Update on F-FCSR Stream Cipher

F. Arnault*, T.P. Berger* and C. Lauradoux†

Abstract

The F-FCSR family of algorithms have been presented about one year ago with [2] and [1]. While some flaws where found in the initial propositions (on the IV-setup procedure, and a TMD tradeoff attack), there are yet no known weaknesses of the core of these algorithms.

We sum up here some of the properties of the automaton that are better understood now, and that have been presented in [2], [3], [4], and [6] and we propose two revised algorithms correcting all known weaknesses.

1 Recalls on F-FCSR

1.1 FCSR automaton

Detailed descriptions can be found in [3, 1, 2].

A Feedback with Carry Shift Register (FCSR) is an automaton which computes the binary expansion of a 2-adic number p/q, where p and q are some integers, with q is odd. We will assume that $q < 0 < p < |q|$. The size n of the FCSR is such that $n + 1$ is the bitlength of $|q|$.

In our applications, p depends on the secret key (and the IV), and q is a public parameter. The choice of q induces many properties of the keystream. The most important one is that it completely determines the length of the period of the keystream. The conditions for an optimal choice are:

Conditions 1

- *q is a (negative) prime of bitsize $n + 1$.*
- *The order of 2 modulo q is $|q| - 1$.*
- *$T = (|q| - 1)/2$ is also prime.*
- *Set $d = (1 + |q|)/2$. The Hamming weight $W(d)$ of the binary expansion of d is not too small. Typically, $W(d) > n/2$.*

1.1.1 Software description of the transition function

The FCSR automaton contains two registers (sets of cells): the main register M and the carries register C.

The main register M contains n cells. We denote m_i $(0 \le i \le n-1)$ the binary digits contained in these cells and we call the integer $m = \sum_{i=0}^{n-1} m_i 2^i$ the content (or state) of M.

*XLIM, Université de Limoges, 123 avenue A. Thomas, 87060 Limoges CEDEX, France
Email : arnault@unilim.fr thierry.berger@unilim.fr

†INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France
Email : cedric.lauradoux@inria.fr

Let d be the positive integer $d = (1-q)/2$ and $d = \sum_{i=0}^{n-1} d_i 2^i$ its binary expansion. The carries register contains ℓ cells where $\ell + 1$ is the number of nonzero d_i digits. More precisely, the carries register contains one cell for each nonzero d_i with $0 \leq i \leq n-2$. We denote c_i the binary digit contained in this cell. We also put $c_i = 0$ when $d_i = 0$ or when $i = n-1$. We call the integer $c = \sum_{i=0}^{n-2} c_i 2^i$ the content (or state) of C. The Hamming weight of the binary expansion of c is at most ℓ.

The transition function can be described by

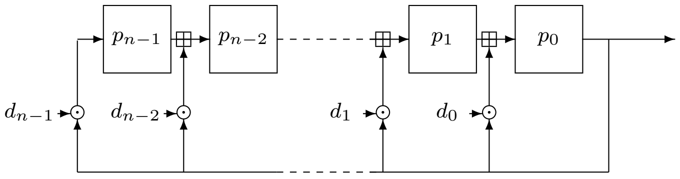
$$\begin{aligned} m(t+1) &:= (m(t) \div 2) \oplus c(t) \oplus m_0(t)d \\ c(t+1) &:= (m(t) \div 2) \otimes c(t) \oplus c(t) \otimes m_0(t)d \oplus m_0(t)d \otimes (m(t) \div 2) \end{aligned}$$

where $\oplus$ denotes bitwise XOR, $\otimes$ denotes bitwise AND, and $\div 2$ is a just a shift to the right.

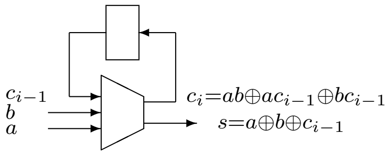
Note that $m_0(t)$ is the least significant bit of $m(t)$. The integers $m(t)$, $c(t)$ and d are integers of bitsize n (or less).

1.1.2 Hardware description of the transition function

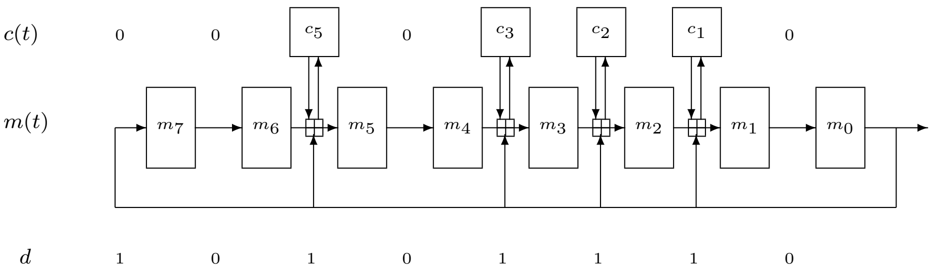
With the same notations, the hardware description of the FCSR generator is



where the symbol $\boxplus$ denotes the addition with carry, i.e., it corresponds to the following scheme:


c_{i-1}
b
a
$c_i = ab \oplus ac_{i-1} \oplus bc_{i-1}$
$s = a \oplus b \oplus c_{i-1}$

As an example, if $q = -347$, so $d = 174 = 0\text{xAE}$, $n = 8$ and $\ell = 4$, we obtain the following diagram:



1.2 Filtering

We extract each pseudorandom bit from the state of the main register of the FCSR automaton using a filter. This filter describes which cells are selected to produce the pseudorandom bit. In order to obtain a multi-bit output, eight or sixteen one bit subfilters are used to extract an output 8 or 16 bits word after each transition of the automaton.

1.2.1 Principle of one bit filtering

The filter F is a bitstring $(f_0, \ldots, f_{n-1})$ of length n (or equivalently the integer $\sum_{i=0}^{n-1} f_i 2^i$). The output bit is obtained by computing the weight parity of the bitwise AND of the state M of the main register and of the filter F:

$$\text{Output bit} := \bigoplus_{i=0}^{n-1} f_i m_i.$$

Or, equivalently: $S = M \otimes F$ Output bit := parity(S)

1.2.2 Word filtering

In a similar way, we propose a method to extract an s bits word from the state of the FCSR. The value of s will be 8 for F-FCSR-H, and 16 for F-FCSR-16.

The filter F is also a bitstring $(f_0, \ldots, f_{n-1})$ of length n (which is a multiple of s). It splits into s subfilters $F_0, \ldots, F_{s-1}$ each defined by

$$F_j = \sum_{i=0}^{n/s-1} f_{si+j} 2^i.$$

Each subfilter F_j selects some cells m_i in the main register among the ones satisfying $i \equiv j$ modulo s. The parity of the binary word obtained gives one pseudorandom bit :

$$\text{bit } j \text{ of output word} := \bigoplus_{i=0}^{n/s-1} f_{si+j} m_{si+j}.$$

As there are s subfilters, we get s bits at each transition of the automaton.

This procedure can be described equivalently as follows. The filter F and the state of M are combined with the AND function. The result is split into n/s words. The pseudorandom word is obtained by XORing these n/s words:

$S := M \otimes F$
Define S_i by $S = \sum_{i=0}^{n/s-1} S_i \cdot 2^{si}$, with $0 \leq S_i \leq 2^s - 1$
Output word := $\bigoplus_{i=0}^{n/s-1} S_i$.

Note that it is faster to extract a whole word than a single bit.

2 Known issues on F-FCSR

2.1 Structure of the cycles of an FCSR automaton

Consider the transition function of an FCSR automaton. It is easy to see that it has two fixed points, namely the state with all cells containing a 0 bit, and the state with all cells containing a 1 bit. The values of (m, c) for these states are $(0, 0)$ and $(2^n - 1, d - 2^n)$ respectively, and they correspond to the developpement of the 2-adic fractions $0/q = 0$ and $|q|/q = -1$. All other states are noninvariant by the transition function.

Since we assume that the order of 2 modulo q is $|q| - 1$, we can prove that the graph of the transition function consists of exactly three connected components: the two single point components corresponding to the two fixed points and another component containing all the $2^{n+l} - 2$ remaining points. Moreover, this component consists in a cycle of size $|q| - 1$ and pathes converging to it. More details on the transition function of FCSR automatons can be found in [3].

Definition 1 *Two states (m_1, c_1) and (m_2, c_2) are said equivalent if they satisfy $m_1 + 2c_1 = m_2 + 2c_2$.*

The following fundamental property can be shown:

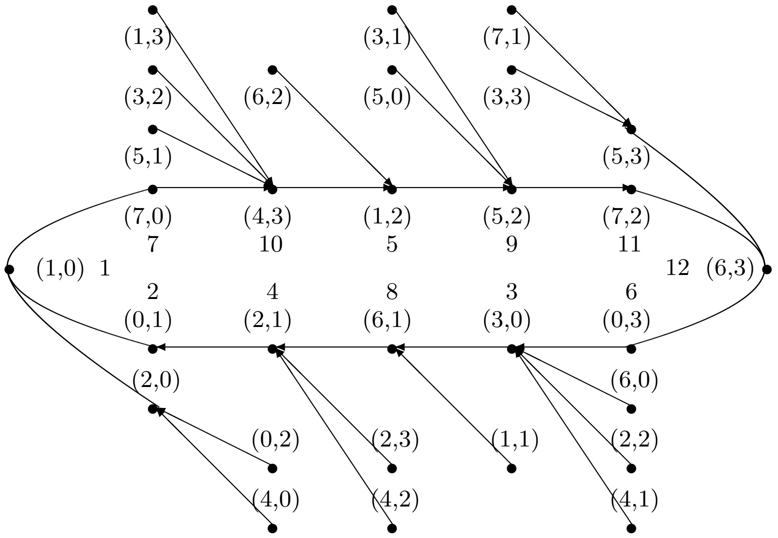
Proposition 1 *Two noninvariant states are equivalent if and only if they eventually converge to the same state of the main cycle in the same number of steps.*

As $|q| - 1 \simeq 2^n$, the expected number of states which eventually converge to a given state of the cycle is approximatively 2^l.

It can be shown also that the relative number of leaves for the transition function is $1-(3/4)^l$, which (for $l \geq 2$) is much larger than for a random function, where it is e^{-1}.

From the existence of a large cycle and of a large number of leaves, we can expect that the length of the pathes converging to the cycle are very short. Experimentally, this is indeed the case. Convergence occurs generally in less than $(n+l)/2$ iterations, while this should be $2^{(n+l)/2}$ if the transition function was a random one.

The following figure shows the main composant of the graph associated to $q = -13$. The couples of numbers correspond to a state (m, c) and the single numbers to the value $p = m + 2c$.



2.2 Consequences for Time/Memory/Data tradeoff attacks

First, the states that are not on the main cycle have only a very small impact on the cost of a Time/Memory/Data tradeoff attacks. So the number of states that should be considered when evaluating security of FCSRs with static filter should be $|q| - 1$ instead of 2^{n+l}.

Hence, in the first version of F-FCSR-8 submitted to Ecrypt, the size $k = 128$ of the key was equal to the length of the main register. But we also used a dynamic filter to increase the number of total states of our automaton.

However, using the fact that we used 8 subfilters of length 16 to output 8 bits at each transition, and using the fact that the number of possible such subfilters was too small, E. Jaulmes and F. Muller showed in [5] that the presence of a dynamic filter does not provide enough security. Their attack has a time cost in 2^{80} and uses data of size about 2^{67}.

The solution to prevent TMD-attacks is to increase the size of the prime q up to $n = 2\times128 = 256$. Note that in this case, it is possible to output two bytes instead of a single one at each iteration. Hence the number of operations per output byte is not increased and in fact the speed of the generator will be slightly better. Moreover, dynamic filter is no longer needed, and this greatly simplifies hardware and also software implementations.

There exist recent developpements on TMDtreadoffs cryptanalysis of stream cipher generators [8]. We want to notice that it possible to increase the size of IV of F-FCSR stream cipher until the size k of the key without any information. Moreover, in the procedure of change of IV, the key is concatened to the IV, which ensure a total entropy of our system equals to the size of the key plus the size of the IV.

2.3 Algebraic cryptanalysis

For the F-FCSR generator, the transition function of the automaton T_q is quadratic, and the filter F_l is linear.

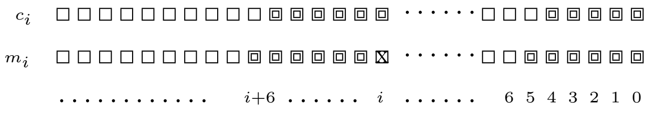
We denote by x the initial state of the generator: it is a binary vector of size equal to the number of the unknown values of the registers. The algebraic attack consists in the determination of x from the equations $F(T^i(x)) = s_i$, where the s_i are the successive observed bits output by the generator.

This leads to a system of equations $F_l(T_q^i(x)) = s_i$. The degree of the i-th equation is the degree of T_q^i. The first equation is linear, the second quadratic. An increase of the degree is expected at each iteration. However this increase depends on many factors as the choice of the filter or the values of I_d. It seems not possible to find a formula available in the general case.

However, In [4], M. Minier and T. Berger studied these equations in more details and designed an attack on an earlier version of F-FCSR proposed at FSE 05 [2]. In that situation, there was only 6 iterations after each change of IV.

The main result is the fact that, even if the degree of equations increases at each iteration, the number of monomials remains smaller than expected as long as the number of iterations is less than the size n of the register. The following Proposition describes this property:

Proposition 2 *The value of the content of the i-th register at the t-iteration $m_i(t)$ depends only on the initial values $(m_0(0), \cdots, m_{t-1}(0), c_0(0), \cdots, c_{t-2}(0)$ et $(m_{i+1}(0), \cdots, m_{i+t}(0), c_i(0)$ $,\cdots, c_{i+t-1}(0))$.*



An example for $t = 6$

In the attack described in [4], the IV values are known, that is the initial values $c_i(0)$ are given. The following table gives the number of distinct monomials obtained in the algebraic equations for a register of size 128.

nb of iterations	0	1	2	3	4	5	6
nb of monomials	128	129	256	758	2490	8830	32836
Algebraic degree	1	1	2	3	4	6	8
Binomial bound	129	129	8257	349633	11017633	$\approx 2^{32}$	$\approx 2^{40}$

There are some remarks about these results:

- From Proposition 2, the number of monomials is linear in the length n of the generator.
- From a computationnal point of view, the first difficult problem is not to solve the equations, but to compute them: we were not able to compute the equations corresponding to the 8-th iteration on a register of size 128. At the present moment, it seems not computationnaly feasible to complete the 12-th iteration.
- The F-FCSR-8 and F-FCSR-H stream ciphers proposed to Ecrypt are resistant to this kind of attacks.

2.4 Diffusion of differences

Another possible weakness of the first designs of F-FCSR stream ciphers resides in the slowness of diffusion of differences. A difference introduced in some cell of the FCSR automaton remains localized when clocking the automaton, as long as this difference does not reach the feedback end of the register. In fact, except when this end is reached, the difference only affects the next right cell after one transition and, with probability 1/2 only, the corresponding carry cell is also changed. This change in this carry cell, when it occurs, will cause subsequent differences at subsequent transitions. However, this change in the carry cell has low probability ($1/2^n$ after n transitions) not to disappear.

We illustrate this fact in the following example, where we choosed $q = -347$. The length n of the main register is then 8. We have chosen randomly a value m_1 for the main register m, strictly less than 2^7. For the initial values m_1 and $m_2 = m_1 + 2^7$, we computed the differences obtained in the main register after i iterations of the transition function, for $i = 0$ up to 9. The following table gives the typical results obtained this way, with two different values for m_1.

Position of carries								Position of carries							
1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0
Diffusion of difference								Diffusion of difference							
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1
1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	0
1	1	1	1	1	0	0	1	0	1	0	1	0	1	0	1

This fact was noticed by E. Jaulmes and F. Muller (cf. [6, 5]) and used to design attacks on the change of IV procedure. They obtained a key recovery attack on F-FCSR-8 and a distinguishing attack on F-FCSR-H.

There are two independant ways in order to stop these attacks:

1. A better insertion of IV and key in the change of IV procedure. In our first version, we used a simple concatenation of the values.
2. A larger number of iterations of the transition function before outputing data, greater than the length n of the register, to ensure a full diffusion of the differences

As an example, in our first version of F-FCSR H , it is sufficient to increase the number of iterations from 160 to 162 in order to stop the distinguishing attack.

2.5 Other attacks

At the present moment, we do not known any other attack against this design, in particular any correlation attack.

3 New Design

3.1 F-FCSR-H: Profile 2, output 1 byte per round

This proposal uses keys of length 80 and IV of bitsize v with $32 \leq v \leq 80$. An IV of value 0 can be used as a default if no value is provided. The core of this new version of the F-FCSR-H algorithm is identical to the one proposed in [1]. Only the key+IV Setup procedure has been updated in view of the attacks presented in [6].

The FCSR length (size of the main register) is $n = 160$. The carries register contains $\ell = 82$ cells. The retroaction prime is

$$q = -1993524591318275015328041611344215036460140087963$$

so addition boxes and carries cells are present at the positions matching the ones (except of the leading one) in the following 160 bits string (which has Hamming weight 83)

$$d = (1 + |q|)/2 = (\mathrm{AE985DFF\ 26619FC5\ 8623DC8A\ AF46D590\ 3DD4254E})_{16}.$$

Filtering

To extract one pseudorandom byte, we use the static filter

$$F = d = (\mathrm{AE985DFF\ 26619FC5\ 8623DC8A\ AF46D590\ 3DD4254E})_{16}$$

The filter F splits in 8 subfilters (subfilter j is obtained by selecting the bit j in each byte of F)

$$\begin{array}{llll}
F_0 & = (0011\ 0111\ 0100\ 1010\ 1010)_2, & F_4 & = (0111\ 0010\ 0010\ 0011\ 1100)_2, \\
F_1 & = (1001\ 1010\ 1101\ 1100\ 0001)_2, & F_5 & = (1001\ 1100\ 0100\ 1000\ 1010)_2, \\
F_2 & = (1011\ 1011\ 1010\ 1110\ 1111)_2, & F_6 & = (0011\ 0101\ 0010\ 0110\ 0101)_2, \\
F_3 & = (1111\ 0010\ 0011\ 1000\ 1001)_2, & F_7 & = (1101\ 0011\ 1011\ 1011\ 0100)_2.
\end{array}$$

Recall that the bit b_i (with $0 \leq i \leq 7$) of each extracted byte is expressed by

$$b_i = \bigoplus_{j=0}^{19} f_i^{(j)} m_{8j+i} \qquad \text{where } F_i = \sum_{j=0}^{19} f_i^{(j)} 2^j$$

and where the m_k are the bits contained in the main register.

a+b. Key+IV setup (Inputs a key K of length $k = 80$ and an IV of length $v \leq 80$)

1. The main register M is initialized with the key and the IV:

$$M := K + 2^{80} \cdot IV = (0^{80-v} \| \mathrm{IV} \| K).$$

2. The carries register is initialized to 0 :

$$C := 0 = (0^{82}).$$

3. A loop is iterated 20 times. Each iteration of this loop consists in clocking the FCSR and then extracting a pseudorandom byte S_i $(0 \leq i \leq 19)$ using the filter.
4. The main register M is reinitialized with these bytes:

$$M := \sum_{i=0}^{19} S_i = (S_{19} \| \cdots \| S_1 \| S_0).$$

5. The FCSR is clocked 162 times (output is discarded in this step).

c. Extraction of pseudorandom data After setup phase, the pseudorandom stream is produced by repeating the following process as many times as needed

- Clock the FCSR
- Extract one pseudorandom byte using filter F as described above.

3.2 Upgrade from F-FCSR-8 to F-FCSR-16

In the F-FCSR-8 algorithm presented in [1], the pseudorandom stream was extracted using a dynamic filter. The purpose of this filter was to enlarge the number of states of the FCSR-automaton, in order to prevent Time-Memory-Data tradeoff attacks. However, the paper [6] shows that such a dynamic filter does not provide the expected security. In the light of this result, the new algorithm F-FCSR-16 uses a static filter and the required number of states of the automation is obtained by enlarging the size of the registers. Note that the larger size of the register allows to extract more pseudorandom bits at each transition of the automaton. So the new algorithm is as fast as the previous one.

3.2.1 F-FCSR-16: Profile 1, output 2 bytes per round

This proposal uses keys of length $k = 128$ and an IV of length $v = 128$ or 64 (any length $v \leq 128$ can be used). An IV of value 0 can be used as a default if no value is provided by the application.

According to Conditions 1 we choose for q the following number

$$-q = 183971440845619471129869161809344131658298317655923135753017128462155618715019$$

as the public parameter of the automaton. The corresponding bitstring $d = (|q| + 1)/2$ which describes the positions of the carries cells is

$$d = (\text{CB5E129F AD4F7E66 780CAA2E C8C9CEDB 2102F996 BAF08F39 EFB55A6E 390002C6})_{16}.$$

Its Hamming weight is 131 and there are $\ell = 130$ cells (the Hamming weight of $d^* = d - 2^{255}$) in the carries register and $n = 256$ cells in the main register.

To extract two pseudorandom bytes, we use the static filter

$$F = d$$

The filter F splits in 16 subfilters (subfilter j is obtained by selecting the bit j in each 16-bit word of F)

$$\begin{array}{llrl}
F_0 & = (0110\ 0011\ 0001\ 1000)_2, & F_8 & = (1010\ 0000\ 1101\ 1010)_2, \\
F_1 & = (1111\ 0101\ 1100\ 0101)_2, & F_9 & = (1101\ 0101\ 0011\ 1101)_2, \\
F_2 & = (1111\ 1100\ 0100\ 1101)_2, & F_{10} & = (0011\ 0001\ 0001\ 1000)_2, \\
F_3 & = (1110\ 1111\ 0001\ 0100)_2, & F_{11} & = (1011\ 1111\ 0111\ 1110)_2, \\
F_4 & = (1100\ 0001\ 0111\ 1000)_2, & F_{12} & = (0101\ 1000\ 0110\ 0110)_2, \\
F_5 & = (0001\ 0100\ 0011\ 1100)_2, & F_{13} & = (0011\ 1100\ 1110\ 1010)_2, \\
F_6 & = (1011\ 0011\ 0010\ 0101)_2, & F_{14} & = (1001\ 1011\ 0100\ 1100)_2, \\
F_7 & = (0100\ 0011\ 0110\ 1001)_2. & F_{15} & = (1010\ 0111\ 0111\ 1000)_2.
\end{array}$$

Recall that the bit b_i (with $0 \le i \le 15$) of each extracted word is expressed by

$$b_i = \bigoplus_{j=0}^{15} f_i^{(j)} m_{16j+i} \qquad \text{where } F_i = \textstyle\sum_{j=0}^{15} f_i^{(j)} 2^j$$

and where the m_k are the bits contained in the main register.

a+b. Change of IV (Input: an IV of bitsize $v \le 128$)

```
$M := K + 2^{128} \cdot \mathrm{IV} = (0^{128-v} \| IV \| K)$
$C := 0 = (0^{130})$        (Clear the carries)
For i from 0 to 15 Repeat
    Clock the FCSR automaton
    Extract a pseudorandom word $S_i$ using the filter $F$
End For
$M := \sum_{i=0}^{15} S_i \cdot 256^i = (S_{15} \| \cdots \| S_0)$
$C := 0 = (0^{130})$        (Clear the carries)
Clock the FCSR automaton 258 times (discard output in this step)
```

c. Extraction of the pseudorandom stream We use the word filtering method described above, with $s = 16$, while pseudorandom data is needed. At each clock of the FCSR automaton, the content of the main register M is ANDed with the filter F:
$S = M \otimes F$
S is split in 16 words each of bitlength 16 $S = \sum_{i=0}^{15} S_i 2^{16i}$
The pseudorandom byte is the XOR of these bytes: Output word $:= \bigoplus_{i=0}^{15} S_i$

3.2.2 F-FCSR-16, Profile 2

The F-FCSR-16 algorithm can also satisfy profile 2. As in this case the key-length is 80, the first line of the Change of IV procedure now reads

$$M := K + 2^{128} \cdot \text{IV} = (0^{128-v} \| IV \| 0^{48} \| K)$$

Comparing to F-FCSR-H, the pseudorandom word extracted at each transition of the automaton is twice larger, while the size of the registers is only 8/5 larger. In applications with profile 2 where extremely high speed of pseudorandom data generation is needed, the F-FCSR-16 algorithm should be also considered.

4 Performances

The software performance of F-FSCR-16 stream cipher depends on the processor register width. For instance, we observe a speedup by four with 128-bits ALTIVEC implementation over 32-bits implementation. This observation was already performed in [9] with F-FCSR-8. The main mechanisn of F-FCSR-H remains unchanged and results on its implementation can be found in [9].

CISC target	parameters		performance		
	Frequency	L2 Cache Size	Speed	Code	Initialization
Pentium 3	800 Mhz	256KB	83 cycles/B	8 KB	39140 cycles/IV
Pentium 4	2.3 Ghz	512KB	85 cycles/B	8 KB	54491 cycles/IV
Pentium 4	2.6 Ghz	512KB	95 cycles/B	8 KB	38351 cycles/IV
Pentium 4	3.2 Ghz	1MB	82 cycles/B	6 KB	43354 cycles/IV

RISC target	parameters		performance		
	Frequency	L2 Cache Size	Speed	Code	Initialization
PPC 7457	1.2 Ghz	512 KB	90 cycles/B	18 KB	44860 cycles/IV
PPC 7457 (ALTIVEC)	1.2 Ghz	512 KB	22 cycles/B	14 KB	11828 cycles/IV

Figure 1: F-FCSR-16 32-bit evaluation and ALTIVEC implementation

References

[1] F. Arnault and T.P. Berger. Design of new pseudorandom generators based on a filtered FCSR automaton. In *SASC*, State of the Art of Stream Ciphers Workshop, pages 109–120, Bruges, Belgium, October 2004.

[2] F. Arnault and T.P. Berger. F-FCSR: design of a new class of stream ciphers. In H. Handschuh H. Gilbert, editor, *Fast Softward Encryption 2005*, number 3557 in Lecture Notes in Computer Science, pages 83–87. Springer, 2005.

[3] F. Arnault and T.P. Berger. Design and properties of a new pseudorandom generator based on a filtered FCSR automaton. *IEEE, Transactions on Computers. IEEE, Transactions on Computers*, 54(11):1374–1383, November 2005.

[4] T.P. Berger and M. Minier. Two algebraic attacks against the F-FCSRs using the IV mode. in S. Maitra, C.E. Veni Madhavan, R. Venkatesan editors, *Progress in Cryptology - INDOCRYPT 2005* number 3797 in Lecture Notes in Computer Science, pages 143–154. Springer, 2005.

[5] E. Jaulmes and F. Muller. Cryptanalysis of Ecrypt candidates F-FCSR-8 and F-FCSR-H. ECRYPT Stream Cipher Project Report 2005/046, 2005. `http://www.ecrypt.eu.org/stream`.

[6] E. Jaulmes and F. Muller. Cryptanalysis of the F-FSCR stream cipher family. In *proceedings of 12th annual workshop on Selected Areas in Cryptography*, LNCS, Springer-Verlag, 2005.

[7] F. Arnault, T.P. Berger and C. Lauradoux. Preventing weaknesses on F-FCSR in IV mode and tradeoff attack on F-FCSR-8. ECRYPT Stream Cipher Project Report 2005/075, 2005. `http://www.ecrypt.eu.org/stream`.

[8] J. Hong and P. Sarkar. New Applications of Time Memory Data Tradeoffs. in B. Roy editor, *Advances in Cryptology - ASIACRYPT 2005* number 3788 in Lecture Notes in Computer Science, pages 353–372. Springer, 2005.

[9] F. Arnault, T.P. Berger and C. Lauradoux. F-FCSR. ECRYPT Stream Cipher Project Report 2005/008, 2005. `http://www.ecrypt.eu.org/stream`.