Decim v2

Come Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Blandine Debraize, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, et al.

Decim^{v2}

Côme Berbain[1], Olivier Billet[1], Anne Canteaut[2], Nicolas Courtois[3], Blandine Debraize[4,5], Henri Gilbert[1], Louis Goubin[4,5], Aline Gouget[4], Louis Granboulan[6], Cédric Lauradoux[2], Marine Minier[7], Thomas Pornin[8] and Hervé Sibert[9].

[1] Orange Labs, France, {come.berbain,olivier.billet, henri.gilbert}@orange-ftgroup.com
[2] INRIA Rocquencourt, France, {anne.canteaut,cedric.lauradoux}@inria.fr
[3] University College of London, United Kingdom, n.courtois@ucl.ac.uk
[4] Gemalto, France, {blandine.debraize,aline.gouget}@gemalto.com
[5] Université de Versailles, France, louis.goubin@prism.uvsq.fr
[6] EADS, France, louis.granboulan@eads.net
[7] INSA Lyon, France, marine.minier@insa-lyon.fr
[8] Cryptolog International, France, thomas.pornin@cryptolog.com
[9] NXP Semiconductors, France, herve.sibert@nxp.com.

Abstract. In this paper, we present Decim^{v2}, a stream cipher hardware-oriented selected for the phase 3 of the ECRYPT stream cipher project eSTREAM. As required by the initial call for hardware-oriented stream cipher contribution, Decim^{v2} manages 80-bit secret keys and 64-bit public initialization vectors. The design of Decim^{v2} combines two filtering mechanisms: a nonlinear Boolean filter over a LFSR, followed by an irregular decimation mechanism called the ABSG. Since designers have been invited to demonstrate flexibility of their design by proposing variants that take 128-bit keys, we also present a 128-bit security version of Decim called Decim-128.

1 Introduction

Decim^{v2} is a hardware-oriented stream cipher selected for the phase 3 of the ECRYPT Stream Cipher Project [1]. Decim^{v2} is the tweaked version of the original submission Decim [3]. Decim^{v2} manages 80-bit secret keys and 64-bit public initialization as required by the initial eSTREAM call for contribution for the hardware-oriented profile. Decim^{v2} has been developed around the ABSG mechanism [9, 12] which provides a method for irregular decimation of pseudorandom sequences. The ABSG mechanism consists of compressing the input sequence in a very simple way and it operates a highly nonlinear transformation. Being an irregular decimation, it prevents algebraic attacks and some fast correlation attacks.

The general running of DECIM^{v2} consists first 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. Next, the sequence $\mathbf{y}$ is filtered by the ABSG mechanism. Wu and Preneel found two weaknesses [15] in the original design of DECIM that have been fixed in DECIM^{v2}. Note that the attacks presented in [15] do not question the main ideas behind DECIM, namely, to filter and then decimate the output of an LFSR using the ABSG mechanism. Since designers have been invited to demonstrate flexibility of their design by proposing variants that take 128-bit keys, we present a 128-bit security version of DECIM called DECIM-128.

The outline of the paper is as follows. In Section 2, we give an overview of DECIM^{v2} and we detail the differences between DECIM and DECIM^{v2}. In Section 3, we provide a full description of DECIM^{v2}. In Section 4, we explain the design rationale. In Section 5, we discuss the hardware implementation. Section 6 is dedicated to the description of DECIM-128. Finally, we conclude in Section 7.

2 Overview of Decim^{v2}

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

2.1 Keystream generation

The size of the inner state of DECIM^{v2} is 192 bits. The keystream generation mechanism is described in Figure 1. 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 Boolean function f is a 13-variable quadratic symmetric function which is balanced. The whole filter F is a 14-variable Boolean function. The output of the function F at time t is denoted by y_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.

2.2 Key/IV setup

The Key/IV setup mechanism consists in clocking $4 \times 192 = 768$ times the LFSR using the nonlinear feedback described in Figure 2.

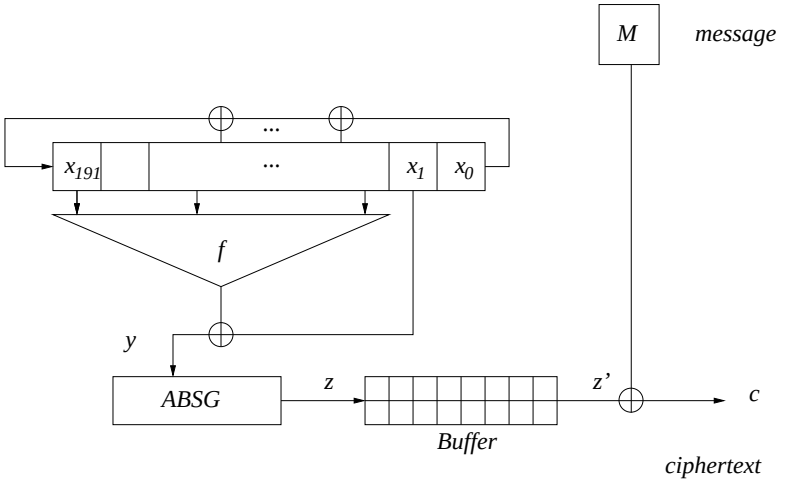

Fig. 1. DECIM^{v2} keystream generation

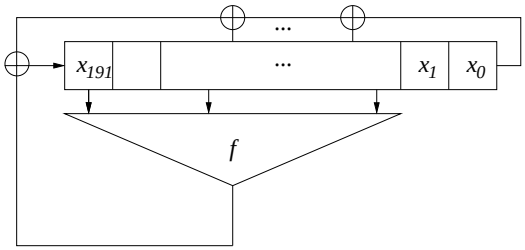

Fig. 2. Key/IV setup mechanism

2.3 Differences between Decim and Decim^{v2}

We do not recall in this paper the full description of the original design of DECIM (see [3] for details). However, we briefly describe the two flaws in DECIM found by Wu and Preneel [15], and we explain how DECIM^{v2} fixed these two weaknesses.

The first flaw lies in the initialization stage, i.e. the computation of the initial inner state for starting the keystream generation. In DECIM, at each clock of the initialization process, one of two 7-variables permutations π_1 and π_2 was applied over the internal state in order to break the linearity of the process faster. However, this mechanism could be exploited to retrieve the key. In DECIM^{v2}, we use an initialization procedure that is both simpler and more secure than the one of DECIM. In particular, the permutations are removed in DECIM^{v2} (we refer to [5] for more details).

Moreover, the number of clocks of the register during the initialization phase is increased in DECIM^{v2} in order to ensure that the nonlinearity of the initialization stage is sufficient.

The second flaw lies in the keystream generation algorithm. More precisely, there is a flaw in DECIM in the generation of the sequence $\mathbf{y}$ which is the output of the filter (the sequence $\mathbf{y}$ is next decimated by the ABSG mechanism). This flaw is due to the fact that the sequence $\mathbf{y}$ is directly the output of a symmetric Boolean function. Indeed, the outputs of the function associated to two input vectors which have one element in common are correlated. It was then shown in [11] that the filter criterion to avoid such correlation is the quasi-immunity criterion. The choice of the filter in DECIM^{v2} takes this design criterion into account.

3 Specification

In this section, we describe each component of DECIM^{v2}.

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. 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 sequence of the linear feedback values of the LFSR is denoted by $\mathbf{s} = (s_t)_{t \geq 0}$ and the recursion that corresponds to P for the LFSR is

$$s_{192+n} = s_{187+n} \oplus s_{176+n} \oplus s_{175+n} \oplus s_{146+n} \oplus s_{115+n} \oplus s_{98+n} \oplus s_{61+n} \oplus s_{60+n} \oplus s_{37+n} \oplus s_{36+n} \oplus s_{23+n} \oplus s_{4+n} \oplus s_{3+n} \oplus s_n \quad .$$

The filter. 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:

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

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

This part describes how the keystream sequence $\mathbf{z}$ is obtained from the sequence $\mathbf{y}$.

The action of the ABSG on $\mathbf{y}$ consists in splitting $\mathbf{y}$ into subsequences of the form $(\overline{b}, b^i, \overline{b})$, with $i \geq 0$ and $b \in \{0, 1\}$; $\overline{b}$ denotes the complement of b in $\{0, 1\}$. For every subsequence $(\overline{b}, b^i, \overline{b})$, the output bit is b for $i = 0$, and $\overline{b}$ otherwise. The ABSG algorithm is given in Figure 3.

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. $j \leftarrow j + 1$

Fig. 3. ABSG Algorithm

3.3 Buffer mechanism

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. The initial filling of the buffer is part of the initialization process detailed in 3.4.

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 during the initialization stage.

Initial filling of the LFSR. The secret key K is a 80-bit key denoted by $K = K_0, \ldots, K_{79}$ and the initialization vector IV is a 64-bit IV denoted by $IV_0, \ldots, IV_{63}$. The initial filling of the LFSR is done as follows.

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

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

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.

Initial filling of the buffer. After the previous step, the buffer has to be filled before starting keystream generation. In order to fill the buffer, we repeat the keystream generation process until the buffer is full. During this step, the buffer is not shifted. In particular, the buffer does not output any bit until it is completely filled. Then, the buffer is filled on average after 96 steps, and it is filled after 234 steps with probability bigger than $1 - 2^{-80}$. Thus, if a constant duration initialization process is required, one can choose to execute 234 steps and throw away the ABSG output bits when the buffer is full.

4 Design rationale

In this section, we give the rationale for every component of DECIM^{v2}.

4.1 The filtered LFSR

The LFSR. The length of the LFSR, which corresponds to the size of the internal state of the cipher, must be at least 160 in order to avoid time-memory-date trade-off attacks [13, 6]. Nevertheless, we add a security margin to the LFSR length in order to deal with a reduction of the size of the potential initial state due to the initialization procedure (see Section 4.3). Therefore, we choose a 192-bit LFSR.

The choice of the primitive feedback polynomial P must be made in accordance with the following constraints. The differences between two consecutive positions of the inputs of the feedback polynomial are pairwise coprime. Furthermore, the weight of P must be large enough in order to prevent the existence of sparse multiples with low degree that could be exploited in fast correlation attacks or in distinguishing attacks. However, we do not want the weight of P to be too large, in order to reduce both the overall computational time of the cipher and its hardware size.

The feedback polynomial has been chosen carefully, i.e. it has not low Hamming weight multiples at least for the first 2^{40} next degrees. However, we mention the possibility of a distinguishing attack similar to the distinguishing attack on the Self-Shrinking Generator given in [8].

The filtering function. An important property for the filter is that the output of the filter must be uniformly distributed. Moreover, the filtering function must satisfy some other well-known cryptographic properties. Indeed, it is expected to be far from an affine function (using the Hamming distance). Moreover, the attack presented by Wu and Preneel against DECIM [15] revealed that the filtering function must also fulfil the *quasi-immunity criterion* [11], which is a criterion weaker than being correlation-immune of order 1.

Since DECIM^{v2} is a hardware-oriented cipher, the Boolean filtering function must have a low-cost hardware implementation. In order to get an efficient computation of the function, the Boolean function f has been chosen to be symmetric, i.e. the value of f only depends on the Hamming weight of the input.

The symmetric Boolean functions that best fulfils the previous mentioned criteria are quadratic and have an odd number of input variables. The whole filter F of DECIM^{v2}, constructed from a balanced 13-variable symmetric function, is balanced and correlation-immune of order 1.

The 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 1).

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

4.2 Decimation

The ABSG mechanism was first presented at the ECRYPT Workshop State of the art of stream ciphers [9] and next published in [12]. The ABSG is a scheme that, like the Shrinking Generator (SG) [7] and the Self-Shrinking Generator (SSG) [14], provides a method for irregular decimation of pseudorandom sequences. The ABSG has the advantage on the one hand over the SG that it operates on a single input sequence instead of two and on the other hand over the SSG that it operates at a rate 1/3 instead of 1/4 (i.e. producing n bits of the output sequence requires on average $3n$ bits of the input sequence instead of $4n$ bits).

The best known attack on the ABSG filtering a single maximum-length LFSR [12, 10] is based on a guess of the most favorable case. Such a guess requires ℓ output bits in order to guess 2ℓ inputs bits. The guess is correct with probability $\frac{1}{2^\ell}$. In order to check the correctness of his guess, the attacker should try to solve the equations in the bits of the initial state of the LFSR that arise from the bits of $\mathbf{y}$ he has guessed. This attack can be used in order to reconstruct $2L$ consecutive bits of the sequence y from L consecutive bits of the sequence z; it costs $\mathcal{O}(2^{\frac{L}{2}})$ and requires $\mathcal{O}(L2^{\frac{L}{2}})$ bits of z.

Let $\Lambda(\mathbf{y})$ denote the linear complexity of $\mathbf{y}$. Then, the minimal length of a linear feedback shift register which generates the sequence $\mathbf{y}$ is $\Lambda(\mathbf{y})$. The previous attack can be used to reconstruct the initial state of the equivalent LFSR that generates the sequence $\mathbf{y}$. Then, this attack costs $\mathcal{O}(2^{\frac{\Lambda(\mathbf{y})}{2}})$ to recover $\Lambda(\mathbf{y})$ consecutive bits of $\mathbf{y}$.

We have checked that the linear complexity of $\mathbf{y}$ is the best linear complexity expected according to the choice of the Boolean function and the primitive polynomial, that is, $\Lambda(\mathbf{y}) = 18528$.

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}. The key schedule includes a non-linear feedback mechanism that is repeated L times, where L is the length of the register. Thus, in order to deal with the reduction of the potential internal state of the register during this phase, and considering that this non-linear feedback behaves randomly, we chose $L = 192$ to ensure that the final internal state is at least twice the key length, that is, 160.

4.4 The buffer mechanism

The buffer mechanism guarantees a constant throughput for the keystream. However, the buffer must have a reasonable length since the keystream generation process starts when the buffer is full.

Recall that for every α bits that are input into the ABSG, the buffer is supposed to output one bit exactly. The output rate of the ABSG is 1/3 in average. Then, the value of α is greater than 3. For $\alpha = 4$ and a buffer of length 32, the probability that the buffer is empty while it has to output one bit is less than 2^{-89} (the analysis of the buffer mechanism is detailed in [3]).

Timing measurements at the output of the keystream generator is useless since a buffer is used and the throughput is constant. However, if the attacker gets timing information from the internal keystream generator, then timing attacks apply.

5 Hardware implementation

There is a trade-off between the size of the hardware implementation and the throughput of the cipher. Indeed, the 32-bit length of the buffer has been chosen to ensure that the buffer is ready with probability $(1 - 2^{-89})$ to output one bit every 4 bits entered into the ABSG.

Since each LFSR clock contributes one bit to the sequence entering the ABSG mechanism, one solution is to clock four times the LFSR before outputting one bit. 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.

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

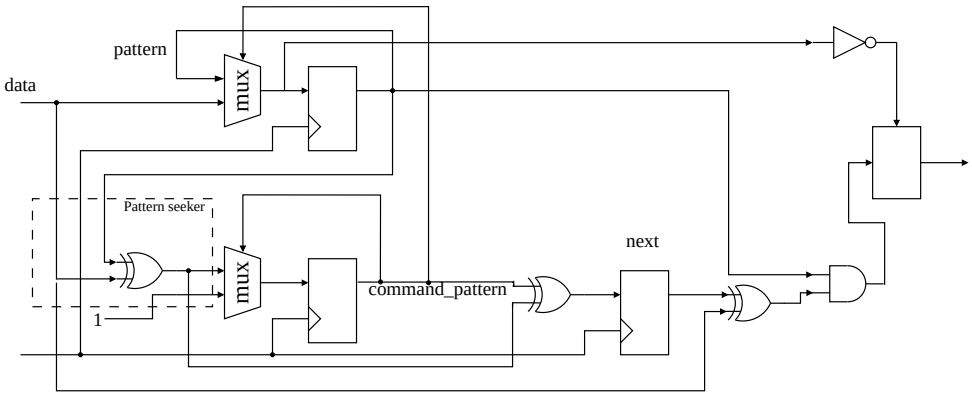

Fig. 4. Hardware implementation of the ABSG

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]. 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.

6 Decim-128

In this section, we describe Decim-128 which is an adaptation of the design of Decim^{v2} to get 128-bit security (we refer to [4] for more details).

DECIM-128 takes as input a 128-bit secret key and a 128-bit public initialization vector. The keystream generation mechanism is similar as the one described in Figure 1 and the Key/IV setup mechanism is similar as the one described in Figure 2 except that the LFSR has length 288.

6.1 The filtered LFSR

The underlying LFSR is a maximum-length LFSR of length 288 (instead of 192) over $\mathbb{F}_2$. It is defined by the following primitive feedback polynomial:

$$P(X) = X^{288} + X^{285} + X^{284} + X^{247} + X^{204} + X^{185} + X^{154} + X^{125} + X^{124} + X^{123} + X^{82} + X^{35} + X^{18} + X^5 + 1$$

The filter function is the same as in DECIM^{v2}. The only difference between DECIM^{v2} and DECIM-128 is a different choice of tap positions:

$$287, 276, 263, 244, 227, 203, 187, 159, 120, 73, 51, 39, 21, 1$$

The sequence $\mathbf{y}$ produced by the filter is of maximal nonlinear complexity, namely equal to $\frac{288 \times 289}{2} = 41616$.

6.2 The buffer mechanism

For DECIM-128, we choose a buffer of 64 bits instead of 32. Since the buffer outputs one bit exactly for every 4 bits that are input into the ABSG, the probability that the buffer is empty while it has to output one bit is less than 2^{-178} at each step.

6.3 Key/IV Setup

The secret key K is a 128-bit key denoted by $K = K_0, \ldots, K_{127}$ and the initialization vector IV is a 128-bit IV denoted by $IV = IV_0, \ldots, IV_{127}$. The initial filling of the LFSR is done as follows.

$$x_i = \begin{cases} K_i & 0 \leq i \leq 127 \\ K_{i-128} \oplus IV_{i-128} & 128 \leq i \leq 255 \end{cases}$$

We complete the register with $x_{256} \ldots x_{287} = 0x55555555$. The number of possible initial values of the LFSR is 2^{256}.

This step slightly differs from the injection in DECIM^{v2}. Namely, it is simpler, partly due to the fact that the key and the IV have the same size.

The update of the LFSR is done in the same way as for DECIM^{v2}. The number of clocks performed is also 4 times the length of the LFSR, so here $4 \times 288 = 1152$ times. After this step, the buffer has to be filled in the same way like for DECIM^{v2}, i.e. by performing the same steps as for keystream generation without shifting the buffer and outputting bits, until the buffer is full. Nevertheless, the buffer is filled with probability bigger than $1 - 2^{-128}$ after 432 steps, which can be used if a constant initialization time is required.

7 Conclusion

We have presented the stream cipher DECIM^{v2} selected in the Phase 3 of the eSTREAM call for stream cipher profile 2, and the 128-bit security version of DECIM^{v2} called DECIM-128.

DECIM^{v2} and DECIM-128 are especially suitable for hardware applications with restricted resources such as limited storage or gate count. Design choices influence the miniaturization of the cipher system:

- the ABSG mechanism has low-cost hardware implementation,
- the filtering function f only depends on the Hamming weight of its input in order to reduce the cost in hardware implementation,
- the IV injection/key schedule re-uses the main components of the keystream generation mechanism.

For applications requiring higher throughputs, speed-up mechanisms can be used to accelerate DECIM^{v2} and DECIM-128 at the expense of a higher hardware complexity. Finally, the security of DECIM^{v2} and DECIM-128 mainly relies on the security of the ABSG, and there is no identified attack better than exhaustive search.

Acknowledgement

This work was partially supported by the French Ministry of Research RNRT Project "X-CRYPT" and by the European Commission via the ECRYPT Network of Excellence IST-2002-507932. Note that this work was done while the 4th author was affiliated to Axalto/Gemalto (France), the 8th and the 13th authors were affiliated to France Télécom R&D/Orange Labs (France), the 9th author was affiliated to the École Normale Supérieure (France), the 11th author was affiliated to INRIA Rocquencourt (France).

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 Workshop SKEW 2005*, 2005. Available at http://www.ecrypt.eu.org/stream/.
4. 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-128. 2007. Available at http://www.ecrypt.eu.org/stream/.
5. 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^{v2}. In *ECRYPT Stream Cipher Workshop SASC 2007*, 2007. Available at http://www.ecrypt.eu.org/stream/.
6. C. De Cannière, J. Lano, and B. Preneel. Comments on the rediscovery of Time Memory Data Tradeoffs. http://www.ecrypt.eu.org/stream/TMD.pdf, 2005.
7. D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In D.R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 1993.
8. P. Ekdahl, T. Johansson, and W. Meier. Predicting the shrinking generator with fixed connections. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
9. A. Gouget and H. Sibert. The Bit-Search Generator. In *The State of the Art of Stream Ciphers: Workshop Record*, pages 60–68, Brugge, Belgium, October 2004.
10. A. Gouget and H. Sibert. How to strengthen pseudo-random generators by using compression. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2006.
11. A. Gouget and H. Sibert. Revisiting correlation-immunity in filter generators. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography*, Lecture Notes in Computer Science. Springer, 2007.
12. A. Gouget, H. Sibert, C. Berbain, N. Courtois, B. Debraize, and C. Mitchell. Analysis of the Bit-Search Generator and sequence compression techniques. In *Fast Software Encryption - FSE 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
13. J. Hong and P. Sarkar. Rediscovery of Time Memory Tradeoffs. http://eprint.iacr.org/2005/090.ps, 2005.
14. W. Meier and O. Staffelbach. The self-shrinking generator. In A. De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1994.
15. H. Wu and B. Preneel. Cryptanalysis of the stream cipher decim. In M.J.B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 30–40. Springer, 2006.