Preventing weaknesses on F-FCSR in IV mode and tradeoff attack on F-FCSR 8

F. Arnault, T.P. Berger, C. Lauradoux

October 24, 2005

Abstract

E. Jaulmes and F. Muller have described some attacks on F-FCSR-8 and F-FCSR-H algorithms [1]. These attacks pointed out three weaknesses on the algorithms. The first one is a bottleneck effect due to a big mistake in our design. This can be repaired by only removing one line of code in the F-FCSR-8 algorithm. The second weakness lies in the diffusion of the IV which is not good for both algorithms, due to a too simple Key+IV–setup procedure. The last weakness is that F-FCSR-8 is vulnerable to a TMD-tradeoff attack, using the fact that the number of possible values of each subfilter is relatively small.

In this paper, we repair all the weaknesses that were pointed out. We propose a better Key+IV–setup procedure to suppress the bottleneck and have a good diffusion of the IV. To thwart the TMD tradeoff attack on F-FCSR-8, we had to increase the size of the main register up to 256 bits. But we can now extract two pseudorandom bytes at each transition of the automaton instead of one, so the performances remain at least as good as before.

1 Repairing F-FCSR-H : a better Key+IV–setup procedure

As in the original version, we put the key and IV bits in the main register, but we then collect the first twenty bytes output by the automaton and feed them back to the main register. Then we wait enough transitions of the automaton, similarly as in the original version, before using the pseudorandom stream.

Description of the new procedure "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 carries register is reinitialized to 0 : $C := 0$.

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

2 Update F-FCSR-8 to F-FCSR-8.2

Change of the connection integer

We had to change the registers length. So we replace the previous 129 bits connection integer by a 257 bits one. The new value is:

$-q = 183971440845619471129869161809344131658298317655923135753017128462155618715019$

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

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

Modification of the extraction procedure

Due to the larger size of the main register (256 bits), it is now possible to output two bytes at each iteration instead of one. Below is the new extraction method:

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 16 bits word extracted is the XOR of these words:

Output word $:= \bigoplus_{i=0}^{15} S_i$

Modification of the test about the quality of the filter

The changes on the extraction procedure implies minor modification of the function "GoodFilter" used to evaluate the quality of the filter F. Here is the new function GoodFilter, which takes as input a bitstring of length 256 and outputs True if this bitstream is suitable as a filter. Else, it outputs False.

Function GoodFilter (F)

Define the 16 subfilters $F_0, \ldots, F_{15}$, each of bitlength 16, by

$$F_j = (f_j, f_{16+j}, f_{32+j}, \ldots, f_{16i+j}, \ldots, f_{240+j}).$$

If any one of the subfilters F_j has an Hamming weight < 3 THEN output False;
Output True (in all other cases).
End Function

3 Key+IV Setup for F-FCSR-8.2

As the registers are now larger, we propose a new Key setup and change of IV procedure, very similar to our new Key+IV Setup procedure described above for F-FCSR-H.

Description of the procedure "Key+IV Setup"

Inputs a key K of length $k = 128$ and an IV of length $v \leq 128$.

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

4 Conclusion

As stated in [1], the core of F-FCSR-H and F-FCSR-8 was not broken. The found weaknesses were related to flawed change of IV procedures and, for F-FCSR-8, to a too small register length to prevent a carefully designed TMD-tradeoff attack. The modifications proposed here do remove all known weaknesses to the F-FCSR algorithms.

References

[1] E. Jaulmes, F. Muller. Cryptanalysis of ECRYPT Candidates F-FCSR-8 and F-FCSR-H
`http://www.ecrypt.eu.org/stream/ffcsr.html`