tr}\ B) \cdot B + (\det\ B) \cdot I = 0$$

  and thus $\left(B + \sqrt{\det(B)} \cdot I\right)^2 = (\mathrm{tr}\ B) \cdot B$. Since $\mathrm{tr}(B) \neq 0$, we obtain $B + \sqrt{\det(B)} \cdot I \neq 0$.
- Suppose that a matrix A exists such that $A^2 = B$. By applying Cayley-Hamilton theorem to A, we have:

$$A^2 - (\mathrm{tr}\ A) \cdot A + (\det\ A) \cdot I = 0$$

  i.e.

$$(\mathrm{tr}\ A) \cdot A = B + \sqrt{\det(B)} \cdot I.$$

  As a result, we have $\mathrm{tr}(A) \neq 0$ and $A = \lambda \cdot (B + \sqrt{\det(B)} \cdot I)$, which gives easily:

$$A = \frac{1}{\sqrt{\mathrm{tr}(B)}} \cdot \Big(B + \sqrt{\det(B)} \cdot I\Big).$$

  Reciprocally, this A is a satisfies $A^2 = B$ and $\mathrm{tr}(A) \neq 0$.
- In conclusion, Φ is a bijective transformation of $\mathcal{E}$, and:

$$\Phi^{-1}(B) = \frac{1}{\sqrt{\mathrm{tr}(B)}} \cdot \Big(B + \sqrt{\det(B)} \cdot I\Big).$$

Note: The function $\sqrt{\ }$ is easy to compute, since we have $\sqrt{\lambda} = \lambda^{2^{m-1}}$ for any $\lambda \in \mathrm{GF}(2^m)$.

We now describe the $[C]$ scheme used in encryption mode.

The set $\mathcal{M}_2(K)$ can be considered as a vector space of dimension 4 over K. Therefore, we can choose $s : K^4 \to \mathcal{M}_2(K)$ and $t : \mathcal{M}_2(K) \to K^4$ two *secret* linear bijections such that:

- s maps the hyperplane $\{x_1 = 0\}$ of K^4 onto the hyperplane $\{\mathrm{tr}(M) = 0\}$ of $\mathcal{M}_2(K)$;
- t maps the hyperplane $\{\mathrm{tr}(M) = 0\}$ of $\mathcal{M}_2(K)$ onto the hyperplane $\{x_1 = 0\}$ of K^4.

Representation of the messages

Each message M is represented by a 4-uple $(x_1,x_2,x_3,x_4) \in K^4$ such that $x_1 \neq 0$. The message space is $\mathcal{M} = \{(x_1,x_2,x_3,x_4) \in K^4,\ x_1 \neq 0\}$.

The quadratic function f

We then define the following quadratic function on the message space:

$$f : \begin{cases} \mathcal{M} \to \mathcal{M} \\ x \mapsto t(s(x)^2) \end{cases}.$$

The hypotheses made on s and t, together with lemma 1, show that the function f is a bijection.

Public key: The 4-uple (p_1,p_2,p_3,p_4) of 4-variate quadratic polynomials over K that represent f. They are defined by:

$$f(x_1,x_2,x_3,x_4) = (p_1(x_1,x_2,x_3,x_4),p_2(x_1,x_2,x_3,x_4),p_3(x_1,x_2,x_3,x_4),p_4(x_1,x_2,x_3,x_4)).$$

Secret key: The two linear bijections s and t.

Note: For s (respectively t), there are $|\mathrm{GL}_3(K)|\cdot|K^3|\cdot|K^*| = (q^3-1)(q^3-q)(q^3-q^2)q^3(q-1) \simeq q^{13}$ possibilities, instead of $|\mathrm{GL}_4(K)| = (q^4-1)(q^4-q)(q^4-q^2)(q^4-q^3) \simeq q^{16}$ if we remove the condition "s maps $\mathcal{M}$ onto $\mathcal{E}$" (respectively "t maps $\mathcal{E}$ onto $\mathcal{M}$").

Encryption

To encrypt the message M represented by $x = (x_1,x_2,x_3,x_4) \in \mathcal{M}$, compute the ciphertext $y = (y_1,y_2,y_3,y_4)$ with the following formulas:

$$\begin{cases} y_1 = p_1(x_1,x_2,x_3,x_4) \\ y_2 = p_2(x_1,x_2,x_3,x_4) \\ y_3 = p_3(x_1,x_2,x_3,x_4) \\ y_4 = p_4(x_1,x_2,x_3,x_4) \end{cases}$$

Decryption

To decrypt the ciphertext $y \in \mathcal{M}$, compute:

$$x = s^{-1}\left(\frac{1}{\sqrt{\mathrm{tr}(t^{-1}(y))}} \cdot \left(t^{-1}(y) + \sqrt{\det(t^{-1}(y))} \cdot I\right)\right).$$

11 First cryptanalysis of $[C]$

The security of the cryptosystem is based on the difficulty of solving the following system of 4 quadratic equations in 4 variables over $K = GF(2^m)$:

$$\begin{cases} y_1 = p_1(x_1,x_2,x_3,x_4) \\ y_2 = p_2(x_1,x_2,x_3,x_4) \\ y_3 = p_3(x_1,x_2,x_3,x_4) \\ y_4 = p_4(x_1,x_2,x_3,x_4) \end{cases}$$

Unfortunately, such a system can always be easily solved by using an algorithm based on Gröbner bases. At the present, the best implementations of Gröbner bases can solve any set of n quadratic equations with n variables over any reasonable field K, when $n \leq 16$ approximately (cf [2]). Therefore, the original $[C]$ is not secure.

This first cryptanalysis shows that the parameter n must not be too small if we want to avoid attacks based on algebraic methods for solving systems of multivariate polynomial equations. That is why we are going to describe a generalization of the scheme to higher dimensions (for which Gröbner bases algorithms will be unefficient) in the next section.

12 The more general $[C]_n$ scheme

We present here a generalization of the $[C]$ scheme of H. Imai and T. Matsumoto, which involves $n \times n$ matrices over the field K, instead of 2×2 matrices. This cryptosystem will be called $[C_n]$.

As in the case of $[C]$, we take a public finite field $K = GF(2^m)$ of cardinality $q = 2^m$.

The basic idea is still to use the transformation $A \mapsto A^2$ of the set $\mathcal{M}_n(K)$ of the $n \times n$ matrices over the field K.

The set $\mathcal{M}_n(K)$ can be considered as a vector space of dimension n^2 over K, so that we can choose $s : K^{n^2} \to \mathcal{M}_n(K)$ and $t : \mathcal{M}_n(K) \to K^{n^2}$ two *secret* affine bijections.

We now describe the $[C]$ scheme used in encryption mode.

Representation of the messages

Each message M is represented by a n^2-uple $(x_1,...,x_{n^2}) \in K^{n^2}$. The message space is $\mathcal{M} = K^{n^2}$.

The quadratic function f

We then define the following quadratic function on the message space:

$$f : \begin{cases} \mathcal{M} \to \mathcal{M} \\ x \mapsto t(s(x)^2) \end{cases}.$$

Public key: The n^2-uple $(p_1,...,p_{n^2})$ of n^2-variate quadratic polynomials over K that represent f. They are defined by:

$$f(x_1,...,x_{n^2}) = (p_1(x_1,...,x_{n^2}),...,p_{n^2}(x_1,...,x_{n^2})).$$

Secret key: The two affine bijections s and t.

Encryption

To encrypt the message M represented by $x = (x_1,...,x_{n^2}) \in \mathcal{M}$, compute the ciphertext $y = (y_1,...,y_{n^2})$ with the following formulas:

$$\begin{cases} y_1 = p_1(x_1,...,x_{n^2}) \\ \quad \vdots \\ y_{n^2} = p_{n^2}(x_1,...,x_{n^2}) \end{cases}$$

Decryption

To decrypt the ciphertext $y \in \mathcal{M}$, one has to solve the equations $A^2 = B$, where $B = t^{-1}(y)$, and then to compute the cleartext $x = s^{-1}(A)$.

It is important to notice that $A \mapsto A^2$ is not a bijection any longer (contrary to the original $[C]$ scheme described in section 10). As a result, there may be several possible cleartexts for a given ciphertext. One solution to avoid this ambiguousness is to put some redundancy in the representation of the messages, by making use of an error correcting code or a hash function (for details, see [10] p. 34, where a similar idea is used in a different scheme).

The feasibility of choosing the right cleartext among the possible ones is due to the fact that – for an average B – the number of solutions A of the equations $A^2 = B$ remains reasonable, as shown in table 6 below:

# pre-images	$n=2$	$n=3$	$n=4$	# pre-images	$n=2$	$n=3$	$n=4$
0	6	252	34440	13-15	0	0	0
1	8	160	22272	16	0	0	672
2	0	42	5040	17-21	0	0	0
3	0	0	0	22	0	2	240
4	2	56	2240	23-315	0	0	0
5-11	0	0	0	316	0	0	2
12	0	0	630	> 316	0	0	0

Table 6: Repartition of the number of pre-images for $[C_n]$ over $K = GF(2)$

Note 1: In the case $n = 4$, the two matrices having 316 pre-images are 0 and I.

Note 2: These results are just "toy simulations" of $[C_n]$, because if $K = GF(2)$, n must be such that $n^2 \geq 64$ in real examples.

To solve the equation $A^2 = B$ when B is a given average matrix of $\mathcal{M}_n(K)$, two methods can be used:

- The first one is based on the Jordan reduction of matrices, and provides a polynomial time algorithm to compute the square roots of a given matrix. For details, see [3] (chapter VIII, p. 231).
- The second one is based on the Cayley-Hamilton theorem. Let us denote by

$$\chi_M(\lambda) = \lambda^n + \alpha_{n-1}(M)\lambda^{n-1} + ... + \alpha_1(M)\lambda + \alpha_0(M)$$

the characteristic polynomial of a matrix $M \in \mathcal{M}_n(K)$.
Since K is a field of characteristic 2, it is easy to prove that $(\chi_M(\lambda))^2 = \chi_{M^2}(\lambda^2)$, and thus $\alpha_i(M^2) = (\alpha_i(M))^2$ $(0 \leq i \leq n-1)$.
Suppose now that A satisfies $A^2 = B$ for a given B. Then, from the Cayley-Hamilton theorem:

$$\chi_A(A) = A^n + \alpha_{n-1}(A) \cdot A + ... + \alpha_1(A) \cdot A + \alpha_0(A) \cdot I = 0.$$

Hence:

$$A\Big(\sqrt{\alpha_1(B)}\cdot I+\sqrt{\alpha_3(B)}\cdot B+\sqrt{\alpha_5(B)}\cdot B^2+\dots\Big) = \sqrt{\alpha_0(B)}\cdot I+\sqrt{\alpha_2(B)}\cdot B+\sqrt{\alpha_4(B)}\cdot B^2+\dots$$

If we make the assumption that $\sqrt{\alpha_1(B)} \cdot I + \sqrt{\alpha_3(B)} \cdot B + \dots$ is invertible, we obtain the following formula to compute A:

$$A = \Big(\sqrt{\alpha_0(B)} \cdot I + \sqrt{\alpha_2(B)} \cdot B + \dots\Big)\Big(\sqrt{\alpha_1(B)} \cdot I + \sqrt{\alpha_3(B)} \cdot B + \dots\Big)^{-1}$$

and thus

$$x = s^{-1}\Big(\Big(\sqrt{\alpha_0(t^{-1}(y))} \cdot I + \sqrt{\alpha_2(t^{-1}(y))} \cdot t^{-1}(y) + \dots\Big) \Big(\sqrt{\alpha_1(t^{-1}(y))} \cdot I + \sqrt{\alpha_3(t^{-1}(y))} \cdot t^{-1}(y) + \dots\Big)^{-1}\Big)$$

Note 1: The first method always works, whereas the second one can be used only for ciphertexts y such that $B = t^{-1}(y)$ satisfies $\alpha_1(B) \cdot I + \alpha_3(B) \cdot B + \ldots$ invertible.

Note 2: The scheme can also be used in signature. To sign a message M, the basic idea is to compute x from $y = h(R||M)$ (as if we were deciphering a message), where h is a hash function and R is a small pad. If we succeed, (x,R) will be the signature of M. If we do not succeed (because the function is not a bijection), we try another pad R (for variants and details, see [10], where a similar idea is used).

13 Cryptanalysis of $[C]_n$

In this section, we describe a polynomial attack against the $[C_n]$ algorithm, which proves that this scheme is insecure.

The key idea is to use the fact that $B = A^2$ implies $AB = BA$ (whereas two random matrices A and B do not commute in general).

- We begin by computing (by Gaussian reductions) all the equations of the following type (which we called "type (1)" in section 5):

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1)$$

  The relation $AB = BA$ gives *a priori* n equations of this type. In fact, when we give explicit values to the y_i variables, we cannot obtain n independent linear equations on the x_i variables, since $AB = BA$ is also true when $B = P(A)$, where P is *any* polynomial in $K[X]$.
  The exact number of independent linear equations coming from $AB = BA$ is given by the following result of [3]:

  Theorem 12 *The number N of linearly independent matrices that commute with the matrix B is given by the formula*

$$N = n_1 + 3n_2 + ... + (2t-1)n_t$$

  where n_1, n_2, ..., n_t are the degrees of the non constant invariant polynomials *$i_1(\lambda)$, ..., $i_t(\lambda)$ of B.*

  (See [3], chapter VI, for the definition of the invariant polynomials, and chapter VIII for a proof of the theorem.)

  In particular, we have $n \leq N \leq n^2$, with $N \simeq n$ in most of the cases.
- It remains – *a priori* – to perform an exhaustive search on $\simeq n$ variables to end the attack. In fact, we have made some simulations (see table 7 below) that suggest that there also exist many equations of type (2) (defined in section 5), and type (4) defined by:

$$\sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (4)$$

	$n=2$	$n=3$	$n=4$
$p=2$	10 16 39	10 18 153	17 32 292
$p=3$	4 4 28	9 9 89	16 16 271
$p=31$	3 3 14	8 8 79	15 15 254
$p=127$	3 3 14	8 8 79	15 15 254

Table 7: Number of equations of $\begin{smallmatrix}\text{type 1 type 4}\\ \text{type 2}\end{smallmatrix}$ **for** $[C_n]$ **over** $K=GF(p)$

Note: For $p=2$, on these examples, we obtain $(n+1)$ (formally) linearly independent equations of type (1). This can be explained by the fact that – on the field $K=GF(2)$ – the equations $B=A^2$ implies $\mathrm{tr}(B)=\mathrm{tr}(A)$.

These equations of type (1), (2) and (4) can be found by Gaussian reductions on a polynomial number of cleartext/ciphertext paris. Therefore, the $[C_n]$ scheme is unlikely to be secure: by using all the found equations of type (1), (2) and (3), a cleartext will be easily found by Gaussian reductions.

14 A suggestion: the *HM* scheme

The cryptanalysis of $[C_n]$ described in section 13 uses the fact that A and B commute when $B=A^2$. In order to avoid that very special algebraix property, we suggest to replace the transformation $B=A^2$ by the equation $B=A^2+MA$, where M is a *secret* matrix randomly chosen in $\mathcal{M}_n(K)$.

The description of the obtained scheme – called *HM* – is exactly the same as for $[C_n]$. As in section 12, the transformation is generally not one-to-one, but the scheme can be used in a practical way because – as in the case of $[C_n]$ –, the number of pre-images of a given average matrix B remains under a reasonable limit. Table 8 below illustrates this fact (for a randomly chosen matrix M).

Note: These results are just "toy simulations" of $[C_n]$, because if $K=GF(2)$, n must be such that $n^2\geq 64$ in real examples.

In order to obtain a practical scheme, one has to be able to solve the equation

$$A^2+MA=B$$

for a given matrix $B\in\mathcal{M}_n(K)$.

There indeed exist a polynomial time algorithm to perform this computation (see [3], chapter VIII). The basic idea of this algorithm is to use the fact that:

$$B=A^2+MA\Rightarrow g(A)=0,$$

where $g(\lambda) = \det(\lambda^2 + \lambda \cdot M - B)$ is a polynomial with *scalar* coefficients (notice that this property is a generalization of the Cayley-Hamilton theorem). The equation $g(A) = 0$ can be solved by using the Jordan reduction of matrices.

# pre-images	$n=2$	$n=3$	$n=4$	# pre-images	$n=2$	$n=3$	$n=4$
0	6	284	39552	16	0	0	72
1	8	112	12024	17	0	0	0
2	0	42	6576	18	0	0	12
3	0	32	2256	19	0	0	24
4	2	34	1868	20	0	0	24
5	0	0	960	21	0	0	0
6	0	0	972	22	0	0	24
7	0	0	168	23-25	0	0	0
8	0	2	324	26	0	0	36
9	0	0	48	27	0	0	0
10	0	2	144	28	0	0	6
11	0	0	96	29-33	0	0	0
12	0	4	162	34	0	0	4
13	0	0	56	35-39	0	0	0
14	0	0	72	40	0	0	8
15	0	0	48	> 40	0	0	0

Table 8: Repartition of the number of pre-images for *HM* over $K = GF(2)$

The *HM* scheme seems to be less vulnerable to attacks based on affine multiple (*i.e.* on equations such as those of type (1), (2) or (4)), as shown in table 9 below:

	$n=2$	$n=3$	$n=4$
$p=2$	10 16 39	3 11 133	3 18 49
$p=3$	1 1 14	1 1 11	1 1 18
$p=31$	0 0 1	0 0 1	0 0 1
$p=127$	0 0 0	0 0 0	0 0 0

Table 9: Number of equations of $\overset{\text{type 1 type 4}}{\text{type 2}}$ for *HM* over $K = GF(p)$

However, we have made computations (see table 10 below) showing that equations of type (3) (defined in section 5) still exist, and also equations of "type (5)", defined by:

$$\sum \mu_{ij} x_i y_j + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (5)$$

HM	$n=2$	$n=3$	$n=4$
Equations (5)	7	17	31
Equations (3)	9	30	58

Table 10: Number of linearly independent equations (after replacement of the y_i variables by explicit values)

Note: It may be noticed that $B = A^2 + MA$ implies the two following identities:

$$\begin{cases} AB - BA = AMA - MA^2 & \text{(type (5))} \\ A^2B - BA^2 = BMA - MAB & \text{(type (3))} \end{cases}$$

This explains – in part – the existence of such equations of type (5) and type (3).

The fact that such equations do exist threatens the *HM* scheme. In fact they make the cryptanalyst able to distinguish between a random quadratic transformation of K^{n^2} and a quadratic transformation corresponding to the *HM* scheme.

This fact explains that we do not recommend the *HM* scheme. However, at the present, the existence of equations of type (5) and type (3) does not seem sufficient to break the scheme. Therefore, the question of the security of *HM* remains open...

15 Conclusion

Among cryptologists that have studied the problem, two main opinions arise as concerns public key schemes built with multivariate polynomials. Some of them think that most of these schemes should be vulnerable to attacks based on general principles, still to be found. According to others, the status of those many schemes can be compared to the one of most secret key algorithms: no relative proof of security is known, but the great flexibility for the choice among the possible variants of the schemes, together with the relative easiness for building efficient schemes that avoid known attacks, may support a certain confidence in the security of the schemes, at least – *a priori* – for those which do not seem too close to known cryptanalytic techniques.

The present article does not settle the question once and for all. Nevertheless, it gives arguments for both opinions. On the one hand, we have shown how to break some schemes for which no cryptanalysis had been given before. On the other hand, we have studied some simple and general ideas (removing equations, adding ones, introducing new variables...) that might – *a priori* – sensibly enforce the security of some asymmetric schemes. Interesting mathematical questions naturally arise: better understanding and detecting orthogonal polynomials, using a non commutative ring of matrices to generate multivariate equations on a (commutative) field, etc. If we had to take a strong line as concerns the unbroken schemes, our current opinion is that the most provocative schemes (C^*_{--}, C^*_{-+}, *HM*) may be too close to known cryptanalysis to be recommended, but more complex schemes (such as HFE_{-+}) may be really secure... However, it is still too soon to have a definitive opinion, and we think that – above all – the important point is to go further into the understanding of the mysterious links between mathematics and the concepts of asymmetric cryptography and cryptanalysis.

References

[1] D. Coppersmith, S. Winograd, *Matrix Multiplication via Arithmetic Progressions*, J. Symbolic Computation, 1990, vol. 9, pp. 251-280.

[2] J.C. Faugere, *Rough evaluation* (personal communication).

[3] F.R. Gantmacher, *The Theory of Matrices*, volume 1, Chelsae Publishing Company, New-York.

[4] H. Imai, T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, Algebraic Algorithms and Error Correcting Codes (AAECC-3), Grenoble, 1985, Springer-Verlag, Lectures Notes in Computer Science n$^{\text{O}}$ 229.

[5] A. Kipnis, J. Patarin, L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, Proceedings of EUROCRYPT'99, Springer, pp. 206-222.

[6] N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics, Volume 3, Springer, 1998.

[7] R. Lidl, H. Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press.

[8] T. Matsumoto, H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Advances in Cryptology, Proceedings of EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[9] J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Advances in Cryptology, Proceedings of CRYPTO'95, Springer, pp. 248-261.

[10] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Advances in Cryptology, Proceedings of EUROCRYPT'96, Springer, pp. 33-48.

[11] J. Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Advances in Cryptology, Proceedings of CRYPTO'96, Springer, pp. 45-60.

[12] J. Patarin, L. Goubin, *Trapdoor One-way Permutations and Multivariate Polynomials*, Proceedings of ICICS'97, Springer, LNCS n°1334, pp. 356-368.

[13] J. Patarin, L. Goubin, *Asymmetric Cryptography with S-Boxes*, Proceedings of ICICS'97, Springer, LNCS n°1334, pp. 369-380.

QUARTZ, 128-bit long digital signatures

CT-RSA'2001 *(Version complète)*
Article avec Nicolas Courtois et Jacques Patarin (Bull CP8)

Abstract

For some applications of digital signatures the traditional schemes as RSA, DSA or Elliptic Curve schemes, give signature size that are not short enough (with security 2^{80}, the minimal length of these signatures is always $\geq$ 320 bits, and even $\geq$ 1024 bits for RSA). In this paper we present a first well defined algorithm and signature scheme, with concrete parameter choice, that gives $128 - bit$ signatures while the best known attack to forge a signature is in 2^{80}. It is based on the basic HFE scheme proposed on Eurocrypt 1996 along with several modifications, such that each of them gives a scheme that is (quite clearly) strictly more secure. The basic HFE has been attacked recently by Shamir and Kipnis (cf [3]) and independently by Courtois (cf this RSA conference) and both these authors give subexponential algorithms that will be impractical for our parameter choices. Moreover our scheme is a modification of HFE for which there is no known attack other that inversion methods close to exhaustive search in practice. Similarly there is no method known, even in theory to distinguish the public key from a random quadratic multivariate function.

QUARTZ is so far the only candidate for a practical signature scheme with length of 128-bits.

QUARTZ has been accepted as a submission to NESSIE (New European Schemes for Signatures, Integrity, and Encryption), a project within the Information Societies Technology (IST) Programme of the European Commission.

1 Introduction

In the present document, we describe the QUARTZ public key signature scheme.

QUARTZ is a HFEV$^-$ algorithm (see [4, 5]) with a special choice of the parameters. QUARTZ belongs to the family of "multivariate" public key schemes, *i.e.* each signature and each hash of the messages to sign are represented by some elements of a small finite field K.

QUARTZ is designed to generate very very short signatures: only 128 bits! Moreover, in QUARTZ, all the state of the art ideas to enforce the security of such an algorithm have been used: QUARTZ is built on a "Basic HFE" scheme secure by itself at present (no practical attack are known for our parameter choice) and, on this underlying scheme, we have introduced some "perturbation operations" such as removing some equations on the originally public key, and introducing some extra variables (these variables are sometime called "vinegar variables"). The resulting schemes look quite complex at first sight, but it can be seen as the resulting actions of

many ideas in the same direction: to have a very short signature with maximal security (*i.e.* the "hidden" polynomial F of small degree d is hidden as well as possible).

As a result, the parameters of QUARTZ have been chosen in order to satisfy an extreme property that no other public key scheme has reached so far: very short signatures. QUARTZ has been specially designed for very specific applications because we thought that for all the classical applications of signature schemes, the classical algorithms (RSA, Fiat-Shamir, Elliptic Curves, DSA, etc) are very nice, but they all generate signatures of 320 bits or more (1024 for RSA) with a security in 2^{80}, so it creates a real practical need for algorithms such as QUARTZ.

QUARTZ was designed to have a security level of 2^{80} with the present state of the art in Cryptanalysis.

2 QUARTZ: the basic ideas

(This paragraph is here to help the understanding of QUARTZ. QUARTZ will then be described in details in the next paragraphs.)

Let $K = \mathbf{F}_q = \mathrm{GF}(q)$ be a small finite field (in QUARTZ we will choose $K = \mathbf{F}_2$). Let d and n be two integers (in QUARTZ we will have $d = 129$ and $n = 103$).

Let α_{ij}, $1 \leq i \leq n$, $1 \leq j \leq n$, be some elements of $\mathbf{F}^{q^n}$ such that:

$$\forall i,j,\ 1 \leq i \leq n,\ 1 \leq j \leq n, q^i + q^j > d \Rightarrow \alpha_{ij} = 0.$$

Let β_i, $1 \leq i \leq n$, be some elements of $\mathbf{F}_{q^n}$ such that

$$\forall i,\ 1 \leq i \leq n, q^i > d \Rightarrow \beta_i = 0.$$

Let γ be an element of $\mathbf{F}_{q^n}$.

Now let F be the following function:

$$F : \begin{cases} \mathbf{F}_{q^n} \to \mathbf{F}_{q^n} \\ X \mapsto \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{n-1} \beta_i X^{q^i} + \gamma \end{cases}$$

This function F can be seen in two different ways:

1. It can be seen as a polynomial function with only one variable $x \in \mathbf{F}_{q^n}$, of degree d.
2. Or, if we write this function F as a function from $\mathbf{F}_{q^n}$ to $\mathbf{F}_{q^n}$ (*i.e.* if we consider $\mathbf{F}_{q^n}$ as a vector space over $\mathbf{F}_q$), it can be seen as a multivariate function of n variables $(x_1, \ldots, x_n) \in K^n$ to n variables $(y_1, \ldots, y_n) \in K^n$ of total degree 2.

   Note: Here the total degree is only 2 because all the functions $X \mapsto X^{q^i}$ are linear functions over $\mathbf{F}_{q^n}$, *i.e.* they can be written as functions from K^n to K^n of total degree one.

From the univariate representation (1) it is possible when d is not too large to invert F (*i.e.* to compute all the roots of $F(X) = Y$ when Y is a given element of $\mathbf{F}_{q^n}$). (Some root finding algorithms exist, such as the Berlekamp algorithm for example, for these univariate algorithms. Their complexity is polynomial in d, so d cannot be too large if we want those algorithms to be efficients.)

From the multivariate representation (2) we will be able to "hide" this function F by introducing two secret bijective affine transformations s and t from K^n to K^n, and we will compute $G' = t \circ F \circ s$, and keep F secret.

This function G' is a quadratic function from K^n to K^n.

Now, two other ideas will be introduced.

Remark: These two other ideas, that we denote by "$-$" and "V", are introduced in order to enforce the security of the scheme, as we will explain in section 8. However, the scheme might be secure even if we did not add these two ideas.

First, we will not publish all the n quadratic equations that define G', but only $n-r$ of these equations ($r = 3$ in the QUARTZ algorithm).

Secondly, we will "mix" the n variables $x_1, \ldots, x_n$ with v "extra variables" ($v = 4$ in the QUARTZ algorithm). These v "extra variables" will be introduced in the β_i and γ parameters. (We will describe in detail in section 4 how this will be done.)

Finally, we obtain a trapdoor one-way function G from 107 bits to 100 bits. Without any secret it is possibe to compute $y = G(x)$ when x is given, and with a secret it is possible to compute all the values of x such that $G(x) = y$ when y is given (x:107 bits, y: 100 bits).

Remark: QUARTZ is a special case of a more general scheme called HFEV$^-$. This scheme is described in [4] and [5]. However, there are many possible parameters in HFEV$^-$, so that we think it is interesting to give an example of the possible choices of these parameters to obtain 128 bit public key digital signatures with 2^{80} security (with the best known attacks).

3 The birthday paradox: how can a digital signature be as short as 128 bits with 2^{80} security

In all signature schemes in which checking the validity of the signature S of a message M consists in verifying an equation $f(S) = g(M)$, where f and g are two public functions, it is always possible, from the birthday paradox, to find a signature S and a message M such that S will be a valid signature of M, after approximately $\sqrt{2^n}$ computations (and storages), where n is the number of bits of the signature and the number of output bits of f and g. (Just store $\sqrt{2^n}$ values $g(M)$, compute $\sqrt{2^n}$ values $f(S)$ and look for a collision).

However, with QUARTZ, we will avoid this "birthday" attack because checking the validity of the signature S of a message M consists in verifying an equation $f(S,M) = 0$, where f is a public function.

Remark If G denotes the trapdoor one-way function from 107 bits to 100 bits that we will use, four computations of this function G will be needed to check whether $f(S,M) = 0$ in the QUARTZ algorithm, as we will see below. A more general theory about how small a digital signature can be, can be found in the extended version of [4], available from the authors (or from our Web page http://www.smartcard.bull.com/sct/uk/partners/bull/index.html).

However, with a signature of only 128 bits, there is still something to be careful with: no more than 2^{64} messages must be signed with the same public key. If more than 2^{64} messages are signed with the same public key, there is a large probability that two different messages will have the same signature and this may create troubles for some applications. However, this is not a very restrictive fact for practical applications since here, only the people who know the secret key can

create or avoid this 2^{64} birthday fact. Somebody who does not know the secret key cannot use this fact to create an attack on the signature scheme with 2^{64} complexity.

This explains why in QUARTZ, the best known attacks are in 2^{80}, despite the fact that the length of the signature is only 128 bits.

4 Notations and Parameters of the Algorithm

In all the present document, $||$ will denote the "concatenation" operation. More precisely, if $\lambda = (\lambda_0, \ldots, \lambda_m)$ and $\mu = (\mu_0, \ldots, \mu_n)$ are two strings of bits, then $\lambda||\mu$ denotes the string of bits defined by:

$$\lambda||\mu = (\lambda_0, \ldots, \lambda_m, \mu_0, \ldots, \mu_n).$$

For a given string $\lambda = (\lambda_0, \ldots, \lambda_m)$ of bits and two integers r, s, such that $0 \leq r \leq s \leq m$, we denote by $[\lambda]_{r \to s}$ the string of bits defined by:

$$[\lambda]_{r \to s} = (\lambda_r, \lambda_{r+1}, \ldots, \lambda_{s-1}, \lambda_s).$$

The QUARTZ algorithm uses the field $\mathcal{L} = \mathbf{F}_{2^{103}}$. More precisely, we chose $\mathcal{L} = \mathbf{F}_2[X]/(X^{103} + X^9 + 1)$. We will denote by φ the bijection between $\{0,1\}^{103}$ and $\mathcal{L}$ defined by:

$$\forall \omega = (\omega_0, \ldots, \omega_{102}) \in \{0,1\}^{103},$$

$$\varphi(\omega) = \omega_{102} X^{102} + \ldots + \omega_1 X + \omega_0 \pmod{X^{103} + X^9 + 1}.$$

4.1 Secret parameters

1. An affine secret bijection s from $\{0,1\}^{107}$ to $\{0,1\}^{107}$. Equivalently, this parameter can be described by the 107×107 square matrix and the 107×1 column matrix over $\mathbf{F}_2$ of the transformation s with respect to the canonical basis of $\{0,1\}^{107}$.
2. An affine secret bijection t from $\{0,1\}^{103}$ to $\{0,1\}^{103}$. Equivalently, this parameter can be described by the 103×103 square matrix and the 103×1 column matrix over $\mathbf{F}_2$ of the transformation s with respect to the canonical basis of $\{0,1\}^{103}$.
3. A family of secret functions $(F_V)_{V \in \{0,1\}^4}$ from $\mathcal{L}$ to $\mathcal{L}$, defined by:

$$F_V(Z) = \sum_{\substack{0 \leq i < j < 103 \\ 2^i + 2^j \leq 129}} \alpha_{i,j} \cdot Z^{2^i + 2^j} + \sum_{\substack{0 \leq i < 103 \\ 2^i \leq 129}} \beta_i(V) \cdot Z^{2^i} + \gamma(V).$$

   In this formula, each $\alpha_{i,j}$ belongs to $\mathcal{L}$ and each β_i $(0 \leq i < 103)$ is an affine transformation from $\{0,1\}^7$ to $\mathcal{L}$, *i.e.* a transformation satisfying

$$\forall V = (V_0, V_1, V_2, V_3) \in \{0,1\}^4,\ \beta_i(V) = \sum_{k=0}^{3} V_k \cdot \xi_{i,k}$$

   with each $\xi_{i,k}$ being an element of $\mathcal{L}$. Finally, γ is a quadratic transformation from $\{0,1\}^7$ to $\mathcal{L}$, *i.e.* a transformation satisfying

$$\forall V = (V_0, V_1, V_2, V_3) \in \{0,1\}^4,\ \gamma(V) = \sum_{k=0}^{3} \sum_{\ell=0}^{3} V_k V_\ell \cdot \eta_{k,\ell}$$

   with each $\eta_{k,\ell}$ being an element of $\mathcal{L}$.
4. A 80-bit secret string denoted by Δ.

4.2 Public parameters

The public key consists in the function G from $\{0,1\}^{107}$ to $\{0,1\}^{100}$ defined by:

$$G(X) = \Big[t\Big(\varphi^{-1}(F_{[s(X)]_{103\to 106}}(\varphi([s(X)]_{0\to 102})))\Big)\Big]_{0\to 99}.$$

By construction of the algorithm, G is a quadratic transformation over $\mathbf{F}_2$, *i.e.* $(Y_0,\ldots,Y_{99}) = G(X_0,\ldots,X_{106})$ can be written, equivalently:

$$\begin{cases} Y_0 = P_0(X_0,\ldots,X_{106}) \\ \quad\vdots \\ Y_{99} = P_{99}(X_0,\ldots,X_{106}) \end{cases}$$

with each P_i being a quadratic polynomial of the form

$$P_i(X_0,\ldots,X_{106}) = \sum_{0\le j<k<107} \zeta_{i,j,k} X_j X_k + \sum_{0\le j<107} \nu_{i,j} X_j + \rho_i,$$

all the elements $\zeta_{i,j,k}$, $\nu_{i,j}$ and ρ being in $\mathbf{F}_2$.

5 Signing a message

In the present section, we describe the signature of a message M by the QUARTZ algorithm.

5.1 The signing algorithm

The message M is given by a string of bits. Its signature S is obtained by applying successively the following operations (see figure 1):

1. Let M_1, M_2 and M_3 be the three 160-bit strings defined by:

$$M_1 = \text{SHA-1}(M),$$

$$M_2 = \text{SHA-1}(M_1),$$

$$M_3 = \text{SHA-1}(M_2).$$

2. Let H_1, H_2, H_3 and H_4 be the four 100-bit strings defined by:

$$H_1 = [M_1]_{0\to 99},$$

$$H_2 = [M_1]_{100\to 159} || [M_2]_{0\to 39},$$

$$H_3 = [M_2]_{40\to 139},$$

$$H_4 = [M_2]_{140\to 159} || [M_3]_{0\to 79}.$$

3. Let $\tilde{S}$ be a 100-bit string. $\tilde{S}$ is initialized to $00\ldots0$.
4. For $i = 1$ to 4, do
   (a) Let Y be the 100-bit string defined by:

$$Y = H_i \oplus \tilde{S}.$$

(b) Let W be the 160-bit string defined by:

$$W = \text{SHA-1}(Y||\Delta).$$

(c) Let R be the 3-bit string defined by:

$$R = [W]_{0\to 2}.$$

(d) Let V be the 4-bit string defined by:

$$V = [W]_{3\to 6}.$$

(e) Let B be the element of $\mathcal{L}$ defined by:

$$B = \varphi\Big(t^{-1}(Y||R)\Big).$$

(f) Consider the following univariate polynomial equation in Z (over $\mathcal{L}$):

$$F_V(Z) = B.$$

- If this equation has a unique solution in $\mathcal{L}$, then let A be this solution.
- Else replace W by SHA-1(W) and go back to (c).

(g) Let X be the 107-bit string defined by:

$$X = s^{-1}\Big(\varphi^{-1}(A)||V\Big).$$

(h) Define the new value of the 100-bit string $\tilde{S}$ by:

$$\tilde{S} = [X]_{0\to 99}\ ;$$

(i) Let X_i be the 7-bit string defined by:

$$X_i = [X]_{100\to 106}.$$

5. The signature S is the 128-bit string given by:

$$S = \tilde{S}||X_4||X_3||X_2||X_1.$$

5.2 Solving the equation $F_V(Z) = B$

To sign a message, we need to solve an equation of the form $F_V(Z) = B$, with B belonging to $\mathcal{L}$ and Z being the unknown, also in $\mathcal{L}$. More precisely, if we refer to step 4.f in section 5.1, we must:

1. Decide whether there is a unique solution or not;
2. In the case of a unique solution, find it.

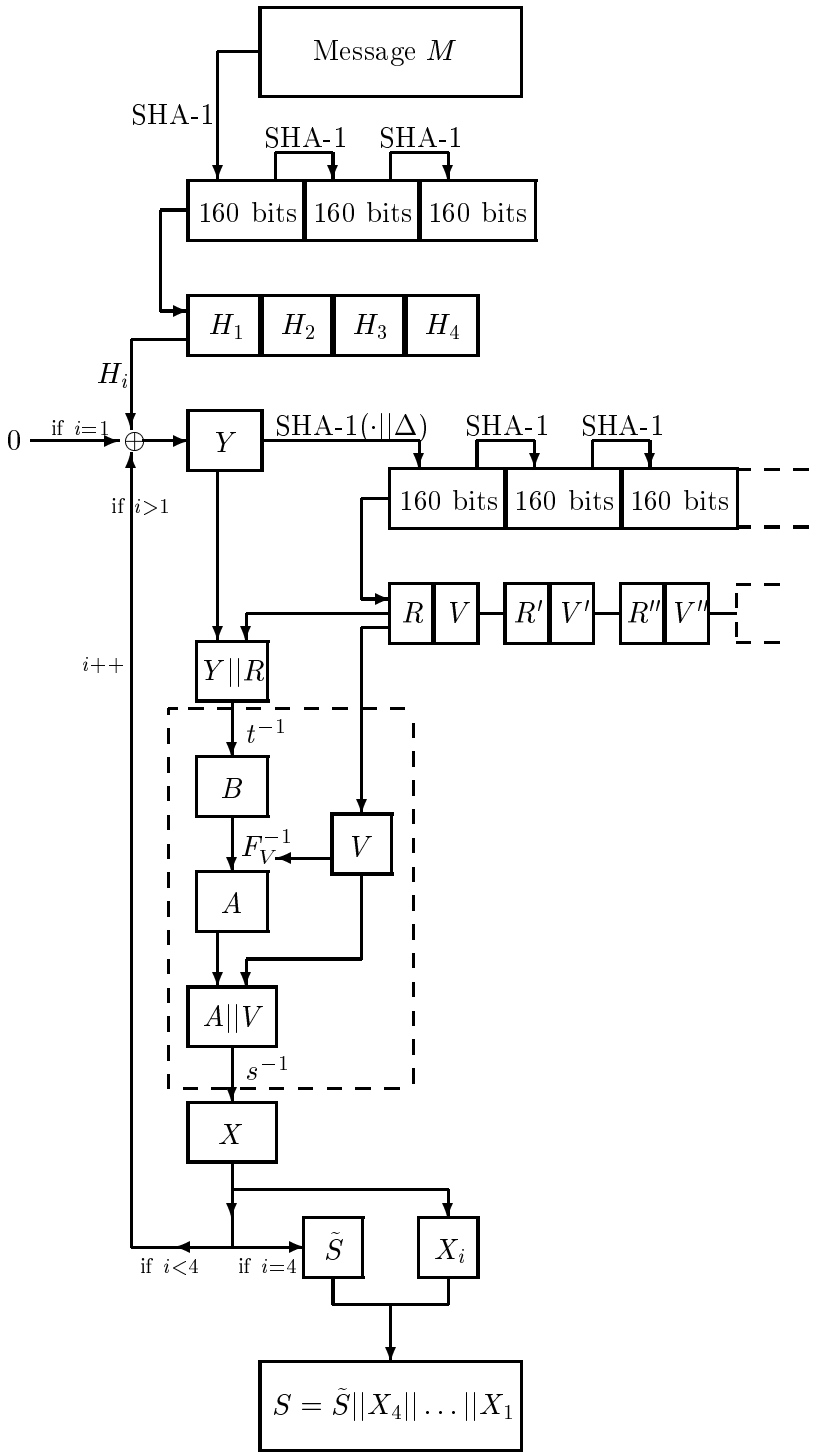


FIG. 1 – *Signature generation with QUARTZ (beginning with $i = 1$)*

The following method can be used: we compute the polynomial

$$\Psi(Z) = \gcd\Big(F_V(Z) - B, Z^{2^{103}} - Z\Big).$$

The equation $F_V(Z) = B$ has a number of solutions (in $\mathcal{L}$) equal to the degree of Ψ over $\mathcal{L}$. As a consequence, if Ψ is *not* of degree one, then the number of solutions is *not* one. On the contrary, if Ψ is of degree one, it is of the form $\Psi(Z) = \kappa \cdot (Z - A)$ (with $\kappa \in \mathcal{L}$) and A is the unique solution of the equation $F_V(Z) = B$.

To compute the gcd above, we can first recursively compute $Z^{2^i} \mod (F_V(Z) - B)$ for $i = 0,1,\ldots,103$ and then compute $\Theta(Z) = Z^{2^{103}} - Z \mod (F_V(Z) - B)$. Finally $\Psi(Z)$ is easily obtained by

$$\Psi(Z) = \gcd\Big(F_V(Z) - B, \Theta(Z)\Big).$$

Thanks to this method, the degrees of the polynomials involved in the computation never exceed $2 \times 129 = 258$.

Note that more refined methods have also been developed to compute $\Psi(Z)$ (see [2]).

5.3 Existence of the signature

The success of the signing algorithm relies on the following fact: for at least one of the successive values of the pair (R,V), there exist a unique solution (in Z) for the equation $F_V(Z) = B$.

It can be proven that, for a randomly chosen B, the probability of having a unique solution in Z is approximately $\frac{1}{e}$. If we suppose that the successive values (R,V) take all the possible values in $\{0,1\}^7$, the probability of never having a unique solution is approximately given by:

$$\Big(1 - \frac{1}{e}\Big)^{128} \simeq 2^{-85}.$$

Since the signing algorithm has to solve this equation four times, the probability that the algorithm fails is:

$$\mathcal{P} \simeq 1 - \Big(1 - \Big(1 - \frac{1}{e}\Big)^{128}\Big)^4 \simeq 2^{-83}.$$

This probability is thus completely negligible.

6 Verifying a signature

Given a message M (*i.e.* a string of bits) and a signature S (a 128-bit string), the following algorithm is used to decide whether S is a valid signature of M or not:

1. Let M_1, M_2 and M_3 be the three 160-bit strings defined by:

$$M_1 = \text{SHA-1}(M),$$

$$M_2 = \text{SHA-1}(M_1),$$

$$M_3 = \text{SHA-1}(M_2).$$

2. Let H_1, H_2, H_3 and H_4 be the four 100-bit strings defined by:

$$H_1 = [M_1]_{0\to 99},$$

$$H_2 = [M_1]_{100\to 159}||[M_2]_{0\to 39},$$

$$H_3 = [M_2]_{40\to 139},$$

$$H_4 = [M_2]_{140\to 159}||[M_3]_{0\to 79}.$$

3. Let $\tilde{S}$ be the 100-bit string defined by:

$$\tilde{S} = [S]_{0\to 99}.$$

4. Let X_4, X_3, X_2, X_1 be the four 7-bit string defined by:

$$X_4 = [S]_{100\to 106},$$

$$X_3 = [S]_{107\to 113},$$

$$X_2 = [S]_{114\to 120},$$

$$X_1 = [S]_{121\to 127}.$$

5. Let U be a 100-bit string. U is initialized to $\tilde{S}$.
6. For $i = 4$ down to 1, do

   (a) Let Y be the 100-bit string defined by:

   $$Y = G(U||X_i).$$

   (b) Define the new value of the 100-bit string U by:

   $$U = Y \oplus H_i.$$

7. – If U is equal to the 100-bit string $00\ldots0$, accept the signature.
   – Else reject the signature.

7 Computation of the G function

The verification algorithm of QUARTZ requires the fast evaluation of the function G, which can be viewed as a set of 100 public quadratic polynomials of the form

$$P_i(x_0,\ldots,x_{106}) = \sum_{0\le j<k<107} \zeta_{i,j,k} x_j x_k + \sum_{0\le j<107} \nu_{i,j} x_j + \rho_i \qquad (0 \le i \le 99)$$

(see section 3.2).

To perform this computation, three methods can be used:

First method:

We can proceed directly, *i.e.* by successively compute the multiplications and the additions involved in P_i.

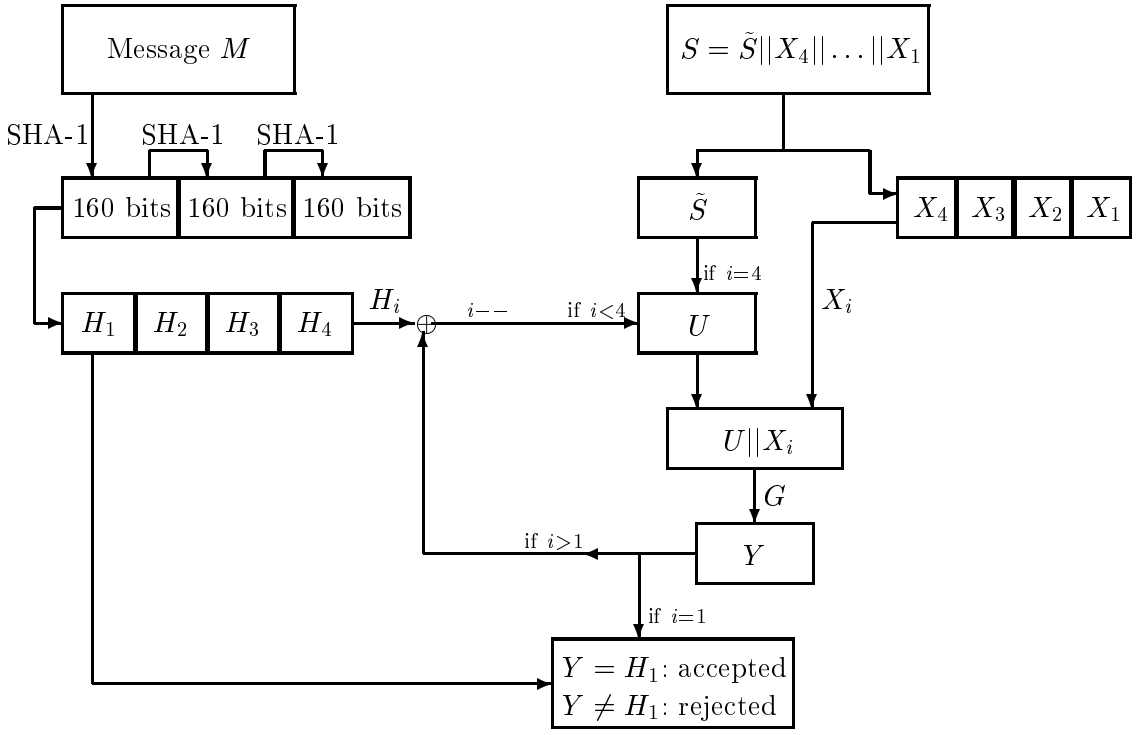


FIG. 2 – *Signature verification with QUARTZ (beginning with $i = 4$)*

Second method:

Each of the P_i can be rewritten as follows:

$$P_i(x_0,\ldots,x_{106}) = x_0\ell_{i,0}(x_0,\ldots,x_{106}) + x_1\ell_{i,1}(x_1,\ldots,x_{106}) + \\ +\ldots + x_{106}\ell_{i,106}(x_{106}) + \rho_i,$$

with the $\ell_{i,0},\ldots,\ell_{i,106}$ $(0 \leq i \leq 99)$ being 107×100 linear forms that can be explicited. As a result, since each x_j equals 0 or 1, we just have to compute modulo 2 additions of x_j variables.

Third method:

Another possible technique consists in writing

$$G(x_0,\ldots,x_{106}) = \sum_{0\leq j<k<107} x_j x_k \cdot Z_{j,k} \oplus \sum_{0\leq j<107} x_j \cdot N_j \oplus R$$

with

$$Z_{j,k} = (\zeta_{0,j,k},\zeta_{1,j,k},\ldots,\zeta_{99,j,k})\,,$$
$$N_j = (\nu_{0,j},\nu_{1,j},\ldots,\nu_{99,j})$$

and

$$R = (\rho_0,\rho_1,\ldots,\rho_{99})\,.$$

The computation can then be performed as follows:

1. Let Y be a variable in $\{0,1\}^{100}$. Let Y be initialized to $R = (\rho_0,\rho_1,\ldots,\rho_{99})$.

2. For each monomial $x_j x_k$ $(0 \leq j < k < 107)$: if $x_j = x_k = 1$ then replace Y by $Y \oplus Z_{j,k}$.
3. For each monomial x_j $(0 \leq j < 107)$: if $x_j = 1$ then replace Y by $Y \oplus N_j$.

If, for instance, we use a 32-bit architecture, this leads to a speed-up of the algorithm: each vector $Z_{j,k}$ or N_j or R can be stored in four 32-bit registers. By using the 32-bit XOR operation, the $\oplus$ operations can be performed 32 bits by 32 bits. This means that we compute 32 public equations simultaneously.

8 Security of the QUARTZ algorithm

Traditionally, the security of public key algorithms relies on a problem which is both simple to describe and has the reputation to be difficult to solve (such as the factorization problem, or the discrete logarithm problem). On the opposite, traditionally, the security of secret key algorithms and of hash functions relies (not on such a problem but) on specific arguments about the construction (such as the soundness of the Feistel construction for example) and on the fact that the known cryptanalytic tools are far to break the scheme.

There are some exceptions. For example the public key scheme based on error correcting codes (such as the McEliece scheme, or the Niederreiter scheme) or the NTRU scheme do not have a security that provably relies on a well defined problem, and some hash functions have been designed on the discrete logarithm problem.

The security of the QUARTZ algorithm is also not proved to be equivalent to a well defined problem. However we have a reasonable confidence in its security due to some arguments that we will present in the sections below, and these arguments are not only subjective arguments.

Remark: As an example, let $\mathcal{F}$ be the composition the five AES finalists, with five independent keys of 128 bits. Almost everybody in the cryptographic community thinks that this $\mathcal{F}$ function will be a very secure function for the next 20 years, despite the fact that it security is not provably relied on a clearly, famous, and simple to describe problem.

Our (reasonable) confidence in the security of QUARTZ comes from the following five different kinds of arguments, that we will explain in more details below:

1. All the known attacks are far from being efficient.
2. There is a kind of "double layered" security in the design of the scheme: algebraic and combinatorial.
3. MQ looks really difficult in average (not only in worst case).
4. When the degree d (of the hidden polynomial F) increases, the trapdoor progressively disappears so that all the attacks must become more and more intractable.
5. The secret key is rather long (but it can be generated from a small seed of 80 bits for example), even for computing very short signatures.

8.1 All the known attacks are far from being efficient

Three kinds of attacks have been studied so far on schemes like the basic HFE or HFEV$^-$ (QUARTZ is a HFEV$^-$ scheme with a special choice for the parameters).

Some attacks are designed to recover the secret key (or an equivalent information)

In this family of attack, we have the exhaustive search of the key (of course intractable) and the (much more clever) Shamir-Kipnis on the basic HFE scheme (cf [3]). However this Shamir-Kipnis attack would not be efficient on the QUARTZ algorithm (much more than 2^{80} computations are required) even if we removed the − and V perturbations. Moreover, the Shamir-Kipnis seems to work only for the basic HFE scheme (*i.e.* without the perturbations − and V) and in QUARTZ we have some − and V. So in fact, at present for a scheme like QUARTZ we do not see how the Shamir-Kipnis attack may work at all.

Some attacks are designed to compute a signature S from a message M directly from the equations of the public key, as if there was no trapdoor (*i.e.* by solving a general system of quadratic equations)

The MQ (= Multivariate Quadratic) problem of solving a general set of multivariate quadratic equations is a NP-Hard problem. Some (non polynomial but sometimes better than exhaustive search) algorithms have been designed for this problem, such as some Gröbner bases algorithms, or the XL and FXL algorithms (see [1]) but for our choices of the QUARTZ parameters, all these algorithms need more than 2^{80} computations.

Some attacks are designed to compute a signature S from a message M by detecting some difference on the public key compared to a system of general quadratic equations

Many analysis have been made in these lines of attacks. Some "affine multiple attacks" have been design, and many variations around these attacks ("higher degree attacks" etc). At present, with the parameters of the QUARTZ algorithm all these attacks need more the 2^{80} computations.

8.2 There is a kind of "double layered" security in the design of the scheme: algebraic and combinatorial

The security of the basic HFE scheme (*i.e.* a HFE scheme with no perturbations such as − and V) can be considered as a kind of "Algebraic" problem, since from the Shamir-Kipnis attack we know that it can be linked to a MinRank problem on very large algebraic fields. (The general MinRank problem is NP-Hard, but for the basic HFE it may not be NP-Hard, but it is still not polynomial when d is not fixed and $d = \mathcal{O}(n)$ for example). However this basic HFE scheme is Hidden in the QUARTZ algorithm with the perturbations − and V. To remove these perturbations seems to be a very difficult combinatorial problem. So to break the QUARTZ scheme, it is expected that a cryptanalyst will have to solve a double problem: Combinatorial and Algebraic, and these problems do not appear separately but in a deeply mixed way to him on the public key.

8.3 MQ looks really difficult in average (not only in worst case)

In the past, some public key schemes apparently (not provably) based on some NP-Hard problems, such as the Knapsack problem were broken. However the MQ problem (*i.e.* solving a general set of multivariate quadratic equations) seems to be a much more difficult problem to solve than the Knapsack Problem: on the Knapsack Problem an algorithm such as LLL is very often efficient, while on the opposite ? on the MQ problem all the known algorithms are not

significantly better than exhaustive search when the number m of equations is about the same as the number n of variables and is larger than, say, about 12.

It is also interesting to notice that almost all the "Knapsack Schemes" were broken due to a new algorithm on the general Knapsack problem (LLL) and not due to the fact that the security of these schemes was not properly proved to be equivalent to the Knapsack problem. Something similar seems to appear with the schemes based on error correcting codes, such as the McEliece Scheme, or the Niederreiter scheme: so far all the attacks on these schemes try to solve the general (and NP-Hard) problem of decoding a word of small weight in a general linear code, and not to try to use the fact that it is not proved that the security of these schemes is equivalent to solving this problem. If, for these schemes as for QUARTZ the practical cryptanalysis becomes in practice the problem of solving the general problem, then for QUARTZ the MQ problem looks really very difficult.

8.4 When the degree d (of the hidden polynomial F) increases, the trapdoor progressively disappears so that all the attacks must become more and more intractable

The degree d of the QUARTZ algorithm is fixed to 129. However if d was not fixed, and d could be as large as $2h$ ($h = 103$ in the QUARTZ algorithm), then all the possible systems of quadratic equations would appear in the public key, so the problem of solving it would be exactly as hard as the general MQ problem (on this number of variables). Of course, we have fixed d to 129 in order to be able to compute a signature in a reasonable time on a computer, but this result shows that when d increases, the trapdoor progressively disappears, so that all the attacks must become more and more intractable. So d is really an important "security parameter". Our choice of $d = 129$ has been made to be far from the current state of the art on the cryptanalysis with small d while still having a reasonable time on a computer to compute a signature.

8.5 The secret key is rather long (but it can be generated from a small seed of 80 bits for example), even for computing very short signatures

Many secrets are used in QUARTZ: the secret affine permutations s and s, the secret function F, the secret vinegar variables V, and the secret removed equations. To specify all the secret we need a rather long secret key. However, it is also possible to compute this secret key from a short seed by using any pseudorandom bit generator. In general the time to generate the secrets from the small seed will not increase a lot the time to generate a signature. Moreover it has to be done only once if we can store the secret key in a safe way on the computer. So for practical applications it is always possible to generate the secret key from a seed of, say, 80 bits, but this secret key for a cryptanalyst of QUARTZ will always be similar to a much larger secret key.

So QUARTZ has a property that already existed in schemes like DSS (where the lengths of p and q are different): the length of the secret key is not directly linked to the length of the signature. (This property does not exist in RSA, where the length of the secret key is never larger than the length of the signature. It explain why a QUARTZ or DSS signature can be much smaller than a RSA signature).

The fact that a cryptanalyst of QUARTZ has to face such a large secret key, may also be an argument to say that in practice the time to find a QUARTZ secret key may be intractable in practice, even if a new sub-exponential algorithm is found and used. (So far many cryptanalysis, such as the "affine multiple attacks", have to solve huge systems of linear equations by Gaussian reductions, and often the number of variables in these systems increases very fast with the length

of the secret, so these attacks become impractical due to space and time limitations). However this argument is not very convincing and is maybe not as strong as the other arguments presented above.

9 Summary of the characteristics of QUARTZ

- Length of the signature: 128 bits.
- Length of the public key: 71 Kbytes.
- Length of the secret key: the secret key (3 Kbytes) is generated from a small seed of at least 128 bits.
- Time to sign a message[1]: 30 seconds on average.
- Time to verify a signature[2]: less than 0.12 ms (3×0.006 ms for the three SHA-1, plus 0.1 ms for the quadratic equations).
- Best known attack: more than 2^{80} computations.

References

[1] N. Courtois, A. Shamir, J. Patarin, A. Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, in Advances in Cryptology, Proceedings of EUROCRYPT'2000, LNCS n° 1807, Springer, 2000, pp. 392-407.

[2] E. Kaltofen, V. Shoup, *Fast polynomial factorization over high algebraic extensions of finite fields*, in Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, 1997.

[3] A. Kipnis, A. Shamir, *Cryptanalysis of the HFE public key cryptosystem*, in Advances in Cryptology, Proceedings of Crypto'99, LNCS n° 1666, Springer, 1999, pp. 19-30.

[4] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms*, in Advances in Cryptology, Proceedings of EUROCRYPT'96, LNCS n° 1070, Springer Verlag, 1996, pp. 33-48.

[5] A. Kipnis, J. Patarin and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, in Advances in Cryptology, Proceedings of EUROCRYPT'99, LNCS n° 1592, Springer, 1999, pp. 206-222.

[6] The HFE cryptosystem web page: http://www.hfe.minrank.org

1. On a Pentium III 500 MHz. This part can be improved: the given software was not optimized.
2. This part can be improved: the given software was not optimized.

FLASH, a fast multivariate signature algorithm

CT-RSA'2001 *(Version complète)*
Article avec Nicolas Courtois et Jacques Patarin (Bull CP8)

Abstract

This article describes the particular parameter choice and implementation details of one of the rare published, but not broken signature schemes, that allow signatures to be computed and checked by a low-cost smart card. The security is controversial, since we have no proof of security, but the best known attacks require more than 2^{80} computations. We called FLASH our algorithm and we also proposed SFLASH, a version that has a smaller public key and faster verification though one should be even more careful about it's security.

FLASH and SFLASH have been accepted as submissions to NESSIE (New European Schemes for Signatures, Integrity, and Encryption), a project within the Information Societies Technology (IST) Programme of the European Commission.

1 Introduction

In the present document, we describe the FLASH public key signature scheme.

FLASH is a C^{*--} algorithm (see [4]) with a special choice of the parameters. FLASH belongs to the family of "multivariate" public key schemes, *i.e.* each signature and each hash of the messages to sign are represented by some elements of a small finite field K.

FLASH is designed to be a very fast signature scheme, both for signature generation and signature verification. It is much faster in signature than RSA and much easier to implement on smart cards without any arithmetic coprocessor for example. However its public key size is larger than the public key size of RSA. Nevertheless this public key size can fit in current smart cards. It may also be noticed that, with the secret key, it is possible to sign AND to check the signature (generated with this particular secret key) without the need of the public key (in some applications this may be useful).

As a result, the parameters of FLASH have been chosen in order to satisfy an extreme property that very few public key scheme have reached so far: efficiency on low-price smart cards. FLASH has been specially designed for this specific application because we thought that for all the classical applications of signature schemes, the classical algorithms (RSA, Fiat-Shamir, Elliptic Curves, DSA, etc) are very nice, but when we need some very specific properties these algorithms just cannot satisfy them, and it creates a real practical need for algorithms such as FLASH.

FLASH was designed to have a security level of 2^{80} with the present state of the art in Cryptanalysis.

2 FLASH: the basic ideas

(This paragraph is here to help the understanding of FLASH. FLASH will then be described in details in the next paragraphs.)

Let $K = \mathbf{F}_q = \mathrm{GF}(q)$ be a small finite field (in FLASH we will choose $K = \mathbf{F}_{256}$ and in SFLASH we will choose $K = \mathbf{F}_{128}$).

Let n and α be two integers (in FLASH and SFLASH we will have $n = 29$ and $\alpha = 11$).

Let F be the following function:

$$F : \begin{cases} \mathbf{F}_{q^n} \to \mathbf{F}_{q^n} \\ x \mapsto x^{1+q^\alpha} \end{cases}$$

This function F can be seen in two different ways:

1. It can be seen as a monomial function with only one variable $x \in \mathbf{F}_{q^n}$, of degree $1 + q^\alpha$.
2. Or, if we write this function F as a function from $\mathbf{F}_{q^n}$ to $\mathbf{F}_{q^n}$, it can be seen as a multivariate function from n variables $(x_1, \ldots, x_n) \in K^n$ to n variables $(y_1, \ldots, y_n) \in K^n$, of total degree 2.

From the univariate representation (1), it is easy to invert F when $1 + q^\alpha$ is coprime to $q^n - 1$ (we will always choose q, n and α such that this condition is satisfied). In this case, it can be proven that the inverse function F^{-1} of F is also a monomial function:

$$F^{-1}(x) = x^h$$

where h is an integer such that

$$h \cdot (1 + q^\alpha) = 1 \bmod (q^n - 1).$$

From the multivariate representation (2), we will be able to "hide" this function F by introducing two secret bijective affine transformations s and t from K^n to K^n, and we will compute $G' = t \circ F \circ s$ and keep F secret. This function G' is a quadratic function from K^n to K^n.

Now, we use another important idea: we will not publish all the n quadratic equations of G', but only $n - r$ of these equations (*ie* r equations will be kept secret). (In FLASH and SFLASH, $r = 11$.)

Let G be the public function from K^n to K^{n-r} obtained like this. Then G will be the public key and t, s and the r equations removed are the secret key. As we will see below from G, we will be able to design the very efficient signature schemes FLASH and SFLASH.

Remark: FLASH and SFLASH are very similar to the scheme C^* published in 1988 by T. Matsumoto and H. Imai (cf [2]). However, there are two major changes:

1. In FLASH and SFLASH, there is only "one branch".
2. In FLASH and SFLASH, r equations of the composition $G' = t \circ F \circ s$ are kept secret, where $q^r \geq 2^{80}$.

Without these changes, the schemes can be broken (see [3] and [4]).

3 Notations and Parameters of the Algorithm

In all the present document, || will denote the "concatenation" operation. More precisely, if $\lambda = (\lambda_0, \ldots, \lambda_m)$ and $\mu = (\mu_0, \ldots, \mu_n)$ are two strings of elements (in a given field), then $\lambda||\mu$ denotes the string of elements (in the given field) defined by:

$$\lambda||\mu = (\lambda_0, \ldots, \lambda_m, \mu_0, \ldots, \mu_n).$$

For a given string $\lambda = (\lambda_0, \ldots, \lambda_m)$ of bits and two integers r, s, such that $0 \leq r \leq s \leq m$, we denote by $[\lambda]_{r \to s}$ the string of bits defined by:

$$[\lambda]_{r \to s} = (\lambda_r, \lambda_{r+1}, \ldots, \lambda_{s-1}, \lambda_s).$$

The FLASH algorithm uses two finite fields.

- The first one, $K = \mathbf{F}_{256}$ is precisely defined as $K = \mathbf{F}_2[X]/(X^8 + X^6 + X^5 + X + 1)$. We will denote by π the bijection between $\{0,1\}^8$ and K defined by:

$$\forall b = (b_0, \ldots, b_7) \in \{0,1\}^8,$$

$$\pi(b) = b_7 X^7 + \ldots + b_1 X + b_0 \pmod{X^8 + X^6 + X^5 + X + 1}.$$

- The second one is $\mathcal{L} = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$. We will denote by φ the bijection between K^{37} and $\mathcal{L}$ defined by:

$$\forall \omega = (\omega_0, \ldots, \omega_{36}) \in K^{37}$$

$$\varphi(\omega) = \omega_{36} X^{36} + \ldots + \omega_1 X + \omega_0 \pmod{X^{37} + X^{12} + X^{10} + X^2 + 1}.$$

3.1 Secret Parameters

1. An affine secret bijection s from K^{37} to K^{37}. Equivalently, this parameter can be described by the 37×37 square matrix and the 37×1 column matrix over K of the transformation s with respect to the canonical basis of K^{37}.
2. An affine secret bijection t from K^{37} to K^{37}. Equivalently, this parameter can be described by the 37×37 square matrix and the 37×1 column matrix over K of the transformation s with respect to the canonical basis of K^{37}.
3. A 80-bit secret string denoted by Δ.

3.2 Public Parameters

The public key consists in the function G from K^{37} to K^{26} defined by:

$$G(X) = (Y_0, Y_1, \ldots, Y_{25}),$$

where

$$Y = (Y_0, Y_1, \ldots, Y_{37}) = t\Big(\varphi^{-1}(F(\varphi(s(X))))\Big).$$

Here F is the function from $\mathcal{L}$ to $\mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{256^{11}+1}.$$

By construction of the algorithm, G is a quadratic transformation over K, *i.e.* $(Y_0,\ldots,Y_{25}) = G(X_0,\ldots,X_{36})$ can be written, equivalently:

$$\begin{cases} Y_0 = P_0(X_0,\ldots,X_{36}) \\ \quad\vdots \\ Y_{25} = P_{25}(X_0,\ldots,X_{36}) \end{cases}$$

with each P_i being a quadratic polynomial of the form

$$P_i(X_0,\ldots,X_{36}) = \sum_{0\le j<k<37} \zeta_{i,j,k} X_j X_k + \sum_{0\le j<37} \nu_{i,j} X_j + \rho_i,$$

all the elements $\zeta_{i,j,k}$, $\nu_{i,j}$ and ρ being in K.

4 Signing a message

In the present section, we describe the signature of a message M by the FLASH algorithm.

4.1 The signing algorithm

The message M is given by a string of bits. Its signature S is obtained by applying successively the following operations (see figure 1):

1. Let M_1 and M_2 be the three 160-bit strings defined by:

$$M_1 = \text{SHA-1}(M),$$

$$M_2 = \text{SHA-1}(M_1).$$

2. Let V be the 208-bit string defined by:

$$V = [M_1]_{0\to 159} || [M_2]_{0\to 47}.$$

3. Let W be the 88-bit string defined by:

$$W = [\text{SHA-1}(V||\Delta)]_{0\to 87}.$$

4. Let Y be the string of 26 elements of K defined by:

$$Y = \Big(\pi([V]_{0\to 7}),\pi([V]_{8\to 15}),\ldots,\pi([V]_{200\to 207})\Big).$$

5. Let R be the string of 11 elements of K defined by:

$$R = \Big(\pi([W]_{0\to 7}),\pi([W]_{8\to 15}),\ldots,\pi([V]_{80\to 87})\Big).$$

6. Let B be the element of $\mathcal{L}$ defined by:

$$B = \varphi\Big(t^{-1}(Y||R)\Big).$$

7. Let A be the element of $\mathcal{L}$ defined by:

$$A = F^{-1}(B),$$

F being the function from $\mathcal{L}$ to $\mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{256^{11}+1}.$$

8. Let $X = (X_0, \ldots, X_{36})$ be the string of 37 elements of K defined by:

$$X = (X_0, \ldots, X_{36}) = s^{-1}\Big(\varphi^{-1}(A)\Big).$$

9. The signature S is the 296-bit string given by:

$$S = \pi^{-1}(X_0)||\ldots||\pi^{-1}(X_{36}).$$

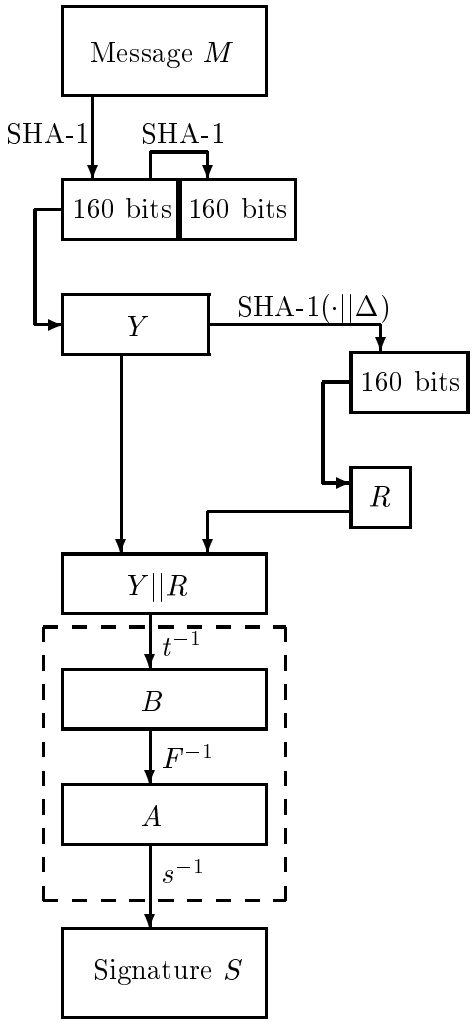


FIG. 1 – *Signature generation with FLASH*

4.2 Computing $A = F^{-1}(B)$

The function F, from $\mathcal{L}$ to $\mathcal{L}$, is defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{256^{11}+1}.$$

As a consequence, $A = F^{-1}(B)$ can be obtained by the following formula:

$$A = B^h,$$

the value of the exponent h being the inverse of $256^{11} + 1$ modulo $256^{37} - 1$. In fact, h can be explicitly given by:

$$h = 2^{295} + \sum_{i=0}^{17} \sum_{j=176i+87}^{176i+174} 2^j.$$

Three methods can be used to compute $A = B^h$:

1. Directly compute the exponentiation B^h by using the "square-and-multiply" principle.
2. Use the following algorithm:
   (a) Initialize A to:
   $$A = B^{2^{87}} \quad \left(= B^{256^{10}} \cdot B^{128} \right).$$
   Note that $B \mapsto B^{256^{10}}$ is a linear transformation of $\mathcal{L}$ if we consider $\mathcal{L}$ as a vector space over K and can thus be easily computed.
   (b) Compute
   $$u = A^{256^{11}-1}.$$
   This value can be computed either by using the "square-and-multiply" principle or by noticing that we also have
   $$u \cdot A = A^{256^{11}}$$
   with $A \mapsto A^{256^{11}}$ being a linear transformation of $\mathcal{L}$ if we consider $\mathcal{L}$ as a vector space over K. We can thus easily find A by solving a system of linear equations over K.
   (c) Apply 18 times the following transformation: replace A by $u \cdot A^{256^{22}}$. This is also practical, since $A \mapsto A^{256^{22}}$ is a linear transformation of $\mathcal{L}$ (considered as a vector space over K).
3. Finally, we can also use the fact that
   $$A \cdot B^{256^{11}} = A^{256^{22}} \cdot B.$$
   Since $B \mapsto B^{256^{11}}$ and $A \mapsto A^{256^{22}}$ are two linear transformations of $\mathcal{L}$ (considered as a vector space over K), A can be found by solving a system of linear equations over K.

5 Verifying a signature

Given a message M (*i.e.* a string of bits) and a signature S (a 296-bit string), the following algorithm is used to decide whether S is a valid signature of M or not:

1. Let M_1 and M_2 be the three 160-bit strings defined by:
   $$M_1 = \text{SHA-1}(M),$$
   $$M_2 = \text{SHA-1}(M_1).$$

2. Let V be the 208-bit string defined by:

$$V = [M_1]_{0\to 159} || [M_2]_{0\to 47}.$$

3. Let Y be the string of 26 elements of K defined by:

$$Y = \Big(\pi([V]_{0\to 7}), \pi([V]_{8\to 15}), \ldots, \pi([V]_{200\to 207})\Big).$$

4. Let Y' be the string of 26 elements of K defined by:

$$Y' = G\Big(\pi([S]_{0\to 7}), \pi([S]_{8\to 15}), \ldots, \pi([S]_{288\to 295})\Big).$$

5. – If Y equals Y', accept the signature.
   – Else reject the signature.

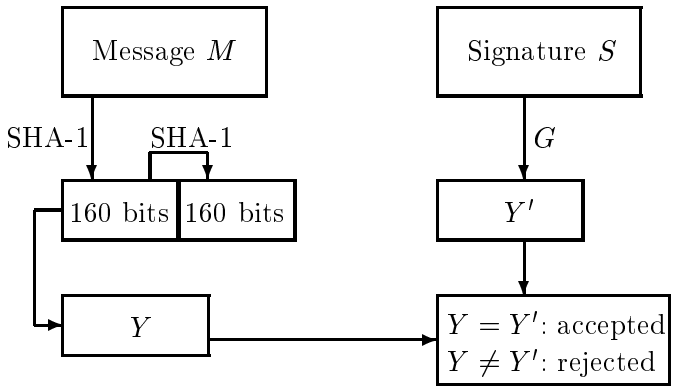


FIG. 2 – *Signature verification with FLASH*

6 Security of the FLASH algorithm

FLASH is a C^{*--} scheme with a special choice of the parameters.

The security of such schemes has been studied in [4].

The security is not proven to be equivalent to a simple to describe and assumed difficult to solve problem. However, here are the present results on the two possible kinds of attacks :

6.1 Attacks that compute a valid signature from the public key as if it was a random set of quadratic equations (*i.e.* without using the fact that we have a C^{*--} scheme)

These attacks have to solve a MQ problem (MQ: Multivariate Quadratic equations), and the general MQ problem is NP-Hard. Moreover, when the parameters are well chosen, the known algorithms for solving such an MQ problem (such as XL, FXL or some Gröbner base algorithms) are efficient. With our choice of parameters for FLASH, they require more computations than the equivalent of 2^{80} operations.

6.2 Attacks that use the fact that the public key comes from a C^{*--} scheme (and is not a random set of quadratic equations)

All the known attacks on this family have a complexity in $\mathcal{O}(qr)$, where r is the number of removed equations ($r = 11$ in the FLASH algorithm), and where q is the number of elements of the finite field K used (so $q = 256 = 2^8$ for the FLASH algorithm). So these attacks will require more than the equivalent of 2^{80} operations for the FLASH algorithm.

7 Summary of the characteristics of FLASH

- Length of the signature: 296 bits.
- Length of the public key: 18 Kbytes.
- Length of the secret key: the secret key (2.75 Kbytes) is generated from a small seed of at least 128 bits.
- Time to sign a message[1]: less than 2.7 ms (maximum time).
- Time to verify a signature[2]: less than 0.8 ms (*i.e.* approximately $37 \times 37 \times 26$ multiplications and additions in K).
- Time to generate a pair of public key/secret key: less than 1 s.
- Best known attack: more than 2^{80} computations.

8 The SFLASH algorithm

In this chapter we introduce a modification of FLASH, that is made in order to have a smaller public key length. For this purpose, we will choose our parameters such that the coefficients of the public key lie in the prime subfield $\mathbf{F}_2$ of K. For this we need to satisfy two conditions. First, the coefficients of the irreducible polynomial that defines $\mathcal{L}$ need to be in $\mathbf{F}_2$. Secondly, the affine invertible transformations s and t need to be defined over $\mathbf{F}_2$.

Moreover, in order to avoid a possible attack described in [1] (this attack uses the factorization of the extension degree 8 and the existence of an intermediate extension), we choose $K = \mathbf{F}_{128}$ in SFLASH.. Then the of [1] attack fails and we obtain an algorithm that is not broken with a difference from FLASH that the public key takes 2.2 Kbytes instead of 18. It is also sensibly faster in verification.

9 Summary of the characteristics of SFLASH

- Length of the signature: 259 bits.
- Length of the public key: 2.3 Kbytes.
- Length of the secret key: the secret key (0.35 Kbytes) is generated from a small seed of at least 128 bits.
- Time to sign a message[3]: less than 2.6 ms (maximum time).
- Time to verify a signature[4]: less than 0.4 ms (*i.e.* approximately $37 \times 37 \times 26 \times \frac{1}{2}$ multiplications and additions in K).

1. On a Pentium III 500 MHz. This part can be improved: the given software was not optimized.
2. This part can be improved: the given software was not optimized.
3. On a Pentium III 500 MHz. This part can be improved: the given software was not optimized.
4. This part can be improved: the given software was not optimized.

- Time to generate a pair of public key/secret key: less than 1 s.
- Best known attack: more than 2^{80} computations.

References

[1] Nicolas Courtois, *Asymmetric cryptography with algebraic problems MQ, MinRank, IP and HFE.* PhD thesis, Paris 6 University, to appear soon.

[2] Tsutomu Matsumoto and Hideki Imai, *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption*, Proceedings of EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[3] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, Proceedings of CRYPTO'95, Springer-Verlag, pp. 248-261.

[4] Jacques Patarin, Louis Goubin, and Nicolas Courtois, C^{*-+} *and HM: Variations around two schemes of T. Matsumoto and H. Imai*, in Advances in Cryptology, Proceedings of ASIACRYPT'98, LNCS n° 1514, Springer Verlag, 1998, pp. 35-49.

A Fast and Secure Implementation of SFLASH

PKC'2003
Article avec Mehdi-Laurent Akkar, Nicolas Courtois et Romain Duteuil (Schlumberger)

Abstract

Sflash is a multivariate signature scheme, and a candidate for standardisation, currently evaluated by the European call for primitives Nessie. The present paper is about the design of a highly optimized implementation of Sflash on a low-cost 8-bit smart card (without coprocessor). On top of this, we will also present a method to protect the implementation protection against power attacks such as Differential Power Analysis.

Our fastest implementation of Sflash takes 59 ms on a 8051 based CPU at 10MHz. Though the security of Sflash is not as well understood as for example for RSA, Sflash is apparently the fastest signature scheme known. It is suitable to implement PKI on low-cost smart card, token or palm devices. It allows also to propose secure low-cost payment/banking solutions.

Key Words: Digital Signatures, PKI, Addition Chains, Multivariate Cryptography, Matsumoto-Imai cryptosystem C^*, C^{*--} trapdoor function, HFE, portable devices, Smart cards, Power Analysis, SPA, DPA.

1 Introduction

The design of Flash and Sflash signature schemes is due to Courtois, Patarin and Goubin [17, 18]. Sflash is based on a so called C^{*--} multivariate signature scheme, the name of C^{*--} being due to Patarin [14]. The idea goes back to the Matsumoto-Imai cryptosystem proposed at Eurocrypt'88, also called C^* in [12], that is a remote cousin of RSA, that uses a power function over an extension of a finite field. At the time the Matsumoto-Imai or C^* cryptosystem was believed very secure, and were amazingly fast. At the same time, in France, the smart cards become a great success: they allow to divide by ten the fraud figures in the payment systems. However RSA is too slow to be used on a smart card, and this keeps the security achieved by smart cards solutions insufficient: unable to implement a real public key signature. From this arises the motivation to find a signature scheme that could be implemented on low-cost smart cards. Unfortunately, at Crypto'95 Patarin shows Matsumoto-Imai is insecure[1], see [13, 10]. Subsequently Patarin studied many other variants and generalizations of Matsumoto-Imai (or C^*) (for example the Dragons). Most of them are broken, and very few remain. Among these, C^{*-} is particularly simple, and remains unbroken. It is simply the original scheme combined

1. Note that H. Dobbertin claims to have independently found this attack in 93/94.

with the idea of preventing structural attacks by simply removing some of the equations that constitute the public key, due initially to Shamir [19]. At Asiacrypt'98 [14], Courtois, Patarin and Goubin show that C^{*-} can be attacked, and if r is the number of removed equations, a factor of q^r appears in the attack. For various reasons it is conjectured that the security of C^{*-} does increase with at least q^r when removing equations [1], and the same is also conjectured for other multivariate cryptosystems[2]. When q^r is very big, e.g. 2^{80}, it is believed that C^{*-} is secure, we then call it C^{*--}, as a lot of equations are removed. It is possible to see that due to many equations removed, C^{*--} can only be used in signature, no longer in encryption.

From C^{*-}, in 2000, Courtois, Patarin and Goubin designed the Flash signature scheme, submitted to Nessie European call for cryptographic primitives, and also a special version of Flash called Sflash that manages to decrease the size of the public key[3]. Unfortunately at Eurocrypt'2002, Gilbert and Minier, showed that this very trick, used to decrease the size of the public key of Sflash, is insecure, and broke Sflash, see [6]. Since then the specification of Sflash has been revised, the new version of Sflash is in fact a version of Flash with a better choice of parameters.

Finally, it is important to note that the design of Flash and Sflash does not reduce to C^{*--}. There is difficulty in the design of multivariate signature schemes that comes from the fact that the systems of equations, have in general many solutions, and only the knowledge of the internal algebraic structure allows to find one of them, which is usually done by fixing some internal variables. If this process is not handled correctly, it might leak information about this internal structure, and eventually allow to recover the private key of Flash/Sflash. See [17] and [1].

Sflash is therefore the best solution that has been found so far to make digital signatures in a low-cost device such as smart card, USB token or a palm device. However, the security of an implementation of a cryptographic algorithm in such a device does not reduce to the security of the cryptographic algorithm itself. It is hard to protect a secret that is entirely in the hands of a potential attacker: the implementation should also have in mind possible side-channel attacks. In 1998, Kocher, Jaffe and Jun showed the feasibility of such attacks [11] using the power consumption of the device, and since then other side-channel attacks have been proposed. In this paper we will also describe, on top of our optimized implementation, a protection against side-channel attacks.

2 Structure of the Smart Card

We consider a low-end smart card built on a 8-bit CPU core, an Intel 8051. It has no arithmetic or cryptographic coprocessor. The memory of this card is divided in three parts as follows:

- The *data*: 128 bytes which one can address directly in one CPU clock.
- The *xdata*: between 1 and 4 Kbytes, indirectly addressable in two CPU clocks.
- The *code*: between 4 and 64 Kbytes of unrewritable memory, indirectly addressable in two CPU clocks.
- Most smart card processors that are based on 8051 contain also between 2 and 128 Kkbytes of E^2PROM memory, that can be used to store keys.

As a comparison one could see the *data*, the *idata* and the *xdata* like the RAM of a classical PC, whereas the *code* would be the ROM. According to those considerations, we will try to store

2. For example see the experiments with Buchberger's algorithm applied to HFE-, presented in [2] (in these proceedings) or on slide 35 of [3].

3. The main drawback of many multivariate cryptosystems.

the most manipulated variables in the *data* in order to save as much time as possible in the computation.

3 Basis Structures and Variables used in Sflash

A complete description of Sflash can be found in [17, 18]. The signature process consists mainly of a composition of three functions defined over the extensions of finite fields: $s^{-1} \circ f^{-1} \circ t^{-1}$, with two multivariate affine functions s,t and one univariate power function f, In this paper we concentrate on the implementation of the basic operations that are used in Sflash.

3.1 Main Structures

The algorithm mainly manipulates elements of the finite field $L = GF(128^{37})$, constructed as an extension of the base Galois field $K = \mathbb{F}128$. The field $K = \mathbb{F}128$ defined as $\mathbb{F}2[X]$ polynomials modulo $(X^7 + X + 1)$:

$$K = \mathbb{F}2[X]/(X^7 + X + 1)$$

Each element of K is written as 7 bits stored in one byte; the coefficient of X^i becomes the coefficient of 2^i, $i = 0..6$. The big field $L = GF(128^{37})$ is defined as follows:

$$L = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1) \quad (1)$$

We also identify L with K^{37} and represent an element of L by 37 elements of K (the coefficients of X^i, $i = 0..36$), that in turn are written as 37 bytes, each of them using only 7 bits.

Our implementation uses two temporary variables called y and *temp*, that are structures containing an element of L, in the *data* zone of the smart card. This allows them to be easily accessible so that the computation is faster. We will also store another structure of type L, called x, in the *xdata* zone because at some moment we need to manipulate three elements of L, when using a function having two inputs and one output (we are not able to store this third one in the *data* as we need additional space in the *data* zone for something else).

3.2 The Private Key of Sflash

The secret parameters of the signature scheme are the two transformations from L to L, s and t, that are multivariate affine functions, *i.e.* they are affine when seen as functions from $K^{37} \to K^{37}$. We do not actually store s and t but their inverses s^{-1} and t^{-1}. We have $t^{-1} : x \mapsto T_m x + T_c$ and $s^{-1} : x \mapsto S_m x + S_c$ with T_m and S_m being two matrixes 37×37 and with T_c and S_c being the constant vectors [4]. All these are stored in either the *code* zone or the E^2PROM of the card.
We will also store the 80-bit secret string Δ in the *code* zone (or E^2PROM).

4. Actually these two constant vectors are not really secret as shown in [8]. Therefore one can choose s and t linear instead of affine, which reduces the size of the secret key.

4 Fast Implementation of the Operations Over the Fields

The Implementation of K:

1. Addition: Easy in characteristic 2 of $K = \mathbb{F}128$, the addition in $K = \mathbb{F}128$ is implemented by XORing the byte representations of the elements.
2. Multiplication: more work is required here. Since the multiplicative group K^* is cyclic, say generated by α, each element of K (but zero) can be seen as a power of α. Powers add when we multiply two non-zero elements: $\alpha^x.\alpha^y = \alpha^{x+y}$, and the multiplication by zero is obvious. In order to execute this operation, we store two tables of 127 bytes, one (named *expo* and stored in the *code* zone) giving the exponent of α corresponding to a given non-zero element of K, and the reciprocal operation (named *log* and also in *code*) giving the element of K corresponding to a given exponent of α.

The Implementation of L: Based on the definition of L in (1).

1. Addition: given two elements of L represented by two polynomials $a = \sum_{i=0}^{36} a_i X^i$ and $b = \sum_{i=0}^{36} b_i X^i$, their sum is computed by XORing the coefficients of the same degree:

$$c \;=\; a + b \;\stackrel{def}{=}\; \sum_{i=0}^{36} (a_i \oplus b_i) X^i$$

   with $\oplus$ being the XOR operation.
2. Multiplication: It is costly, because we compute the product of two polynomials a and b which will be of degree 72, and then compute its euclidian reduction modulo the irreducible polynomial $X^{37} + X^{12} + X^{10} + X^2 + 1$ (as there are no trinomials for this field).
   (a) Build $c' = \sum_{i=0}^{72} c'_i X^i$, where $c_i = \sum_{l+k=i} a_l.b_k$.
   (b) Then reduce c' by $X^{37} + X^{12} + X^{10} + X^2 + 1$, the result c is the product $a.b$ in L.
3. Square: It is interesting to code the squaring operation in L independently, it can be done at least 5 times faster than a multiplication of an element by itself. The above multiplication algorithm requires a quadratic-time computation on the coordinates of the two operands (the building of c' in (a) above) whereas, to square an element a, we only have to compute:

$$a' = \sum_{i=0}^{36} a_i^2 X^{2i}$$

   which is linear in the a_i, and then reduce a' modulo $X^{37} + X^{12} + X^{10} + X^2 + 1$, like in step (b) above.

The Affine Transformations s and t: They are computed as classical matrix multiplications, with additional XOR with the constant vector. As in each step of the matrix multiplication we have to compute several multiplications in K (one of the coordinates of the input with one of the matrix' coefficients), and regarding to how we compute such a multiplication (*cf* above), it will be faster if we store base α logarithms of the coefficients of the matrix.

5 How to Compute $A = f^{-1}(B)$ in Sflash?

We need to compute $A = B^h$ in L, with:

$$\begin{aligned} h &= (128^{11}+1)^{-1} \mod (128^{37}-1) \\ &= 1000000\ 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000 \\ &\quad 1000000\ 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000 \\ &\quad 1000000\ 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000 \\ &\quad 1000000\ 0111111\ 0111111\ 0111111\ 0111111\ 1000000\ 1000000\ 1000000 \\ &\quad 1000000\ 0111111\ 0111111\ 0111111\ 1000000 \end{aligned}$$

The cost of a classical "square and multiply" algorithm to carry on this power h would be, at least, 259 squaring and 145 multiplications! The slowest operation is the multiplication in the field L (squarings are faster). Our best implementation of multiplication in L requires about 10 000 CPU cycles (which takes at least 2.4 ms on a 10 MHz smart card).

We need to find an addition chain for h involving as little multiplications as possible. We did not limit ourselves to finding a "classical" addition chain for h, but also privileged some special powers, namely 128 and 128^7 which, as we will see in the next part, are quite easy to compute. After numerous attempts we have found the following method:

- $\alpha \longleftarrow (B^2)^2$;
- $\beta \longleftarrow B \times \alpha$;
- $\gamma \longleftarrow (\alpha^2)^2$;
- $\delta \longleftarrow \beta \times \gamma$;
- $u \longleftarrow \delta^2 \times \delta$;
- $v \longleftarrow (\gamma^2)^2$;
- $t \longleftarrow ((v^{128})^{128})^{128} \times u$;
- $w \longleftarrow ((t^{128} \times t)^{128} \times t)^{128}$;
- $x \longleftarrow ((w \times u)^{128})^{128^7}$;
- $z \longleftarrow v^{128^7} \times w \times v \times x$;
- $A \longleftarrow (((z^{128^7})^{128^7})^{128} \times x)^{128^7} \times z$.

This method has a particularly low number of multiplications: 12, instead of 145 for "square and multiply".

5.1 Special Operations Involved in the Computation of f^{-1}:

We already described the implementation of the multiplication and the squaring. What remain, are the efficient implementations of $x \mapsto x^{128}$ and $x \mapsto x^{128^7}$. In $K = \mathbb{F}128$, those two operations are K-linear on L, so they can be seen fulfilled with a matrix multiplication. Moreover, due to the fact that the polynomial by which we reduced $(X^{37} + X^{12} + X^{10} + X^2 + 1)$ has only coefficients 0 and 1, the matrices involved are only made up of 0s and 1s, which allows us to store their coefficients on single bits.

The Function $\mathbf{x \mapsto x^{128^7}}$: As the matrix used for the raising to the power 128^7 is fixed, we can also accelerate this computation by finding a "cheap" road in the matrix, which is related to the idea of the Gray code. The Gray code is an ordering of all binary n-bits words, such two consecutive words differ by only one bit, see [9] for example. This allows to compute all possible linear combinations over $GF(2)$ of some n vectors, with one XOR by combination, instead of

$n/2$. We use an even better, specific solution that is adapted to the particular matrix that is fixed (even for two different private keys). Our goal is to compute:

$$y = M.x \ ; \quad \text{with } x,y \in K^{37} \text{ and } M \in \mathbf{M}_{37,37}(\mathbb{F}2)$$

For this we divide the matrix in some $37 \times n$ submatrices for some small n. For each submatrix we look for a "cheap" road: each y_i is a XOR of different x_i, how to compute them minimizing the number of XORs as possible. For example we assume that the first submatrix begins with:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \dots \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ . & . & . & . & . & . & . \end{pmatrix} \cdot \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{pmatrix}^T$$

We store x_1,...,x_n in separate registers. Let A be the main register. We first put $A \leftarrow x_5$. Then we XOR A with x_3: $A \leftarrow A \oplus x_3$ and then put $y_1 \leftarrow A$, now $y_3 = x_3 \oplus x_5$. Then we put $y_2 \leftarrow A \leftarrow A \oplus x_4$. Finally we do $y_4 \leftarrow A \leftarrow A \oplus x_5$, etc.. A "cheap" road should use about 1 XOR per y_i computed. We need to find the cheapest road in each submatrix and also to find the best n for such an operation. For our specific matrices 37×37 $n = 7$ seemed to be optimal and our best solution has been found with some computer simulations.
This technique allows to accelerate quite a lot (about 40 times !) the operation $\mathbf{x} \mapsto \mathbf{x^{128^7}}$, however it takes a lot of space in term of program code (0.7 Kbyte).

The Function $\mathbf{x \mapsto x^{128}}$: It is useless to use the above technique to compute the power $128 = 2^7$. Doing 7 successive squarings is faster than a matrix multiplication, even if done in a clever way. It is also much cheaper in term of code size, not only we have no matrix to store, but we will mostly re-use the existing code. In addition to that, $\mathbf{x \mapsto x^{128}}$ can be computed faster than doing seven squarings in a row. Indeed, seven squares will be a succession of squares in $GF(128)$ done position by position on 37 values, and a multivariate linear operation over $GF(128)^{37}$ that comprises expansion and modular reduction modulo the irreducible polynomial. It is easy to see that "position by position" squaring commutes with the multivariate linear part. Thus we may postpone all the 7 "position by position" squarings to the end, and then we realize that they are not needed, because in $GF(128)$ we have always $x^{128} = x$.

6 The Performance Data

To summarize, our implementation of Sflash requires:

- 2 matrix products to apply T_m and S_m.
- 12 multiplications in L.
- 7 squarings in L.
- 8 raisings to the power 128 in L.
- 5 raisings to the power 128^7.

From here we have two possible implementations of Sflash:

- A **fast** one, using the technique above to compute the power 128^7 which is quite fast but a bit large in term of code size:
  - RAM: 334 bytes (112 bytes of *data* and 222 bytes of *xdata*).
  - Code size (ROM): 3.1 Kbytes.

- A **slower** one, using a classical matrix product which is slower (but still acceptable) and have a shorter code:
  - RAM: 334 bytes (112 bytes of *data* and 222 bytes of *xdata*).
  - Code size (ROM): 2.5 Kbytes.

We have implemented the two versions of Sflash on two Intel 8051 CPU based components: an original Intel 8051 CPU and an Infineon SLE66 component without cryptoprocessor. The timings (without hashing) are the following:

Component	8051	8051	SLE66	SLE66	8051	8051	SLE66	SLE66
Frequency [MHz]	3.57	10	3.57	10	3.57	10	3.57	10
Code version	fast	fast	fast	fast	slow	slow	slow	slow
ROM size [kbytes]	3.1	3.1	3.1	3.1	2.5	2.5	2.5	2.5
Timings [ms]	750	268	164	**59**	1075	384	230	82

We see that on a smart card without any coprocessor and in usual conditions (10 MHz is today a normal frequency for a low-cost component such as SLE66) one can compute digital signatures in as little as 59 ms! It is much less than for RSA or Elliptic Curves, even if a cryptographic co-processor is used (!), see the comparison in Section 7.8.

7 Protecting Sflash Against Side-Channel Attacks

In the side-channel attacks, the adversary tries to recover the private key of a signature scheme (or other useful information) given the information that leaks from the intermediate data that appears during the computation, or from the computation itself, see for example [11].

7.1 Protecting Against SPA-like Attacks

For Sflash, the computation is always the same, whatever the value of the private key is. Moreover, all the computations in Sflash use full bytes of the private key, making the dependency of power consumption on the key very complex to exploit. This prevents SPA attacks known for unprotected implementations of RSA, in which the power consumption allows to see the values of single bits of the private key.

7.2 Protecting Against DPA-like Attacks

For DPA-like attacks, the protection boils down to masking the intermediate data. The signature of Sflash involves mainly the composition of 3 functions, t^{-1}, f^{-1} and s^{-1}. The best way to prevent an information leak (of any type) is to mask completely the two intermediate values: the output of t^{-1} and of f^{-1}. For this, we will use the homomorphic properties of the functions t^{-1} and f^{-1} with regard to respectively the addition and the multiplication. Thus, since t^{-1} is affine, its output is still masked additively (with another mask). Similarly we may pass through f^{-1} with a multiplicative masking.

The proposed method for a secured implementation of Sflash is shown on Figure (1).

7.3 Algorithmic Considerations

A. Computation of $f(\lambda)$: The function f is the following:

$$f : x \mapsto x^{128^{11}+1} = \left((((x^{128^7})^{128})^{128})^{128})^{128}\right) \cdot x$$

$x \longleftarrow$ hashed message to sign.
$r \longleftarrow$ random $\in L$.
$\lambda \longleftarrow$ random $\in L^*$.

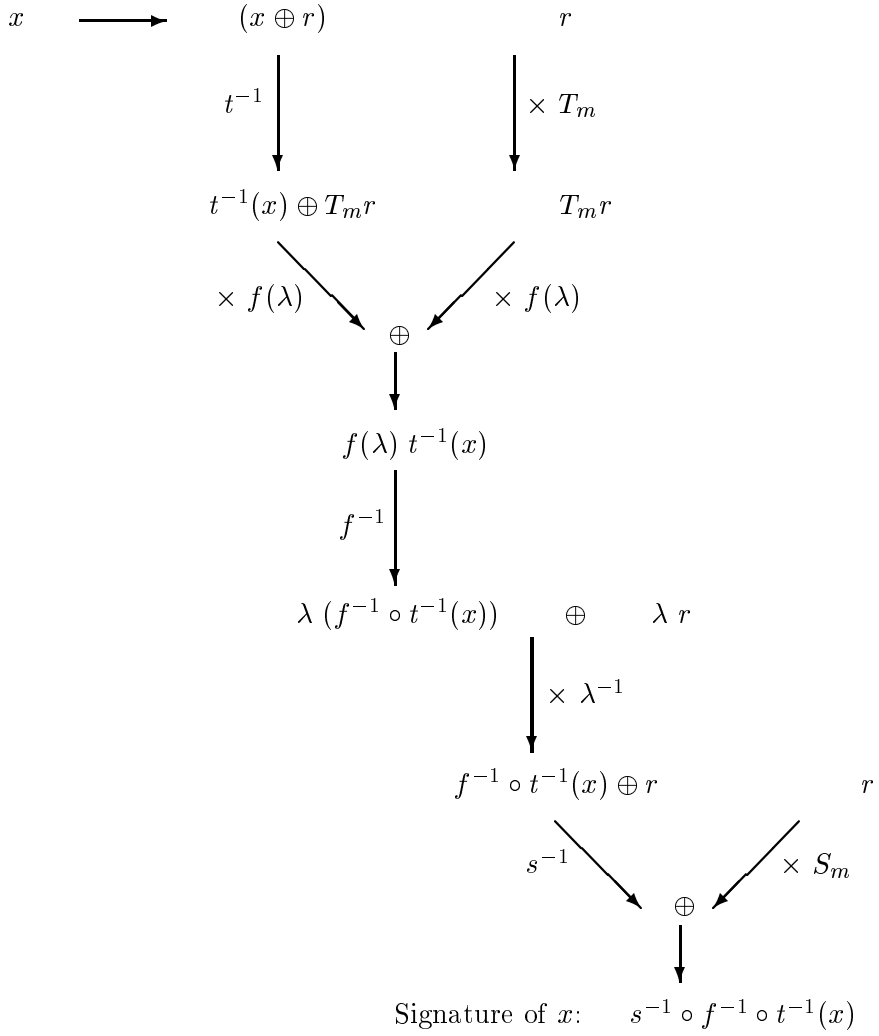


FIG. 1 – *A randomized masking method to protect Sflash against side-channel attacks.*

so we can compute $f(\lambda)$ with one raising to the power 128^7, 4 raisings to the power 128 and one multiplication.

B. Computation of λ^{-1}: First of all we begin with remarking that $|L^*| = 128^{37} - 1$, so inverting an element of L can be done by raising it to the power $128^{37} - 2$. Besides $128^{37} - 2 = (11...110)_2$ (*i.e.* 258 "1" followed by "0" in basis 2). Thus we can compute λ^{-1} as follows:

- $z \longleftarrow \lambda$
- $z \longleftarrow z^2.z$ (i.e. $z = \lambda^{(11)_2}$ that we store)
- $z \longleftarrow z^{2^2}.z$ $(z = \lambda^{(1111)_2})$
- $z \longleftarrow z^{2^4}.z$ $(z = \lambda^{(11111111)_2})$
- $z \longleftarrow (z^{128})^2.z$ $(z = \lambda^{(1111111111111111)_2})$
- $z \longleftarrow ((z^{128})^{128})^4.z$ $(z = \lambda^{(11111111111111111111111111111111)_2})$
- $z \longleftarrow ((((z^{128})^{128})^{128})^{128})^{2^4}.z$ $(z = \lambda^{(11...11)_2}$ with 64 "1")
- $z \longleftarrow z^{2^{64}}.z = (((z^{128^7})^{128})^{128})^2.z$ $(z = \lambda^{(11...11)_2}$ with 128 "1")
- $z \longleftarrow z^{2^{128}}.z = ((((z^{128^7})^{128^7})^{128})^{128})^{128})^{128})^4.z$
  (now we have $z = \lambda^{(11...11)_2}$ with 256 "1")
- $z \longleftarrow z^{2^2}.\lambda^{(11)_2}$ $(z = \lambda^{(11...11)_2}$ with 258 "1")
- $z \longleftarrow z^2$ which gives indeed $z = \lambda^{-1} = \lambda^{(11...110)_2}$ with 258 "1" and one "0".

C. Computation of $T_m r$ et $S_m r$: We compute them with a classical matrix product.

Computations Added Compared to an Unprotected Version:

- 2 matrix products to compute $T_m r$ and $S_m r$.
- 13 multiplications in L.
- 20 squarings in L.
- 17 raisings to the power 128 in L.
- 4 raisings to the power 128^7 in L.

Comparing to what we achieved with our (unprotected) implementation of Sflash, see the results in Section 7.6, the version which is protected against DPA-like attacks implies to approximately double the running time.

8 Digital Signatures on a Smart Card - a Comparison

In this section we compare Sflash to some other known implementations of the signature schemes in smart cards. The numerical data for schemes other than Sflash are based on unverified claims of the vendors.

cryptosystem	Sflash	NTRU-251	RSA-1024	RSA-1024	ECC-191
platform	SLE-66	Philips 8051	SLE-66	ST-19XL	SLE-66
word size [bits]	8	8	8	8	8
ROM size [Kbytes]	3.1	5			
speed [MHz]	10	16	10	10	10
co-processor	no	no	no	yes	yes
Signature Length	259	1757	1024	1024	382
Timings	59 ms	160 ms	many s	111 ms	180 ms

Apparently, the only signature scheme known that might be able to compete with Sflash is NTRUSign. An NTRUSign signature seems to be slower, but also about 6 times longer. Knowing

that the communication ports of low-end smart cards are quite slow, at 9600 bit/s, an NTRUSign card would take additional 200 ms to transmit the signature.

9 Conclusion

In this paper we described a highly optimized implementation of the Sflash signature schemes on a low-cost smart card. Our fastest implementation of Sflash takes 59 ms on a 8051 based CPU at 10MHz. We also presented a method to protect this implementation against DPA-like attacks that requires about twice as much time.

Though the security of Sflash is not as well understood as for example for RSA, Sflash is apparently the fastest signature scheme known. It is suitable to implement PKI on low-cost smart card, token or palm devices. It allows also to propose secure low-cost payment/banking solutions.

References

[1] Nicolas Courtois, *La sécurité des primitives cryptographiques basées sur les problèmes algébriques multivariables MQ, IP, MinRank, et HFE*, PhD Thesis, Paris 6 University, 2001, in French. Available at `http://www.minrank.org/phd.pdf`

[2] Nicolas Courtois, Magnus Daum, Patrick Felke, *On the Security of HFE, HFEv- and Quartz*, PKC'2003, to appear in LNCS, Springer.

[3] Magnus Daum, Patrick Felke, *Some new aspects concerning the Analysis of HFE type Cryptosystems*, Presented at Yet Another Conference on Cryptography (YACC'02), June 3-7, 2002, Porquerolles Island, France.

[4] Magnus Daum, *Das Kryptosystem HFE und quadratische Gleichungssysteme über endlichen Körpern*, Diplomarbeit, Universität Dortmund, 2001. Available at `daum@itsc.ruhr-uni-bochum.de`

[5] Jean-Charles Faugère, *Report on a successful attack of HFE Challenge 1 with Gröbner bases algorithm F5/2*, announcement that appeared in `sci.crypt` newsgroup on the internet on April 19th 2002.

[6] Henri Gilbert, Marine Minier, *Cryptanalysis of Sflash*, EUROCRYPT'2002, LNCS 2332, Springer, pp. 288-298.

[7] Michael Garey, David Johnson, *Computers and Intractability, a guide to the theory of NP-completeness*, Freeman, p. 251.

[8] Willi Geiselmann, Rainer Steinwandt, Thomas Beth, *Revealing 441 Key Bits of SFLASH-v2*, Third NESSIE Workshop, November 6-7, 2002, Munich, Germany.

[9] A page about the Gray code, `http://www.nist.gov/dads/HTML/graycode.html`

[10] Neal Koblitz, *Algebraic aspects of cryptography*, Springer, ACM3, 1998, Chapter 4: "Hidden Monomial Cryptosystems", pp. 80-102.

[11] Paul Kocher, Joshua Jaffe, Benjamin Jun, *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Available at `http://www.cryptography.com/dpa/technic/index.html`

[12] Tsutomu Matsumoto, Hideki Imai, *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption*, EUROCRYPT'88, LNCS 330, Springer 1998, pp. 419-453.

[13] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, CRYPTO'95, LNCS 963, Springer, pp. 248-261.

[14] Jacques Patarin, Nicolas Courtois , Louis Goubin, *C*-+ and HM - Variations around two schemes of T. Matsumoto and H. Imai*, ASIACRYPT'98, LNCS 1514, Springer, pp. 35-49.

[15] Jacques Patarin, Louis Goubin, Nicolas Courtois, *Quartz,* **1**28*-bit long digital signatures*, Cryptographers' Track RSA Conference 2001, San Francisco 8-12 April 2001, LNCS 2020, Springer, pp. 282-297.
Note: The Quartz signature scheme has been updated since, see [16].

[16] Jacques Patarin, Louis Goubin, Nicolas Courtois, *Quartz,* **1**28*-bit long digital signatures*, An updated version of Quartz specification. available at `http://www.cryptosystem.net/quartz/` or `http://www.cryptonessie.org`

[17] Jacques Patarin, Louis Goubin, Nicolas Courtois, *Flash, a fast multivariate signature algorithm*, Cryptographers' Track RSA Conference 2001, San Francisco 8-12 April 2001, LNCS 2020, Springer, pp. 298-307.

[18] An updated version of Sflash specification. Available at `http://www.cryptosystem.net/sflash/` or `http://www.cryptonessie.org`

[19] Adi Shamir, *Efficient signature schemes based on birational permutations*, CRYPTO'93, LNCS 773, Springer, pp. 1-12.

Attaques physiques et contre-mesures

Dans les cinq articles qui suivent, on étudie la sécurité des algorithmes cryptographiques face aux attaques physiques, et particulièrement celles qui s'appuient sur l'analyse différentielle de consommation électrique. Des contre-mesures génériques sont proposées, notamment pour les algorithmes utilisant plusieurs structures algébriques différentes. Un traitement général des attaques par analyse différentielle d'ordre supérieur est aussi proposé.

Page 347 – CHES'99

DES and Differential Power Analysis.

Louis Goubin et Jacques Patarin

Cet article analyse les attaques par analyse différentielle de consommation électrique. Une condition nécessaire et suffisante pour que la DPA puisse s'appliquer est mise en évidence. À partir de là, une contre-mesure générique, la méthode de duplication, est décrite, ainsi que son application concrète dans le cas du DES et de RSA.

Page 361 – CHES'2000

On Boolean and Arithmetic Masking against Differential Power Analysis.

Jean-Sébastien Coron et Louis Goubin

Pour obtenir des implémentations sûres contre les attaques DPA, Messerges a proposé une stratégie générale consistant à protéger les opérations élémentaires au moyen de masques aléatoires. En pratique, pour mener cette idée à bien, deux masques sont souvent nécessaires, d'où la nécessité d'une méthode de conversion de l'un à l'autre. Nous montrons ici que la méthode de conversion proposée par Messerges peut être mise en défaut par une attaque DPA adéquate.

Page 367 – CHES'2001

A Sound Method for Switching between Boolean and Arithmetic Masking.

Louis Goubin

Dans cet article, une méthode de conversion entre les masquages booléens et arithmétiques est décrite, ainsi que la preuve formelle qu'elle est immunisée contre les attaques DPA. Ce résultat permet de mettre en œuvre concrètement une méthode de protection générale pour les algorithmes où coexistent opérations booléennes et arithmétiques. Une conséquence potentielle est la mise au point d'une architecture anti-DPA pour les microprocesseurs.

Page 379 – PKC'2003

A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems.

Louis Goubin

Cet article présente une analyse des contre-mesures anti-DPA existantes pour les algorithmes à base de courbes elliptiques. Une nouvelle attaque par analyse de

consommation est proposée, montrant que les méthodes classiques de coordonnées projectives aléatoires, ou de randomisation du corps ou de la courbe, ne sont pas suffisantes en elles-mêmes, et doivent être renforcées par d'autres techniques.

Page 391 – FSE'2003

A Generic Protection against High-Order Differential Power Analysis.
Mehdi-Laurent Akkar et Louis Goubin

La notion d'attaque par analyse différentielle d'ordre supérieur est une généralisation naturelle de la DPA. Dans cet article, une analyse de cette extension est proposée. On voit ainsi que les contre-mesures classiques ne permettent pas d'éviter les attaques HO-DPA, mais une nouvelle contre-mesure générique est décrite pour y remédier.

DES and Differential Power Analysis

The "Duplication" Method

CHES'99
Article avec Jacques Patarin (Bull SC&T)

Abstract

Paul Kocher recently developped attacks based on the electric consumption of chips that perform cryptographic computations. Among those attacks, the "Differential Power Analysis" (DPA) is probably one of the most impressive and most difficult to avoid.

In this paper, we present several ideas to resist this type of attack, and in particular we develop one of them which leads, interestingly, to rather precise mathematical analysis. Thus we show that it is possible to build an implementation that is provably DPA-resistant, in a "local" and restricted way (*i.e.* when – given a chip with a fixed key – the attacker only tries to detect predictable local deviations in the differentials of mean curves). We also briefly discuss some more general attacks, that are sometimes efficient whereas the "original" DPA fails. Many measures of consumption have been done on real chips to test the ideas presented in this paper, and some of the obtained curves are printed here.

1 Introduction

This paper is about a way of securing a cryptographic algorithm that makes use of a secret key. More precisely, the goal consists in building an implementation of the algorithm that is not vulnerable to a certain type of physical attacks – so-called "Differential Power Analysis".

These DPA attacks belong to a general family of attacks that look for information about the secret key by studying the electric consumption of the electronic device during the execution of the computation. In this family, we usually distinguish between SPA attacks ("Simple Power Analysis") and DPA attacks.

In SPA attacks, the aim is essentially to guess – from the values of the consumption – which particular instruction is being computed at a certain time and with which input or output, and then to use this information to deduce some part of the secret. Figure 1 shows the electric consumption of a chip, measured during a DES computation on a real smartcard. The fact that the 16 rounds of the DES algorithm are clearly visible is a good sign that power analysis attacks may indeed provide information about what the chip is doing.

In DPA attacks, some differentials on two sets of average consumption are computed, and the attacks succeed if an unusual phenomenon appears – on these differentials of consumption – for a good choice of some of the key bits (we give details below), so that we are able to find out those key bits. What makes DPA attacks so impressive, when they work, is the fact

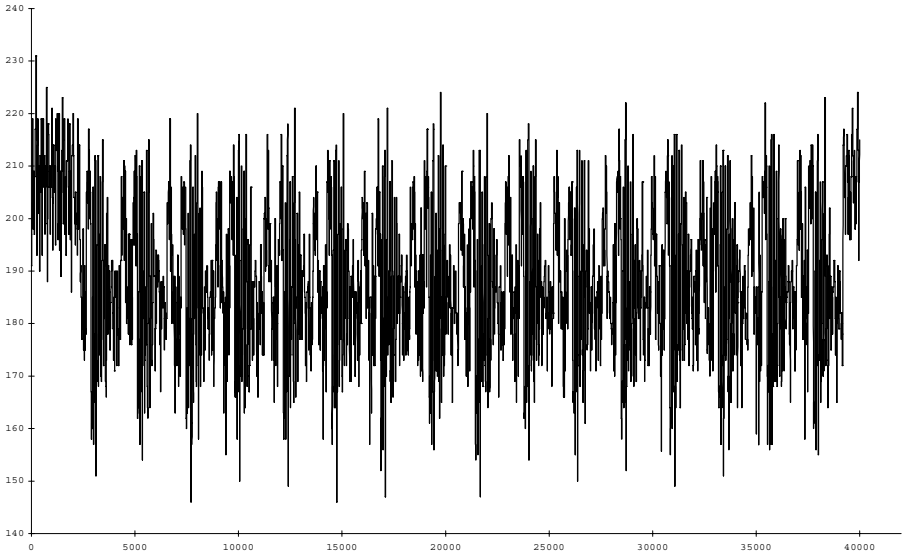


FIG. 1 – *Electric consumption measured on the 16 rounds of a DES computation*

that they can find out the secret key of a public algorithm (for example DES, but also many other algorithms) without knowing anything (nor trying to find anything) about the particular implementation of that algorithm. Implementations exist that are DPA-resistant (differentials do not show anything special) but not SPA-resistant (some critical information can be deduced from the consumption curves). On the contrary, other implementations exist that are SPA-resistant but not DPA-resistant (some critical information can be found by studying differentials of two mean curves of consumption). Finally, some implementations can be found that resist both types of attack (at least at the present), or none of them.

Throughout this paper, we study more particularly DPA and we will not deal any longer with SPA. Indeed, as we see below, DPA can easily be analyzed in a mathematical way (and not only in an empirical way). There exist many attacks based on the electric consumption. We do not claim to give here solutions to all the problems that may result from these attacks.

The cryptographic algorithms we consider here make use of a secret key in order to compute an output information from an input information. It may be a ciphering, a deciphering or a signature operation. In particular, all the material described in this paper applies to "secret key algorithms" and also to the so-called "public key algorithms".

2 The "Differential Power Analysis" attacks

The "Differential Power Analysis" attacks, developped by Paul Kocher and Cryptographic Research (see [1]), start from the fact that the attacker can get many more information (than the knowledge of the inputs and the outputs) during the execution of the computation, such as for instance the electric consumption of the microcontroller or the electromagnetic radiations of the circuit. The "Differential Power Analysis" (DPA to be brief) is an attack that allows to obtain information about the secret key (contained in a smartcard for example), by performing a statistical analysis of the electric consumption records measured for a large number of computations with the same key. Let us consider for instance the case of the DES algorithm (Data Encryption

Standard). It executes in 16 steps, called "rounds". In each of these steps, a transformation F is performed on 32 bits. This F function uses eight non-linear transformations from 6 bits to 4 bits, each of which is coded by a table called "S-box". The DPA attack on the DES can be performed as follows (the number 1000 used below is just an example):

Step 1: We measure the consumption on the first round, for 1000 DES computations. We denote by E_1, ..., E_{1000} the input values of those 1000 computations. We denote by C_1, ..., C_{1000} the 1000 electric consumption curves measured during the computations. We also compute the "mean curve" MC of those 1000 consumption curves.

Step 2: We focus for instance on the first output bit of the first S-box during the first round. Let b be the value of that bit. It is easy to see that b depends on only 6 bits of the secret key. The attacker makes an hypothesis on the involved 6 bits. He computes – from those 6 bits and from the E_i – the expected (theoretical) values for b. This enables to separate the 1000 inputs E_1, ..., E_{1000} into two categories: those giving $b = 0$ and those giving $b = 1$.

Step 3: We now compute the mean MC' of the curves corresponding to inputs of the first category (*i.e.* the one for which $b = 0$). If MC and MC' show an appreciable difference (in a statistical meaning, *i.e.* a difference much greater than the standard deviation of the measured noise), we consider that the chosen values for the 6 key bits were correct. If MC and MC' do not show any sensible difference, we repeat step 2 with another choice for the 6 bits.

Note: In practice, for each choice of the 6 key bits, we draw the curve representing the difference between MC and MC'. As a result, we obtain 64 curves, among which one is supposed to be very special, *i.e.* to show an appreciable difference, compared to all the others.

Step 4: We repeat steps 2 and 3 with a "target" bit b in the second S-box, then in the third S-box, ..., until the eighth S-box. As a result, we finally obtain 48 bits of the secret key.

Step 5: The remaining 8 bits can be found by exhaustive search.

Note: It is also possible to focus (in steps 2, 3 and 4) on the set of the four output bits for the considered S-boxes, instead of only one output bit. This is what we actually did for real smartcards. In that case, the inputs are separated into 16 categories: those giving 0000 as output, those giving 0001, ..., those giving 1111. In step 3, we may compute for example the mean MC' of the curves corresponding to the last category (*i.e.* the one which gives 1111 as output). As a result, the mean MC' is computed on approximately $\frac{1}{16}$ of the curves (instead of approximately half of the curves with step 3 above): this may compel us to use a number of DES computations greater than 1000, but it generally leads to a more appreciable difference between MC and MC'.

We presented in figures 2 and 3 two mean curves, resulting from steps 2 and 3, for a classical implementation of DES on a real smartcard (with '1111' as target output of the first S-box and with 2048 different inputs, even if we noted that 512 inputs are sufficient). A detailed analysis of the 64 obtained curves (that we cannot all print here, due to the lack of place) shows that the one corresponding to the correct choice of the 6 key-bits can easily be detected (it contains much greater peaks than all the others).

This attack does not require any knowledge about the individual electric consumption of each instruction, nor about the position in time of each of these instructions. It applies exactly the same way as soon as the attacker knows the outputs of the algorithm and the corresponding consumption curves. It only relies on the following fundamental hypothesis:

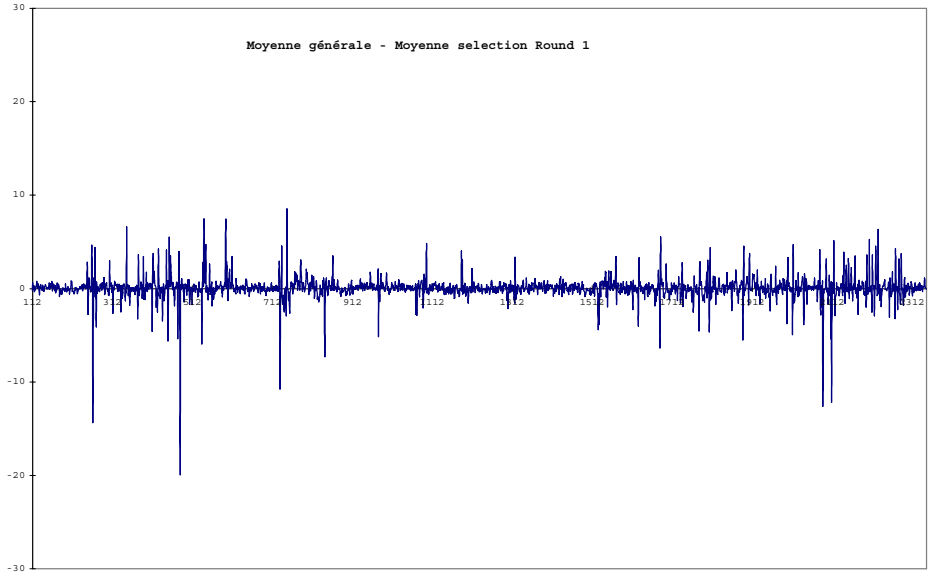


FIG. 2 – *An example of difference of the curves MC and MC' when the 6 bits are false*

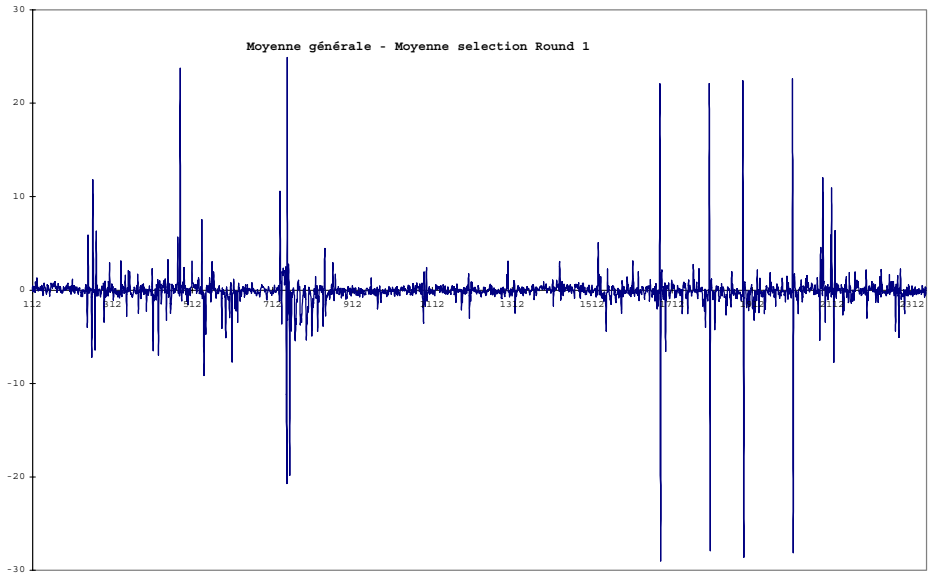


FIG. 3 – *Difference of the curves MC and MC' when the 6 bits are correct*

Fundamental hypothesis: *There exists an intermediate variable, that appears during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for this variable.*

All the algorithms that use S-boxes, such as DES, are potentially vulnerable to the DPA attack, because the "natural" implementations generally remain within the hypothesis mentioned above.

3 Securing the algorithm

Several countermeasures against DPA attacks can be conceived. For instance:

1. Introducing random timing shifts, so that the computed means do not correspond any longer to the consumption of the same instruction. The crucial point consists here in performing those shifts so that they cannot be easily eliminated by a statistical treatment of the consumption curves.
2. Replacing some of the critical instructions (in particular the basic assembler instructions involving writings in the carry, readings of data from an array, etc) by assembler instructions whose "consumption signature" is difficult to analyze.
3. For a given algorithm, giving an explicit way of computing it, so that DPA is provably unefficient on the obtained implementation. For instance, for a DES-like algorithm, we detail in section 4 how to compute the non-linear transformations of the S-boxes in order to avoid some DPA attacks.

In the present paper, we essentially study the third idea because it leads to a quite precise mathematical analysis. We give in this section a general method to implement an algorithm with a secret key so as to avoid the DPA attacks described above. The basic principle consists in programming the algorithm so that the fundamental hypothesis above is not true any longer (*i.e.* an intermediate variable never depends on the knowledge of an easily accessible subset of the secret key).

The main idea

In this paper, we mainly study how this can be done by using the following main idea: replacing each intermediate variable V, occuring during the computation and depending on the inputs (or the outputs), by k variables V_1, ..., V_k, such that V_1, V_2, ..., V_k allows us – if we want – to retrieve V. More precisely, to guarantee the security of the algorithm in its new form, it is sufficient to choose a function f satisfying the identity $V = f(V_1,...,V_k)$, together with the two following conditions:

Condition 1: *From the knowledge of a value v and for any fixed value i, $1 \leq i \leq k$, it is not feasible to deduce information about the set of the values v_i such that there exist a $(k-1)$-uple $(v_1,...,v_{i-1},v_{i+1},...,v_k)$ satisfying the equation $f(v_1,...,v_k) = v$.*

Condition 2: *The function f is such that the transformations to be performed on V_1, V_2, ..., or V_k during the computation (instead of the transformations usually performed on V) can be implemented without calculating V.*

First example for condition 1: If we choose $f(v_1,...,v_k) = v_1 \oplus v_2 \oplus ... \oplus v_k$, where $\oplus$ denotes the bit-by-bit "exclusive-or" function, condition 1 is obviously satisfied, because – for any fixed index i between 1 and k – the considered set of the values v_i contains all the possible values and thus does not depend on v.

Second example for condition 1: If we consider some variable V whose values lie in the multiplicative group of $\mathbf{Z}/n\mathbf{Z}$, we can choose the function $f(v_1,...,v_k) = v_1 \cdot v_2 \cdot ... \cdot v_k \bmod n$, where the new variables v_1, v_2, ..., v_k also have values in the multiplicative group of $\mathbf{Z}/n\mathbf{Z}$. Condition 1 is also obviously true because – for any fixed index i between 1 and k – the considered set of the values v_i contains all the possible values and thus does not depend on v.

We then "translate" the algorithm by replacing each intermediate variable V depending on the inputs (or the outputs) by the k variables V_1, ..., V_k. In the following sections, we study how conditions 1 and 2 can be achieved in the case of the DES or RSA algorithms.

4 The DES algorithm: first example of implementation for DPA resistance

In this section, we consider the particular case of the DES algorithm. We choose here to separate each intermediate variable V, occuring during the computation and depending on the inputs (or the outputs), into two variables V_1 and V_2 (*i.e.* we take $k = 2$). Let us choose the function $f(v_1,v_2) = v = v_1 \oplus v_2$ (see the first example of section 3), which satisfies condition 1. From the construction of the algorithm, it is easy to see that the transformations performed on v always belong to one of the five following categories:

1. permutation of the bits of v;
2. expansion of the bits of v;
3. "exclusive-or" between v and another variable v' of the same type;
4. "exclusive-or" between v and a variable depending only on the key;
5. transformation of v using a S-box.

The first two cases correspond to linear transformations on the bits of the variable v. For these ones, condition 2 is thus very easy to satisfy: we just have – instead of the transformation usually performed on v – to perform the permutation or the expansion on v_1, then on v_2, and the identity $f(v_1,v_2) = v$, which was true before the transformation, is also true afterwards.

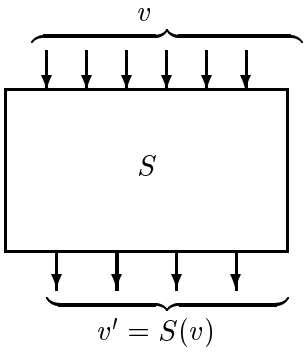
In the same way, in the third case, we just have to replace the computation of $v'' = v \oplus v'$ by the computation of $v_1'' = v_1 \oplus v_1'$ and $v_2'' = v_2 \oplus v_2'$. The identities $f(v_1,v_2) = v$ and $f(v_1',v_2') = v'$ give indeed $f(v_1'',v_2'') = v''$, so that condition 2 is true again.

As concerns the exclusive-or between v and a variable c depending only on the key, condition 2 is also very easy to satisfy: we just have to replace the computation of $v \oplus c$ by $v_1 \oplus c$ (or $v_2 \oplus c$) and that gives condition 2.

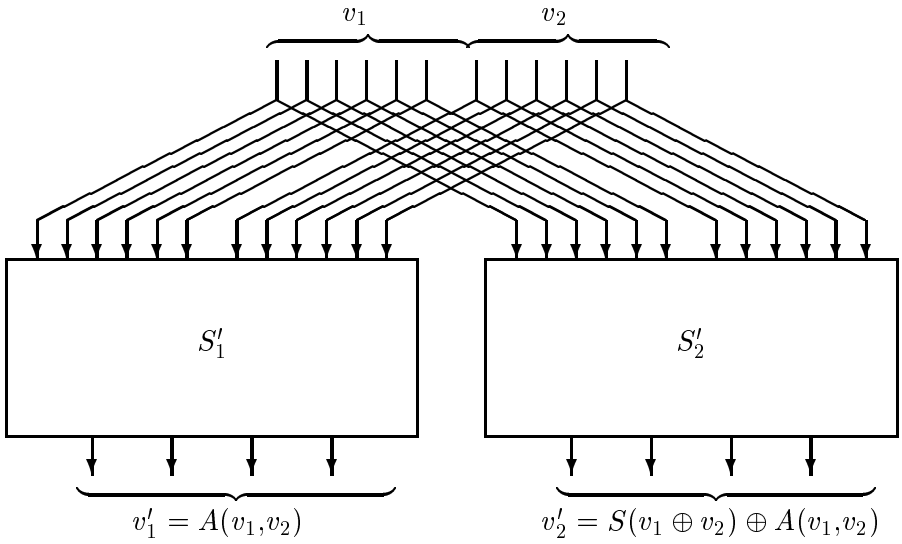
Finally, instead of the non-linear transformation $v' = S(v)$, given under the form of a S-box (which in that example has 6-bits inputs and 4-bits outputs), we implement the transformation $(v_1',v_2') = S'(v_1,v_2)$ by using two new S-boxes (each of which sending 12 bits onto 4 bits). In order to keep the identity $f(v_1',v_2') = v'$, we may choose:

$$(v_1',v_2') = S'(v_1,v_2) = (A(v_1,v_2), S(v_1 \oplus v_2) \oplus A(v_1,v_2)).$$

where A denotes a *randomly* chosen *secret* transformation from 12 bits to 4 bits (see figure 4). The first of the new S-boxes corresponds to the table of the transformation $(v_1,v_2) \mapsto A(v_1,v_2)$, and the second one corresponds to the table of the transformation $(v_1,v_2) \mapsto S(v_1 \oplus v_2) \oplus A(v_1,v_2)$. Thanks to the randomly chosen function A, condition 1 is satisfied. Moreover, the use of tables allows us to avoid the computation of $v_1 \oplus v_2$, so that condition 2 is also true.



Initial implementation: the predictable values v and v' appear in RAM at some time



Modified implementation: the values $v = v_1 \oplus v_2$ and $v' = v'_1 \oplus v'_2$ never explicitely appear in RAM

FIG. 4 – *Standard transformation of a S-box*

The solution presented in this section is quite realistic for chips that compute DES in hardware (and are not embedded in a card), or for PCs, because – in those cases – enough memory is available. More precisely, the size of the memory required to store the S-boxes is 32 Kbytes for the method described in this section. It is too much for smartcards, for which specific variations

using less memories are described in section 5 below.

5 Smartcard implementations of DES

First variation

In order to reduce the ROM used by the algorithm, it is quite possible to use the same random function A for the eight S-boxes (of the initial description of the DES), so that we have only nine (new) S-boxes (*i.e.* 18 Kbytes) to store in ROM, instead of sixteen S-boxes.

Second variation

In order to reduce the size of the ROM needed to store the S-boxes, we can also use the following method: instead of each non-linear transformation $v' = S(v)$ of the initial implementation, given under the form of a S-box (with 6-bits inputs and 4-bits outputs in the case of the DES), we implement the transformation $(v'_1,v'_2) = S'(v_1,v_2)$ by using two S-boxes, each of which sending 6 bits onto 4 bits. The initial implementation of the computation $v' = S(v)$ is replaced by the two following successive computations:

- $v_0 = \varphi(v_1 \oplus v_2)$
- $(v'_1,v'_2) = S'(v_1,v_2) = (A(v_0),S(\varphi^{-1}(v_0)) \oplus A(v_0))$

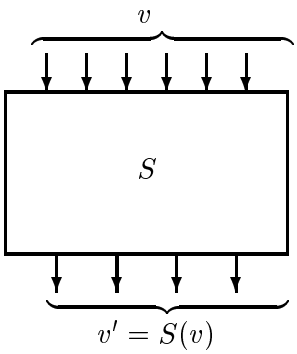
where φ is a *bijective* and *secret* function from 6 bits to 6 bits and where A denotes a *random* and *secret* transformation from 6 bits to 4 bits. The first of the two new S-boxes corresponds to the table of the transformation $v_0 \mapsto A(v_0)$ and the second one corresponds to the table of the transformation $v_0 \mapsto S(\varphi^{-1}(v_0)) \oplus A(v_0)$. From this construction, the identity $f(v'_1,v'_2) = v'$ is always true. Thanks to the random function A, condition 1 is satisfied. Moreover, the use of tables allows us to avoid the computation of $\varphi^{-1}(v_0) = v_1 \oplus v_2$, so that condition 2 is also true. This solution (shown in figure 5) requires 512 bytes to store the S-boxes.

In order to satisfy condition 2, it remains to choose the bijective transformation φ such that the computation of $v_0 = \varphi(v_1 \oplus v_2)$ is feasible without computing $v_1 \oplus v_2$. We give below two examples of possible choice for the function φ.

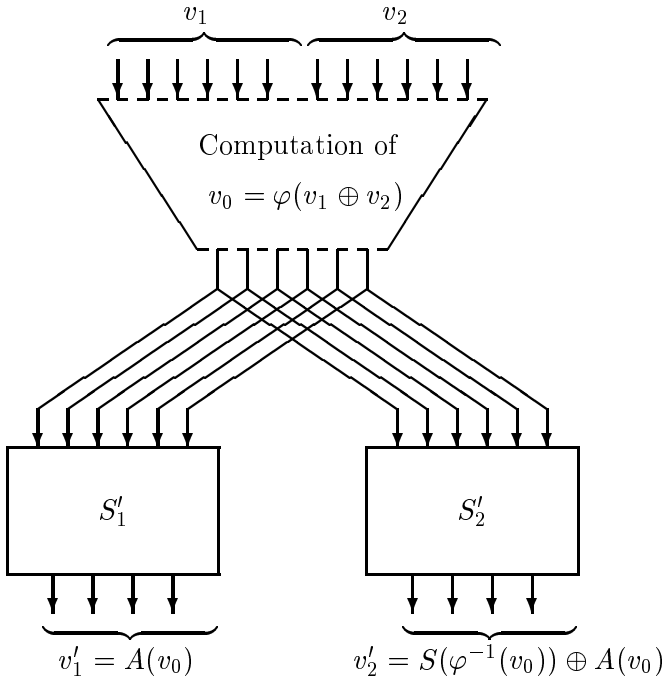
Exemple 1: a linear bijection

We choose φ as a *linear secret* and *bijective* function from 6 bits to 6 bits (we consider the set of the 6-bits values as a vectorial space of dimension 6 on the finite field $\mathbf{F}_2$ with 2 elements). In practice, choosing φ is equivalent to choosing a *random* and *invertible* 6×6 *matrix* whose coefficients are 0 or 1. With this choice of φ, it is easy to see that condition 2 is satisfied. Indeed, to compute $\varphi(v_1 \oplus v_2)$, we just have to compute $\varphi(v_1)$, then $\varphi(v_2)$ and finally to compute the "exclusive-or" of the two obtained results.

For instance, the matrix $\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$ is invertible. It corresponds to the linear bijection φ from 6 bits to 6 bits defined by $\varphi(u_1,u_2,u_3,u_4,u_5,u_6) = (u_1 \oplus u_2 \oplus u_4, u_1 \oplus u_2 \oplus u_4 \oplus u_6, u_2 \oplus u_3 \oplus u_5, u_1 \oplus u_2 \oplus u_3 \oplus u_5, u_2 \oplus u_3 \oplus u_4 \oplus u_5, u_3 \oplus u_4 \oplus u_6)$.



Initial implementation: the predictable values v and v' appear in RAM at some time



Modified implementation: the values $v = v_1 \oplus v_2$ and $v' = v'_1 \oplus v'_2$ never explicitely appear in RAM

FIG. 5 – *Transformation of a S-box (second variation)*

Let $v_1 = (v_{1,1},v_{1,2},v_{1,3},v_{1,4},v_{1,5},v_{1,6})$ and $v_2 = (v_{2,1},v_{2,2},v_{2,3},v_{2,4},v_{2,5}, v_{2,6})$. To compute $\varphi(v_1 \oplus v_2)$, we successively compute:

- $\varphi(v_1) = (v_{1,1} \oplus v_{1,2} \oplus v_{1,4}, v_{1,1} \oplus v_{1,2} \oplus v_{1,4} \oplus v_{1,6}, v_{1,2} \oplus v_{1,3} \oplus v_{1,5},\ v_{1,1} \oplus v_{1,2} \oplus v_{1,3} \oplus v_{1,5}, v_{1,2} \oplus v_{1,3} \oplus v_{1,4} \oplus v_{1,5}, v_{1,3} \oplus v_{1,4} \oplus v_{1,6})$
- $\varphi(v_2) = (v_{2,1} \oplus v_{2,2} \oplus v_{2,4}, v_{2,1} \oplus v_{2,2} \oplus v_{2,4} \oplus v_{2,6}, v_{2,2} \oplus v_{2,3} \oplus v_{2,5}, v_{2,1} \oplus v_{2,2} \oplus v_{2,3} \oplus v_{2,5}, v_{2,2} \oplus v_{2,3} \oplus v_{2,4} \oplus v_{2,5}, v_{2,3} \oplus v_{2,4} \oplus v_{2,6})$

Then we compute the "exclusive-or" of the two obtained results.

Exemple 2: a quadratic bijection

We choose φ as a *quadratic secret* and *bijective* function from 6 bits to 6 bits. Here, "quadratic" means that each bit of the output is given by a polynomial function of *total degree two* of the 6 bits of the input (which are identified to 6 elements of the finite field $\mathbf{F}_2$). In practice, we may choose the function φ defined by $\varphi(x) = t(s(x)^5)$, where s is a secret linear bijection from $(\mathbf{F}_2)^6$ to $\mathcal{L}$, t is a secret linear bijection from $\mathcal{L}$ to $(\mathbf{F}_2)^6$ and $\mathcal{L}$ denotes an algebraic extension of degree 6 over the finite field $\mathbf{F}_2$. The bijectivity of this function φ follows from the fact that $a \mapsto a^5$ is a bijection on the extension $\mathcal{L}$ (whose inverse is $b \mapsto b^{38}$). To be convinced that condition 2 is still satisfied, just notice that we can write:

$$\varphi(v_1 \oplus v_2) = \psi(v_1,v_1) \oplus \psi(v_1,v_2) \oplus \psi(v_2,v_1) \oplus \psi(v_2,v_2),$$

where the function ψ is defined by $\psi(x,y) = t(s(x)^4 \cdot s(y))$.

For instance, if we identify $\mathcal{L}$ to $\mathbf{F}_2[X]/(X^6+X+1)$ and if we choose s and t whose matrices are

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

with respect to the basis $(1,X,X^2,X^3,X^4,X^5)$ of $\mathcal{L}$ over $\mathbf{F}_2$ and to the canonical basis of $(\mathbf{F}_2)^6$ over $\mathbf{F}_2$, we obtain the following quadratic bijection φ from 6 bits to 6 bits:

$$\varphi(u_1,u_2,u_3,u_4,u_5,u_6) = (u_2u_5 \oplus u_1u_4 \oplus u_4 \oplus u_6 \oplus u_6u_2 \oplus u_4u_6 \oplus u_2 \oplus u_5 \oplus u_3 \oplus u_4u_3, u_2u_5 \oplus u_5u_1 \oplus u_1u_4 \oplus u_4 \oplus u_6 \oplus u_4u_5 \oplus u_2 \oplus u_3 \oplus u_3u_1, u_2u_5 \oplus u_5u_1 \oplus u_6u_5 \oplus u_1u_4 \oplus u_3u_5 \oplus u_1 \oplus u_4u_6 \oplus u_6u_3 \oplus u_4u_3 \oplus u_3u_1, u_1u_4 \oplus u_2u_3 \oplus u_6u_1 \oplus u_4u_6 \oplus u_5 \oplus u_6u_3 \oplus u_4u_3, u_5u_1 \oplus u_1u_4 \oplus u_6 \oplus u_3u_5 \oplus u_4u_5 \oplus u_1 \oplus u_6u_1 \oplus u_4u_6 \oplus u_3 \oplus u_6u_3 \oplus u_4u_2, u_4 \oplus u_6 \oplus u_3u_5 \oplus u_1 \oplus u_4u_6 \oplus u_6u_3).$$

To compute $\varphi(v_1 \oplus v_2)$, we use the function $\psi(x,y) = t(s(x)^4 \cdot s(y))$ from 12 bits to 6 bits, which gives the 6 output bits from the 12 input bits as follows:

$$\psi(x_1,x_2,x_3,x_4,x_5,x_6,y_1,y_2,y_3,y_4,y_5,y_6) = (x_3y_5 \oplus x_6y_2 \oplus x_6y_3 \oplus x_6y_4 \oplus x_3y_1 \oplus x_6y_1 \oplus x_1y_3 \oplus x_1y_5 \oplus x_5y_2 \oplus x_5y_5 \oplus x_5y_1 \oplus x_6y_6 \oplus x_1y_6 \oplus x_1y_2 \oplus x_1y_4 \oplus x_2y_1 \oplus x_2y_2 \oplus x_4y_4 \oplus x_3y_3 \oplus x_3y_6 \oplus x_4y_3 \oplus x_5y_3, x_4y_5 \oplus x_3y_1 \oplus x_6y_1 \oplus x_2y_5 \oplus x_5y_1 \oplus x_6y_6 \oplus x_1y_6 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2 \oplus x_4y_1 \oplus x_4y_4 \oplus x_3y_3, x_6y_2 \oplus x_6y_3 \oplus x_6y_4 \oplus x_6y_5 \oplus x_3y_1 \oplus x_6y_1 \oplus x_2y_5 \oplus x_5y_1 \oplus x_1y_6 \oplus x_1y_1 \oplus x_1y_2 \oplus x_1y_4 \oplus x_2y_1 \oplus x_2y_4 \oplus x_4y_2 \oplus x_2y_6 \oplus x_3y_4 \oplus x_5y_3, x_3y_1 \oplus x_6y_2 \oplus x_2y_6 \oplus x_5y_3 \oplus x_5y_4 \oplus x_5y_6 \oplus x_6y_3 \oplus x_2y_3 \oplus x_4y_6 \oplus x_6y_5 \oplus x_1y_3 \oplus x_5y_5 \oplus x_2y_4 \oplus x_4y_2 \oplus x_4y_5 \oplus x_3y_5 \oplus x_4y_3 \oplus x_6y_1 \oplus x_4y_1, x_3y_1 \oplus x_6y_6 \oplus x_5y_3 \oplus x_5y_6 \oplus x_5y_2 \oplus x_1y_5 \oplus x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_3 \oplus x_3y_6 \oplus x_6y_5 \oplus x_1y_3 \oplus x_2y_4 \oplus x_3y_3 \oplus x_4y_5 \oplus x_2y_5 \oplus x_6y_1 \oplus x_4y_1 \oplus x_6y_4 \oplus x_3y_2, x_6y_6 \oplus x_4y_4 \oplus x_5y_4 \oplus x_5y_6 \oplus x_6y_3 \oplus x_1y_6 \oplus x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_6y_5 \oplus x_2y_4 \oplus x_4y_2 \oplus x_4y_5 \oplus x_3y_5 \oplus x_6y_1 \oplus x_6y_4).$$

By using these formulas, we successively compute $\psi(v_1,v_1)$, $\psi(v_1,v_2)$, $\psi(v_2,v_1)$ and $\psi(v_2,v_2)$. Finally, we compute the "exclusive-or" of the four obtained results.

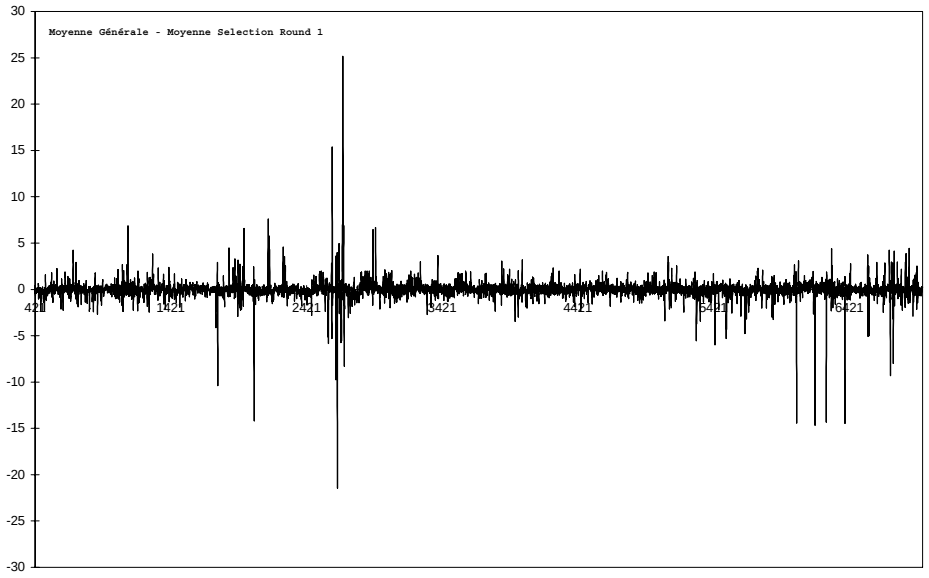


FIG. 6 – *An example of difference of the curves MC and MC′ when the 6 bits are false*

Third variation

To further reduce the size of the ROM needed to store the S-boxes, we can apply simultaneously the ideas of both variations 1 and 2: we use the second variation, with the same secret bijection φ (from 6 bits to 6 bits) and the same secret random function A (from 6 bits to 6 bits) in the new implementation of each non-linear transformation given by a S-box. This variation thus requires only 288 bytes to store the S-boxes. We have applied the Differential Power Analysis on real smartcard implementations of this third variation. Two examples of differential mean curves (with 2048 inputs and with '1111' as target output of the first S-box) are presented in figures 6 and 7. A precise analysis of the 64 curves given by the DPA (see note after step 3, in section 2) shows that none of them appears to be "very special", compared to the others, so that we can say that this implementation resists the DPA attack (at least in its basic form, see appendix 2 for a possible generalization that could still be dangerous).

Fourth variation

In this last variation, instead of implementing the transformation $(v'_1,v'_2) = S'(v_1,v_2)$ (which replaces the non-linear transformation $v' = S(v)$ of the initial implementation, given by a S-box) by using two S-boxes, we perform the computation of v'_1 (respectively v'_2) by using a simple algebraic function (*i.e.* the bits of v'_1 (respectively v'_2) are given by a polynomial function of total degree 1 or 2 of the bits of v_1 and v_2), then we compute v'_2 (respectively v'_1) by using a table. This enables to reduce again the needed ROM for the implementation. This last variation requires only 256 bytes to store the S-boxes.

6 The RSA algorithm

The "Power Analysis" attacks also threaten the classical implementations of the RSA algorithm. Indeed, these implementations often use the so-called "*square-and-multiply*" principle

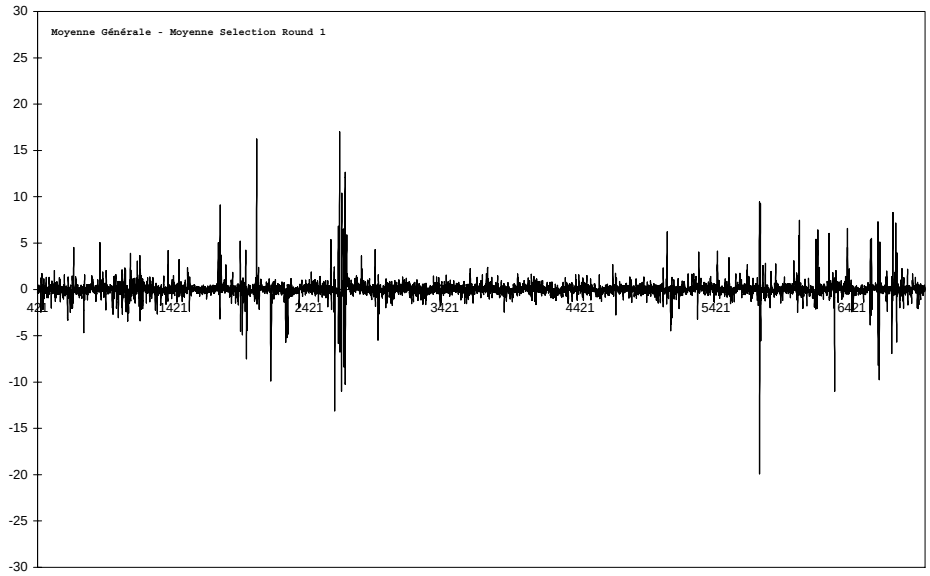


FIG. 7 – *Difference of the curves MC and MC' when the 6 bits are correct*

to perform the computation of $x^d \bmod n$. It consists in writing the binary decomposition $d = d_{m-1}2^{m-1} + d_{m-2}2^{m-2} + ... + d_1 2^1 + d_0 2^0$ of the secret exponent d, and then in performing the computation as follows:

1. $z \leftarrow 1$;
   For i going backwards from $m-1$ to 0 do:
2. $z \leftarrow z^2 \bmod n$;
3. if $d_i = 1$ then $z \leftarrow z \times x \bmod n$.

In this computation, we see that – among the successive values taken by the z variable – the first ones depend on only a few bits of the secret key d. The fundamental hypothesis that enables the DPA attack is thus satisfied. As a result, we can guess for instance the 10 most significant bits of d by studying the consumption measures on the part of the algorithm corresponding to i going from $m-1$ to $m-10$. We can then continue the attack by using comsumption measures on the part of the algorithm corresponding to i going from $m-11$ to $m-20$, which gives the 10 next bits of d, and so on. We finally find all the bits of the secret exponent d.

The method described in section 3 also applies to securing the RSA algorithm. We use here a separation of each intermediate variable V (whose values lie in the multiplicative group of $\mathbf{Z}/n\mathbf{Z}$), occuring during the computation and depending on the inputs (or the outputs), into two variables V_1 and V_2 (*i.e.* we take $k = 2$), and we choose the function $f(v_1,v_2) = v = v_1 \cdot v_2 \bmod n$. We already saw in section 3 (cf "second example") that this function f satisfies condition 1.

We thus replace x by (x_1,x_2) such that $x = x_1 \cdot x_2 \bmod n$ and z by (z_1,z_2) such that $z = z_1 \cdot z_2 \bmod n$ (in practice, we can for instance choose x_1 randomly and deduce x_2). Considering again the three steps of the "square-and-multiply" method, we perform the following transformations:

1. $z \leftarrow 1$ is replaced by $z_1 \leftarrow 1$ and $z_2 \leftarrow 1$;
2. $z \leftarrow z^2 \bmod n$ is replaced by $z_1 \leftarrow z_1^2 \bmod n$ and $z_2 \leftarrow z_2^2 \bmod n$;
3. $z \leftarrow z \times x \bmod n$ is replaced by $z_1 \leftarrow z_1 \times x_1 \bmod n$ and $z_2 \leftarrow z_2 \times x_2 \bmod n$.

It is easy to check that the identity $z = f(z_1,z_2)$ remains true all along the computation, which shows that condition 2 is satisfied.

Let us notice that the computations performed respectively on the z_1 variable and on the z_2 variable are completely independent. We thus can imagine to perform the two computations either in a sequential way, or in an overlapped way, or simultaneously in the case of multiprogrammation, or simultaneously in different processors working concurrently.

7 Generalized Attacks

Recently, more general attacks were introduced, where the attacker tries to correlate different points of a power consumption curve. We have no place here to analyze in detail the effect of this idea on the "Duplication Method". However, it is possible to show that if each variable is splitted in, say, k variables, then the complexity of the implementation increases in $\mathcal{O}(k)$, while the complexity of the attack increases exponentially in k.

As concerns DES implementations, we also recommend, when it is possible, to use different S-Boxes for each smartcard (stored in EEPROM). In particular, this avoids some attacks which use a smartcard with a known key to help finding the key in another smartcard whose key is unknown.

8 Conclusion

In this paper, we investigate how the study of the electric consumption measures of an electronic device can be used by an attacker to get information about the secret key of the cryptographic algorithm computed by the chip. More precisely, we focus on the so-called Differential Power Attacks, which were recently introduced by Paul Kocher, and which use a statistical analysis of a set of consumption curves measured for many different inputs of the cryptographic algorithm.

We study more precisely how DPA attacks work, and what precise hypotheses they rely on. We then present several ways of securing cryptosystems. In particular, concrete examples of such countermeasures are described in the cases of DES and RSA, which are the most used cryptographic algorithms at the present.

To secure those algorithms, we essentially study the main idea that consists in splitting each intermediate variable, occuring in the computation, into two (or more) variables, such that the values of these new variables cannot be easily predicted. The obtained implementations can be proved to resist the "local" version of Differential Power Analysis (where the attacker only tries to detect local deviations in the differentials of mean curves). Nevertheless other attacks can be conceived, still using the analysis of electric consumption. We do not pretend to solve all security problems linked to these threats. These latter attacks are not only theoretical, since we found real products that are defeated by them, but it also shows that theoretical investigations have to be continued in that sensitive subject.

References

[1] Paul Kocher, Joshua Jaffe, Benjamin Jun, *Introduction to Differential Power Analysis and Related Attacks*, 1998. This paper is available at http://www.cryptography.com/dpa/technical/index.html

On Boolean and Arithmetic Masking against Differential Power Analysis

CHES'2000
Article avec Jean-Sébastien Coron (Gemplus)

Abstract

Since the announcement of the Differential Power Analysis (DPA) by Paul Kocher and al., several countermeasures were proposed in order to protect software implementations of cryptographic algorithms. In an attempt to reduce the resulting memory and execution time overhead, Thomas Messerges recently proposed a general method that "masks" all the intermediate data.

This masking strategy is possible if all the fundamental operations used in a given algorithm can be rewritten with masked input data, giving masked output data. This is easily seen to be the case in classical algorithms such as DES or RSA.

However, for algorithms that combine Boolean and arithmetic functions, such as IDEA or several of the AES candidates, two different kinds of masking have to be used. There is thus a need for a method to convert back and forth between Boolean masking and arithmetic masking.

In the present paper, we show that the 'BooleanToArithmetic' algorithm proposed by T. Messerges is not sufficient to prevent Differential Power Analysis. In a similar way, the 'ArithmeticToBoolean' algorithm is not secure either.

Key words: Physical attacks, Differential Power Analysis, Electric consumption, AES, IDEA, Smartcards, Masking Techniques.

1 Introduction

Paul Kocher and al. introduced in 1998 ([10]) and published in 1999 ([11]) the concept of *Different Power Analysis* attack, also known as DPA. It belongs to a general family of attacks that look for information about the secret key of a cryptographic algorithm, by studying the electric consumption of the electronic device during the execution of the computation.

The initial focus was on symmetrical cryptosystems such as DES (see [10, 14]) and the AES candidates (see [1, 3, 6]), but public-key cryptosystems have since been shown to be also vulnerable to the DPA attacks (see [15, 5, 9]).

Therefore, the research for countermeasures has considerably increased. In [6], Daemen and Rijmen proposed several countermeasures, including the insertion of dummy code, power consumption randomization and balancing of data. But these methods were proven to be unsufficient:

in [4], Chari and al. suggested that signal processing can be used by clever attackers to remove dummy code or to cancel the effects of randomization and data balancing. They propose a better approach, consisting in splitting all the intermediate variables. A similar "duplication" method was proposed as a particular case by Goubin and al. in [9]

However, these general methods generally increase dramatically the amount of memory needed, or the computation time, as was pointed by Chari and al. in [3]. Moreover, it has been shown in [8] that even inner rounds can be aimed by "Power-Analysis"-type attacks, so that the splitting should be performed on all rounds of the algorithm. This makes the issue of the memory and time computation overhead even more crucial, especially for embedded systems such as smart cards.

In [13], Thomas Messerges investigated on DPA attacks applied on the AES candidates. He developped a general countermeasure, consisting in masking all the inputs and outputs of each elementary operations used by the microprocessor. This generic technique allowed him to evaluate the impact of these countermeasures on the five AES algorithms.

This masking strategy is possible if all the fundamental operations used in a given algorithm can be rewritten with masked input data, giving masked output data. This is easily seen to be the case for the DES algorithm, because a single masking (using the XOR operation) can be used throughout the computation of the 16 rounds. For RSA, a masking using the multiplication operation in the multiplicative group modulo n is also sufficient.

However, for algorithms that combine Boolean and arithmetic functions, two different kinds of masking have to be used. There is thus a need for a method to convert back and forth between Boolean masking and arithmetic masking. This is typically the case for IDEA [12] and for three AES candidates: MARS [2], RC6 [16] and TWOFISH [17].

Thomas Messerges proposed in [13] an algorithm in order to perform this conversion between a "$\oplus$ mask" and a "$+$ mask". Unfortunately, we show in the present paper that the 'Boolean-ToArithmetic' algorithm proposed by T. Messerges is not sufficient to prevent Differential Power Analysis. In a similar way, the 'ArithmeticToBoolean' algorithm is not secure either. A detailed attack is described.

2 The "Differential Power Analysis" Attack

The "Differential Power Analysis" attack, developped by Paul Kocher and Cryptographic Research (see [10, 11], see also [7]), starts from the fact that the attacker can get much more information (than the knowledge of the inputs and the outputs) during the execution of the computation, such as for instance the electric consumption of the microcontroller or the electromagnetic radiations of the circuit.

The "Differential Power Analysis" (DPA) is an attack that allows to obtain information about the secret key (contained in a smartcard for example), by performing a statistical analysis of the electric consumption records measured for a large number of computations with the same key.

Let us consider for instance the case of the DES algorithm (Data Encryption Standard). It executes in 16 steps, called "rounds". In each of these steps, a transformation F is performed on 32 bits. This F function uses eight non-linear transformations from 6 bits to 4 bits, each of which is coded by a table called "S-box".

The DPA attack on the DES can be performed as follows (the number 1000 used below is just an example):

Step 1: We measure the consumption on the first round, for 1000 DES computations. We denote by E_1, ..., E_{1000} the input values of those 1000 computations. We denote by C_1, ..., C_{1000} the

1000 electric consumption curves measured during the computations. We also compute the "mean curve" MC of those 1000 consumption curves.

Step 2: We focus for instance on the first output bit of the first S-box during the first round. Let b be the value of that bit. It is easy to see that b depends on only 6 bits of the secret key. The attacker makes an hypothesis on the involved 6 bits. He computes – from those 6 bits and from the E_i – the expected (theoretical) values for b. This enables to separate the 1000 inputs E_1, ..., E_{1000} into two categories: those giving $b = 0$ and those giving $b = 1$.

Step 3: We now compute the mean MC' of the curves corresponding to inputs of the first category (*i.e.* the one for which $b = 0$). If MC and MC' show an appreciable difference (in a statistical meaning, *i.e.* a difference much greater than the standard deviation of the measured noise), we consider that the chosen values for the 6 key bits were correct. If MC and MC' do not show any sensible difference, we repeat step 2 with another choice for the 6 bits.

Note: In practice, for each choice of the 6 key bits, we draw the curve representing the difference between MC and MC'. As a result, we obtain 64 curves, among which one is supposed to be very special, *i.e.* to show an appreciable difference, compared to all the others.

Step 4: We repeat steps 2 and 3 with a "target" bit b in the second S-box, then in the third S-box, ..., until the eighth S-box. As a result, we finally obtain 48 bits of the secret key.

Step 5: The remaining 8 bits can be found by exhaustive search.

Note: It is also possible to focus (in steps 2, 3 and 4) on the set of the four output bits for the considered S-boxes, instead of only one output bit. This is what we actually did for real smartcards. In that case, the inputs are separated into 16 categories: those giving 0000 as output, those giving 0001, ..., those giving 1111. In step 3, we may compute for example the mean MC' of the curves corresponding to the last category (*i.e.* the one which gives 1111 as output). As a result, the mean MC' is computed on approximately $\frac{1}{16}$ of the curves (instead of approximately half of the curves with step 3 above): this may compel us to use a number of DES computations greater than 1000, but it generally leads to a more appreciable difference between MC and MC'.

This attack does not require any knowledge about the individual electric consumption of each instruction, nor about the position in time of each of these instructions. It applies exactly the same way as soon as the attacker knows the outputs of the algorithm and the corresponding consumption curves. It only relies on the following fundamental hypothesis:

Fundamental hypothesis*: There exists an intermediate variable, that appears during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for this variable.*

3 Review of Countermeasures

Several countermeasures against DPA attacks can be conceived. For instance:

1. Introducing random timing shifts, so that the computed means do not correspond any longer to the consumption of the same instruction. The crucial point consists here in performing those shifts so that they cannot be easily eliminated by a statistical treatment of the consumption curves.

2. Replacing some of the critical instructions (in particular the basic assembler instructions involving writings in the carry, readings of data from an array, etc) by assembler instructions whose "consumption signature" is difficult to analyze.
3. For a given algorithm, giving an explicit way of computing it, so that DPA is provably unefficient on the obtained implementation. The masking strategy, detailed below is an example of this third kind of method.

4 The Masking Method

In the present paper, we focus on the "masking method", initially suggested by Chari and al. in [3], and studied further in [4].

The basic principle consists in programming the algorithm so that the fundamental hypothesis above is not true any longer (*i.e.* an intermediate variable never depends on the knowledge of an easily accessible subset of the secret key). In a concrete way, using a secret sharing scheme, each intermediate that appears in the cryptographic algorithm is splitted. Therefore, an attacker has to analyze multiple point distributions, which makes his task grow exponentially in the number of elements in the splitting.

In [13], Messerges applied this fundamental idea for all the elementary operations that can occur in the AES algorithms. For algorithms that combine Boolean and arithmetic functions, such as MARS, RC6 and TWOFISH, two different kinds of masking have to be used :

Boolean masking : $x' = x \oplus r$
Arithmetic masking : $x' = (x - r) \bmod 2^k$

Here the variable x is masked with random r to give the masked value x'.

The conversion from boolean masking to arithmetic masking as described in [13] works as follows :

BooleanToArithmetic
Input : (x',r) such that $x = x' \oplus r$.
Output : (A,r) such that $x = A + r$
Randomly select : $C = 0$ or $C = -1$
$B = C \oplus r$; /* $B = r$ or $b = \bar{r}$ /*
$A = B \oplus x'$; /* $A = x$ or $A = \bar{x}$ /*
$A = A - B$; /* $A = x - r$ or $A = \bar{x} - \bar{r}$ /*
$A = A + C$; /* $A = x - r$ or $A = \bar{x} - \bar{r} - 1$ /*
$A = A \oplus C$; /* $A = x - r$ /*
Return(A,r);

The conversion from the arithmetic masking to the boolean masking can be done with a similar algorithm.

The conversion from one type of masking to another should be done in such a way that it is not vulnerable to DPA attacks. The previous algorithm takes as input the couple (x',r) such that $x = x' \oplus r$. The unmasked data is x and the masked data is x'. The algorithm works by unmasking x' using the XOR operation and then remasking it using the addition operation.

The issue is that the variable x or $\bar{x}$ is computed during the execution of the algorithm. It is stated in [13] that a DPA attack will not work against this algorithm because the attacker does not know whether x or $\bar{x}$ is processed. This is true for a DPA selecting one bit of x : since x and $\bar{x}$ are processed with equal probability, the processed bit is decorrelated from the key and the

single-bit DPA does not work. This is not the case if we perform a DPA with 2 selected bits, as shown in the next section.

5 A DPA attack against the conversion algorithm

The attack is based on the fact that if 2 bits of x are equal, the corresponding bits are also equal in $\bar{x}$. Consequently, we modify the DPA attack described in section 7.2. Instead of selecting the curves from the predicted value of a given bit of x, we consider 2 bits and divide the power samples into 2 groups : in the first group, the 2 bits are equal, and in the second group they are distinct. The classification is not affected by the processing of x and $\bar{x}$. Consequently, if the power consumption when 2 bits are equal differs from the power consumption when 2 bits are distinct, the 2-bits DPA works : the proper key hypothesis should show a peak, while the others will be mostly flat, so that the all the key bits will be recovered.

Consider the four conditional laws for the power consumption and denote their respective mean values μ_{00},μ_{01},μ_{10},μ_{11}. For the proper key hypothesis, the mean value of the first group is :

$$\frac{\mu_{00} + \mu_{11}}{2}$$

and the mean value of the second group is :

$$\frac{\mu_{01} + \mu_{10}}{2}$$

The mean of the difference between the two groups is thus :

$$D = \frac{\mu_{00} + \mu_{11} - \mu_{01} - \mu_{10}}{2} \tag{1}$$

Consequently, the 2-bits DPA works if $D \neq 0$.

We would like to stress that our attack is not a high-order DPA. A high-order DPA [11] consists in looking at joint probability distributions of multiple points in the power signal. As shown in [4], a high-order DPA attack requires a number of experiments exponential in the number of points considered. Instead, our attack concentrates on a single point in the power signal. Consequently, the number of required experiments should be of the same order as for a single-bit DPA.

6 Conclusion

We have described a DPA attack against the conversion algorithm proposed in [13]. Our attack is a straighforward extension of the classical DPA attack. We did not perform the experiments to validate our attack in practice but we think that the threat is real and such algorithm for converting from boolean masking to arithmetic masking should be avoided.

We are currently investigating an algorithm for converting from boolean masking to arithmetic masking and conversely, which would be secure against DPA. We hope that we will able to include it in the final version of the paper.

References

[1] Eli Biham and Adi Shamir, "Power Analysis of the Key Scheduling of the AES Candidates", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[2] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S.M. Matyas, L. O'Connor, M. Peyravian, D. Safford, and N. Zunic, "MARS - A Candidate Cipher for AES", NIST AES Proposal, Jun 98.

[3] Suresh Chari, Charantjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi, "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[4] Suresh Chari, Charantjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks", in *Proceedings of Advances in Cryptology – CRYPTO'99*, Springer-Verlag, 1999, pp. 398-412.

[5] Jean-Sébastien Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 292-302.

[6] John Daemen and Vincent Rijmen, "Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[7] John Daemen, Michael Peters and Gilles Van Assche, "Bitslice Ciphers and Power Analysis Attacks", in *Proceedings of Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.

[8] Paul N. Fahn and Peter K. Pearson, "IPA: A New Class of Power Attacks", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 173-186.

[9] Louis Goubin and Jacques Patarin, "DES and Differential Power Analysis – The Duplication Method", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 158-172.

[10] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Introduction to Differential Power Analysis and Related Attacks", http://www.cryptography.com/dpa/technical, 1998.

[11] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Differential Power Analysis", in *Proceedings of Advances in Cryptology – CRYPTO'99*, Springer-Verlag, 1999, pp. 388-397.

[12] X. Lai and J. Massey, "A Proposal for a New Block Encryption Standard", in *Advances in Cryptology - EUROCRYPT '90 Proceedings*, Springer-Verlag, 1991, pp. 389-404.

[13] Thomas S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks", in *Proceedings of Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.

[14] Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Investigations of Power Analysis Attacks on Smartcards", in *Proceedings of USENIX Workshop on Smartcard Technology*, May 1999, pp. 151-161.

[15] Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 144-157.

[16] R.L. Rivest, M.J.B. Robshaw, R. Sidney and Y.L. Yin, "The RC6 Block Cipher", v1.1, August 20, 1998.

[17] B. Schneier, J. Kemsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, "Twofish: A 128-Bit Block Cipher", AES submission available at: http://www.nist.gov/aes.

A Sound Method for Switching between Boolean and Arithmetic Masking

CHES'2001

Abstract

Since the announcement of the Differential Power Analysis (DPA) by Paul Kocher and al., several countermeasures were proposed in order to protect software implementations of cryptographic algorithms. In an attempt to reduce the resulting memory and execution time overhead, a general method was recently proposed, consisting in "masking" all the intermediate data.

This masking strategy is possible if all the fundamental operations used in a given algorithm can be rewritten with masked input data, giving masked output data. This is easily seen to be the case in classical algorithms such as DES or RSA.

However, for algorithms that combine boolean and arithmetic functions, such as IDEA or several of the AES candidates, two different kinds of masking have to be used. There is thus a need for a method to convert back and forth between boolean masking and arithmetic masking.

A first solution to this problem was proposed by Thomas Messerges in [15], but was unfortunately shown (see [6]) insufficient to prevent DPA. In the present paper, we present two new practical algorithms for the conversion, that are proven secure against DPA.

The first one ("BooleanToArithmetic") uses a constant number of elementary operations, namely 7, on the registers of the processor. The number of elementary operations for the second one ("ArithmeticToBoolean"), namely $5K + 5$, is proportional to the size K (in bits) of the processor registers.

Key words: Physical attacks, Differential Power Analysis, Electric consumption, AES, IDEA, Smartcards, Masking Techniques.

1 Introduction

Paul Kocher and al. introduced in 1998 ([12]) and published in 1999 ([13]) the concept of *Differential Power Analysis* attack, also known as DPA. The initial focus was on symmetrical cryptosystems such as DES (see [12, 16]) and the AES candidates (see [1, 3, 7]), but public key cryptosystems have since been shown to be also vulnerable to the DPA attacks (see [17, 5, 11]). In [10, 11], Goubin and Patarin proposed a generic countermeasure consisting in splitting all the intermediate variables. A similar "duplication" method was suggested shortly after by Chari and al. in [3] and [4]. Although the authors of [3] state that these general methods generally increase dramatically the amount of memory needed, or the computation time, Goubin and

Patarin proved that realistic implementations could be reached with the "duplication" method. However, it has been shown in [9] that even inner rounds can be aimed by "Power-Analysis"-type attacks, so that the splitting should be performed on all rounds of the algorithm. This makes the issue of the memory and time computation overhead even more crucial, especially for embedded systems such as smartcards.

In [15], Thomas Messerges investigated on DPA attacks applied on the AES candidates. He developped a general countermeasure, consisting in masking all the inputs and outputs of each elementary operation used by the microprocessor. This generic technique allowed him to evaluate the impact of these countermeasures on the five AES algorithms.
However, for algorithms that combine boolean and arithmetic functions, two different kinds of masking have to be used. There is thus a need for a method to convert back and forth between boolean masking and arithmetic masking. This is typically the case for IDEA [14] and for three AES candidates: MARS [2], RC6 [18] and Twofish [19].
T. Messerges proposed in [15] an algorithm in order to perform this conversion between a "$\oplus$ mask" and a "$+$ mask". Unfortunately, Coron and Goubin described in [6] a specific attack, showing that the "BooleanToArithmetic" algorithm proposed by T. Messerges is not sufficient to prevent Differential Power Analysis. In a similar way, his "ArithmeticToBoolean" algorithm is not secure either.

In the present paper, we present two new "BooleanToArithmetic" and "ArithmeticToBoolean" algorithms, proven secure against DPA attacks. Each of these algorithms uses only very simple operations: "XOR", "AND", subtractions and "logical shift left". Our "BooleanToArithmetic" algorithm uses a constant number (namely 7) of such elementary operations, whereas the number of elementary operations involved in our "ArithmeticToBoolean" algorithm is proportional (namely equal to $5K + 5$) to the size (*i.e.* the number K of bits) of the processor registers.

2 Background

2.1 The "Differential Power Analysis" Attack

The "Differential Power Analysis" (DPA) is an attack that allows to obtain information about the secret key (contained in a smartcard for example), by performing a statistical analysis of the electric consumption records measured for a large number of computations with the same key.

This attack does not require any knowledge about the individual electric consumption of each instruction, nor about the position in time of each of these instructions. It applies exactly the same way as soon as the attacker knows the outputs of the algorithm and the corresponding consumption curves. It only relies on the following fundamental hypothesis:

Fundamental hypothesis: *There exists an intermediate variable, that appears during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for this variable.*

2.2 The Masking Method

In the present paper, we focus on the "masking method", initially suggested by Goubin and Patarin in [10], and studied further in [11].
The basic principle consists in programming the algorithm so that the fundamental hypothesis above is not true any longer (*i.e.* an intermediate variable never depends on the knowledge of

an easily accessible subset of the secret key). More precisely, using a secret sharing scheme, each intermediate variable that appears in the cryptographic algorithm is splitted. Therefore, an attacker has to analyze multiple point distributions, which makes his task grow exponentially in the number of elements in the splitting.

2.3 The Conversion Problem

For algorithms that combine boolean and arithmetic functions, two different kinds of masking have to be used:

Boolean masking: $x' = x \oplus r$
Arithmetic masking: $A = x - r \bmod 2^K$

Here the variable x is masked with random r to give the masked value x' (or A). Our goal is to find an efficient algorithm for converting from boolean masking to arithmetic masking and conversely, in which all intermediate variables are decorrelated from the data to be masked, so that it is secure against DPA.

In all the present paper, we suppose that the processor has K-bit registers (in practice, K is most of the time equal to 8, 16, 32 or 64). All the arithmetic operations (such as the addition "+", the subtraction "−", or the doubling "$z \mapsto 2z$") are considered modulo 2^K. For simplicity, the "mod2^K" will often be omitted in the sequel.

3 From Boolean to Arithmetic Masking

3.1 A useful algebraic property

Let $I = \{0,1,2,\ldots,2^K - 1\}$, with $K \geq 1$ being an integer. Let $x' \in I$. We consider the function $\Phi_{x'} : I \to I$, defined by:

$$\Phi_{x'}(r) \equiv (x' \oplus r) - r \bmod 2^K.$$

We identify each element of I with the sequence of coefficients in its binary representation, so that I can be viewed as a vector space of dimension K over GF(2), isomorphic to GF(2)^K.

Theorem 13

$$\Phi_{x'}(r) = x' \oplus \bigoplus_{i=1}^{K-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \overline{x'}) \Big) \wedge (2^i x') \wedge (2^i r) \Big],$$

where $\overline{x'}$ stands for the ones complement of x', and $\wedge$ stands for the boolean "AND" operator.

See Appendix 1 for a proof of Theorem 1.

Corollary 13.1 *The function $\Phi_{x'}$ is affine over GF(2).*

This result is an easy consequence of Theorem 1.

3.2 The "BooleanToArithmetic" algorithm

Since $\Phi_{x'}$ is affine over GF(2), the function $\Psi_{x'} = \Phi_{x'} \oplus \Phi_{x'}(0)$ is *linear* over GF(2). Therefore, for any value γ,

$$\Psi_{x'}(r) = \Psi_{x'}(\gamma \oplus (r \oplus \gamma)) = \Psi_{x'}(\gamma) \oplus \Psi_{x'}(r \oplus \gamma).$$

Corollary 13.2 *For any value γ, if we denote $A = (x' \oplus r) - r$, we also have*

$$A = [(x' \oplus \gamma) - \gamma] \oplus x' \oplus [(x' \oplus (r \oplus \gamma)) - (r \oplus \gamma)].$$

$A = (x' \oplus r) - r$ can thus be obtained from the following algorithm:

Algorithme 8 BooleanToArithmetic

Input: (x',r) such that $x = x' \oplus r$

Output: (A,r) such that $x = A + r$

Initialize Γ to a random value γ
$T \Leftarrow x' \oplus \Gamma$
$T \Leftarrow T - \Gamma$
$T \Leftarrow T \oplus x'$
$\Gamma \Leftarrow \Gamma \oplus r$
$A \Leftarrow x' \oplus \Gamma$
$A \Leftarrow A - \Gamma$
$A \Leftarrow A \oplus T$

The "BooleanToArithmetic" algorithm uses 2 auxiliary variables (T and Γ), 1 random generation and 7 elementary operations (more precisely: 5 "XOR" and 2 subtractions).

3.3 Proof of security against DPA

From the description of the "BooleanToArithmetic" algorithm, we easily obtain the list of all the intermediate values V_0, ..., V_6 that appear during the computation of $A = (x' \oplus r) - r$:

$$\begin{cases} V_0 = \gamma \\ V_1 = \gamma \oplus r \\ V_2 = x' \oplus \gamma \\ V_3 = (x' \oplus \gamma) - \gamma \\ V_4 = [(x' \oplus \gamma) - \gamma] \oplus x' \\ V_5 = x' \oplus \gamma \oplus r \\ V_6 = (x' \oplus \gamma \oplus r) - (\gamma \oplus r) \end{cases}$$

If we suppose that γ is randomly chosen with a uniform distribution on $I = \{0,1\}^K$, it is easy to see that:

- the values V_0, V_1, V_2 and V_5 are uniformly distributed on I.
- the distributions of V_3, V_4 and V_6 depend on x' but not on r.

4 From Arithmetic to Boolean Masking

4.1 A useful recursion formula

Theorem 14 *If we denote $x' = (A+r) \oplus r$, we also have $x' = A \oplus u_{K-1}$, where u_{K-1} is obtained from the following recursion formula:*

$$\begin{cases} u_0 = 0 \\ \forall k \geq 0,\ u_{k+1} = 2[u_k \wedge (A \oplus r) \oplus (A \wedge r)]. \end{cases}$$

See Appendix 2 for a proof of Theorem 2.

4.2 The "ArithmeticToBoolean" algorithm

Let γ be any value. The change of variable $t_k = 2\gamma \oplus u_k$ leads to the following consequence of Theorem 2.

Corollary 14.1 *For any value γ, if we denote $x' = (A+r) \oplus r$, we also have $x' = A \oplus 2\gamma \oplus t_{K-1}$, where t_{K-1} is obtained from the following recursion formula:*

$$\begin{cases} t_0 = 2\gamma \\ \forall k \geq 0,\ t_{k+1} = 2[t_k \wedge (A \oplus r) \oplus \omega], \end{cases}$$

in which $\omega = \gamma \oplus (2\gamma) \wedge (A \oplus r) \oplus A \wedge r$.

As a consequence, $x' = (A + r) \oplus r$ can be obtained from the "ArithmeticToBoolean" algorithm below.
This method requires 3 auxiliary variables (T, Ω and Γ), 1 random generation and $(5K + 5)$ elementary operations (more precisely: $(2K + 4)$ "XOR", $(2K + 1)$ "AND" and K "logical shift left").

Algorithme 9 ArithmeticToBoolean

Input: (A,r) such that $x = A + r$
Output: (x',r) such that $x = x' \oplus r$
  Initialize Γ to a random value γ
  $T \Leftarrow 2\Gamma$
  $x' \Leftarrow \Gamma \oplus r$
  $\Omega \Leftarrow \Gamma \wedge x'$
  $x' \Leftarrow T \oplus A$
  $\Gamma \Leftarrow \Gamma \oplus x'$
  $\Gamma \Leftarrow \Gamma \wedge r$
  $\Omega \Leftarrow \Omega \oplus \Gamma$
  $\Gamma \Leftarrow T \wedge A$
  $\Omega \Leftarrow \Omega \oplus \Gamma$
  for $k = 1$ to $K - 1$ **do**
    $\Gamma \Leftarrow T \wedge r$
    $\Gamma \Leftarrow \Gamma \oplus \Omega$
    $T \Leftarrow T \wedge A$
    $\Gamma \Leftarrow \Gamma \oplus T$
    $T \Leftarrow 2\Gamma$
  end for
  $x' \Leftarrow x' \oplus T$

4.3 Proof of security against DPA

From the description of the "BooleanToArithmetic" algorithm, we easily obtain the list of all the intermediate values W_0, ..., W_{5K+4} that appear during the computation of $x' = (A + r) \oplus r$:

$$\begin{cases} W_0 = \gamma \\ W_1 = 2\gamma \\ W_2 = \gamma \oplus r \\ W_3 = \gamma \oplus \gamma \wedge r \\ W_4 = 2\gamma \oplus A \\ W_5 = \gamma \oplus 2\gamma \oplus A \\ W_6 = (\gamma \oplus 2\gamma \oplus A) \wedge r \\ W_7 = \gamma \oplus (2\gamma) \wedge r \oplus A \wedge r \\ W_8 = (2\gamma) \wedge A \\ W_9 = \gamma \oplus (2\gamma) \wedge (A \oplus r) \oplus A \wedge r = \omega \\ \text{for } k = 1 \text{ to } K-1 : \begin{cases} W_{5k+5} = (2\gamma \oplus u_{k-1}) \wedge r \\ W_{5k+6} = \gamma \oplus (2\gamma) \wedge A \oplus u_{k-1} \wedge r \oplus A \wedge r \\ W_{5k+7} = (2\gamma \oplus u_{k-1}) \wedge A \\ W_{5k+8} = \gamma \oplus u_{k-1} \wedge (A \oplus r) \oplus A \wedge r \\ W_{5k+9} = 2\gamma \oplus u_k \end{cases} \end{cases}$$

If we suppose that γ is randomly chosen with a uniform distribution on $I = \{0,1\}^K$, it is easy to see that:

- the values W_0, W_2 and W_{5k+8} $(1 \leq k \leq K-1)$ are uniformly distributed on I.
- the values W_1 and W_{5k+9} are uniformly distributed on the subset $\{0,1\}^{K-1} \times \{0\}$ of I.
- the distributions of W_3 and W_{5k+5} $(1 \leq k \leq K-1)$ depend on r but not on A.
- the distributions of W_4, W_8 and W_{5k+7} $(1 \leq k \leq K-1)$ depend on A but not on r.

To study the distribution of the remaining values (W_5, W_6, W_7, W_9 and W_{5k+6}), we will make use of the following result:

Theorem 15 *For any $\delta \in I$, the following function is bijective:*

$$\Theta_\delta : \begin{cases} I \to I \\ \gamma \mapsto \gamma \oplus (2\gamma) \wedge \delta. \end{cases}$$

See Appendix 3 for a proof of Theorem 3. As a result:

- the values $W_5 = \Theta_{-1}(\gamma) \oplus A$, $W_7 = \Theta_r(\gamma) \oplus A \wedge r$, $W_9 = \Theta_{A \oplus r}(\gamma) \oplus A \wedge r$ and $W_{5k+6} = \Theta_r(\gamma) \oplus u_{k-1} \wedge r \oplus A \wedge r$ $(1 \leq k \leq K-1)$ are uniformly distributed on I.
- the distribution of $W_6 = (\Theta_{-1}(\gamma) \oplus A) \wedge r$ depends on r but not on A.

5 Conclusion

In this paper, we solved the following open problem (stated in [6]): "find an efficient algorithm for converting from boolean masking to arithmetic masking and conversely, in which all intermediate variables are decorrelated from the data to be masked, so that it is secure against DPA".
The construction of our "BooleanToArithmetic" and "ArithmeticToBoolean" algorithms also led us to prove some results of independent interest. In particular we proved that $r \mapsto (a \oplus r) - r \bmod 2^K$ is an affine function, which seems to be a new result.
Finally, a direction for further research would be to find an improved version of the "ArithmeticToBoolean" algorithm, in which the number of elementary operations is less than $5K + 5$, or (even better) a constant independent of the size K of the registers.

Acknowledgement

I would like to thank Jean-Sébastien Coron for interesting discussions and suggestions.

References

[1] Eli Biham and Adi Shamir, "Power Analysis of the Key Scheduling of the AES Candidates", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[2] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas, Luke O'Connor, Mohammad Peyravian, David Safford and Nevenko Zunic, "MARS - A Candidate Cipher for AES", NIST AES Proposal, June 1998. Available at: http://www.research.ibm.com/security/mars.pdf

[3] Suresh Chari, Charantjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi, "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[4] Suresh Chari, Charantjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks", in *Proceedings of Advances in Cryptology – CRYPTO'99*, Springer-Verlag, 1999, pp. 398-412.

[5] Jean-Sébastien Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 292-302.

[6] Jean-Sébastien Coron and Louis Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 2000.

[7] John Daemen and Vincent Rijmen, "Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals", in *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm, March 1999.

[8] John Daemen, Michael Peters and Gilles Van Assche, "Bitslice Ciphers and Power Analysis Attacks", in *Proceedings of Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.

[9] Paul N. Fahn and Peter K. Pearson, "IPA: A New Class of Power Attacks", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 173-186.

[10] Louis Goubin and Jacques Patarin, "Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique", European Patent, Schlumberger, February 4th, 1999, Publication Number: 2789535.

[11] Louis Goubin and Jacques Patarin, "DES and Differential Power Analysis – The Duplication Method", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 158-172.

[12] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Introduction to Differential Power Analysis and Related Attacks", http://www.cryptography.com/dpa/technical, 1998.

[13] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Differential Power Analysis", in *Proceedings of Advances in Cryptology – CRYPTO'99*, Springer-Verlag, 1999, pp. 388-397.

[14] Xuejia Lai and James Massey, "A Proposal for a New Block Encryption Standard", in *Advances in Cryptology - EUROCRYPT '90 Proceedings*, Springer-Verlag, 1991, pp. 389-404.

[15] Thomas S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks", in *Proceedings of Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.

[16] Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Investigations of Power Analysis Attacks on Smartcards", in *Proceedings of USENIX Workshop on Smartcard Technology*, May 1999, pp. 151-161.

[17] Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 144-157.

[18] Ronald L. Rivest, Matthew J.B. Robshaw, Ray Sidney and Yiqun L. Yin, "The RC6 Block Cipher", v1.1, August 20, 1998. Available at: ftp://ftp.rsasecurity.com/pub/rsalabs/aes/rc6v11.pdf

[19] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson, "Twofish: A 128-Bit Block Cipher", June 15, 1998, AES submission available at: http://www.counterpane.com/twofish.pdf

Annex 1: Proof of Theorem 1

To prove theorem 1, we prove the following more precise result:

Lemma 1 *For any integer $k \geq 1$:*

$$\Phi_a(r) \equiv \Big\{ a \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a}) \Big) \wedge (2^i a) \wedge (2^i r) \Big] \Big\}$$

$$- \Big[\Big(\bigwedge_{j=1}^{k-1} (2^j \bar{a}) \Big) \wedge (2^k a) \wedge (2^k r) \Big] \bmod 2^K,$$

where $\bar{a}$ stands for the ones complement of a, and $\wedge$ stands for the boolean "AND" operator.

Theorem 1 easily follows from Lemma 1, by considering the particular value $k = K$ (and taking $a = x'$).
To prove Lemma 1, we will use the following elementary result.

Lemma 2 *For any integers u and v:*

$$u - v \equiv (u \oplus v) - 2(\bar{u} \wedge v) \bmod 2^K.$$

Proof of Lemma 2 (sketch): $u \oplus v$ gives almost the same result as $u - v$, except that carries have been forgotten. For a given index, a carry appears if and only if a '1' bit (from v) is subtracted from a '0' bit (from u), which corresponds to a '1' bit in $\bar{u} \wedge v = 1$. Since the carry is then subtracted in the *next* index, $\bar{u} \wedge v$ has to be shifted left, which is the same as to be doubled, before being subtracted from $u \oplus v$.

Proof of Lemma 1: We proceed by induction on k.

- We first apply Lemma 2 with $u = a \oplus r$ and $v = r$:

$$\Phi_a(r) \equiv (a \oplus r) - r \equiv a - 2(\overline{a \oplus r} \wedge r) \bmod 2^K.$$

Since $\overline{a \oplus r} = a \oplus \bar{r}$, we have:

$$\Phi_a(r) \equiv a - 2((a \oplus \bar{r}) \wedge r) \equiv a - 2(a \wedge r) \bmod 2^K,$$

which proves the case $k = 1$ of Lemma 1 (conventionally, the empty product $\bigwedge_{j=1}^{0}$ equals the identity element of the $\wedge$ operator).

- Let us suppose that the result of Lemma 1 is true for k:

$$\Phi_a(r) \equiv \Big\{a \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big]\Big\}$$

$$-\Big[\Big(\bigwedge_{j=1}^{k-1} (2^j \bar{a})\Big) \wedge (2^k a) \wedge (2^k r)\Big] \bmod 2^K$$

and let us show that it is also true for $k+1$.

Let

$$u = a \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big]$$

and

$$v = \Big(\bigwedge_{j=1}^{k-1} (2^j \bar{a})\Big) \wedge (2^k a) \wedge (2^k r).$$

We first obtain:

$$u \oplus v = a \oplus \bigoplus_{i=1}^{k} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big].$$

Moreover,

$$\bar{u} = \overline{a \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big]}$$

$$= \bar{a} \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big],$$

so that:

$$\bar{u} \wedge v = \Big\{\bar{a} \oplus \bigoplus_{i=1}^{k-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a})\Big) \wedge (2^i a) \wedge (2^i r)\Big]\Big\}$$

$$\wedge \Big(\bigwedge_{j=1}^{k-1} (2^j \bar{a})\Big) \wedge (2^k a) \wedge (2^k r).$$

Therefore

$$\bar{u} \wedge v = \Big(\bigwedge_{j=0}^{k-1} (2^j \bar{a})\Big) \wedge (2^k a) \wedge (2^k r)$$

because to each index i, $1 \leq i \leq k-1$ in u corresponds an index j, $1 \leq j \leq k-1$ in v (namely $j = i$), such that:

$$(2^i a) \wedge (2^j \bar{a}) = 0.$$

Therefore, applying Lemma 2:

$$\Phi_a(r) \equiv \Big\{ a \oplus \bigoplus_{i=1}^{k} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j \bar{a}) \Big) \wedge (2^i a) \wedge (2^i r) \Big] \Big\}$$

$$- \Big[\Big(\bigwedge_{j=1}^{k} (2^j \bar{a}) \Big) \wedge (2^{k+1} a) \wedge (2^{k+1} r) \Big] \bmod 2^K.$$

Annex 2: Proof of Theorem 2

We begin by the following elementary result:

Lemma 3 *For any z and δ, the following identity holds:*

$$z + \delta \equiv \overline{\bar{z} - \delta} \bmod 2^K.$$

Proof of Lemma 3: It is easy to see that, for any λ,

$$\lambda + \bar{\lambda} + 1 \equiv 0 \bmod 2^K.$$

Applying this identity successively with $\lambda = \bar{z} - \delta$ and $\lambda = z$, we obtain:

$$\overline{\bar{z} - \delta} \equiv -(\bar{z} - \delta) - 1 \equiv -((-z - 1) - \delta) - 1 = z + \delta \bmod 2^K.$$

Proof of Theorem 2: We first apply Lemma 3 with $z = A$ and $\delta = r$:

$$A + r = \overline{\bar{A} - r}$$

Moreover,

$$\bar{A} = A \oplus (-1) = ((A \oplus r) \oplus (-1)) \oplus r = \overline{A \oplus r} \oplus r.$$

Hence

$$A + r = \overline{(\overline{A \oplus r} \oplus r) - r} = \overline{\Phi_{\overline{A \oplus r}}(r)}.$$

From Theorem 1 (with $\overline{A \oplus r}$ instead of x'), we know that:

$$\Phi_{\overline{A \oplus r}}(r) = \overline{A \oplus r} \oplus \bigoplus_{i=1}^{K-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j (A \oplus r)) \Big) \wedge (2^i (\overline{A \oplus r})) \wedge (2^i r) \Big],$$

so that

$$A + r = A \oplus r \oplus \bigoplus_{i=1}^{K-1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j (A \oplus r)) \Big) \wedge (2^i A)) \wedge (2^i r) \Big].$$

Let us denote, for any integer $k \geq 0$,

$$u_k = \bigoplus_{i=1}^{k} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j (A \oplus r)) \Big) \wedge (2^i A)) \wedge (2^i r) \Big].$$

From the definition of u_k, we have $u_0 = 0$ and $A + r = A \oplus r \oplus u_{K-1}$. Moreover for all $k \geq 0$,

$$u_{k+1} = \bigoplus_{i=1}^{k+1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j (A \oplus r) \Big) \wedge (2^i A)) \wedge (2^i r) \Big]$$

$$= 2(A \wedge r) \oplus \bigoplus_{i=2}^{k+1} \Big[\Big(\bigwedge_{j=1}^{i-1} (2^j (A \oplus r) \Big) \wedge (2^i A)) \wedge (2^i r) \Big],$$

so that, if we denote $i' = i - 1$ and $j' = j - 1$:

$$u_{k+1} = 2(A \wedge r) \oplus \bigoplus_{i'=1}^{k} \Big[\Big(\bigwedge_{j'=0}^{i'-1} (2^{j'+1} (A \oplus r) \Big) \wedge (2^{i'+1} A)) \wedge (2^{i'+1} r) \Big]$$

$$= 2\Big\{ (A \wedge r) \oplus \bigoplus_{i'=1}^{k} \Big[\Big(\bigwedge_{j'=0}^{i'-1} (2^{j'} (A \oplus r) \Big) \wedge (2^{i'} A)) \wedge (2^{i'} r) \Big] \Big\}$$

$$= 2\Big\{ (A \wedge r) \oplus (A \oplus r) \wedge \bigoplus_{i'=1}^{k} \Big[\Big(\bigwedge_{j'=1}^{i'-1} (2^{j'} (A \oplus r) \Big) \wedge (2^{i'} A)) \wedge (2^{i'} r) \Big] \Big\}$$

$$= 2[(A \wedge r) \oplus (A \oplus r) \wedge u_k].$$

Annex 3: Proof of Theorem 3

Let δ be any value in I. We begin by proving that Θ_δ is surjective.
Let $y \in I$. If we denote:

$$\gamma = \bigoplus_{i=0}^{K-1} \Big[\Big(\bigwedge_{j=1}^{i} (2^{j-1} \delta) \Big) \wedge (2^i y) \Big]$$

(conventionally, the empty product $\bigwedge_{j=1}^{0}$ equals the identity element of the $\wedge$ operator), we have:

$$\gamma \oplus (2\gamma) \wedge \delta = \gamma \oplus \bigoplus_{i=0}^{K-1} \Big[\Big(\bigwedge_{j=0}^{i} (2^j \delta) \Big) \wedge (2^{i+1} y) \Big],$$

so that, if we denote $i' = i + 1$ and $j' = j + 1$:

$$\gamma \oplus (2\gamma) \wedge \delta = \gamma \oplus \bigoplus_{i'=1}^{K} \Big[\Big(\bigwedge_{j'=1}^{i'} (2^{j'-1} \delta) \Big) \wedge (2^{i'} y) \Big].$$

From the definition of γ, it is easy to see that:

$$\bigoplus_{i'=1}^{K} \Big[\Big(\bigwedge_{j'=1}^{i'} (2^{j'-1} \delta) \Big) \wedge (2^{i'} y) \Big] = \gamma \oplus y$$

Therefore:

$$\gamma \oplus (2\gamma) \wedge \delta = y.$$

We have proven that, for any $y \in I$, a value $\gamma \in I$ exists such that $\Theta_\delta(\gamma) = y$. As a consequence, Θ_δ is surjective. Since it maps I onto itself, we deduce that Θ_δ is bijective.

A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems

PKC'2003

Abstract

As Elliptic Curve Cryptosystems are becoming more and more popular and are included in many standards, an increasing demand has appeared for secure implementations that are not vulnerable to side-channel attacks. To achieve this goal, several generic countermeasures against Power Analysis have been proposed in recent years.

In particular, to protect the basic scalar multiplication – on an elliptic curve – against Differential Power Analysis (DPA), it has often been recommended using "random projective coordinates", "random elliptic curve isomorphisms" or "random field isomorphisms". So far, these countermeasures have been considered by many authors as a cheap and secure way of avoiding the DPA attacks on the "scalar multiplication" primitive.

However we show in the present paper that, for many elliptic curves, such a DPA-protection of the "scalar" multiplication is not sufficient. In a *chosen message* scenario, a Power Analysis attack is still possible even if one of the three aforementioned countermeasures is used. We expose a new Power Analysis strategy that can be successful for a large class of elliptic curves, including most of the sample curves recommended by standard bodies such as ANSI, IEEE, ISO, NIST, SECG or WTLS.

This result means that the problem of randomizing the basepoint may be more difficult than expected and that "standard" techniques have still to be improved, which may also have an impact on the performances of the implementations.

Keywords.

Public-key cryptography, Side-channel attacks, Power Analysis, Differential Power Analysis (DPA), Elliptic curves, Smartcards.

1 Introduction

Since their introduction by V. Miller [21] and N. Koblitz [15], elliptic curve cryptosystems have been included in many international standards. One of their advantages is the small size of their keys, compared to those of RSA and ElGamal-type cryptosystems. Therefore, there has been a growing interest in implementing such cryptographic schemes in low-cost cryptographic devices such as smartcards.

Whereas the mathematical aspects of the security of such elliptic curve cryptosystems have been scrutinized for years now, a new threat appeared in 1998 when P. Kocher *et al.* [16, 17] introduced attacks based on power analysis. The idea of this new class of attacks is to monitor the power consumption of the electronic device while it is performing the cryptographic computation and

then to use a statistical analysis of the measured consumption curves to deduce some information about the secret key stored in the device. The initial focus was on symmetric cryptosystems such as DES but public key cryptosystems were also shown vulnerable, including RSA [20] and elliptic curve cryptosystems [7].

The simple power analysis (SPA) only uses a single observed information. Two main stategies have been suggested to avoid this SPA attack.

The first strategy consists in hiding the fact that, during the computation of a scalar multiplication $d.P$ (d being an integer and P a point of the elliptic curve), the nature of the basic operations (*e.g.* addition or doubling) executed at each step depends on the value of the secret exponent d. Following this strategy, J.S. Coron proposed the "double-and-add-always" method [7]. The "Montgomery" method [23] also proved useful, giving a natural way of avoiding both timing and SPA attacks [25, 27]. For binary fields GF(2^m) a trick allows the computation of the scalar multiplication to be performed without using the y-coordinates [1, 19]. This property was extended to the case of prime fields GF(p) for elliptic curves which have "Montgomery-form" [28, 22] and then for any elliptic curve on GF(p) [12, 4, 8].

The second strategy consists in using indistinguishable addition and doubling in the scalar multiplication [5]. This has been shown feasible for some classes of curves over a prime field GF(p): Hesse-type [29, 13] and Jacobi-type [18] elliptic curves give a unified formula for computing both addition and doubling. A unified formula was recently proposed by E. Brier and M. Joye [4] to achieve the same indistinguishability for any elliptic curve on GF(2^m) or GF(p). For the binary field case, the insertion of dummy operations is also possible [3] to build an indistinguishable adding and doubling.

As pointed out in [7, 27, 14], these anti-SPA methods are not sufficient to prevent DPA attacks. However, many countermeasures have been proposed to transform an SPA-resistant scheme into a DPA-resistant scheme.

In [7], J.S. Coron suggested three anti-DPA methods: randomizing the secret exponent d, adding a random point R to P and using randomized projective homogeneous coordinates. The first two methods have been considered with skepticism in [27] and [12], but the third one is widely accepted: see [25], [27] or [18].

In the same spirit, M. Joye and C. Tymen [14] proposed two other generic methods: performing the computations in another elliptic curve which is deduced from the usual one through a random isomorphism, and performing the basic field operations with another representation of the field which is deduced from the usual one though a random field isomorphism. Note that [14] also gives a specific method for ABC curves (see also [9]).

Hence [7, 14] propose three generic methods ("random projective coordinates", "random elliptic curve isomorphisms" and "random field isomorphisms"), which so far have been considered by many authors as a cheap and secure way of thwarting the DPA attacks: see *e.g.* [26], [3], [12]. For example it is stated in [4] that DPA attacks are not really a threat for elliptic curve cryptography since they are easily avoided by randomizing the inputs.

However, in the present paper we prove that, for a large class of elliptic curves, a Power Analysis attack can still work, even if we apply one of the three countermeasures above (together with an SPA countermeasure, such as "Add-and-double always", the Montgomery method, or a unified add/double formula).

In our scenario, the attacker can choose the message, *i.e.* the input of the "scalar multiplication" primitive. The only way the sensitive data are blinded is by using random projective coordinates (for the input), random elliptic curve isomorphisms (for the curve itself) or random field isomorphism (for the algebraic structure).

The paper is organized as follows. In section 2, we give the mathematical background about

elliptic curves and scalar multiplication. In section 3, we describe our new strategy of attack, for each of the three DPA-countermeasures of [7, 14]. In section 4, we study more precisely the necessary conditions on the elliptic curve for our attack to work, and show that most of the sample curves proposed by standardization bodies [2, 10, 11, 24, 30, 31] verify these conditions.

2 Mathematical Background

2.1 Parametrizations of Elliptic Curves

General (affine) Weierstraß form

We consider the elliptic curve defined over a field K by its Weierstraß equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

We denote by $E(K)$ the set of points $(x,y) \in K^2$ satisfying this equation. If we introduce a formal "point at infinity" denoted by $\mathcal{O}$, the set $E(K) \bigcup \mathcal{O}$ can be equiped with an operation $+$ which makes it an abelian group whose identity element is $\mathcal{O}$.

Projective coordinates

To avoid costly inversions, it is convenient to use projective coordinates. Among many possibilities developed in [6], we describe *homogeneous* and *Jacobian* projective coordinates.
Homogeneous projective coordinates are obtained by setting $x = X/Z$ and $y = Y/Z$, so that the general Weierstraß equation becomes

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

The point at infinity $\mathcal{O}$ is then represented by $(0,\theta,0)$ for some $\theta \in K^*$, the affine point (x,y) is represented by a projective point $(\theta x,\theta y,\theta)$ for some $\theta \in K^*$ and a projective point $(X,Y,Z) \neq \mathcal{O}$ corresponds to the affine point $(X/Z,Y/Z)$.
Jacobian projective coordinates are obtained by setting $x = X/Z^2$ and $y = Y/Z^3$, so that the general Weierstraß equation becomes

$$E : Y^2 + a_1XYZ + a_3YZ^3 = X^3 + a_2X^2Z^2 + a_4XZ^4 + a_6Z^6.$$

The point at infinity $\mathcal{O}$ is then represented by $(\theta^2,\theta^3,0)$ for some $\theta \in K^*$, the affine point (x,y) is represented by a projective point $(\theta^2x,\theta^3y,\theta)$ for some $\theta \in K^*$ and a projective point $(X,Y,Z) \neq \mathcal{O}$ corresponds to the affine point $(X/Z^2,Y/Z^3)$.

Simplified (affine) Weierstraß forms

When $\mathrm{Char}(K) \neq 2,3$, the general Weierstraß equation can be simplified to

$$E : y^2 = x^3 + ax + b$$

and the addition formulas, giving $P + Q = (x_3,y_3)$ from $P = (x_1,y_1)$ and $Q = (x_2,y_2)$, become

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases} \quad \text{with} \quad \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

When $\mathrm{Char}(K) = 2$ and the curve is non-supersingular, the general Weierstraß equation can be simplified to

$$E : y^2 + xy = x^3 + ax^2 + b$$

and the addition formulas to

$$\begin{cases} x_3 = \lambda^2 + \lambda + a + x_1 + x_2 \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \end{cases} \quad \text{with} \quad \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{if } P \neq Q \\ x_1 + \frac{y_1}{x_1} & \text{if } P = Q \end{cases}$$

Montgomery form

In order to ease the additions, P.L. Montgomery considered in [23] the family of elliptic curves of the following form (on a field K of characteristic $\neq 2$):

$$E : By^2 = x^3 + Ax^2 + x \qquad \text{with } B(A^2 - 4) \neq 0.$$

As noticed in [27], on such elliptic curves, the point (0,0) is of order 2 and the cardinality of $E(K)$ is always divisible by 4.

Hessian form

The Hessian-type elliptic curves were considered because they provide a unified formula for adding and doubling. Defined as the intersection of two quadrics, they can be given in the following form (on a field $K = \mathrm{GF}(q)$ with $q \equiv 2 \bmod 3$)

$$E : x^3 + y^3 + 1 = 3Dxy \qquad \text{with } D \in K, D^3 \neq 1.$$

As mentioned in [29, 13], point $(-1,0)$ has order 3, that implies that the cardinality of $E(K)$ is always divisible by 3.

2.2 Usual SPA Countermeasures

To compute the scalar multiplication $d.P$, where $d = d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + ... + d_1 2 + d_0$, with $d_{n-1} = 1$ and $P \in E(K)$, the following generic schemes have been proposed.

Classical binary method

This method (see Algorithm 1) is analogous to the "square-and-multiply" principle used in RSA. Note that an analogous method exists, which is from the least significant bit. As noticed in [7], both are vulnerable to SPA attacks. That is why two other methods were introduced: the "Double-and-add-always" and the "Montgomery" methods.

Double-and-add-always

This method (Algorithm 2) was proposed in [7]. Note that an analogous method also exists, which is from the least significant bit [7, 12]. Both are SPA-resistant.

Montgomery method

This method (Algorithm 3) was originally proposed in [23] and then elaborated in [1, 19, 25, 27, 28, 26, 22, 12, 4, 8]. It is SPA-resistant.

Algorithme 10 Binary method (from the most significant bit)

Input: d, P
Output: $Q = d.P$
  $Q := P$
  for $i = n - 2$ down to 0 **do**
    $Q := 2.Q$
    if $d_i = 1$ **then**
      $Q := Q + P$
    end if
  end for
  Return Q

Algorithme 11 Double-and-add-always (from the most significant bit)

Input: d, P
Output: $Q_0 = d.P$
  $Q_0 := P$
  for $i = n - 2$ down to 0 **do**
    $Q_0 := 2.Q_0$
    $Q_1 := Q_0 + P$
    $Q_0 := Q_{d_i}$
  end for
  Return Q_0

Algorithme 12 Montgomery's method

Input: d, P
Output: $Q_0 = d.P$
  $Q_0 := P$
  $Q_1 := 2.P$
  for $i = n - 2$ down to 0 **do**
    $Q_{1-d_i} := Q_0 + Q_1$
    $Q_{d_i} := 2.Q_{d_i}$
  end for
  Return Q_0

3 Our new Power Analysis attack

We present here a Power Analysis attack that can work on many elliptic curves, even if an SPA-countermeasure (such as *Double-and-add-always* or the *Montgomery* method) is used, together with one of three aforementioned DPA-countermeasures (*Random projective coordinates*, *Random elliptic curve isomorphisms* or *Random field isomorphisms*).

3.1 The strategy of the attack

In this section, we describe the generic attack on an elliptic curve scalar multiplication, SPA-protected with *Double-and-add-always* or the *Montgomery* method. Note however that the attack is not limited to the case of binary methods (such as Algorithm 2 or Algorithm 3) and can be extended to the case of other addition chains.
Suppose the attacker already knows the highest bits d_{n-1}, ..., d_{i+1} of the secret multiplier d. We illustrate below how he can find the next bit d_i.
Let us suppose that the elliptic curve $E(K)$ contains a "special" point $P_0 \neq \mathcal{O}$, *i.e.* a point $P_0 \neq \mathcal{O}$ such that one of the (affine or projective) coordinates equals 0 in K.
Note that, for each of the three aforementioned DPA-countermeasures, the randomization does not affect the "special" property of the point P_0 (see section 3.2).

Double-and-add-always

In Algorithm 2, for any given input point P, the value Q_0 obtained at the end of the i-th step of the loop is

$$Q_0 = \Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i} + d_i \Big).P.$$

We then have two cases:

- If $d_i = 0$, the values that appear during the $(i+1)$-st step of the loop are $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} \Big).P$ and $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big).P$.
- If $d_i = 1$, the values that appear during the $(i+1)$-st step of the loop are $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 2 \Big).P$ and $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3 \Big).P$.

We consider the point P_1 given by

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big)^{-1} \bmod |E(K)| \Big].P_0$$

if $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1 \Big)$ is coprime to $|E(K)|$ (this corresponds to the guess $d_i = 0$), or

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3 \Big)^{-1} \bmod |E(K)| \Big].P_0$$

if $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3\Big)$ is coprime to $|E(K)|$ (this corresponds to the guess $d_i = 1$). In many cases, both possibilities can be chosen.
Let us now denote by C_r, for $1 \leq r \leq R$, the power consumption curves associated to r distinct computations of $d.P_1$. Because of the randomization performed before each computation, two curves corresponding to the same input value can be different.
We then consider the mean curve

$$\mathcal{M}_{P_1} = \frac{1}{R} \sum_{r=1}^{R} C_r.$$

If the guess for d_i (*i.e.* the choice for the point P_1) is incorrect, then $\mathcal{M}_{P_1} \simeq 0$, since the values appearing in the $(i+1)$-st step of the loop in Algorithm 2, are correctly randomized.
On the contrary, if the guess for d_i is correct, the mean curve $\mathcal{M}_{P_1}$ shows appreciable consumption "peaks" (compared to the mean power consumption of random points), corresponding to the treatment of the zero value in the $(i+1)$-st step of the loop.
Once d_i is known, the remaining bits d_{i-1}, ..., d_0 are recovered recursively, in the same way.

The Montgomery method

In Algorithm 3, for any given input point P, the values Q_0 and Q_1 obtained at the end of the i-th step of the loop are

$$Q_0 = \Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i} + d_i\Big).P$$

$$Q_1 = \Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i} + d_i + 1\Big).P$$

We then have two cases:

- If $d_i = 0$, the values that appear during the $(i+1)$-st step of the loop are $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1\Big).P$ on the one hand, and $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1}\Big).P$ or $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 2\Big).P$ on the other hand.
- If $d_i = 1$, the values that appear during the $(i+1)$-st step of the loop are $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3\Big).P$ on the one hand, and $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 2\Big).P$ or $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 4\Big).P$ on the other hand.

We then consider then point P_1 given by

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1\Big)^{-1} \bmod |E(K)|\Big].P_0$$

if $\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1\Big)$ is coprime to $|E(K)|$ (the guess is $d_i = 0$), or

$$P_1 = \Big[\Big(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3\Big)^{-1} \bmod |E(K)|\Big].P_0$$

if $\left(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3\right)$ is coprime to $|E(K)|$ (the guess is $d_i = 1$).
The rest of the attack is then exactly the same as for the "Double-and-add-always" method: the bit d_i is found by power analysis, and the remaining bits d_{i-1}, ..., d_0 in the same way.

3.2 Application to three usual DPA-countermeasures

Random projective coordinates

The basic idea of this method is the following. The computation $Q = d.P$ is performed in projective coordinates. The basepoint $P = (x,y)$ can be represented by $(\theta x,\theta y,\theta)$ (*homogeneous* projective coordinates) or $(\theta^2 x,\theta^3 y,\theta)$ (*Jacobian* projective coordinates) for some $\theta \in K^*$.
Thus the computation is performed in 3 steps:

1. Choose a random $\theta \in K^*$ and let $P' = (\theta x,\theta y,\theta)$ (*homogeneous* projective coordinates) or $P' = (\theta^2 x,\theta^3 y,\theta)$ (*Jacobian* projective coordinates).
2. Compute $Q' = (X',Y',Z') = d.P'$.
3. Compute $Q = (X'/Z',Y'/Z')$ (*homogeneous* projective coordinates) or $Q = (X'/Z'^2,Y'/Z'^3)$ (*Jacobian* projective coordinates).

It is easy to see that the "special" point mentioned in section 3.1 remains of the form $(X,0,Z)$ or $(0,Y,Z)$, whatever the random value θ may be. This shows that the above strategy applies.

Random elliptic curve isomorphisms

This method applies for an elliptic curve $E : y^2 = x^3 + ax + b$ on a field K of characteristic $\neq 2,3$. For $P = (x,y)$, the computation of $Q = d.P$ is performed as follows:

1. Choose a random $\theta \in K^*$ and let $P' = (\theta^2 x,\theta^3 y,1)$, $a' = \theta^{-4} a$ and $b' = \theta^{-6} b$.
2. Compute $Q' = (X',Y',Z') = d.P'$ in $E' : Y^2 Z = X^3 + a' X Z^2 + b' Z^3$ (homogeneous projective coordinates).
3. Compute $Q = (\theta^2 X'/Z',\theta^3 Y'/Z')$.

A variant consists in computing $Q' = d.P'$ in $E' : Y^3 = X^3 + a' X Z^4 + b' Z^6$ (Jacobian projective coordinates). It is easy to see that the "special" point mentioned in section 3.1 remains of the form $(X,0,Z)$ or $(0,Y,Z)$, whatever the random value θ may be. This shows that the strategy of section 3.1 applies again.

Random field isomorphisms

This method applies for an elliptic curve over a field $K = \mathrm{GF}(2^m) = \mathrm{GF}(2)[X]/\Pi(X)$, where Π is an irreducible polynomial of degree m over $\mathrm{GF}(2)$. The idea is that there are many such irreducible polynomials, so that K can be replaced (randomly) by an isomorphic field K'. The computation of $Q = d.P$ is performed as follows:

1. Choose a random irreducible polynomial Π' of degree m over $\mathrm{GF}(2)$ and let $K' = \mathrm{GF}(2)[X]/\Pi'(X)$.
2. Let φ be the field isomorphism between K and K' and $P' = \varphi(P)$.
3. Compute $Q' = d.P' \in K'^2$ in $E_{/K'}$.
4. Compute $Q = \varphi^{-1}(Q') \in K^2$.

Again, the "special" point mentioned in section 3.1 remains of the form $(x,0)$ or $(0,y)$ (with the usual representation of the zero value), whatever the random polynomial Π' may be, so that the strategy of section 3.1 also applies.

4 Practical Applications

4.1 Computation of the "special" point

Special points $(0,y)$

For a non-singular binary elliptic curve, whose reduced Weierstraß form is $E : y^2 + xy = x^3 + ax^2 + b$ over $K = \mathrm{GF}(2^m)$, we can choose $P_0 = (0,b^{2^{m-1}})$ as the "special" point.
For an elliptic curve $E : y^2 = x^3 + ax + b$ over a prime field $K = \mathrm{GF}(p)$ $(p > 3)$, a special point of the form $(0,y)$ exists if and only if b is a quadratic residue modulo p, *i.e.* $\left(\frac{b}{p}\right) = 1$, where $\left(\frac{\cdot}{\cdot}\right)$ is the Legendre symbol.
Among the standardized curves over a prime field satisfying the condition are: four curves proposed in FIPS 186-2 [24], the basic curve (curve number 7) proposed in WTLS [31], the seven curves proposed in ANSI X9.62 [2] (Annex J5), and two curves proposed in the working draft ISO/IEC 15946-4 [11] (Annexes A2.1 and A3.1). Only one curve of FIPS 186-2 (P224) and four curves of ISO/IEC 15946-4 (Annexes A1.1[1], A4.1, A5.1 and A6.1) have no special point $(0,y)$.

Special points $(x,0)$

For an elliptic curve $E : y^2 = x^3 + ax + b$ over a prime field $K = \mathrm{GF}(p)$ $(p > 3)$, a special point of the form $(x,0)$ exists if and only if the equation $x^3 + ax + b = 0$ has at least one root α in K. Note that $P_0 = (\alpha,0)$ is then a point of order 2 in $E(K)$. At first glance, it may seem that the strategy of section 3.1 fails, because P_1 does not depend on the guess made on d_i (P_1 is always equal to P_0). However, the successive values of Q that appear during Algorithm 2, for $i = n-2,...,0$ are either $\mathcal{O}$ (if $d_i = 0$) or P (if $d_i = 1$). Therefore the mean curve $\mathcal{M}_{P_1}$ shows in fact many peaks: for instance if Algorithm 2 is applied, with random homogeneous projective coordinates, the chip instructions manipulating $\mathcal{O} = (0,\theta,0)$ are likely to create 2 such peaks (one for each 0), whereas the instructions manipulating $(\theta x,0,\theta)$ are likely to create only 1 peak. This allows the attacker to recover all the bits d_i of the secret exponent d with only one application of the strategy of 3.1.
Some particular classes of curves automatically have such points of order 2. As mentioned in section 2.1, for all Montgomery-form elliptic curves, $(0,0)$ is of order 2: its double is $\mathcal{O} = (0,1,0)$. For the Hessian form, all (x,x) are of order 2: their double is $\mathcal{O} = (-1,1,0)$.

4.2 Cardinality of the elliptic curve

Another condition for our strategy of attack to work is the fact that at least one of the values $\left(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 1\right)$ and $\left(\sum_{j=i+1}^{n-1} d_j 2^{j-i+1} + 3\right)$ is coprime to $|E(K)|$.
Over a prime field, FIPS 186-2 [24] or SECG [30] recommend to use elliptic curves of prime cardinality, and binary curves of cardinality $2q$ or $4q$ (q prime). All the curves proposed by WTLS [31] and ISO/IEC 15946-4 [11] have cardinality q, $2q$, $4q$, $6q$ (q prime). It is also true for most of the curves of ANSI X9.62 [2].
This shows that the condition above is true for most standardized elliptic curves.

1. This curve however has a point of order 2, hence a special point $(x,0)$.

5 Conclusion

This attack we present here shows that the problem of randomizing the basepoint may be more difficult than expected and that "standard" techniques for securing the "scalar multiplication" primitive still have to be improved. Evaluating the performances of secure implementations of elliptic curve cryptosystems will require to take those improvements into account. The results of this paper also highlight the necessity to choose a message blinding method (before entering the "scalar multiplication" primitive) that prevents an attacker from choosing the messages.

References

[1] G.B. Agnew, R.C. Mullin, S.A. Vanstone, *An Implementation of Elliptic Curve Cryptosystems over* $\mathbf{F}_{2^{155}}$. IEEE Journal on Selected Areas in Communications, vol. 11, n. 5, pp 804-813, 1993.

[2] ANSI X9.62, Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 1999.

[3] A. Bellezza, *Countermeasures against Side-Channel Attacks for Elliptic Curve Cryptosystems*. IACR, Cryptology ePrint Archive, 2001/103, 2001. Available from `http://eprint.iacr.org/2001/103/`

[4] E. Brier, M. Joye, *Weierstraß Elliptic Curves and Side-Channel Attacks*. In Proceedings of PKC'2002, LNCS 2274, pp. 335-345, Springer-Verlag, 2002.

[5] C. Clavier, M. Joye, *Universal Exponentiation Algorithm – A First Step towards Provable SPA-Resistance*. In Proceedings of CHES'2001, LNCS 2162, pp. 300-308, Springer-Verlag, 2001.

[6] H. Cohen, A. Miyaji, T. Ono, *Efficient Elliptic Curve Exponentiation Using Mixed Coordinates*. In Proceedings of ASIACRYPT'98, LNCS 1514, pp. 51-65, Springer-Verlag, 1998.

[7] J.-S. Coron, *Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems*. In Proceedings of CHES'99, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.

[8] W. Fischer, C. Giraud, E.W. Knudsen, J.-P. Seifert, *Parallel Scalar Multiplication on General Elliptic Curves over* $\mathbf{F}_p$ *hedged against Non-Differential Side-Channel Attacks*. IACR, Cryptology ePrint Archive, 2002/007, 2002. Available from `http://eprint.iacr.org/2002/007/`

[9] M.A. Hasan, *Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems*. In Proceedings of CHES'2000, LNCS 1965, pp. 93-108, Springer-Verlag, 2000.

[10] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, 2000. Available from `http://groupe.ieee.org/groups/1363/`

[11] ISO/IEC 15946-4, *Information technology - Security techniques – Cryptographic techniques based on elliptic curves - Part 4: Digital signatures giving message recovery*. Working Draft, JTC 1/SC 27, December 28th, 2001.

[12] T. Izu, T. Takagi, *A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks*. In Proceedings of PKC'2002, LNCS 2274, pp. 280-296, Springer-Verlag, 2002.

[13] M. Joye, J.-J. Quisquater, *Hessian Elliptic Curves and Side-Channel Attacks*. In Proceedings of CHES'2001, LNCS 2162, pp. 412-420, Springer-Verlag, 2001.

[14] M. Joye, C. Tymen, *Protections against Differential Analysis for Elliptic Curve Cryptography – An Algebraic Approach*. In Proceedings of CHES'2001, LNCS 2162, pp. 377-390, Springer-Verlag, 2001.

[15] N. Koblitz, *Elliptic curve cryptosystems*. Mathematics of Computation, Vol. 48, pp. 203-209, 1987.

[16] P. Kocher, J. Jaffe, B. Jun, *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Available from `http://www.cryptography.com/dpa/technical/index.html`

[17] P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.

[18] P.-Y. Liardet, N.P. Smart, *Preventing SPA/DPA in ECC system using the Jacobi Form*. In Proceedings of CHES'2001, LNCS 2162, pp. 401-411, Springer-Verlag, 2001.

[19] J. López, R. Dahab, *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*. In Proceedings of CHES'99, LNCS 1717, pp. 316-327, Springer-Verlag, 1999.

[20] T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*. In Proceedings of CHES'99, pp. 144-157, Springer-Verlag, 1999.

[21] V. Miller, *Uses of elliptic curves in cryptography*. In Proceedings of CRYPTO'85, LNCS 218, pp. 417-426, Springer-Verlag, 1986.

[22] B. Möller, *Securing Elliptic Curve Point Multiplication against Side-Channel Attacks*. In Proceedings of ISC'2001, LNCS 2200, pp. 324-334, Springer-Verlag, 2001.

[23] P.L. Montgomery, *Speeding the Pollard and Elliptic Curve Methods for Factorizations*. Mathematics of Computation, vol. 48, pp. 243-264, 1987.

[24] National Institute of Standards and Technology (NIST), *Recommended Elliptic Curves for Federal Government Use*. In the appendix of FIPS 186-2, available from `http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf`

[25] K. Okeya, H. Kurumatani, K. Sakurai, *Elliptic Curve with the Montgomery Form and their cryptographic Applications*. In Proceedings of PKC'2000, LNCS 1751, pp. 238-257, Springer-Verlag, 2000.

[26] K. Okeya, K. Miyazaki, K. Sakurai, *A Fast Scalar Multiplication Method with Randomized Projective Coordinates on a Montgomery-form Elliptic Curve Secure against Side Channel Attacks*. In Pre-proceedings of ICICS'2001, pp. 475-486, 2001.

[27] K. Okeya, K. Sakurai, *Power Analysis Breaks Elliptic Curve Cryptosystem even Secure against the Timing Attack*. In Proceedings of INDOCRYPT'2000, LNCS 1977, pp. 178-190, Springer-Verlag, 2000.

[28] K. Okeya, K. Sakurai, *Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-coordinate on a Montgomery-form Elliptic Curve*. In Proceedings of CHES'2001, LNCS 2162, pp. 126-141, Springer-Verlag, 2001.

[29] N.P. Smart, *The Hessian Form of an Elliptic Curve*. In Proceedings of CHES'2001, LNCS 2162, pp. 118-125, Springer-Verlag, 2001.

[30] Standards for Efficient Cryptography Group (SECG), *Specification of Standards for Efficient Cryptography*, Ver. 1.0, 2000. Available from `http://www.secg.org/secg_docs.htm`

[31] Wireless Application Protocol (WAP) Forum, *Wireless Transport Layer Security (WTLS) Specification*. Available from `http://www.wapforum.org`

A Generic Protection against High-Order Differential Power Analysis

FSE'2003
Article avec Mehdi-Laurent Akkar (Schlumberger)

Abstract

Differential Power Analysis (DPA) on smart-cards was introduced by Paul Kocher [11] in 1998. Since, many countermeasures have been introduced to protect cryptographic algorithms from DPA attacks. Unfortunately these features are known not to be efficient against high order DPA (even of second order). In these paper we will first describe new specialized first order attack and remind how are working high order DPA attacks. Then we will show how these attacks can be applied to two usual actual countermeasures. Eventually we will present a method of protection (and apply it to the DES) which seems to be secure against any order DPA type attacks. The figures of a real implementation of this method will be given too.

Keywords: Smart-cards, DES, Power analysis, High-Order DPA

1 Introduction

The framework of Differential Power Analysis (also known as DPA) was introduced by P. Kocher, B. Jun and J. Jaffe in 1998 ([11]) and subsequently published in 1999 ([12]). The initial focus was on symmetrical cryptosystems such as DES (see [11, 14, 1]) and the AES candidates (see [3, 4, 7]), but public key cryptosystems have since also been shown to be also vulnerable to the DPA attacks (see [15, 6, 9, 10, 16]).

Two main families of countermeasures against DPA are known:

- In [9, 10], L. Goubin and J. Patarin described a generic countermeasure consisting in splitting all the intermediate variables, using the secret sharing principle. This *duplication method* was also proposed shortly after by S. Chari *et al.* in [4] and [5].
- In [2], M.-L. Akkar and C. Giraud introduced the *transformed masking method*, an alternative countermeasure to the DPA. The basic idea is to perform all the computation such that all the data are XORed with a random mask. Moreover, the tables (*e.g.* the DES S-Boxes) are modified such that the output of a round is masked by the same mask as the input.

Both these methods have been proven secure against the initial DPA attacks, and are now widely used in real life implementations of many algorithms. However, they do not take into

consideration more elaborated attacks called "High-order DPA". These attacks, as described in [11] by P. Kocher or in [13] by T. Messerges, consist in studying correlations between the secret data and *several* points of the electric consumption curves (instead of *single* points for the basic DPA attack).
In what follows, we study the impact of the High-order DPA attacks on both countermeasures mentioned above. Moreover, we describe new secure ways of implementing a whole class of algorithms (including DES) against these new attacks.

The paper is organized as follows:

- In section 2, we recall three basic notions: the (high-order) differential power analysis, the duplication method and the transformed masking method.
- In Section 3, we study "duplication method" and show that an implementation of DES (or AES), which splits all the variables into n sub-variables is still vulnerable to an n-th order DPA attack. Section 3.1 gives the general principle of the attack and section 3.2 discusses practical aspects.
- Section 4 is devoted to the analysis of the "transformed masking" (see [2]). For such an implementation of DES, section 4.1 describes how a "second order" DPA can work. A new variant we call the "superposition attack" is also presented. In section 4.2, we show that an AES (=Rijndael) implementation protected with the "transformed masking" method can also be attacked, either by second order DPA, or by the "Zero problem" attack.
- Section 5 presents our new generic countermeasure: the "unique masking method". We illustrate it on the particular case of DES. In 5.1, we explain the main idea of "unique mask". In 5.2, we apply it to the full protection of a DES implementation. The security of this implementation against n-th order DPA attacks is investigated in sections 5.3 and 5.4.
- Section 6 focuses on the problem of securely constructing the modified S-Boxes used in our new countermeasure. The details of the algorithm are presented, together with practical impacts on the amount of time and memory needed.
- In Section 7, we give our conclusions.

2 Background

2.1 The (High-Order) Differential Power Analysis

In basic DPA attack (see [11, 12], or [8]), also known as first-order DPA (or DPA when there is no risk of confusion), the attacker records the power consumption signals and compute statistical properties of the signal for each individual instant of the computation. This attack does not require any knowledge about the individual electric consumption of each instruction, nor about the position in time of each of these instructions. It only relies on the following fundamental hypothesis (quoted from [10]):

Fundamental hypothesis (order 1): *There exists an intermediate variable, that appears during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for this variable.*

In this paper, we consider the so-called *High-Order Differential Power Analysis* attacks (HODPA), which generalize the first-order DPA: the attacker now compute statistical correlations between the electrical consumptions considered at several instants. More precisely, an "n-th order" DPA

attack takes into account n values of the consumption signal, which correspond to n intermediate values occurring during the computation.
These attacks now rely on the following fundamental hypothesis (in the spirit of [10]):
Fundamental hypothesis (order n): *There exists a set of n intermediate variables, that appear during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for a known function of these n variables.*

2.2 The "Duplication" Method

The "duplication method" was initially suggested by L. Goubin and J. Patarin in [9], and studied further in [4, 10, 5]. It basically consists in splitting the data (manipulated during the computation) into several parts, using a secret sharing scheme, and computing a modified algorithm on each part to recombine the final result at the end. For example, a way of splitting X into two parts can consist in choosing a random R and splitting X into $(X \oplus R)$ and R.

2.3 The "Transformed Masking" Method

The "Transformed Masking" Method was introduced in [2] by M.-L. Akkar and C. Giraud. The basic idea is to perform all the computation that all the data are XORed with a random mask. By using suitably modified tables (for instance S-Boxes in the case of DES), it is possible to have the output of a round masked by exactly the same mask that masked the input. The computation is thus divided into two main steps: the first one consists in generating the modified tables and the second one consists in applying the usual computation using these modified tables (the initial input being masked before starting the computation and the final output being unmasked after the computation).

3 Attack on the Duplication Method

3.1 Example: Second Order DPA on DES

In what follows, we suppose that two bits b_1 and b_2, appearing during the computation, are such that $b_1 \oplus b_2$ equals the value b of the first output of the first S-Box in the first DES round. The attacker performs the following steps:

1. Record the consumption curves C_i corresponding to N different inputs E_i $(1 \leq i \leq N)$. For instance $N = 1000$.
2. The attacker guesses the interval δ between the instant corresponding to the treatment of b_1 and the instant corresponding to the treatment of b_2. Each curve C_i is then replaced by $C_{i,\delta}$, which is the difference between C_i and (C_i translated by δ). He then computes the mean curve CM_δ of the N curves $C_{i,\delta}$.
3. The attacker guesses the 6 bits of the key on which the value of b depends. From these 6 key bits, he computes for each E_i the expected value for b. he then computes the mean curve CM'$_\delta$ of all the $C_{i,\delta}$ such that the expected b equals 0, and CM"$_\delta$ the mean curve of all the other $C_{i,\delta}$
4. If CM'$_\delta$ and CM"$_\delta$ do not show any appreciable difference, go back to 3 with another choice for the 6 key bits.
5. If no choice for the 6 key bits was satisfactory, go back to 2, with another choice for δ.

6. Iterate the steps 2, 3, 4, 5 with two bits whose "exclusive-or" comes from the second S-Box, the third S-Box, ..., until the eighth S-Box.
7. Find the 8 remaining key bits by exhaustive search.

3.2 The Attack in Practice

As specified in the original paper [10], it is clear that the n-th duplication is vulnerable to an n-th order DPA attack. An important point is to notice that if the method is not carefully implemented, it will be easily detected on the consumption curve, just by identifying n repetitive parts in the calculus. In this case, it would be easy for the attacker to just superpose the different parts of the curves (in a constant, or proportional to $\log(n)$, time, but not exponential in n).
Moreover, in certain scenarios, the attacker has full access to the very details of the implementation. In particular, for high-level security certifications (ITSEC, Common Criteria), it is assumed that the attacker knows the contents of the smartcard ROM.

4 Attack on the Transformed Masking Method

4.1 DES: Second Order DPA

4.1.1 Usual Second Order DPA:

For the DES algorithm, the input of a round is masked with a 64 bits value $R = R_{0-31}||R_{32-63}$ divided in two independent masks of 32 bits each. The modified S-boxes S' are the following (where S are the original ones).

$$S'(X) = S(X \oplus EP(R_{32-63})) \oplus P^{-1}(R_{0-31} \oplus R_{32-64})$$

Where EP represents the Expansion Permutation, and P^{-1} the inverse of the P permutation after the S-Boxes. We can see that using this formula the output mask of the value at the end of a DES round is nearly R. To get exactly the R masked value, the left part of the value has to be remasked with $R_{0-31} \oplus R_{32-64}$.
It is clear, like noticed in the article, than this countermeasure is subject to a second-order DPA attack. Indeed, the real output of the S-boxes is correlated to the masked value and the value R; so getting the electrical trace of these two values one can combine them and get a trace on which will work a classical DPA attack. In order to perform efficiently such an attack, without need of n^2 point like in the general attack, the attacker should get precise information about the implementation of the algorithm: he should know precisely where the interesting values are manipulated.

4.1.2 Superposition Attack:

In this section we will present a new kind of DPA attack. In theory it is a second-order DPA attack; but in practice it is nearly as simple as an usual DPA attack. The idea is the following: in a second order DPA the most difficult thing is to localize the time where the precise needed values are manipulated. On the contrary localizing a whole DES round is often quite easy. So instead of correlating precise part of the consumption traces we will just correlate the whole trace of the first and the last round. With these method one can notices than at one moment we will have the trace consumption T of the following value which is the output S-Boxes values:

$$\begin{array}{rcccc} T & = & (S'(E(R_{15}) \oplus K_{16}) \oplus R') & \oplus & (S'(E(R_1) \oplus K_1) \oplus R') \\ & = & S'(E(R_{15}) \oplus K_{16}) & \oplus & S'(E(R_1) \oplus K_1) \end{array}$$

Where R' is the right part of the mask permutated by the expansion permutation. One can notice that the T value does not depend of the random masking value and than R_1 and R_{15} [1] are often known. Considering this, it is easy to sea that performing a guess on the 2×6 bits of the subkey of the first and last round, it is possible to guess the XORed value of the output of the S-Boxes of the first and last round. After that once can perform an usual DPA-type attack attacker and find the values of the different sub-keys of K_1 and K_{16}. Due to redundancy of the key-bits one can moreover check the coherency of the results: indeed with such an attack one will find $2 \times 6 \times 8 = 96 >> 56$ bits for the key. The detailed algorithm is the following:

- Correlate (usually an addition or subtraction of the curves) the first and last round traces.
- For All the messages M, For the S-box $j = 1..8$
- For k=0 to 63, For l=0 to 63
- Separate the Messages , considering one bit of the XOR values of the output of the j^{th} Sbox (round 1 and 16) for the message M considering that the subkey of the S-Box j of the first round is k, and the subkey of the S-Box j of the last round is l.
- Average and subtract the separated curves.
- Choose the value k,l where the greatest peak appear.
- Check the coherency of the keybits found.

A cautionary look of the attack could convince the reader that any error of one bit on the guess of K_1 or K_{16} eliminate all the correlation. Comparing to an usual second order DPA attack, even if this attack require the analyze of $2^{12} = 4096$ possibilities, it has the advantage not to need a precise knowing of the code. And from a complexity point of view it increases by a constant factor ($2^6 = 64$) the amount of time and memory needed for the attacker and not by a linear factor.

4.1.3 Conclusion:

The superposition attack, even if it is a theoretical second order attack is very efficient in practice. Therefore to use transformed masking method, one must use different masks at each step of the algorithm. This idea have been developed and adapted to produce the protection described in this article.

4.2 AES

For the AES, the countermeasure is nearly the same than in DES. The only difference is that no transformed tables are used for the non-linear part of the AES (the inversion in the field GF(256)) but the same table with a multiplicative mask. The distributivity of the multiplication over XOR (addition in the field) is used. So from an additive mask it is easy, without unmasking the value, to switch to a multiplicative one, to go through the Sboxes and to get back to an the mask.

1. R_{15} can be deduced from the output applying the inverse of the final permutation

4.2.1 Usual Second Order DPA:

For AES it is exactly the same than in the DES transformed masking method. Correlating the masked value and the mask allow an effective attack against this method.

4.2.2 The "Zero" problem:

Because a multiplicative mask is used during the inversion, one can see that if the inverted value is zero -and this value just depend of 8 bits of the key in the first and last round- then whatever is the masking value, the inverted value will be unmasked. Therefore if someone is able to detect in the consumption trace that the value is zero instead of a random masked value, one will be able to break such an implementation. Of course probabilistic tools such as variance analysis are devoted to such analysis.

4.2.3 Superposition Method:

As in the DES, one can say that using the same superposition method it would be possible to find the key 16 bits by 16 bits superposing the first and last round of AES because these are using the same mask. Unfortunately after the last round a last subkey is added to the output of the round. So the attacker need at least to guess 8 more bits of the key. It increase the attacker amount of work to 24 bits for each Sbox. In theory it is not a quadratic attack in the number of samples but in practice it is not so easy to perform more than 16 billions manipulation of the curve for each tables and each message.

4.2.4 Conclusion:

Judging by these attacks we can consider that the adaptive mask countermeasure on AES is not efficient even against some simpler attack than second order ones.

5 Unique Masking Method Principle

We have seen that the actual countermeasure against DPA are intrinsically vulnerable to high order DPA. Often the order of vulnerability is two, and even when it is theoretically more; practically it is one or two. In the next section we will present a method to protect the DES that seem to be efficient against any order DPA attacks. We will first describe the elementary steps of the method for after see how to construct a complete secure DES and why it seems to be secure.

5.1 Masked Rounds

Given any 32 bits value α we will define two new functions $\tilde{S}_1$ and $\tilde{S}_2$ based on the Sboxes function S.

$$\begin{cases} \forall x \in \{0,1\}^{48} & \tilde{S}_1(x) = S(x \oplus E(\alpha)) \\ \forall x \in \{0,1\}^{48} & \tilde{S}_2(x) = S(x) \oplus P^{-1}(\alpha) \end{cases}$$

where E is the expansion permutation and P^{-1} is the inverse of the permutation after the Sboxes. We define f_{K_i} to be the composition of E, the XOR of the i^{th} round subkey K_i the Sboxes and the permutation P. We then define $\tilde{f}_{1,K_i}$ and $\tilde{f}_{2,K_i}$ by replacing S by $\tilde{S}_1$ and $\tilde{S}_2$ in f.

Remark We can see that $\tilde{f}_1$ gives an unmasked value from a α-masked value and that, $\tilde{f}_2$ gives a α-masked result from an unmasked one.

Using the function f, $\tilde{f}_1$ and $\tilde{f}_2$ one can obtain 5 different rounds using masked/unmasked values. The figure 1 represents these five different rounds. The plain fill represents the unmasked value and the dashed fill represents masked values.

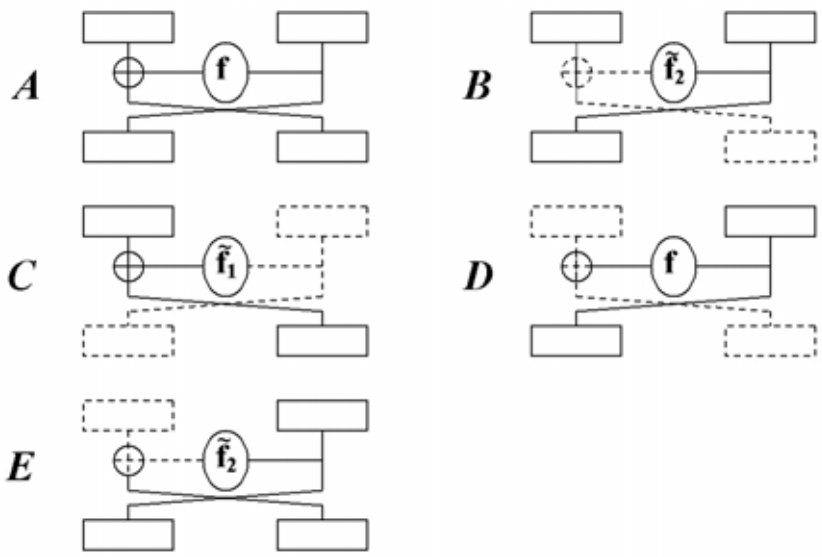


FIG. 1 – *Masked rounds of DES*

The following automata (cf fig. 2) shows how these rounds are compatible with each other. The input states are the rounds where the input is unmasked (A and B) and the output states are the one where the output of the rounds are unmasked (A and E).

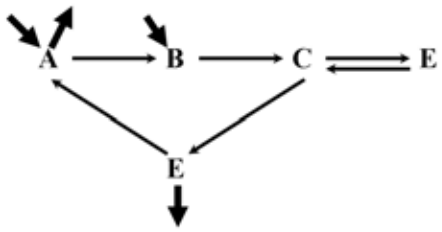


FIG. 2 – *Combination of the rounds*

5.2 Complete DES with Masked Rounds

It is easy to see that one could obtain a 16 round complete DES with these requirements. $IP - BCDCDCEBCDCDCDCE - FP$ represents a correct example (IP represents the initial permutation of DES and FP the final one).

5.3 Security Requirements

In all this section we will consider that the modified Sboxes are already constructed and that the mask α changes at each DES computation.

The first step is to analyze in the DES of how many key bits depends the bits of the data at each round. This simple analyze is summarized in the figure 3. We have also considered that the clear and the cipher were known, explaining the symmetry of the figure.

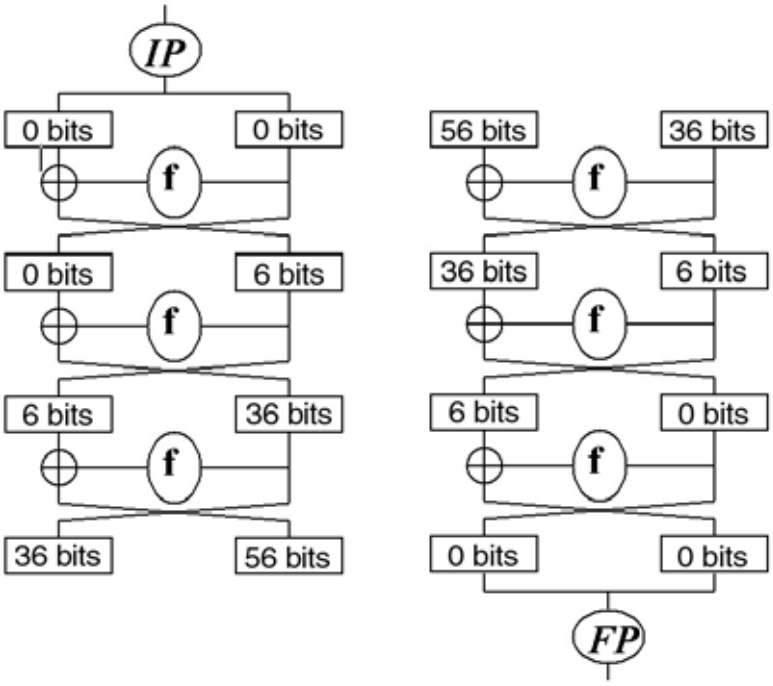


FIG. 3 – *Number of key bits / bits of data*

To get a correct security we have considered that the critical data are the one where the bits are dependant of less than 36^2 bits of the key. So we can see that only two parts have to be protected: the one connecting R_2 and L_3 and the one connecting R_{15} and L_{16}. We define as usual L_i (respectively R_i) as the left part (respectively the right part) of the message at the end of the i^{th} round. Of course the one depending of none bits of the keys have not to be protected.
Therefore these values must be masked and oblige the first three rounds to be of the form:

$$BCD \text{ or } BCE$$

The last three rounds must be of the form:

$$BCE \text{ or } DCE$$

Taking in account these imperatives

$$IP - BCDCDCEBCDCDCDCE - FP$$

is -for example- a good combination.

2. If we consider that a curve contains 128 8 bits-samples, 36 bits represents an amount of 2 Tb of memory needed

5.4 Resistance to DPA

5.4.1 Classical DPA:

This countermeasure clearly protect the DES against DPA of order one. Indeed all the value depending of less than 36 bits of the key are masked by a random mask which is used only once.

5.4.2 Enhanced Attacks:

First we have to notice that this countermeasure is vulnerable against the superposition method guessing 12 bits of the key. Indeed the same mask is used in the first and last round of the DES. So to counteract this attack we will from know consider that there's two different masks α_1 and α_2 which are used in the first and last round of DES. It is easy to see that the proposed combination of round permit at the 7^{th} and 8^{th} round to switch from α_1 to α_2 because of the structure of E-round/B-round which leave their output/input unmasked. With evident notations we can get the following example of DES:

$$IP - B_{\alpha_1} C_{\alpha_1} D_{\alpha_1} C_{\alpha_1} D_{\alpha_1} C_{\alpha_1} E_{\alpha_1} B_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} D_{\alpha_2} C_{\alpha_2} E_{\alpha_2} - FP$$

Let now consider n-th order DPA attack. The idea is to correlate several value to get the consumption of an important value. For us an important value is consider to be a value which could be guessed with less than 36 bit of the key. But we have seen that all these value are masked. Moreover the mask appear only once in all the calculus[3], so even with high order correlation it is impossible to get any information about the masked value

5.5 Variation

- If we want the mask never to appear several times (even on values depending on more than 36 bits of the key) one can use the following combination instead of the proposed one:

$$IP - B_{\alpha_1} C_{\alpha_1} E_{\alpha_1} AAAAAAAAAAB_{\alpha_2} C_{\alpha_2} E_{\alpha_2} - FP$$

- For paranoid people it is even possible to add two new masks and to mask every values depending on less than 56 bits of the key.
- This method is modular: if one uses a protocol where the input or the output are not known, one can eliminate the associated mask.

6 Effective Construction of the Modified S-Boxes

In this section algorithms will be described using pseudo c-code.

6.1 Principle

It is easy to see that the following operation must be performed securely in order to construct the Sboxes $\tilde{S}_1$.

- Generate a random α.
- Perform a permutation on α (permutation P^{-1}).

3. We remind the reader that we have considered that the tables are already constructed. This part will be analyzed in the next section

– XOR a value ($P^{-1}(\alpha)$) to a table.

For the construction of $\tilde{S}_2$, we need to:

– Recuperate α because it is the same than in $\tilde{S}_1$.
– Permutate it ($E(\alpha)$).
– XOR to a table containing (1..63).

Of course securely means that all these operations must be done without giving any information about the consumption of α at any order (1,2 ...).

6.2 Generation of a Random Number: for example 64 bits

We consider that we have access to a 64 bytes array t and to a random generator (for example a generator of bytes). We can proceed like the following:

– for(i=0..63) { t[i]=rand()%2 }
– for(i=0..63) { swap(t[i],t[rand%64]) }

With this this method one can see that we get in memory a 64 bits random value and that an attacker just know the hamming weight of α (if he can perform an SPA attack). For this we have considered that the attacker could not in one shot determinate what is the array entry addressed when we swap the entries ; hypothesis which looks quite reasonable.

Variant 1: To save time and memory we can imagine the following method which is much faster and does not look too weak. We will get 16 4-bits values in a 16 bytes array:

– for(i=0..16) { t[i]=rand() }
– for(i=0..16) { swap(t[i] AND 7,t[rand%16] AND 7) }

Indeed we can consider that the 4 bits of high weight will strongly influence the consumption.

Variant 2: This other method produces and 8 bytes random array. It is faster but less secure.

– for(i=0..8) { t[i]=rand() }
– for(i=0..16) { t[rand()%8] XOR= rand() }

6.3 Permutation

Classically it can be done bit per bit randomly. Against it only allow the attacker to get the hamming weight of the permuted value.
To speed up and have a memory gain, one could perform randomly the permutation quartet per quartet or even byte per byte. An idea could be to add some dummy values and perform the permutation. The dummy values would just not be considered after the permutation time.

6.4 XOR

Here a general method could be to XOR the value bit per bit in a random order to the table. Once again many compromise are possible to perform the XOR: do it byte per byte, add dummy values ...

6.5 Practical Considerations

The usual Sboxes are using 256 bytes. We need them but they could be stored in ROM. For the additional tables we need to store them in RAM. In the normal security method (two masks α_1 and α_2) we need to store 4 new tables. So the total requirement in RAM is of 1024 bytes.

We have seen that the construction of the Sboxes could be performed quite securely. Of course the most secure method is very slow and will really slow down the DES execution and use a lot of memory. The idea was just to show that it was theoretically possible to build the table without filtering any information [4] with a reasonable model of security [5] But we have also seen that it is possible to increase the speed and decrease the memory without loosing too much security.

Lets now have a look at how could be applied our countermeasure to the AES algorithm. Due to the higher number of tables (more than 16 instead of 8) and because they are bigger (8→8 bits instead of 6→4) compared to DES, our countermeasure would require about 8 Kb (or 16 Kb for a high security level) of RAM, a size which is too big for usual smart-cards. Some simplifications -which would unfortunately decrease the level of security- are therefore necessary to apply our countermeasure to AES implementation.

7 Real implementation on the DES algorithm

A real implementation of this method have been completed on an ST19 component. It includes the following features described in the last sections:

- SPA protections: Randomization and masking method for the permutations and the manipulation of the key (permutations, Sboxes access...).
- DPA protection: HO-DPA Protection of the first and last three rounds of the DES.
- S-Boxes constructions is done bit per bit with bit per bit randomization while computing the masking value.
- DFA Protection: multiple computation, coherence checking ...

With all this features we get an implementation with:

- 3 KB of ROM code.
- 81 bytes of RAM and 668 bytes of extended RAM
- An execution time of 38 ms at 10 Mhz.

This implementation have been submitted to our internal SPA/DPA/DFA laboratory which have tried to attack it without success.

8 Conclusion

Opposed to other proposed countermeasures, the unique masking method presents the following advantages:

- It is actually the only protection known against high-order DPA.
- The core of the DES is exactly the same than ordinary; so one can use with very light modification its implementation just adding the Sbox generation routine.
- The important values are masked with a unique mask which never appear in the DES computation. For example with the transformed masking method the mask were appearing often (for a first mask at the whole beginning and at each rounds). Here one do not even have to mask the entry or unmask the output.
- The only part where the mask is appearing (but it could be randomly and bit per bit) does not depend neither of the key and neither of the message. Therefore the security is totally focused at this point.

4. But the hamming weight of the value
5. The attacker is not able to read the exact memory access in one shot.

- This method is very flexible and modular without important changes in the code: it could even be a compilation parameter to determine which level of security one wants.
- A real implementation have been performed proving the feasibility of this countermeasure in reasonable time (less than 40ms with full protections).

References

[1] M.-L. Akkar, R. Bevan, P. Dischamp, D. Moyart, *Power Analysis: What is now Possible*. In Proceedings of ASIACRYPT'2000, LNCS 1976, pp. 489-502, Springer-Verlag, 2000.

[2] M.-L. Akkar, C. Giraud, *An Implementation of DES and AES Secure against Some Attacks*. In Proceedings of CHES'2001, LNCS 2162, pp. 309-318, Springer-Verlag, 2001.

[3] E. Biham, A. Shamir, *Power Analysis of the Key Scheduling of the AES Candidates*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`

[4] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`

[5] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power-Analysis Attacks*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 398-412, Springer-Verlag, 1999.

[6] J.-S. Coron, *Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems*. In Proceedings of CHES'99, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.

[7] J. Daemen, V. Rijmen, *Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`

[8] J. Daemen, M. Peters, G. Van Assche, *Bitslice Ciphers and Power Analysis Attacks*. In Proceedings of FSE'2000, LNCS 1978, Springer-Verlag, 2000.

[9] L. Goubin, J. Patarin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. European Patent, SchlumbergerSema, February 4th, 1999, Publication Number: 2789535.

[10] L. Goubin, J. Patarin, *DES and Differential Power Analysis – The Duplication Method*. In Proceedings of CHES'99, LNCS 1717, pp. 158-172, Springer-Verlag, 1999.

[11] P. Kocher, J. Jaffe, B. Jun, *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Available from `http://www.cryptography.com/dpa/technical/index.html`

[12] P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.

[13] T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant software*. In Proceedings of CHES'2000, LNCS 1965, pp. 238-251, Springer-Verlag, 2000.

[14] T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Investigations of Power Analysis Attacks on Smartcards*. In Proceedings of the USENIX Workshop on Smartcard Technology, pp. 151-161, May 1999. Available from `http://www.eecs.uic.edu/~tmesserg/papers.html`

[15] T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*. In Proceedings of CHES'99, LNCS 1717, pp. 144-157, Springer-Verlag, 1999.

[16] K. Okeya, K. Sakurai, *Power Analysis Breaks Elliptic Curve Cryptosystem even Secure against the Timing Attack*. In Proceedings of INDOCRYPT'2000, LNCS 1977, pp. 178-190, Springer-Verlag, 2000.

Résumé

La carte à microprocesseur est un cadre privilégié d'application de la cryptologie, qui met en jeu d'une manière toute particulière les interactions entre la théorie de la complexité, et le comportement physique des circuits électroniques.

Une première partie de ce mémoire s'attache à l'étude de nouveaux schémas cryptographiques particulièrement bien adaptés aux contraintes de mémoire et de temps spécifiques des cartes à microprocesseur. La proposition et l'analyse de nouvelles hypothèses calculatoires, utilisant la notion de polynômes multivariables, la mise en place de méthodes générales de cryptanalyse, ont permis d'aboutir à des algorithmes de signature électronique particulièrement intéressants. Le premier, QUARTZ, permet d'obtenir des signatures très courtes, utilisables dans des contextes tels que la protection des droits pour les logiciels, ou encore les cartes d'identité. L'autre SFLASH présente les meilleures performances connues sur une carte à microprocesseur bas de gamme.

Dans la seconde partie, on s'intéresse à la vulnérabilité propre aux systèmes de calcul cryptographique embarqués, face à des attaques physiques. On propose ainsi une analyse précise des conditions rendant possibles les attaques par analyse physique de la consommation électrique (ou des rayonnements électromagnétique), notamment dans le cas d'algorithmes symétriques (DES, AES) ou asymétriques (RSA, ECDSA) classiques. Une compréhension des principes fondamentaux de ces attaques permet de mettre en place des contre-mesures qui s'appliquent à une large classe d'algorithmes cryptographiques. La méthode de duplication, la méthode du masque unique, ainsi qu'une technique générale de conversion entre des représentations booléennes et arithmétiques des données dans le microprocesseur, ouvrent la voie à un traitement systématique, et même automatisable dans une certaine mesure, des attaques par analyse physique.

Abstract

Smart card is a privileged context of application for cryptology. It involves in a very specific way the interactions between complexity theory and physical behaviour of the electronic circuits.

The first part of this thesis deals with the study of new cryptographic schemes, particularly well suited to the memory and time constraints of smart cards. The proposal and analysis of new algorithmic assumptions, using the notion of multivariate polynomials, the setting of general cryptanalytic methods, lead to particularly interesting digital signature algorithms. The first one, QUARTZ, gives very short signatures, that can be used in contexts such as digital right management, or identity cards. The other one, SFLASH, presents the best performances today on a low-cost smart card.

In the second part, we investigate the specific vulnerability of cryptographic embedded systems, facing physical attacks. We propose a precise analysis of the conditions that enable power analysis attacks (or electromagnetic attacks), particularly for the case of classical symmetric (DES, AES) or asymmetric algorithms (RSA, ECDSA). Thanks to a deep understanding of the fundamental underlying principles of these attacks, we are able to define counter-mesures that can be applied to a large class of cryptographic algorithms. The duplication method, together with the unique masking method and a general conversion technique between boolean and arithmetic representations, opens the way to a systematic treatment of power analysis attacks, which is even automatable to a certain extent.