Near Collision Attack on the Grain v1 Stream Cipher [⋆]

Bin Zhang[*], Zhenqi Li[†], Dengguo Feng[†] and Dongdai Lin[*]

[*]State Key Laboratory of Information Security, IIE, Chinese Academy of Sciences,
Beijing, 100093, China.
[†]IOS, Chinese Academy of Sciences, Beijing, 100190, China.
{zhangbin, lizhenqi}@is.iscas.ac.cn

Abstract. Grain v1 is one of the 7 finalists selected in the final portfolio by the eSTREAM project. It has an elegant and compact structure, especially suitable for a constrained hardware environment. Though a number of potential weaknesses have been identified, no key recovery attack on the original design in the single key model has been found yet. In this paper, we propose a key recovery attack, called near collision attack, on Grain v1. The attack utilizes the compact NFSR-LFSR combined structure of Grain v1 and works even if all of the previous identified weaknesses have been sewed and if a perfect key/IV initialization algorithm is adopted. Our idea is to identify near collisions of the internal states at different time instants and restore the states accordingly. Combined with the BSW sampling and the non-uniform distribution of internal state differences for a fixed keystream difference, our attack has been verified on a reduced version of Grain v1 in experiments. An extrapolation of the results under some assumption indicates an attack on Grain v1 for any fixed IV in $2^{71.4}$ cipher ticks after the pre-computation of $2^{73.1}$ ticks, given $2^{62.8}$-bit memory and $2^{67.8}$ keystream bits, which is the best key recovery attack against Grain v1 so far. Hopefully, it provides some new insights on such compact stream ciphers.

Keywords. Stream ciphers, Cryptanalysis, Grain, Near collision.

1 Introduction

Grain v1, designed by Martin Hell, Thomas Johansson and Willi Meier [13], is a stream cipher for restricted hardware environments. It uses 80-bit key and 64-bit IV and consists of two combined registers, one NFSR and one LFSR, filtered together by a non-linear function. During the eSTREAM competition, Grain v1

[⋆] This work was supported by the National Grand Fundamental Research 973 Program of China(Grant No. 2013CB338002), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701), IIE's Research Project on Cryptography (Grant No. Y3Z0016102) and the programs of the National Natural Science Foundation of China (Grant No. 60833008, 60603018, 61173134, 91118006, 61272476).

has successfully withstood huge cryptanalytic efforts and in April 2008, it was selected into the final portfolio by the eSTREAM project, as it has pushed the state of the art of stream ciphers in terms of compact implementation [10].

Grain v1 has a compact structure with carefully chosen tap positions, feedback functions and output function. The feedback function of NFSR and the filter function are chosen in such a way that the correlation [3] and distinguishing attacks [16] on the former version, Grain v0, have been made impossible in time faster than exhaustive search. The companion cipher, Grain-128 [14], is designed in a similar way except that the feedback function is of low algebraic degree (a property not in Grain v1), which results in a dynamic cube attack of full initialization rounds [9, 8] and a new version, Grain-128a [15], with optional authentication. In [7], a slide property in the initialization phase was discovered, which can be used to reduce by half the cost of exhaustive key search for a fixed IV and to mount related-key chosen IV attacks [7, 17] against Grain v1 and Grain-128.

In this paper, we propose a new key recovery attack, called near collision attack, on Grain v1. The attack utilizes the compact NFSR-LFSR combined structure of Grain v1 and works even if all of the previous identified weaknesses have been sewed and if a perfect key/IV initialization algorithm is adopted, e.g., the slide property does not exist any more and there are a sufficiently large number of initialization rounds. It is observed that the NFSR and LFSR are of length exactly 80-bit (the same as the key length, with no redundance) and the LFSR updates independently in the keystream generation phase. Further, if the 160-bit internal states at two different time instants differ in only a small number of positions, the output keystreams they generate will be similar to each other. In fact, the keystream segment differences in this case can not take all the possible values, i.e., there are lots of impossible keystream segment differences and even for the possible differences, the distribution is heavily non-uniform. Some differences occur with very high probability, while others do not. This is due to the fact that for some keystream segment differences, there are many low weight internal state differences that can cause them. Based on the near match generalization of the birthday paradox, such near collisions of the internal states do exist given enough keystream and the problem is how to explicitly and efficiently identify them.

We develop an approach to detect such near collision internal states and the basic attack is called NCA-1.0[1]. Combined with BSW sampling, an enhanced attack, NCA-2.0, is proposed and it can reduce the attack complexity compared to NCA-1.0. We further improve it to NCA-3.0 by utilizing the heavily non-uniform distribution of the internal state differences for a fixed keystream difference. Then our attack has been launched and verified on a reduced version of Grain v1 with 32-bit LFSR and 32-bit NFSR. An extrapolation of the results under some reasonable assumption indicates an attack on Grain v1 for any fixed IV in $2^{71.4}$ cipher ticks after the pre-computation of $2^{73.1}$ ticks, given $2^{62.8}$-bit memory and $2^{67.8}$ keystream bits. This is the best key recovery attack against

[1] near collision attack version 1.0.

Grain v1 so far[2]. The results of all the NCA attacks are summarized in the following table. Our attack is just a starting point for further analysis of Grain-like stream ciphers and hopefully it provides some new insights on the design of such compact stream ciphers.

Table 1. The attack complexity

Attack model	Pre-computation time	Data	Memory	Time
NCA-1.0	$2^{95.7}$	$2^{45.8}$	$2^{78.6}$	$2^{85.9}$
NCA-2.0	$2^{83.4}$	2^{62}	$2^{65.9}$	$2^{76.1}$
NCA-3.0	$2^{73.1}$	$2^{67.8}$	$2^{62.8}$	$2^{71.4}$

This paper is structured as follows. Some notations and preliminaries are given in Section 2. Then, some key observations used in our attack and the description of Grain v1 are presented in Section 3. The general attack model and its complexity analysis are formalized in Section 4. The NCA-2.0 based on BSW sampling resistance is given in Section 5 and the NCA-3.0 based on the non-uniform distribution of keystream segment differences is presented in Section 6, respectively. The basis simulation results on the reduced version of Grain is provided in Section 7. Finally, we conclude in Section 8.

2 Notations & Preliminaries

In this section, we give a brief description of Grain[3] and propose some lemmas that we will use. The following notations are used throughout the paper.

- $w_H(\cdot)$: the Hamming weight function, output the number of 1s in the binary representation of the input argument.
- d : the maximum Hamming weight of the internal state difference.
- l : the length of the keystream segment, measured in bit.
- n : the length of the internal state, measured in bit.
- Δs : the internal state difference.
- $V(n,d)$: the total number of the internal state differences with $w_H(\Delta s) \leq d$.
- $Q(n,d,l)$: the total number of all the possible keystream segment differences, while traversing all the $V(n,d)$ internal state differences.
- $R(n,d,l)$: the average number of the internal state differences, corresponding to a fixed keystream segment difference.
- B_d : the set of the internal state differences with $\Delta s \in B_d$ and $w_H(\Delta s) \leq d$.
- $I_{\Delta s}$: the set of the difference position indexes of Δs. The difference position indexes range from 0 to 159, corresponding to $n_0, n_1, ..., n_{79}, l_0, l_1, ..., l_{79}$.
- P : the pre-computation time complexity.

[2] We give a rigorous analysis on the time complexity of the brute force attack on Grain v1 in Section 3.3 and find that the actual complexity is $2^{87.4}$ cipher ticks, which is higher than 2^{80} ticks.

[3] We use Grain to denote Grain v1 hereafter.

- T : the on-line time complexity.
- M : the memory requirement.
- D : the data complexity.
- Ω : the number of CPU clock cycles to generate one bit keystream in software.

It is easy to see that[4] $B_d = \{\Delta s \in \mathbb{F}_2^n | w_H(\Delta s) \leq d\} = \{\Delta s_1, \Delta s_2, ..., \Delta s_{V(n,d)}\}$ and $|B_d| = V(n,d) = \sum_{i=0}^{d} \binom{n}{i}$. The definition of d-near-collision for two binary strings is as follows.

Definition 1 *Two n-bit strings s, s' are d-near-collision, if $w_H(s \oplus s') \leq d$.*

Similar to the birthday paradox, which states that two random subsets of a space with 2^n elements are expected to intersect when the product of their sizes exceeds 2^n, we present the following lemma of d-near-collision.

Lemma 1 *Given two random subsets A, B of a space with 2^n elements, then there exists a pair (a, b) with $a \in A$ and $b \in B$ that is an d-near-collision if*

$$|A| \cdot |B| \geq \frac{2^n}{V(n,d)} \tag{1}$$

holds, where $|A|$ and $|B|$ are the size of A and B respectively.

Proof. Let $A = \{a_1, a_2, ..., a_{|A|}\}$ and $B = \{b_1, b_2, ..., b_{|B|}\}$. Each $a_i \in A$, $b_j \in B$ are uniformly random variables with values in $\mathbb{F}_2^n$. Consider the random variables $w_H(a_i \oplus b_j)$ and let ϕ be the characteristic function of the event $w_H(a_i \oplus b_j) \leq d$, that is,

$$\phi(w_H(a_i \oplus b_j) \leq d) = \begin{cases} 1 & \text{if } w_H(a_i \oplus b_j) \leq d \\ 0 & \text{otherwise.} \end{cases}$$

For $1 \leq i \leq |A|$, $1 \leq j \leq |B|$, we consider the number $N_{A,B}(d)$ of pairs (a_i, b_j) satisfying $w_H(a_i \oplus b_j) \leq d$ (the number of d-near-collisions): $N_{A,B}(d) = \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} \phi(w_H(a_i \oplus b_j) \leq d)$. The expected value of $N_{A,B}(d)$ of pairwise-independent random variables can be computed as $E(N_{A,B}(d)) = |A| \cdot |B| \cdot \frac{V(n,d)}{2^n}$. Therefore, if we choose the size of A and B satisfying equation (1), the expected number of d-near-collisions pairs is at least 1. □

If $d = 0$, then $V(n,d) = 1$ and Lemma 1 reduces to the common collision, otherwise the data required of finding a d-near-collision is much less than that of finding a complete collision. If $|A| \cdot |B| = 2^n / V(n,d)$, then the probability to find a d-near-collision is about 50%. If $|A| \cdot |B| = 3 \cdot 2^n / V(n,d)$, then the probability to find a d-near-collision is larger than 98%.

[4] $| \cdot |$ denotes the cardinality of a set.

2.1 Grain-v1

Grain-v1 is one of the 7 finalists selected in the final portfolio by the eSTREAM project. It is a bit-oriented stream cipher taking an 80-bit key and a 64-bit IV. The cipher consists of a pair of linked 80-bit shift registers, one is linear feedback shift register (LFSR) and another is non-linear feedback shift register (NFSR), denoted as $\{l_i, l_{i+1}, ..., l_{i+79}\}$ and $\{n_i, n_{i+1}, ..., n_{i+79}\}$ respectively. The update function of the LFSR is $l_{i+80} = l_{i+62} + l_{i+51} + l_{i+38} + l_{i+23} + l_{i+13} + l_i$ and the update function of the NFSR is

$$\begin{aligned} n_{i+80} = l_i &+ n_{i+62} + n_{i+60} + n_{i+52} + n_{i+45} + n_{i+37} + n_{i+33} + n_{i+28} + n_{i+21} \\ &+ n_{i+14} + n_{i+9} + n_i + n_{i+63}n_{i+60} + n_{i+37}n_{i+33} + n_{i+15}n_{i+9} \\ &+ n_{i+60}n_{i+52}n_{i+45} + n_{i+33}n_{i+28}n_{i+21} + n_{i+63}n_{i+45}n_{i+28}n_{i+9} \\ &+ n_{i+60}n_{i+52}n_{i+37}n_{i+33} + n_{i+63}n_{i+60}n_{i+21}n_{i+15} \\ &+ n_{i+63}n_{i+60}n_{i+52}n_{i+45}n_{i+37} + n_{i+33}n_{i+28}n_{i+21}n_{i+15}n_{i+9} \\ &+ n_{i+52}n_{i+45}n_{i+37}n_{i+33}n_{i+28}n_{i+21}. \end{aligned}$$

During keystream generation phase, shown in Fig.1, the output bit z_i is filtered by a non-linear function $h(x)$, which is balanced and correlation immune of the first order, defined as follows.

$$\begin{aligned} h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 \\ + x_1x_2x_4 + x_2x_3x_4, \end{aligned}$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to the tap positions l_{i+3}, l_{i+25}, l_{i+46}, l_{i+64} and n_{i+63} respectively. The output function is taken as $z_i = \sum_{k\in\mathcal{A}} n_{i+k} + h(l_{i+3}, l_{i+25}, l_{i+46}, l_{i+64}, n_{i+63})$, where $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$.

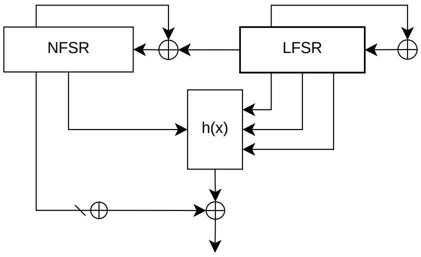

Fig. 1. Keystream generation mode

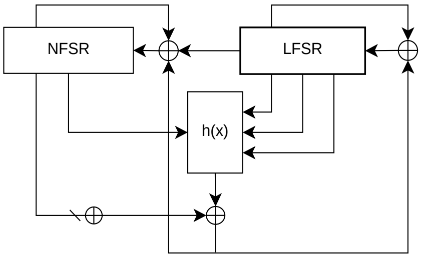

Fig. 2. Initialization mode

Let the bits of the key be k_i, $0 \leq i \leq 79$ and the bits of the IV be IV_i, $0 \leq i \leq 63$. In the initialization phase, shown in Fig.2, first load the NFSR with the key bits, $n_i = k_i$, $0 \leq i \leq 79$, then load the first 64 bits of the LFSR with the IV, $s_i = IV_i$, $0 \leq i \leq 63$. The remaining bits of the LFSR are filled with ones, i.e., $s_i = 1$, $64 \leq i \leq 79$. Then the cipher is clocked 160 times without producing any keystream bit. Instead the output function is fed back and xored with the input, both to the LFSR and to the NFSR.

3 Some Key Observations

We start with some key observations used in our attacks. More technical descriptions of the various steps will be provided in the next sections.

3.1 State Recovery with Known State Difference

According to NFSR-LFSR combined structure of Grain, the internal states at two different time instants can be recovered in a reasonable time if we know the state difference. More precisely, during the keystream generation phase, we denote the LFSR state as $L^{t_1} = (l_0^{t_1}, l_1^{t_1}, ..., l_{79}^{t_1})$ at time t_1 and $L^{t_2} = (l_0^{t_2}, l_1^{t_2}, ..., l_{79}^{t_2})$ at time t_2 $(0 \le t_1 < t_2)$.

Suppose that we know the difference $\Delta L = (l_0^{t_1} \oplus l_0^{t_2}, ..., l_{79}^{t_1} \oplus l_{79}^{t_2}) = (\Delta l_0, \Delta l_1, ..., \Delta l_{79})$ with the time interval $\Delta t = t_2 - t_1$. Since the LFSR is clocked independently (never affected by the NFSR or the keystream bits) in the keystream generation phase of Grain, each $l_i^{t_2}$ in L^{t_2} can thus be linearly expressed by the variables in L^{t_1}:

$$\begin{cases} l_0^{t_2} = c_0^0 l_0^{t_1} + c_1^0 l_1^{t_1} + ... + c_{79}^0 l_{79}^{t_1} \\ l_1^{t_2} = c_0^1 l_0^{t_1} + c_1^1 l_1^{t_1} + ... + c_{79}^1 l_{79}^{t_1} \\ \quad\vdots \\ l_{79}^{t_2} = c_0^{79} l_0^{t_1} + c_1^{79} l_1^{t_1} + ... + c_{79}^{79} l_{79}^{t_1}, \end{cases}$$

where c_i^j, $0 \le i, j \le 79$ can be pre-computed according to Δt and the update function of the LFSR, not depending on t_1 and t_2. Combined with ΔL, we can easily derive the following linear system.

$$\begin{cases} \Delta l_0 = l_0^{t_2} \oplus l_0^{t_1} = (c_0^0 + 1) l_0^{t_1} + c_1^0 l_1^{t_1} + ... + c_{79}^0 l_{79}^{t_1} \\ \Delta l_1 = l_1^{t_2} \oplus l_1^{t_1} = c_0^1 l_0^{t_1} + (c_1^1 + 1) l_1^{t_1} + ... + c_{79}^1 l_{79}^{t_1} \\ \quad\vdots \\ \Delta l_{79} = l_{79}^{t_2} \oplus l_{79}^{t_1} = c_0^{79} l_0^{t_1} + c_1^{79} l_1^{t_1} + ... + (c_{79}^{79} + 1) l_{79}^{t_1}. \end{cases} \tag{2}$$

Variables in L^{t_1} can then be determined by solving (2), which means that we can obtain the internal state of LFSR at t_1. The time complexity of this step is upper bounded by $T_L \approx 2^{18.9}$ basic operations [19]. Suppose one basic operation needs one CPU cycle, then according to appendix C that one tick of Grain needs $2^{10.4}$ CPU clock cycles, hence $T_L = 2^{18.9}/2^{10.4} = 2^{8.5}$ cipher ticks in software.

The next step is to recover the NFSR state at t_1. This process can be found in appendix A. The time complexity of this phase is upper bounded by $T_N = 2^{20.3}$ cipher ticks. The key can then be easily recovered by the running internal state at t_1 backward. To sum up, given the internal state difference with the time interval, the time complexity to retrieve the internal state is $T_K = T_L + T_N = 2^{20.3}$ cipher ticks.

3.2 The Distribution of the Keystream Segment Differences

The second observation is that the distribution of keystream segment differences (KSDs) is heavily biased, given a specific internal state differential (IS-D). For instance, we choose $d = 4$, $l = 16$, $I_{\Delta s_1} = \{9, 31, 39, 69\}$[5], $I_{\Delta s_2} = \{99, 121, 134, 149\}$[6], $I_{\Delta s_3} = \{29, 64, 101, 147\}$[7] and $I_{\Delta s_4} = \{20, 26, 53, 141\}$[8]. Then we randomly choose 10^4 internal states, calculate the companion states by adding Δs_1, Δs_2, Δs_3, Δs_4 and generate the corresponding KSDs for Δs_1, Δs_2, Δs_3 and Δs_4 in Table 2.

Table 2. The distribution of KSDs

ISD	KSD	Proportion	ISD	KSD	Proportion
Δs_1	0xa120	49.4%	Δs_4	0x0000	52.0%
	0xe120	50.6%		0x0080	48.0%
Δs_2	0x0000	12.9%	Δs_3	0x0001	13.2%
	0x0001	13.8%		0x0201	12.1%
	0x2000	38.3%		0x0801	37.2%
	0x2001	35.1%		0x0a01	37.5%

From Table 2, there are only 2 values of KSD, each occurred with proportion close to 1/2 for Δs_1 and Δs_4 respectively. There are 4 values of KSD with varying proportions for Δs_2 and Δs_3 respectively, e.g, if the ISD is Δs_2, then 38.3% KSDs are `0x2000`. We also test other ISDs with different d and l, the results are similar to Table 2.

In many cases, there exists some impossible KSDs when d and l are fixed. To illustrate this, given $1 \leq d \leq 4, l \in \{8, 16, 24, 32\}$, we enumerated each $\Delta s \in B_d$ and count the number of $Q(n, d, l)$ for all the possible ISDs in B_d. The results show that there exists some impossible differences for most of (d, l) pairs. Thus the value of $Q(n, d, l)$ can be estimated as $2^{l-\gamma}$ where 2^{γ} is the number of impossible differences, e.g, for $(d, l) = (3, 24)$, $\gamma = 4.7$. Even for the possible differences, the distribution is non-uniform, which causes some entropy leakage as well. These features can be further utilized to enhance our attack.

3.3 Complexity of the Brute Force Attack

The third observation is that the complexity of brute force attack is higher than 2^{80} ticks and such an attack can only be mounted for each fixed IV, while our attack can be applied to the scenario with arbitrary IVs.

As a baseline, we analyzed the time complexity of the brute force attack on Grain. Given a known fixed IV and a 80-bit keystream segment w, generated by (K,IV) pair, the goal is to recover K using the exhaustive search strategy.

[5] $\Delta s_1 = $ 0x0002008080000000200000000000000000000000

[6] $\Delta s_2 = $ 0x0000000000000000000000080000000240002000

[7] $\Delta s_3 = $ 0x0000002000000000010000002000000000000800

[8] $\Delta s_4 = $ 0x0000100400002000000000000000000000200000

For each enumerated k_i, $1 \leq i \leq 2^{80}-1$, the attacker first needs to proceed the initialization phase which needs 160 ticks. During the keystream generation phase, once a keystream bit is generated, the attacker compares it to the corresponding bit in w. If they are equal, the attacker continue to generate the next keystream bit and do the comparison. If not, the attacker search another key and repeat the previous steps. If each keystream bit is treated as a random independent variable, then for each k_i, the probability that the attacker need to generate l ($1 \leq l \leq 80$) bits keystream is 1 for $l = 1$ and $2^{-(l-1)}$ for $l > 1$, which means that the previous $l-1$ bits are equal to the counter bits in w. Let N_w be the expected number of bits needed to generate for each enumerated key, which is $N_w = \sum_{l=1}^{80} l \cdot P_l = \sum_{l=1}^{80} l \cdot 2^{-(l-1)} \approx 4$. Then, the total time complexity is $(2^{80}-1) \cdot (160+4) \approx 2^{87.4}$ cipher ticks.

4 The General Attack Model

In this section, we will give a general description of our attack model. From Section 3.1, it is easy to recover the internal state by utilizing the known ISD and the time interval, thus the main concern is to retrieve the ISD derived from the two d-near-collision internal states. Our attack consists of two phases, i.e., an off-line stage only performed once followed by an on-line stage.

4.1 Off-line Stage

In the off-line stage, some well structured differential tables are pre-computed. Given l and d, we enumerated the $V(n,d)$ different ISDs in B_d and generate their corresponding KSDs with proportions. In total, $Q(n,d,l)$ different tables will be constructed and indexed with KSD. The ISDs with the proportions, which will generate the indexed KSD, will be stored in each KSD table. For example, following Table 2 in Section 3.2, $\varDelta s_2$ together with the proportion 12.9% will be stored in one line of table-0x0000 and $\varDelta s_4$ together with the proportion 52.0% will also be saved in another line of table-0x0000. The table structure can be illustrated as follows.

$$\text{table-}\texttt{0x0000}\begin{cases}\varDelta s_4 & 52.0\% \\ \varDelta s_2 & 12.9\% \\ \quad\vdots & \end{cases} \quad \text{table-}\texttt{0x0001}\begin{cases}\varDelta s_2 & 13.8\% \\ \varDelta s_3 & 13.2\% \\ \quad\vdots & \end{cases} \qquad \ldots$$

$$\text{table-}\texttt{0x0080}\begin{cases}\varDelta s_4 & 48.0\% \\ \quad\vdots & \end{cases} \qquad \ldots$$

The total number of tables is $Q(n,d,l)$ and the average number of rows in each table is $R(n,d,l)$. Due to the non-uniform distribution of the KSDs for a fixed ISD, we only consider at most 100 KSDs whose proportions are the first 100 largest among all the KSDs, then each ISD will be stored in at most 100 different KSD tables. Hence $R(n,d,l)$ is upper bounded by $100 \cdot V(n,d)/Q(n,d,l)$. The memory requirement is thus $M_1 = Q(n,d,l) \cdot R(n,d,l) = V(n,d) \cdot 2^{6.6}$ entries, each containing $n+\delta$ bits where δ is used to store the proportion and $\delta = 7$ bits[9]. We sort each table with respect to the values of those proportions so that the ISD with the maximum proportion will appear in the first row. All the tables will be sorted with respect to their KSD indexes. Let N be the sampling number of the random internal states when determining the projection from ISD to KSD, then we have $P = 2 \cdot N \cdot V(n,d) \cdot l$ cipher ticks.

4.2 On-line Stage

Now we discuss how to obtain the ISD by utilizing the pre-computed tables and the truncated keystreams. Let the length of the keystream segment be $\hat{l} = l + \beta$, where β is the length of the keystream suffix used for verification. The on-line stage contains the following steps:

Step 1. We randomly collect two keystream segments sets A and B, each element $a_i \in A$, $b_j \in B$ of which is $\hat{l}$-bit. Let $a_i^{[l]}$ and $b_j^{[l]}$ denote the first l bits of the keystream segments and the time instants for each $a_i^{[l]}$ and $b_j^{[l]}$ are also recorded. Let s_i^A and s_j^B be the internal states corresponding to $a_i^{[l]}$ and $b_j^{[l]}$ respectively, from Lemma 1, in order to assure that there exists at least one pair (s_i^A, s_j^B) so that $s_i^A \oplus s_j^B \in B_d$, it is required that $|A| \cdot |B| \geq 2^n / V(n,d)$.

Step 2. We sort A and B with respect to the value of the first l bits and divide A, B into m different groups $G_1^A, G_2^A, ..., G_m^A$ and $G_1^B, G_2^B, ..., G_m^B$ respectively. The keysream segments in A (B) with the same $a_i^{[l]}$ ($b_j^{[l]}$) will be put into the same group with the index $a_i^{[l]}$ ($b_j^{[l]}$). The size of each group can be estimated as $|G_i^A| = |A|/2^l$, $|G_i^B| = |B|/2^l$, $1 \leq i \leq m$. Note that if $|A| \geq 2^l$, then $m = 2^l$. If $|A| < 2^l$, then there may be some empty groups and we define $m = |A|$. The sorting time is $T_1 = |A| \cdot \log|A| + |B| \cdot \log|B|$ comparisons.

Step 3. Now we need to identify the candidate (s_i^A, s_j^B) pairs that is d-near-collision. Denote the $Q(n,d,l)$ different KSDs in the off-line stage by $W = \{w_1, w_2, ..., w_{Q(n,d,l)}\}$ where each w_k, $1 \leq k \leq Q(n,d,l)$ is of l-bit lengths. For each $w_k \in W$, we need to find all the pairs $(a_i^{[l]}, b_j^{[l]})$ satisfying $a_i^{[l]} \oplus b_j^{[l]} = w_k$. There are two strategies to achieve this goal:

Strategy I. For each $w_k \in W$, we xor it to each group index of G_i^A in A and get A^*. If there is one group $G_i^{A^*}$ with the group index same as the index of another group G_i^B in B, then we get $a_p^{[l]} \oplus b_q^{[l]} = w_k$ for any $1 \leq p \leq |G_i^{A^*}|$ and $1 \leq q \leq |G_i^B|$, for if we xor w_k to each $a_p^{[l]} \in G_j^{A^*}$ and get $a_p^{[l]^*} = a_p^{[l]} \oplus w_k$, then any $(a_p^{[l]^*}, b_q^{[l]})$ pair is a match

[9] We use a 7-bit string to store the percentage number, e.g., for 67%, we only store binary representation of 67 (67 < 128).

satisfying $a_p^{[l]^*} = a_p^{[l]} \oplus w_k = b_q^{[l]}$, see Fig.3. The time complexity is $T_2^I = Q(n,d,l) \cdot m \cdot \log m$ comparisons.

Strategy II. For each G_i^A in A, we xor its index to each index of G_j^B in B (get B^*) and search a match in the sorted $W = \{w_1, w_2, ..., w_{Q(n,d,l)}\}$. If a match $w_k \in W$ is found, then we have $(a_p^{[l]}, b_q^{[l]})$ satisfying $a_p^{[l]} \oplus b_q^{[l]} = w_k$ with $a_p^{[l]} \in G_i^A, b_q^{[l]} \in G_j^B$, see Fig.4. The time complexity is $T_2^{II} = m \cdot m \cdot \log Q(n,d,l)$ comparisons.

The time complexity of Step 3 is $T_2 = min\{T_2^I, T_2^{II}\}$.

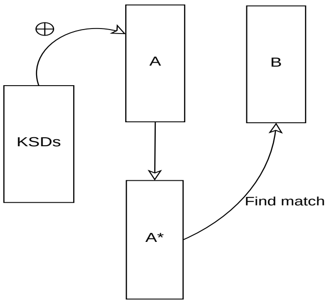

Fig. 3. Strategy I

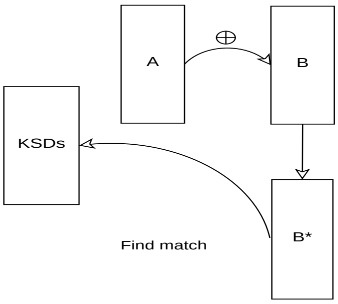

Fig. 4. Strategy II

Step 4. From the previous steps, if we choose Strategy I in step 3, then for each w_i, there are at most $C_{w_i} = 2^l \cdot |G_j^A| \cdot |G_j^B| = |A| \cdot |B|/2^l$ matches, thus the total number of matches is at most $C = C_{w_i} \cdot Q(n,d,l) = Q(n,d,l) \cdot |A| \cdot |B|/2^l = |A| \cdot |B|$ (Strategy II get at most C=$|A| \cdot |B|$ matches), among which there are many pseudo-collisions [10]. This step is to filter out these pseudo-collisions and find the real one. Precisely, for each pair $(a_p^{[l]}, b_q^{[l]})$, we look for the table with the index $a_p^{[l]} \oplus b_q^{[l]} = w_i$, read the corresponding ISD list $B_d^{w_i}$ and for each $\Delta s \in B_d^{w_i}$, we can derive the internal state by using the method described in Section 3.1. Finally, we can easily verify the correctness of the state by running the cipher forward and compare the generated keystream with those β bits collected in A and B. The average size of the table is $R(n,d,l) = 100 \cdot V(n,d)/Q(n,d,l)$. The time complexity is $T_3 = C \cdot R(n,d,l) \cdot T_K = |A| \cdot |B| \cdot V(n,d) \cdot 2^{6.6} \cdot T_K/2^l$ cipher ticks[11] (for strategy II, $T_3 = |A| \cdot |B| \cdot V(n,d) \cdot 2^{6.6} \cdot T_K/Q(n,d,l)$).

The total online complexity of the on-line state is thus $T = T_1 + T_2 + T_3$ and the memory complexity is $M_2 = |A| + |B|$ entries, each containing $\hat{l}$ bits.

[10] Pseudo-collision indicates the case that $a_i^{[l]} \oplus b_j^{[l]}$ matches to a KSD, but the internal state recovered from the ISD found in the table indexed with the KSD is not correct.

[11] we ignore the cost for verifying the correctness of the state, since it does not make significant change in T_3

4.3 Complexity Analysis

The unit of the time complexity is one tick of Grain. It is obvious that $Q(n,d,l)$ is upper bounded by 2^l and $R(n,d,l)$ is upper bounded by $100 \cdot V(n,d)/Q(n,d,l)$. The pre-computation time is $P = 2 \cdot N \cdot V(n,d) \cdot l$. The data complexity is $D = |A| + |B|$ $\hat{l}$-bit keystream segments and the memory requirement is $M = M_1 + M_2 = V(n,d) \cdot 2^{6.6} + |A| + |B|$ entries.

The time complexity in Step 2 is $T_1 = |A| \cdot \log |A| + |B| \cdot \log |B|$ comparisons. Suppose each comparison is done in one CPU cycle. Since one tick of Grain needs $\Omega = 2^{10.4}$ CPU clock cycles (see Appendix C for details), $T_1 = (|A| \cdot \log |A| + |B| \cdot \log |B|)/\Omega$ cipher ticks. Similarly, the time complexity in Step 3 is $T_2 = min\{Q(n,d,l) \cdot m \cdot \log m/\Omega,\ m^2 \cdot \log Q(n,d,l)/\Omega\}$ ticks.

For Grain $n = 160$ and $d = 16$, then $V(n,d) \approx 2^{72}$. If we choose $|A| = |B| = \sqrt{3} * 2^{n/2}/\sqrt{V(n,d)} \approx 2^{44.8}$ and the sample size $N = 2^{16}$, then the data complexity is thus $D = |A| + |B| = 2^{45.8}$, the memory requirement is $M = 2^{78.6}$ 167-bit entries and the time complexity of pre-computation is $P = 2^{89} \cdot l$. The complexities varying with l are shown in Table 3. From Table 3, our attack has a rather uniform complexity tradeoff. Besides, from $|A| \cdot |B| = 3 \cdot 2^n/V(n,d)$, the estimated success probability is about 98%. We name this basic attack as NCA-1.0. However, the pre-computation complexity $P = 2^{95.7}$ exceeds the brute force attack complexity of $2^{87.4}$. In the following sections, we will propose some enhanced attacks.

Table 3. The attack complexity with various l

l	P	T_1	T_2	T_3	T
102	$2^{95.7}$	$2^{40.9}$	$2^{85.8}$	$2^{86.4}$	$2^{86.4}$
104	$2^{95.7}$	$2^{40.9}$	$2^{85.9}$	$2^{84.4}$	$2^{85.9}$
106	$2^{95.7}$	$2^{40.9}$	$2^{85.9}$	$2^{72.4}$	$2^{85.9}$

$n = 160,\ d = 16,\ D = 2^{45.8},\ M = 2^{78.6}$
Strategy II is chosen in Step 3.

5 Improvement I

The first improvement is designated by combining the sampling resistance property of Grain with NCA-1.0. Biryukov and Shamir proposed the concept of sampling resistance in [5], named BSW-sampling. It can be used to obtain larger choices of tradeoff parameters on the Biryukov-Shamir tradeoff curve.

5.1 Sampling Resistance of Grain

The main idea is to find an efficient way to generate and enumerate special states, from which some subsequent generated keystream bits have a fixed pattern (e.g., a string of zeros). If the length of the fixed pattern is k, then the sampling resistance of the cipher is $R = 2^{-k}$. In [6], it was proved that the sampling resistance of Grain is 2^{-21} by guess and determine strategy. We use a simple method to derive the sampling resistance of Grain which has lower complexity than the guess and determine strategy in [6]. Here comes our Lemma 2, proved in Appendix B.

Lemma 2 *Given the value of* 139 *particular state bits of Grain and the first* 21 *keystream bits produced from that state, another* 21 *internal state bits can be deduced directly.*

The 139 particular state bits contained 60 bits of the NFSR state and 79 bits of the LFSR state. From Lemma 2, the sampling resistance of Grain is $R = 2^{-21}$. Thus, we define a restricted one-way function $\tau : \{0,1\}^{139} \to \{0,1\}^{139}$ by choosing a prefix of 0^{21}.

1. For each 139-bit input value x, the remaining 21-bit internal state can be determined by Lemma 2 and the prefix of 0^{21}.
2. Run the cipher forward for 160 ticks, generate an 160-bit segment $0^{21}||y$ and output y.

Now, the searching space is reduced to a special subset of the internal states.

5.2 Complexity Analysis Based on Sampling Resistance

Now, the goal is to recover the $n^* = 139$ bits ISD which contains 60 NFSR state bits and 79 LFSR state bits, instead of the $n = 160$ bits ISD. Note that if we observe l-bit keystream from the output y of τ, we need additional 42 ticks, 21 ticks to compute the remaining 21-bit internal state and 21 ticks to generate the prefix keystream. The pre-computation time complexity is thus $P^* = 2 \cdot N \cdot V(n^*, d) \cdot (l + 42)$ ticks. In the on-line stage, we need to collect those keystream segments with the prefix pattern 0^{21}, which can ensure that the corresponding internal state existed in the reduced searching space. Hence, the data complexity is $D = (|A| + |B|) \cdot 2^{21}$. Given $d = 13$, then $V(n^*, d) \approx 2^{59.3}$ and $|A| = |B| = \sqrt{3} \cdot 2^{n^*/2} / \sqrt{V(n^*, d)} \approx 2^{40}$. Thus, the data complexity is $D = (|A| + |B|) \cdot 2^{21} = 2^{62}$, the memory complexity is $M = V(n^*, d) \cdot 2^{6.6} + |A| + |B| = 2^{65.9}$ entries, each containing $n^* + \delta$ bits instead of $n + \delta$ bits. The pre-computation

Table 4. The attack complexities with various l based on sampling resistance

l	P^*	T_1	T_2	T_3	T
92	$2^{83.4}$	$2^{35.9}$	$2^{76.1}$	$2^{75.4}$	$2^{76.1}$
94	$2^{83.4}$	$2^{35.9}$	$2^{76.2}$	$2^{73.4}$	$2^{76.2}$
96	$2^{83.4}$	$2^{35.9}$	$2^{76.2}$	$2^{71.4}$	$2^{76.2}$

$n^* = 139,\ d = 13,\ D = 2^{62},\ M = 2^{65.9}$.
Strategy II is chosen in Step 3.

time is $P^* = 2 \cdot N \cdot V(n^*, d) \cdot (l + 42) = 2^{76.3} \cdot (l + 42)$. The time complexities with various l are summarized in Table 4. From Table 4, compared to NCA-1.0, our improved attack reduces P by a factor of $2^{12.3}$ and it saves 10-bit storage for each entry in A and B. All the complexities are under the brute force attack complexity of $2^{87.4}$. We name this combined attack as NCA-2.0.

6 Improvement II

The second improvement is based on NCA-2.0 by utilizing the non-uniform distribution of KSDs among all the tables.

6.1 Special Tables

As we have observed in Section 3.2, the distribution of the table size for each (d, l) are non-uniform. For $(d, l) = (4, 16)$, there are altogether $Q(160, 4, 16) = 2^{15.5}$ tables among which some tables like table-`0x0000`, table-`0x0008`, table-`0x0004` contains 10 times more rows than those like table-`0x0012` and table-`0x0048`. Table-`0x0000` contains the most rows among all the tables. Furthermore, most tables like table-`0xfe00`, table-`0xfd68` and table-`0xfad1` only contain a single row. Those tables with low Hamming weight indexes contains most of the ISD. The distributions among other (d, l) pairs are similar.

For each (d, l) pair that $1 \le d \le 4, l \in \{8, 16, 24, 32\}$, we found that the tables with low Hamming weight indexes satisfying $w_H(\text{KSD}) \le 3$ contain about 80% of all the $V(n, d)$ different ISDs. We call these tables special tables. In general, we make the following assumption which is verified in random experiments.

Assumption 1 *On average, the special tables can cover* 50% *of all the* $V(n^*, d)$ *different ISDs, when* d *and* l *becomes larger.*

The assumption indicates that in the off-line stage, we only need to construct those special tables.

6.2 Complexity Analysis Based on Special Tables

All the complexities remain unchanged except $T_2 = min\{l^3 \cdot m \cdot \log m,\ m^2 \cdot \log l^3\}$. In NCA-2.0, $n^* = 139$, then given $d = 10$, $V(n^*, d) \approx 2^{49}$. If we choose $|A| = |B| = \sqrt{3} \cdot 2^{n^*/2} / \sqrt{V(n^*, d)} \approx 2^{45.8}$ and the sample size $N = 2^{16}$, then the pre-computation time complexity is $P^* = 2^{66} \cdot (l + 42)$ and the data complexity is $D = (|A| + |B|) \cdot 2^{21} = 2^{67.8}$ entries[12], each containing $\hat{l}$ bits. The memory complexity is $M = V(n, d) \cdot 2^{6.6} + |A| + |B| = 2^{55.6}$ entries, each containing $n^* + \delta = 146$ bits. The time complexities with various l are summarized in Table 5. From it, we can obtain an attack of $T = 2^{71.4}$, $M = 2^{62.8}$ and $D = 2^{67.8}$ with the pre-computation complexity $P = 2^{73.1}$. We name this enhanced attack as NCA-3.0.

[12] In the on-line stage, the data can be collected in an overlapping way, thus to get $2^{67.8}$ keysream segments, each containing $\hat{l}$ bits, we only need $\hat{l} + 2^{67.8} - 1 \approx 2^{67.8}$ keystream bits

Table 5. The attack complexity on Grain with various l based on special tables

l	P^*	T_1	T_2	T_3	T
92	$2^{73.1}$	$2^{41.9}$	$2^{60.5}$	$2^{75.4}$	$2^{75.4}$
94	$2^{73.1}$	$2^{41.9}$	$2^{60.6}$	$2^{73.4}$	$2^{73.4}$
96	$2^{73.1}$	$2^{41.9}$	$2^{60.7}$	$2^{71.4}$	$2^{71.4}$

$n^* = 139,\ d = 10,\ M = 2^{62.8}$ bits, $D = 2^{67.8}$.
Strategy I is chosen in Step 3.

7 Simulations and Results

In this section, we validate our attacks by simulating a reduced version of Grain v1 stream cipher. We first give a brief description of the reduced version and then verify the Assumption 1 under various parameters. Finally, we apply NCA-2.0 and NCA-3.0 to the reduced cipher.

7.1 Reduced Version

The reduced version of Grain v1 cipher consists of an LFSR of 32 bits and an NFSR of 32 bits. $f'(x) = 1+x^2+x^7+x^{16}+x^{32}$ is a primitive polynomial of degree 32. The update function of LFSR is defined as $l'_{i+32} = l'_{i+30} + l'_{i+25} + l'_{i+16} + l'_i$. The feedback polynomial of the NFSR, $g'(x)$ is defined as

$$g'(x) = 1 + x^7 + x^9 + x^{17} + x^{24} + x^{32} + x^7x^9 + x^{17}x^{24} \\ + x^7x^9x^{17} + x^9x^{17}x^{24} + x^7x^9x^{17}x^{24},$$

Which is a balanced function of degree 4. Similar to Grain, the update function of NFSR with the bit l'_i masked to the input is

$$n'_{i+32} = l'_i + n'_{i+25} + n'_{i+23} + n'_{i+15} + n'_{i+8} + n'_i + n'_{i+25}n'_{i+23} + n'_{i+15}n'_{i+8} \\ + n'_{i+25}n'_{i+23}n'_{i+15} + n'_{i+23}n'_{i+15}n'_{i+8} + n'_{i+25}n'_{i+23}n'_{i+15}n'_{i+8}.$$

We use the same non-linear filter function $h(x)$ as in Grain v1 and take the output function as $z'_i = \sum_{k\in\mathcal{A}'} n'_{i+k} + h(l'_{i+3}, l'_{i+11}, l'_{i+21}, l'_{i+25}, n'_{i+24})$, where $\mathcal{A} = \{1, 4, 10, 21\}$. The key initialization is similar to Grain-v1. First, load the NFSR with the 32-bit key, then load the fisrt 24 bits of LFSR with the 24-bit IV. The remaining bits of the LFSR are filled with ones. Then the cipher is clocked 64 times without producing any keystream. Instead the output function is feedback and xored with the input, both to the LFSR and to the NFSR. The actual complexity of the brute force attack on the reduced version is $2^{38.1}$ cipher ticks. By using the same strategy in Appendix B, given the value of 53 particular state bits of the reduced Grain (including 32 bits LFSR and 21 bits NFSR) and the first 11 keystream bits produced from that state, another 11 internal state bits can be deduced directly. Then the sampling resistance is $R' = 2^{-11}$.

7.2 Verification of Assumption 1

Recall that the special tables are those with low Hamming weight indexes satisfying $w_H(\text{KSD}) \leq 3$. We first verify Assumption 1 in a random experiment. More precisely, we randomly chose 10^4 ISDs in B_d and generate their corresponding KSDs with the proportions. For each ISD, N random internal states were generated to determine the projection from ISD to KSD. Only those KSDs satisfying $w_H(\text{KSD}) \leq 3$ will be recorded and their corresponding ISDs will be stored in a text file named with KSD. Similar to the process of the off-line stage, we only consider at most η KSDs whose proportions are the first η largest among all the KSDs. Finally, we count the number of different ISDs in these special tables. For the reduced version of Grain, the length of the internal state is $n = 53$ and we set $d = 4$, $N = 2^{12}$. thus we randomly chose 10^4 ISDs, each of which has Hamming weight at most 4. The number of different ISDs in the special tables under various l and η are summarized in the following Table. From it, we can see that when $\eta = 50$ and $l = 32$, the special tables can cover more than 90% of all the ISDs, which corroborate our theoretical assumption very well. Under this configuration, we will apply NCA-2.0 and NCA-3.0 to the reduced version in the following section.

Table 6. Verification of Assumption 1

η	l	No. of ISDs	Proportion
50	24	9842	98.4%
1000	24	9851	98.5%
50	32	9202	92.0%
1000	32	9153	91.5%

$n = 53,\ d = 4.$

7.3 Simulations

In the off-line stage, we set $\eta = 50$, $N = 2^{12}$ and $d = 4$. The theoretical complexity with various l are given in the following table. The contents of a table will be stored in a text file named with the corresponding KSD. The pre-computation time under various keystream length l are summarized in the follow table. From

Table 7. Theoretical complexity on reduced version of Grain

Attack	l	P	D	M	T
NCA-2.0	24	$2^{36.3}$	$2^{29.2}$	$2^{23.9}$	$2^{36.2}$
NCA-3.0	24	$2^{36.3}$	$2^{29.2}$	$2^{23.9}$	$2^{36.2}$
NCA-2.0	32	$2^{36.7}$	$2^{29.2}$	$2^{23.9}$	$2^{31.4}$
NCA-3.0	32	$2^{36.7}$	$2^{29.2}$	$2^{23.9}$	$2^{28.2}$

$\eta = 50,\ N = 2^{12},\ d = 4.$

Table 8. Pre-computation time of NCA-2.0 & NCA-3.0

Attack	l	Time	Memory	No. of tables
NCA-2.0	24	9 hours, 50 mins	643 MB	8192
NCA-3.0	24	6 hours, 35 mins	216 MB	378
NCA-2.0	32	27 hours,41 mins	4.45 GB	2097152
NCA-3.0	32	6 hours, 37 mins	11.6 MB	1562

$\eta = 50,\ N = 2^{12},\ d = 4.$

Table 8, the table construction of NCA-2.0 takes more time than that of NCA-3.0, since NCA-3.0 only need to construct those special tables and the number of text files is much less than that of NCA-2.0, which indicates a lower cost of table look-up. In the on-line stage, we collected $|A| = |B| = \sqrt{3} \cdot 2^{n^*/2}/\sqrt{V(n^*, d)} \approx 2^{18}$ keystream segments. We apply NCA-2.0 and NCA-3.0 to the reduced version of Grain respectively for 140 randomly generated (K,IV) pairs. The average time for each attack and the experimental success probability are summarized in Table 9. For both NCA-2.0 and NCA-3.0, the experimental time is based on running an non-optimized C++ program on a 1.83 GHz CPU with 2GB RAM and 1TB harddisk. The success probability is given in the last column of Table 10. It is the proportion of the number of the correct internal state difference stored in the KSD tables.

Table 9. The simulation results on reduced version of Grain

Attack	l	Average Attack Time[a]	Success Probability
NCA-2.0	24	1 hours, 53 mins	9%
NCA-3.0	24	1 hours, 31 mins	7%
NCA-2.0	32	2 hours, 12 mins	6%
NCA-3.0	32	41 mins	4%

[a] This is the average time for each on-line attack.

We also conducted an experiment to analyze all those randomly collected keystream segments in the on-line stage of NCA-3.0. We attempt to find all those keystream segment pairs satisfying $w_H(\text{KSD}) \leq 3$ and $w_H(\text{ISD}) \leq d$, but get no results. We then repeat the experiment by increasing the maximum Hamming weight of the special indexes from 3 to 5, we finally get an average success rate of 10%, i.e., 1 qualified keystream segment pair out of 10 simulations. These experimental results suggest that we can not ensure a very stable success probability of NCA-3.0 on the full version currently. It need to be refined further and we indeed get some improvements by reducing the complexity of recovering the NFSR given the LFSR and the state difference by a factor of about 2^9. We will provide the details in the upcoming papers.

From Table 9, we can also see that the experimental success probability of NCA-2.0 is lower than estimated in theory. The reason is that we choose a restricted value of η and N. These two parameters directly influence the size and the number of the pre-computed tables, hence affect the success probability. How to theoretically derive the relationship between the success probability and these two parameters is our future work.

8 Further Explanations and Discussions

To link the reduced version results with the full version analysis, the following assumption is used in our analysis.

Assumption 2 *The attack parameters in the full version analysis are chosen based on a linear extrapolation according to the state length ratio.*

Precisely, the sampling size of $N = 2^{16}$ in the full version of our attack is chosen as follows. For the reduced version, we have a non-negligible success probability when $N = 2^{12}$. The proportion of the state length for the full version and the reduced version is $160/64 \approx 3$. Similarly, the KSD length proportions are $106/32 \approx 4$ and $96/32 = 3$. This indicates that $N = 2^{12} \cdot 4 = 2^{14}$ is an appropriate choice for the full version. To further enhance the success probability, we choose the sampling size $N = 2^{16}$. The theoretical relationship between N and the success probability is not easy to determine analytically, but we will try to pursue this issue in our future work.

In our simulations, we also tested the time of one tick for the reduced version on our PC. One tick needs approximately 2^9 CPU clocks (about $0.24 * 10^{-6}$ seconds to generate one keystream bit). When $l = 32$, the cost for NCA-2.0 is about 2 hours and 12 mins, which contains a number of I/O operations for table look-ups (access to hard disk). The size of the pre-computed tables is about 4 GB, which contains millions of txt files (This can be replaced by binary file, which is faster for read and write operations). These hard disk operations will add some burden to the running time our non-optimized C++ program. Therefore, there exist some gaps between the time complexity of Table 7, Table 8 and Table 9. We will further optimized our experimental code in the future. We can actually improve the running time by loading all the tables into CPU memory before the on-line attack, then the I/O operation costs can be reduced in the on-line phase.

9 Conclusion

In this paper, we have proposed a key recovery attack, called near collision attack on Grain v1. Based on some key observations, we have presented the basic attack called NCA-1.0 and further enhance it to NCA-2.0 and NCA-3.0 by combining the sampling resistance of Grain v1 and the non-uniform distribution of the KSD table size respectively. Our attack has been verified on a reduced version of Grain v1. Under some assumption, an extrapolation of the results indicates an attack on the original Grain v1 for any fixed IV in $2^{71.4}$ cipher ticks after the pre-computation of $2^{73.1}$ ticks, given $2^{62.8}$-bit memory and $2^{67.8}$ keystream bits, while the brute force attack can only be mounted for a fixed IV. Our attack is just a starting point for further analysis of Grain-like stream ciphers and hopefully it provides some new insights on the design of such compact stream ciphers.

References

1. Afzal, M., Masood, A.: Algebraic Cryptanalysis of A NLFSR Based Stream Cipher. *Information and Communication Technologies: From Theory to Applications, ICT-TA 2008*, pp. 1-6. (2008)
2. Aumasson, J.-P., Dinur, I., Henzen, L., Meier, W. and Shamir, A.,: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128, *Special Purpose Hardware for Attacking Cryptographic Systems-SHARCS'09'*. (2009)
3. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of Grain. In: M.J.B. Robshaw (Ed.): *Fast Software Encryption-FSE'2006*, LNCS vol 4047, pp. 15-29. (2006).
4. Berbain, C., Gilbert, H. and Joux, A.: Algebraic and correlation attacks against linearly filtered non linear feedback shift registers, in R. Avanzi, L. Keliher and F. Sica, eds, *Selected Areas in Cryptography-SAC 2008*, LNCS Vol. 5381, Springer-Verlag, pp. 184-198.
5. Biryukov, A. and Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers, in T. Okamoto, ed., *Advances in Cryptology-ASIACRYPT 2000*, LNCS Vol. 1976, Springer-Verlag, pp. 1-13. (2008)
6. Bjorstad, TE.: Cryptanalysis of Grain using Time/Memory/Data Tradeoffs. ecrypt.eu.org. (2006)
7. De Cannière, C., Kucuk, O, and Preneel, B. Analysis of Grain's initialization algorithm, in S. Vaudenay, ed., *Progress in Cryptology-AFRICACRYPT 2008*, LNCS Vol. 5023, Springer-Verlag, pp. 276-289. (2008)
8. Dinur, I., Guneysu, T., Paar, C., Shamir, A. and Zimmermann, R.: An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware, *Progress in Cryptology-ASIACRYPT'2011*, LNCS Vol. 7073, Springer-Verlag, pp. 327-373. (2011)
9. Dinur, I. and Shamir, A.: Breaking Grain-128 with dynamic cube attacks, in A. Joux, ed., *Fast Software Encryption-FSE'2011*, LNCS vol. 6733, Springer-Verlag, pp. 167-187. (2011)
10. http://www.ecrypt.eu.org/stream/e2-grain.html
11. Magma Computational Algebra System. http://magma.maths.usyd.edu.au/.
12. Grain-v1 Software Implementation using C. http://www.ecrypt.eu.org/stream/e2-grain.html/.
13. Hell, M., Johansson, T., Meier, W.,: Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing (IJWMC)*, vol. 2, No. 1, 2007. pp. 86-93. (2007)
14. Hell, M., Johansson, T., Maximov, A. and Meier, W.: A Stream Cipher Proposal: Grain-128, *IEEE International Symposium on Information Theory-ISIT'2006*, pp. 1614-1618. (2006)
15. Agren M., Hell, M., Johansson T., and Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. *International Journal of Wireless and Mobile Computing (IJWMC)* vol. 5, No. 1, pp. 48-59. (2011)
16. Khazaei, S., Hassanzadeh, M. and Kiaei., M.: Distinguishing Attack on Grain. ECRYPT Stream Cipher Project Report 2005/071, 2005. Available at http://www.ecrypt.eu.org/stream. (2005)

17. Lee, Y., Jeong, K., Sung, J. and Hong, S. Related-key chosen IV attacks on Grain-v1 and Grain-128, in Y. Mu, W. Susilo and J. Seberry, eds, *13th Australasian Conference on Information Security and Privacy-ACISP 2008*, LNCS Vol. 5107, Springer-Verlag, pp. 321-335. (2008)
18. Stankovski, P.: Greedy distinguishers and nonrandomness detectors, in G. Gong and K. C. Gupta, eds, *Progress in Cryptology-INDOCRYPT 2010*, LNCS vol. 6498, Springer-Verlag, pp. 210-226. (2010)
19. Strassen, V.: Gaussian elimination is not optimal: Nume. Math., vol. 13, pp. 354-356. (1969)
20. Zhang, H. and Wang, X.: Cryptanalysis of stream cipher Grain family, Cryptology ePrint Archive, Report 2009/109. http://eprint.iacr.org/. (2009)

A Recovering the NFSR Initial State

We will discuss how to recover the NFSR state at t_1, once the LFSR state at t_1 has been recovered. In [1], Afzal *et al.* conducted several experiments to retrieve the maximum number of bits that can be obtained when the other bits are guessed. Results show that no more than 77 bits can be recovered out of 160 bits, while guessing the remaining 83 bits (including all 80 bits LFSR and 3 bits of NFSR). The method is to solve an equation system containing the 77 unknown NFSR state bits. Furthermore, they also generate algebraic equations of Grain-v1 in Maple 10, and solved the equations with Magma V 2.13-5 [11] on a PC with CPU at 1.73 GHz and 1 GB RAM. The results are summarized in the following table. From Table 10, suppose that one operation (basic operation for solving the non-linear euqation) acts in one clock cycle of the CPU, then $1.73 \cdot 10^9 \approx 2^{29.9}$ operations are executed per second for 1.73 GHz PC. Therefore, we need $2^{29.9} \cdot 0.204 \cdot 2^3 \approx 2^{30.7}$ operations (or CPU clock cycles) and 150 bits keystream to retrieve all the 80 NFSR state at t_1. Now we need to convert this time complexity to one tick of Grain. According to Appendix C, one tick of Grain needs $\Omega = 2^{10.4}$ CPU clock cycles. Therefore, the complexity to recover the NFSR state is $T_N = 2^{30.7}/\Omega = 2^{20.3}$ cipher ticks.

Table 10. Simulation results of algebraic analysis of Grain

Version	No. of bits guessed[a]	No. of bits recovered	Time to find solution	Keystream bits used
Grain-v1	3	77	0.204 sec	150

[a] Since the LFSR state is known, only 3 NFSR need to be guessed.

B Proof of Lemma 2

Proof. The listing strategy is same to the proof of the Lemma 1 in [6]. However, we extend the steps from 18 to 21. From the output function of Grain $z_i = \sum_{k\in\mathcal{A}} n_{i+k} + h(l_{i+3}, l_{i+25}, l_{i+46}, l_{i+64}, n_{i+63},)$ where $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$.

We attempt to enumerate all the NFSR bits from n_{i+10} to n_{i+31}. It is important to know that the non-linear feedback of Grain does not affect the output function until the cipher has been clocked 18 times. Thus, we can easily derive the following 17 steps

$$\begin{aligned}
n_{10} &= z_0 + n_1 + n_2 + n_4 + n_{31} + n_{43} + n_{56} + h(l_3, l_{25}, l_{46}, l_{64}, n_{63}), \\
&\vdots \\
n_{25} &= z_{15} + n_{16} + n_{17} + n_{19} + n_{46} + n_{58} + n_{71} + h(l_{18}, l_{40}, l_{61}, l_{79}, n_{78}), \\
n_{26} &= z_{16} + n_{17} + n_{18} + n_{20} + n_{47} + n_{59} + n_{72} + h(l_{19}, l_{41}, l_{62}, l_{80}, n_{79}),
\end{aligned}$$

In step 1, the value of n_{10} can be determined by fixing 4 LFSR bits and 7 NFSR bits. We continue this procedure to derive the following values of $n_{11}, n_{12}, ..., n_{25}$. At this point, we have fixed 57 NFSR bits, 64 LFSR bits and deduced 16 NFSR bits. In step 17, l_{80} is involved in the computation of n_{26}, according to the linear feedback function, we need to fix 5 LFSR bits and 1 NFSR bit.

In step 18, n_{80} and l_{81} exist in the expression of n_{27}, we have to fix 5 LFSR bits and 2 NFSR bits (n_{28} and n_0) to deduce n_{27}. In the step 19, we can not deduce n_{28} in this step, since it was fixed in the last step. However, n_{29} exists in the expression of z_{18}, we can thus derive the value of n_{29} by fixing 3 LFSR bits. In step 20, n_{30} can be obtained by fixing 2 LFSR bits. In the last step (step 21), l_{45} can be deduced directly without fixing any state bit. All the state bits are known from step 22. The number of the fixed state bits are summarized in Table 11. To sum up, we have recovered the 20 NFSR state bits and 1 LFSR state bit using 60 bits of the NFSR state and 79 bits of the LFSR state (Altogether 139 bits of internal state). □

C How Can We Measure One-clock Cycle of Grain

The goal of this section is to measure the CPU clock cycle cost by one tick of Grain in software. The source code of Grain we used here is written by the designers [12]. We performed the testing several times on a PC with 2.83 GHz CPU and the average time to generate one keystream bit is $0.475 \cdot 10^{-6}$ seconds. Thus, one tick of Grain is $\Omega = 0.475 \cdot 10^{-6} \cdot 2.83 \cdot 10^{9} \approx 2^{10.4}$ CPU cycles. The testing codes is in Fig.5 (This program should run with the source code of Grain).

Table 11. The fixed state bits in each step

Step	Deduced bit	Fixed LFSR	Fixed NFSR bits
1	n_{10}	$l_3, l_{25}, l_{46}, l_{64}$	$n_1, n_2, n_4, n_{31}, n_{43}, n_{56}, n_{63}$
2	n_{11}	$l_4, l_{26}, l_{47}, l_{65}$	$n_3, n_5, n_{32}, n_{44}, n_{57}, n_{64}$
3	n_{12}	$l_5, l_{27}, l_{48}, l_{66}$	$n_6, n_{33}, n_{45}, n_{58}, n_{65}$
4	n_{13}	$l_6, l_{28}, l_{49}, l_{67}$	$n_7, n_{34}, n_{46}, n_{59}, n_{66}$
5	n_{14}	$l_7, l_{29}, l_{50}, l_{68}$	$n_8, n_{35}, n_{47}, n_{60}, n_{67}$
6	n_{15}	$l_8, l_{30}, l_{51}, l_{69}$	$n_9, n_{36}, n_{48}, n_{61}, n_{68}$
7	n_{16}	$l_9, l_{31}, l_{52}, l_{70}$	$n_{37}, n_{49}, n_{62}, n_{69}$
8	n_{17}	$l_{10}, l_{32}, l_{53}, l_{71}$	n_{38}, n_{50}, n_{70}
9	n_{18}	$l_{11}, l_{33}, l_{54}, l_{72}$	n_{39}, n_{51}, n_{71}
10	n_{19}	$l_{12}, l_{34}, l_{55}, l_{73}$	n_{40}, n_{52}, n_{72}
11	n_{20}	$l_{13}, l_{35}, l_{56}, l_{74}$	n_{41}, n_{53}, n_{73}
12	n_{21}	$l_{14}, l_{36}, l_{57}, l_{75}$	n_{42}, n_{54}, n_{74}
13	n_{22}	$l_{15}, l_{37}, l_{58}, l_{76}$	n_{55}, n_{75}
14	n_{23}	$l_{16}, l_{38}, l_{59}, l_{77}$	n_{76}
15	n_{24}	$l_{17}, l_{39}, l_{60}, l_{78}$	n_{77}
16	n_{25}	$l_{18}, l_{40}, l_{61}, l_{79}$	n_{78}
17	n_{26}	$l_{19}, l_{41}, l_{62}, l_0, l_{23}$	n_{79}
18	n_{27}	$l_{20}, l_{42}, l_{63}, l_1, l_{24}$	n_{28}, n_0
19	n_{29}	l_{21}, l_{43}, l_2	-
20	n_{30}	l_{22}, l_{44}	-
21	l_{45}	-	-

```
ECRYPT_ctx* ctx=new ECRYPT_ctx;
u32 KSLen=2000;
u8 key[10],IV[8],keyStream[KSLen];
ECRYPT_keysetup(ctx,key,80,64);
ECRYPT_ivsetup(ctx,IV);
clock_t start, finish;
double duration, speed;
start=clock();
ECRYPT_keystream_bytes(ctx,keyStream,KSLen);
finish=clock();
duration=((double)finish-start)/CLOCKS_PER_SEC;
speed=duration*2.83*1000*1000*1000/((double)KSLen*8);
printf("time%4.4f sec\n"
"The encryption speed is %3.4f cycles/bit \n",duration,speed);
```

Fig. 5. Code of testing.