Solving Underdefined Systems of Multivariate Quadratic Equations

Nicolas Courtois[1], Louis Goubin[1], Willi Meier[2] and Jean-Daniel Tacier[2]

[1] CP8 Crypto Lab, SchlumbergerSema, 36-38 rue de la Princesse
BP 45, F-78430 Louveciennes Cedex, France
[2] FH Aargau, CH-5210 Windisch, Switzerland

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 ef-

ficiency 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

$$G_l(x_1, ..., x_n) = \sum_{i=1}^{n} \sum_{j \geq i}^{n} \alpha_{ijl} x_i x_j + \sum_{k=1}^{n} \gamma_{kl} x_k, \quad l = 1, ..., m.$$

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, $i = 1, \ldots, n$ 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, $Q = f_1 f_2 + f_3 f_4 + \ldots$, where the f_i's, $i = 1, \ldots, n$ denote appropriate linear forms. Then*

a) the coefficient vectors $\underline{f}_i$, $i = 1, \ldots, n$, 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 $x_1, \ldots, x_n$ 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:

[1] In time polynomial in the size of the initial system that can be exponential.

$$\begin{cases} n \geq m(m+1) & \text{if } \mathbf{F} \text{ has characteristic 2;} \\ n \geq \text{roughly } 2^{\frac{m}{7}}(m+1) & \text{if } \mathbf{F} \text{ has an odd characteristic.} \end{cases}$$

Let $(\mathcal{S})$ be the following system:

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

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

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 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 3.2 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 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

$$Q_2 = (\sum_{i=2}^n \alpha_i' g_i + b') g_2 + g_3 g_4 + ... + g_{n-1} g_n + q_2 = (g_3 + \alpha_4' g_2)(g_4 + \alpha_3' g_2) + \quad (2)$$

$$+ ... + (g_{n-1} + \alpha_n' g_2)(g_n + \alpha_{n-1}' g_2) + (\alpha_2' + \alpha_3' \alpha_4' + ... \alpha_{n-1}' \alpha_n') g_2^2 + b' g_2 + q_2,$$

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 $\alpha_i, \beta_j \in \mathbf{F}$, $i, j = 3, ..., n$,

$$\sum_{i=3}^n \alpha_i \underline{f}_i + \sum_{j=3}^n \beta_j \underline{g}_j = 0, \quad (3)$$

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.