

# DECIM–128 \*

C. Berbain<sup>1</sup>, O. Billet<sup>1</sup>, A. Canteaut<sup>2</sup>, N. Courtois<sup>3</sup>, B. Debraize<sup>3,4</sup>, H. Gilbert<sup>1</sup>,  
L. Goubin<sup>4</sup>, A. Gouget<sup>5</sup>, L. Granboulan<sup>6</sup>, C. Lauradoux<sup>2</sup>, M. Minier<sup>7</sup>,  
T. Pornin<sup>8</sup> and H. Sibert<sup>9</sup>

## Abstract

DECIM is a hardware oriented stream cipher submitted to the ECRYPT stream cipher project. The design of DECIM is based on both a non linearly filtered LFSR and an irregular decimation mechanism called the ABSG. While the initial call for contributions required hardware-oriented stream ciphers to manage 80-bit secret keys and 64-bit public initialization, designers have been invited to demonstrate flexibility of their cipher design by proposing variants that take 128-bit keys as well. In this note, we propose a 128-bit security version of DECIM, called DECIM–128, and we detail the steps required to adapt DECIM to different security levels.

## 1 Introduction

DECIM [2] is a hardware oriented stream cipher submitted to the ECRYPT Stream Cipher Project [1]. As required in the call for contributions initiated by the project, DECIM handles 80-bit secret keys and 64-bit public initialization vectors. Two flaws were pointed out in the original version of Decim [2] by Wu and Preneel in [5]. They were corrected in a revised version [3], called DECIM<sup>v2</sup>, by using a more classical initialization procedure, and choosing a filtering Boolean function that is 1-resilient. We now refer to this revised version as DECIM.

In this note, we first describe a 128-bit version of DECIM, which is dedicated to 128-bit secret keys and initialization vectors.

In a nutshell, the modifications from DECIM are:

---

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

<sup>2</sup>INRIA-Rocquencourt, projet CODES, domaine de Voluceau, B.P. 105, F-78153 Le Chesnay cedex, {anne.canteaut,cedric.lauradoux}@inria.fr

<sup>3</sup>Axalto Smart Cards, 36-38 rue de la Princesse - B.P. 45, F-78431 Louveciennes cedex, {nicolas.courtois,blandine.debraize}@gemalto.com

<sup>4</sup>Laboratoire PRiSM, Université de Versailles, 45 avenue des Etats-Unis, F-78035 Versailles cedex, louis.goubin@prism.uvsq.fr

<sup>5</sup>Gemplus, 34 rue Guynemer, F-92447 Issy-Les-Moulineaux Cedex, aline.gouget@gemalto.com

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

<sup>7</sup>INSA Lyon, 21, avenue Jean Capelle, F-69621 Villeurbanne Cedex, marine.minier@insa-lyon.fr

<sup>8</sup>Cryptolog International, 16-18 rue Vulpian, F-75013 Paris, thomas.pornin@cryptolog.com

<sup>9</sup>France Télécom Recherche et Développement, 42 rue des Coutures, BP 6243, F-14066 Caen cedex 4, herve.sibert@orange-ft.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. Choice of longer components:
  - (a) the LFSR is now 288-bit long instead of 192-bit long in DECIM,
  - (b) the buffer is now 64-bit long instead of 32-bit long in DECIM.
2. Subsequent modifications due to new components and parameters sizes
  - (a) Modification of the key and IV injection (which are now the same size).
  - (b) Change of LFSR feedback polynomial (its weight remains the same as in DECIM).
  - (c) Change of taps of the filtering Boolean function (which is the same as in DECIM).

The outline of the paper is as follows. In Section 2, we fully describe DECIM–128. Then, in Section 3, we explain the general design modifications that are necessary to adapt the original DECIM to various key and IV sizes. Finally, we conclude in Section 4.

## 2 Description of Decim–128

DECIM–128 takes as an input a 128-bit length secret key and a 128-bit length public initialization vector.

### 2.1 Keystream generation

The LFSR in DECIM–128 is 288-bit long (instead of 192 for DECIM). The keystream generation mechanism is described in Figure 1. The bits of the internal state of the LFSR are numbered from 0 to 287, and they are denoted by  $(x_0, \dots, x_{287})$ . The sequence of the linear feedback values of the LFSR is denoted by  $\mathbf{s} = (s_t)_{t \geq 0}$ . The feedback polynomial of DECIM–128 has the same weight as that of DECIM.

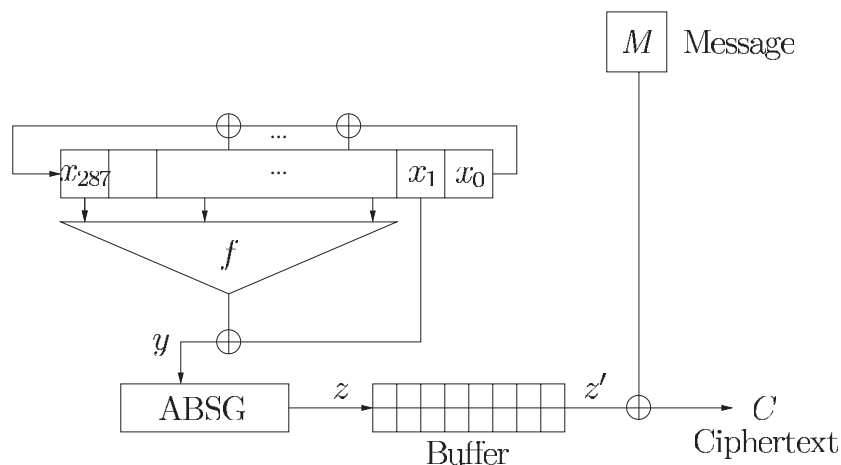


Figure 1: DECIM–128 keystream generation

## 2.2 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 288 over  $\mathbf{F}_2$ . It is defined by the following primitive feedback polynomial:

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

Thus, the linear feedback bit at stage  $t$  is:

$$s_{t+288} = s_{t+283} \oplus s_{t+270} \oplus s_{t+253} \oplus s_{t+206} \oplus s_{t+165} \oplus s_{t+164} \oplus s_{t+163} \oplus s_{t+134} \oplus s_{t+103} \oplus s_{t+84} \\ \oplus s_{t+41} \oplus s_{t+4} \oplus s_{t+3} \oplus s_t$$

**The filter.** The filter function is the same as in DECIM. This 14-variable Boolean function is defined by:

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

where  $f$  is the symmetric quadratic Boolean function defined by:

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

As a reminder, the function  $f$  is a 13-variable quadratic symmetric function which is balanced. The only difference between DECIM and DECIM-128 regarding the filter is a different choice of tap positions:

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

The input of the ABSG at stage  $t$  is thus:

$$y_t = f(s_{t+287}, s_{t+276}, s_{t+263}, s_{t+244}, s_{t+227}, s_{t+203}, s_{t+187}, s_{t+159}, s_{t+120}, s_{t+73}, s_{t+51}, s_{t+39}, s_{t+21}) \oplus s_{t+1}.$$

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

## 2.3 Decimation

Like in DECIM, the keystream sequence  $\mathbf{z}$  is obtained from the sequence  $\mathbf{y}$  by applying the ABSG algorithm, described in Figure 2.

**Remark 1** *Whenever the buffer is empty, the decimation step is not performed. However, with the current parameters, this should not happen, as described in Section 2.4.*

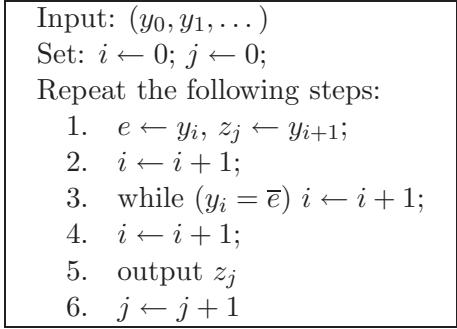


Figure 2: ABSG Algorithm

## 2.4 Buffer mechanism

The rate of the ABSG mechanism is irregular, therefore we use a buffer in order to guarantee a constant throughput. For DECIM–128, we choose a buffer of 64 bits, instead of 32 in DECIM. Like in DECIM, the buffer outputs one bit exactly for every 4 bits that are input into the ABSG. With these parameters, the probability that the buffer is empty while it has to output one bit is less than  $2^{-178}$  at each step.

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 2.5.3.

## 2.5 Key/IV Setup

This subsection describes the computation of the initial inner state for starting the keystream generation.

### 2.5.1 Initial filling of the LFSR.

The secret key  $K$  is a 128-bit key denoted by  $K = K_0, \dots, K_{127}$  and the initialization vector  $IV$  is a 128-bit IV denoted by  $IV_0, \dots, IV_{127}$ .

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  $x_{256} \dots x_{287} = 0x5555555555$ . The number of possible initial values of the LFSR state is  $2^{256}$ .

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

### 2.5.2 Update of the LFSR state.

This step consists in updating the LFSR at each clock using the same nonlinear feedback as in DECIM, described in Figure 3.

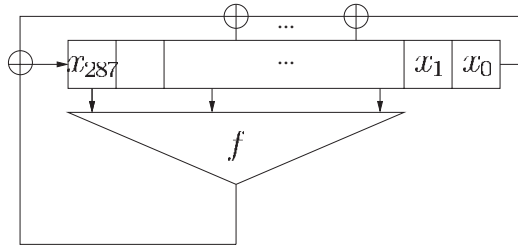


Figure 3: Key/IV setup mechanism

Namely, if we denote by  $y_t$  the output of  $f$  at time  $t$ , and by  $lv_t$  the linear feedback value at time  $t > 0$ , the feedback bit  $s_{t+288}$  is given by:

$$s_{t+288} = lv_t \oplus y_t .$$

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

The number of clocks performed that way is 4 times the length of the LFSR, as in DECIM. Thus, in DECIM-128, the LFSR is clocked  $4 \times 288 = 1152$  times in this step.

### 2.5.3 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 192 steps, and it is filled after 432 steps with probability more than  $1 - 2^{-128}$ . Thus, if a constant duration initialization process is required, one can choose to execute 432 steps and throw away the ABSG output bits when the buffer is full.

## 3 Adapting Decim to different security levels

The design of DECIM can be easily adapted to different key and IV sizes. Changes in the size of key and IV impact the following design choices in DECIM:

1. linear feedback shift register and buffer lengths,
2. subsequent choices of the feedback polynomial and of the filtering function taps,
3. key and IV injection in the initialization phase.

The other design choices, and the rationale behind them, are detailed in [3].

### 3.1 Length of the LFSR

We denote by  $\ell$  the security parameter. Notice first that, in order to obtain the desired security with a secret key of length  $\ell$  exactly, then the initialization vector should be at least  $\ell$ -bit long in order to thwart time-memory trade-off attacks [4]. In this case, the length of the LFSR is chosen at least as being twice the security parameter. Nevertheless, as a part of the initialization is nonlinear, it is necessary to choose a register that is slightly more than twice as long as the secret key. We propose a register length of  $L = (2 + \frac{1}{4})\ell$ .

### 3.2 Length of the buffer

The length of the buffer must be such that the probability that it becomes empty when it has to output one bit is less than  $\frac{1}{2^\ell}$ . In order to ensure that, we check that the sum

$$\sum_{i \geq 1} \frac{1 - \sum_{j=i-\ell}^{2i-1} 2^{j+1} \binom{4i-j-1}{j} - \sum_{j=i-\ell}^{2i} 2^j \binom{4i-j-1}{j-1}}{2^{4i}}$$

is less than  $\frac{1}{2^\ell}$ .

### 3.3 Feedback polynomial and filter

Both the LFSR feedback polynomial and the filtering function should remain of the same complexity as in DECIM. We advise that the filtering function remains exactly the same, except for the positions of the taps, for which criteria appear in [3]. One should also check that the sequence produced by the filter is of maximal nonlinear complexity (using the Berlekamp-Massey algorithm), that is, equal to  $\frac{L(L+1)}{2}$ , where  $L$  is the length of the LFSR.

### 3.4 Initialization process

For the initialization process, the key and IV injection remains the same as for DECIM-128, with completion by alternating zeros and ones if the register is more than twice as long as the key/IV. Then, we perform  $4\ell$  nonlinear LFSR updates as in DECIM.

The buffer has to be filled before keystream generation starts. There are several possibilities, depending on the implementation. The usual keystream generation round (without buffer shifting) can be performed until the buffer is full, with an upper bound on the number of rounds. Another possibility is to perform exactly a fixed number of rounds. Both the upper bound and the fixed number are to be chosen such that the buffer is full with probability at least  $1 - \frac{1}{2^\ell}$  at the end of the process. This is ensured by choosing a number  $N$  of rounds such that we have

$$1 - 2^{L_B} \sum_{i=2L_B}^N \frac{\binom{i-L_B-1}{i-2L_B}}{2^{-i}} < \frac{1}{2^\ell},$$

where  $L_B$  is the length of the buffer.

## 4 Conclusion

We have described DECIM, a 128-bit security version of DECIM. We have also provided guidelines to adapt the original design to different key and IV sizes. At last, it should be noted that the hardware complexity of the cipher increases linearly with the security parameter/key size.

## References

- [1] eStream, Stream cipher project of the European Network of Excellence in Cryptology ECRYPT. <http://www.ecrypt.eu.org/stream/>.
- [2] 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/>.
- [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<sup>v2</sup>. In *SASC 2006 - Stream Ciphers Revisited: Workshop Record*, Leuven, Belgium, 2006. Available at <http://www.ecrypt.eu.org/stream/>.
- [4] Jin Hong and Palash Sarkar. New Applications of Time Memory Data Tradeoffs. In *Advances in Cryptology - ASIACRYPT 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
- [5] Hongjun Wu and Bart Preneel. Cryptanalysis of Stream Cipher Decim. In *Fast Software Encryption, FSE 2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006. Available at <http://www.ecrypt.eu.org/stream/>.