An AEAD variant of the Grain stream cipher

Martin Hell[1], Thomas Johansson[1], Willi Meier[2], Jonathan Sönnerup[1], and Hirotaka Yoshida[3]

[1] Dept. of Electrical and Information Technology, Lund University, Sweden
{martin.hell,thomas.johansson,jonathan.sonnerup}@eit.lth.se
[2] FHNW, Windisch, Switzerland
willi.meier@fhnw.ch
[3] Cyber Physical Security Research Center (CPSEC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
hirotaka.yoshida@aist.go.jp

Abstract. A new Grain stream cipher, denoted Grain-128AEAD is presented, with support for authenticated encryption with associated data. The cipher takes a 128-bit key and a 96-bit IV and produces a pseudo random sequence that is used for encryption and authentication of messages. The design is based on Grain-128a but introduces a few changes in order to increase the security and protect against recent cryptanalysis results. The MAC is 64 bits, as specified by the NIST requirements in their lightweight security standardization process.

Keywords: Grain · Stream Cipher · AEAD · NIST

1 Introduction

Due to widespread usage of Internet of Things (IoT) technology, the need of protection from security threats on resource-constrained devices has been continuously growing. Since 2003, the cryptography community has already recognized the importance of this need, and researchers and developers have focused on cryptography tailored to limited computation resources in hardware and software implementations. This has resulted in opening up a new subfield of cryptography, namely, lightweight cryptography, which led to the launch of the eSTREAM project. This project running from 2004 to 2008 can be viewed as the most important research activity in the area of lightweight stream ciphers. The eSTREAM portfolio contains four software-oriented ciphers and three hardware-oriented ciphers.

From an industrial point of view, it has been widely recognized that *maturity* is important regarding deployment of cryptographic mechanisms. In fact, the ISO/IEC 18033-1 [32] standard states this property as one criteria for inclusion of cryptographic mechanisms. The concept behind this it that, if cryptographic mechanisms are standardized, they should be in the public domain for many years. In this way, security and performance analysis of them can be performed by third parties, which would give the users a significant amount of confidence in

security. The above mentioned eSTREAM project activity affected industry: one of the eSTREAM portfolio cipher, Trivium [14], is standardized in the lightweight stream cipher standard, ISO/IEC 29192-3 [31] together with Enocoro [48]. Grain-128a, which is based on the eSTREAM portfolio cipher Grain v1, is standardized in ISO/IEC 29167-13 [33] for the RFID application standard.

Despite of the above extensive academic and industry efforts, there is still an important gap to fill. There has been no authenticated encryption with associated data (AEAD) mechanism that meets very severe performance requirements in hardware and still offers 128-bit security, accompanied by serious evidence on cryptanalysis. In 2013, NIST initiated a lightweight cryptography project, followed by two workshops on the same subject. In 2017, NIST published a call for submissions for lightweight cryptographic mechanisms. One remarkable feature is that NIST requires each submission to implement the AEAD functionality. In [9], it was shown that lightweight stream ciphers are typically more suitable than lightweight block ciphers for energy optimization when encrypting longer messages, in particular when the speed can be increased at the expense of moderate extra hardware. Thus, a lightweight stream cipher seems to be a good starting point for a lightweight AEAD design.

This paper presents Grain-128AEAD, an authenticated encryption algorithm with support for associated data. The specification is in line with the requirements given by NIST and is based on the Grain stream cipher family. More specifically, it is closely based on Grain-128a, introduced in 2011, which has, already for several years, been analyzed in the literature. To benefit from the maturity of the Grain family, our strategy in the design of Grain-128AEAD is to have the changes made to Grain-128a as small as possible. Grain-128a is in turn based on Grain v1 and Grain-128, which have both been extensively analyzed, providing much insight into the security of the design approach. All Grain stream ciphers also allow the throughput to be increased by adding additional copies of the Boolean functions involved.

Industrial relevance of the Grain family can be explained as follows: Grain-128a receives a lot of attention from industry. ISO/IEC 29167-13:2015 specifying Grain-128a has been adopted in industrial applications. For instance, the passive IT70 RFID tag [30] that Honeywell has designed for automotive applications implements this security standard.

The outline of the paper is as follows. In Section 2 the specification of the new primitive is given. Then the overall design rationale, motivating the design choices, are given in Section 3. A security analysis, focusing on cryptanalysis of Grain-128a is then given in Section 4. The hardware implementation is described in Section 5 and the paper is concluded in Section 6.

2 Design Details

Grain-128AEAD consists of two main building blocks. The first is a pre-output generator, which is constructed using a Linear Feedback Shift Register (LFSR), a Non-linear Feedback Shift Register (NFSR) and a pre-output function, while

the second is an authenticator generator consisting of a shift register and an accumulator. The design is very similar to Grain-128a, but has been modified to allow for larger authenticators and to support AEAD. Moreover, the modes of usage have been updated.

2.1 Building Blocks and Functions

The pre-output generator generates a stream of pseudo-random bits, which are used for encryption and the authentication tag. It is depicted in Fig. 1. The

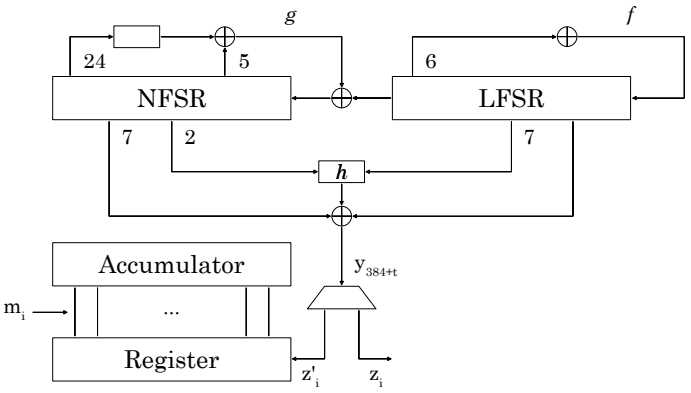

Fig. 1. An overview of the building blocks in Grain-128AEAD.

content of the 128-bit LFSR is denoted $S_t = [s_0^t, s_1^t, \ldots, s_{127}^t]$ and the content of the 128-bit NFSR is similarly denoted $B_t = [b_0^t, b_1^t, \ldots, b_{127}^t]$. These two shift registers represent the 256-bit state of the pre-output generator.

The primitive feedback polynomial of the LFSR, defined over GF(2) and denoted $f(x)$, is defined as

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

The corresponding update function of the LFSR is given by

$$\begin{aligned} s_{127}^{t+1} &= s_0^t + s_7^t + s_{38}^t + s_{70}^t + s_{81}^t + s_{96}^t \\ &= \mathcal{L}(S_t). \end{aligned}$$

The nonlinear feedback polynomial of the NFSR, denoted $g(x)$ and also defined over GF(2), is defined as

$$\begin{aligned} g(x) = 1 &+ x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} \\ &+ x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117} \\ &+ x^{46}x^{50}x^{58} + x^{103}x^{104}x^{106} + x^{33}x^{35}x^{36}x^{40} \end{aligned}$$

and the corresponding update function is given by

$$\begin{aligned} b_{127}^{t+1} &= s_0^t + b_0^t + b_{26}^t + b_{56}^t + b_{91}^t + b_{96}^t + b_3^t b_{67}^t + b_{11}^t b_{13}^t \\ &+ b_{17}^t b_{18}^t + b_{27}^t b_{59}^t + b_{40}^t b_{48}^t + b_{61}^t b_{65}^t + b_{68}^t b_{84}^t \\ &+ b_{22}^t b_{24}^t b_{25}^t + b_{70}^t b_{78}^t b_{82}^t + b_{88}^t b_{92}^t b_{93}^t b_{95}^t \\ &= s_0^t + \mathcal{F}(B_t). \end{aligned}$$

Nine state variables are taken as input to a Boolean function $h(x)$. Two of these bits are taken from the NFSR and seven are taken from the LFSR. The function is defined as

$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8,$$

where the variables $x_0, \ldots, x_8$ correspond to, respectively, the state variables $b_{12}^t, s_8^t, s_{13}^t, s_{20}^t, b_{95}^t, s_{42}^t, s_{60}^t, s_{79}^t$ and s_{94}^t.

The output of the pre-output generator, is then given by the pre-output function

$$y_t = h(x) + s_{93}^t + \sum_{j \in \mathcal{A}} b_j^t,$$

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

The authenticator generator consists of a shift register, holding the most recent 64 odd bits from the pre-output, and an accumulator. Both are of size 64 bits. We denote the content of the accumulator at instance i as $A_i = [a_0^i, a_1^i, \ldots, a_{63}^i]$. Similarly, the content of the shift register is denoted $R_i = [r_0^i, r_1^i, \ldots, r_{63}^i]$.

2.2 Key and IV Initialization

Before the pre-output can be used as keystream and for authentication, the internal state of the pre-output generator and the authenticator generator registers are initialized with a key and IV. Denote the key bits as k_i, $0 \leq i \leq 127$ and the IV bits as IV_i, $0 \leq i \leq 95$. Then the state is initialized as follows. The 128 NFSR bits are loaded with the bits of the key $b_i^0 = k_i$, $0 \leq i \leq 127$ and the first 96 LFSR elements are loaded with the IV bits, $s_i^0 = IV_i$, $0 \leq i \leq 95$. The last 32 bits of the LFSR are filled with 31 ones and a zero, $s_i^0 = 1, 96 \leq i \leq 126$, $s_{127}^0 = 0$. Then, the cipher is clocked 256 times, feeding back the pre-output function and XORing it with the input to both the LFSR and the NFSR, i.e.,

$$\begin{aligned} s_{127}^{t+1} &= \mathcal{L}(S_t) + y_t, \quad 0 \leq t \leq 255, \\ b_{127}^{t+1} &= s_0^t + \mathcal{F}(B_t) + y_t, \quad 0 \leq t \leq 255. \end{aligned}$$

Once the pre-output generator has been initialized, the authenticator generator is initialized by loading the register and the accumulator with the pre-output keystream as

$$\begin{aligned} a_j^0 &= y_{256+j}, \qquad 0 \leq j \leq 63, \\ r_j^0 &= y_{320+j}, \qquad 0 \leq j \leq 63. \end{aligned}$$

When the register and the accumulator are initialized, the key is simultaneously shifted into the LFSR,

$$s_{127}^{t+1} = \mathcal{L}(S_t) + k_{t-256}, \quad 256 \leq t \leq 383,$$

while the NFSR is updated as

$$b_{127}^{t+1} = s_0^t + \mathcal{F}(B_t), \quad 256 \leq t \leq 383.$$

Thus, when the cipher has been fully initialized the LFSR and the NFSR states are given by S_{384} and B_{384}, respectively, and the register and accumulator are given by R_0 and A_0, respectively. The initialization procedure is summarized in Fig 2.

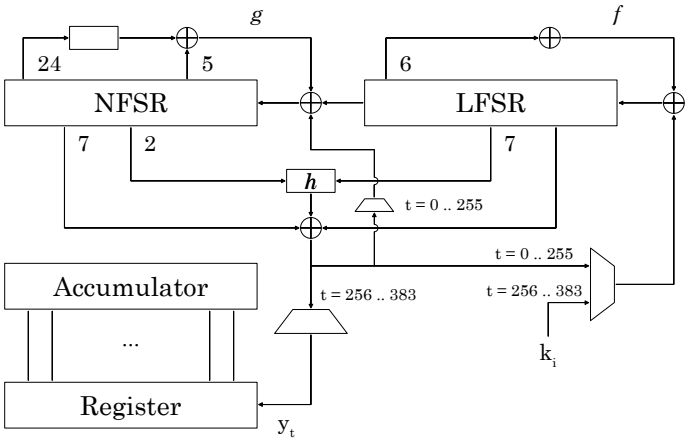

Fig. 2. An overview of the initialization in Grain-128AEAD. Note that, in hardware, the accumulator initialization is realized by first loading 64 pre-output bits into the register, followed by moving them to the accumulator.

2.3 Operating Mode

For a message $\boldsymbol{m}$ of length L, denoted $m_0, m_1, \ldots, m_{L-1}$, set $m_L = 1$ as padding in order to ensure that $\boldsymbol{m}$ and $\boldsymbol{m}\|0$ have different tags.

After initializing the pre-output generator, the pre-output is used to generate keystream bits z_i for encryption and authentication bits z_i' to update the register in the accumulator generator. The keystream is generated as

$$z_i = y_{384+2i},$$

i.e., every even bit (counting from 0) from the pre-output generator is taken as a keystream bit. The authentication bits are generated as

$$z_i' = y_{384+2i+1},$$

i.e., every odd bit from the pre-output generator is taken as an authentication bit. The message is encrypted as

$$c_i = m_i \oplus z_i, \quad 0 \leq i < L.$$

The accumulator is updated as

$$a_j^{i+1} = a_j^i + m_i r_j^i, \qquad 0 \leq j \leq 63, \quad 0 \leq i \leq L,$$

and the shift register is updated as

$$\begin{aligned} r_{63}^{i+1} &= z_i', \\ r_j^{i+1} &= r_{j+1}^i, \qquad 0 \leq j \leq 62. \end{aligned}$$

An AEAD scheme allows for data that is authenticated, but unencrypted. Grain-128AEAD achieves this simply by explicitly setting $y_{384+2i} = 0$ for bits that should not be encrypted, but should still be authenticated. This means that it is possible to control the associated data on bit level, and this data can appear anywhere in the message.

3 Design Rationale

This section presents a short overview of the Grain stream ciphers and how the design has evolved through the different versions. It also enumerates and discusses the differences between Grain-128a and the proposed Grain-128AEAD.

3.1 A Short History of the Grain Family of Stream Ciphers

The Grain family of stream ciphers are based on the idea behind the nonlinear filter generator. In a nonlinear filter, an LFSR is used to provide a sequence with large period, and a nonlinear function, taking parts of the LFSR sequence as input, is used to add nonlinearity to the keystream sequence. Much work has been put into analyzing the nonlinear filter generator and it is clear that it is very difficult to design a secure nonlinear filter generator with a reasonable hardware footprint [13]. In particular algebraic attacks have been shown to be very strong against this design, see e.g., [17, 41].

In order to better withstand algebraic attacks, and to make the relation between state/key and keystream more complex, Grain adds an NFSR to the nonlinear combiner. The initial submission to the ECRYPT eSTREAM project was analyzed in [10, 37], showing that the nonlinear functions required higher resiliency and nonlinearity. The modified design was subsequently published as Grain v1 [28] and was later selected for the final portfolio in eSTREAM. Grain v1 uses an 80-bit key, and a 128-bit key variant was proposed in [27]. Based on previous results on the Grain construction, Grain-128 was more aggressively designed, making the nonlinear NFSR feedback function of degree 2, but with high nonlinearity and resiliency. The relatively small functions compensated for

the fact that the shift registers were increased to 128 bits each, which increased the hardware footprint. The low degree functions were exploited in [3, 44] in order to cryptanalyze a significant number of initializations rounds. These results suggested that the nonlinear functions needed a higher security margin. Grain-128a was proposed in [50], and in addition to increasing the degree of the nonlinear feedback function, an optional authentication mode was added. Work on Grain-128 were subsequently improved [19–21, 35], emphasizing the need for more complex Boolean functions, and Grain-128 is considered broken and should not be used. The design proposed in this paper, Grain-128AEAD, is closely based on Grain-128a, using the same feedback and output functions. However, slight modifications have been made in order to add security and make it resistant to the attack proposed in [46].

3.2 Differences Between Grain-128AEAD and Grain-128a.

Grain-128AEAD takes Grain-128a as starting point, but introduces a number of slight modifications. The modifications are primarily motivated by the NIST Lightweight Cryptography Standardization Process, but inspiration also comes from recent results in [25, 46].

Larger MACs The register and the authenticator has been increased to 64 bits (instead of 32 bits) in order to allow for authentication tags (MACs) of size 64 bits.

No Encryption-only Mode Grain-128a allowed for an operation mode with only encryption, where the authentication was removed. This mode resulted in smaller hardware footprint since the two additional registers, and their associated logic, could be left out from an implementation. The encryption-only mode was also more efficient since the initialization process does not include initializing the register and the accumulator, and every pre-output bit was used as keystream. The proposed Grain-128AEAD is a pure authenticated encryption algorithm, and authentication of data is always supported. Thus, there is only one mode of operation.

Initialization Hardening Based on the ideas in [25] and used in Lizard [26], Grain-128AEAD re-introduces the key into the internal state during the initialization clock cycles. More specifically, it is serially shifted into the LFSR in parallell to the initialization of the register and the accumulator. Several variants can be considered here, including where and when to add the key. The LFSR is chosen due to the fact that if the LFSR is recovered (e.g., in a fast correlation attack as in [46]), it is comparably easy to recover the NFSR state. Moreover, since the LFSR output is XORed with the NFSR input, the key bits will continue to affect also the NFSR during pre-output generation. As for when, we choose to re-introduce it during the last 128 clocks of the initialization. This

provides maximum separation between its first introduction in the key loading part, where the key is loaded into the NFSR, and when it is re-introduced. Relations between keys are e.g., more difficult to exploit if the key is properly mixed into the state before the key is re-introduced.

By introducing the key as the last part of the initialization, we achieve the attractive effect that a state recovery attack does not immediately imply key recovery, as was the case for previous versions of Grain. While a state recovery would still render the cipher to be considered broken, the practical effect to deployed devices is highly limited. Recovering the state will only compromise the security of the current message, and not all messages using the same key.

Keystream Limitation Grain stream ciphers have been designed to allow for encrypting large chunks of data using the same key/IV pair. Previously, the Grain ciphers have not had any explicit limitation on the keystream length. However, to rule out attacks that use very large keystream sequences, Grain-128AEAD restricts the number of keystream bits for each key/IV to 2^{80}. We believe that this is well above what will be needed in the foreseeable future. Restricting the number of keystream bits will also make attacks that use linear approximations more difficult, e.g., [46].

4 Security Analysis

The security of the Grain family of stream ciphers has been investigated by a large number of third party analysts, publishing various analysis results on the different variants of Grain. Since its first introduction in 2005, much have been learned about the construction and the design approach. There have also been several published ciphers inspired by the design, e.g., Sprout [2] and its successor Plantlet [42]. Also Fruit [23] and Fruit-80 [1] are based on the same design idea. These ciphers have in common that they attempt to realize extremely resource constrained encryption. To minimize the hardware footprint, the key is assumed to be stored in non-volatile memory (NVM) on a device, and this memory is made part of the cryptographic algorithm. Since the key needs to be stored on a device anyway, using the key directly from NVM in the algorithm does not impose additional hardware to the construction. This is not the case for Grain, as we allow the key to be updated in the device, and the key storage is not a part of the cipher. Still, the fact that the above mentioned ciphers use the Grain design idea shows that the design seems to be very suitable for lightweight cryptography.

4.1 General Security Analysis

A main class of attacks on stream ciphers is the Time/Memory/Data tradeoff (TMD-TO) attack, an efficient method of finding either the key or the state of ciphers by balancing between time, memory and keystream data. This can sometimes be much more efficient and more practically applicable than a simple

exhaustive key search attack. Some stream ciphers are vulnerable to TMD-TO attacks and their effective key lengths could then be reduced. This typically happens if the state size is too small. A famous practical TMD-TO attack on A5/1 was given in [12].

A TMD-TO attack consists of two parts. The first is a preprocessing phase, during which a table is constructed. The mapping of different keys or internal states to some keystream segment is computed and stored in the table. It is sorted on keystream segments and this process is assumed to use time complexity P and memory M. In the second (real-time) phase, the attacker has intercepted D keystream segments and search for a collision with the table with time complexity T. A collision will recover the corresponding input. By a tradeoff between parameters P, D, M, and T, attackers can devise attacks according to available time, memory and data. Examples of tradeoffs are Babbage-Golic (BG) [4, 24] and Biryukov-Shamir (BS) [11] with curves $TM = N$, $P = M$ with $T \leq D$; and $MT^2D^2 = N^2$, $P = N/D$ with $T \geq D^2$, where N is the input space, respectively.

For Grain-128AEAD, attackers have no direct way to reconstruct the internal state, since the cipher has an internal state of size 256 bits (128-bit LFSR + 128-bit NFSR), i.e. $N = 2^{256}$. The best attack complexity achieved under BG tradeoff is with $T = M = D = N^{1/2} = 2^{128}$, which is not favourable compared to exhaustive key search. Also the BS tradeoff does not give complexity parameters of particular interest. Some improvements to TMD-TO attacks can be achieved through socalled BSW sampling [12] and the performance of such an approach is characterized by the sampling resistance of the stream cipher. Various generalizations of the concept of sampling resistance can be considered, e.g. [34], but it seems unlikely that this will lead to an attack with better performance than a standard Hellman-type time-memory tradeoff attack on the keyspace, a generic attack applicable to any cipher. Also, our limit on the length of keystreams affects such attacks.

Another class of general attacks are algebraic attacks, where the attacker derives a system of nonlinear equations in unknown key bits or unknown state bits and then solves the system. In general, solving a system of nonlinear equations is not known to be solvable in polynomial time, but there might be special cases that can be solved efficiently [16]. Due to the NFSR, the degree of the equations will gradually increase and it does not look promising to try to derive a system of nonlinear equations due to this property as well as the algebraic degree of the h function.

A general cryptanalytic technique is a guess-and-determine attack, where one guesses parts of the state and then from the keystream tries to determine other parts of the state. The goal is to guess as few positions as possible and determine as many as possible from equations involving the keystream. Again, since the dependence between a keystream symbol and the state includes many different positions in the state and some of them in nonlinear expressions, one has to guess a large portion of state variables in order to use an equation to determine a single state variable.

Being a binary additive stream cipher, Grain-128AEAD does not allow reuse of a key/IV pair since this will leak information about the corresponding plaintexts. Moreover, since Grain-128AEAD closely resembles Grain-128a, a key/IV pair used in one cipher may also not be reused in the other. Such cross-cipher key/IV reuse in a related cipher model is outside the security model of Grain-128AEAD.

In the subsequent subsections, we now describe the attacks that we consider as the main threat against lightweight stream ciphers in general and Grain-128AEAD in particular.

4.2 Correlation Attacks

Grain-128a was designed to resist conventional (fast) correlation attacks that exploit correlations between the state of the LFSR and the corresponding key stream. There has been devised a fast correlation attack on small state Grain-like stream ciphers in [49]. Due a much bigger state, this attack does not apply to Grain-128a. On the other hand, a recent paper [46] reveals that there are multiple linear approximations in Grain-128a that together with a viewpoint based on a finite field allow a fast correlation attack on the raw encryption mode of Grain-128a (and on the other members of the Grain family), where every keystream bit is assumed to be accessible by an opponent. This attack recovers the state of Grain-128a with data and time complexity of about 2^{114}. The data needs to come from the same secret key and the same IV.

It should be noted that this fast correlation attack does not apply to Grain-128a in authentication mode, as then only every second key stream bit may be accessible to an opponent. Thus, it does not apply to Grain-128AEAD.

4.3 Chosen IV Attacks

A variety of chosen IV attacks on Grain have been proposed, in both fixed key scenario as well as in the related key setting, and either for distinguishing purpose or for key recovery. In a fixed key scenario, chosen IV attacks have been devised on reduced-round versions using conditional differentials and using cube attacks, or combinations of both [22,38–40]. On Grain-128, a dynamic cube attack has been developed that succeeds in finding the secret key for the full 256-round initialization for a fraction of keys, [19]. Dynamic cube attacks have not been successful on Grain-128a thus far. Most of these results are experimental in nature, and do work only if the computational effort is practically feasible.

More recently, division property has been developed to improve cube attacks. Division property is an iterated technique for integral distinguishers introduced by Todo, in [45] and was applied initially to block ciphers. It turned out that it also applies to the initialization of stream ciphers, not only for distinguishers but also for key recovery. As opposed to conventional cube attacks, it can provide theoretical results. The latest result on Grain-128a in this direction is a key recovery on 184 initialization rounds, [47]. The data complexity is 2^{95}, and the computational complexity corresponds to about 2^{110} operations.

An attack that reaches the largest number of initialization rounds of Grain-128a in a fixed key scenario thus far is a conditional differential distinguishing attack and reaches 195 initialization rounds, but it works only for a fraction of all keys, [40].

The relevance of related key cryptanalysis of stream ciphers has been a subject of debate. A related key attack on Grain-128a in [18] recovers the secret key with a computational complexity 2^{96}, requiring 2^{96} chosen IVs and about 2^{104} keystream bits. It requires only 2 related keys. Another related key attack in [8] recovers the secret key using 2^{64} chosen IVs and 2^{32} related keys, where these figures need to be multiplied by some factor (about 2^8).

4.4 Fault Attacks

In the scenario of fault attacks on stream ciphers, the attacker is allowed to inject faults into the internal state, which means either flipping a binary value in memory or assigning a value to zero. By analyzing the difference in keystreams for the faulty and the fault-free case, one attempts to deduce the complete or some partial information about the internal state or the secret key. Fault attacks on stream ciphers have recently received some attention, starting with the work of Hoch and Shamir [29]. The most common methods of injecting faults is by using laser or through clock glitches. Fault attacks usually rely on assumptions that is beyond the model of cryptanalysis and for this reason one can often find rather efficient fault attacks on most ciphers. In some scenarios they are, however, not unrealistic and the exact complexity and the related requirements are of interest to study.

Fault attacks on the Grain family of stream ciphers were studied in [15] and [36]. More recently, there was a number of papers providing improved attacks, [5–7, 43]. In [43] the model is the most realistic one as it considers that the cipher has to be re-initialized only a few times and faults are injected to any random location and at any random clock cycle. No further assumptions are needed over location and timing for injections. In the attack one constructs algebraic equations based on the description of the cipher by introducing new variables so that the degrees of the equations do not increase. Following algebraic cryptanalysis, such equations based on both fault-free and faulty key-stream bits are collected. Then a solving phase using the SAT Solver recovers the state of any Grain member in minutes, For Grain v1, Grain-128 and Grain-128a, it uses only 10, 4 and 10 injected faults, respectively.

We stress that we are not claiming resistance against fault attacks for Grain-128AEAD. Rather, when fault attacks is a realistic threat, one has to implement protection mechanisms against fault injection.

5 Implementation

Lightweight ciphers are important in constrained devices. A minimal design is desirable, e.g., minimum area and very low power consumption since they often

must operate for an extended period of time, without a battery change. In some cases, devices run without its own power supply, something that is often the case with RFID tags.

Table 1. The gate count for different functions.

Function	**Gate Count**
NAND2	1.0
NAND3	1.5
NAND4	2.0
XOR2	2.5
XOR3	6.5
Flip flop	8.0

Grain-128AEAD can be constructed using primitive hardware building blocks, such as NAND gates, XOR gates and flip flops. In order to get an idea of the hardware footprint related to an implementation of the cipher, we implement the stream cipher using 65 nm library from ST Microelectronics, `stm065v536`. For synthesis and power simulation, the Synopsys Design Compiler 2013.12 is used. It can be noted that the result is highly dependent on what kind of gates are available and how the tool utilizes the standard cells. We define a 2-input NAND gate to have a gate count of 1 and other gate counts are given in relation to this NAND gate. An excerpt from the standard-cell library documentation is given in Table 1.

Table 2. Gate count for the different building blocks, for different levels of parallelization, s.

Building Block	**Gate Count**		
	$s = 1$	$s = 2$	$s = 32$
LFSR	1024	1024	1024
NFSR	1024	1024	1024
f	19	38	608
g	62.5	125	2000
h	41.5	83	1328
Control logic	219.5	475.5	942.5
Accumulator	512	512	512
Register	512	512	512
Accumulator logic	224	224	4160
Total	3638.5	4017.5	12110.5

We synthesize the design and extract the gate count for each building block. A summary of the gate count for each building block, and for different parallelization levels, is given in Table 2. The control logic and accumulator logic is extra circuitry and state machines for controlling the stream cipher, i.e., loading key and IV, multiplexing data, etc.

The gate count remains constant during synthesis, but the physical area, power and speed changes based on the optimization techniques employed. First, we synthesize the design at clock frequency 100 kHz. The design is synthesized for three levels of parallelization; 1, 2, and 32 times. The result is given in Table 3.

Table 3. Implementation results running at 100 kHz, for different levels of parallelization.

Parallelization	Area	Power	Throughput
1	4934 μm^2	313 nW	50 kbit/s
2	5336 μm^2	368 nW	100 kbit/s
32	16853 μm^2	574 nW	1600 kbit/s

We also synthesize for the maximum possible speed, to achieve maximum throughput, without constraints on area. The results are given in Table 4.

Table 4. Implementation results running at maximum possible speed, for different levels of parallelization.

Parallelization	Speed	Area	Power	Throughput
1	1.12 GHz	5258 μm^2	3.6 mW	560 Mbit/s
2	1.18 GHz	5629 μm^2	4.3 mW	1.18 Gbit/s
32	662 MHz	17632 μm^2	4.0 mW	10.59 Gbit/s

6 Conclusions

We have presented Grain-128AEAD, a new cipher in the Grain family. It is closely based on Grain-128a and takes advantage of the well-analyzed design principle behind the Grain stream ciphers. By making slight modifications to Grain-128a, the cipher meets the requirements in the NIST lightweight standardization process, providing 64-bit MAC, 128-bit key and 96-bit IV. The hardware footprint makes the cipher well suited for constrained environments, but the design is flexible enough to allow for also very high speed requirements at the expense of additional hardware.

References

1. Amin Ghafari, V., Hu, H.: Fruit-80: A secure ultra-lightweight stream cipher for constrained environments. Entropy **20**(3), 180 (2018)
2. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. In: Leander, G. (ed.) Fast Software Encryption. pp. 451–470. Springer Berlin Heidelberg (2015)
3. Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. SHARCS'09 Special-purpose Hardware for Attacking Cryptographic Systems p. 147 (2009)
4. Babbage, S.: Improved "exhaustive search" attacks on stream ciphers. IET Conference Proceedings pp. 161–166(5) (January 1995)
5. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on Grain-128a using MACs. In: Bogdanov, A., Sanadhya, S. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 111–125. Springer Berlin Heidelberg (2012)
6. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the Grain family of stream ciphers. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. pp. 122–139. Springer Berlin Heidelberg (2012)
7. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the Grain family under reasonable assumptions. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. pp. 191–208. Springer Berlin Heidelberg (2012)
8. Banik, S., Maitra, S., Sarkar, S., Meltem Sönmez, T.: A chosen IV related key attack on Grain-128a. In: Boyd, C., Simpson, L. (eds.) Information Security and Privacy. pp. 13–26. Springer Berlin Heidelberg (2013)
9. Banik, S., Mikhalev, V., Armknecht, F., Isobe, T., Meier, W., Bogdanov, A., Watanabe, Y., Regazzoni, F.: Towards low energy stream ciphers. IACR Transactions on Symmetric Cryptology **2018**(2), 1–19 (Jun 2018)
10. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of Grain. In: Robshaw, M. (ed.) Fast Software Encryption. pp. 15–29. Springer Berlin Heidelberg (2006)
11. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto, T. (ed.) Advances in Cryptology — ASIACRYPT 2000. pp. 1–13. Springer Berlin Heidelberg (2000)
12. Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) Fast Software Encryption. pp. 1–18. Springer Berlin Heidelberg (2001)
13. Braeken, A., Lano, J.: On the (im)possibility of practical and secure nonlinear filters and combiners. In: Preneel, B., Tavares, S. (eds.) Selected Areas in Cryptography. pp. 159–174. Springer Berlin Heidelberg (2006)
14. Cannière, C.D., Preneel, B.: Trivium. New Stream Cipher Designs - The eSTREAM Finalists pp. 244–266 (2008)
15. Castagnos, G., Gouget, A., Paillier, P., Berzati, A., Canovas, C., Goubin, L., Salgado, S., Debraize, B.: Fault analysis of GRAIN-128. In: Hardware-Oriented Security and Trust, IEEE International Workshop on(HST). pp. 7–14 (2009)
16. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) Advances in Cryptology — EUROCRYPT 2000. pp. 392–407 (2000)
17. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Annual International Cryptology Conference, CRYPTO 2003. pp. 176–194. Springer (2003)

18. Ding, L., Guan, J.: Related key chosen IV attack on Grain-128a stream cipher. IEEE Transactions on Information Forensics and Security **8**(5), 803–809 (2013)
19. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. pp. 327–343. Springer Berlin Heidelberg (2011)
20. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: Joux, A. (ed.) Fast Software Encryption. pp. 167–187. Springer Berlin Heidelberg (2011)
21. Fu, X., Wang, X., Chen, J., Stevens, M.: Determining the nonexistent terms of nonlinear multivariate polynomials: How to break Grain-128 more efficiently. IACR Cryptology ePrint Archive **2017**, 412 (2017)
22. Ghafari, V.A., Hu, H.: A new chosen IV statistical attack on Grain-128a cipher. In: Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference on. pp. 58–62. IEEE (2017)
23. Ghafari, V.A., Hu, H., Xie, C.: Fruit: ultra-lightweight stream cipher with shorter internal state. eSTREAM, ECRYPT Stream Cipher Project (2016)
24. Golić, J.D.: Cryptanalysis of alleged a5 stream cipher. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT '97. pp. 239–255. Springer Berlin Heidelberg (1997)
25. Hamann, M., Krause, M.: On stream ciphers with provable beyond-the-birthday-bound security against time-memory-data tradeoff attacks. Cryptography and Communications **10**(5), 959–1012 (2018)
26. Hamann, M., Krause, M., Meier, W.: Lizard–a lightweight stream cipher for power-constrained devices. IACR Transactions on Symmetric Cryptology **2017**(1), 45–79 (2017)
27. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: Information Theory, 2006 IEEE International Symposium on. pp. 1614–1618. IEEE (2006)
28. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. International Journal of Wireless and Mobile Computing **2**(1), 86–93 (2007)
29. Hoch, J.J., Shamir, A.: Fault analysis of stream ciphers. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 240–253. Springer Berlin Heidelberg (2004)
30. Honeywell: IT70 secure passive rfid tag. Technical Specifications (2017), `https://www.honeywellaidc.com/products/rfid/tags-labels/it70`
31. ISO/IEC 29192-3:2012 information technology – security techniques – lightweight cryptography – part 3: Stream ciphers (2012)
32. ISO/IEC 18033-1:2015 information technology – security techniques – encryption algorithms – part 1: General (2015)
33. ISO/IEC 29167-13:2015 information technology — automatic identification and data capture techniques — part 13: Crypto suite Grain-128A security services for air interface communications (2015)
34. Jiao, L., Zhang, B., Wang, M.: Two generic methods of analyzing stream ciphers. In: Lopez, J., Mitchell, C.J. (eds.) Information Security. pp. 379–396. Springer International Publishing (2015)
35. Karlsson, L., Hell, M., Stankovski, P.: Not so greedy: Enhanced subset exploration for nonrandomness detectors. In: Mori, P., Furnell, S., Camp, O. (eds.) Information Systems Security and Privacy. pp. 273–294. Springer International Publishing (2018)

36. Karmakar, S., Roy Chowdhury, D.: Fault analysis of Grain-128 by targeting NFSR. In: Nitaj, A., Pointcheval, D. (eds.) Progress in Cryptology – AFRICACRYPT 2011. pp. 298–315. Springer Berlin Heidelberg (2011)
37. Khazaei, S., Hasanzadeh, M.M., Kiaei, M.S.: Linear sequential circuit approximation of Grain and Trivium stream ciphers. IACR Cryptology ePrint Archive **2006**, 141 (2006)
38. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 130–145. Springer Berlin Heidelberg (2010)
39. Lehmann, M., Meier, W.: Conditional differential cryptanalysis of Grain-128a. In: Pieprzyk, J., Sadeghi, A.R., Manulis, M. (eds.) Cryptology and Network Security. pp. 1–11. Springer Berlin Heidelberg (2012)
40. Ma, Z., Tian, T., Qi, W.F.: Conditional differential attacks on Grain-128a stream cipher. IET Information Security **11**(3), 139–145 (2016)
41. Meier, W., Pasalic, E., Carlet, C.: Algebraic attacks and decomposition of boolean functions. In: International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2004. pp. 474–491. Springer (2004)
42. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Transactions on Symmetric Cryptology pp. 52–79 (2016)
43. Sarkar, S., Banik, S., Maitra, S.: Differential fault attack against Grain family with very few faults and minimal assumptions. IEEE Transactions on Computers **64**(6), 1647–1657 (2015)
44. Stankovski, P.: Greedy distinguishers and nonrandomness detectors. In: Gong, G., Gupta, K.C. (eds.) Progress in Cryptology - INDOCRYPT 2010. pp. 210–226. Springer Berlin Heidelberg (2010)
45. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 287–314. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
46. Todo, Y., Isobe, T., Meier, W., Aoki, K., Zhang, B.: Fast correlation attack revisited. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 129–159. Springer International Publishing (2018)
47. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 275–305. Springer International Publishing (2018)
48. Watanabe, D., Owada, T., Okamoto, K., Igarashi, Y., Kaneko, T.: Update on Enocoro stream cipher. In: 2010 International Symposium On Information Theory Its Applications. pp. 778–783 (Oct 2010)
49. Zhang, B., Gong, X., Meier, W.: Fast correlation attacks on Grain-like small state stream ciphers. IACR Trans. Symmetric Cryptol. **2017**(4), 58–81 (2017)
50. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. International Journal of Wireless and Mobile Computing **5**(1), 48–59 (2011)

A Test Vectors

Here, we give some test vectors for different keys, IVs, and messages. The test vectors are given in hexadecimal, e.g., the key

`0x01234FFFFFFFFFFFFFFFFFFFFFFFFFFF`

corresponds to

$$(k_0, ..., k_{127}) = (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, ..., 1).$$

The message stream is given with the padding included. A padding bit of 1 equals a padding byte of 0x80. Note that for an empty message, the message stream is just the padding.

```
Key:       0x00000000000000000000000000000000
IV:        0x000000000000000000000000
Keystream: 0xc800a52f948b89b85cee6cfd8571f90f
Message:   0x80
Tag:       0xaab555c073e67664

Key:       0x0123456789abcdef123456789abcdef0
IV:        0x0123456789abcdef12345678
Keystream: 0xc2b918c6baf6dea0865200d46858a37b
Message:   0xFF80
Tag:       0x782f4c4a8907ba7f
```