The Grain Family of Stream Ciphers

Martin Hell[1], Thomas Johansson[1], Alexander Maximov[2], and Willi Meier[3,⋆]

[1] Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{martin,thomas}@eit.lth.se
[2] Ericsson AB, Lund, Sweden
alexander.maximov@ericsson.com
[3] FHNW, CH-5210 Windisch, Switzerland
willi.meier@fhnw.ch

Abstract. A new family of stream ciphers, Grain, is proposed. Two variants, a 80-bit and a 128-bit variant are specified, denoted Grain and Grain-128 respectively. The designs target hardware environments where gate count, power consumption and memory are very limited. Both variants are based on two shift registers and a nonlinear output function. The ciphers also have the additional feature that the speed can be easily increased at the expense of extra hardware.

When designing a cryptographic primitive there are many different properties that have to be addressed. These include e.g., speed and security. 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. 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. A radio-frequency identification (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. The Grain family of stream ciphers is designed to be very easy and small to implement in hardware.

Several recent LFSR based stream cipher proposals, see e.g., [1,2] 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. In the Grain ciphers, this issue has been

⋆ Supported by Hasler Foundation http://www.haslerfoundation.ch under project number 2005.

M. Robshaw and O. Billet (Eds.): New Stream Cipher Designs, LNCS 4986, pp. 179–190, 2008.

addressed by basing the 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. This property is not explicitly found in most other stream ciphers.

The proposed designs, denoted Grain (or more formally Grain Version 1 or Grain V1) and Grain-128, are bit oriented synchronous stream ciphers. The designs are 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 output function 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.

The first, unpublished, version of the cipher is denoted version 0. This version was cryptanalyzed in [3,4,5]. The design of version 0 will not be given in this paper but the attack will be discussed in Section 3.1.

The paper is organized as follows. Section 1 provides a detailed description of the Grain and Grain-128 designs. The possibility to easily increase the throughput is discussed in Section 2. The security of Grain is discussed in Section 3 together with a motivation for the different design parameters. Section 4 concludes the paper.

1 Design Specifications

This section specifies the details of the designs of both Grain and Grain-128. Both ciphers follow the same design principle. They consist of three main building blocks, namely an LFSR, an NFSR and an output function. The contents of the two shift registers represent the state of the cipher and their sizes are $|K|$ bits each, where K is the key. In the following, the content of the LFSR is denoted $S_t = s_t, s_{t+1}, \ldots, s_{t+|K|-1}$ and the content of the NFSR is denoted $B_t = b_t, b_{t+1}, \ldots, b_{t+|K|-1}$. The output function, denoted $H(B_t, S_t)$ consists of two parts. A nonlinear Boolean function $h(x)$ and a set of linear terms added to $h(x)$. The output of $H(B_t, S_t)$ is the keystream bit z_t. A general overview of the design is given in Fig. 1.

1.1 Grain - Design Parameters

The keysize of Grain is $|K| = 80$ bits and the cipher supports an IV of size $|IV| = 64$ bits. The feedback polynomial of the LFSR, denoted $f(x)$ is a primitive polynomial of degree 80. It is defined as

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

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

$$s_{t+80} = s_{t+62} \oplus s_{t+51} \oplus s_{t+38} \oplus s_{t+23} \oplus s_{t+13} \oplus s_t. \quad (2)$$

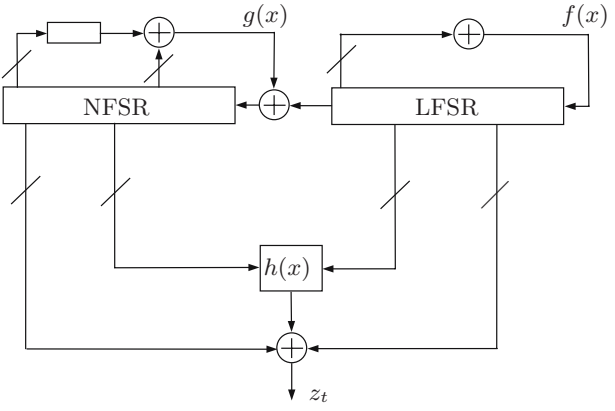

Fig. 1. Overview of the different design blocks in the Grain family of stream ciphers

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

$$\begin{aligned} g(x) = 1 &+ x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + \\ &+ x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + \\ &+ x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + \\ &+ x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}. \end{aligned} \tag{3}$$

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

$$\begin{aligned} b_{t+80} = s_t &\oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \oplus b_{t+37} \oplus b_{t+33} \oplus b_{t+28} \oplus \\ &\oplus b_{t+21} \oplus b_{t+14} \oplus b_{t+9} \oplus b_t \oplus b_{t+63}b_{t+60} \oplus b_{t+37}b_{t+33} \oplus \\ &\oplus b_{t+15}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+45} \oplus b_{t+33}b_{t+28}b_{t+21} \oplus \\ &\oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \oplus \\ &\oplus b_{t+63}b_{t+60}b_{t+21}b_{t+15} \oplus b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \oplus \\ &\oplus b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \oplus b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}. \end{aligned} \tag{4}$$

From the two registers, 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 function is defined as

$$\begin{aligned} &h(x) = h(x_0, x_1, \ldots, x_4) = \\ &= x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4 \end{aligned} \tag{5}$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to the tap positions s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64} and b_{t+63} respectively. The output function $H(B_t, S_t)$ is given by

$$z_t = H(B_t, S_t) = \bigoplus_{j \in \mathcal{A}} b_{t+j} \oplus h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}) \tag{6}$$

where $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$.

Cipher 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 the NFSR and LFSR are loaded with key and IV bits as

$$\begin{cases} b_i = k_i, & 0 \leq i \leq 79, \\ s_i = IV_i, & 0 \leq i \leq 63. \end{cases} \tag{7}$$

The remaining bits of the LFSR are filled with ones, $s_i = 1,\ 64 \leq i \leq 79$. Then the cipher is clocked 160 times without producing any keystream. Instead the output function is fed back and xored with the input, both to the LFSR and to the NFSR, see Fig. 2.

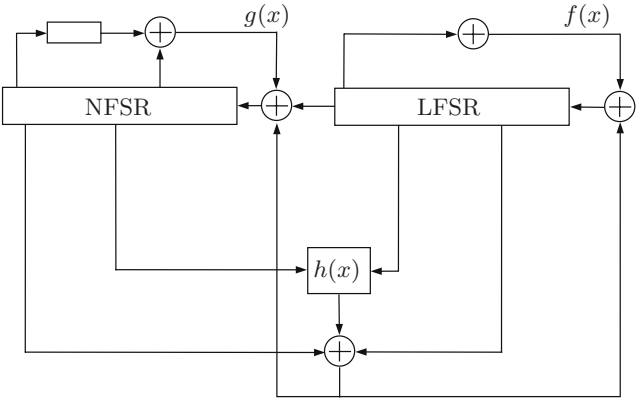

Fig. 2. Overview of the key initialization

1.2 Grain-128 — Design Parameters

Grain-128 supports a keysize of $|K| = 128$ bits, as suggested by the name. The size of the IV is specified to be $|IV| = 96$ bits. The feedback polynomial of the LFSR, $f(x)$, is a primitive polynomial of degree 128. It is defined as

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}. \tag{8}$$

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

$$s_{t+128} = s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96}. \tag{9}$$

The nonlinear feedback polynomial of the NFSR, $g(x)$, is the sum of one linear and one bent function. It is defined as

$$\begin{aligned} g(x) = 1 &+ x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + \\ &+ x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}. \end{aligned} \tag{10}$$

Again, we also write the corresponding update function of the NFSR. In the update function below, note that the bit s_t which is masked with the input to the NFSR is included, while omitted in the feedback polynomial.

$$\begin{aligned} b_{t+128} = s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus \\ \oplus b_{t+11}b_{t+13} \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus \\ \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84}. \end{aligned} \tag{11}$$

From the state, nine variables are taken as input to a Boolean function, $h(x)$. Two inputs to $h(x)$ are taken from the NFSR and seven are taken from the LFSR. This function is of degree $\deg(h(x)) = 3$ and very simple. It is defined as

$$h(x) = h(x_0, x_1, \ldots, x_8) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8 \tag{12}$$

where the variables x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7 and x_8 correspond to the tap positions b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79} and s_{t+95} respectively. The output function $H(B_t, S_t)$ is defined as

$$z_t = H(B_t, S_t) = \bigoplus_{j \in \mathcal{A}} b_{t+j} \oplus h(x) \oplus s_{t+93}, \tag{13}$$

where $\mathcal{A} = \{2, 15, 36, 45, 64, 73, 89\}$.

Cipher Initialization: The initialization is very similar to the initialization of the 80-bit variant of the cipher. The bits of the key K, denoted k_i, $0 \leq i \leq 127$, and the bits of the IV, denoted IV_i, $0 \leq i \leq 95$, are loaded into the NFSR and LFSR respectively as

$$\begin{cases} b_i = k_i, & 0 \leq i \leq 127, \\ s_i = IV_i, & 0 \leq i \leq 95. \end{cases} \tag{14}$$

The last 32 bits of the LFSR are filled with ones, $s_i = 1$, $96 \leq i \leq 127$. After loading key and IV bits, the cipher is clocked 256 times without producing any keystream. The output function is fed back and xored with the input, both to the LFSR and to the NFSR.

2 Throughput Rate

It is possible to increase the throughput rate of the Grain ciphers by adding some additional hardware. This is an important feature of the Grain family of stream ciphers compared to many other stream ciphers. Increasing the speed can very easily be done by just implementing the feedback functions, $f(x)$ and $g(x)$, and the output function several times. In order to simplify this implementation, the last 15 bits in Grain and the last 31 bits in Grain-128 of the shift registers are not used in the feedback functions or in the input to the output function. I.e., s_i, $65 \leq i \leq 79$ and b_i, $65 \leq i \leq 79$ in Grain and s_i, $97 \leq i \leq 127$ and b_i, $97 \leq i \leq 127$ in Grain-128 are not used in the three functions. This allows the speed to be easily multiplied by up to 16 for Grain and 32 for Grain-128 if

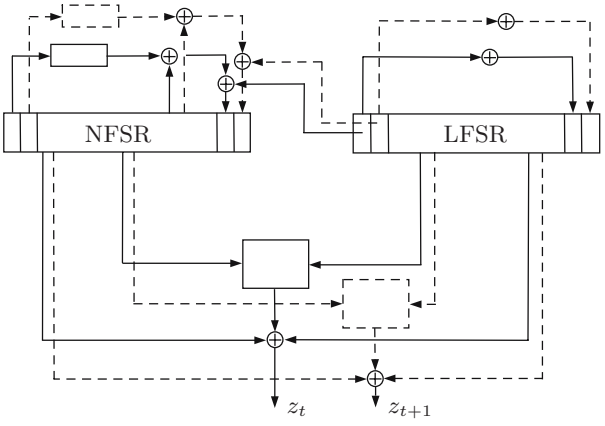

Fig. 3. Implementation of Grain which outputs 2 bits/clock

a sufficient amount of hardware is available. An overview 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 δ steps instead of one when the speed is increased by a factor δ. Since, in the key initialization, the cipher is clocked 160 times (Grain) or 256 times (Grain-128), the possibilities to increase the speed is limited to factors that are divisors of 160 or 256 respectively. The number of clockings needed in the key initialization phase is then $160/\delta$ or $256/\delta$. Since the output and feedback functions are small, it is quite feasible to increase the throughput in this way.

3 Security and Design Choices

In this section we give a security analysis of the construction and motivate the different design choices.

3.1 Linear Approximations

Attacking Grain using linear approximations of the two nonlinear functions turned out to be successful on the first version of Grain, (version 0). This attack was discovered by several independent researchers and the details can be found in [3,4,5]. Some design choices in the current versions are influenced by this attack. In this subsection, we temporarily switch to the notation $s(t)$ instead of s_t as previously used to denote a value at time t. We also use the notation $x \stackrel{p}{=} y$ meaning that $\Pr(x = y) = p$.

With a slight abuse of notation, let us rewrite the update function of the NFSR as

$$0 = g(B_t) \oplus s(t). \tag{15}$$

Let the weight of a binary linear function ℓ, denoted $w(\ell)$, be the number of terms in the function. I.e., if $\ell = \bigoplus_{i=0}^{n} c_i x_i$, then

$$w(\ell) = |\{i \in 0..n : c_i = 1\}|. \tag{16}$$

Assume that we have found a linear approximation $\ell_g(t)$ of $g(B_t)$ i.e.,

$$\ell_g(t) = \bigoplus_{i=0}^{w(\ell_g)-1} b(t+\phi_i), \tag{17}$$

where $\phi_0, \phi_1, \ldots, \phi_{w(\ell_g)-1}$ denote the positions in the NFSR that are present in the linear approximation. The bias of $\ell_g(t)$ is denoted ε_g, i.e.,

$$\Pr(\ell_g(t) = g(B_t)) = \Pr(\ell_g(t) = s(t)) = \frac{1}{2}(1+\varepsilon_g), \quad 0 < |\varepsilon_g| \leq 1. \tag{18}$$

Similarly, a linear approximation $\ell_H(t)$ of the output function $H(B_t, S_t)$ can be found. Let $w_N(\ell)$ and $w_L(\ell)$ be the number of terms from the NFSR and from the LFSR respectively. Then $\ell_H(t)$ can be written as

$$\ell_H(t) = \bigoplus_{i=0}^{w_N(\ell_H)-1} b(t+\xi_i) \oplus \bigoplus_{i=0}^{w_L(\ell_H)-1} s(t+\psi_i), \tag{19}$$

where $\xi_0, \xi_1, \ldots, \xi_{w_N(\ell_H)-1}$ and $\psi_0, \psi_1, \ldots, \psi_{w_L(\ell_H)-1}$ determine the location of the taps in the NFSR and LSFR used in the linear approximation. The bias of (19) is denoted ε_H, i.e.,

$$\Pr(\ell_H(t) = z(t)) = \frac{1}{2}(1+\varepsilon_H), \quad 0 < |\varepsilon_H| \leq 1. \tag{20}$$

Now, sum up the keystream bits determined by ϕ_i in (17),

$$\begin{aligned} z(t+\phi_0) \oplus z(t+\phi_1) \oplus \ldots \oplus z(t+\phi_{w(\ell_g)-1}) &\stackrel{p}{=} \\ \ell_H(t+\phi_0) \oplus \ell_H(t+\phi_1) \oplus \ldots \oplus \ell_H(t+\phi_{w(\ell_g)-1}). & \end{aligned} \tag{21}$$

Using the piling-up lemma, the relation (21) holds with probability $p = 1/2(1+\varepsilon_H^{w(\ell_g)})$. The terms on the right hand side of (21) will consist of $w_N(\ell_H) \cdot w(\ell_g)$ terms from the NFSR and $w_L(\ell_H) \cdot w(\ell_g)$ terms from the LFSR. All terms from the NFSR can now be approximated using (17) resulting in a relation involving only keystream bits and LFSR bits as

$$\bigoplus_{i=0}^{w(\ell_g)-1} z(t+\phi_i) \stackrel{p'}{=} \bigoplus_{i=0}^{w(\ell_g)-1} \bigoplus_{j=0}^{w_L(\ell_H)-1} s(t+\phi_i+\psi_j) \oplus \bigoplus_{i=0}^{w_N(\ell_H)-1} s(t+\xi_i), \tag{22}$$

which holds with probability $p' = 1/2(1+\varepsilon_{tot})$ with

$$\varepsilon_{tot} = \varepsilon_g^{w_N(\ell_H)} \cdot \varepsilon_H^{w(\ell_g)}. \tag{23}$$

From this point there are several possibilities for attacks. By finding a multiple of the LFSR feedback polynomial of weight 3, a distinguishing attack can be mounted. The expected degree of this multiple would be around $2^{|K|/2}$ (see e.g., [6]). Combining the keystream bits given by the multiple and using the approximation that $1/\varepsilon^2$ samples are needed in the distinguisher, about

$$N = 2^{|K|/2} + \frac{1}{\varepsilon_{tot}^6} \tag{24}$$

keystream bits are required in the attack.

Another approach is to try to recover the state of the LFSR. An obvious way of doing this is to exhaustively search the state and determine which state gives the bias in (23). In this case, only about

$$N = \frac{|K| \cdot 2 \ln 2}{\varepsilon_{tot}^2} \tag{25}$$

keystream bits are needed. This expression can be derived from the capacity of a binary symmetric channel, see e.g. [7]. Since the size of the LFSR is the same as the key size, this method is obviously more expensive than exhaustive key search. A faster algorithm was given in [4], where they generate more equations of the form (22). By only using equations of a certain form, and by using the Fast Walsh Transform, the attack complexity could be made significantly lower. We refer to [4] for more details on this attack.

Due to this attack, the parameters of the original version of Grain were changed. A higher resiliency was added to the NFSR feedback function, increasing $w(\ell_g)$ and several linear terms from the NFSR were added to the output function, increasing $w_N(\ell_H)$.

The design of Grain-128 is inspired by the analysis in this section. Thus, the NFSR feedback function should satisfy the following three criteria

- *High resiliency*, implying many terms in the linear approximation (high $w(\ell_g)$). This can be achieved by adding several linear terms to the function. Each linear term will increase the resiliency by one.
- *High nonlinearity*, implying small bias of the linear approximations (small ε_g). This can be achieved by using a bent function, i.e., a function with maximum nonlinearity.
- *Small hardware implementation*, implying that the design is attractive in low-cost implementations.

A well-known n-variable bent function is the function $x_1x_2 \oplus x_3x_4 \oplus \ldots \oplus x_{n-1}x_n$. This function is also very small in hardware. Using $n = 14$ and adding 5 linear terms gives a 4-resilient Boolean function with nonlinearity 260096. The best linear approximations have bias $\varepsilon_g = 2^{-7}$ and $w(\ell_g) \geq 5$.

The output function has the same design criteria as the NFSR feedback function. However, to increase the algebraic degree it has a term of degree 3. It has nonlinearity 61440 and resiliency 7. The best linear approximations have bias $\varepsilon_H = 2^{-4}$ and $w_N(\ell_H) \geq 7$.

3.2 Time-Memory Tradeoff Attacks

It is well known that the state of a stream cipher must be at least twice the key size in order to prevent time-memory tradeoff attacks [8,9,10]. Both the LFSR and NFSR are of size $|K|$ bits, and thus the state is exactly twice the key size. Since Grain is designed to be as small as possible in hardware, no extra state bits are added to the design. The state is relatively expensive to implement in hardware and it is important to keep it as small as possible. In [11] it was noted that the initialization process of a stream cipher could be seen as a one-way function i.e., the function taking the key K and the IV IV as input and outputs the first $|K|+|IV|$ bits of the keystream. In this case the search space is $2^{|K|+|IV|}$ and new data is generated by repeated initializations of the cipher. If we allow a preprocessing time P that is higher than exhaustive key search $2^{|K|}$, then it is possible to have an attack with real time complexity lower than exhaustive key search. Table 1 gives attack complexities for Grain and Grain-128 in the time-memory tradeoff setting of [11] i.e., $N^2 = TM^2D^2$ and $P = N/D$, where N is the search space, T the computational complexity in the realtime phase, D the number of initializations, M the amount of memory and P the computational complexity in the preprocessing phase. If $|IV| < \frac{1}{2}|K|$ then it is possible to have the preprocessing time also smaller than exhaustive key search. In this case we need to initialize with several different keys and we will only retrieve one of these keys in the real time phase. In the Grain ciphers $|IV| > \frac{1}{2}|K|$ so this is not applicable here.

Table 1. Time-Memory tradeoff attack with real time complexity T, D initializations, M memory words and preprocessing time P

Attack Complexities				
	T	D	M	P
Grain	2^{80}	2^{40}	2^{64}	2^{104}
	2^{72}	2^{36}	2^{72}	2^{108}
Grain-128	2^{128}	2^{64}	2^{96}	2^{160}
	2^{112}	2^{56}	2^{112}	2^{168}

3.3 Algebraic Attacks

Algebraic attacks can be very successful on nonlinear filter generators. Especially if the output function is of very low degree. Grain is very similar to a nonlinear filter. However, the introduction of the NFSR in the design will defeat all algebraic attacks known today. Since the update function of the NFSR is nonlinear, the later state bits of the NFSR as a function of the initial state bits will have varying but large algebraic degree. As the output function has several inputs from the NFSR, the algebraic degree of the keystream bits expressed as functions of key bits will be large in general. This will defeat known algebraic attacks.

3.4 Chosen-IV Attacks

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. Considering the 80-bit variant of Grain, 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 $\mathbf{x}$ be the input variables to the output function, H, after the first initialization and let $\mathbf{x}_\Delta$ be the input variables to the output function after the second initialization. Now, compute the distribution of $\Pr(\mathbf{x}, \mathbf{x}_\Delta)$. If this distribution is biased, it is possible[1] that the distribution of the difference in the first output bit,

$$\Pr(H(\mathbf{x}) \oplus H(\mathbf{x}_\Delta)), \tag{26}$$

is biased. Assume that

$$\Pr(H(\mathbf{x}) \oplus H(\mathbf{x}_\Delta) = 0) = 1/2(1 + \varepsilon), \quad 0 < |\varepsilon| \leq 1. \tag{27}$$

then the number of initializations we need will be in the order of $1/\varepsilon^2$. 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 $H(\mathbf{x})$ 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.

[1] It is possible, but maybe not very likely. One unbiased linear variable is enough to make the output unbiased.

It is possible to reduce the required number of initial clockings by loading the NFSR and LFSR differently. If each entry of the registers is loaded with the xor of a few key and IV bits and each key and IV bit influences the loading of several entries, differences in the IV will propagate faster. The reason for not doing this is mainly that all the extra xors needed would make the cipher larger in hardware.

3.5 Fault Attacks

Amongst the strongest attacks conceivable on any cipher, are fault attacks. Fault attacks against stream ciphers have been initiated in [12], 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 [12] 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 [12] 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 $H(B_t, S_t)$, and to derive information on the inputs. As long as the difference induced by the fault in the LFSR does not propagate to position b_{t+63} in Grain or b_{t+95} in Grain-128, the difference observed in the output of the cipher is coming from inputs of $H(B_t, S_t)$ 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 propagate nonlinearly and their evolution will be harder to predict. Thus, a fault attack on the NFSR seems more difficult.

4 Conclusions

In this paper we introduced the Grain family of stream ciphers. Two different versions, denoted Grain and Grain-128, have been specified. The designs target hardware environments where small area is of high importance. The basic implementation is very small but outputs only one bit/clock. An important feature in

the Grain ciphers is the possibility to easily increase the throughput by adding some extra hardware. This is done by simply implementing the relatively small feedback and output functions several times. This flexibility makes the Grain ciphers attractive for a wide range of applications spanning from the most demanding in terms of small hardware area to applications requiring a very high throughput.

References

1. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
2. Hawkes, P., Rose, G.: Primitive specification for SOBER-128. Cryptology ePrint Archive, Report 2003/081 (2003), http://eprint.iacr.org/
3. Maximov, A.: Cryptanalysis of the Grain family of stream ciphers. In: Lin, F., Lee, D., Lin, B., Shieh, S., Jajodia, S. (eds.) ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006), pp. 283–288. ACM, New York (2006)
4. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of Grain. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 15–29. Springer, Heidelberg (2006)
5. Khazaei, S., Hassanzadeh, M., Kiaei, M.: Distinguishing attack on Grain. eSTREAM, ECRYPT Stream Cipher Project, Report2005/071 (2005), http://www.ecrypt.eu.org/stream
6. Golić, J.: Computation of low-weight parity-check polynomials. Electronic Letters 32(21), 1981–1982 (1996)
7. Hell, M.: On the design and analysis of stream ciphers. PhD thesis, Lund University (2007)
8. Babbage, S.: A space/time tradeoff in exhaustive search attacks on stream ciphers. In: European Convention on Security and Detection. IEE Conference Publication, vol. 408 (1995)
9. Golić, J.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
10. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
11. Hong, J., Sarkar, P.: New applications of time memory data tradeoffs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005)
12. Hoch, J., Shamir, A.: Fault analysis of stream ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004)