Grain - A Stream Cipher for Constrained Environments

Martin Hell[1], Thomas Johansson[1] and Willi Meier[2]

[1] Dept. of Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{martin,thomas}@it.lth.se
[2] FH Aargau, CH-5210 Windisch, Switzerland
meierw@fh-aargau.ch

Abstract. A new stream cipher, Grain, is proposed. The design targets hardware environments where gate count, power consumption and memory is very limited. It is based on two shift registers and a nonlinear filter function. The cipher has the additional feature that the speed can be increased at the expense of extra hardware. The key size is 80 bits and no attack faster than exhaustive key search has been identified. The hardware complexity and throughput compares favourably to other hardware oriented stream ciphers like E0 and A5/1.

1 Motivation

When designing a cryptographic primitive there are many different properties that have to be addressed. These include e.g. speed, security and simplicity. Comparing several ciphers, it is likely that one is faster on a 32 bit processor, another is faster on an 8 bit processor and yet another one is faster in hardware. The simplicity of the design is another factor that has to be taken into account, but while the software implementation can be very simple, the hardware implementation might be quite complex.

There is a need for cryptographic primitives that have very low hardware complexity. An RFID tag is a typical example of a product where the amount of memory and power is very limited. These are microchips capable of transmitting an identifying sequence upon a request from a reader. Forging an RFID tag can have devastating consequences if the tag is used e.g. in electronic payments and hence, there is a need for cryptographic primitives implemented in these tags. Today, a hardware implementation of e.g. AES on an RFID tag is not feasible due to the large number of gates needed. Grain is a stream cipher primitive that is designed to be very easy and small to implement in hardware.

Many stream ciphers are based on linear feedback shift registers (LFSR), not only for the good statistical properties of the sequences they produce, but also for the simplicity and speed of their hardware implementation. Several recent LFSR based stream cipher proposals, see e.g. [5, 6] and their predecessors, are based on word oriented LFSRs. This allows them to be efficiently implemented in software

but it also allows them to increase the throughput since words instead of bits are output. In hardware, a word oriented cipher is likely to be more complex than a bit oriented one. We have addressed this issue by basing our design on bit oriented shift registers with the extra feature of allowing an increase in speed at the expense of more hardware. The user can decide the speed of the cipher depending on the amount of hardware available.

The proposed primitive is a bit oriented synchronous stream cipher. In a synchronous stream cipher the keystream is generated independently from the plaintext. The design is based on two shift registers, one with linear feedback (LFSR) and one with nonlinear feedback (NFSR). The LFSR guarantees a minimum period for the keystream and it also provides balancedness in the output. The NFSR, together with a nonlinear filter introduces nonlinearity to the cipher. The input to the NFSR is masked with the output of the LFSR so that the state of the NFSR is balanced. Hence, we use the notation NFSR even though this is actually a filter. What is known about cycle structures of nonlinear feedback shift registers cannot immediately be applied here. Both shift registers are 80 bits in size. The key size is 80 bits and the IV size is specified to be 64 bits. The cipher is designed such that no attack faster than exhaustive key search should be possible, hence the best attack should require a computational complexity not significantly lower than 2^{80}.

Grain provides a higher security than several other well known ciphers intended to be used in hardware applications. Well known examples of such ciphers are E0 used in Bluetooth and A5/1 used in GSM. These ciphers, while also having a very small hardware implementation, have been proven to be very insecure. Compared to E0 and A5/1, Grain provides higher security while maintaining a small hardware complexity.

The paper is organized as follows. Section 2 provides a detailed description of the design. Section 3 gives the design criterias and the design choices and the strengths and limitations of the design are presented in Section 4. In Section 5 we consider the hardware implementation of the cipher and in Section 6 we give the results of our security analysis. Section 7 concludes the paper.

2 Design Specification

This section specifies the details of the design. An overview of the different blocks used in the cipher can be found in Fig. 1 and the specification will refer to this figure. The cipher consists of three main building blocks, namely an LFSR, an NFSR and a filter function. The content of the LFSR is denoted by $s_i, s_{i+1}, \ldots, s_{i+79}$ and the content of the NFSR is denoted by $b_i, b_{i+1}, \ldots, b_{i+79}$. The feedback polynomial of the LFSR, $f(x)$ is a primitive polynomial of degree 80. It is defined as

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}.$$

To remove any possible ambiguity we also define the update function of the LFSR as

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i.$$

The feedback polynomial of the NFSR, $g(x)$, is defined as

$$\begin{aligned} g(x) = 1 &+ x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + \\ &+ x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + \\ &+ x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + \\ &+ x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}. \end{aligned}$$

Again, to remove any possible ambiguity we also write the update function of the NFSR. Note that the bit s_i which is masked with the input is included in the update function below.

$$\begin{aligned} b_{i+80} = s_i &+ b_{i+63} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + \\ &+ b_{i+15} + b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + \\ &+ b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + \\ &+ b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + \\ &+ b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + \\ &+ b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}. \end{aligned}$$

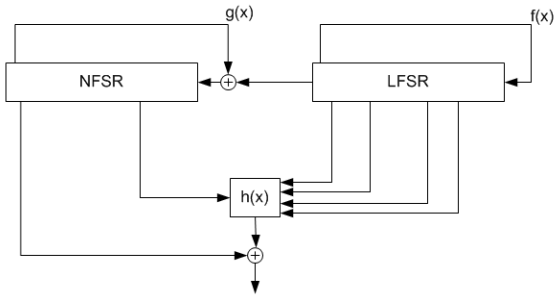

Fig. 1. The cipher.

The contents of the two shift registers represent the state of the cipher. From this state, 5 variables are taken as input to a boolean function, $h(x)$. This filter function is chosen to be balanced, correlation immune of the first order and has algebraic degree 3. The nonlinearity is the highest possible for these functions, namely 12. The input is taken both from the LFSR and from the NFSR. The function is defined as

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

where the variables x_0, x_1, x_2, x_3 and x_4 corresponds to the tap positions s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} and b_{i+63} respectively. The output of the filter function is masked with the bit b_i from the NFSR to produce the keystream.

2.1 Key Initialization

Before any keystream is generated the cipher must be initialized with the key and the IV. Let the bits of the key, k, be denoted $k_i,\ 0 \leq i \leq 79$ and the bits of the IV be denoted $IV_i,\ 0 \leq i \leq 63$. The initialization of the key is done as follows. First load the NFSR with the key bits, $b_i = k_i,\ 0 \leq i \leq 79$, then load the first 64 bits of the LFSR with the IV, $s_i = IV_i,\ 0 \leq i \leq 63$. The remaining bits of the LFSR are filled with ones, $s_i = 1,\ 64 \leq i \leq 79$. Because of this the LFSR cannot be initialized to the all zero state. Then the cipher is clocked 160 times without producing any running key. Instead the output of the filter function, $h(x)$, is fed back and xored with the input, both to the LFSR and to the NFSR, see Fig. 2.

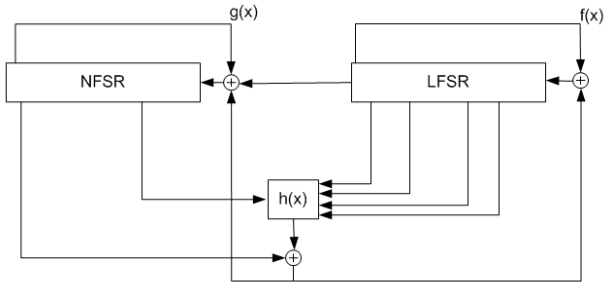

Fig. 2. The key initialisation.

3 Design Criteria

The design of the cipher is chosen to be as simple as possible for a hardware implementation. The security requirements correspond to a computational complexity of 2^{80}, equivalent to an exhaustive key search. To meet this requirement it is necessary to build the cipher with a memory of 160 bits. Implementing 160 memory bits in hardware can be seen a lower bound for the complexity. To develop a small hardware design we have to focus on minimizing the functions that are used together with this memory. The functions used need to be small in order to save gates but still large enough to provide high security. It is well known that an LFSR with primitive feedback polynomial of degree d produces an output with period $2^d - 1$. The LFSR in the cipher is of size 80 and since the feedback polynomial is primitive it guarantees that the period is at least $2^{80} - 1$. Because of the NFSR and the fact that the input to this is masked with the output of the LFSR the exact period will depend on the key and the IV used. The input to the NFSR is masked with the output of the LFSR in order to make

sure that the NFSR state is balanced. The nonlinear feedback is also balanced since the term x^{80} only appears linearily.

The filter function is quite small, only 5 variables and nonlinearity 12. However, this is compensated by the fact that one of the inputs is from the NFSR. The input bit from the NFSR will depend nonlinearily on other bits in the state, both from the LFSR and from the NFSR.

In the key initialization phase the goal is to scramble the contents of the shift registers before the running key is generated. The number of clockings is a tradeoff between security and speed. If the cipher is to be reinitialized often with a new IV, then the efficiency of the initialization is a possible bottleneck. Before initialization the LFSR contains the IV and 16 ones. For initialization with two different IVs, differing by only one bit, the probability that a shift register bit is the same for both initializations should be close to 0.5. Simulations show that this is achieved after 160 clockings. See section 6.4 for further discussion about this. Finally, no hidden weaknesses have been inserted by the designers.

3.1 Throughput Rate

Both shift registers are regularly clocked so the cipher will output 1 bit/clock. It is possible to increase the speed of the cipher at the expense of more hardware. This can very easily be done by just implementing the feedback functions, $f(x)$ and $g(x)$ and the filter function, $h(x)$ several times. In order to simplify this implementation, the last 15 bits of the shift registers, $s_i,\ 65 \leq i \leq 79$ and $b_i,\ 65 \leq i \leq 79$ are not used in the feedback functions or in the input to the filter function. This allows the speed to be easily multipled by up to 16 if a sufficient amount of hardware is available. An example of the implementation when the speed is doubled can be seen in Fig. 3. Naturally, the shift registers also need to be implemented such that each bit is shifted t steps instead of one when the speed is increased by a factor t. By increasing the speed by a factor 16, the cipher outputs 16 bits/clock. Since, in the key initialization, the cipher is clocked 160 times, the possibilities to increase the speed is limited to factors ≤ 16 that are divisible by 160. The number of clockings used in the key initialization is then $160/t$. Since the filter and feedback functions are small, it is quite feasible to increase the throughput in this way.

4 Strengths and Limitations

The design of a cipher needs to be focused on some specific properties. It is not possible to have a design that is perfect for all purposes i.e., processors of all word lengths, all hardware applications, all memory constraints etc. Grain is designed to be very small in hardware, using as few gates as possible while maintaining high security. The cipher is intended to be used in environments where gate count, power consumption and memory needs to be very small. While Grain is still possible to use in general application software, there are several ciphers that are designed with software efficiency in mind and thus are more appropriate

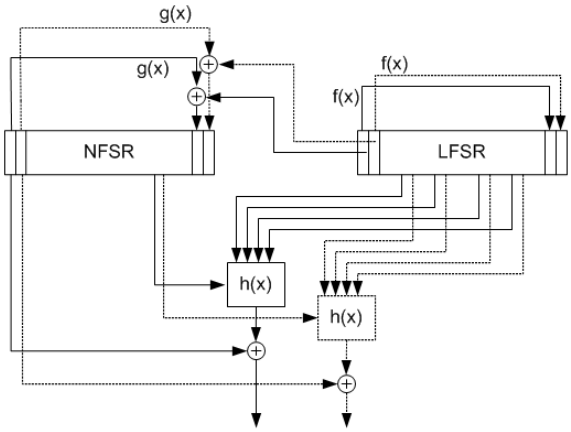

Fig. 3. The cipher when the speed is doubled.

when high speed in software is required. Because of this it does not make sense to compare the software performance of Grain to other ciphers. To emphasize the focus on hardware, no software speed measurements have been conducted.

The basic implementation has rate 1 bit/clock cycle. The speed of a word oriented cipher is typically higher since the rate is then 1 word/clock. Grain is bit oriented due to the high focus on small hardware complexity and this has been compensated by the possibility to increase the speed at the cost of more hardware. This allows a vendor to choose how fast the cipher should be according to the amount of hardware available in the product produced.

5 Hardware Complexity

To get some practical indications on complexity and other important features of a possible hardware implementation of the stream cipher, we performed a design based on standard FPGA architectures.

Starting with Fig. 1 (normal operating mode) and Fig. 2 (key initialisation) we added a third mode (loading key bits into NFSR and IV into LFSR, as described in Sect. 2.1). This whole circuit was described in VHDL (about 300 lines of code) depending on the parameter t as defined in Sect. 3.1. The ALTERA MAX 3000A family was choosen since we have most experience with the associated design equipment and it is seen as adequate for this purpose. MAX 3000A is a low end product using flash Memory as storage for the programming data; i.e. these data are persistent, and no loading procedure is necessary as with RAM-based FPGAs. The EPM3256 is the smallest chip of this family which will meet our requirements (more than 160 flipflops and some combinatorial logic). Using the ALTERA Quartus design tool, we carried out logical synthesis, place/route

and post-layout timing analysis. We found that $t \leq 4$ fits into the EPM3256, leading to a usage of about 90% of the 256 available macrocells. The maximum clock frequency is in the range of 35–50 MHz, depending on the operating mode and the output interface. Also $t = 8$ fits into this chip, but the maximum clock frequency was limited to 30 MHz. The number of output bits per second is t times clock frequency.

In order to make a fair comparison between different ciphers, the implementations has to be tested on the same FPGA. To highlight the performance difference between different FPGAs we have simulated our design on two additional FPGA families, namely the ALTERA MAX II and ALTERA Cyclone. These two allowed the cipher to be clocked at higher speed and it also allowed an implementation of the cipher when the speed was increased 16 times the original speed, i.e. when $t = 16$. It should be mentioned that there are other manufacturers of FPGAs (e.g. Actel, Xilinx), which may offer devices that will meet all requirements too at lower prices. Some products are including security mechanisms, prohibiting reverse engineering of a programmed chip.

The gate count for a function varies depending on the complexity and functionality. The numbers are no natural constants and will depend on the implementation in an actual chip. We have chosen a gate count of 8 for a flip flop. This figure ensures enough functionality for our application. Table 1 lists the factors chosen in our implementation.

Table 1. The gate count used for different functions.

Function	*Gate Count*
D flip flop	8
NAND2	1
NAND3	1.5
NAND4	2
NAND5	2.5
NAND6	3
XOR2	2.5
MUX3	5

In our design we have calculated the gate count for $t = 1, 2, 4, 8$ and 16. Table 2 shows the gate count and the corresponding throughput for the 3 different FPGA/CPLDs. More details regarding the figures in the hardware implementation can be found in Appendix A.

This gate count can be compared to other hardware oriented stream ciphers, e.g. E0 used in Bluetooth and A5/1 used in GSM. Using figures taken from [2], the gate count for E0 is about the same as for Grain. A5/1 has a gate count of approximately half. In all 3 ciphers, most of the gates are used for memory implementation. Grain, E0 and A5/1 use 160, 128 and 64 bits memory respectively. Moreover, the throughput of Grain also compares favourably to E0 and

Table 2. The gate count and throughput of Grain for $t = 1, 2, 4, 8$ and 16.

t	*Gate Count*	*Throughput*		
		MAX 3000A	MAX II	Cyclone
1	1435	49 Mbit/s	200 Mbit/s	282 Mbit/s
2	1607	98.4 Mbit/s	422 Mbit/s	576 Mbit/s
4	1950	196 Mbit/s	632 Mbit/s	872 Mbit/s
8	2636	240 Mbit/s	1184 Mbit/s	1736 Mbit/s
16	4008	—	2128 Mbit/s	3136 Mbit/s

A5/1, mostly due to the fact that it can be increased efficiently with just a small increase in gate count. Both E0 and A5/1 have been proven to be very insecure. In [8], an attack against E0 using 2^{35} frames and computational complexity 2^{40} was shown. This attack is on the borderline of being practical. Also, several attacks against A5/1 have been shown, see e.g. [1, 4, 9]. Grain has been designed to provide much better security than both E0 and A5/1 while maintaining a low gate count.

6 Cryptanalysis

In this section we consider some general attacks on stream ciphers and investigate to which extent they can be applied to Grain. Resistance against all known cryptanalytic attacks is the most important property of a new cipher. There should be no attack faster than exhaustive key search. Initial cryptanalytic attempts against the cipher show the following.

6.1 Correlations

Due to the statistical properties of maximum-length LFSR sequences, the bits in the LFSR are (almost) exactly balanced. This may not be the case for a NFSR when it is driven autonomously. However, as the feedback $g(x)$ is xored with a LFSR-state, the bits in the NFSR are balanced. Moreover, recall that g is a balanced function. Therefore, the bits in the NFSR may be assumed to be uncorrelated to the LFSR bits.

The function h is chosen to be correlation immune of first order. This does not preclude that there are correlations of the output of $h(x)$ to sums of inputs. As one input comes from the NFSR and as $h(x)$ is xored with a state bit of the NFSR, correlations of the output of the generator to sums of LFSR-bits will be so small that they will not be exploitable by (fast) correlation attacks.

6.2 Algebraic Attack

A filter generator alone with output function $h(x)$ of degree only three would be very vulnerable to algebraic attacks. On the other hand, algebraic attacks

will not work for solving for the initial 160-bit state of the full generator, as the update function of the NFSR is nonlinear, and the later state bits of the NFSR as a function of the inital state bits will have varying but large algebraic degree. Using key initialization, it may be possible to express the output of the generator as a function of state bits of the LFSR alone. As the filter function $h(x)$ has one input coming from the NFSR, and $h(x)$ is xored with a NFSR-state bit, the algebraic degrees of the output bits when expressed as a function of LFSR-bits, are large in general, and varying in time. This will defeat algebraic attacks.

6.3 Time/Memory/Data Tradeoff Attack

The cost of time/memory/data tradeoff attacks on stream ciphers is $O(2^{n/2})$, where n is the number of inner states of the stream cipher, [3]. To obey the margins set by this attack, $n = 160$ has been chosen. It is known that stream ciphers with low sampling resistance have tradeoff attacks with fewer table lookups and a wider choice of parameters, [3]. The sampling resistance of $h(x)$ is reasonable: This function does not become linear in the remaining variables by fixing less than 3 of its 5 variables. Similarly, the variables occuring in monomials of $g(x)$ are sufficiently disjoint. Hence the resulting sampling resistance is large, and thus time/memory/data tradeoff attacks are expected to have complexity not lower than $O(2^{80})$.

6.4 Chosen-IV Attack

A necessary condition for defeating differential-like or statistical chosen-IV attacks is that the initial states for any two chosen IV's (or sets of IV's) are algebraically and statistically unrelated. The number of cycles in key initialization has been chosen so that the Hamming weight of the differences in the full initial 160-bit state for two IV's after initialization is close to random. This should prevent chosen-IV attacks.

It may be tempting to improve the efficiency of the key initialization by just decreasing the number of initial clockings. Indeed, after only 80 clocks, all bits in the state will depend on both the key and the IV. However, in a chosen-IV attack it is possible to reinitialize the cipher with the same key but with an IV that differs in only one position from the previous IV. Consider the case when the number of initial clockings is 80 and the last bit of the IV is flipped i.e., s_{63} is flipped. This is the event that occurs if the IV is chosen as a sequence number. Looking at the difference of the states after initialization it is clear that several positions will be predictable. The bit s_{63} is not used in the feedback or in the filter function, hence, the first register update will be the same in both cases. Consequently, the bit s_0 will be the same in both initializations. In the next update, the flipped bit will be in position s_{62}. This position is used in the linear feedback of the LFSR, and consequently the bit s_1 will always be different for the two initializations. Similar arguments can be used to show that the difference in the state will be deterministic in more than half of the 160 state bits. This deterministic difference in the state can be exploited

in a distinguishing attack. Let $\underline{x} = x_0, x_1, x_2, x_3, x_4$ be the input variables to $h(x)$ after the first initialization and let $\underline{x}_\Delta = x_0, x_1, x_2, x_3, x_4$ be the input variables to $h(x)$ after the second initialization. Now, compute the distribution of $P(\underline{x}, \underline{x}_\Delta)$. If this distribution is biased, it is likely that the distribution of the difference in the first output bit,

$$P(h(\underline{x}) \oplus h(\underline{x}_\Delta)),$$

is biased. Assume that

$$P(h(\underline{x}) \oplus h(\underline{x}_\Delta) = 0) = 1/2 + \epsilon,$$

then the number of initializations we need will be in the order of $1/\epsilon^2$. This attack can be optimized by calculating which output bit will give the highest bias since it is not necessarily the bits in the registers corresponding to the input bits of $h(x)$ that have deterministic difference after the initializations. This attack shows that it is preferred that the probability that any state bit is the same after initialization with two different IVs should be close to 0.5. As with the case of 80 initialization clocks, it is easy to show that after 96, 112 and 128 there are also state bits that will always be the same or that will always differ.

6.5 Fault Attack

Amongst the strongest attacks conceivable on any cipher, are fault attacks. Fault attacks against stream ciphers have been initiated in [7], and have shown to be efficient against many known constructions of stream ciphers. This suggests that it is hard to completely defeat fault attacks on stream ciphers. In the scenario in [7] it is assumed that the attacker can apply some bit flipping faults to one of the two feedback registers at his will. However he has only partial control over their number, location, and exact timing, and similarly on what concerns his knowledge. A stronger assumption one can make, is that he is able to flip a single bit (at a time instance, and thus at a location, he does not know exactly). In addition, he can reset the device to its original state and then apply another randomly chosen fault to the device. We adapt the methods in [7] to the present cipher. Thereby, we make the strongest possible assumption (which may not be realistic) that an attacker can induce a single bit fault in the LFSR, and that he is somehow able to determine the exact position of the fault. The aim is to study input-output properties for $h(x)$, and to derive information on the 5 inputs, out of known input-output pairs (similar as for S-boxes in differential cryptanalysis of DES). As long as the difference induced by the fault in the LFSR does not propagate to position b_{i+63}, the difference observed in the output of the cipher is coming from inputs of $h(x)$ from the LFSR alone. If an attacker is able to reset the device and to induce a single bit fault many times and at different positions that he can correctly guess from the output difference, we cannot preclude that he will get information about a subset of the state bits in the LFSR. Such an attack seems more difficult under the (more realistic) assumption that the fault induced affects several state bits at (partially) unknown positions, since

in this case it is more difficult to determine the induced difference from output differences.

Likewise, one can consider faults induced in the NFSR alone. These faults do not influence the contents of the LFSR. However, faults in the NFSR propopagate nonlinearly and their evolution will be harder to predict. Thus, a fault attack on the NFSR seems more difficult.

7 Conclusion

A new stream cipher, Grain, has been introduced. It is designed with small hardware implementation in mind. A complete description of the algorithm as well as a security analysis based on known attacks have been given. The construction is based on two shift registers, one with linear feedback and one with nonlinear feedback, and a nonlinear filter function. The key size is 80 bits and no attack with complexity better than exhaustive key search has been identified.

Grain is a bit oriented stream cipher producing 1 bit/clock in its simplest implementation. However, as an important feature, it is very easy to increase the rate up to 16 bits/clock if some additional hardware is used.

Acknowledgement

We are indebted to Werner Witz and Peter Steigmeier for carrying out an implementation of Grain in hardware and extracting the data as given in Section 5.

References

1. E. Barkan, E. Biham and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. Crypto 2003, Springer Verlag, LNCS 2729, Oct 2003, Pages 600–616
2. L. Batina, J. Lano, N. Mentens, S. B. Örs, B. Preneel and I. Verbauwhede. Energy, Performance, Area Versus Security Trade-offs for Stream Ciphers. In *The State of the Art of Stream Ciphers: Workshop Record, Brugge, Belgium, October 2004*, pages 302–310, 2004.
3. A. Biryukov, A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. Asiacrypt 2000, Springer Verlag, LNCS 1976, pp. 1–13.
4. A. Biryukov, A. Shamir and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC. FSE 2000, Springer Verlag, LNCS 1978, Jan 2001, pp. 1–13.
5. P. Ekdahl and T. Johansson. A New Version of the Stream Cipher SNOW. Selected Areas in Cryptography, SAC 2002, Springer Verlag, LNCS 2595, pp. 47–61, 2002.
6. P. Hawkes and G. Rose. Primitive Specification for SOBER-128. IACR ePrint Archive, `http://eprint.iacr.org/2003/81/`, 2003.
7. J.J. Hoch, A. Shamir. Fault Analysis of Stream Ciphers. CHES 2004, Springer Verlag, LNCS 3156, pp. 240–253.
8. Y. Lu and S. Vaudenay. Cryptanalysis of Bluetooth Keystream Generator Two-Level E0. Asiacrypt 2004, Springer Verlag, LNCS 3329, Nov 2004, pp 483–499.

9. A. Maximov, T. Johansson and S. Babbage. An Improved Correlation Attack on A5/1. Selected Areas in Cryptography, SAC 2004, Springer Verlag, LNCS 3357, Jan 2004, Pages 1 - 18.

A Hardware Figures

In Table 3 we summarize hardware figures when the implementation was simulated on three different FPGAs. A fair comparison between different implementations and different ciphers requires that the same FPGA family is targeted.

Table 3. Hardware related figures for Grain.

	Supplier	ALTERA	ALTERA	ALTERA
	FPGA/CPLD	MAX 3000A	MAX II	Cyclone
	Type	EPM3256ATC144-7	EPM570T100C3	EP1C3T100C6
	Total LABs [LE]	[256]	57	291
t=1	Gate Count	1435	1435	1435
	Max. Clock	49.2 MHz	200 MHz	282 MHz
	%LAB [LE] usage	[68%]	38%	8%
	Power drain	700 mW	365 mW	835 mW
	Throughput	49 Mbit/s	200 Mbit/s	282 Mbit/s
	Efficiency	70 Mbit/Joule	0.55 Gbit/Joule	0.38 Gbit/Joule
t=2	Gate Count	1607	1607	1607
	Max. Clock	49.2 MHz	211 MHz	288 MHz
	%LAB [LE] usage	[75%]	42%	10%
	Power drain	750 mW	395 mW	855 mW
	Throughput	98.4 Mbit/s	422 Mbit/s	576 Mbit/s
	Efficiency	131 Mbit/Joule	1.07 Gbit/Joule	0.67 Gbit/Joule
t=4	Gate Count	1950	1950	1950
	Max. Clock	49 MHz	158 MHz	218 MHz
	%LAB [LE] usage	[89%]	49%	12%
	Power drain	835 mW	330 mW	660 mW
	Throughput	196 Mbit/s	632 Mbit/s	872 Mbit/s
	Efficiency	235 Mbit/Joule	1.95 Gbit/Joule	1.32 Gbit/Joule
t=8	Gate Count	2636	2636	2636
	Max. Clock	30 MHz	148 MHz	217 MHz
	%LAB [LE] usage	[98%]	61%	11%
	Power drain	775 mW	350 mW	665 mW
	Throughput	240 Mbit/s	1184 Mbit/s	1736 Mbit/s
	Efficiency	310 Mbit/Joule	3.38 Gbit/Joule	2.6 Gbit/Joule
t=16	Gate Count		4008	4008
	Max. Clock		133 MHz	196 MHz
	%LAB [LE] usage		85%	19%
	Power drain		420 mW	625 mW
	Throughput		2128 Mbit/s	3136 Mbit/s
	Efficiency		5.06 Gbit/Joule	5.18 Gbit/Joule

0 The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, as well as by Gebert Rüf Stiftung, and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

B Test Vectors

```
Key:       00000000000000000000
IV:        0000000000000000
Keystream: 4c75acdcf238b81bb819


Key:       4444444444444444444
IV :       5555555555555555
Keystream: 6fc1eeb05d1f7d2809f3


Key:       aaaaaaaaaaaaaaaaaaaa
IV :       bbbbbbbbbbbbbbbb
Keystream: fbe066238078d96dc6af
```