A Chosen IV Related Key Attack on Grain-128a

Subhadeep Banik[1], Subhamoy Maitra[1], Santanu Sarkar[2], Meltem Sönmez Turan[2]

[1] Applied Statistics Unit, Indian Statistical Institute Kolkata, 203, B.T. Road, Kolkata-108.
`s.banik_r@isical.ac.in, subho@isical.ac.in`

[2] National Institute of Standards and Technology, 100 Bureau Drive, Stop 8930 Gaithersburg, MD 20899-8930,USA
`santanu.sarkar@nist.gov, meltem.turan@nist.gov`

Abstract. Due to the symmetric padding used in the stream cipher Grain v1 and Grain-128, it is possible to find Key-IV pairs that generate shifted keystreams efficiently. Based on this observation, Lee et al. presented a chosen IV related key attack on Grain v1 and Grain-128 at ACISP 2008. Later, the designers introduced Grain-128a having an asymmetric padding. As a result, the existing idea of chosen IV related key attack does not work on this new design. In this paper, we present a key recovery attack on Grain-128a, in a chosen IV related key setting. We show that using around $\gamma \cdot 2^{32}$ (γ is a experimentally determined constant and it is sufficient to estimate it as 2^8) related keys and $\gamma \cdot 2^{64}$ chosen IVs, it is possible to obtain $32 \cdot \gamma$ simple nonlinear equations and solve them to recover the secret key in Grain-128a.

Keywords: Cryptanalysis, eStream, Grain-128a, Related Keys, Stream Cipher.

1 Introduction

The Grain family of stream ciphers, proposed by Martin Hell, Thomas Johansson and Willi Meier in 2005, is designed for constrained devices. Grain v1 [16] is included in the final hardware portfolio of the eStream project [1]. To meet increased security requirements, the designers proposed a 128-bit version called Grain-128 in ISIT 2006 [17]. In both ciphers, the symmetric padding of all ones is used during the initialization of the internal state of the cipher, before the Key-IV mixing. Due to this symmetric padding, slide attacks based on the observation that one could obtain Key-IV pairs that produce ϵ-bit shifted keystream with probability $2^{-2\epsilon}$ were reported in [9]. This probability was improved to $2^{-\epsilon}$ in [6]. In the SKEW conference of 2011, the designers proposed the Grain-128a cipher that accommodated both functionalities of message encryption and authentication [2,3]. In order to protect against the previous attacks, the designers used an asymmetric padding in the design of Grain-128a. For detailed cryptanalytic results related to this family, the reader may refer to [4, 7–9, 11–14, 18–21, 23, 24].

The symmetric padding used in the initialization of Grain v1 and Grain-128 was also exploited in [20] to mount a chosen IV related key attack. Their main idea is to use related keys and chosen IVs to obtain shifted keystream and then to carefully study the scenario to obtain the secret key bits. The same attack fails against Grain-128a, for the following reasons:

1. The padding used in Grain-128a is a string of 31 ones followed by a zero. Because of this asymmetric nature of the padding it is not possible to obtain related key-IV pairs that produce shifted keystream bits for less than 32 bit shifts by using the idea of [9, 20]. Following their idea, getting related key-IV pairs for 32-bit shifted keystream should require an expected 2^{64} trials.
2. In Grain-128a, the first 64 keystream bits and thereafter every alternate keystream bit are used for computation of a MAC and hence are not directly available to the attacker. This also ensures that Grain-128a is resistant against the attack proposed in [20].

Thus one can argue that an attack against Grain-128a in the chosen IV related key setting is much more difficult and hence requires much higher computational effort compared to [20].

In this paper, first, we present a novel approach to obtain related key-IV pairs that produce 32-bit shifted keystream with an expected of 2^{32} trials. Using these Key-IV pairs, we present a key recovery attack on Grain-128a, in a chosen IV related key setting. We show that using around $\gamma \cdot 2^{32}$ (γ is an experimentally determined constant and it is sufficient to estimate it as 2^8) related keys and $\gamma \cdot 2^{64}$ chosen IVs, it is possible to obtain $32 \cdot \gamma$ simple nonlinear equations and solve them to recover the secret key in Grain-128a. We experimentally verified that solving these equations are practical, due to the simplicity of the equations.

The paper is organized as follows. In the next section, a brief explanation of chosen IV attacks and the structure of the Grain family of stream ciphers are presented. In Section 3, the Key-IV pairs that produce shifted keystreams in Grain-128a are discussed. In Section 4, the details of the chosen IV related key attack are presented, along with the experimental results. Finally, in Section 5 the conclusions of the paper are given.

2 Preliminaries

2.1 Chosen IV Attacks

The model used in chosen IV attacks is as follows. The adversary is given access to an Oracle which is in possession of an unknown quantity (typically the secret key). The adversary can choose a public parameter of his choice (typically the IV) and ask the Oracle to encrypt a message of his choice. In the context of stream ciphers, this implies that the adversary is able to obtain keystream bits by querying the Oracle possessing the secret key with any IV of his choice (See Fig. 3). The above process can be repeated with different IVs of the adversary's choice. The task of the adversary could be either (i) to compute the secret key efficiently or, (ii) to distinguish the keystream output from random stream.

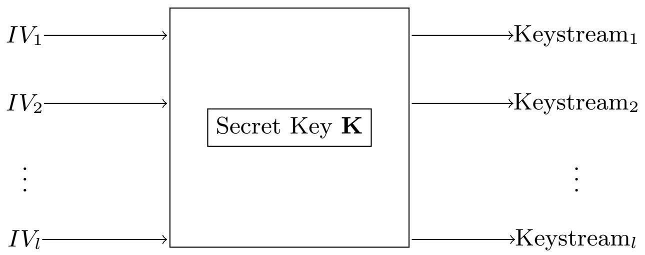

Fig. 1: Chosen IV Attack

The first model has been successfully employed in cube attacks on stream ciphers [12] whereas the second model has been used in distinguishing attacks on reduced round variants of stream and block ciphers [13, 14, 19, 21].

Chosen IV Related Key Attack This attack model relaxes the requirements of the chosen IV attack slightly. It is assumed that the adversary can somehow obtain keystream bits corresponding to the Key-IV pair $[f_i(\mathbf{K}), IV_{i,j}],\ i, j = 0, 1, 2, \ldots$, where $f_i : \mathcal{K} \to \mathcal{K}$ is a function from the Key-space $\mathcal{K}$ on to itself (See Fig. 2). As before the adversary attempts to recover the value of $\mathbf{K}$. Chosen IV related key attacks were successfully reported against Grain v1 and Grain-128 [20].

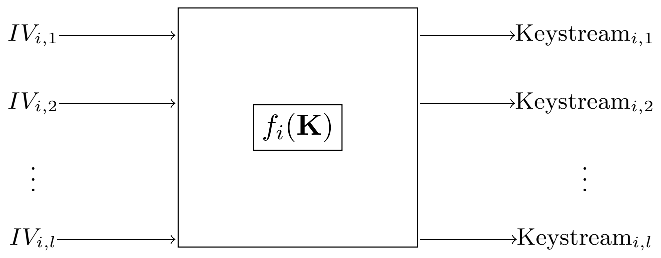

Fig. 2: Chosen IV Related Key Attack

2.2 Grain Family of Stream Ciphers

The Grain family of stream ciphers consists of two shift registers; an n-bit LFSR and an n-bit NFSR. Certain bits of both the registers are taken as inputs to a combining Boolean function, whence the keystream is produced. The structure of the Grain family is explained in Fig. 3. The update function of the LFSR is given by the equation $y_{t+n} = f(Y_t)$, where $Y_t = [y_t, y_{t+1}, \ldots, y_{t+n-1}]$ is an n-bit vector that denotes the LFSR state at time t and f is a linear function on the LFSR state bits obtained from a primitive polynomial in $GF(2)$ of degree n. The NFSR state is updated as $x_{t+n} = y_t + g(X_t)$. Here, $X_t = [x_t, x_{t+1}, \ldots, x_{t+n-1}]$ is an n-bit vector that denotes the NFSR state at time t and g is a nonlinear function of the NFSR state bits.

The output keystream is produced by combining the LFSR and NFSR bits as $z_t = h'(X_t, Y_t) = \bigoplus_{a \in A} x_{t+a} + h(X_t, Y_t)$, where A is a subset of $\{0, 1, 2, \ldots, n-1\}$ fixed by the specification of each Grain variant.

Key Loading Algorithm (KLA) The Grain family uses an n-bit key K, and an m-bit initialization vector IV, with $m < n$. The key is loaded in the NFSR and the IV is loaded in the first m bits of the LFSR. The remaining $n - m$ bits of the LFSR are loaded with some fixed pad $P \in \{0, 1\}^{n-m}$. Hence at this stage, the $2n$ bit initial state is of the form $K||IV||P$.

Key Scheduling Algorithm (KSA) After the KLA, for the first $2n$ clocks, the output of the function h' is XOR-ed to both the LFSR and NFSR update functions, i.e., during the first $2n$ clock intervals, the LFSR and the NFSR bits are updated as $y_{t+n} = z_t + f(Y_t)$, $x_{t+n} = y_t + z_t + g(X_t)$.

Pseudo-Random keystream Generation Algorithm (PRGA) After the of KSA, z_t is no longer XOR-ed to the LFSR and NFSR update functions but it is used as the output keystream bit. Therefore during this phase, the LFSR and NFSR are updated as $y_{t+n} = f(Y_t), x_{t+n} = y_t + g(X_t)$.

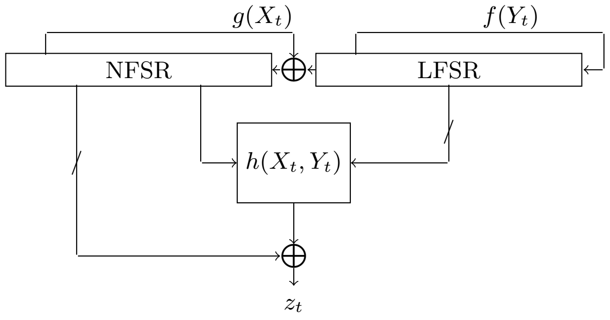

Fig. 3: Structure of Stream Cipher in Grain Family

2.3 Description of Grain-128a

Grain-128a authenticated encryption scheme consists of a 128 bit LFSR and a 128 bit NFSR. The size of the Key and IV is $n = 128$ and $m = 96$ bits, respectively. The value of the pad is $P = 0\mathtt{xffff\ fffe}$. The LFSR update function is given by

$$y_{t+128} \overset{\Delta}{=} f(Y_t) = y_{t+96} + y_{t+81} + y_{t+70} + y_{t+38} + y_{t+7} + y_t.$$

The NFSR state is updated as follows

$$\begin{aligned} x_{t+128} = \; & y_t + g(x_{t+96}, x_{t+95}, x_{t+93}, x_{t+92}, x_{t+91}, x_{t+88}, x_{t+84}, x_{t+82}, x_{t+78}, \\ & x_{t+70}, x_{t+68}, x_{t+67}, x_{t+65}, x_{t+61}, x_{t+59}, x_{t+48}, x_{t+40}, x_{t+27}, \\ & x_{t+26}, x_{t+25}, x_{t+24}, x_{t+22}, x_{t+13}, x_{t+11}, x_{t+3}, x_t), \end{aligned}$$

where $g(x_{t+96}, x_{t+95}, \ldots, x_t) \overset{\Delta}{=} g(X_t) =$

$$\begin{aligned} & x_t + x_{t+26} + x_{t+56} + x_{t+91} + x_{t+96} + x_{t+3}x_{t+67} + x_{t+11}x_{t+13} + \\ & x_{t+17}x_{t+18} + x_{t+27}x_{t+59} + x_{t+40}x_{t+48} + x_{t+61}x_{t+65} + x_{t+68}x_{t+84} + \\ & x_{t+88}x_{t+92}x_{t+93}x_{t+95} + x_{t+22}x_{t+24}x_{t+25} + x_{t+70}x_{t+78}x_{t+82}. \end{aligned}$$

The pre-output function z_t is defined as

$$\sum_{j\in A} x_{t+j}+y_{t+93}+h(x_{t+12},y_{t+8},y_{t+13},y_{t+20},x_{t+95},y_{t+42},y_{t+60},y_{t+79},y_{t+94})$$

where $A = \{2,15,36,45,64,73,89\}$ and $h(s_0,\dots,s_8) = s_0s_1 + s_2s_3 + s_4s_5 + s_6s_7 + s_0s_4s_8$. The output function is defined as $y_t = z_{64+2t}$.

Authentication We use the description as explained in [3]. Assume that we have a message of length L defined by the bits $m_0,\dots,m_{L-1}$. Set $m_L = 1$ as padding. To provide authentication, two registers, called accumulator and shift register of size 32 bits each, are used. The content of accumulator and shift register at time t are denoted by $a_t^0,\dots,a_t^{31}$ and $r_t,\dots,r_{t+31}$, respectively. The accumulator is initialized through $a_0^t = z_t, 0 \le t \le 31$ and the shift register is initialized through $r_t = z_{32+t}, 0 \le t \le 31$. The shift register is updated as $r_{t+32} = z_{64+2t+1}$. The accumulator is updated as $a_{t+1}^j = a_t^j + m_t r_{t+j}$ for $0 \le j \le 31$ and $0 \le t \le L$. The final content of accumulator, $a_{L+1}^0,\dots,a_{L+1}^{31}$ is used for authentication.

3 Key-IV Pairs that Produce Shifted Keystream in Grain-128a

In [9], a method to obtain Key-IV pairs K, IV and K', IV' in Grain v1 and Grain-128, that produce ϵ-bit shifted keystream bits by performing a random experiment $2^{2\epsilon}$ many times is presented. The complexity was improved to 2^ϵ in [6]. Both these techniques utilized the fact that the padding P used in Grain v1 and Grain-128 was symmetric, i.e. a string of all ones. And in both [6,9], it was suggested that the method would fail if an asymmetric padding was used. This is precisely the strategy employed in Grain-128a, where the padding is $P =$`0x ffff fffe` is a set of 31 ones followed by a single zero.

In this section, we explain how despite of the asymmetric nature of P, one can obtain related key-IV pairs K, IV and K', IV' in Grain-128a such that they produce exactly 32-bit shifted keystream by running a random experiment 2^{32} times. We begin by noting that the state update functions in both the KSA and PRGA in the Grain family are one-to-one and invertible. This is because the state update functions of the NFSR and the LFSR can be written in the form

$$g(x_0,x_1,\dots,x_{127}) = x_0 + g'(x_1,\dots,x_{127})$$

$$f(y_0, y_1, \ldots, y_{127}) = y_0 + f'(y_1, \ldots, y_{127}).$$

This implies that one can construct the KSA^{-1} routine that takes a $2n$ bit vector S_i denoting the internal state of the cipher at any i^{th} round of the KSA, returns the $2n$ bit vector S_{i-1} denoting the internal state of the cipher at the previous round of the KSA. The same is true for the PRGA. A detailed description of the KSA^{-1} routine are given in Algorithm 1.

Algorithm 1: KSA^{-1} routine for the Grain-128a

```
Input: State $S_i = (x_0, \ldots, x_{127}, y_0, \ldots, y_{127})$
Output: The preceding State $S_{i-1} = (x_0, \ldots, x_{127}, y_0, \ldots, y_{127})$
$l = y_{127}$ and $n = x_{127}$
for $t = 127$ to $1$ do
    $y_t = y_{t-1}$ and $x_t = x_{t-1}$
end
$z = {}_{a \in A} x_a + y_{93} + h(x_{12}, y_8, y_{13}, y_{20}, x_{95}, y_{42}, y_{60}, y_{79}, y_{94})$
$y_0 = z + l + f'(y_1, \ldots, y_{127})$
$x_0 = z + n + y_0 + g'(x_1, \ldots, x_{127})$
```

Given this information, our strategy to find related key-IV pairs in Grain-128a will be as follows. Let $K = (k_0.k_1, k_2, \ldots, k_{127})$ be the key. We choose a 96-bit IV of the form

$$IV = (v_0, v_1, \ldots, v_{63}, \underbrace{1, 1, \ldots, 1, 0}_{32})$$

Therefore the initial state

$$\begin{aligned} S = K||IV||P =& (s_0, s_1, \ldots, s_{255}) \\ =& (k_0, \ldots, k_{127},\ v_0, \ldots, v_{63}, \underbrace{1, 1, \ldots, 1, 0}_{32}, \underbrace{1, 1, \ldots, 1, 0}_{32}). \end{aligned}$$

If we apply the KSA^{-1} to S, 32 times, then we get the following internal state;

$$S' = (a_0, a_1, \ldots, a_{31}, k_0, k_1, \ldots, k_{95},\ b_0, b_1, \ldots, b_{31}, v_0, v_1, v_{63}, 1, \ldots, 1, 0).$$

where the values of a_i, b_i for $0 \leq i \leq 31$ are given by polynomial functions in $k_0, \ldots, k_{127}$, $v_0, \ldots, v_{63}$. The exact form of these functions can be found out by executing the KSA^{-1} routine 32 times.

Note that S' is a valid initial state for Grain-128a, since it is of the form $K'||IV'||P$, where $K' = (a_0, a_1, \ldots, a_{31}, k_0, k_1, \ldots, k_{95})$ and $IV' =$

$(b_0, b_1, \ldots, b_{31}, v_0, v_1, v_{63})$. Therefore if one were to initialize Grain-128a with K', IV' then the internal state of the cipher after the KSA round $32+t$ will be the same as the internal state after t rounds of initialization with K, IV. This would be true for all $t \leq 224$. After this, the cipher initialized with K', IV' would enter the PRGA phase while the one initialized with K, IV would still be in the KSA phase. As we have already seen, in the Grain family of ciphers, the output bit feedback to the internal state, is discontinued after the KSA. Therefore the state updates in the next 32 rounds are not guaranteed to be identical. The situation has been explained pictorially in Fig. 4.

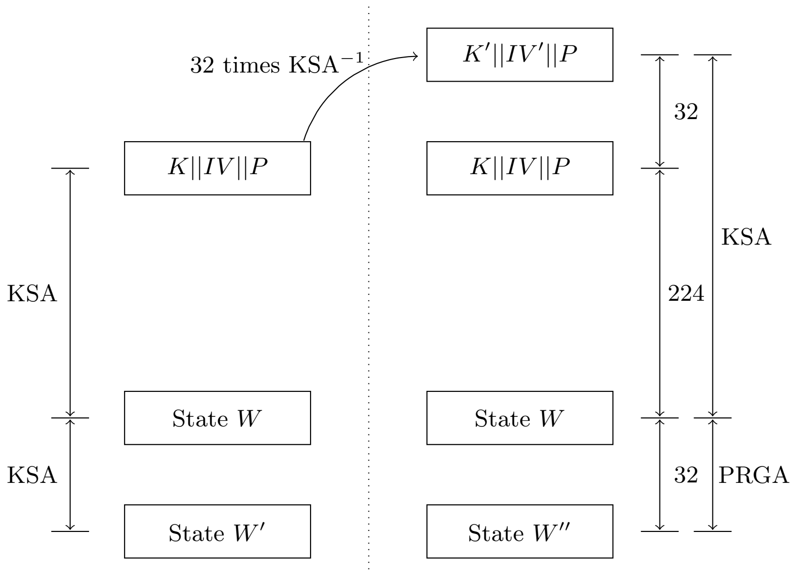

Fig. 4: Construction of Related Key-IV pairs in Grain Family

For the state updates to be identical in the next 32 rounds, it is necessary and sufficient that the cipher initialized with K', IV' produces zero keystream bits for each of these 32 rounds. After this, both systems run in PRGA mode and so if the internal state of the cipher with K, IV just after the KSA is equal to the internal state of the cipher with K', IV' after 32 PRGA rounds, then they will remain the same forever thereafter. In such a situation the $(32+t)^{th}$ PRGA state produced by K', IV' will be equal to the t^{th} PRGA state produced by K, IV for all $t > 0$. In such a situation it is natural that K', IV' and K, IV will produce 32 bit shifted keystream bits.

Now if we choose random values of $K \in \{0,1\}^{128}$ and $IV = V||P$ with $V \in \{0,1\}^{64}$, then it is expected that in one out of 2^{32} trials we will obtain a K', IV' which produces an all zero output stream in the first 32 PRGA rounds. If so, K', IV' and K, IV will produce 32 bit shifted keystream bits. The arguments are formalized in Algorithm 2.

Algorithm 2: Constructing Key-IV pairs that generate 32 bit shifted keystream

```
Output: Key-IV pairs $K', IV'$ and $K, IV$ that generate 32 bit shifted
        keystream
$s \leftarrow 0$;
while $s = 0$ do
    Choose $K \in_R \{0,1\}^{128}$, $V \in_R \{0,1\}^{64}$;
    $IV \leftarrow V||P$;
    Run KSA$^{-1}(K||IV||P)$ routine for 32 clocks and produce state
    $S' = (K'||IV'||P)$;
    if $K', IV'$ produces all zero keystream bits in the first 32 PRGA rounds
    then
        $s \leftarrow 1$;
        Return $(K, IV)$ and $(K', IV')$;
    end
end
```

Example 1. In the following table, we present two Key-IV pairs that generate 32-bit shifted keystreams for Grain-128a. It can be seen that the second Key-IV pair has been obtained by the right shifting the first Key-IV pair by 32 bits. The pairs were found in around 2^{32} random trials using Algorithm 2. It should be noted that output bits given in the table includes the bits used for authentication and encryption.

Pair	Key	IV	Output bits
1	9bbe 7e2b b99d 1477	5a7c 21e9 3a77	41d5c1f0387c
	0317 9f3b a1aa 8c70	52ce ffff fffe	3bf64e031725
2	f32a 7bd3 9bbe 7e2b	032d 0fee 5a7c	0000000041d5c1f0387c
	b99d 1477 0317 9f3b	21e9 3a77 52ce	3bf64e031725

Remark 1. It is also possible to obtain two Key-IV pairs K_1, IV_1 and K_2, IV_2 that produce r-bit shifted keystream bits (where $1 \leq r \leq 31$) by using a slight modification of the ideas presented in [6]. But in that case

K_1, IV_1 and K_2, IV_2 would be structurally unrelated i.e. no meaningful similarity exists between these pairs. Such pairs cannot be used to mount a Chosen IV related key attack of the nature that we are about to describe.

4 A Chosen IV Related Key attack on Grain-128a

We will now present a technique to cryptanalyze Grain-128a in the related key and chosen IV setting i.e. in accordance to the model presented in Section 2.1. It is worth noting that Algorithm 2 cannot be directly applied in this problem due to two reasons. First, the key is assumed to be secret in this model and so executing the KSA^{-1} routine 32 times has to be done over the key variables $k_0, k_1, \ldots, k_{127}$ rather than bits. A second reason is that in Grain-128a the first 64 keystream bits and every alternate keystream bit thereof goes towards the computation of MAC and is unavailable to the attacker directly. Hence it is not possible to check if the first 32 keystream bits produced by any Key-IV pair is all zero or not.

Let $K = (k_0, k_1, k_2, \ldots, k_{127})$ be the 128-bit secret key. We will write $K = \alpha_0||\alpha_1||\alpha_2||\alpha_3$ where each α_i is a 32-bit word given by the equation $\alpha_i = (k_{32i}, k_{32i+1}, \ldots, k_{32i+31})$ etc. Let the initial vector $IV = \beta_0||\beta_1||P$, where β_is are 32 bit words. If we initialize the cipher with K, IV we get the initial state

$$S = \alpha_0||\alpha_1||\alpha_2||\alpha_3 \;||\; \beta_0||\beta_1||P||P.$$

Now let us fix β_0 and β_1 to some fixed 32-bit values and let the α_is be unknowns, and apply the KSA^{-1} routine over S, 32 times. We will get a new state S' of the form

$$S' = \chi||\alpha_0||\alpha_1||\alpha_2 \;||\; \varUpsilon||\beta_0||\beta_1||P.$$

where each bit in χ can be expressed as polynomial functions over the secret key variables $k_0, k_1, \ldots, k_{127}$. The form of these polynomials will of course depend on the exact values of β_0, β_1. So, we can write

$$\chi = f_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3),$$

where $f_{\beta_0||\beta_1} : \{0,1\}^{128} \to \{0,1\}^{32}$ denotes a set of 32 Boolean functions. Similarly one can write

$$\varUpsilon = g_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3),$$

where $g_{\beta_0||\beta_1} : \{0,1\}^{128} \to \{0,1\}^{32}$ denotes another set of 32 Boolean functions. The exact forms of the functions $f_{\beta_0||\beta_1}, g_{\beta_0||\beta_1}$ for any value of

β_0, β_1 can be computed efficiently by implementing the KSA^{-1} routine in any computer algebra system like Sage [22]. Note that for S and S' to produce 32-bit shifted keystream we need that the first 32 output bits produced by S' be all 0s. Again this cannot be checked as the first 64 keystream bits are not directly available. Therefore our strategy to proceed will be as follows

1. Fix some value of β_0 and β_1.
2. Calculate the polynomials $f_{\beta_0||\beta_1}, g_{\beta_0||\beta_1}$.
3. Query the oracle for keystream bits produced by $K = \alpha_0||\alpha_1||\alpha_2||\alpha_3$, $IV = \beta_0||\beta_1||P$ and $K' = f_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3)||\alpha_0||\alpha_1||\alpha_2$, $IV' = \eta||\beta_0||\beta_1$, where η varies over all possible 32 bit words.
4. We check if, for any value of η, we get 32-bit shifted keystream bits. This can be done by checking the keystream bits after round 64 that Grain-128a makes directly available.
5. We will get 32-bit shifted keystream if and only if the following two occur simultaneously
   A. $\eta = g_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ AND
   B. The first 32 keystream bits produced by K', IV' are all zeros.
6. We know that **A.** will be satisfied for exactly one value of η and for that value of η, the condition **B.** may not hold and so for this value of β_0, β_1, none of the 2^{32} values of η yields shifted keystream bits. In such an event we take a new value of β_0 and β_1 and repeat the process.

Now we know that on expectation, by trying out 2^{32} random values of β_0 and β_1, we are likely to land up with a related key-IV pair K', IV' that produces all zeroes in the first 32 output rounds. Therefore by running the above experiment 2^{32} times we are likely to obtain some values of β_0, β_1, η such that $K' = f_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3)||\alpha_0||\alpha_1||\alpha_2$, $IV' = \eta||\beta_0||\beta_1$ such that K', IV' and K, IV produce 32-bit shifted keystream bits. When this happens, we obtain the following set of 32 nonlinear equations in the secret key bits;

$$\eta = g_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3).$$

Hence, by repeating the above process for $\gamma_1 \cdot 2^{32}$ different values of β_0, β_1 for any fixed value of the secret key we will on expectation be able to obtain $32 \cdot \gamma_1$ equations.

Next, we can start the above process for the single bit left rotated version of the secret key K i.e. $K \lll 1 = k_1, k_2, \ldots, k_{127}, k_0$. Then by expending the same computational effort we would be able to obtain another $32 \cdot \gamma_1$ nonlinear equations in the key bits. In general by starting the routine with the i-bit cyclically left rotated key $K \lll i$, for $i = 0, 1, 2, \ldots, \gamma_2 - 1$, we would in total get $32 \cdot \gamma_1 \cdot \gamma_2 = 32 \cdot \gamma$ equations in the key bits which can be solved together to recover the secret key for some suitable value of γ.

4.1 Complexity of the attack

For each phase of the attack that yields $32 \cdot \gamma_1$ equations we need to try out on an average $\gamma_1 \cdot 2^{32}$ different values of β_0, β_1, and for each these values of β_0, β_1 we need to try out in the worst case 2^{32} different values of η. This leads to the use of $\gamma_1 \cdot 2^{32}$ related keys. Since we use 2^{32} IVs for each related key this leads to a total of $\gamma_1 \cdot 2^{64}$ chosen IVs for each phase of the attack. We need to check at least the first 128 keystream bits output from each related key-IV pair to determine if the keystreams are 32-bit shifts or not and so that requires $\gamma.2^{64} \cdot 128 = \gamma_1 \cdot 2^{71}$ keystream bits. Repeating each phase γ_2 number of times for each rotated version of the secret key, increases the number of related keys to $\gamma_1 \cdot \gamma_2 \cdot 2^{32} = \gamma \cdot 2^{32}$, the number of chosen IVs to $\gamma \cdot 2^{64}$ and the number of keystream bits to $\gamma \cdot 2^{71}$. The computational effort is therefore proportional to $\gamma \cdot 2^{64}$.

4.2 Experimental Results

After obtaining $32 \cdot \gamma$ nonlinear equations, the attacker needs to solve these equations to obtain the secret key. For the attack to be meaningful, the nonlinear equations should be as simple as possible so that the attacker can solve the system efficiently. However, to get $32 \cdot \gamma$ equations, the attacker needs a computational effort of the order of $\gamma \cdot 2^{64}$, which is infeasible with the processing resources at our disposal. So, in order to prove that after obtaining the required number of equations, the attacker can recover the secret key efficiently we make the following assumptions.

1. We assume that the attacker has succeeded in obtaining $32 \cdot \gamma_1$ equations by using γ_1 random tuples of $[\beta_0, \beta_1, \eta]$. Using each such tuple we construct the set of 32 equations

$$\eta = g_{\beta_0||\beta_1}(\alpha_0, \alpha_1, \alpha_2, \alpha_3).$$

   We have simulated this situation as determining the actual values of $[\beta_0, \beta_1, \eta]$ is difficult to obtain in practical time using the computational resources that we have.

2. We have observed that for each tuple $[\beta_0, \beta_1, \eta]$, only a few of the equations are very complex, therefore they were not used in our system of equations. Out of each set of 32 equations around $20-22$ equations are of low degree and are used for solving the system.
3. We repeat the above process for γ_2 cyclically left rotated versions of the secret key and thus obtain $32 \cdot \gamma_1 \cdot \gamma_2 = 32 \cdot \gamma$ equations in the secret key bits.

After this we attempt to solve these equations using the standard SAT solver available in the computer algebra system Sage 5.4.1 [22]. Table 1 lists the total number of polynomials and the required time to solve these equations to obtain the secret key. As seen from the table, it is possible to solve these equations in less than 1 hour using a dual core PC, with a CPU speed of 1.83 GHz and 2 GB RAM. These results show that once the attacker can obtain enough equations, then he can solve them efficiently to recover the secret key.

γ_1	γ_2	$\gamma = \gamma_1 \cdot \gamma_2$	Total # Polynomials	Time (in seconds)
20	13	260	20γ	112.80
20	11	220	22γ	3240.62

Table 1: Experimental Results

4.3 Possible Countermeasures

The computational complexity required to mount our attack is given as $\gamma \cdot 2^{64} = \gamma \cdot 2^{2|P|}$ where $|P| = 32$ is the length of the padding used in the initialization of Grain-128a. Thus to make the attack worse than brute force $|P|$ needed to be more than or equal to half the length in bits of the secret key i.e. $\frac{128}{2} = 64$. So prevention of such an attack on Grain-like ciphers requires that the bit-length of the padding be atleast half the bit-length of the Secret Key. For Trivium like ciphers where there is no difference in the operations performed during the KSA and the PRGA, the length of padding must be atleast equal to the length of the secret key (this is indeed the case for Trivium whose keylength is 80 and where the length of the pad is 128).

Another popular approach used to prevent slide attacks altogether is the ones used in KATAN [10] and Quark [5], where update of two shift registers would be controlled by a third register which is usually initialized to a fixed constant at the start of operations. Performing a slide attack on then would require a simultaneous synchronization of the third register for the related key-IV pair, which is not possible as it always starts with a fixed constant. This of course requires extra hardware and hence increases the area and power consumption of the device implementing the cipher.

5 Conclusion

In this paper we present a chosen IV related key attack against the stream cipher Grain-128a. A similar attack against Grain v1 and Grain-128 were proposed in [20]. The attack worked due to the symmetric padding used in both Grain v1 and Grain-128. The new design, Grain-128a, uses an asymmetric padding and consequently the attack of Lee et. al. [20] does not work in this scenario. We show that by using around $\gamma \cdot 2^{32}$ related keys and $\gamma \cdot 2^{64}$ chosen IVs the attacker can obtain $32 \cdot \gamma$ nonlinear equations in the secret key bits which he can then solve to recover the secret key in Grain-128a. Our attack on Grain-128a requires higher complexities than that of [20] on Grain v1 and Grain-128. However obtaining attacks against Grain-128a with lesser complexities is elusive due to the asymmetric padding.

Acknowledgments: The authors like to thank the Centre of Excellence in Cryptology, Indian Statistical Institute for relevant support towards this research. The authors also like to thank the anonymous reviewers for their helpful suggestions.

References

1. The ECRYPT Stream Cipher Project. eSTREAM Portfolio of Stream Ciphers. Revised on September 8, 2008.
2. M. Ågren, M. Hell, T. Johansson and W. Meier. A New Version of Grain-128 with Authentication. Symmetric Key Encryption Workshop 2011, DTU, Denmark, February 2011.
3. M. Ågren, M. Hell, T. Johansson and W. Meier. Grain-128a: A New Version of Grain-128 with Optional Authentication. IJWMC, 5(1): 48–59, 2011. This is the journal version of [2].
4. J. P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir. Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. In SHARCS - Special-purpose Hardware for Attacking Cryptographic Systems. 2009.

5. J. P. Aumasson, L. Henzen, W. Meier, M. Naya-Plasencia. Quark: A Lightweight Hash. Journal of Cryptology 26(2): pp. 313–339, 2013.
6. S.Banik, S.Maitra and S.Sarkar. Some Results on Related Key-IV Pairs of Grain. In SPACE 2012, LNCS, Vol. 7644, pp. 94–110, 2012.
7. C. Berbain, H. Gilbert and A. Maximov. Cryptanalysis of Grain. In FSE 2006, LNCS, Vol. 4047, pp. 15–29, 2006.
8. T. E. Bjørstad. Cryptanalysis of Grain using Time/Memory/Data tradeoffs (v1.0 / 2008-02-25). Available at `http://www.ecrypt.eu.org/stream`.
9. C. De Cannière, O. Küçük and B. Preneel. Analysis of Grain's Initialization Algorithm. In AFRICACRYPT 2008, LNCS, Vol. 5023, pp. 276–289, 2008.
10. C. De Cannière, O. Dunkelman, M. Knezevic. KATAN and KTANTAN–a family of small and efficient hardware-oriented block ciphers. In CHES 2009, LNCS, Vol. 5747, pp. 272–288, 2009.
11. I. Dinur, T. Güneysu, C. Paar, A. Shamir, R. Zimmermann. An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware. In ASIACRYPT 2011, LNCS Vol. 7073, pp. 327–343, 2011.
12. I. Dinur, A. Shamir. Breaking Grain-128 with Dynamic Cube Attacks. In FSE 2011, LNCS, Vol. 6733, pp. 167–187, 2011.
13. H. Englund, T. Johansson, and M. Sönmez Turan. A Framework for Chosen IV Statistical Analysis of Stream Ciphers. In INDOCRYPT 2007, LNCS, Vol. 4859, pp. 268–281, 2007.
14. S. Fischer, S. Khazaei, and W. Meier. Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In AFRICACRYPT 2008, LNCS, Vol. 5023, pp. 236–245, 2008.
15. H. Fredricksen. A Survey of Full Length Nonlinear Shift Register Cycle Algorithms, SIAM Rev., 24 (1982), 195-221.
16. M. Hell, T. Johansson and W. Meier. Grain - A Stream Cipher for Constrained Environments. ECRYPT Stream Cipher Project Report 2005/001, 2005. Available at `http://www.ecrypt.eu.org/stream`.
17. M. Hell, T. Johansson, A. Maximov and W. Meier. A Stream Cipher Proposal: Grain-128. In IEEE International Symposium on Information Theory (ISIT 2006), 2006.
18. S. Khazaei, M. Hassanzadeh and M. Kiaei. Distinguishing Attack on Grain. ECRYPT Stream Cipher Project Report 2005/071, 2005. Available at `http://www.ecrypt.eu.org/stream`
19. S. Knellwolf, W. Meier and M. Naya-Plasencia. Conditional Differential Cryptanalysis of NLFSR-based Cryptosystems. In ASIACRYPT 2010, LNCS, Vol. 6477, pp. 130–145, 2010.
20. Y. Lee, K. Jeong, J. Sung and S. Hong. Related-Key Chosen IV Attacks on Grain-v1 and Grain-128. In ACISP 2008, LNCS, Vol. 5107, pp. 321–335, 2008.
21. P. Stankovski. Greedy Distinguishers and Nonrandomness Detectors. In INDOCRYPT 2010, LNCS, Vol. 6498, pp. 210–226, 2010.
22. W. Stein. Sage Mathematics Software. Free Software Foundation, Inc., 2009. Available at `http://www.sagemath.org`. (Open source project initiated by W. Stein and contributed by many).
23. B. Zhang and Z. Li. Near Collision Attack on the Grain v1 Stream Cipher. To appear in FSE 2013.
24. H. Zhang and X. Wang. Cryptanalysis of Stream Cipher Grain Family. IACR Cryptology ePrint Archive 2009: 109. Available at `http://eprint.iacr.org/2009/109`.