Key recovery attacks on Grain family using BSW sampling and certain weaknesses of the filtering function

Y. Wei * E. Pasalic † F. Zhang ‡ W. Wu §

Abstract

A novel internal state recovery attack on the whole Grain family of ciphers is proposed in this work. It basically uses the ideas of BSW sampling along with employing a weak placement of the tap positions of the driving LFSRs. The *currently best known* complexity trade-offs are obtained, and due to the structure of Grain family these attacks are also key recovery attacks. It is shown that the internal state of Grain-v1 can be recovered with the time complexity of about 2^{66} operations using a memory of about $2^{58.91}$ bits, assuming availability of 2^{45} keystream sequences each of length 2^{49} bits generated for different initial values. Moreover, for Grain-128 or Grain-128a, the attack requires about 2^{105} operations using a memory of about $2^{82.59}$ bits, assuming availability of 2^{75} keystream sequences each of length 2^{76} bits generated for different initial values. These results further show that the whole Grain family, due to the choice of tap positions mainly, does not provide enough security margins against internal state recovery attacks. A simple modification of the selection of the tap positions, as a countermeasure against the attacks described here, is given.

Keywords : Stream cipher, Grain cipher, State recovery attack, BSW sampling.

1 Introduction

The Grain family of stream ciphers was initially developed through the Grain-v0 version, which was submitted in the initial phase of the eSTREAM project in 2005 [18]. This design is well suited for hardware restricted environments, and in 2008 Grain-v1 (a tweaked version of Grain-v0) entered the last evaluation phase being one of the three remaining candidates in the final hardware portfolio of the eSTREAM project [26]. Grain-v1 employs an 80-bit secret key and a 64-bit initial value, whereas the number of internal state bits is 160 [19]. To satisfy higher security requirements, a version of Grain-v1, known as Grain-128, was proposed by the designers in 2006 [20]. This version uses a 128-bit secret key and a 96-bit initial value. Later, to meet both functionalities of message authentication

*Guilin University of Electronic Technology, Guilin, P.R. China, e-mail: walker_wei@msn.com.

†University of Primorska, FAMNIT and IAM, Koper, Slovenia, e-mail: enes.pasalic6@gmail.com

‡School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu 221116, P.R. China, e-mail:zhfl203@cumt.edu.cn.

§Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.R. China, e-mail: wwl@is.iscas.ac.cn

and encryption, the designers presented a modified version Grain-128a at SKEW2011 [2], whose key length and the length of the initialization vector is the same as for Grain-128. The number of internal state bits for both Grain-128 and Grain-128a is 256.

The Grain family of stream ciphers has a compact and high-level structure, especially suitable for hardware constrained applications. The design rationale of the Grain family is based on the use of two shift registers, that is, one linear feedback shift register (LFSR) and one nonlinear feedback shift register (NFSR). These registers provide the inputs to a nonlinear filtering function h whose output is then added to one particular (secret) state bit of the NSFR to provide the output keystream bit [19, 20, 2]. Due to the simplicity of its design, the Grain family of stream ciphers has been a favourite target for the application of diverse cryptanalytic methods. In general, the main cryptanalytic methods applied to the Grain family of encryption algorithms may be divided into several major groups, namely into distinguishing attacks [27, 25], algebraic attacks [1], various types of chosen IV and (dynamic) cube attacks [12, 13, 24], related key chosen IV attacks [11, 10, 14], near collision and differential fault attacks [28, 9], and internal state recovery attacks [5, 22, 23].

Our technique described here belongs to state (key) recovery attacks that are closely related to the problem of inverting a one way function $y = f(x)$. Notice that the idea of Time-Memory trade-off attack (TMTO) was firstly proposed by Hellman in [17], which combines the exhaustive key attack and the table lookup attack to solve the problem of inverting one way functions. More precisely, the trade-off curve of TMTO in [17] is described as $TM^2 = N^2$ and $P = N$, where T is the online time complexity, M is the memory cost, P is the offline time complexity, and N is the number of possible keys. Later, some new time-memory-data trade-off (TMDTO) attacks on stream cipher are respectively presented by Babbage [4] and Golić [16]. The tradeoff curve of Babbage-Golic attack (i.e., BG-TMDTO attack) can be described as $TM = N, P = M$ and $T = D$, where D denotes the data complexity and N is the number of possible internal states. At Asiacrypt 2000, using multiple data points, Biryukov and Shamir proposed an extension of TMDTO attacks [6]. The trade-off curve of Biryukov-Shamir attack (BS-TMDTO) in [6] can be described as $TM^2D^2 = N^2$ and $P = N/D$, where $1 \leq D^2 \leq T$.

At FSE 2000, a new sampling technique was proposed by Biryukov, Shamir, and Wagner (denoted by BSW sampling) [7], which can improve upon the BS-TMDTO attack by providing a more flexible choice of parameters. The main idea of BSW sampling is to find an efficient way to generate and enumerate "special" cipher states, from which the first output segment of keystream bits of the cipher is a fixed string (such as a run of consecutive zeros). If this can be done for a run of u bits, the sampling resistance of the cipher is defined to be $R = 2^{-u}$. Being of no particular importance for small values of u, this approach may lead to improved trade-off attacks if u is sufficiently large. In [5], some special states of Grain-v1 were identified giving the sampling resistance $R = 2^{-21}$. This resistance allowed the authors to mount an attack on Grain-v1 with online time complexity of $O(2^{70})$ and memory complexity of $O(2^{69})$ after a precomputation of $O(2^{104})$ steps and using $O(2^{56})$ bits of a known keystream sequence.

Recently, at Africacrypt 2014, Ding *et al.* [15] proposed a generalization of BG-TMDTO attack, which combines the original BG-TMDTO attack and the BSW sampling technique given that multiple data is available for the attacker generated from the same

Sample (d, d')	Time (off-line)	Memory	Time (online)	Success rate	Resource
$(1, 2^{56})$	2^{104}	2^{69}	2^{70}	1	[5]
$(2^{46}, 2^{50})$	2^{70}	2^{46}	2^{72}	1	[22]
$(1, 2^{67.8})$	$2^{73.1}$	$2^{62.8}$	$2^{71.4}$	unresolved	[28]
$(2^{45.25}, 2^{45.25})$	$2^{69.5}$	$2^{69.5}$	$2^{69.5}$	1	[15]
$(2^{45}, 2^{49})$	2^{66}	$2^{58.91}$	2^{66}	1	*new*

Table 1: Summary on the main attack on Grain-v1 - a single-key framework

secret key and different IV values. The trade-off curve of this attack can be described as $MT = rRN, MD = N, P = M$ and $D = d \cdot d'$, where r is an integer ($1 \leq r \leq R^{-1}$), d denotes the number of keystream sequences the attacker can capture (generated by the same key but different IVs), and d' stands for the length of each sequence. It was shown that this generalization of BG-TMDTO attack on Grain-v1 and Grain-128 stream ciphers is rather efficient since the online time, offline time and memory complexity of the attack are all lower than an exhaustive key search. Even though the data complexity $D = d \cdot d' > 2^K$ [15] (see also Table 1 and Table 2), the attack scenario is more realistic compared to the case when the same amount of output data D is generated by a fixed secret key and a single IV. Prior to this work, two major improvements regarding the cryptanalysis of Grain-v1 and Grain-128 were proposed recently in [22, 23], where by employing the normality of filtering function h the best known trade-off parameters could be obtained (cf. Table 1 and Table 2). Notice that this approach also requires $D > 2^K$, though $d, d' \ll 2^K$.

Nevertheless, it appears that the Grain family leaves a lot of space for improvements to prevent the existence of these special internal states of relatively moderate size. In particular, due to the choice of tap positions of the Grain variants and the sparseness of the filtering function, we have been able to reduce the sampling resistance to $R = 2^{-28}$ for Grain-v1 and to $R = 2^{-46}$ for Grain-128(a). For Grain-v1 (and similarly for Grain-128), this is done by manipulating the output keystream bit expression $z_t = \sum_{j \in A} b_{t+j} \oplus h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63})$, where $A = \{1, 2, 4, 10, 31, 43, 56\}$, with respect to the 7 masking NFSR bits b_{t+j}. Due to the designers selection of the tap positions the masking NFSR bits do not influence the function h during a certain time interval. This further implies that these bits can be computed by guessing the remaining state bits involved in the computation, and in addition the sparseness of h also allows us to retrieve a few more bits by fixing suitable state bits. Compared to the currently best known attack presented in [22][15], our approach gives better complexity estimates.

Table 1 and Table 2 summarise relevant previous work and our main attack results on the Grain variants. Apparently, our attack has smaller time complexity compared to other approaches, and in general it performs better when all the complexity measures are taken into account.

The rest of the paper is organized as follows. In Section 2, a brief description of the structure of Grain family is given. In Section 3, based on a few simple observations, a state (key) recovery attack is proposed against all Grain variants. Finally, Section 4 concludes

Sample (d, d')	Time (off-line)	Memory	Time (online)	Success rate	Resource
$(1, 2^{59})$	2^{113}	2^{59}	2^{113}	0.001	[12]
$(1, 2^{63})$	2^{90}	2^{63}	2^{90}	0.05	[13]
$(2^{50}, 2^{95})$	2^{111}	2^{111}	2^{116}	1	[23]
$(2^{69.5}, 2^{69.5})$	2^{117}	2^{117}	2^{117}	1	[15]
$(2^{75}, 2^{76})$	2^{105}	$2^{82.59}$	2^{105}	1	*new*

Table 2: Summary on the main attacks on Grain-128(a)- a single-key framework

the paper with a brief discussion concerning the possibility of thwarting similar kind of attacks as the one presented here.

2 Preliminaries

2.1 A brief description of the Grain family of stream ciphers

For the self-completeness of the discussion that follows, we give a brief description concerning the structure of the Grain family of stream ciphers. Due to a varying length of the secret key and the initialization vector (as well as the length of the registers) used in different versions, a full specification of the filtering functions used is given in Section 3.1 and Section 3.2.

It is convenient to denote by n the length of the shift registers, alternatively the length of the secret key. The structure of the Grain cipher is depicted in Figure 1. The LFSR is updated through a linear feedback polynomial f of degree n and is given by,

$$s_{t+n} = f(\mathbf{s}_t), \tag{1}$$

where $\mathbf{s}_t = (s_t, s_{t+1}, \ldots, s_{t+n-1})$ is the secret state of the LFSR at time t. Similarly, the NFSR is updated through a nonlinear feedback polynomial g of degree n,

$$b_{t+n} = s_t \oplus g(\mathbf{b}_t), \tag{2}$$

where $\mathbf{b}_t = (b_t, b_{t+1}, \ldots, b_{t+n-1})$ is a secret state of the NFSR at time t. The keystream sequence is generated by combining the secret state bits of both registers in a nonlinear manner,

$$z_t = h(\mathbf{s}_t, \mathbf{b}_t) \oplus \sum_{a \in A} b_{t+a}, \tag{3}$$

where $A \subset \{0, 1, \ldots, n-1\}$, and A is fixed for each Grain variant.

The key loading algorithm (KLA) loads an n-bit key into the NFSR and an m-bit IV vector into the LFSR (at first m stages). Since $m < n$, the remaining $n - m$ bits of the LFSR are loaded with some fixed padding scheme P which depends on the version considered. However, the $2n$ bit initial secret state is of the form $K||IV||P$, where the first n bits correspond to the content of the NFSR.

The key scheduling algorithm (KSA) is run for the first $2n$ clocks in a non-output mode, which is formally accomplished by feeding back the output bits z_t to the registers. Thus, the update functions during the KSA are:

$$s_{t+n} = f(\mathbf{s}_t) \oplus z_t; \;\; b_{t+n} = s_t \oplus g(\mathbf{b}_t) \oplus z_t. \tag{4}$$

After the completion of the KSA, z_t is no longer used as a part of the update function (which is now given by (1) and (2)), it is rather considered as the output keystream bit.

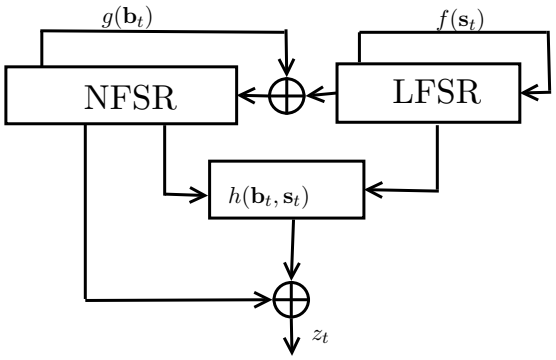

Figure 1: Block description of Grain cipher

2.2 BSW Sampling and a generalization of the BG-TMDTO attack

The concept of BSW-sampling was firstly introduced by Biryukov, Shamir, and Wagner in [7], which can be used to obtain wider choices of trade-off parameters for the BS-TMDTO attack. In general, the BSW sampling trade-off works if the following assumption is satisfied for a given cipher.

Assumption 1 *For a given stream cipher with $n = \log_2 N$ bits of internal state, given a value of $n-l$ particular (special) state bits of the cipher and the first l bits of the keystream sequence generated from the (partially unknown) internal state, then the remaining l bits of the internal state can be recovered directly (and efficiently).*

Assume that the sampling resistance of a given stream cipher is $R = 2^{-l}$. The adversary can then use a family of one-way functions $f' : \{0,1\}^{n-l} \to \{0,1\}^{n-l}$ as follows [7]:

Step 1 *Fix a specific function by choosing an l-bit string S as the first l bits of the keystream sequence, e.g. take 0^l.*[1]

Step 2 *Given S and an $(n-l)$-bit input value x as a part of the secret state, recover the remaining l bits of the secret state using Assumption 1.*

Step 3 *Clock the cipher n times, thus generating an n-bit output string $S|y$.*

Step 4 *Output y.*

[1]Without loss of generality we consider the all-zero sequence of length l though any other string may be used as well.

Actually, these functions map $(n-l)$-bit of states to $(n-l)$-bit keystream segments, restricted to a subset of so-called special states. Finally, given D bits of the actual keystream, the expected number of special states encountered is DR. Therefore, the adversary can only consider the cost of inverting $f' : \{0,1\}^{n-l} \to \{0,1\}^{n-l}$ rather than inverting the full state of the cipher.

At Africacrypt 2014 [15], Ding *et al.* combined the BG-TMDTO attack with the BSW sampling technique to mount a new TMDTO attack, which is called here a generalization of the BG-TMDTO attack. The main idea of this generalization is the use of multiple data points, thus the keystream segments available for cryptanalysis are generated by the same key and different IVs. More precisely, for a given stream cipher, assume the adversary can collect a set of d keystream sequences generated by the same secret key but different IVs, and the length of each sequence is d' bits. Hence, the set of samples available for the attack consists of approximately $D = d \times d'$ keystream segment samples each of length $2n$ bits.

Below, for self-completeness, we briefly describe the generalization of BG-TMDTO attack, its offline and online phase.

The Offline phase[15]

Randomly collect r strings $S_1, \cdots, S_r$, where each string consists of l bits and $1 \leq r \leq R^{-1}$. For each fixed string S_i, perform the following steps.

Step 1 *Randomly collect θ strings $I_1, \cdots, I_\theta$, each I_j being of length $n-l$ bits.*

Step 2 *By treating I_j as $n-l$ particular state bits and S_i as the first l bits of the keystream, compute the remaining l bits of the secret state, and clock the stream cipher n times to generate an n-bit keystream segment. Moreover, put this pair (n-bit internal state, n-bit keystream segment) in the table T_i corresponding to S_i.*

The Online phase[15]

For each $2n$-bit keystream segment sample, check the first l bits of the sample and one of r strings $S_1, \cdots, S_r$. If the match is not found, then go to check the next $2n$-bit keystream segment sample. If the match is obtained, then perform the following steps.

Step 1 *For each match (say S_i), check if the first n-bit keystream segment exists in the second column of the corresponding table (i.e., T_i). If it does not exist, consider the next $2n$-bit keystream segment sample. If there is an agreement of these n bits, record the corresponding n-bit internal state in the first column of the table, clock the cipher $2n$ times to generate $2n$ keystream bits, and then compare the output with the sample. If these are identical, go to the output (a). Otherwise, consider the next $2n$-bit keystream segment sample.*

Step 2 *If none of the keystream segment samples give a match, go to the output (b).*

Output: *(a) recorded n-bit internal state; or (b) a flag that returns the algorithm is failed.*

The trade-off curve of this generalization of BG-TMDTO attack can be described as $MT = rRN, MD = N, P = M$ and $D = d \cdot d'$, where r is a integer $(1 \leq r \leq R^{-1})$, T is the online time complexity, M is the memory cost, P is the offline time complexity, and N is the number of possible keys [15]. Notice that there exists an important assumption for this attack, that is, the availability of d keystream sequences each of length d' bits generated using different initial values, where $d \ll 2^m$, $d' \ll 2^N$, and m stands for the

size of initial value vector. In this case, the keystream segment samples $D = d \times d'$ even can be larger than 2^N. For instance, in [15], it is shown the internal state of Grain-v1 can be recovered with the time complexity of about $2^{69.5}$ operations using a memory of about $2^{69.5}$ bits, assuming availability of $2^{45.25}$ keystream sequences each of length $2^{45.25}$ bits generated for different initial values, where $D = d \times d' = 2^{45.25} \times 2^{45.25} = 2^{90.5} > 2^{80}$.

3 Internal state recovery attacks on Grain family

In this section, we describe an application of BSW sampling on Grain family used for mounting state recovery attacks on the variants of the Grain cipher. It was originally noticed [5] that Grain-v1 allows the attacker to recursively recover the secret key once the whole secret state at some instance t has been revealed. The importance of this fact lies in the fact that even though we are formally referring to internal state recovery attacks it should be understood that we at the same time refer to key recovery attack as well. A similar observation also applies to Grain-128 and Grain-128a, see [8] and [10].

3.1 Attacking Grain-v1

The internal secret state of Grain-v1 has 160 bits, and it consists of an 80-bit LFSR, (denoted as $(s_0, \cdots, s_{79})$) and an 80-bit NFSR, (denoted as $(b_0, \cdots, b_{79})$). The update functions of the LFSR and NFSR are, respectively, described as follows:

$$\begin{aligned}
s_{t+80} &= s_{t+62} \oplus s_{t+51} \oplus s_{t+38} \oplus s_{t+23} \oplus s_{t+13} \oplus s_t, \\
b_{t+80} &= s_t \oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \oplus b_{t+37} \oplus b_{t+33} \oplus b_{t+28} \oplus b_{t+21} \oplus b_{t+14} \\
&\quad \oplus b_{t+9} \oplus b_t \oplus b_{t+63}b_{t+60} \oplus b_{t+37}b_{t+33} \oplus b_{t+15}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+45} \\
&\quad \oplus b_{t+33}b_{t+28}b_{t+21} \oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \\
&\quad \oplus b_{t+63}b_{t+60}b_{t+21}b_{t+15} \oplus b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \\
&\quad \oplus b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \oplus b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}. \qquad (5)
\end{aligned}$$

The output function is defined as

$$z_t = h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}) \oplus \sum_{j \in A} b_{t+j}, \qquad (6)$$

where $A = \{1, 2, 4, 10, 31, 43, 56\}$. The output bit z_t is filtered by a nonlinear function $h(x)$ given by,

$$h(x) = x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4, \quad (7)$$

where the variables x_0, x_1, x_2, x_3, x_4 of $h(x)$ correspond to the tap positions s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}, respectively.

Grain-v1 is initialized with a 64-bit IV injected directly into the LFSR (the remaining bits of the LFSR are assigned value one), and an 80-bit key that is loaded into the NFSR. Then the cipher is clocked 160 times without producing any keystream, but feeding the output bits back into both the LFSR and the NFSR.

Notice that the NFSR bit b_{t+63} will be updated through the nonlinear feedback (5), and this bit affects the computation of z_t. To obtain more equations that indicate the relationship between the state bits and z_t, our strategy is to eliminate the dependence on b_{t+63} in z_t. From (6), it is clear that the dependency of z_t on b_{t+63} is equivalent to the dependency of $h(x)$ on the variable x_4. More precisely, by setting certain constraints on the secret bits of the LFSR we deduce two linear operation modes (OM) [2] of the cipher:

1. (OM 1) Let $x_2 = 0, x_3 = 1$ (i.e., $s_{t+46} = 0, s_{t+64} = 1$), then $h(x) = x_0 \oplus x_1$,
2. (OM 2) Let $x_0 = 1, x_1 = 0, x_2 = 1$ (i.e., $s_{t+3} = 1, s_{t+25} = 0, s_{t+46} = 1$), then $h(x) = x_3$.

Then using (6), the value of b_{t+10} can be computed directly from the values of z_t and the 11 other state variables occurring in the output equation. Combining this observation in the framework of OM 1, we can compute b_{t+10}, for $t = 0, \ldots, 15$, as follows:

$$\begin{aligned} b_{10} &= z_0 \oplus b_1 \oplus b_2 \oplus b_4 \oplus b_{31} \oplus b_{43} \oplus b_{56} \oplus s_3 \oplus s_{25}, \text{ where } s_{46} = 0, s_{64} = 1, \\ &\vdots \\ b_{25} &= z_{15} \oplus b_{16} \oplus b_{17} \oplus b_{19} \oplus b_{46} \oplus b_{58} \oplus b_{71} \oplus s_{18} \oplus s_{40}, \text{ where } s_{61} = 0, s_{79} = 1. \end{aligned} \tag{8}$$

Consequently, 32 constraint secret bits are assigned for which $h(x) = x_0 \oplus x_1$, that is, we need to set $s_{46} = s_{47} = \cdots = s_{61} = 0$ and $s_{64} = s_{65} = \cdots = s_{79} = 1$ using OM 1. The remaining assignment of the special state bits depends on the tap positions of the driving LFSR.

The distance distribution of the LFSR's tap positions $s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, s_{t+79}$ yields the following differences $\{25 - 3 = 22, 46 - 25 = 21, 64 - 46 = 18, 79 - 64 = 15\}$. Notice that the latest updated bit of the LFSR s_{t+79} is also taken into account since s_{t+79} will be shifted to the position of s_{t+64} after exactly $(79 - 64) + 1 = 15 + 1 = 16$ clocks (steps). After these 16 clocks, we derive further constraints using OM 2. We notice that the tap position difference $64 - 46 = 18$ implies that during the next 2 clocks the bits s_{62} and s_{63} will arrive at position s_{46} and therefore these bits must be set to one in order to satisfy OM 2. Furthermore, since in this mode we also have the condition that $s_{t+3} = 1, s_{t+25} = 0$, which for $t = 16, 17$, implies assigning 4 more bits to the above values. So far, 32 + 6 =38 state bits have been specified. Similarly, taking into account the tap position differences $25 - 3 = 22, 46 - 25 = 21$, we have been able to specify 13 more state bits that satisfy the requirements for the two operation modes. Thus, in total 51 bits have been specified. A full specification of these special state bits of the LFSR is summarised in Table 3. Here, " $*$ " means an unspecified bit and there are 29 unspecified bits in total.

The next step is, based on the already assigned state bits (51 in total), to recover 25 secret bits of the NFSR and the 3 secret bits of the LFSR (namely s_0, s_1 and s_2) by additionally assigning 81 bits of the secret state. The process of recovering secret state bits of the NFSR, by further specifying suitable state bits, is pretty straightforward. For instance, to compute b_{10} using (8) it is enough to assign seven bits $b_1, b_2, b_4, b_{31}, b_{43}, b_{56}, s_3$

[2] Note that (OM 1) has the same linear relation as the property 2 in [21], but (OM 2) has completely different linear relation compared to the property 2 in [21]

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
s_{16}	s_{17}	s_{18}	s_{19}	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}	s_{26}	s_{27}	s_{28}	s_{29}	s_{30}	s_{31}
$*$	$*$	$*$	1	1	1	1	1	1	1	1	1	1	1	1	$*$
s_{32}	s_{33}	s_{34}	s_{35}	s_{36}	s_{37}	s_{38}	s_{39}	s_{40}	s_{41}	s_{42}	s_{43}	s_{44}	s_{45}	s_{46}	s_{47}
$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	0	0	0	0	0	0	0
s_{48}	s_{49}	s_{50}	s_{51}	s_{52}	s_{53}	s_{54}	s_{55}	s_{56}	s_{57}	s_{58}	s_{59}	s_{60}	s_{61}	s_{62}	s_{63}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
s_{64}	s_{65}	s_{66}	s_{67}	s_{68}	s_{69}	s_{70}	s_{71}	s_{72}	s_{73}	s_{74}	s_{75}	s_{76}	s_{77}	s_{78}	s_{79}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3: Special state bits of the LFSR

for a given output z_0. The whole process of computing 28 secret state bits by additionally specifying 81 bits of the secret state is depicted in Table 4. For convenience of the reader, we analyse in detail a more involved procedure of recovering the secret bit b_{29} to b_0 in Appendix A, which corresponds to the step 16 to the step 27 in Table 4.

The above discussion, along with Table 3 and Table 4, implies the following result.

Lemma 1 *Given the value of 132 particular (special) state bits of Grain-v1 and the first 28 keystream output bits generated from the internal state, then the remaining 28 bits of the internal state can be recovered directly.*

3.1.1 The complexity estimates

The attack has two phases, i.e., the preprocessing phase (off-line) and the internal state recovery phase (on-line). A trade-off parameter l used in both phases gives a flexibility of adjusting different complexities of the attack, that is, to choose a suitable point on the TMDTO (time-memory-data-trade-off) curve. The *preprocessing phase* contains the steps depicted below, under the assumption that the cipher generates a keystream sequence with the 28-zero prefix. Notice that during this phase, apart from having specified 51 special state bits and recovering 28 bits of the secret state by using Lemma 1, in addition l bits of the secret state are guessed.

Input: The trade-off parameter l, $2 \leq l \leq 15$.
Step 1: Set $s_{19} = 1, \cdots, s_{30} = 1$; $s_{41} = 0, \cdots, s_{61} = 0; s_{62} = 1, \cdots, s_{79} = 1$ (51 constraint bits in total, see Table 3), and $s_3 = 0, \cdots, s_{l+2} = 0$.

Step 2: Based on the above constraints, for each of $2^{160-51-28-l} = 2^{81-l}$ possible patterns of the internal state of Grain-v1 perform the steps below:

(1) For each given pattern, deduce the bits $b_0, b_{10}, \cdots, b_{30}, b_{77}, \cdots, b_{79}, s_0, s_1, s_3$ (28 bits in total) by using Table 3 and Table 4, cf. Lemma 1.
(2) Perform a backward computation recursively to recover the initial state consisting of an 80-bit secret key, a 64-bit initial value, and a 16-bit padding

Step	Recovered bit	Fixed NFSR and LFSR bits	Constraint conditions
0	b_{10}	$b_1, b_2, b_4, b_{31}, b_{43}, b_{56}, s_3$	$z_0 = 0, s_{46} = 0, s_{64} = 1$
1	b_{11}	$b_3, b_5, b_{32}, b_{44}, b_{57}, s_4$	$z_1 = 0, s_{47} = 0, s_{65} = 1$
2	b_{12}	$b_6, b_{33}, b_{45}, b_{58}, s_5$	$z_2 = 0, s_{48} = 0, s_{66} = 1$
3	b_{13}	$b_7, b_{34}, b_{46}, b_{59}, s_6$	$z_3 = 0, s_{49} = 0, s_{67} = 1$
4	b_{14}	$b_8, b_{35}, b_{47}, b_{60}, s_7$	$z_4 = 0, s_{50} = 0, s_{68} = 1$
5	b_{15}	$b_9, b_{36}, b_{48}, b_{61}, s_8$	$z_5 = 0, s_{51} = 0, s_{69} = 1$
6	b_{16}	$b_{37}, b_{49}, b_{62}, s_9, s_{31}$	$z_6 = 0, s_{52} = 0, s_{70} = 1$
7	b_{17}	$b_{38}, b_{50}, b_{63}, s_{10}, s_{32}$	$z_7 = 0, s_{53} = 0, s_{71} = 1$
8	b_{18}	$b_{39}, b_{51}, b_{64}, s_{11}, s_{33}$	$z_8 = 0, s_{54} = 0, s_{72} = 1$
9	b_{19}	$b_{40}, b_{52}, b_{65}, s_{12}, s_{34}$	$z_9 = 0, s_{55} = 0, s_{73} = 1$
10	b_{20}	$b_{41}, b_{53}, b_{66}, s_{13}, s_{35}$	$z_{10} = 0, s_{56} = 0, s_{74} = 1$
11	b_{21}	$b_{42}, b_{54}, b_{67}, s_{14}, s_{36}$	$z_{11} = 0, s_{57} = 0, s_{75} = 1$
12	b_{22}	$b_{55}, b_{68}, s_{15}, s_{37}$	$z_{12} = 0, s_{58} = 0, s_{76} = 1$
13	b_{23}	b_{69}, s_{16}, s_{38}	$z_{13} = 0, s_{59} = 0, s_{77} = 1$
14	b_{24}	b_{70}, s_{17}, s_{39}	$z_{14} = 0, s_{60} = 0, s_{78} = 1$
15	b_{25}	b_{71}, s_{18}, s_{40}	$z_{15} = 0, s_{61} = 0, s_{79} = 1$
16	b_{29}	b_{75}	$z_{19} = 0, s_{22} = 1, s_{44} = 0, s_{65} = 1$
17	b_{30}	b_{76}	$z_{20} = 0, s_{23} = 1, s_{45} = 0, s_{66} = 1$
18	b_{77}	$-$	$z_{21} = 0, s_{24} = 1, s_{46} = 0, s_{67} = 1$
19	s_0	b_{72}, b_{73}	$z_{16} = z_{17} = z_{25} = 0, s_{19} = s_{20} = s_{28} = 1$ $s_{41} = s_{42} = s_{50} = 0, s_{62} = s_{63} = s_{71} = 1$
20	s_1	b_{74}	$z_{17} = z_{18} = z_{26} = 0, s_{20} = s_{21} = s_{29} = 1$ $s_{42} = s_{43} = s_{51} = 0, s_{63} = s_{64} = s_{72} = 1$
21	b_{28}	$-$	$z_{27} = 0, s_{30} = 1, s_{52} = 0, s_{73} = 1$
22	s_2	$-$	$z_{18} = 0, s_{21} = 1, s_{43} = 0, s_{64} = 1$
23	b_{27}	$-$	$z_{26} = 0, s_{29} = 1, s_{51} = 0, s_{72} = 1$
24	b_{26}	$-$	$z_{25} = 0, s_{28} = 1, s_{50} = 0, s_{71} = 1$
25	b_{78}	$-$	$z_{22} = 0, s_{25} = 1, s_{47} = 0, s_{68} = 1$
26	b_{79}	$-$	$z_{23} = 0, s_{26} = 1, s_{48} = 0, s_{69} = 1$
27	b_0	$-$	$z_{24} = 0, s_{27} = 1, s_{49} = 0, s_{70} = 1$

Table 4: The deduced state bits of Grain-v1

value. Note that the 16-bit padding value is a 16-bit string of ones used in the key initialization of Grain-v1 cipher [19]. Then accordingly, generate a 160-bit keystream segment, and memorize the three values (the initial state, the observed state, the 160-bit keystream segment).

Step 3: Output a three column table $\mathbf{T}$ with $2^{81-16-l} = 2^{65-l}$ rows. Notice that the size of the table is reduced by a factor 2^{-16} since the same padding scheme is used, thus applying some sort of 16-bit filtering condition.

For Grain-v1, we assume the availability of 2^r keystream sequences each of length 2^s bits generated using different initial values, where $2^r \ll 2^{64}$ and $2^s \ll 2^{80}$. Within this sample of 2^{r+s} bits in total it is expected that 2^{r+s-28} many segments has the prefix consisting of 28 zeros, but similarly to [22] we only collect the segments with a desired prefix thus ignoring segments with other prefixes. This means that the observed data is actually 2^{r+s} bits but only a portion of 2^{r+s-28} bits is used.

The online phase of the attack consists of the steps described below.

> **Inputs:** The trade-off parameter l, $2 \leq l \leq 15$, and the table $\mathbf{T}$ from the off-line phase.
>
> **Step a:** Construct a sample space consisting of $[\frac{2^{r+s-28}}{320}]$ different 320-bit keystream segments generated by Grain-v1, with a 28-zero prefix.
>
> **Step b:** For each 320-bit segment of the $[\frac{2^{r+s-28}}{320}]$ samples, perform the steps below:
>
> (1) If the first 160 bits of the segment match with some entry in the third column of $\mathbf{T}$, say S^*, retrieve this internal state S^*.
>
> (2) Using this internal state S^*, generate a 320-bit keystream segment. If the generated 320-bit keystream segment and the considered sample segment are identical, then the internal state S^* is the correct one. Otherwise, repeat the procedure with the next sample.

The complexity analysis of the attack is as follows. Similarly to the complexity evaluation in [22], taking into account the sample keystream segments having 28-zeroes as a prefix, we have that the data requirement is about $2^{r+s} = 2^{51+28+l} = 2^{79+l}$ bits. This means that the special state bits of the LFSR (51 bits in total) in Table 3 will occur with probability 1. This amount of data also implies that a unique secret state (and consequently the secret key) is recovered since the probability of recovering a wrong secret state is about $2^{79+l-320} = 2^{-241+l} \approx 0$. The time complexity of the preprocessing phase is $O(2^{81-l})$ operations, which corresponds to Step 2 above. The memory complexity of the preprocessing phase is about $2^{65-l} \times 160 \times 3 \approx 2^{73.91-l}$ bits.

The time complexity of the internal state recovery (on-line phase) is about 2^{51+l} operations. One should remark that these operations are simple table look-ups if a state recovery attack is performed, whereas in the case of a key recovery attack the operations are more complex due to the backward computation of the secret key bits. The memory complexity of the internal state recovery (on-line phase) is about $2^{73.91-l}$ bits, which corresponds to the size of the table $\mathbf{T}$.

In particular, if for instance $l = 15$, then the size of the observed sample is about 2^{94} bits, e.g. the availability of 2^{45} keystream sequences each of length 2^{49} bits generated for different initial values is assumed. The time complexity of preprocessing is about 2^{66} operations and the memory complexity of this phase is about $2^{58.91}$ bits. The time complexity of the internal state recovery (on-line phase) is also about 2^{66} operations. The space complexity of the internal state recovery (on-line phase) is also about $2^{58.91}$ bits. The success rate of this attack is one. The attack parameters are summarized in Table 1.

Remark 1 *The core idea of our attack is to first eliminate the dependence on b_{t+63} (i.e., the variable x_4) in z_t. Actually, we can use another strategy to eliminate this dependence. For instance, by letting $x_2 = 0$, we would have $h(x) = x_1 \oplus x_4 \oplus x_0x_3 \oplus x_3x_4$. It is clear that $x_3 \cdot h(x) = (x_0 \oplus x_1)x_3$. Then, by additionally requiring that $z_t = 0$ and $\sum_{j\in A} b_{t+j} = 0$, we have $(s_{t+3} \oplus s_{t+25}) \cdot s_{t+64} = 0$ so that the dependence on b_{t+63} in z_t (or $h(x)$) is eliminated. Therefore, a similar attack on Grain-v1 can be mounted using the above observations.*

3.2 Attacking Grain-128

The internal state of Grain-128 has 256 bits, which consists of a 128-bit LFSR (denoted as $(s_0, \cdots, s_{127})$) and a 128-bit NFSR (denoted as $(b_0, \cdots, b_{127})$). The update functions of both the LFSR and NFSR are respectively defined as follows.

$$\begin{aligned} s_{t+128} &= s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96} \\ b_{t+128} &= s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus b_{t+11}b_{t+13} \\ &\quad \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84} \end{aligned} \tag{9}$$

The output function is defined as

$$z_t = \sum_{j\in A} b_{t+j} \oplus h(b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79}, s_{t+95}) \oplus s_{t+93},$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$. Similarly to Grain-v1, the output bit z_t is also filtered by a nonlinear function $h(x_0, \cdots, x_8)$, where

$$h(x_0, \cdots, x_8) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8.$$

The variables $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, respectively, correspond to the tap positions

$$b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79}, s_{t+95} \text{ (or } s_{t+94}).$$

Grain-128 is initialized with a 96-bit IV that are input to the LFSR (the remaining bits of the LFSR are filled with ones), whereas the secret 128-bit key is loaded into the NFSR. Then the cipher is clocked 256 times without producing any keystream, but feeding the output bits back into both the LFSR and the NFSR.

The main difference between Grain-128 and Grain-128a is a modification of the NFSR update function given by,

$$\begin{aligned} s_{t+128} &= s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96}, \\ b_{t+128} &= s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus b_{t+11}b_{t+13} \\ &\quad \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84} \\ &\quad \oplus b_{t+22}b_{t+24}b_{t+25} \oplus b_{t+70}b_{t+78}b_{t+82} \oplus b_{t+88}b_{t+92}b_{t+93}b_{t+95}. \end{aligned} \tag{10}$$

Its pre-output function is defined as,

$$y_t = \sum_{j\in A} b_{t+j} \oplus h(b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79}, s_{t+94}) \oplus s_{t+93},$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$. The pre-output function $y_t(\cdot)$ shares the same filtering nonlinear function $h(x_0, \cdots, x_8)$ as Grain-128, where

$$h(x_0, \cdots, x_8) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8.$$

In difference to Grain-128, the variables $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ of the filtering nonlinear function in Grain-128a 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}, s_{t+94}, respectively (the bit s_{t+95} is replaced by s_{t+94}).

Grain-128a has two different modes of operation, i.e., the first mode is with authentication ($IV_0 = 1$), whereas the second mode is without authentication ($IV_0 = 0$). For $IV_0 = 1$, the keystream output function is defined as $z_t = y_{64+2t}, (t \geq 0)$, i.e., every second pre-output bit is used as the keystream bit. Notice that the first 64 pre-output bits and the other half of Grain-128a are used for generating the MAC, see [3]. Moreover, for $IV_0 = 0$, the keystream output function is defined as $z_t = y_t$, for $t \geq 0$, i.e., the pre-output bits of Grain-128a are directly used as the keystream bits.

3.2.1 Attack details for Grain-128 or Grain-128a without authentication

Note that both Grain-128 and Grain-128a share the same filtering function h. The variables $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ respectively correspond to the tap positions

$$b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79}, s_{t+94} \text{ (or } s_{t+95}).$$

Since the approach taken here goes along the same lines as previously described for Grain-v1, we give just a rough guidelines of these steps. More precisely, if we let $x_0 = 0, x_5 = 0$ (i.e., $b_{t+12} = 0, s_{t+42} = 0$), then we have $h(x) = x_2x_3 \oplus x_6x_7$, which does not depend on x_4. In this case we assign $b_{t+12} = 0$, $s_{t+42} = 0$, for $t = 0, \ldots, 41$ and $t = 46, \ldots, 49$, thus specifying 92 bits of the secret state in total. Similarly, from the structure of Grain-128 and Grain-128a without authentication, we observe that the internal state variables satisfy the following equation,

$$b_{t+89} = z_t \oplus b_{t+2} \oplus b_{t+15} \oplus b_{t+36} \oplus b_{t+45} \oplus b_{t+64} \oplus b_{t+73} \oplus s_{t+93} \oplus s_{t+13}s_{t+20} \oplus s_{t+60}s_{t+79},$$

where $b_{t+12} = 0, s_{t+42} = 0$ and $t \in \{0, \cdots, 41, 46, \ldots, 49\}$. Moreover, using the above relationship and by additionally specifying 118 secret state bits, we are able to deduce 46 bits of the internal state of Grain-128 or Grain-128a without authentication directly. The whole procedure is summarized in Table 5 and Table 6, see Appendix B.

Lemma 2 *Given the value of 210 particular state bits of Grain-128 (or Grain-128a without authentication model) and the corresponding 46 keystream output bits generated from the internal state, then the remaining 46 bits of the internal state can be recovered directly.*

3.2.2 The complexity analysis

Similarly to the internal state recovery attack on Grain-v1, the attack on Grain-128 or Grain-128a without authentication also has the preprocessing phase (off-line) and the

internal state recovery phase (on-line). In this case, we assume that the cipher generates a keystream segment with 46-zero in a 50-bit prefix, where $z_t = 0$, for $t = 0, \ldots, 41, 46, \ldots, 49$.

The preprocessing phase contains the following steps.

Input : The trade-off parameter l, $0 \leq l \leq 13$.

Step 1: Set $b_{t+12} = 0$, $s_{t+42} = 0$, for $t \in \{0, \ldots, 41, 46, \ldots, 49\}$, and $s_{13} = 0, \ldots, s_{l+12} = 0$.

Step 2: Based on the above constraints, for each of $2^{256-46\times2-46-l} = 2^{118-l}$ possible patterns of internal state of Grain-128 perform the steps below:

(1) For each given pattern, compute the bits $b_0, b_1, b_{89}, \ldots, b_{127}, s_6, s_9, \ldots, s_{12}$, by using Table 5 and Table 6, cf. Lemma 2.

(2) Recover the initial state (i.e.,128-bit secret key, 96-bit initial value, and 32-bit padding value). Note that the 32-bit padding value is a 32-bit one string (or consists of a single 0 and 31 ones - Grain-128a). Then accordingly, generate a 256-bit keystream segment by Grain-128 (or Grain-128a), and memorize the triplet (the initial state, the observed state, 256-bit keystream segment).

Step 3: Output the three column table $\mathbf{T}$ containing $2^{118-l-32} = 2^{86-l}$ rows.

Similarly to the on-line attack procedure for Grain-v1, the internal state recovery consists of the following steps.

Input: The trade-off parameter l, $0 \leq l \leq 13$, and the table $\mathbf{T}$ created in the preprocessing phase.

Step a: Collect a sample space generated by Grain-128(a) consisting of $[\frac{2^{r+s-46}}{512}]$ different 512-bit keystream segment with 46-zero in the 50-bit prefix by Grain-128 (or Grain-128a), where $z_t = 0$, for $t = (0, \cdots, 41, 46, \cdots, 49)$.

Step b: For each 512-bit segment of $[\frac{2^{r+s-46}}{512}]$ samples, perform the steps below:

(1) If the first 256 bits of the segment matches with the third column in $\mathbf{T}$, retrieve the corresponding internal state S^* from the table.

(2) Using the internal state S^*, generate a new 512-bit keystream segment. Moreover, check the new 512-bit keystream segment with the considered sample segment. If the two 512-bit keystream segments are identical, then the internal state S^* is the correct one. Otherwise, repeat the procedure with the next sample segment.

The complexity estimates are summarized below under the assumption that a sample collection of about $2^{r+s} = 2^{46+46\times2+l} = 2^{138+l}$ bits is available to the attacker. The preprocessing phase has the following complexities :

- Time complexity : It corresponds to Step 2 and requires 2^{118-l} operations for constructing the table $\mathbf{T}$.
- Memory complexity: To store the table $\mathbf{T}$ $2^{86-l} \times 256 \times 3 \approx 2^{95.59-l}$ bits of memory are required.

The on-line phase of the attack is characterized by the following complexity estimates:

- Time complexity : The internal state (key) recovery requires 2^{92+l} operations.
- Memory complexity: The size of the input table $\mathbf{T}$ which is $2^{95.59-l}$ bits.

Moreover, an "optimal" choice of the trade-off parameter $l = 13$ gives the following complexity estimates. The complexity of the sample collection is about 2^{151} bits, e.g. availability of 2^{75} keystream sequences each of length 2^{76} bits generated for different initial values. The time complexity of the preprocessing phase is about 2^{105} operations, whereas the memory complexity is about $2^{82.59}$ bits. The time complexity of the internal state (key) recovery (online phase) is about 2^{105} operations using table $\mathbf{T}$ of $2^{82.59}$ bits as the input, see Table 2. As before, the success rate of this attack is one.

4 Summary

In this work a simple but powerful attack on Grain family of ciphers was presented. To resist the new attack, the design must be more robust against the possibility of recovering modest number of internal state bits under certain constraint conditions. For Grain-v1, whose output function is $z_t = \sum_{j \in A} b_{t+j} \oplus h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63})$, where $A = \{1, 2, 4, 10, 31, 43, 56\}$, the problem seems to be the placement of the NFSR bit s_{t+56} which has a long delay before its update. In order to thwart BSW like attacks, we suggest a replacement of the tap position 56 in $A = \{1, 2, 4, 10, 31, 43, 56\}$. For instance, if a tap position i, for $70 < i < 79$, is used instead, then the adversary may only deduce about $79 - i$ equations (internal state bits) by using the constraint conditions as discussed in this article. Similarly, for Grain-128 and Grain-128a, the tap position 89 in $A = \{2, 15, 36, 45, 64, 73, 89\}$ may be replaced by some other tap position, for instance the output may be taken at position i, where $120 < i < 127$.

References

[1] Afzal, M., and Masood, A.: Algebraic Cryptanalysis of a NLFSR Based Stream Cipher. In: *Information and Communication Technologies: From Theory to Applications (ICTTA 2008)*, Umayyad Palace, Damascus, Syria, pp. 1-6, 2008.

[2] Agren, M., Hell, M., Johansson, T., and Meier,W.: A New Version of Grain-128 with Authentication. In: *Symmetric Key Encryption Workshop (SKEW) 2011*, Lyngby, Denmark, Feb. 2011. Available: http://skew2011.mat.dtu.dk/

[3] Agren, M., Hell, M., Johansson, T., and Meier, W.: A New Version of Grain-128 with Optional Authentication. *International Journal of Wireless and Mobile Computing*, vol. 5, no. 1, pp. 48–59, 2011.

[4] Babbage, S.: Improved exhaustive search attacks on stream ciphers. In: *European Convention on Security and Detection 1995*. IEE Conference Publication, pp. 161C166. IEEE Press, New York, 1995.

[5] Bjorstad, T.: Cryptanalysis of Grain using Time/Memory/Data Tradeoffs. Available at: http://www.ii.uib.no/ tor/pdf/grain.pdf

[6] Biryukov, A., and Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In *Proceedings of ASIACRYPT 2000*. LNCS vol. 1976, Springer-Verlag, pp. 1-13, 2000.

[7] Biryukov, A., Shamir, A., and Wagner, D.: Real Time Cryptanalysis of A5/1 on a PC. In: *Proceedings of FSE 2000*, LNCS vol. 1978, pp. 1-18, Springer, Heidelberg, 2001.

[8] Banik, S., Maitra, S., and Sarkar, S.: Some Results on Related Key-IV Pairs of Grain. In: *Security, Privacy, and Applied Cryptography Engineering (SPACE 2012)*. LNCS, vol. 7644, pp. 94-110. Springer, Heidelberg, 2012.

[9] Banik, S., Maitra, S., and Sarkar, S.: A Differential Fault Attack on the Grain Family of Stream Ciphers. In: *Proc. 14th Int. Workshop on Cryptographic Hardware and Embedded System CHES 2012*. LNCS, vol. 7428, pp. 122-139. Springer, Heidelberg, 2012.

[10] Banik, S., Maitra, S., Sarkar, S., and Sonmez, T.: Chosen IV Related Key Attack on Grain-128a. In: *ACISP 2013*. LNCS, vol. 7959, pp. 13-26. Springer, Heidelberg, 2013.

[11] Canniere, C., Kucuk, O., and Preneel, B.: Analysis of Grains Initialization Algorithm. In: *AFRICACRYPT 2008*. LNCS, vol. 5023, pp. 276-289, Springer, Heidelberg, 2008.

[12] Dinur, I. and Shamir, A.: Breaking Grain-128 with Dynamic Cube Attacks. In: *Fast Software Encryption - FSE'2011*. LNCS, vol. 6733, pp. 167-187. Springer, Heidelberg, 2011.

[13] Dinur, I., Guneysu, T., Paar, C., Shamir, A. and Zimmermann, R.: An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware. In: *Progress in Cryptology - ASIACRYPT'2011*. LNCS, vol. 7073, pp. 327-373. Springer, Heidelberg, 2011.

[14] Ding, L., Guan, J.: Related Key Chosen IV Attack on Grain-128a Stream Cipher. *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 5, pp. 803-809, 2013.

[15] Ding, L., Jin, C., Guan, J., and Qi, C.: New Treatment of the BSW Sampling and Its Applications to Stream Ciphers In: *AFRICACRYPT 2014*, LNCS 8469, pp. 136-146, Springer International Publishing Switzerland, 2014.

[16] Golić, J.D.: Cryptanalysis of alleged A5 stream cipher. In: *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 239C255. Springer, Heidelberg, 1997.

[17] Hellman, M.: A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, vol. 26, no.4, 401C406, 1980.

[18] Hell, M., Johansson, T., and Meier, W.: Grain - A Stream Cipher for Constrained Environments. *ECRYPT Stream Cipher Project Report*, 2005. Available: http://www.ecrypt.eu.org/stream

[19] Hell, M., Johansson, T., and Meier, W.,: Grain: A Stream Cipher for Constrained Environments. *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, 2007. pp. 86-93, 2007.

[20] Hell, M., Johansson, T., Maximov,A., and Meier, W.: A Stream Cipher Proposal: Grain-128. In: *IEEE International Symposium on Information Theory ISIT 2006*, Seattle, Washington, USA, pp.1615-1618, 2006.

[21] Lee, Y., Jeong, K., Sung, J., and Hong, S.: Related-Key Chosen IV Attacks on Grain- v1 and Grain-128. In: *ACISP 2008*, LNCS, vol. 5107, pp. 321-335, Springer, Heidelberg, 2008.

[22] Mihaljevic, M., Gangopadhyay, S., Paul, G., and Imai, H.: Internal State Recovery of Grain-v1 Employing Normality Order of the Filter Function. *IET Information Security*, vol.6, no.2, pp.55-64, 2012.

[23] Mihaljevic, M., Gangopadhyay, S., Paul, G., and Imai, H.: Generic Cryptographic Weakness of K-normal Boolean Functions in Certain Stream Ciphers and Cryptanalysis of Grain-128. *Periodica Mathematica Hungarica*, vol. 65, no. 2, pp. 205-227, 2012.

[24] Rahimi, M., Barmshory, M., Mansouri, M., and Aref, M.: Dynamic Cube Attack on Grain-v1. Cryptology ePrint Archive: Report 2013/268. Available: http://eprint.iacr.org/2013/268

[25] Stankovski, P.: Greedy Distinguishers and Nonrandomness Detectors. In: *Progress in Cryptology-INDOCRYPT 2010*. LNCS, vol. 6498, pp. 210-226. Springer, Heidelberg, 2010.

[26] The ECRYPT Stream Cipher Project. eSTREAM Portfolio Revision, 2008. Available: http://www.ecrypt.eu.org/stream

[27] Zhang, H., and Wang, X.: Cryptanalysis of Stream Cipher Grain Family, Cryptology ePrint Archive, Report 2009/109. Available: http://eprint.iacr.org/.

[28] Zhang, B., Li, Z.: Near Collision Attack on the Grain v1 Stream Cipher. In: *proceedings of FSE 2013*, LNCS, vol. 8424, pp. 518-538, Springer, Heidelberg, 2014.

Appendix A (The step 16 to the Step 27 in Table 4)

The step 16 in Table 4 aims at recovering the secret bit b_{29} of the NFSR. We compute b_{t+10} in OM 2 for t=19,

$\underline{b_{29}} = z_{19} \oplus b_{20} \oplus b_{21} \oplus b_{23} \oplus b_{50} \oplus b_{62} \oplus \underline{b_{75} \oplus s_{83}}$, where $s_{22} = 1, s_{44} = 0, s_{65} = 1$.

Using the update function of LFSR, we have

$s_{83} = s_{65} \oplus s_{54} \oplus s_{41} \oplus s_{26} \oplus s_{16} \oplus s_3$.

Hence we only need to guess one new bit b_{75} to compute b_{29}.

The step 17 in Table 4 aims at recovering the secret bit b_{30} of the NFSR. We compute b_{t+10} in OM 2 for $t = 20$,

$\underline{b_{30}} = z_{20} \oplus b_{21} \oplus b_{22} \oplus b_{24} \oplus b_{51} \oplus b_{63} \oplus \underline{b_{76} \oplus s_{84}}$, where $s_{23} = 1, s_{45} = 0, s_{66} = 1$.

Using the update function of LFSR, we have

$s_{84} = s_{66} \oplus s_{55} \oplus s_{42} \oplus s_{27} \oplus s_{17} \oplus s_4$.

Hence we only need to guess one new bit b_{76} to compute b_{30}.

The step 18 in Table 4 aims at recovering the secret bit b_{77} of the NFSR. We compute b_{t+56} in OM 2 for $t = 21$,

$\underline{b_{77}} = z_{21} \oplus b_{22} \oplus b_{23} \oplus b_{25} \oplus b_{52} \oplus b_{64} \oplus b_{31} \oplus \underline{s_{85}}$, where $s_{24} = 1, s_{46} = 0, s_{67} = 1$.

Using the update function of LFSR, we have

$s_{85} = s_{67} \oplus s_{56} \oplus s_{43} \oplus s_{28} \oplus s_{18} \oplus s_5$.

Hence we need to guess no more new bit to compute b_{77}.

The step 19 in Table 4 aims at recovering the secret bit s_0 of the LFSR. Notice that,

$b_{35} = z_{25} \oplus \underline{b_{26} \oplus b_{27}} \oplus b_{29} \oplus b_{56} \oplus b_{68} \oplus \underline{b_{81} \oplus s_{89}}$, where $s_{21} = 1, s_{43} = 0, s_{64} = 1$.

$b_{26} = z_{16} \oplus b_{17} \oplus b_{18} \oplus b_{20} \oplus b_{47} \oplus b_{59} \oplus \underline{b_{72} \oplus s_{80}}$, where $s_{19} = 1, s_{41} = 0, s_{62} = 1$.

$b_{27} = z_{17} \oplus b_{18} \oplus b_{19} \oplus b_{21} \oplus b_{48} \oplus b_{60} \oplus \underline{b_{73} \oplus s_{81}}$, where $s_{20} = 1, s_{42} = 0, s_{63} = 1$.

On the other hand, we have

$s_{80} = s_{62} \oplus s_{51} \oplus s_{38} \oplus s_{23} \oplus s_{13} \oplus \underline{s_0}$.

$s_{81} = s_{63} \oplus s_{52} \oplus s_{39} \oplus s_{24} \oplus s_{14} \oplus \underline{s_1}$.

$s_{89} = s_{71} \oplus s_{60} \oplus s_{47} \oplus s_{32} \oplus s_{22} \oplus s_9$.

$b_{81} = \underline{s_1} \oplus b_{63} \oplus b_{61} \oplus b_{53} \oplus b_{46} \oplus b_{38} \oplus b_{34} \oplus b_{29} \oplus b_{22} \oplus b_{15} \oplus b_{10} \oplus b_1 \oplus b_{64}b_{61} \oplus b_{38}b_{34} \oplus b_{16}b_{10} \oplus b_{61}b_{53}b_{46} \oplus b_{34}b_{29}b_{22} \oplus b_{64}b_{46}b_{29}b_{10} \oplus b_{61}b_{53}b_{38}b_{34} \oplus b_{64}b_{61}b_{22}b_{16} \oplus b_{64}b_{61}b_{53}b_{46}b_{38} \oplus b_{34}b_{29}b_{22}b_{16}b_{10} \oplus b_{53}b_{46}b_{38}b_{34}b_{29}b_{22}$.

Moreover, we have

$b_{35} = z_{25} \oplus z_{16} \oplus b_{17} \oplus b_{18} \oplus b_{20} \oplus b_{47} \oplus b_{59} \oplus \underline{b_{72}} \oplus s_{62} \oplus s_{51} \oplus s_{38} \oplus s_{23} \oplus s_{13} \oplus \underline{s_0} \oplus z_{17} \oplus b_{18} \oplus b_{19} \oplus b_{21} \oplus b_{48} \oplus b_{60} \oplus \underline{b_{73}} \oplus s_{63} \oplus s_{52} \oplus s_{39} \oplus s_{24} \oplus s_{14} \oplus \underline{s_1} \oplus b_{29} \oplus b_{56} \oplus b_{68} \oplus \underline{s_1} \oplus b_{64} \oplus b_{61} \oplus b_{53} \oplus b_{46} \oplus b_{38} \oplus b_{34} \oplus b_{29} \oplus b_{22} \oplus b_{16} \oplus b_{10} \oplus b_1 \oplus b_{64}b_{61} \oplus b_{38}b_{34} \oplus b_{16}b_{10} \oplus b_{61}b_{53}b_{46} \oplus b_{34}b_{29}b_{22} \oplus b_{64}b_{46}b_{29}b_{10} \oplus b_{61}b_{53}b_{38}b_{34} \oplus b_{64}b_{61}b_{22}b_{16} \oplus b_{64}b_{61}b_{53}b_{46}b_{38} \oplus b_{34}b_{29}b_{22}b_{16}b_{10} \oplus b_{53}b_{46}b_{38}b_{34}b_{29}b_{22} \oplus s_{89}$

Therefore, we have

$\underline{s_0} = b_{35} \oplus z_{25} \oplus z_{16} \oplus b_{17} \oplus b_{18} \oplus b_{20} \oplus b_{47} \oplus b_{59} \oplus \underline{b_{72}} \oplus s_{62} \oplus s_{51} \oplus s_{38} \oplus s_{23} \oplus s_{13} \oplus z_{17} \oplus b_{18} \oplus b_{19} \oplus b_{21} \oplus b_{48} \oplus b_{60} \oplus \underline{b_{73}} \oplus s_{63} \oplus s_{52} \oplus s_{39} \oplus s_{24} \oplus s_{14} \oplus b_{29} \oplus b_{56} \oplus b_{68} \oplus b_{64} \oplus b_{61} \oplus b_{53} \oplus b_{46} \oplus b_{38} \oplus b_{34} \oplus b_{29} \oplus b_{22} \oplus b_{16} \oplus b_{10} \oplus b_1 \oplus b_{64}b_{61} \oplus b_{38}b_{34} \oplus b_{16}b_{10} \oplus b_{61}b_{53}b_{46} \oplus b_{34}b_{29}b_{22} \oplus b_{64}b_{46}b_{29}b_{10} \oplus b_{61}b_{53}b_{38}b_{34} \oplus b_{64}b_{61}b_{22}b_{16} \oplus b_{64}b_{61}b_{53}b_{46}b_{38} \oplus b_{34}b_{29}b_{22}b_{16}b_{10} \oplus b_{53}b_{46}b_{38}b_{34}b_{29}b_{22} \oplus s_{89}$.

It is clear that we need to guess two new bits (b_{72}, b_{73}) for obtaining the bit s_0.

The step 20 in Table 4 aims at recovering the secret bit s_1 of the LFSR. Similarly, we can deduce that we only need to guess one new bits b_{74} for obtaining the bit s_1 by using exactly the same way as step 19.

The step 21 in Table 4 aims at recovering the secret bit b_{28} of the NFSR. We compute b_{t+1} in OM 2 for $t = 27$,

$\underline{b_{28}} = z_{27} \oplus b_{29} \oplus b_{31} \oplus b_{37} \oplus b_{58} \oplus b_{70} \oplus \underline{b_{83} \oplus s_{91}}$, where $s_{30} = 1, s_{52} = 0, s_{73} = 1$.

Using the update function of LFSR and NFSR, we have

$s_{91} = s_{73} \oplus s_{62} \oplus s_{49} \oplus s_{34} \oplus s_{24} \oplus s_{11}$.

$b_{83} = s_3 \oplus b_{65} \oplus b_{63} \oplus b_{55} \oplus b_{48} \oplus b_{40} \oplus b_{36} \oplus b_{31} \oplus b_{24} \oplus b_{17} \oplus b_{12} \oplus b_3 \oplus b_{66}b_{63} \oplus b_{40}b_{36} \oplus b_{18}b_{12} \oplus b_{63}b_{55}b_{48} \oplus b_{36}b_{31}b_{24} \oplus b_{66}b_{48}b_{31}b_{12} \oplus b_{63}b_{55}b_{40}b_{36} \oplus b_{66}b_{63}b_{24}b_{18} \oplus b_{66}b_{63}b_{55}b_{48}b_{40} \oplus b_{36}b_{31}b_{24}b_{18}b_{12} \oplus b_{55}b_{48}b_{40}b_{36}b_{31}b_{24}$.

Hence we need to guess no more new bit to compute b_{28}.

The step 22 in Table 4 aims at recovering the secret bit s_2 of the LFSR. We compute s_{t+64} in OM 2 for $t = 18$,

$\underline{s_{82}} = z_{18} \oplus b_{19} \oplus b_{20} \oplus b_{22} \oplus b_{28} \oplus b_{49} \oplus b_{61} \oplus b_{74}$, where $s_{21} = 1, s_{43} = 0, s_{64} = 1$.

Using the update function of LFSR, we have

$s_{82} = s_{64} \oplus s_{53} \oplus s_{40} \oplus s_{25} \oplus s_{15} \oplus \underline{s_2}$.

Hence we need to guess no more new bit to compute s_2.

The step 23 in Table 4 aims at recovering the secret bit b_{27} of the NFSR. We compute b_{t+1} in OM 2 for $t = 26$,

$\underline{b_{27}} = z_{26} \oplus b_{28} \oplus b_{30} \oplus b_{36} \oplus b_{57} \oplus b_{69} \oplus \underline{b_{82} \oplus s_{90}}$, where $s_{29} = 1, s_{51} = 0, s_{72} = 1$.

Using the update function of LFSR and NFSR, we have

$s_{90} = s_{72} \oplus s_{61} \oplus s_{48} \oplus s_{33} \oplus s_{23} \oplus s_{10}$.

$b_{82} = s_2 \oplus b_{64} \oplus b_{62} \oplus b_{54} \oplus b_{47} \oplus b_{39} \oplus b_{35} \oplus b_{30} \oplus b_{23} \oplus b_{16} \oplus b_{11} \oplus b_2 \oplus b_{65}b_{62} \oplus b_{39}b_{35} \oplus b_{17}b_{11} \oplus b_{62}b_{54}b_{47} \oplus b_{35}b_{30}b_{23} \oplus b_{65}b_{47}b_{30}b_{11} \oplus b_{62}b_{54}b_{39}b_{35} \oplus b_{65}b_{62}b_{23}b_{17} \oplus b_{65}b_{62}b_{54}b_{47}b_{39} \oplus b_{35}b_{30}b_{23}b_{17}b_{11} \oplus b_{54}b_{47}b_{39}b_{35}b_{30}b_{23}$.

Hence we need to guess no more new bit to compute b_{27}.

The step 24 in Table 4 aims at recovering the secret bit b_{26} of the NFSR. We compute b_{t+1} in OM 2 for $t = 25$,

$\underline{b_{26}} = z_{25} \oplus b_{27} \oplus b_{29} \oplus b_{35} \oplus b_{56} \oplus b_{68} \oplus \underline{b_{81} \oplus s_{89}}$, where $s_{28} = 1, s_{50} = 0, s_{71} = 1$.

Using the update function of LFSR and NFSR, we have

$s_{89} = s_{71} \oplus s_{60} \oplus s_{47} \oplus s_{32} \oplus s_{22} \oplus s_9$.

$b_{81} = s_1 \oplus b_{63} \oplus b_{61} \oplus b_{53} \oplus b_{46} \oplus b_{38} \oplus b_{34} \oplus b_{29} \oplus b_{22} \oplus b_{15} \oplus b_{10} \oplus b_1 \oplus b_{64}b_{61} \oplus b_{38}b_{34} \oplus b_{16}b_{10} \oplus b_{61}b_{53}b_{46} \oplus b_{34}b_{29}b_{22} \oplus b_{64}b_{46}b_{29}b_{10} \oplus b_{61}b_{53}b_{38}b_{34} \oplus b_{64}b_{61}b_{22}b_{16} \oplus b_{64}b_{61}b_{53}b_{46}b_{38} \oplus b_{34}b_{29}b_{22}b_{16}b_{10} \oplus b_{53}b_{46}b_{38}b_{34}b_{29}b_{22}$.

Hence we need to guess no more new bit to compute b_{26}.

The step 25 in Table 4 aims at recovering the secret bit b_{78} of the NFSR. We compute b_{t+56} in OM 2 for $t = 22$,

$\underline{b_{78}} = z_{22} \oplus b_{23} \oplus b_{24} \oplus b_{26} \oplus b_{53} \oplus b_{65} \oplus b_{32} \oplus \underline{s_{86}}$, where $s_{25} = 1, s_{47} = 0, s_{68} = 1$.

Using the update function of LFSR, we have

$s_{86} = s_{68} \oplus s_{57} \oplus s_{44} \oplus s_{29} \oplus s_{19} \oplus s_6$.

Hence we need to guess no more new bit to compute b_{78}.

The step 26 in Table 4 aims at recovering the secret bit b_{79} of the NFSR. We compute b_{t+56} in OM 2 for $t = 23$,

$\underline{b_{79}} = z_{23} \oplus b_{24} \oplus b_{25} \oplus b_{27} \oplus b_{54} \oplus b_{66} \oplus b_{33} \oplus \underline{s_{87}}$, where $s_{26} = 1, s_{48} = 0, s_{69} = 1$.

Using the update function of LFSR, we have

$s_{87} = s_{69} \oplus s_{58} \oplus s_{45} \oplus s_{30} \oplus s_{20} \oplus s_7$.

Hence we need to guess no more new bit to compute b_{78}.

The step 27 in Table 4 aims at recovering the secret bit b_0 of the NFSR. We compute b_{t+56} in OM 2 for $t = 24$,

$\underline{b_{80}} = z_{24} \oplus b_{25} \oplus b_{26} \oplus b_{28} \oplus b_{55} \oplus b_{67} \oplus b_{34} \oplus \underline{s_{88}}$, where $s_{27} = 1, s_{49} = 0, s_{70} = 1$.

Using the update function of LFSR and NFSR, we have

$s_{88} = s_{70} \oplus s_{59} \oplus s_{46} \oplus s_{31} \oplus s_{21} \oplus s_8$.

$b_{80} = s_0 \oplus b_{62} \oplus b_{60} \oplus b_{52} \oplus b_{45} \oplus b_{37} \oplus b_{33} \oplus b_{28} \oplus b_{21} \oplus b_{14} \oplus b_9 \oplus \underline{b_0} \oplus b_{63}b_{60} \oplus b_{37}b_{33} \oplus b_{15}b_9 \oplus b_{60}b_{52}b_{45} \oplus b_{33}b_{28}b_{21} \oplus b_{63}b_{45}b_{28}b_9 \oplus b_{60}b_{52}b_{37}b_{33} \oplus b_{63}b_{60}b_{21}b_{15} \oplus b_{63}b_{60}b_{52}b_{45}b_{37} \oplus b_{33}b_{28}b_{21}b_{15}b_9 \oplus b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}$.

Hence we need to guess no more new bit to compute b_0.

Appendix B (Recovered bits in Table 5 and Table 6)

For the output function of Grain-128, if we let x_0=0, x_5=0 (i.e., $b_{t+12} = 0$, $s_{t+42} = 0$), then we have $z_t = b_{t+2} \oplus b_{t+15} \oplus b_{t+36} \oplus b_{t+45} \oplus b_{t+64} \oplus b_{t+73} \oplus b_{t+89} \oplus s_{t+93} \oplus s_{t+13}s_{t+20} \oplus s_{t+60}s_{t+79}$.

The step i ($i = 0, \cdots, 34$) in Table 5 (or Table 6) aims at recovering the secret bit b_{i+89} of the NFSR. We compute b_{t+89} for $t = i$,

$b_{i+89} = z_i \oplus b_{i+2} \oplus b_{i+15} \oplus b_{i+36} \oplus b_{i+45} \oplus b_{i+64} \oplus b_{i+73} \oplus s_{i+93} \oplus s_{i+13}s_{i+20} \oplus s_{i+60}s_{i+79}$,where $b_{i+12} = 0, s_{i+42} = 0$.

Clearly, we only need to guess bits $b_{i+2}, b_{i+15}, b_{i+36}, b_{i+45}, b_{i+64}, b_{i+73}, s_{i+93}, s_{i+13}, s_{i+20}, s_{i+60}, s_{i+79}$ to compute b_{i+89}, which is showed in Table 5 (or Table 6).

The step 35 in Table 6 aims at recovering the secret bit s_9 of the LFSR. We compute b_{t+89} for $t = 48$,

$\underline{b_{137}} = z_{48} \oplus b_{50} \oplus b_{63} \oplus b_{84} \oplus b_{93} \oplus b_{112} \oplus b_{121} \oplus \underline{s_{141}} \oplus s_{61}s_{68} \oplus s_{108}s_{127}$,where $b_{60} = 0, s_{90} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{141} = s_{13} \oplus s_{20} \oplus s_{51} \oplus s_{83} \oplus s_{94} \oplus s_{109}$.

$b_{137} = \underline{s_9} \oplus b_9 \oplus b_{35} \oplus b_{65} \oplus b_{100} \oplus b_{105} \oplus b_{12}b_{76} \oplus b_{20}b_{22} \oplus b_{26}b_{27} \oplus b_{36}b_{68} \oplus b_{49}b_{57} \oplus b_{70}b_{74} \oplus b_{77}b_{93} \oplus b_{31}b_{33}b_{34} \oplus b_{79}b_{87}b_{91} \oplus b_{97}b_{101}b_{102}b_{104}$.

Hence we need to guess no more new bit to compute s_9.

The step 36 in Table 6 aims at recovering the secret bit s_{10} of the LFSR. We compute b_{t+89} for $t = 49$,

$\underline{b_{138}} = z_{49} \oplus b_{51} \oplus b_{64} \oplus b_{85} \oplus b_{94} \oplus b_{113} \oplus b_{122} \oplus \underline{s_{142}} \oplus s_{62}s_{69} \oplus s_{109}\underline{s_{128}}$, where $b_{61} = 0, s_{91} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{128} = \underline{s_0 \oplus s_7} \oplus s_{38} \oplus s_{70} \oplus s_{81} \oplus s_{96}$.

$s_{142} = s_{14} \oplus s_{21} \oplus s_{52} \oplus s_{84} \oplus s_{95} \oplus s_{110}$.

$b_{138} = \underline{s_{10}} \oplus b_{10} \oplus b_{36} \oplus b_{66} \oplus b_{101} \oplus b_{106} \oplus b_{13}b_{77} \oplus b_{21}b_{23} \oplus b_{27}b_{28} \oplus b_{37}b_{69} \oplus b_{50}b_{58} \oplus b_{71}b_{75} \oplus b_{78}b_{94} \oplus b_{32}b_{34}b_{35} \oplus b_{80}b_{88}b_{92} \oplus b_{98}b_{102}b_{103}b_{105}$.

Hence we need to guess two new bits s_0, s_7 to compute s_{10}.

The step 37 in Table 6 aims at recovering the secret bit b_{124} of the NFSR. We compute b_{t+89} for $t = 35$,

$\underline{b_{124}} = z_{35} \oplus b_{37} \oplus b_{50} \oplus b_{71} \oplus b_{80} \oplus b_{99} \oplus b_{108} \oplus \underline{s_{128}} \oplus s_{48}s_{55} \oplus s_{95}s_{114}$,where $b_{47} = 0, s_{77} = 0$.

Using the update function of LFSR, we have

$s_{128} = s_0 \oplus s_7 \oplus s_{38} \oplus s_{70} \oplus s_{81} \oplus s_{96}$.

Hence we need to guess no more new bit to compute b_{124}.

The step 38 in Table 6 aims at recovering the secret bit b_{125} of the NFSR. We compute b_{t+89} for $t = 36$,

$\underline{b_{125}} = z_{36} \oplus b_{38} \oplus b_{51} \oplus b_{72} \oplus b_{81} \oplus b_{100} \oplus b_{109} \oplus \underline{s_{129}} \oplus s_{49}s_{56} \oplus s_{96}s_{115}$,where $b_{48} = 0, s_{78} = 0$.

Using the update function of LFSR, we have

$s_{129} = \underline{s_1 \oplus s_8} \oplus s_{39} \oplus s_{71} \oplus s_{82} \oplus s_{97}$.

Hence we need to guess two new bits s_1, s_8 to compute b_{125}.

The step 39 in Table 6 aims at recovering the secret bit b_{126} of the NFSR. We compute b_{t+89} for $t = 37$,

$\underline{b_{126}} = z_{37} \oplus b_{39} \oplus b_{52} \oplus b_{73} \oplus b_{82} \oplus b_{101} \oplus b_{110} \oplus \underline{s_{130}} \oplus s_{50}s_{57} \oplus s_{97}s_{116}$,where $b_{49} = 0, s_{79} = 0$.

Using the update function of LFSR, we have

$s_{130} = \underline{s_2} \oplus s_9 \oplus s_{40} \oplus s_{72} \oplus s_{83} \oplus s_{98}$.

Hence we need to guess one new bits s_2 to compute b_{126}.

The step 40 in Table 6 aims at recovering the secret bit b_{127} of the NFSR. We compute b_{t+89} for $t = 38$,

$\underline{b_{127}} = z_{38} \oplus b_{40} \oplus b_{53} \oplus b_{74} \oplus b_{83} \oplus b_{102} \oplus b_{111} \oplus \underline{s_{131}} \oplus s_{51}s_{58} \oplus s_{98}s_{117}$,where $b_{50} = 0, s_{80} = 0$.

Using the update function of LFSR, we have

$s_{131} = \underline{s_3} \oplus s_{10} \oplus s_{41} \oplus s_{73} \oplus s_{84} \oplus s_{99}$.

Hence we need to guess one new bits s_3 to compute b_{127}.

The step 41 in Table 6 aims at recovering the secret bit s_{11} of the LFSR. We compute b_{t+89} for $t = 46$,

$\underline{b_{135}} = z_{46} \oplus b_{48} \oplus b_{61} \oplus b_{82} \oplus b_{91} \oplus b_{110} \oplus b_{119} \oplus \underline{s_{139}} \oplus s_{59}s_{66} \oplus s_{106}s_{125}$,where $b_{58} = 0, s_{88} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{139} = \underline{s_{11}} \oplus s_{18} \oplus s_{49} \oplus s_{81} \oplus s_{92} \oplus s_{107}$.

$b_{135} = s_7 \oplus b_7 \oplus b_{33} \oplus b_{63} \oplus b_{98} \oplus b_{103} \oplus b_{10}b_{74} \oplus b_{18}b_{20} \oplus b_{24}b_{25} \oplus b_{34}b_{66} \oplus b_{47}b_{55} \oplus b_{68}b_{72} \oplus b_{75}b_{91} \oplus b_{29}b_{31}b_{32} \oplus b_{77}b_{85}b_{89} \oplus b_{95}b_{99}b_{100}b_{102}$.

Hence we need to guess no more new bit to compute s_{11}.

The step 42 in Table 6 aims at recovering the secret bit s_{12} of the LFSR. We compute b_{t+89} for $t = 47$,

$\underline{b_{136}} = z_{47} \oplus b_{49} \oplus b_{62} \oplus b_{83} \oplus b_{92} \oplus b_{111} \oplus b_{120} \oplus \underline{s_{140}} \oplus s_{60}s_{67} \oplus s_{107}s_{126}$,where $b_{59} = 0, s_{89} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{140} = \underline{s_{12}} \oplus s_{19} \oplus s_{50} \oplus s_{82} \oplus s_{93} \oplus s_{108}$.

$b_{136} = s_8 \oplus b_8 \oplus b_{34} \oplus b_{64} \oplus b_{99} \oplus b_{104} \oplus b_{11}b_{75} \oplus b_{19}b_{21} \oplus b_{25}b_{26} \oplus b_{35}b_{67} \oplus b_{48}b_{56} \oplus b_{69}b_{73} \oplus b_{76}b_{92} \oplus b_{30}b_{32}b_{33} \oplus b_{78}b_{86}b_{90} \oplus b_{96}b_{100}b_{101}b_{103}$.

Hence we need to guess no more new bit to compute s_{12}.

The step 43 in Table 6 aims at recovering the secret bit s_6 of the LFSR. We compute b_{t+89} for $t = 41$,

$\underline{b_{130}} = z_{41} \oplus b_{43} \oplus b_{56} \oplus b_{77} \oplus b_{86} \oplus b_{105} \oplus b_{114} \oplus \underline{s_{134}} \oplus s_{54}s_{61} \oplus s_{101}s_{120}$,where $b_{53} = 0, s_{83} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{134} = \underline{s_6} \oplus s_{13} \oplus s_{44} \oplus s_{76} \oplus s_{87} \oplus s_{102}$.

$b_{130} = s_2 \oplus b_2 \oplus b_{28} \oplus b_{58} \oplus b_{93} \oplus b_{98} \oplus b_5b_{69} \oplus b_{13}b_{15} \oplus b_{19}b_{20} \oplus b_{29}b_{61} \oplus b_{42}b_{50} \oplus b_{63}b_{67} \oplus b_{70}b_{86} \oplus b_{24}b_{26}b_{27} \oplus b_{72}b_{80}b_{84} \oplus b_{90}b_{94}b_{95}b_{97}$.

Hence we need to guess no more new bit to compute s_6.

The step 44 in Table 6 aims at recovering the secret bit b_0 of the NFSR. We compute b_{t+89} for $t = 39$,

$\underline{b_{128}} = z_{39} \oplus b_{41} \oplus b_{54} \oplus b_{75} \oplus b_{84} \oplus b_{103} \oplus b_{112} \oplus \underline{s_{132}} \oplus s_{52}s_{59} \oplus s_{99}s_{118}$,where $b_{51} = 0, s_{81} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{132} = \underline{s_4} \oplus s_{11} \oplus s_{42} \oplus s_{74} \oplus s_{85} \oplus s_{100}$.

$b_{128} = s_0 \oplus \underline{b_0} \oplus b_{26} \oplus b_{56} \oplus b_{91} \oplus b_{96} \oplus b_3b_{67} \oplus b_{11}b_{13} \oplus b_{17}b_{18} \oplus b_{27}b_{59} \oplus b_{40}b_{48} \oplus b_{61}b_{65} \oplus b_{68}b_{84} \oplus b_{22}b_{24}b_{25} \oplus b_{70}b_{78}b_{82} \oplus b_{88}b_{92}b_{93}b_{95}$.

Hence we need to guess one new bits s_4 to compute b_0.

The step 45 in Table 6 aims at recovering the secret bit b_1 of the NFSR. We compute b_{t+89} for $t = 40$,

$\underline{b_{129}} = z_{40} \oplus b_{42} \oplus b_{55} \oplus b_{76} \oplus b_{85} \oplus b_{104} \oplus b_{113} \oplus \underline{s_{133}} \oplus s_{53}s_{60} \oplus s_{100}s_{119}$,where $b_{52} = 0, s_{82} = 0$.

Using the update function of LFSR and NFSR, we have

$s_{133} = \underline{s_5} \oplus s_{12} \oplus s_{43} \oplus s_{75} \oplus s_{86} \oplus s_{101}$.

$b_{129} = s_1 \oplus \underline{b_1} \oplus b_{27} \oplus b_{57} \oplus b_{92} \oplus b_{97} \oplus b_4b_{68} \oplus b_{12}b_{14} \oplus b_{18}b_{19} \oplus b_{28}b_{60} \oplus b_{41}b_{49} \oplus b_{62}b_{66} \oplus b_{69}b_{85} \oplus b_{23}b_{25}b_{26} \oplus b_{71}b_{79}b_{83} \oplus b_{89}b_{93}b_{94}b_{96}$.

Hence we need to guess one new bits s_5 to compute b_1.

Step	Recovered bit	Fixed NFSR and LFSR bits	Constraint conditions
0	b_{89}	$b_2, b_{64}, b_{73}, s_{13}, s_{20}, s_{93}$	$z_0 = 0, b_{12} = 0, s_{42} = 0$
1	b_{90}	$b_3, b_{65}, b_{74}, s_{14}, s_{21}, s_{94}$	$z_1 = 0, b_{13} = 0, s_{43} = 0$
2	b_{91}	$b_4, b_{66}, b_{75}, s_{15}, s_{22}, s_{95}$	$z_2 = 0, b_{14} = 0, s_{44} = 0$
3	b_{92}	$b_5, b_{67}, b_{76}, s_{16}, s_{23}, s_{96}$	$z_3 = 0, b_{15} = 0, s_{45} = 0$
4	b_{93}	$b_6, b_{68}, b_{77}, s_{17}, s_{24}, s_{97}$	$z_4 = 0, b_{16} = 0, s_{46} = 0$
5	b_{94}	$b_7, b_{69}, b_{78}, s_{18}, s_{25}, s_{84}, s_{98}$	$z_5 = 0, b_{17} = 0, s_{47} = 0$
6	b_{95}	$b_8, b_{70}, b_{79}, s_{19}, s_{26}, s_{85}, s_{99}$	$z_6 = 0, b_{18} = 0, s_{48} = 0$
7	b_{96}	$b_9, b_{71}, b_{80}, s_{27}, s_{86}, s_{100}$	$z_7 = 0, b_{19} = 0, s_{49} = 0$
8	b_{97}	$b_{10}, b_{72}, b_{81}, s_{28}, s_{87}, s_{101}$	$z_8 = 0, b_{20} = 0, s_{50} = 0$
9	b_{98}	$b_{11}, b_{54}, b_{82}, s_{29}, s_{102}$	$z_9 = 0, b_{21} = 0, s_{51} = 0$
10	b_{99}	$b_{55}, b_{83}, s_{30}, s_{103}$	$z_{10} = 0, b_{22} = 0, s_{52} = 0$
11	b_{100}	$b_{56}, b_{84}, s_{31}, s_{104}$	$z_{11} = 0, b_{23} = 0, s_{53} = 0$
12	b_{101}	$b_{57}, b_{85}, s_{32}, s_{105}$	$z_{12} = 0, b_{24} = 0, s_{54} = 0$
13	b_{102}	$b_{86}, s_{33}, s_{92}, s_{106}$	$z_{13} = 0, b_{25} = 0, s_{55} = 0$
14	b_{103}	b_{87}, s_{34}, s_{107}	$z_{14} = 0, b_{26} = 0, s_{56} = 0$
15	b_{104}	b_{88}, s_{35}, s_{108}	$z_{15} = 0, b_{27} = 0, s_{57} = 0$
16	b_{105}	s_{36}, s_{109}	$z_{16} = 0, b_{28} = 0, s_{58} = 0$
17	b_{106}	b_{62}, s_{37}, s_{110}	$z_{17} = 0, b_{29} = 0, s_{59} = 0$
18	b_{107}	b_{63}, s_{38}, s_{111}	$z_{18} = 0, b_{30} = 0, s_{60} = 0$
19	b_{108}	s_{39}, s_{112}	$z_{19} = 0, b_{31} = 0, s_{61} = 0$
20	b_{109}	s_{40}, s_{113}	$z_{20} = 0, b_{32} = 0, s_{62} = 0$
21	b_{110}	s_{41}, s_{114}	$z_{21} = 0, b_{33} = 0, s_{63} = 0$
22	b_{111}	s_{115}	$z_{22} = 0, b_{34} = 0, s_{64} = 0$
23	b_{112}	s_{116}	$z_{23} = 0, b_{35} = 0, s_{65} = 0$
24	b_{113}	s_{117}	$z_{24} = 0, b_{36} = 0, s_{66} = 0$
25	b_{114}	s_{118}	$z_{25} = 0, b_{37} = 0, s_{67} = 0$
26	b_{115}	s_{119}	$z_{26} = 0, b_{38} = 0, s_{68} = 0$
27	b_{116}	s_{120}	$z_{27} = 0, b_{39} = 0, s_{69} = 0$
28	b_{117}	s_{121}	$z_{28} = 0, b_{40} = 0, s_{70} = 0$

Table 5: The deduced state bits of Grain-128 and Grain-128a

Step	Recovered bit	Fixed NFSR and LFSR bits	Constraint conditions
29	b_{118}	s_{122}	$z_{29} = 0, b_{41} = 0, s_{71} = 0$
30	b_{119}	s_{123}	$z_{30} = 0, b_{42} = 0, s_{72} = 0$
31	b_{120}	s_{124}	$z_{31} = 0, b_{43} = 0, s_{73} = 0$
32	b_{121}	s_{125}	$z_{32} = 0, b_{44} = 0, s_{74} = 0$
33	b_{122}	s_{126}	$z_{33} = 0, b_{45} = 0, s_{75} = 0$
34	b_{123}	s_{127}	$z_{34} = 0, b_{46} = 0, s_{76} = 0$
35	s_9	$-$	$z_{48} = 0, b_{60} = 0, s_{90} = 0$
36	s_{10}	s_0, s_7	$z_{49} = 0, b_{61} = 0, s_{91} = 0$
37	b_{124}	$-$	$z_{35} = 0, b_{47} = 0, s_{77} = 0$
38	b_{125}	s_1, s_8	$z_{36} = 0, b_{48} = 0, s_{78} = 0$
39	b_{126}	s_2	$z_{37} = 0, b_{49} = 0, s_{79} = 0$
40	b_{127}	s_3	$z_{38} = 0, b_{50} = 0, s_{80} = 0$
41	s_{11}	$-$	$z_{46} = 0, b_{58} = 0, s_{88} = 0$
42	s_{12}	$-$	$z_{47} = 0, b_{59} = 0, s_{89} = 0$
43	s_6	$-$	$z_{41} = 0, b_{53} = 0, s_{83} = 0$
44	b_0	s_4	$z_{39} = 0, b_{51} = 0, s_{81} = 0$
45	b_1	s_5	$z_{40} = 0, b_{52} = 0, s_{82} = 0$

Table 6: The deduced state bits of Grain-128 and Grain-128a