

New Configurations of Grain Ciphers: Security Against Slide Attacks

Diana Maimuț¹  and George Teșeleanu^{1,2} 

¹ Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
{diana.maimut,tgeorge}@dcti.ro

² Department of Computer Science
“Al.I.Cuza” University of Iași 700506 Iași, Romania,
george.teseleanu@info.uaic.ro

Abstract. eSTREAM brought to the attention of the cryptographic community a number of stream ciphers including Grain v0 and its revised version Grain v1. The latter was selected as a finalist of the competition’s hardware-based portfolio. The Grain family includes two more instantiations, namely Grain-128 and Grain-128a.

The scope of our paper is to provide an insight on how to obtain secure configurations of the Grain family of stream ciphers. We propose different variants for Grain and analyze their security with respect to slide attacks. More precisely, as various attacks against initialization algorithms of Grain were discussed in the literature, we study the security impact of various parameters which may influence the LFSR’s initialization scheme.

1 Introduction

The Grain family of stream ciphers consists of four instantiations Grain v0 [12], Grain v1 [13], Grain-128 [11] and Grain-128a [18]. Grain v1 is a finalist of the hardware-based eSTREAM portfolio [1], a competition for choosing both hardware and software secure and efficient stream ciphers.

The design of the Grain family of ciphers includes an LFSR. The loading of the LFSR consists of an initialization vector (IV) and a certain string of bits P whose lengths and structures depend on the cipher’s version. Following the terminology used in [6], we consider the IV as being padded with P . Thus, throughout this paper, we use the term *padding* to denote P . Note that Grain v1 and Grain-128 make use of *periodic* IV padding and Grain-128a uses *aperiodic* IV padding.

A series of attacks against the Grain family padding techniques appeared in the literature [5,6,8,16] during the last decade. In the light of these attacks, our paper proposes the first security analysis³ of generic IV padding schemes for Grain ciphers in the *periodic* as well as the *aperiodic* cases.

In this context, the concerns that arise are closely related to the security impact of various parameters of the padding, such as the position and structure of the padding block. Moreover, we consider both *compact* and *fragmented* padding blocks in our study. We refer to the original padding schemes of the Grain ciphers as being compact (*i.e.* a single padding block is used). We denote as fragmented padding the division of the padding block into smaller blocks of equal length⁴.

By examining the structure of the padding and analyzing its compact and especially fragmented versions, we actually study the idea of extending the key’s life. The latter could be achieved by introducing a variable padding according to suitable constraints. Hence, the general question that

³ against slide attacks

⁴ we consider these smaller blocks as being spread among the linear feedback register’s data

arises is the following: *what is to be loaded in the LFSRs of Grain ciphers in order to obtain secure settings?* Note that our study is preliminary, taking into account only slide attacks. We consider other types of attacks as future work.

We stress that finding better attacks than the ones already presented in the literature is outside the scope of our paper, as our main goal is to establish sound personalized versions of the Grain cipher. Hence, our work does not have any immediate implication towards breaking any cipher of the Grain family. Nevertheless, our observations become meaningful either in the lightweight cryptography scenario or in the case of an enhanced security context (e.g. secure government applications).

Lightweight cryptography lies at the crossroad between cryptography, computer science and electrical engineering [17]. Thus, trade-offs between performance, security and cost must be considered. Given such constraints and the fact that embedded devices operate in hostile environments, there is an increasing need for new and varied security solutions, mainly constructed in view of the current ubiquitous computing tendency. As the Grain family lies precisely within the lightweight primitives' category, we believe that the study presented in the current paper is of interest for the industry and, especially, government organizations.

When dealing with security devices for which the transmission and processing of the IV is neither so costly nor hard to handle (e.g. the corresponding communication protocols easily allow the transmission), shrinking the padding up to complete removal might be considered. More precisely, we suggest the use of a longer IV in such a context in order to increase security. Moreover, many Grain-type configurations could be obtained if our proposed padding schemes are used. Such configurations could be considered as personalizations of the main algorithm and, if the associated parameters are kept secret, the key's life can be extended.

Structure of the Paper. We introduce notations and give a quick reminder of the Grain family technical specifications in Section 2. Section 3 describes generic attacks against the Grain ciphers. In Section 4 we discuss the core result of our paper: a security analysis of IV padding schemes for Grain ciphers. We conclude and underline various interesting ideas as future work in Section 5. We recall Grain v1 in Appendix A, Grain-128 in Appendix B and Grain-128a in Appendix C. We do not recall the corresponding parameters of Grain v0, even though the results presented in the current paper still hold in that case. In Appendices D and E we provide test values for our proposed algorithms.

2 Preliminaries

Notations. During the following, capital letters will denote padding blocks and small letters will refer to certain bits of the padding. We use the big-endian convention. Hexadecimal strings are marked by the prefix 0x.

$MSB_\ell(Q)$	stands for the most significant ℓ bits of Q
$LSB_\ell(Q)$	stands for the least significant ℓ bits of Q
$MID_{[\ell_1, \ell_2]}(Q)$	stands for the bits of Q between position ℓ_1 and ℓ_2
$x y$	represents the string obtained by concatenating y to x
\in_R	selecting an element uniformly at random
$ x $	the bit-length of x
b^t	stands for t consecutive bits of b
$NULL$	stands for an empty variable

2.1 Grain Family

Grain is a hardware-oriented stream cipher initially proposed by Hell, Johansson and Meier [12] and whose main building blocks are an n bit *linear feedback shift register* (LFSR), an n bit *non-linear feedback shift register* (NFSR) and an *output function*. Because of a weakness in the output function, a key recovery attack [7] and a distinguishing attack [14] on Grain v0 were proposed. To solve these security issues, Grain v1 [13] was introduced. Also, Grain-128 [11] was proposed as a variant of Grain v1. Grain-128 uses 128-bit keys instead of 80-bit keys. Grain 128a [18] was designed to address cryptanalysis results [4, 9, 10, 15, 19] against the previous version. Grain 128a offers optional authentication. We stress that, in this paper, we do not address the authentication feature of Grain-128a.

Let $X_i = [x_i, x_{i+1}, \dots, x_{i+n-1}]$ denote the state of the NFSR at time i and let $g(x)$ be the nonlinear feedback polynomial of the NFSR. $g(X_i)$ represents the corresponding update function of the NFSR. In the case of the LFSR, let $Y_i = [y_i, y_{i+1}, \dots, y_{i+n-1}]$ be its state, $f(x)$ the linear feedback polynomial and $f(Y_i)$ the corresponding update function. The filter function $h(X_i, Y_i)$ takes inputs from both the states X_i and Y_i .

We shortly describe the generic algorithms KLA, KSA and PRGA below. As KSA is invertible, a state $S_i = X_i || Y_i$ can be rolled back one clock to S_{i-1} . We further refer to the transition function from S_i to S_{i-1} as KSA^{-1} .

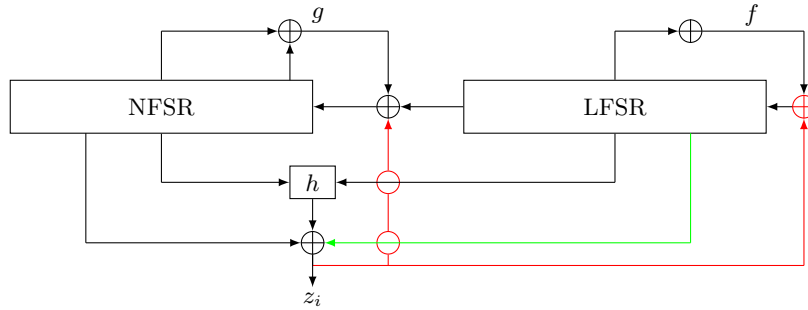


Fig. 1: Output Generator and Key Initialization of Grain ciphers

Key Loading Algorithm (KLA). The Grain family uses an n -bit key K , an m -bit initialization vector IV with $m < n$ and some fixed padding $P \in \{0, 1\}^\alpha$, where $\alpha = n - m$. The key is loaded in the NFSR, while the pair (IV, P) is loaded in the LFSR using a one-to-one function further denoted as $\text{Load}_{IV}(IV, P)$.

Key Scheduling Algorithm (KSA). After running KLA, the output⁵ z_i is XOR-ed to both the LFSR and NFSR update functions, *i.e.*, during one clock the LFSR and the NFSR bits are updated as $y_{i+n} = z_i + f(Y_i)$, $x_{i+n} = y_i + z_i + g(X_i)$.

⁵ during one clock

Pseudorandom Keystream Generation Algorithm (PRGA). After performing KSA routine for $2n$ clocks, z_i is no longer XOR-ed to the LFSR and NFSR update functions, but it is used as the output keystream bit. During this phase, the LFSR and NFSR are updated as $y_{i+n} = f(Y_i)$, $x_{i+n} = y_i + g(X_i)$.

Figure 1 depicts an overview of KSA and PRGA. Common features are depicted in black. In the case of Grain v1, the pseudorandom keystream generation algorithm does not include the green path. The red paths correspond to the key scheduling algorithm.

The corresponding parameters of Grain v1 are described in Appendix A, while Grain-128 is tackled in Appendix B and Grain-128a in Appendix C. The appendices also include the Load_{IV} functions and the KSA^{-1} algorithms for all versions.

Security Model. In the *Chosen IV - Related Key* setting (according to [6, Section 2.1]), an adversary is able to query an encryption oracle (which has access to the key K) in order to obtain valid ciphertexts. For each query i , the adversary can choose the oracle’s parameters: an initialization vector IV_i , a function $\mathcal{F}_i : \{0,1\}^n \rightarrow \{0,1\}^n$ and a message m_i . The oracle encrypts m_i using the Key-IV pair $(\mathcal{F}_i(K), IV_i)$. The adversary’s task is to distinguish the keystream output from a random stream.

Assumptions. Based on the results of the experiments we conducted, we further assume that the output of KSA, KSA^{-1} and PRGA is independently and uniformly distributed. More precisely, all previous algorithms were statistically tested applying the NIST Test Suite [2]. During our experiments we used the following setup:

1. X_i is a randomly generated n -bit state using the GMP library [3];
2. Y_i'' is either $0^{2\alpha}$ or $1^{2\alpha}$;
3. $Y_i = Y_i' || Y_i''$, where Y_i' is a randomly generated $(m - \alpha)$ -bit state using the GMP library.

3 Generic Grain Attacks

As already mentioned in Section 2, the Grain family uses an NFSR and a nonlinear filter (which takes input from both shift registers) to introduce nonlinearity. If after the initialization process, the LFSR is in an all zero state, only the NFSR is actively participating to the output. As already shown in the literature, NFSRs are vulnerable to distinguishing attacks [7, 15, 20].

Weak Key-IV pair. If the LFSR reaches the all zero state after $2n$ clocks we say that the pair (K, IV) is a *weak Key-IV pair*. An algorithm which produces weak Key-IV pairs for Grain v1 is presented in [20]. We refer the reader to Algorithm 1 for a generalization of this algorithm to any of the Grain ciphers.

Given a state V , we define it as *valid* if there exists an $IV \in \{0,1\}^m$ such that $\text{Load}_{IV}(IV, P) = V$, where P is the fixed padding. We further use a function $\text{Extract}_{IV}(V)$ which is the inverse of $\text{Load}_{IV}(\cdot, P)$. The probability to obtain a weak Key-IV pair by running Algorithm 1 is $1/2^\alpha$.

A refined version of the attack from [20] is discussed in [5] and generalized in Algorithm 2. The authors of [5] give precise differences between keystreams generated using the output of Algorithm 2 for Grain v1 (see Theorem 1), Grain-128 (see Theorem 2) and Grain-128a (see Theorem 3).

Algorithm 1. Generic Weak Key-IV Attack

Output: A Key-IV pair (K', IV')

```
1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n$  and let  $V \in \{0, 1\}^n$  be the zero LFSR state  $(0, \dots, 0)$ 
4   Run  $\text{KSA}^{-1}(K \| V)$  routine for  $2n$  clocks and produce state  $S' = K' \| V'$ 
5   if  $V'$  is valid then
6     Set  $s \leftarrow 1$  and  $IV' \leftarrow \text{Extract}_{IV}(V')$ 
7     return  $(K', IV')$ 
8   end
9 end
```

Theorem 1. For Grain v1, two initial states S_0 and $S_{0,\Delta}$ which differ only in the 79th position of the LFSR, produce identical output bits in 75 specific positions among the initial 96 key stream bits obtained during the PRGA.

Remark 1. More precisely, the 75 positions are the following ones:

$$k \in [0, 95] \setminus \{15, 33, 44, 51, 54, 57, 62, 69, 72, 73, 75, 76, 80, 82, 83, 87, 90, 91, 93 - 95\}.$$

Theorem 2. For Grain-128, two initial states S_0 and $S_{0,\Delta}$ which differ only in the 127th position of the LFSR, produce identical output bits in 112 specific positions among the initial 160 key stream bits obtained during the PRGA.

Remark 2. More precisely, the 112 positions are the following ones:

$$k \in [0, 159] \setminus \{32, 34, 48, 64, 66, 67, 79 - 81, 85, 90, 92, 95, 96, 98, 99, 106, 107, 112, 114, 117, 119, 122, 124 - 126, 128, 130 - 132, 138, 139, 142 - 146, 148 - 151, 153 - 159\}.$$

Theorem 3. For Grain-128a, two initial states S_0 and $S_{0,\Delta}$ which differ only in the 127th position of the LFSR, produce identical output bits in 115 specific positions among the initial 160 key stream bits obtained during the PRGA.

Remark 3. More precisely, the 115 positions are the following ones:

$$k \in [0, 159] \setminus \{33, 34, 48, 65 - 67, 80, 81, 85, 91, 92, 95, 97 - 99, 106, 107, 112, 114, 117, 119, 123 - 125, 127 - 132, 138, 139, 142 - 146, 149 - 151, 154 - 157, 159\}.$$

We further present an algorithm that checks which keystream positions produced by the states S and S_Δ are identical (introduced in Algorithm 2). Note that if we run Algorithm 3 we obtain less positions than claimed in Theorems 1 to 3, as shown in Appendix E. This is due to the fact that Algorithm 3 is prone to producing internal collisions and, thus, eliminate certain positions that are identical in both keystreams. Note that Theorem 4 is a refined version of Remarks 1 to 3 in the sense that it represents an automatic tool for finding identical keystream positions.

Modified Pseudorandom Keystream Generation Algorithm (PRGA'). To obtain our modified PRGA we replace + (XOR) and \cdot (AND) operations in the original PRGA with $|$ (OR) operations.

Algorithm 2. Search for Key-IV pairs that produce almost similar initial keystream

Input: An integer $r \in \{0, 2n\}$
Output: Key-IV pairs (K, IV) and (K', IV')

```
1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n$  and  $IV \in_R \{0, 1\}^m$ 
4   Run  $\text{KSA}(K\|IV)$  routine for  $2n$  clocks to obtain an initial state  $S_0 \in \{0, 1\}^{2n}$ 
5   Construct  $S_{0,\Delta}$  from  $S_0$  by flipping the bit on position  $r$ 
6   Run  $\text{KSA}^{-1}(S_{0,\Delta})$  routine for  $2n$  clocks and produce state  $S' = K'\|V'$ 
7   if  $V'$  is valid then
8     Set  $s \leftarrow 1$  and  $IV' \leftarrow \text{Extract}_{IV}(V')$ 
9     return  $(K, IV)$  and  $(K', IV')$ 
10  end
11 end
```

Theorem 4. Let r be a position of Grain’s internal state, q_1 the number of desired identical positions in the keystream and q_2 the maximum number of search trials. Then, Algorithm 3 finds at most q_1 identical positions in a maximum of q_2 trials.

Proof. We note that in Algorithm 3 the bit b_r on position r is set. If b_r is taken into consideration while computing the output bit of PRGA then the output of PRGA’ is also set due to the replacement of the original operations ($+$ and \cdot) with $|$ operations. The same argument is valid if a bit of Grain’s internal state is influenced by b_r .

The above statements remain true for each internal state bit that becomes set during the execution of Algorithm 3. \square

Algorithm 3. Search for identical keystream position in Grain

Input: Integers $r \in \{0, 2n\}$ and $q_1, q_2 > 0$
Output: Keystream positions φ

```
1 Set  $s \leftarrow 0$  and  $\varphi \leftarrow \emptyset$ 
2 Let  $S \in \{0, 1\}^{2n}$  be the zero state  $(0, \dots, 0)$ 
3 Construct  $S_\Delta$  from  $S$  by flipping the bit on position  $r$ 
4 while  $|\varphi| \leq q_1$  and  $s < q_2$  do
5   Set  $b \leftarrow \text{PRGA}'(S_\Delta)$  and update state  $S_\Delta$  with the current state
6   if  $b = 0$  then
7     Update  $\varphi \leftarrow \varphi \cup \{s\}$ 
8   end
9   Set  $s \leftarrow s + 1$ 
10 end
11 return  $\varphi$ 
```

4 Proposed Ideas

4.1 Compact Padding

Attacks that exploit the periodic padding used in Grain-128 where first presented in [8, 16] and further improved in [5]. We generalize and simplify these attacks below.

Setup. Let $Y_1 = [y_0, \dots, y_{d_1-1}]$, where $|Y_1| = d_1$, let $Y_2 = [y_{d_1+\alpha}, \dots, y_{n-1}]$, where $|Y_2| = d_2$ and let $IV = Y_1 \| Y_2$. We define

$$\text{Load}_{IV}(IV, P) = Y_1 \| P \| Y_2.$$

Let $S = [s_0, \dots, s_{n-1}]$ be a state of the LFSR, then we define

$$\text{Extract}_{IV}(S) = s_0 \| \dots \| s_{d_1-1} \| \dots \| s_{d_1+\alpha} \| \dots \| s_{n-1}.$$

Padding. Let $\alpha = \lambda\omega$ and $|P_0| = \dots = |P_{\omega-1}| = \lambda$, then we define $P = P_0 \| \dots \| P_{\omega-1}$. We say that P is a *periodic padding of order λ* if λ is the smallest integer such that $P_0 = \dots = P_{\omega-1}$.

Periodic padding of order α is further referred to as *aperiodic padding*.

Theorem 5. *Let P be a periodic padding of order λ and let $i = 1, 2$ denote an index. For each (set of) condition(s) presented in Column 2 of Table 1 there exists an attack whose corresponding success probability is presented in Column 3 of Table 1.*

	Conditions	Success Probability
1.	$d_1 \geq \lambda$ or $d_2 \geq \lambda$	$1/2^\lambda$
2.	$d_1 \geq \lambda$ and $d_2 \geq \lambda$	$1/2^{\lambda-1}$
3.	$d_i < \lambda$	$1/2^{2\lambda-d_i}$

Table 1: Attack Parameters for Theorem 5

Proof. 1. The proof follows directly from Algorithms 5 and 7. Given the assumptions in Section 2, the probability that the first λ keystream bits are zero is $1/2^\lambda$.

2. The proof is a direct consequence of Item 1.

3. The proof is straightforward in the light of Algorithms 8 and 9. Given the assumptions in Section 2, the probability that $V'_1 = P_0$ is $1/2^{\lambda-d_1}$ and the probability that $V'_2 = P_{\omega-1}$ is $1/2^{\lambda-d_2}$. Also, the probability that the first λ keystream bits are zero is $1/2^\lambda$. Since the two events are independent, we obtain the desired success probability.

Algorithm 4. $\text{Pair}_1(\sigma, S)$

Input: Number of clocks σ and a state S .

Output: A Key-IV pair (K', IV') or \perp

- 1 Run $\text{KSA}^{-1}(S)$ routine for σ clocks and produce state $S' = (K' \| V'_1 \| P \| P_{\omega-1} \| V'_2)$, where $|V'_1| = d_1$ and $|V'_2| = d_2 - \lambda$
 - 2 Set $IV' \leftarrow V'_1 \| P_{\omega-1} \| V'_2$
 - 3 **if** (K', IV') produces all zero keystream bits in the first λ PRGA rounds **then**
 - 4 **return** (K', IV')
 - 5 **end**
 - 6 **return** \perp
-

□

Algorithm 5. Constructing Key-IV pairs that generate λ bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

```

1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n, V_1 \in_R \{0, 1\}^{d_1-\lambda}$  and  $V_2 \in_R \{0, 1\}^{d_2}$ 
4   Set  $IV \leftarrow V_1 \| P_0 \| V_2, S \leftarrow K \| V_1 \| P_0 \| P \| V_2$  and  $output \leftarrow \text{Pair}_1(\lambda, S)$ 
5   if  $output \neq \perp$  then
6     Set  $s \leftarrow 1$ 
7     return  $(K, IV)$  and  $output$ 
8   end
9 end

```

Algorithm 6. $\text{Pair}_2(\sigma, S)$

Input: Number of clocks σ and a state S .
Output: A Key-IV pair (K', IV') .

```

1 Run  $\text{KSA}(S)$  routine for  $\sigma$  clocks and produce state  $S' = (K' \| V_1' \| P_0 \| P \| V_2')$ , where  $|V_1'| = d_1 - \lambda$  and  $|V_2'| = d_2$ 
2 Set  $IV' \leftarrow V_1' \| P_0 \| V_2'$ 
3 return  $(K', IV')$ 

```

Remark 4. Let $d_2 = 0, \lambda = 1, P_0 = 1$. If $\alpha = 16$, then the attack described in [16] is the same as the attack we detail in Algorithm 9. The same is true for [8] if $\alpha = 32$. Also, if $\alpha = 32$ then Algorithm 5 is a simplified version of the attack presented in [5].

Remark 5. To minimize the impact of Theorem 5, one must choose a padding value such that $\lambda = \alpha$ and either $d_1 < \alpha$ or $d_2 < \alpha$. In this case, because of the generic attacks described in Section 3, the success probability can not drop below $1/2^\alpha$. The designers of Grain-128a have chosen $d_2 = 0$ and $P = 0\text{xffffffe}$. In [6], the authors introduce an attack for Grain-128a, which is a special case of the attack we detail in Algorithm 5.

Theorem 6. Let P be an aperiodic padding, $1 \leq \gamma < \alpha/2$ and $d_2 < \alpha$. Also, let $i = 1, 2$ denote an index. If $\text{LSB}_\gamma(P) = \text{MSB}_\gamma(P)$, then for each condition presented in Column 2 of Table 2 there exists an attack whose corresponding success probability is presented in Column 3 of Table 2.

	Condition	Success Probability
1.	$d_i \geq \alpha - \gamma$	$1/2^{\alpha-\gamma}$
2.	$d_i < \alpha - \gamma$	$1/2^{2\alpha-2\gamma-d_i}$

Table 2: Attack Parameters for Theorem 6

Proof. 1. The first part of proof follows from Algorithm 5 with the following changes:

- (a) λ is replaced by $\alpha - \gamma$;
- (b) P_0 is replaced by $\text{MSB}_{\alpha-\gamma}(P)$;
- (c) $P_{\omega-1}$ is replaced by $\text{LSB}_{\alpha-\gamma}(P)$.

Therefore, the probability that the first $\alpha - \gamma$ keystream bits are zero is $1/2^{\alpha-\gamma}$. Similarly, the second part follows from Algorithm 7.

- 2. To prove the first part, we use the above changes on Algorithm 8, except that instead of replacing $P_{\omega-1}$ we replace $\text{LSB}_{d_1}(P_0)$ with $\text{MID}_{[\gamma+d_1-1, \gamma]}(P)$. Thus, we obtain the probability $1/2^{\alpha-\gamma}$. Similarly, for the second part we use Algorithm 9.

Algorithm 7. Constructing Key-IV pairs that generate λ bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

```
1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n, V_1 \in_R \{0, 1\}^{d_1}$  and  $V_2 \in_R \{0, 1\}^{d_2 - \lambda}$ 
4   Set  $IV \leftarrow V_1 \| P_{\omega-1} \| V_2$ 
5   if  $(K, IV)$  produces all zero keystream bits in the first  $\lambda$  PRGA rounds then
6     Set  $s \leftarrow 1$  and  $S \leftarrow (K \| V_1 \| P \| P_{\omega-1} \| V_2)$ 
7     return  $(K, IV)$  and  $\text{Pair}_2(\lambda, S)$ 
8   end
9 end
```

Algorithm 8. Constructing Key-IV pairs that generate λ bit shifted keystream

Output: Key-IV pairs (K'', IV'') and (K, IV)

```
1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n$  and  $V_2 \in_R \{0, 1\}^{d_2}$ 
4   Set  $IV \leftarrow \text{LSB}_{d_1}(P_0) \| V_2$ 
5   Run  $\text{KSA}^{-1}(K \| \text{LSB}_{d_1}(P_0) \| P \| V_2)$  routine for  $\lambda - d_1$  clocks and produce state  $S' = (K' \| V'_1 \| P \| V'_2)$ ,
   where  $|V'_1| = \lambda$  and  $|V'_2| = d_2 - \lambda + d_1$ 
6   if  $V'_1 = p_0$  then
7     Set  $S \leftarrow K' \| P_0 \| P \| V'_2$  and  $\text{output} \leftarrow \text{Pair}_1(d_1, S)$ 
8     if  $\text{output} \neq \perp$  then
9       Set  $s \leftarrow 1$ 
10      return  $(K, IV)$  and  $\text{output}$ 
11    end
12  end
13 end
```

Remark 6. To prevent the attacks presented in the proof of Theorem 6, the padding must be chosen such that $\text{MSB}_\gamma(P) \neq \text{LSB}_\gamma(P), \forall 1 \leq \gamma < \alpha/2$. Grain 128a uses such a padding $P = 0\text{xfffffff}e$. Another example was suggested in [8] to counter their proposed attacks: $P = 0\text{x}00000001$.

Constraints. Taking into account all the previous remarks, we may conclude that *good*⁶ compact padding schemes are aperiodic and, in particular, satisfy $\text{MSB}_\gamma(P) \neq \text{LSB}_\gamma(P), \forall 1 \leq \gamma < \alpha/2$. Also, another constraint is the position of the padding, *i.e.* $d_1 < \alpha$ or $d_2 < \alpha$ must be satisfied.

Remark 7. In the compact padding case, the number of padding schemes that verify the security restrictions represent 26% of the total 2^α . The previous percentage and the values we mention below were determined experimentally.

For $\alpha = 16$ and $0 \leq d_1, d_2 < 16$ we obtain $17622 \simeq 2^{14}$ compact padding schemes resistant to previous attacks. Thus, the complexity of a brute-force attack increases with 2^{19} .

For $\alpha = 32$ and $0 \leq d_1, d_2 < 32$ we obtain $1150153322 \simeq 2^{30}$ compact padding schemes resistant to previous attacks. Thus, the complexity of a brute-force attack increases with 2^{36} .

⁶ resistant to the aforementioned attacks

Algorithm 9. Constructing Key-IV pairs that generate λ bit shifted keystream

Output: Key-IV pairs (K'', IV'') and (K, IV)

```

1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n$  and  $V_1 \in_R \{0, 1\}^{d_1}$ 
4   Set  $IV \leftarrow V_1 \| MSB_{d_2}(P_{\omega-1})$ 
5   if  $K, IV$  produces all zero keystream bits in the first  $\lambda$  PRGA rounds then
6     Run  $KSA(K \| V_1 \| P \| MSB_{d_2}(P_{\omega-1}))$  routine for  $\lambda - d_2$  clocks and produce state  $S' = (K' \| V_1' \| P \| V_2')$ ,
       where  $|V_1'| = d_1 - \lambda + d_2$  and  $|V_2'| = \lambda$ 
7     if  $V_2' = P_{\omega-1}$  then
8       Set  $s \leftarrow 1$  and  $S \leftarrow (K' \| V_1' \| P \| P_{\omega-1})$ 
9       return  $(K, IV)$  and  $\text{Pair}_2(d_2, S)$ 
10    end
11  end
12 end

```

4.2 Fragmented Padding

Setup. Let $\alpha = c \cdot \beta$, where $c > 1$. Also, let $IV = B_0 \| B_1 \| \dots \| B_c$ and $P = P_0 \| P_1 \| \dots \| P_{c-1}$, where $|B_0| = d_1, |P_0| = \dots = |P_{c-1}| = |B_1| = \dots = |B_{c-1}| = \beta$ and $|B_c| = d_2$. In this case, we define

$$\text{Load}_{IV}(IV, P) = B_0 \| P_0 \| B_1 \| P_1 \| \dots \| B_{c-1} \| P_{c-1} \| B_c.$$

Let $S = S_0 \| \dots \| S_{2c}$ be a state of the LFSR, such that $|S_0| = d_1, |S_1| = \dots = |S_{2c-1}| = \beta$ and $|S_{2c}| = d_2$. Then we define

$$\text{Extract}_{IV}(S) = S_0 \| S_2 \| \dots \| S_{2c}.$$

Theorem 7. Let $i = 1, 2$ denote an index. In the previously mentioned setting, for each (set of) condition(s) presented in Column 2 of Table 3 there exists an attack whose corresponding success probability is presented in Column 3 of Table 3.

	Conditions	Success Probability
1.	$d_1 \geq \beta$ or $d_2 \geq \beta$	$1/2^\beta$
2.	$d_1 \geq \beta$ and $d_2 \geq \beta$	$1/2^{\beta-1}$
3.	$d_i < \beta$	$1/2^{\beta-d_i}$

Table 3: Attack Parameters for Theorem 7

Proof. 1. We only prove the case $i = 1$ as the case $i = 2$ is similar in the light of Algorithm 7. The proof follows directly from Algorithm 12. Given the assumptions in Section 2, the probability that the first β keystream bits are zero is $1/2^\beta$.

2. The proof is a direct consequence of Item 1.

3. Again, we only prove the case $i = 1$. The proof is straightforward in the light of Algorithm 16. Given the assumptions in Section 2, the probability that $V_1' = P_0$ is $1/2^{\beta-d_1}$. Also, the probability that the first β keystream bits are zero is $1/2^\beta$. Since the two events are independent, we obtain the desired success probability.

Algorithm 10. $\text{Update}_1()$

Output: Variable $value$

- 1 Set $value \leftarrow P_0$
- 2 **for** $i = 1$ to $c - 1$ **do**
- 3 Update $value \leftarrow value \parallel P_i \parallel P_i$
- 4 **end**
- 5 **return** $value$

Algorithm 11. $\text{Pair}_3(\sigma, S)$

Input: Number of clocks σ and a state S .

Output: A Key-IV pair (K', IV') or \perp

- 1 Run $\text{KSA}^{-1}(S)$ routine for σ clocks and produce state $S' = (K' \parallel V_1' \parallel value \parallel V_2')$, where $|V_1'| = d_1$ and $|V_2'| = d_2 - \beta$
- 2 Set $IV' \leftarrow V_1' \parallel P \parallel V_2'$
- 3 **if** (K', IV') produces all zero keystream bits in the first β PRGA rounds **then**
- 4 **return** (K', IV')
- 5 **end**
- 6 **return** \perp

Algorithm 12. Constructing Key-IV pairs that generate β bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

- 1 Set $s \leftarrow 0$
- 2 **while** $s = 0$ **do**
- 3 Choose $K \in_R \{0, 1\}^n$, $V_1 \in_R \{0, 1\}^{d_1 - \beta}$ and $V_2 \in_R \{0, 1\}^{d_2}$
- 4 Set $value \leftarrow P_0 \parallel \text{Update}_1()$, $IV \leftarrow V_1 \parallel P \parallel V_2$, $S \leftarrow K \parallel V_1 \parallel value \parallel V_2$ and $output \leftarrow \text{Pair}_3(\beta, S)$
- 5 **if** $output \neq \perp$ **then**
- 6 Set $s \leftarrow 1$
- 7 **return** (K, IV) and $output$
- 8 **end**
- 9 **end**

□

Remark 8. Let $\delta < \beta$ and $\beta > 1$. To prevent the attacks presented in Theorem 7, we have to slightly modify the structure of the IV . We need at least one block $|B_i| = \delta$, where $1 \leq i \leq c - 1$. We further consider that $|B_i| = \delta, \forall 1 \leq i \leq c - 1$.

Theorem 8. Let $|B_i| = \delta, \forall 1 \leq i \leq c - 1$. Also, let $1 \leq \gamma \leq \beta, 1 \leq t \leq c$ and $0 \leq j \leq t - 1$. If $LSB_\gamma(P_{c-1-j}) = MSB_\gamma(P_{t-1-j}) \forall j$ then for each (set of) condition(s) presented in Column 2 of Table 4 there exists an attack whose corresponding success probability is presented in Column 3 of Table 4.

	Conditions	Success Probability
1.	$d_1 \geq \beta - \gamma + (\beta + \delta)(c - t), \delta \geq \beta - \gamma$	$1/2^{\beta - \gamma + (\beta + \delta)(c - t)}$
2.	$d_1 \geq \beta - \gamma + (\beta + \delta)(c - t), \delta < \beta - \gamma,$ $MSB_{\beta - \gamma - \delta}(P_{c-1-j}) = LSB_{\beta - \gamma - \delta}(P_{t-2-j}) \forall j$	$1/2^{\beta - \gamma + (\beta + \delta)(c - t)}$
3.	$d_1 < \beta - \gamma + (\beta + \delta)(c - t), \delta \geq \beta - \gamma$	$1/2^{2\beta - 2\gamma + 2(\beta + \delta)(c - t) - d_1}$
4.	$d_1 < \beta - \gamma + (\beta + \delta)(c - t), \delta < \beta - \gamma,$ $MSB_{\beta - \gamma - \delta}(P_{c-1-j}) = LSB_{\beta - \gamma - \delta}(P_{t-2-j}) \forall j$	$1/2^{2\beta - 2\gamma + 2(\beta + \delta)(c - t) - d_1}$

Table 4: Attack Parameters for Theorem 8

Proof. 1. The proof follows directly from Algorithm 19 (described in the last appendix of our paper). Given the assumptions in Section 2, the probability that the first $\beta - \gamma + (\beta + \delta)(c - t)$ keystream bits are zero is $1/2^{\beta - \gamma + (\beta + \delta)(c - t)}$.

The proofs for the remaining cases presented in Table 4 follow directly from previous results. Thus, we omit them. \square

Theorem 9. Let $|B_i| = \delta, \forall 1 \leq i \leq c - 1$. Also, let $1 \leq \gamma \leq \beta, 1 \leq t \leq c$ and $0 \leq j \leq t - 2$. If $\delta \geq \beta - \gamma$ then for each (set of) condition(s) presented in Column 2 of Table 5 there exists an attack whose corresponding success probability is presented in Column 3 of Table 5.

	Conditions	Success Probability
1.	$d_1 \geq \delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t),$ $MSB_{\gamma}(P_{c-1-j}) = LSB_{\gamma}(P_{t-2-j}) \forall j$	$1/2^{\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)}$
2.	$d_1 < \delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t),$ $MSB_{\gamma}(P_{c-1-j}) = LSB_{\gamma}(P_{t-2-j}) \forall j$	$1/2^{2\delta - 2\beta + 2\gamma + 2\beta(c - t + 1) + 2\delta(c - t) - d_1}$

Table 5: Attack Parameters for Theorem 9

Proof. 1. The proof follows directly from Algorithm 20 (described in the last appendix of our paper). Given the assumptions in Section 2, the probability that the first $\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)$ keystream bits are zero is $1/2^{\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)}$.

2. The proof is similar to the proof of Theorem 7, Item 3. \square

Remark 9. Taking into account the generic attacks described in Section 3, any probability bigger than $1/2^\alpha$ is superfluous. As an example, when $\alpha = 32$ we obtain a good padding scheme for the following parameters $d_2 = 0, \beta = 16, \delta = 14, P_0 = 0x8000, P_1 = 0x7fff$.

Remark 10. Let $c = 2, \delta \leq \beta - 2, \gamma < \beta$ and $P_0 \neq P_1$. The best success probability of a slide attack when the following conditions are met:

$$\begin{aligned} \gamma > 1 : \quad & LSB_\gamma(P_1) \neq MSB_\gamma(P_0) \\ & LSB_\gamma(P_0) \neq MSB_\gamma(P_1), \\ \gamma > 0 : \quad & LSB_\gamma(P_1) \neq MSB_\gamma(P_1) \\ & LSB_\gamma(P_0) \neq MSB_\gamma(P_0), \end{aligned}$$

is $1/2^{\alpha-1+\delta} \geq 1/2^\alpha$. The number of padding schemes that verify the security restrictions represent 2% of the total 2^α . The previous percentage and the values we mention below were determined experimentally.

For $\alpha = 16, \beta = 8, 1 \leq \delta \leq 6, \gamma < 8$ and $d_1 = d_2 = 0$ we obtain 1840 $\simeq 2^{10}$ fragmented padding schemes resistant to previous attacks. Thus, the complexity of a brute-force attack increases with 2^{14} .

For $\alpha = 32, \beta = 16, 1 \leq \delta \leq 14, \gamma < 16$ and $d_1 = d_2 = 0$ we obtain 117113488 $\simeq 2^{23}$ fragmented padding schemes resistant to previous attacks. Thus, the complexity of a brute-force attack increases with 2^{28} .

5 Conclusion

We analyzed the security of various periodic and aperiodic IV padding methods⁷ for the Grain family of stream ciphers, proposed corresponding attacks and discussed their success probability.

Future Work. A closely related study which naturally arises is analyzing the security of breaking the padding into aperiodic blocks. Another idea would be to study how the proposed padding techniques interfere with the security of the authentication feature of Grain-128a. A question that arises is if the occurrence of slide pairs may somehow be converted into a distinguishing or key recovery attack. Another interesting point would be to investigate what would happen to the security of the Grain family with respect to differential, linear or cube attacks in the various padding scenarios we outlined. One more future work idea could be to analyze various methods of preventing the all zero state of Grain's LFSR.

References

1. eSTREAM: the ECRYPT Stream Cipher Project. <http://www.ecrypt.eu.org/stream/>
2. NIST SP 800-22: Download Documentation and Software. <https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software>
3. The GNU Multiple Precision Arithmetic Library. <https://gmplib.org/>
4. Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. <https://eprint.iacr.org/2009/218.pdf> (2009)
5. Banik, S., Maitra, S., Sarkar, S.: Some Results on Related Key-IV Pairs of Grain. In: Proceedings of the 2nd International Conference on Security, Privacy, and Applied Cryptography Engineering – SPACE'12. Lecture Notes in Computer Science, vol. 7644, pp. 94–110. Springer (2012)
6. Banik, S., Maitra, S., Sarkar, S., Meltem Sönmez, T.: A Chosen IV Related Key Attack on Grain-128a. In: Proceedings of the 18th Australasian Conference on Information Security and Privacy – ACISP'13. Lecture Notes in Computer Science, vol. 7959, pp. 13–26. Springer (2013)

⁷ compact and fragmented

7. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of Grain. In: Proceedings of the 18th International Workshop on Fast Software Encryption – FSE’06. Lecture Notes in Computer Science, vol. 4047, pp. 15–29. Springer (2006)
8. Cannière, C., Küçük, Ö., Preneel, B.: Analysis of Grain’s Initialization Algorithm. In: Progress in Cryptology – AFRICACRYPT’08. Lecture Notes in Computer Science, vol. 5023, pp. 276–289. Springer (2008)
9. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware. In: Advances in Cryptology – ASIACRYPT’11. Lecture Notes in Computer Science, vol. 7073, pp. 327–343. Springer (2011)
10. Dinur, I., Shamir, A.: Breaking Grain-128 with Dynamic Cube Attacks. In: Proceedings of the 18th International Workshop on Fast Software Encryption – FSE’11. Lecture Notes in Computer Science, vol. 6733, pp. 167–187. Springer (2011)
11. Hell, M., Johansson, T., Maximov, A., Meier, W.: A Stream Cipher Proposal: Grain-128. In: International Symposium on Information Theory – ISIT’06. pp. 1614–1618. IEEE (July 2006)
12. Hell, M., Johansson, T., Meier, W.: Grain - A Stream Cipher for Constrained Environments. Tech. Rep. 010 (2005), eCRYPT Stream Cipher Project Report
13. Hell, M., Johansson, T., Meier, W.: Grain: A Stream Cipher for Constrained Environments. International Journal of Wireless and Mobile Computing **2**(1), 86–93 (May 2007)
14. Khazaei, S., Hassanzadeh, M., Kiaei, M.: Distinguishing Attack on Grain. Tech. Rep. 071 (2005), eCRYPT Stream Cipher Project Report
15. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential cryptanalysis of NLFSR-Based Cryptosystems. In: Advances in Cryptology – ASIACRYPT’10. Lecture Notes in Computer Science, vol. 6477, pp. 130–145. Springer (2010)
16. Küçük, Ö.: Slide Resynchronization Attack on the Initialization of Grain 1.0. <http://www.ecrypt.eu.org/stream> (2006)
17. Maimut, D.: Authentication and Encryption Protocols: Design, Attacks and Algorithmic Improvements. Ph.D. thesis, École normale supérieure (2015)
18. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: A New Version of Grain-128 with Optional Authentication. International Journal of Wireless and Mobile Computing **5**(1), 48–59 (Dec 2011)
19. Stankovski, P.: Greedy Distinguishers and Nonrandomness Detectors. In: Progress in Cryptology – INDOCRYPT’10. Lecture Notes in Computer Science, vol. 6498, pp. 210–226. Springer (2010)
20. Zhang, H., Wang, X.: Cryptanalysis of Stream Cipher Grain Family. <https://eprint.iacr.org/2009/109.pdf> (2009)

A Grain v1

In the case of Grain v1, $n = 80$ and $m = 64$. The padding value is $P = 0\text{xffff}$. The values IV and P are loaded in the LFSR using the function $LoadIV(IV, P) = IV \parallel P$. Given $S \in \{0, 1\}^{80}$, we define $ExtractIV(S) = MSB_{64}(S)$.

We denote by $f_1(x)$ the primitive feedback of the LFSR:

$$f_1(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}.$$

We denote by $g_1(x)$ the nonlinear feedback polynomial of the NFSR:

$$\begin{aligned} g_1(x) = & 1 + x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + x^{71} + x^{80} \\ & + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} \\ & + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} \\ & + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}. \end{aligned}$$

The boolean filter function $h_1(x_0, \dots, x_4)$ is

$$h_1(x_0, \dots, x_4) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4.$$

The output function is

$$z_i^1 = \sum_{j \in \mathcal{A}_1} x_{i+j} + h_1(y_{i+3}, y_{i+25}, y_{i+46}, y_{i+64}, x_{i+63}), \text{ where } \mathcal{A}_1 = \{1, 2, 4, 10, 31, 43, 56\}.$$

Algorithm 13. KSA⁻¹ routine for Grain v1

Input: State $S_i = (x_0, \dots, x_{79}, y_0, \dots, y_{79})$

Output: The preceding state $S_{i-1} = (x_0, \dots, x_{79}, y_0, \dots, y_{79})$

1 $v = y_{79}$ and $w = x_{79}$

2 **for** $t = 79$ to 1 **do**

3 $y_t = y_{t-1}$ and $x_t = x_{t-1}$

4 **end**

5 $z = \sum_{j \in \mathcal{A}_1} x_j + h_1(y_3, y_{25}, y_{46}, y_{64}, x_{63})$

6 $y_0 = z + v + y_{13} + y_{23} + y_{38} + y_{51} + y_{62}$

7 $x_0 = z + w + y_0 + x_9 + x_{14} + x_{21} + x_{28} + x_{33} + x_{37} + x_{45} + x_{52} + x_{60} + x_{62} + x_{63}x_{60} + x_{37}x_{33} + x_{15}x_9 +$
 $x_{60}x_{52}x_{45} + x_{33}x_{28}x_{21} + x_{63}x_{45}x_{28}x_9 + x_{60}x_{52}x_{37}x_{33} + x_{63}x_{60}x_{21}x_{15} + x_{63}x_{60}x_{52}x_{45}x_{37} + x_{33}x_{28}x_{21}x_{15}x_9 +$
 $x_{52}x_{45}x_{37}x_{33}x_{28}x_{21}$

B Grain-128

In the case of Grain-128, $n = 128$ and $m = 96$. The padding value is $P = 0\text{xffffffff}$. The values IV and P are loaded in the LFSR using the function $LoadIV(IV, P) = IV || P$. Given $S \in \{0, 1\}^{128}$, we define $ExtractIV(S) = MSB_{96}(S)$.

We denote by $f_{128}(x)$ the primitive feedback of the LFSR:

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

We denote by $g_{128}(x)$ the nonlinear feedback polynomial of the NFSR:

$$g_{128}(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} \\ + x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}.$$

The boolean filter function $h_{128}(x_0, \dots, x_8)$ is

$$h_{128}(x_0, \dots, x_8) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8.$$

The output function is

$$z_i^{128} = \sum_{j \in \mathcal{A}_{128}} x_{i+j} + y_{i+93} + h_{128}(x_{i+12}, y_{i+8}, y_{i+13}, y_{i+20}, x_{i+95}, y_{i+42}, y_{i+60}, y_{i+79}, y_{i+95}),$$

where $\mathcal{A}_{128} = \{2, 15, 36, 45, 64, 73, 89\}$.

Algorithm 14. KSA^{-1} routine for Grain-128

Input: State $S_i = (x_0, \dots, x_{127}, y_0, \dots, y_{127})$

Output: The preceding state $S_{i-1} = (x_0, \dots, x_{127}, y_0, \dots, y_{127})$

1 $v = y_{127}$ and $w = x_{127}$

2 **for** $t = 127$ to 1 **do**

3 $y_t = y_{t-1}$ and $x_t = x_{t-1}$

4 **end**

5 $z = \sum_{j \in \mathcal{A}_{128}} x_{i+j} + y_{93} + h_{128}(x_{12}, y_8, y_{13}, y_{20}, x_{95}, y_{42}, y_{60}, y_{79}, y_{95}),$

6 $y_0 = z + v + y_7 + y_{38} + y_{70} + y_{81} + y_{96}$

7 $x_0 = z + w + y_0 + x_{26} + x_{56} + x_{91} + x_{96} + x_{84}x_{68} + x_{65}x_{61} + x_{48}x_{40} + x_{59}x_{27} + x_{18}x_{17} + x_{13}x_{11} + x_{67}x_3$

C Grain-128a

In the case of Grain-128a, $n = 128$ and $m = 96$. The padding value is $P = 0\text{xfffffff}$. The values IV and P are loaded in the LFSR using the function $LoadIV(IV, P) = IV \parallel P$. Given $S \in \{0, 1\}^{128}$, we define $ExtractIV(S) = MSB_{96}(S)$.

We denote by $f_{128a}(x)$ the primitive feedback of the LFSR:

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

We denote by $g_{128a}(x)$ the nonlinear feedback polynomial of the NFSR:

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

The boolean filter function $h_{128a}(x_0, \dots, x_8)$ is

$$h_{128a}(x_0, \dots, x_8) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8.$$

The output function is

$$z_i^{128a} = \sum_{j \in \mathcal{A}_{128a}} x_{i+j} + y_{i+93} + h_{128a}(x_{i+12}, y_{i+8}, y_{i+13}, y_{i+20}, x_{i+95}, y_{i+42}, y_{i+60}, y_{i+79}, y_{i+94}),$$

where $\mathcal{A}_{128a} = \{2, 15, 36, 45, 64, 73, 89\}$.

Algorithm 15. KSA^{-1} routine for Grain-128a

Input: State $S_i = (x_0, \dots, x_{127}, y_0, \dots, y_{127})$

Output: The preceding state $S_{i-1} = (x_0, \dots, x_{127}, y_0, \dots, y_{127})$

- 1 $v = y_{127}$ and $w = x_{127}$
 - 2 **for** $t = 127$ to 1 **do**
 - 3 $y_t = y_{t-1}$ and $x_t = x_{t-1}$
 - 4 **end**
 - 5 $z = \sum_{j \in \mathcal{A}_{128a}} x_j + y_{93} + h_{128a}(x_{12}, y_8, y_{13}, y_{20}, x_{95}, y_{42}, y_{60}, y_{79}, y_{94})$
 - 6 $y_0 = z + v + y_7 + y_{38} + y_{70} + y_{81} + y_{96}$
 - 7 $x_0 = z + w + y_0 + x_{26} + x_{56} + x_{91} + x_{96} + x_3x_{67} + x_{11}x_{13} + x_{17}x_{18} + x_{27}x_{59} + x_{40}x_{48} + x_{61}x_{65} + x_{68}x_{84} + \\ x_{88}x_{92}x_{93}x_{95} + x_{22}x_{24}x_{25} + x_{70}x_{78}x_{82}$
-

D Examples

Within Tables 6 to 8, the padding is written in blue, while the red text denotes additional data necessary to mount the proposed attacks. Test vectors presented in this section are expressed as hexadecimal strings. For simplicity, we omit the 0x prefix.

Table 6: Examples of Generic Attacks.

	Cipher	Key	LFSR Loading
Algorithm 1	Grain v1	a8af910f2755c064d713	1c60b94e09512adbffff
	Grain 128	525c3676953eccec2bc5388f1474cdc61	b78d3637b64425015fa3ef63fffffffff
	Grain 128a	a04f944e6ca1e1406537a0ef215689a3	aaaebb010224478f48567997ffffffffe

Table 7: Examples of Compact Padding Attacks (index $i = 1$).

	Cipher	Key	LFSR Loading	Keystream
Theorem 5 Condition 1 (Algorithm 5)	Grain v1	7e72b6f960cf9165b891	1007bc3594e07f7f7fa5	004e2da99a27392383696e9e7120370a
		72b6f960cf9165b89145	07bc3594e07f7f7fa580	4e2da99a27392383696e9e7120370a48
	Grain-128	00166499157d39c9 5a723b601eccfffb	4a9a37ef1e3dfc13 7fff7fff7fffeb05	000076755ac4cd53028caa577964929e
		6499157d39c95a72 3b601eccfffb2fd1	37ef1e3dfc137fff 7fff7fffeb05d636	76755ac4cd53028caa577964929ef1c0
	Grain-128a	b9e20a7619a8d622 5152cfa83eb73361	ef53aafa3c6c47ca 7fff7fff7fff5cd	0000bac1203a11b554d69fd7f9f27b7f
		0a7619a8d6225152 cfa83eb7336175a5	aafa3c6c47ca7fff 7fff7fff5cd98ba	bac1203a11b554d69fd7f9f27b7fd545
Theorem 5 Condition 3 (Algorithm 8)	Grain v1	455b5df993b367e37b60 5b5df993b367e37b604d	07f7f7fe9b4a3044efd1 f7f7fe9b4a3044efd139	0095e584ea234610f7ec250a948a8267 95e584ea234610f72ec250a948a8267c
		9302f6b9d7136599 ac1caee130c596bb	8d7fff7fff7fff10 d59595e5568beb11	00007ca563c6831b63868259f547cdff
	Grain-128	f6b9d7136599ac1c aee130c596bb0dc8	ff7fff7fff10d595 95e5568beb11628c	7ca563c6831b63868259f547cdff695b
		0f478aa147938251 5e0a94d3357764f4	cd7fff7fff7fffed bb0e00ddcb18d1eb	000059362a172d8748185e0850be7cb8
	Grain-128a	8aa1479382515e0a 94d3357764f4b8bb	ff7fff7fffedbb0e 00ddcb18d1eb0416	59362a172d8748185e0850be7cb824a0
		Grain v1	4febc079167f99bdb1db bc079167f99bdb1db338	bd4710804f9eff0ff0fa 710804f9eff0ff0fa272
Grain-128	5a0d4b3907f65ce5 f036b3671614244b		0bbd00872ecb0732 ffff00ffff00fffe	0000006b2014ecdee8d499646ba08a9f
	Grain-128a	3907f65ce5f036b3 671614244be57112	872ecb0732ffff00 ffff00fffeaf68a2	6b2014ecdee8d499646ba08a9fd93085
Grain-128a		6472c21093cd2225 4118e1a69230e0ac	2c9c47771ed4f648 ffff00ffff00ffde	0000009e196e7e866193867ea31b1df0
	Grain-128a	1093cd22254118e1 a69230e0ac668222	771ed4f648ffff00 ffff00ffdeb9f179	9e196e7e866193867ea31b1df09f306a
Theorem 6 Condition 2		Grain v1	701aa599737c957a0b5e aa599737c957a0b5eb77	07ff0ff0f0dedd9bd4d1b f0ff0f0dedd9bd4d1b1bf
	Grain-128		30bfe11f3b7080be 47396a37f889b57c	aafdffff00ffff00 ff38ff5b14da5371
		1f3b7080be47396a 37f889b57cac5367	ff00ffff00ff38ff 5b14da53715a4291	8a735f3adf71728258dcaf47fd6edad1
	Grain-128a	c4b8607e854abc5f 7a74eba33d563ad1	950bffff00ffff00 ff7182c277b77e8f	000000681060aa4bf10c0181bd7e4d95
		7e854abc5f7a74eb a33d563ad125aaff	ff00ffff00ff7182 c277b77e8f5db61f	681060aa4bf10c0181bd7e4d957b5f2e

Table 8: Examples of Fragmented Padding Attacks (index $i = 1$).

	Cipher	Key	LFSR Loading	Keystream	
Theorem 7 Condition 1 (Algorithm 12)	Grain v1	cc0d50254f72d88d3c71	3a86d17377777777b2c	04c79ebb4db7bc675644b3d0bf2a59a4	
		c0d50254f72d88d3c714	a86d17377777777b2cf	4c79ebb4db7bc675644b3d0bf2a59a47	
	Grain-128	c506d0ca5bff72e1 6ea07fd8f98d7ba3	63ba70cf067f7f7f 7f7f7f7f7f879f9b	004e2c99a48677b4c217f9e14e620d48	
		06d0ca5bff72e16e a07fd8f98d7ba368	ba70cf067f7f7f7f 7f7f7f7f879f9be1	4e2c99a48677b4c217f9e14e620d4884	
	Grain-128a	0948bd1a0a5d275c 54744db3dc27cec8	895ba804147f7f7f 7f7f7f7f2f9892	003a5f1e38d9c44670b0dc017377e698	
		48bd1a0a5d275c54 744db3dc27cec82b	5ba804147f7f7f7f 7f7f7f7f2f9892f1	3a5f1e38d9c44670b0dc017377e698d7	
Theorem 7 Condition 3 (Algorithm 16)	Grain v1	77a73157cabfa60349dc	7777777318f59ac6aff	0c61bfa06e1c22011dcefe673765acb7	
		7a73157cabfa60349dc3	7777777318f59ac6affd	c61bfa06e1c22011dcefe673765acb7f	
	Grain-128	9aca3bd2cf312080 769338bec86f9da6	7f7f7f7f7f7f7f7f b6f7e83b3793f746	004624d2271d3420104b2fd1058675fd	
		ca3bd2cf31208076 9338bec86f9da63f	7f7f7f7f7f7f7fb6 f7e83b3793f746ff	4624d2271d3420104b2fd1058675fd45	
	Grain-128a	0e9eb1a896077e93 5b21de8700f3ef44	7f7f7f7f7f7f7f7f 29b03ff3e82cda8b	007f06d63e3545f6b7c4b50d255b6663	
		9eb1a896077e935b 21de8700f3ef4462	7f7f7f7f7f7f7f29 b03ff3e82cda8bfc	7f06d63e3545f6b7c4b50d255b6663ea	
Theorem 8 Condition 1 (Algorithm 19)	Grain-128	d3ea84c99a8b1354 71d8c320b870e109	ed52bf1b25ff0ff0 fff0ff0f4ed8f575	0001590b803ff3c9972d96481a6e8ad4	
		a84c99a8b135471d 8c320b870e109120	2bf1b25ff0ff0fff 0ff0f4ed8f575dac	1590b803ff3c9972d96481a6e8ad48ee	
		9ee02802ccf920e6 868a8aa46113a406	ab24f8ab82ff0ff0 fff0ff0fd32dc4e9	00082e1cbbb25fa325518665a17f2efc	
	Grain-128a	02802ccf920e6868 a8aa46113a40681d	4f8ab82ff0ff0fff 0ff0fd32dc4e9473	82e1cbbb25fa325518665a17f2efc2eb	
		Grain-128	8d89931ae1e13215 77bba20640c193a1	f18ccfbf3cff0ff0 ff0ff0fde5af2b58	000e612c620ae1765ded57a835b713ac
			9931ae1e1321577b ba20640c193a13b8	ccfbf3cff0ff0ff0 ff0fde5af2b58811	e612c620ae1765ded57a835b713ace4a
Grain-128a	626262808f0ca24c cc517bb93fb5c3cb	c4ca6f9535ff0ff0 ff0ff0fde92e568	0003f5a6d1b7f615dfb32e34cea7cc4a		
	262808f0ca24ccc5 17bb93fb5c3cb22f	a6f9535ff0ff0ff0 ff0fde92e568a4f	3f5a6d1b7f615dfb32e34cea7cc4a106		
Theorem 8 Condition 3	Grain-128	416ddd14b4c096cb 0181ae8830ada69d	80ff0ff0ff0ff0ff d7ef096c7a8700a3	00076a8e9def620dfe704b264988da02	
		ddd14b4c096cb018 1ae8830ada69d3b6	f0ff0ff0ff0ff0d7e f096c7a8700a318f	76a8e9def620dfe704b264988da02cc0	
		724d58601b44396d 60e83723a65bfa7b	84ff0ff0ff0ff0ff 6c25a1d79af2a85c	0008ab9f20d8a418932150d3ba97400e	
	Grain-128a	d58601b44396d60e 83723a65bfa7b973	f0ff0ff0ff0ff0c2 5a1d79af2a85c626	8ab9f20d8a418932150d3ba97400ebd5	

Theorem 8 Condition 4	Grain 128	97516dced374a089 88ce86acaa2ff1a4	3aff0ff0ff0ff0f1 12b72427d44b92f1	000a8e820bedfb8cd9d651d8221f3b34
		16dced374a08988c e86acaa2ff1a4399	f0ff0ff0ff0f112b 72427d44b92f1bba	a8e820bedfb8cd9d651d8221f3b34846
	Grain-128a	a29ae6fb8b23f747 f3723e59df0d3a8e	4bff0ff0ff0ff0fc 92ace3a64691e733	000cd469723847db72f6f856e51f9d96
		ae6fb8b23f747f37 23e59df0d3a8eabb	f0ff0ff0ff0fc92a ce3a64691e733a54	cd469723847db72f6f856e51f9d96b38
Theorem 9 Condition 1 (Algorithm 20)	Grain-128	930cb0086c93293e 9722a710e28a1375	f767352c26395e8a ffffb0ffff80ffffb	0000000a44dcae9a68c7b66389e440eb
		086c93293e9722a7 10e28a1375ec5696	2c26395e8affffb0 ffff80ffffbb6fcf2	0a44dcae9a68c7b66389e440ebddf198
	Grain-128a	270f72277e7540cf 9a58fa4426e28aae	c7df3ee9c792f5d5 ffffd0ffff00fff1	000000fd8bbdb3d3a8c885704f43a022
		277e7540cf9a58fa 4426e28aaebc06e1	e9c792f5d5ffffd0 ffff00fff13204c5	fd8bbdb3d3a8c885704f43a022557a89
Theorem 9 Condition 2	Grain-128	895bea372ffe4e76 e84113dd18afa6b9	a8147ffff80fffffe 0fff2cd80e83e74	0000004b5394f9baf0f6a6ff3d921542
		372ffe4e76e84113 dd18afa6b9fb5cef	fff80ffff0fff2c d80e83e74e3d134e	4b5394f9baf0f6a6ff3d9215422cbdbb
	Grain-128a	70a2fecddbc94115 017b571df0854817	9e132ffff50ffffd 0fff5cf89b04484d	0000002839a6bec77a007d3d12b4d597
		cddbc94115017b57 1df08548178142d5	fff50ffffd0fff5c f89b04484d01fb4b	2839a6bec77a007d3d12b4d597c9041b

E Propagation of Single Bit Differentials

Parameters. In Theorem 4, let $q_2 = 96$ for Grain v1⁸ and $q_2 = 160$ for Grain-128 and Grain-128a⁹.

Table 9: Propagation of a Single Bit Differential in the case of Grain v1’s LFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
15	50	0-11, 13-17, 19-30, 33-35, 37, 38, 40-46, 48, 51, 53, 55, 58, 61-63, 71
31	59	0-5, 7-23, 25-27, 29-33, 35-41, 43-46, 49-51, 54, 56-59, 61, 62, 64, 67, 69, 74, 77, 79, 87
47	63	0, 2-21, 23, 24, 26-39, 41, 42, 45-49, 51-53, 55-57, 59, 60, 62, 65, 66, 70, 73-75, 77, 78, 80, 95
63	63	0-16, 18-27, 29-34, 36, 37, 39, 40, 42-45, 47-52, 54, 55, 58, 61-63, 65, 68, 69, 72, 73, 76, 81, 90, 91, 94
79	74	0-14, 16-32, 34-43, 45-50, 52, 53, 55, 56, 58-61, 63-68, 70, 71, 74, 77-79, 81, 84, 85, 88, 89, 92

⁸ as in Theorem 1

⁹ as in Theorem 2, respectively Theorem 3

Table 10: Propagation of a Single Bit Differential in the case of Grain v1's NFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
15	23	0-4, 6-10, 12, 15, 16, 19, 20-22, 26, 27, 28, 29, 31, 33
31	32	1-19, 22-26, 28, 31, 32, 35, 36, 42, 43, 49
47	32	0-15, 17, 18, 20-25, 28, 29, 30, 32, 33, 35, 40, 41, 42
63	25	1-6, 8-16, 19, 21-23, 26, 29-31, 33, 39
79	41	0-15, 17-22, 24-32, 35, 37-39, 42, 45-47, 49, 55

Table 11: Propagation of a Single Bit Differential in the case of Grain-128's LFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
31	92	0-10, 12-17, 19-22, 24-56, 58, 60-63, 65, 67-69, 71, 72, 74-79, 81-85, 87, 88, 90, 93, 94, 97, 100, 103, 109, 116, 119, 126, 129, 135, 141, 148
55	97	0-12, 14-34, 36-41, 43-46, 48, 49, 51, 53-65, 67-80, 86, 87, 89, 91-93, 95, 96, 100-102, 105-107, 109, 111, 112, 118, 121, 127, 133, 153, 159
79	101	1-18, 20-36, 38-41, 43, 45-57, 60-65, 67-70, 72, 73, 75, 78-88, 92-94, 96-99, 101, 103, 104, 110, 111, 113, 115, 119, 120, 125, 126, 130, 131, 133, 145, 151, 157
103	86	0-7, 9, 11-23, 25-39, 41, 44-54, 58-60, 62-65, 67, 69, 70, 73, 76-81, 84-86, 91, 92, 94, 96, 97, 99, 105, 109, 110-112, 116, 117, 123, 128, 143, 144
127	108	0-31, 33, 35-47, 49-63, 65, 68-78, 82-84, 86-89, 91, 93, 94, 97, 100-105, 108-110, 115, 116, 118, 120, 121, 123, 129, 133-136, 140, 141, 147, 152

Table 12: Propagation of a Single Bit Differential in the case of Grain-128's NFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
31	52	0-15, 17, 18, 20-28, 30-36, 39-42, 45, 48-50, 54-56, 58, 62, 63, 65, 66, 71, 72
55	65	0-9, 11-18, 20-39, 41, 42, 44, 45, 47, 49-52, 55-60, 63-66, 69, 73, 74, 82, 87, 89, 95, 96
79	55	0-5, 7-14, 16-33, 35-42, 46, 48, 49, 52, 54, 55, 58, 60, 61, 63, 65, 68, 71, 74, 80
103	63	0-7, 9-13, 15-29, 31-38, 41-44, 47-50, 53-57, 59-61, 63-66, 70, 73, 79, 85, 87, 92, 98
127	87	0-31, 33-37, 39-53, 55-62, 65-68, 71-74, 77-81, 83-85, 87-90, 94, 97, 103, 109, 111, 116, 122

Table 13: Propagation of a Single Bit Differential in the case of Grain-128a's LFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
31	83	0-10, 12-17, 19-22, 24-57, 60-63, 67-69, 71, 72, 74-79, 81-85, 87-89, 93, 94, 109, 111, 115
55	94	0-12, 14-34, 36-41, 43-46, 48-50, 53-65, 67-81, 86, 87, 91-93, 95, 96, 100-102, 105-108, 111, 112, 118, 133, 139
79	100	1-18, 20-36, 38-42, 45-57, 60-65, 67-70, 72-74, 78-89, 92-94, 96-100, 103, 104, 110, 111, 115, 119, 120, 125, 126, 130-132, 136, 157
103	93	0-8, 11-23, 25-40, 44-55, 58-60, 62-66, 69, 70, 72, 76-81, 84-87, 91, 92, 94, 96-98, 102, 109, 110-113, 116, 117, 123, 124, 128, 134, 143, 144, 149, 156
127	113	0-32, 35-47, 49-64, 68-79, 82-84, 86-90, 93, 94, 96, 100-105, 108-111, 115, 116, 118, 120-122, 126, 133-137, 140, 141, 147, 148, 152, 158

Table 14: Propagation of a Single Bit Differential in the case of Grain-128a's NFSR.

Flipped Bit Position	Number of Identical Keystream Bits	Positions of Identical Keystream Bits
31	44	0-15, 17, 18, 20-28, 30-36, 41, 49, 50, 54-56, 58, 63, 65, 66
55	55	0-9, 11-18, 20-39, 41, 42, 44, 45, 47, 49-52, 55-60, 65, 74
79	48	0-5, 7-14, 16-33, 35-39, 41, 46, 49, 52, 54, 55, 58, 60, 61, 63, 68
103	43	0-7, 9-13, 15-29, 31-38, 42, 53, 55-57, 59, 61
127	67	0-31, 33-37, 39-53, 55-62, 66, 77, 79-81, 83, 85

F Algorithms

Algorithm 16. Constructing Key-IV pairs that generate β bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

```

1 Set  $s \leftarrow 0$ 
2 while  $s = 0$  do
3   Choose  $K \in_R \{0, 1\}^n$  and  $V_2 \in_R \{0, 1\}^{d_2}$ 
4   Set  $value \leftarrow \text{Update}_1()$  and  $IV \leftarrow \text{LSB}_{\alpha-\beta+d_1}(P) \parallel V_2$ 
5   Run  $\text{KSA}^{-1}(K \parallel \text{LSB}_{d_1}(P_0) \parallel value \parallel V_2)$  routine for  $\beta - d_1$  clocks and produce state
      $S' = (K' \parallel V'_1 \parallel value \parallel V'_2)$ , where  $|V'_1| = \beta$  and  $|V'_2| = d_2 - \beta + d_1$ 
6   if  $V'_1 = P_0$  then
7     Set  $S \leftarrow K' \parallel P_0 \parallel value \parallel V'_2$  and  $output \leftarrow \text{Pair}_3(d_1, S)$ 
8     if  $output \neq \perp$  then
9       Set  $s \leftarrow 1$ 
10      return  $(K, IV)$  and  $output$ 
11    end
12  end
13 end
```

Algorithm 17. $\text{Update}_2(start, stop)$

Input: Indexes $start$ and $stop$
Output: Variable $value$

```

1 Set  $value \leftarrow \text{NULL}$ 
2 for  $i = start$  to  $stop$  do
3   Choose  $C_i \in_R \{0, 1\}^\delta$ 
4   Update  $value \leftarrow value \parallel C_i \parallel P_i$ 
5 end
6 return  $value$ 
```

Algorithm 18. $\text{Update}_3(value_1, value_2)$

Input: Variables $value_1$ and $value_2$
Output: Variable $value$

```

1 for  $i = t$  to  $c - 1$  do
2   Choose  $B_i \in_R \{0, 1\}^\delta$ 
3   Update  $value_1 \leftarrow value_1 \parallel B_i \parallel P_i$  and  $value_2 \leftarrow value_2 \parallel B_i$ 
4 end
5 Set  $value \leftarrow value_1 \parallel value_2$ 
6 return  $value$ 
```

Algorithm 19. Constructing Key-IV pairs that generate $\beta - \gamma + (\beta + \delta)(c - t)$ bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

- 1 Set $s \leftarrow 0$
- 2 **while** $s = 0$ **do**
- 3 Choose $K \in_R \{0, 1\}^n$, $V_1 \in_R \{0, 1\}^{d_1 - \beta + \gamma - (\beta + \delta)(c - t)}$ and $V_2 \in_R \{0, 1\}^{d_2}$
- 4 Set $value_1 \leftarrow P_0 \parallel \text{Update}_2(0, c - t - 2) \parallel C_{c-t-1} \parallel \text{MSB}_{\beta - \gamma}(P_{c-t})$ and $value_2 \leftarrow value_1$
- 5 Update $value_1 \leftarrow value_1 \parallel P_0$
- 6 **for** $i = 1$ to $t - 1$ **do**
- 7 Choose $B_i \in_R \{0, 1\}^{\delta - \beta + \gamma}$
- 8 Update $value_1 \leftarrow value_1 \parallel B_i \parallel \text{MSB}_{\beta - \gamma}(P_{c-t+i}) \parallel P_i$ and $value_2 \leftarrow value_2 \parallel B_i \parallel \text{MSB}_{\beta - \gamma}(P_{c-t+i})$
- 9 **end**
- 10 Set $value_1 \parallel value_2 \leftarrow \text{Update}_3(value_1, value_2)$ and $IV \leftarrow V_1 \parallel value_2 \parallel V_2$
- 11 Run $\text{KSA}^{-1}(K \parallel V_1 \parallel value_1 \parallel V_2)$ routine for $\beta - \gamma + (\beta + \delta)(c - t)$ clocks and produce state $S' = (K' \parallel V_1' \parallel value_1 \parallel V_2')$, where $|V_1'| = d_1$ and $|V_2'| = d_2 - \beta + \gamma - (\beta + \delta)(c - t)$
- 12 Set $IV' \leftarrow V_1' \parallel value_1 \parallel V_2'$
- 13 **if** (K', IV') produces all zero keystream bits in the first $\beta - \gamma + (\beta + \delta)(c - t)$ PRGA rounds **then**
- 14 Set $s \leftarrow 1$
- 15 **return** (K, IV) and (K', IV')
- 16 **end**
- 17 **end**

Algorithm 20. Constructing Key-IV pairs that generate $\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)$ bit shifted keystream

Output: Key-IV pairs (K', IV') and (K, IV)

- 1 Set $s \leftarrow 0$
- 2 **while** $s = 0$ **do**
- 3 Choose $K \in_R \{0, 1\}^n$, $V_1 \in_R \{0, 1\}^{d_1 - \delta + \beta - \gamma - \beta(c - t + 1) - \delta(c - t)}$, $V_2 \in_R \{0, 1\}^{d_2}$ and $C_{c-t+1} \in_R \{0, 1\}^{\delta - \beta + \gamma}$
- 4 Set $value_1 \leftarrow P_0 \parallel \text{Update}_2(1, c - t) \parallel C_{c-t+1}$ and $value_2 \leftarrow value_1$
- 5 Update $value_1 \leftarrow value_1 \parallel P_0$
- 6 **for** $i = 1$ to $t - 1$ **do**
- 7 Choose $B_i \in_R \{0, 1\}^{\delta - \beta + \gamma}$
- 8 Update $value_1 = value_1 \parallel \text{LSB}_{\beta - \gamma}(P_{c-t+i}) \parallel B_i \parallel P_i$ and $value_2 = value_2 \parallel \text{LSB}_{\beta - \gamma}(P_{c-t+i}) \parallel B_i$
- 9 **end**
- 10 Set $value_1 \parallel value_2 \leftarrow \text{Update}_3(value_1, value_2)$ and $IV \leftarrow V_1 \parallel value_2 \parallel V_2$
- 11 Run $\text{KSA}^{-1}(K \parallel V_1 \parallel value_1 \parallel V_2)$ routine for $\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)$ clocks and produce state $S' = (K' \parallel V_1' \parallel value_1 \parallel V_2')$, where $|V_1'| = d_1$ and $|V_2'| = d_2 - \delta + \beta - \gamma - \beta(c - t + 1) - \delta(c - t)$
- 12 Set $IV' \leftarrow V_1' \parallel value_1 \parallel V_2'$
- 13 **if** (K', IV') produces all zero keystream bits in the first $\delta - \beta + \gamma + \beta(c - t + 1) + \delta(c - t)$ PRGA rounds **then**
- 14 Set $s \leftarrow 1$
- 15 **return** (K, IV) and (K', IV')
- 16 **end**
- 17 **end**
