DECIM^{v2} *

C. Berbain[1], O. Billet[1], A. Canteaut[2], N. Courtois[3], B. Debraize[3,4], H. Gilbert[1], L. Goubin[4], A. Gouget[5], L. Granboulan[6], C. Lauradoux[2], M. Minier[2], T. Pornin[7] and H. Sibert[5]

Abstract

DECIM is a hardware oriented stream cipher with 80-bit key and 64-bit IV which was submitted to the ECRYPT stream cipher project. The design of DECIM is based on both a nonlinear filter LFSR and an irregular decimation mechanism called the ABSG. As a consequence, DECIM is of low hardware complexity. Recently, Hongjun Wu and Bart Preneel pointed out two flaws in the stream cipher DECIM. The first flaw concerns the initialization stage and the second one, which is the more serious flaw, concerns the filter used in the keystream generation algorithm; the ABSG mechanism is not affected by these two flaws. In this paper, we propose a new version of DECIM, called DECIM^{v2}, which does not only appear to be more secure, but also has a lower hardware complexity than DECIM.

1 Introduction

DECIM [3] is a hardware oriented stream cipher submitted to the ECRYPT Stream Cipher Project [1]; we now call it DECIM^{v1}. It has been developed around the ABSG mechanism which provides a method for irregular decimation of pseudorandom sequences. The general running of DECIM (and also DECIM^{v2}) consists in generating a binary sequence $\mathbf{y}$ in a regular way from a Linear Feedback Shift Register (LFSR) which is filtered by a Boolean function. The sequence $\mathbf{y}$ is next filtered by the ABSG mechanism.

Recently, Hongjun Wu and Bart Preneel [4] found two flaws in the stream cipher DECIM^{v1}. The first flaw concerns the initialization stage, i.e. the computation of the initial inner state for starting the keystream generation. In a nutshell, the initialization mechanism of DECIM^{v1} works as follows.

[1]France Télécom Recherche et Développement, 38/40 rue du Général Leclerc, F-92794 Issy les Moulineaux cedex 9, {come.berbain,olivier.billet,henri.gilbert}@francetelecom.com

[2]INRIA-Rocquencourt, projet CODES, domaine de Voluceau, B.P. 105, F-78153 Le Chesnay cedex, {anne.canteaut,marine.minier,cedric.lauradoux}@inria.fr

[3]Axalto Smart Cards, 36-38, rue de la Princesse - B.P. 45, F-78431 Louveciennes cedex, {ncourtois,bdebraize}@axalto.com

[4]Laboratoire PRiSM, Université de Versailles, 45 avenue des Etats-Unis, F-78035 Versailles cedex, louis.goubin@prism.uvsq.fr

[5]France Télécom Recherche et Développement, 42 rue des Coutures, BP 6243, F-14066 Caen cedex, {aline.gouget,herve.sibert}@francetelecom.com

[6]Département d'Informatique, École Normale Supérieure, 45 rue d'Ulm, F-75230 Paris cedex 05, louis.granboulan@ens.fr

[7]Cryptolog International, 16-18 rue Vulpian, F-75013 Paris, thomas.pornin@cryptolog.com

*Work partially supported by the French Ministry of Research RNRT Project "X-CRYPT" and by the European Commission via ECRYPT network of excellence IST-2002-507932.

1. Filling of the LFSR from a 80-bit secret key and a 64-bit public IV.
2. 192 updates of the LFSR. One update consists of the three following steps:
   (a) Computation of the feedback value (in a nonlinear way);
   (b) Application of one among two permutations over 7 elements of the current LFSR state; the choice of the permutation is controlled by the output of the ABSG;
   (c) Shifting by one position of the LFSR.

The aim of the permutations is to provide high nonlinearity during the initialization stage. However, the side effect of the permutations is that a large number of elements of the LFSR (after the initial filling) may never be updated with a high probability during the initialization process. This flaw allowed Hongjun Wu and Bart Preneel to mount an efficient key recovery attack on DECIM^{v1}. For DECIM^{v2}, we propose a simpler and more secure initialization procedure than the one of DECIM^{v1} (in particular, the permutations involved in the initialization procedure of DECIM^{v1}, which imply a significant increase of the hardware cost, are removed in DECIM^{v2}).

The main flaw pointed out by Hongjun Wu and Bart Preneel [4] is in the keystream generation algorithm which is described in Figure 1. More precisely, the flaw is in the generation

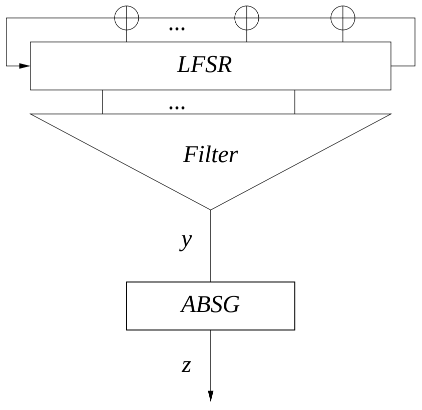

Figure 1: DECIM keystream generation

of the sequence **y** which is the output of the filter (the sequence **y** is next decimated by the ABSG mechanism). In a few words, this flaw is due to the fact that the sequence **y** is directly the output of a symmetric Boolean function which is not *correlation-immune of order* 1. There exists a correlation between the outputs of the function associated to two input vectors which have one element in common. By using this weakness, Hongjun Wu and Bart Preneel show a correlation between some bits of the keystream sequence and then they show that the keystream of DECIM^{v1} is heavily biased. For DECIM^{v2}, we propose a simpler and more secure filter than the one of DECIM^{v1} by choosing a filter which is correlation immune of order 1.

The outline of the paper is as follows. In Section 2, we give an overview of DECIM^{v2} and we describe the slight modifications between DECIM^{v1} and DECIM^{v2}. In Section 3, we provide a full description of DECIM^{v2}. In Section 4, we explain the design modifications. In Section 5, we discuss the hardware implementation. Finally, we conclude in Section 6.

2 Overview of Decim^{v2}

In accordance with the specification given by the Ecrypt stream cipher project, DECIM^{v2} takes as an input a 80-bit length secret key and a 64-bit length public initialization vector.

2.1 Keystream generation

The size of the inner state of DECIM^{v2} is unchanged, i.e. 192 bits. The keystream generation mechanism is described in Figure 2. The bits of the internal state of the LFSR are numbered from 0 to 191, and they are denoted by $(x_0, \ldots, x_{191})$. The sequence of the linear feedback values of the LFSR is denoted by $\mathbf{s} = (s_t)_{t \geq 0}$.

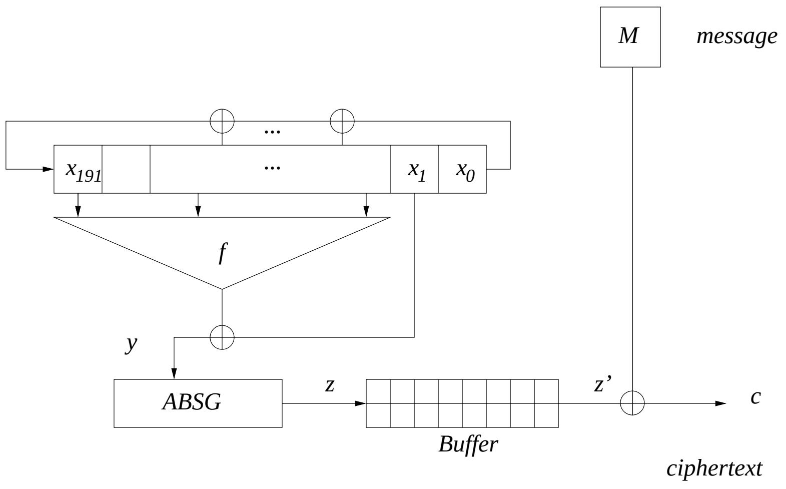

Figure 2: DECIM^{v2} keystream generation

The Boolean function f is a 13-variable quadratic symmetric function which is balanced. Let $x_{i_1}, \ldots, x_{i_{14}}$ denote the 14 initial internal state bits of the LFSR that are the inputs of the filter. The sequence $\mathbf{y}$ outputs by the filter is defined by:

$$y_t = f(s_{i_1+t}, \ldots, s_{i_{13}+t}) \oplus s_{i_{14}+t}$$

The ABSG takes as an input the sequence $\mathbf{y} = (y_t)_{t \geq 0}$. The sequence output by the ABSG is denoted by $\mathbf{z} = (z_t)_{t \geq 0}$. The buffer mechanism guarantees a constant throughput for the keystream; we choose a 32 bit-length buffer and the buffer outputs 1 bit for every 4 shifts by one position of the LFSR (see [3] for details).

Remark 1 *For the keystream generation, the gap between* DECIM^{v1} *and* DECIM^{v2} *is the choice of the filter. In* DECIM^{v1}*, the filter is a vectorial function defined by:*

$$F : \mathbb{F}_2^{14} \longrightarrow \mathbb{F}_2^2; \quad x_{i_1}, \ldots, x_{i_{14}} \mapsto (f(x_{i_1}, \ldots, x_{i_7}), f(x_{i_8}, \ldots, x_{i_{14}}))$$

where f is a 7-variable symmetric Boolean function which is balanced but which is not correlation immune of order 1*.*

2.2 Key/IV setup

The initial filling of the LFSR from the key and the initialization vector is modified in Decim^{v2} compared to DECIM^{v1} (see Section 3). The Key/IV setup mechanism consists in clocking $4 \times 192 = 768$ times the LFSR using the nonlinear feedback which is described in Figure 3.

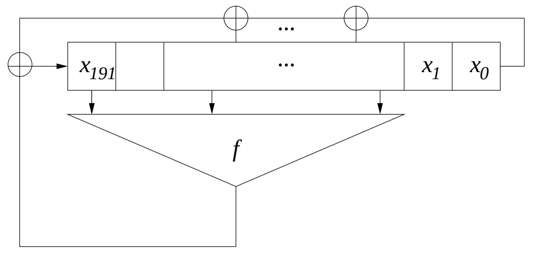

Figure 3: Key/IV setup mechanism

Remark 2 *For the initialization stage, the main differences between* DECIM^{v1} *and* DECIM^{v2} *are the filling of the LFSR which is changed, the deletion of the permutations and the choice of the filter. As a consequence, the number of clocks in the initialization stage increases from* 192 *up to* 768*.*

3 Specification

In this section, we describe each component of DECIM^{v2} and we describe the changes between DECIM^{v1} and DECIM^{v2}; we refer to [3] when no modification has been done.

3.1 The filtered LFSR

This section describes the filtered LFSR that generates the sequence $\mathbf{y}$ (the sequence $\mathbf{y}$ is the input of the ABSG mechanism).

The LFSR (unchanged). The underlying LFSR is a maximum-length LFSR of length 192 over $\mathbf{F}_2$. It is defined by the following primitive feedback polynomial:

$$P(X) = X^{192} + X^{189} + X^{188} + X^{169} + X^{156} + X^{155} + X^{132} + X^{131} + X^{94} + X^{77} + X^{46} + X^{17} + X^{16} + X^{5} + 1 \; .$$

The filter (changed). The filter function is the 14-variable Boolean function defined by:

$$F : \mathbb{F}_2^{14} \longrightarrow \mathbb{F}_2; \quad a_1, \ldots, a_{14} \mapsto f(a_1, \ldots, a_{13}) \oplus a_{14}$$

where f is the symmetric quadratic Boolean function defined by:

$$f(a_1, \ldots, a_{13}) = \bigoplus_{1 \le i < j \le 13} a_i a_j \bigoplus_{1 \le i \le 13} a_i$$

The tap positions of the filter are:

$$191 - 186 - 178 - 172 - 162 - 144 - 111 - 104 - 65 - 54 - 45 - 28 - 13 - 1$$

and the input of the ABSG at the stage t is:

$$y_t = f(s_{t+191}, s_{t+186}, s_{t+178}, s_{t+172}, s_{t+162}, s_{t+144}, s_{t+111}, s_{t+104}, s_{t+65}, s_{t+54}, s_{t+45}, s_{t+28}, s_{t+13}) \oplus s_{t+1}$$

3.2 Decimation (unchanged)

This part describes how the keystream sequence $\mathbf{z}$ is obtained from the sequence $\mathbf{y}$. The ABSG algorithm is given in Figure 4.

```
Input: $(y_0, y_1, \dots)$
Set: $i \leftarrow 0$; $j \leftarrow 0$;
Repeat the following steps:
   1.  $e \leftarrow y_i$, $z_j \leftarrow y_{i+1}$;
   2.  $i \leftarrow i + 1$;
   3.  while $(y_i = \overline{e})$ $i \leftarrow i + 1$;
   4.  $i \leftarrow i + 1$;
   5.  output $z_j$
   6.  $j \leftarrow j + 1$
```

Figure 4: ABSG Algorithm

3.3 Buffer mechanism (unchanged)

The rate of the ABSG mechanism is irregular and therefore we use a buffer in order to guarantee a constant throughput. We choose a buffer of length 32 and for every 4 bits that are input into the ABSG, the buffer is supposed to output one bit exactly. With these parameters, the probability that the buffer is empty while it has to output one bit is less than 2^{-89}.

If the ABSG outputs one bit when the buffer is full, then the newly computed bit is not added into the queue, i.e. it is dropped. Assuming that the initial inner state is computed (it is denoted by $z_0, \dots, z_{191}$), the ABSG mechanism starts at the beginning loop and the buffer is empty. The keystream generation process starts when the buffer is full.

3.4 Key/IV Setup

This subsection describes the computation of the initial inner state for starting the keystream generation. Notice that the ABSG mechanism is not used anymore during the initialization stage.

3.4.1 Initial filling of the LFSR (changed)

The secret key K is a 80-bit key denoted by $K = K_0, \dots, K_{79}$ and the initialization vector IV is a 64-bit IV denoted by $IV_0, \dots, IV_{63}$.

The initial filling of the LFSR is done as follows.

$$x_i = \begin{cases} K_i & 0 \le i \le 79 \\ K_{i-80} \oplus IV_{i-80} & 80 \le i \le 143 \\ K_{i-80} \oplus IV_{i-144} \oplus IV_{i-128} \oplus IV_{i-112} \oplus IV_{i-96} & 144 \le i \le 159 \\ IV_{i-160} \oplus IV_{i-128} \oplus 1 & 160 \le i \le 191 \end{cases}$$

The number of possible initial values of the LFSR state is $2^{80+64} = 2^{144}$.

3.4.2 Update of the LFSR state

The LFSR is clocked $4 \times 192 = 768$ times using a nonlinear feedback relation. Let y_t denote the output of f at time t and let lv_t denote the linear feedback value at time $t > 0$. Then, the value of x_{191} at time t is computed using the equation:

$$x_{191} = lv_t \oplus y_t \ .$$

Notice that there is no bit of the LFSR state output during this step.

4 Design rationale

The rationale behind the design of Decim^{v2} relies on the fact that the main ideas behind Decim^{v1}, namely, to filter and then decimate the output of an LFSR using the ABSG mechanism was in no way questioned. Thus, the core of DECIM^{v2} is a single Boolean function-based filtering, followed by an ABSG-based decimation.

4.1 Choice of the filter

The choice of the filter is very important since the filter must not introduce some weaknesses in the stream cipher (as it is the case for DECIM^{v1}). An important property for the filter is that the output of the filter must be uniformly distributed. In DECIM^{v1}, the 7-variable Boolean function f used in the filter is balanced, i.e., the value of f is uniformly distributed in $\{0, 1\}$ when the evaluation of f is done uniformly over $\{0, 1\}^7$.

DECIM^{v1} is a hardware-oriented stream cipher and the filter must have a low-cost hardware implementation. In DECIM^{v1}, the filter is a symmetric Boolean function f (i.e. the value of f only depends on the Hamming weight of the input) in order to reduce the hardware cost and the function f is balanced.

The attack given by Hongjun Wu and Bart Preneel [4] has shown that it is important to choose a Boolean function f which is *correlation-immune of order* 1, i.e. a function such that there is no correlation between the outputs of the function associated to two input vectors which have one element in common. Since the Boolean function f must also be balanced, that means that f must be 1-*resilient*. In DECIM^{v1}, the Boolean function is balanced but it is not 1-resilient.

The filter of DECIM^{v2} is constructed from a balanced 13-variable symmetric function (which is not correlation immune of order 1) and the whole filter F is a 1-resilient Boolean function.

4.2 Tap positions : filter and feedback polynomial

Assuming knowledge of the keystream $\mathbf{z}$, an attacker will have to guess some bits of the sequence $\mathbf{y}$ in order to attack the function f. The knowledge of the bits of $\mathbf{y}$ directly yields equations in the bits of the initial state of the LFSR. Thus, the number of monomials in the bits of the initial state of the LFSR that are involved in these equations has to be maximized. Moreover, this number has to grow quickly during the first clocks of the LFSR. This implies the following two conditions:

1. each difference between two positions of bits that are input to f should appear only once;
2. some inputs of f should be taken at positions near the one of the feedback bit (which means that some inputs should be leftmost on Figure 2).

Finally, the tap positions of the inputs of the Boolean function f and the inputs of the feedback relation should be independent.

4.3 Key/IV Setup

The components of the keystream generation are re-used for the key/IV setup; we do not introduce new components.

By using a 80-bit key and a 64-bit IV, the number of possible initial states is at most 2^{144} which is the case in DECIM^{v2} whereas the number of possible initial states is 2^{136} in DECIM^{v1}.

The first attack given in [4] exploits the effects of the permutations π_1 and π_2 used in the initialization process. Indeed, some bits of the LFSR are improperly updated. Then, the attack consists in tracing some bits during the initialization process. In DECIM^{v2}, the permutations are removed and the number of clocks of the register is increased in order to ensure that the nonlinearity of the initialization stage is sufficient.

5 Hardware implementation

The number of gates involved in an hardware implementation can be estimated as follows, based on the estimation for elementary components given in [2], i.e., 12 gates for a flip-flop, 2.5 gates for an XOR, 1.5 gates for an AND and 5 gates for a MUX.

Here, we have the following values for each component in the circuit:

- LFSR: 2339 gates corresponding to 192 flip-flops and 14 XORs (instead of 3334 gates for DECIM^{v1}).
- Filtering function: 86.5 gates corresponding to 6 Full Adders and 7 XORs (instead of 74 gates for DECIM^{v1}; details on the hardware implementation of quadratic symmetric functions are given in [3]).
- 1-input ABSG, as described in Figure 5: 67 gates corresponding to 2 MUX, 3 XORs, 1 AND, and 4 flip-flops.

Remark 3 *For the proposed hardware implementation, the main differences between* DECIM^{v1} *and* DECIM^{v2} *is that the LFSR has now to be clocked 4 times instead of 2 before outputting a bit.*

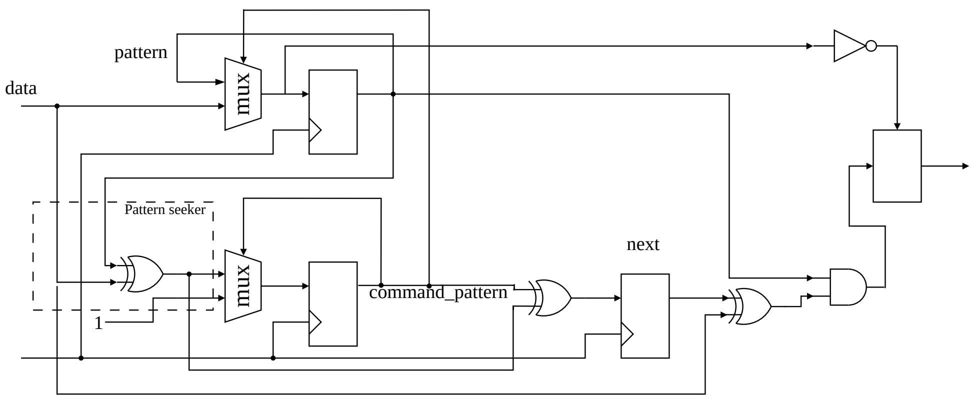

Figure 5: Hardware implementation of the ABSG

Moreover, the throughput of the generator can be doubled at a low implementation cost by using a simple speed-up mechanism. This can be done with a circuit which computes two feedback bits for the LFSR, simultaneously, as described in [3, Section 6.1]. This LFSR with doubled clock rate can be implemented within 192 flip-flops and 28 XORs. One additional copy of the filtering function is also required, and a 2-input ABSG mechanism must be used (see [3] for further details).

6 Conclusion

We have proposed a new stream cipher DECIM^{v2}. The design is based on the eStream proposal DECIM^{v1} and addresses all weaknesses found in the original construction. A complete description of DECIM^{v2} was given and the differences from DECIM^{v1} were discussed.

The stream cipher DECIM^{v2} is especially suitable for hardware applications with restricted resources such as limited storage or gate count. For applications requiring higher throughputs, speed-up mechanisms can be used to accelerate DECIM^{v2} at the expense of a higher hardware complexity.

References

[1] eStream, Stream cipher project of the European Network of Excellence in Cryptology ECRYPT. `http://www.ecrypt.eu.org/stream/`.

[2] L. Batina, J. Lano, S.B. Örs, B. Preneel, and I. Verbauwhede. Energy, perfomance, area versus security trade-offs for stream ciphers. In *The State of the Art of Stream Ciphers: Workshop Record*, pages 302–310, Brugge, Belgium, October 2004.

[3] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim – A new Stream Cipher for Hardware applications. In *ECRYPT Stream Cipher Project Report 2005/004*. Available at `http://www.ecrypt.eu.org/stream/`.

[4] Hongjun Wu and Bart Preneel. Cryptanalysis of Stream Cipher Decim. Available at `http://www.ecrypt.eu.org/stream/`.