Probabilistic Signature Based Framework for Differential Fault Analysis of Stream Ciphers

Santanu Sarkar[1], Prakash Dey[2],
Avishek Adhikari[2] and Subhamoy Maitra[3]

[1] Indian Institute of Technology Madras, Sardar Patel Road, Chennai-600036, India.
[2] Department of Pure Mathematics, University of Calcutta, Kolkata-700019, India.
[3] Indian Statistical Institute, 203 B. T. Road, Kolkata- 700108, India.
sarkar.santanu.bir@gmail.com, pdprakashdey@gmail.com,
avishek.adh@gmail.com, subho@isical.ac.in

Abstract. Differential Fault Attack (DFA) has received serious attention in cryptographic literature and very recently such attacks have been mounted against several popular stream ciphers for example Grain v1, MICKEY 2.0 and Trivium, that are parts of the eStream hardware profile. The basic idea of the fault attacks consider injection of faults and the most general set-up should consider faults at random location and random time. Then one should identify the exact location and the exact timing of the fault (as well as multi bit faults) with the help of fault signatures. In this paper we consider this most general set-up and solve the problem of fault attack under a general framework, where probabilistic signatures are exploited. Our ideas subsume all the existing DFAs against the Grain family, MICKEY 2.0 and Trivium. In the process we provide improved fault attacks for all the versions of Grain family and also for MICKEY 2.0 (the attacks against Trivium are already quite optimal and thus there is not much scope to improve). Our generalized method can also take care of the cases where certain parts of the keystream bits are missing for authentication purpose. In particular, we show that the unsolved problem of identifying the faults in random time for Grain 128a can be solved in this manner. Our techniques can easily be applied to mount fault attack on any stream cipher of similar kind.

Keywords: Differential Attack, Fault Attack, Grain Family, MICKEY 2.0, Probabilistic Signatures, Stream ciphers.

1 Introduction

Differential Fault Attack on stream ciphers is an interesting area of research that is evident from many publications in this area. Given that the design of the stream ciphers are becoming more matured over the years, it is natural that it would be quite challenging to find the weaknesses of such designs unless certain assumptions are made. The assumptions should be realistic so that it is meaningful to study those models and there should be experimental verification for such efforts. A stream cipher state is initially populated by a secret key, some public variables such as Initialization Vector (IV), Nonce or Counter and some constant padding. This may be referred as Key Loading Algorithm (KLA). Then the cipher is run a number of rounds without producing any keystream bits, rather those bits are fed into the system to achieve randomness (by randomness, in this paper we will usually refer to pseudo-randomness). This part is known as Key Scheduling Algorithm (KSA). After that it is expected that the cipher will land into a random state. Then that state is evolved further to generate the random looking keystream, which is known as Pseudo Random Generator Algorithm (PRGA).

A popular model of cryptanalysis against such stream ciphers are as follows. As if the attacker does not know the key, but she may try out different public variables (we will now on refer this as IV only) to obtain different keystreams and from those keystreams, some kind of non-randomness may be explored. Further, it will be a stronger attack if the secret key can be obtained with a complexity less than the exhaustive key search. Given that designers are quite experienced, they generally run the cipher a large number of rounds in KSA, so that the initial difference in IVs cannot be exploited at the PRGA (that is after the completion of KSA). To provide more ammunition to the attacker, the differential fault attack is considered as an accepted model in this scenario. This model says that one can inject a fault in the state to generate a differential

during the PRGA. With this, the attacker obtains different keystreams and that may be studied to obtain the secret key. To be very precise about this model, let us enumerate the following points.

- The attacker can run the cipher with the same key and IV several times.
- For each run, the attacker can inject fault during the PRGA, but she will not be able to control
  - where: the exact bit location in the state and
  - when: the exact round of the PRGA, i.e., the timing information

  regarding the fault that has been injected.
- It may happen very well that the fault may toggle a few neighbouring bits in the state instead of a single location.
- Lesser the number of faults injected, the better it is for the cryptanalysis. This improves the attack, as well as it tampers the circuit less in case of actually mounting the attack.

While the fault attack is indeed optimistic and actually exploits a less secure model of the cipher, there are enough evidences that such an attack is possible to realize with proper hardware set-up.

The way a DFA is built up is as follows.

- Online Phase:
  - The cipher is run in fault-free condition and the keystream is preserved.
  - Then the cipher is run a few times with the same key and IV, but the faults are injected during the PRGA. The corresponding keystreams are also stored.
- Offline Phase:
  - The cipher is studied beforehand to generate certain signatures.
  - Once the fault-free and faulty keystreams are available, the signatures are used to locate the position(s) of the faults [at which state bit(s) the difference(s) had been incorporated] and the timing of the fault(s) [at which round(s) of the PRGA the fault(s) have been injected].
  - When these information are available, a lot of equations are formed with the unknown state bits as the variables and those equations are solved to obtain the state information.
  - If the PRGA and KSA are reversible, then one can easily obtain the key and IV. Otherwise, the attacker may use the state information for suitable purposes.

As we have already discussed, this is nothing but a differential attack. Cryptanalysts are well aware that differences in IVs during initial key/IV loading may not identify any weakness from the keystream during the PRGA if the KSA is sufficiently complicated. Similarly, if each PRGA round is substantially complicated then it would be quite hard to mount DFA on the cipher. For example, consider that the next state is obtained from the previous state by using a well designed hash function. In such a case, mounting such DFA may not be possible on a tamper-resilient stream cipher [11, Section 6]. Since the stream cipher designs are required to be very fast and need to be managed in a small hardware size (specially for hardware stream ciphers), during the PRGA it is not practical to put a lot of effort so that the states in two consecutive rounds will become un-correlated. Thus the ideal situation presented in [11, Section 6] may not be achievable in practice given speed and area constraints together. For practical stream ciphers, this situation is equivalent to the scenario when the keystream bits are available only at distant locations after quite a few rounds so that the states corresponding to those distant rounds become un-correlated. However, the situation has merit in case of authenticated encryption. Quite a few designs in this direction use stream ciphers as basic building blocks and they use some portion of the keystream bits for authentication, while rest of the bits are used for encryption. In this model, for DFA, the attacker will not have all the consecutive keystream bits and the generation of the signatures will be more challenging.

Contribution: With the background explained, the contribution of this paper are as follows.

- We simplify the existing complicated descriptions of the signatures in many recent papers such as [4–7, 10, 18]. Apart from that we introduce a new signature evolved from the probability value related to equality of each pair of fault-free and faulty keystream bits. We show that this new signature provides much improved results than what obtained in existing literature for the Grain family and MICKEY 2.0.
- Given the simplified description of the signatures, it becomes easier to study the combinatorial structures of the signatures, that in turn help in identifying the location and timing of the faults in an improved manner.

– Further the signatures can be defined when certain keystream bits are not available. This may be the situation when stream ciphers are used as building blocks in authenticated encryption, where some amount of keystream bits are used for authentication purposes. One such example cipher is Grain 128a. It has been noted in [18] that due to the unavailability of certain keystream bits, the timing information of the faults are required to mount the fault attack. However, with our generalized framework for signatures, we could show that it is indeed possible to mount fault attack on Grain 128a without knowing at what time instant the fault has been injected.

		Grain v1	Grain 128	Grain 128a, authentication mandatory	Grain 128a, authentication forbidden	MICKEY 2.0	Trivium
Control over Fault Timing	[18]	Fault must be injected at $t = 0$ or $t \in [0, 9]$	Fault must be injected at $t = 0$ or $t \in [0, 14]$	Fault must be injected at $t = 0$	-	-	-
	[6]	-	-	-	-	Fault must be injected at $t = 0$	-
	[14]	-	-	-	-	-	Any t but t must be known
	[9]	-	-	-	-	-	Any t and t could be unknown
	TW	Any t and t could be unknown	Any t and t could be unknown	Any t and t could be unknown	Any t and t could be unknown	Any t and t could be unknown	No scope for improvement
Prob. of Fault Location Identification	[18]	0.99 when $t = 0$ and 0.91 when $t \in [0, 9]$	1.0 when $t = 0$ and 0.92 when $t \in [0, 14]$	0.81 when $t = 0$	-	-	-
	[6]	-	-	-	-	Only two locations with probability 1 when round is known	-
	TW	1.0 when $t = 0$, 0.94 when $t \in [0, 5]$, 0.86 when $t \in [0, 30]$	0.89 when $t \in [0, 5]$, 0.66 when $t \in [0, 30]$	0.99 when $t = 0$, 0.76 when $t \in [0, 5]$, 0.53 when $t \in [0, 30]$	1.0 when $t = 0$, 0.86 when $t \in [0, 5]$, 0.61 when $t \in [0, 30]$	0.33 or 0.28 according as fault affects register R or S in all other positions when $t = 0$	-

Table 1: Single bit fault location identification: Survey on Contributions. [t denotes the fault injection round, TW denotes this work.]

Organization of the paper: The rest of the paper is organised in the following way. In Section 2 we study the combinatorial properties of fault signatures and describe the fault location identification procedure. In Section 3 we introduce probabilistic signature on stream ciphers and illustrate the same for eStream finalists in the hardware portfolio. In Section 4 we propose internal state recovery of the Grain family of ciphers with fault at random rounds. Finally Section 5 concludes the paper. Some Illustrations are provided in the Appendices A to C.

2 Fault Location Identification

This section deals with the fault location identification procedure for general setup. In the offline stage of this process, the attacker identifies, corresponding to each fault (possible in the underlying fault model) a deterministic or a highly probable pattern in the bitwise XOR difference stream of the fault free and faulty keystreams. These patterns are independent of the key-IV pair and are termed as signatures which are stored in the database. In this section, we first formally define few signatures of a fault for any stream cipher. Then

we study fault signature generation procedure and describe the generalized fault location identification in the online phase. Next we make an analysis when two faults can be differentiated deterministically. Finally we study some combinatorial aspects of fault signatures for the Grain family of ciphers that in turn help in identifying the location and timing of the faults in an improved manner. One should note that different stream ciphers might crucially affect the capabilities and the complexities of the attack strategy.

Let us develop our theories for general setup. Let $\mathcal{C}$ be a stream cipher with state size η. We assume that the algorithm of the stream cipher $\mathcal{C}$ is public. The cipher $\mathcal{C}$ could be such that certain keystream bits are not available. This may be the situation when keystream bits are used as building blocks in authenticated encryption. But since the design of cipher is known we can simulate the case as if keystream bits are available at each PRGA round of the cipher. We find patterns in case when all keystream bits were available and later these can simply be used to deduce patterns in case keystream bits are being suppressed. Therefore without loss of generality we assume that all keystream bits are available for the stream cipher $\mathcal{C}$.

Let $\mathcal{S}_i$ denote the internal state of $\mathcal{C}$ at the i-th PRGA round and z_i be the keystream bit produced at that point. We label the PRGA round from where the output keystream bit generation starts as 0. We abuse the " + " notation for Boolean XOR, i.e., $GF(2)$ addition as well as standard arithmetic addition. Later we use the symbol + for extended XOR and the sum of a set and an integer and that would be clear from the context. With this background, we now provide some of the definitions that would be required in the development of the theory for signatures.

Definition 1. (Fault Position and Fault Location) *A* fault position *is a non-empty subset ϕ of $\{0, \ldots, \eta-1\}$. A fault at fault position ϕ in round t, flips exactly the register bits given by ϕ. The pair (ϕ, t) is called a* fault location *or simply a* fault *when no ambiguity arises.*

(ϕ, t) represents a single bit fault or multi-bit fault at round t according as ϕ is singleton or not. When no ambiguity arises, if ϕ is singleton i.e., $\phi = \{e\}$, we simply denote ϕ by e. Also when $t = 0$, we simply denote the fault (ϕ, t) by ϕ.

Definition 2. (The XOR differential keystream) *If a fault (ϕ, t) is injected to the cipher at PRGA round t, we denote by $z_i^{(\phi,t)}$ the faulty keystream bit produced by the cipher at PRGA round i. Let $\delta z_i^{(\phi,t)} = z_i + z_i^{(\phi,t)}$ be the XOR difference of the fault-free and faulty keystream bit at PRGA round i. Then*
1. $\delta z^{(\phi,t)} = (\delta z_0^{(\phi,t)}, \delta z_1^{(\phi,t)}, \ldots)$ will be called the XOR differential keystream.
2. $\delta z^{(\phi,t,\ell)} = (\delta z_t^{(\phi,t)}, \ldots, \delta z_{t+\ell-1}^{(\phi,t)})$ will be called the XOR differential keystream of length ℓ from the point where the fault has been introduced.

Remark 1. It is evident that before the fault is injected, both the faulty and fault-free keystreams are exactly the same and thus, $\delta z_i^{(\phi,t)} = z_i + z_i^{(\phi,t)} = 0$, for $i < t$ and hence if a fault is injected at round t, then the previous rounds $0, \ldots, t-1$ will hold no clue for the fault. Thus we have to try to find key-IV independent patterns onwards the fault injection round t. But it is also not possible to test the entire XOR differential keystream onwards round t and so we restrict ourselves and try to find key-IV independent patterns in the XOR differential keystream at rounds $t, \ldots, t+\ell-1$ for some suitably chosen ℓ i.e., in the XOR differential keystream $\delta z^{(\phi,t,\ell)} = (\delta z_t^{(\phi,t)}, \ldots, \delta z_{t+\ell-1}^{(\phi,t)})$.

Definition 3. (Ψ, E, N and X signatures of a fault)
1. We denote the Ψ-signature of the fault (ϕ, t) simply by $\Psi(\phi, t, \ell)$ and express it as a string $s_0 \cdots s_{\ell-1} \in \{0, 1, .\}^\ell$ of length ℓ, where $s_i = b$ indicates that $P[\delta z_{i+t}^{(\phi,t)} = 1] = b$, $b \in \{0, 1\}$ and $s_i = .$ indicates that $0 < P[\delta z_{i+t}^{(\phi,t)} = 1] < 1$, $i \in \{0, \ldots, \ell-1\}$. This signature of a fault represents the probabilistic pattern of occurrence of 1 in the XOR differential keystream $\delta z^{(\phi,t,\ell)} = (\delta z_t^{(\phi,t)}, \ldots, \delta z_{t+\ell-1}^{(\phi,t)})$.
2. We denote the E-signature of the fault (ϕ, t) simply by $E(\phi, t, \ell)$ and express it as a collection of sets of the form $\{i, j\}$ of distinct rounds $i, j \in \{t, \ldots, t+\ell-1\}$ at which the XOR differential keystream bits must be deterministically equal, independent of the key and IV i.e., $P[\delta z_i^{(\phi,t)} + \delta z_j^{(\phi,t)} = 0] = 1$ but $P[\delta z_i^{(\phi,t)} = 0], P[\delta z_j^{(\phi,t)} = 0] \notin \{0, 1\}$.
3. We denote the N-signature of the fault (ϕ, t) simply by $N(\phi, t, \ell)$ and express it as a collection of sets of the form $\{i, j\}$ of distinct rounds $i, j \in \{t, \ldots, t+\ell-1\}$ at which the XOR differential keystream bits

must be deterministically different, independent of the key and IV i.e., $P[\delta z_i^{(\phi,t)} + \delta z_j^{(\phi,t)} = 1] = 1$ *but* $P[\delta z_i^{(\phi,t)} = 0], P[\delta z_j^{(\phi,t)} = 0] \notin \{0, 1\}$.
4. First we define p *to be the round* $\in \{t, \ldots, t+\ell-1\}$ *that corresponds the first occurrence of* 1*, if it exists in the* Ψ*-signature of the fault* (ϕ, t) *i.e.,* p *is such that* $P[\delta z_{t+p}^{(\phi,t)} = 1] = 1$ *but* $P[\delta z_t^{(\phi,t)} = 1], \ldots, P[\delta z_{t+p-1}^{(\phi,t)} = 1] \neq 1$*. In case no such round exists we define* $p = \ell - 1$*. We denote the* X*-signature of the fault* (ϕ, t) *simply by* $X(\phi, t, \ell)$ *and express it as a set of rounds at which the first 1 could occur in the XOR differential keystream i.e.,* $X(\phi, t, \ell) = \left\{ j \in \{t, \ldots, t+p\} \ : \ P[\delta z_j^{(\phi,t)} = 0] \neq 1 \right\}$.

2.1 Signature Generation

Let us now explain how one can generate the signatures of a fault. One may note that z_i and $z_i^{(\phi,t)}$ $(i \geq t)$ are both functions (call it f_i) of the state bits (i.e., function of $\mathcal{S}_t$) at the round t when the fault had been injected.
Let $\delta z_i^{(\phi,t)} = z_i + z_i^{(\phi,t)} = f_i(\mathcal{S}_t) + f_i(\mathcal{S}_t^{(\phi,t)}) = g_i^{(\phi,t)}(\mathcal{S}_t)$, say where $\mathcal{S}_t^{(\phi,t)}$ represents the faulty state at round t due to the fault (ϕ, t). Then if $g_i^{(\phi,t)}(\mathcal{S}_t)$ can be written explicitly and it becomes a constant function, then for the all zero function $P[\delta z_i^{(\phi,t)} = 1] = 0$, and for the all one function $P[\delta z_i^{(\phi,t)} = 1] = 1$. Otherwise, for non-constant $g_i^{(\phi,t)}(\mathcal{S}_t)$, we will have the $s_i = .$ that indicates that $0 < P[\delta z_i^{(\phi,t)} = 1] < 1$.

The case when we cannot write the expressions of the functions explicitly, we may have to perform a large number of experiments to estimate $P[\delta z_i^{(\phi,t)} = 1]$. Appendix B presents the 160 Ψ-signatures (80 for the single-bit faults in the LFSR and the other 80 for the single-bit faults in the NFSR) for Grain v1 with faults at round 0 and the length of the signature is $\ell = 130$. In the existing literature, the signatures have never been visualized in this way and thus such nice combinatorial patterns in the signatures could not be explored. For the data in Appendix A for Grain v1, we made 2^{16} experiments for each fault. Other signatures (such as E, N and X) could also be obtained in that way.

2.2 Combinatorial Aspects of Signatures

In this section we study the combinatorial properties of the signatures. This study not only simplifies the existing complicated descriptions of the signatures in many recent papers such as [4–7, 10, 18] but also helps us in identifying the location and timing of the faults in an improved manner as mentioned in Section 2.3. To develop the theories, let us start with the following definitions.

Definition 4. *Consider two strings* $u, v \in \{0, 1, .\}^\ell$*. Let us extend the definition of XOR to 'extended XOR'* $(\hat{\oplus})$ *as follows. For two inputs* $u_i, v_i \in \{0, 1, .\}$*,* $u_i \hat{\oplus} v_i = u_i + v_i$*, when* $u_i, v_i \in \{0, 1\}$*, otherwise,* $u_i \hat{\oplus} v_i = .$ *is defined. We will abuse the notation* $+$ *such that it can operate on two strings* $u = u_0 \cdots u_{\ell-1}$ *and* $v = v_0 \cdots v_{\ell-1}$ *as the string* $u + v = (u_0 \hat{\oplus} v_0) \cdots (u_{\ell-1} \hat{\oplus} v_{\ell-1})$*. Later we simply denote* $\hat{\oplus}$ *by* $+$ *when no ambiguity arises.*

Definition 5. *Let* $u = \Psi(\alpha_1, t, \ell)$ *and* $v = \Psi(\alpha_2, t, \ell)$*. If the string* $u + v$ *has the property that there exists at least one* i *such that* $u_i + v_i$ *is 1, then we will call* $\Psi(\alpha_1, t, \ell)$ *and* $\Psi(\alpha_2, t, \ell)$ *as non-matching patterns. Otherwise we will call* $\Psi(\alpha_1, t, \ell)$ *and* $\Psi(\alpha_2, t, \ell)$ *are matching patterns.*

In the online phase, given a fault (α, t) at an unknown location α, the string $w = \delta z_t^{(\alpha,t)} \cdots \delta z_{t+\ell-1}^{(\alpha,t)}$ will be a binary string. Now we need to find out the signature(s) $\Psi(\phi_j, t, \ell)$ that will match with w.

Definition 6. *We say that a binary pattern* w *of length* ℓ *matches with a signature* $\Psi(\phi_j, t, \ell)$*, if* w *and* $\Psi(\phi_j, t, \ell)$ *match.*

In this regard, we have the following technical result.

Proposition 1. *If* w *matches with both* $\Psi(\alpha_1, t, \ell)$ *and* $\Psi(\alpha_2, t, \ell)$*, then* $\Psi(\alpha_1, t, \ell)$ *and* $\Psi(\alpha_2, t, \ell)$ *will match with each other.*

It should be noted that the relation of matching between two signatures is reflexive and symmetric but not transitive. That is, it may very well happen that the pair $\Psi(\alpha_1, t, \ell)$ and $\Psi(\alpha_2, t, \ell)$ matches, and also the pair $\Psi(\alpha_2, t, \ell)$ and $\Psi(\alpha_3, t, \ell)$ matches, but the pair $\Psi(\alpha_1, t, \ell)$ and $\Psi(\alpha_3, t, \ell)$ does not match.

One may have a look at Appendix B for the faults that generate matching signatures in case of Grain v1. Naturally, for such cases, there exist possibilities, that given a string w, one may get confused in certain cases to identify the fault location exactly. As there are 160 state bits in Grain v1, we will have $\binom{160}{2}$ pairs of single faults and we note that out of them there are 537 pairs that match.

We now consider other signatures. We shall use the following notations [9]:
(a) For any integer i, $\emptyset + i = \emptyset$ ($\emptyset$ being the empty set).
(b) For any set S of integers and for any integer i, $S + i = \{s + i : s \in S\}$.
(c) For any set S if $s \in S$ implies that s is a set of integers then for any integer i, $S + i = \{s + i : s \in S\}$.

Theorem 1. *If all the PRGA rounds are identical in algorithm, then for any fault* (ϕ, t),
A. $\Psi(\phi, t, \ell) = \Psi(\phi, 0, \ell)$.
B. $E(\phi, t, \ell) = E(\phi, 0, \ell) + t$.
C. $N(\phi, t, \ell) = N(\phi, 0, \ell) + t$.

We now define the notion of 'possible fault' to simplify the online process when pre-computed signatures will be used to identify the possible fault locations. The notation ($\approx$) will be used in the fault location detection algorithms.

Definition 7. (Possible Fault) *Let at the online stage a fault* (α, T) *is injected. Then the XOR differential keystream* $\delta z^{(\alpha,T,\ell)} = (\delta z_T^{(\alpha,T)}, \ldots, \delta z_{T+\ell-1}^{(\alpha,T)})$ *is a binary sequence and is available to the attacker. Also let the attacker has pre-computed the* Ψ, E *and* N *signatures corresponding to a fault, say* (ϕ, t). *Now we shall say that '*(ϕ, t) *is a possible fault' if*

A. The Ψ*-signature* $\Psi(\phi, t, \ell)$ *matches with the binary string* $\delta z_T^{(\alpha,T)} \cdots \delta z_{T+\ell-1}^{(\alpha,T)}$.

B. For the E*-signature* $E(\phi, t, \ell)$, *either* $E(\phi, t, \ell) = \emptyset$ *or* $\{e, f\} \in E(\phi, t, \ell)$ *implies that* $\delta z_e^{(\alpha,T)} + \delta z_f^{(\alpha,T)} = 0$.

C. For the N*-signature* $N(\phi, t, \ell)$, *either* $N(\phi, t, \ell) = \emptyset$ *or* $\{e, f\} \in N(\phi, t, \ell)$ *implies that* $\delta z_e^{(\alpha,T)} + \delta z_f^{(\alpha,T)} = 1$.

If (ϕ, t) *is a possible fault corresponding to the XOR differential keystream* $\delta z^{(\alpha,T,\ell)} = (\delta z_t^{(\alpha,T)}, \ldots, \delta z_{T+\ell-1}^{(\alpha,T)})$, *we simply write* $(\phi, t) \approx \delta z^{(\alpha,T,\ell)}$.

Remark 2. One should note that, for any fault (α, T), it is immediate that $(\alpha, T) \approx \delta z^{(\alpha,T,\ell)}$.

2.3 The Online Phase

In the online stage, let a fault (α, T) be injected, where the fault position α and the fault injection round T are both unknown to the attacker. The attacker has the XOR differential keystream $\delta z^{(\alpha,T)}$ and wishes to identify the injected fault.

In order to identify the fault location, the attacker will use Algorithm 1 or Algorithm 2. But the choice will depend on the cipher and information available to the attacker. For example if the attacker knows that $T \in [t_{min}, t_{max}]$ she will use Algorithm 1, otherwise she will use Algorithm 2.

Algorithm 1:

```
P = ∅
for t = t_min to t_max do
    for each fault position φ do
        if (φ, t) ≈ δz^(α,T,ℓ) then
            P = P ⋃ {(φ, t)}
return P
```

Algorithm 2:

```
Compute the position p at which the first 1 occurs in the XOR differential keystream
P = ∅
for each fault position φ do
    for each i ∈ X(φ, 0, ℓ) do
        if p ≥ i then
            t = p − i
            if (φ, t) ≈ δz^(α,T,ℓ) then
                P = P ⋃ {(φ, t)}
return P
```

Algorithm 1 is self-explanatory. We now explain Algorithm 2 for Grain v1. Let us consider the LFSR bit position 0. The signature of the fault $(\{0\}, 0)$ is:

0000000000000000..0000001000000000..0.000.10..00010.....0.....

Clearly the first one occurs at the position 25 and for the LFSR bit position 0, $X(\{0\}, 0, 130) = \{17, 18, 25\}$. In this case, the fault $(\{0\}, 0)$ occurs at round 0 and so the first 1 in the XOR differential stream can occur in the positions given by $X(\{0\}, 0, 130)$. Therefore in the online stage, for the fault $(\{0\}, t)$ at PRGA round t, the first 1 in the XOR differential stream can occur at the positions $t+17$, $t+18$ or $t+25$ i.e., positions given by $p = t + j$ where $j \in X(\{0\}, 0, 130)$. Thus if the first 1 actually occurs at the round p, then $t = p - j$ must hold for some j, where $j \in X(\{0\}, 0, 130)$.

Previously approach of [18] for Grain 128a to identify faults at random rounds fails as mentioned in [18, Page 14]. However, now that could be done.

2.4 Deterministic Fault Identification

Following the above discussion, the next questions appear immediately: Whether the fault location identification can be done deterministically? In this section we make an analysis when it is possible to deterministically identify faults. For that it is absolutely necessary to deterministically differentiate between the occurrence of two faults.

Two faults (α_1, t_1) and (α_2, t_2) can be deterministically differentiated if their Ψ signatures do not match.

One should also note that, it may happen very well that the Ψ-signatures $u = \Psi(\alpha_1, t, \ell), v = \Psi(\alpha_2, t, \ell)$ match, but it is still possible to obtain the exact fault location from w. Let $w_i = b \in \{0, 1\}$ and one of u_i, v_i be $1 - b$, say $u_i = 1 - b$. Then one may note that the fault cannot occur for α_1, but for α_2.

Now as stated earlier for Grain v1, experiments show that out of $\binom{160}{2}$ possible pairs of single faults there are 537 pairs that match and thus cannot be identified deterministically. Also each and every single bit fault possible for Grain 128a, when authentication is forbidden, can be uniquely identified with certainty.

2.5 Some Combinatorial Properties of Signatures for Grain Family

This section deals with the combinatorial properties of the signatures for single bit faults on Grain family. The notion of equivalent faults, introduced in this section, helps us in reducing the number of signature generation by considering the faults only at the tap positions.

Given a state of size η, we will have η many single-bit faults at a particular round but closer analysis shows that in the offline stage we need to compute signatures for few faults for Grain family of ciphers.

A cipher in the Grain family (Appendix A) consists of an n-bit NFSR and an n-bit LFSR. For Grain v1 the value of n is 80, while for both Grain 128 and Grain 128a the value of n is 128. The number of possible single bit faults is $\eta = 2n$. As stated earlier we shall first analyze the Grain family of ciphers when all keystream bits are produced as output and later that can be used to consider the cases of keystream bit suppression. In the online phase, a single bit fault may occur at any one of the η positions. Each fault has its own signature. In this section, we study relationships among the signatures of the single bit faults and show that in the offline phase we only need to compute signatures for q many faults at round 0, where q is the number of taps (both update and output) for the cipher. We first illustrate the fact using Grain v1 and later we shall build it up formally for the Grain family of ciphers.

We shall also use the following convention: For a single bit fault position $\{\alpha\}$, $\{\alpha\} \subset \{0, \ldots, n-1\}$ shall mean that the fault is at the NFSR bit α, whereas $\{\alpha\} \subset \{n, \ldots, 2n-1\}$ mean that the fault is at the LFSR bit $\alpha - n$.

Let us consider the NFSR bits 4, 5, 6, 7, 8 of Grain v1. The Ψ-signature (with length $\ell = 130$) of the corresponding faults $(\{4\}, 0)$, $(\{5\}, 0)$, $(\{6\}, 0)$, $(\{7\}, 0)$, $(\{8\}, 0)$ are:

$(\{4\}, 0)$: 10110000000000000000.0000001000000000..0.000.10..00010.....0.....
$(\{5\}, 0)$: 01011000000000000000.0000001000000000..0.000.10..00010.....0.....
$(\{6\}, 0)$: 00101100000000000000.0000001000000000..0.000.10..00010.....0.....
$(\{7\}, 0)$: 00010110000000000000.0000001000000000..0.000.10..00010.....0.....
$(\{8\}, 0)$: 00001011000000000000.0000001000000000..0.000.10..00010.....0.....

One should note that, the Ψ-signatures of the faults $(\{5\}, 0)$, $(\{6\}, 0)$, $(\{7\}, 0)$, $(\{8\}, 0)$ are right shifts of the Ψ-signature of the fault $(\{4\}, 0)$ with pre-appended zeros, i.e.,

$\Psi(\{5\}, 0, 130) = 0||\Psi(\{4\}, 0, 129)$,
$\Psi(\{6\}, 0, 130) = 00||\Psi(\{4\}, 0, 128)$,

$\Psi(\{7\}, 0, 130) = 000||\Psi(\{4\}, 0, 127)$ and
$\Psi(\{8\}, 0, 130) = 0000||\Psi(\{4\}, 0, 126)$, $||$ being the string concatenation operator.

First we investigate why this happens. The NFSR bits 4, 5, 6, 7, 8 are such that 4 is a tap position but 5, 6, 7, 8 are not. When a single bit fault occurs at the NFSR position $\{8\}$ at round 0, it has no immediate effect on the keystream until it reaches the tap $\{4\}$ (due to bit shifting in the registers) at round 4 (See Fig. 5 in Appendix). Thus the keystream bit produced at the rounds $0, 1, 2, 3$ are not faulty at all. Hence the XOR difference of the normal and faulty keystream bit produced at rounds $0, 1, 2, 3$ due to fault at $\{0\}$ at round 0 must be certainly 0. Hence the Ψ-signature of the fault $(\{8\}, 0)$ must have 4 leading zeros. At round 4 the faulty bit arrives at the NFSR tap $\{4\}$. Thus the effect of the fault $(\{8\}, 0)$ is identical with that of the faults $(\{7\}, 1)$,$(\{6\}, 2)$, $(\{5\}, 3)$ and $(\{4\}, 4)$. But the Ψ-signature of the fault $(\{4\}, 4)$ is identical with the Ψ-signature of the fault $(\{4\}, 0)$ due to the fact that each PRGA round is identical. Therefore the Ψ-signature of the fault $(\{4\}, 0)$ is enough to compute the Ψ-signatures of the fault $(\{8\}, 0)$. Similar comments hold for the faults $(\{5\}, 0)$, $(\{6\}, 0)$, $(\{7\}, 0)$. We now present it formally for the Grain family of ciphers.

The NFSR and LFSR tap positions (both update and output taps) are provided in Appendix C, Table 8. The tap positions are written in the increasing order of the indices.
For a particular version of Grain, let the NFSR and LFSR taps are respectively given by $\alpha_0, \ldots, \alpha_{k-1}$ and $\beta_0, \ldots, \beta_{r-1}$ (with $\alpha_i < \alpha_{i+1}$ and $\beta_j < \beta_{j+1}$). We set $\alpha_k = \beta_r = n$ which are originally neither a NFSR nor LFSR tap position but we now consider them as fake taps. Our objective is to partition the $2n$-bit state of the cipher into $k + r$ non-overlapping blocks in such a way that the number of pre-computed signatures of faults could be minimized.
We define k NFSR blocks as $[\alpha_i, \alpha_{i+1} - 1]$, $i \in \{0, \ldots, k-1\}$. Similarly we define r LFSR blocks as $[\beta_i + n, \beta_{i+1} + n - 1]$, $i \in \{0, \ldots, r-1\}$. Clearly this is a $k + r$ non-overlapping partition of the $2n$-bit state of the cipher. We call them register blocks. Our first objective is to show that when the injected faults are of single bit in nature, pre-computing the signatures of the $k + r$ faults $(\{\alpha_0\}, 0)$, $\ldots$,$(\{\alpha_{k-1}\}, 0)$, $(\{\beta_0 + n\}, 0)$, $\ldots$, $(\{\beta_{r-1} + n\}, 0)$ is enough. To develop the theory, let us consider the following definition of equivalent faults.

Definition 8. (Equivalent Faults) *Two faults (α_1, t_1) and (α_2, t_2) are said to be* equivalent *if they produce the same faulty keystream bits at all the PRGA rounds for each and every key-IV pair (both faults are injected on states generated by the same key-IV pair), i.e., $z_i^{(\alpha_1, t_1)} = z_i^{(\alpha_2, t_2)}$, $\forall i \geq 0$ and for any key-IV pair.*

Theorem 2. *If $[a, b]$ is a block in the state of the cipher, then for any $t \geq 0$, the $b - a + 1$ single bit faults $(\{b\}, t)$, $(\{b-1\}, t+1)$, $\ldots$, $(\{a+1\}, t+b-a-1)$, $(\{a\}, t+b-a)$ are all equivalent.*

Corollary 1. *For a particular version of Grain, if the NFSR and LFSR taps are respectively given by $\alpha_0, \ldots, \alpha_{k-1}$ and $\beta_0, \ldots, \beta_{r-1}$ (with $\alpha_i < \alpha_{i+1}$ and $\beta_j < \beta_{j+1}$), then without loss of generality it may be assumed that the single bit faults are injected at the tap positions at some round.*

Theorem 3. *If $[a, b]$ is a register block, then for any c with $a \leq c \leq b$, the Ψ-signature of the fault $(\{c\}, 0)$ (when all the keystream bits are available) is given by $\Psi(\{c\}, 0, \ell) = \underbrace{0 \cdots 0}_{c-a}||\Psi(\{a\}, 0, \ell - c + a)$.*

Theorem 4. *If $[a, b]$ is a register block, then for any c with $a \leq c \leq b$, the E-signature and N-signature of the fault $(\{c\}, 0)$ (when all the keystream bits are available) are respectively given by*

A. $E(\{c\}, 0, \ell) = E(\{a\}, 0, \ell - c + a) + c - a$ and
B. $N(\{c\}, 0, \ell) = N(\{a\}, 0, \ell - c + a) + c - a$.

3 Probabilistic Signatures on Stream Ciphers

The description of signatures as in the literature of DFAs on Grain family or MICKEY are not optimal in the sense that they consider only partial information in the online stage. For example the Ψ, E and N signatures correspond to the cases that occur with probability 1 in XOR differential keystreams. In this section we propose new signatures $P_1(\phi, t, \ell)$ and $P_2(\phi, t, \ell)$ for a fault that consider events that might occur with high probability. We call these as 'probabilistic signatures of a fault'. These are fixed length vectors consisting of probabilities of pre-fixed events and may be considered as a generalisation of the Ψ, E and N signatures.

It has been reported in [18] that using the Ψ, E and N signatures, one can identify the fault location (faults are injected at round 0) for Grain 128a (when authentication is mandatory) with probability 0.81. In this section we consider the fault location identification problem from coding theoretic point of view. Using Maximum Likelihood Approach (MLA) and the probabilistic signatures, we show that one can improve the fault identification probability up to 0.99 for Grain 128a (when authentication is mandatory). Also for MICKEY 2.0 the probabilistic signatures provide significant improvement in fault identification probability.

For simplicity of explanation, we first assume that faults are injected at the known round $t = 0$ i.e., we consider faults of the form $(\phi, 0)$. Later we relax this assumption. Recall that $\delta z^{(\phi,0,\ell)} = (\delta z_0^{(\phi,0)}, \ldots, \delta z_{\ell-1}^{(\phi,0)})$ is the XOR differential keystream of length ℓ from the point where the fault has been introduced.

We define $P_1(\phi, 0, \ell) = (P[\delta z_0^{(\phi,0)} = 1], \ldots, P[\delta z_{\ell-1}^{(\phi,0)} = 1])$ i.e., the sequence of probabilities of occurrence of 1 in the XOR differential keystream.

While $P_1(\phi, 0, \ell)$ is a ℓ length real vector, we define $P_2(\phi, 0, \ell)$ as the $\frac{l(l-1)}{2}$ length real vector $(P[\delta z_0^{(\phi,0)} + \delta z_1^{(\phi,0)} = 1], \ldots, P[\delta z_0^{(\phi,0)} + \delta z_{\ell-1}^{(\phi,0)} = 1], P[\delta z_1^{(\phi,0)} + \delta z_2^{(\phi,0)} = 1], \ldots, P[\delta z_1^{(\phi,0)} + \delta z_{\ell-1}^{(\phi,1)} = 1], \ldots, P[\delta z_{\ell-1}^{(\phi,0)} + \delta z_{\ell-2}^{(\phi,0)} = 1])$.

One should note that these are similar probabilities as considered in Section 2.4 for deterministic fault location identification. Similar to the Ψ, E and N signatures, these signatures can be computed using simulation. Let μ be the number of fault positions. For single bit faults $\mu = \eta$. In the online phase the attacker starts with (1) 2μ pre-computed signatures $P_1(\phi_j, 0, \ell)$, $P_2(\phi_j, 0, \ell)$ where $j \in [0, \mu - 1]$ and (2) the binary XOR differential keystream $\delta z^{(\alpha,0,\ell)} = (\delta z_0^{(\alpha,0)}, \ldots, \delta z_{\ell-1}^{(\alpha,0)})$ corresponding to an unknown fault position α. Adversary now wants to identify α on the basis of the probabilistic signatures.

In order to use the probabilistic signature $P_1(\phi_j, 0, \ell)$, we first define a distance function on a binary vector $B = (b_0, \ldots, b_{\ell-1})$ and a real vector $A = (a_0, \ldots, a_{\ell-1})$ as $d(B, A) = -\sum_{i=0}^{\ell-1} \Big(\log(1 - a_i) \cdot I_{b_i=0} + \log a_i \cdot I_{b_i=1} \Big)$, where $I_{b_i=b}$ is a indicator function which is 1 if $b_i = b$ else 0, $b \in \{0, 1\}$. Function $d(B, A)$ measures the distance between B and A. Note that if for any $i \in [0, \ell - 1]$, $a_i = b$ but $b_i = 1 - b$, distance function $d(B, A) = \infty$ assuming $\log(0) = -\infty$.

Now in the pruning, using MLA one should minimize

$$\min_{0 \le j \le \mu-1} d\Big(\delta z^{(\alpha,0,\ell)}, P_1(\phi_j, 0, \ell) \Big).$$

Let us now explain the rationality behind the definition of the function $d(B, A)$. Suppose $\delta z^{(\alpha,0,\ell)} = (b_0, \ldots, b_{\ell-1})$ and $P_1(\phi_j, 0, \ell) = (a_0, \ldots, a_{\ell-1})$. Let i-th coordinate of the vector $\delta z^{(\alpha,0,\ell)}$ be $b_i = 0$. On the other hand $a_i = P[\delta z_i^{(\phi_j,0)} = 1]$. Thus the quantity $1 - a_i$ contributes the possibility that $\delta z^{(\alpha,0,\ell)}$ is generated due to fault at ϕ_j-th location. Similarly when $b_i = 1$, a_i contributes the possibility that $\delta z^{(\alpha,0,\ell)}$ is generated due to fault at ϕ_j-th location.

Thus using Maximum Likelihood Approach (MLA), we want to maximize $M = \prod_{i=0}^{\ell-1} \Big((1 - a_i) \cdot I_{b_i=0} + a_i \cdot I_{b_i=1} \Big)$. This problem is same as to minimize the quantity $-\sum_{i=0}^{\ell-1} \Big(\log(1 - a_i) \cdot I_{b_i=0} + \log a_i \cdot I_{b_i=1} \Big)$. That is we want to minimize $\min_{0 \le j \le \mu-1} d\Big(\delta z^{(\alpha,0,\ell)}, P_1(\phi_j, 0, \ell) \Big)$.

Note that if $a_i = 1$ and $b_i = 0$ or $a_i = 0$ and $b_i = 1$, M will be zero. And in that case, our distance function becomes ∞. Thus the distance function captures the pruning of the Ψ-signature automatically.

To use $P_2(\phi_j, 0, \ell)$, we define another binary vector $U = (w_{0,1}, \ldots, w_{l-2,l-1})$ of length $\frac{l(l-1)}{2}$, where $w_{i,j} = b_i + b_j = \delta z_i^{(\alpha,0)} + \delta z_j^{(\alpha,0)}$. In our pruning we want to minimize $\min_{0 \le j \le \mu-1} \Big(d\big(\delta z^{(\alpha,0,\ell)}, P_1(\phi_j, 0, \ell)\big) + d\big(U, P_2(\phi_j, 0, \ell)\big) \Big)$.

Our approach is presented in Algorithm 3.

We now consider the fault (α, T) where both α and T are unknown. Now suppose $T \in [t_{min}, t_{max}]$. This is actually no assumption since the first occurrence of 1 in the XOR differential keystream provides an upper bound of the fault injection round. Next for each t, we consider $\delta z^{(\alpha,t,\ell)} = (\delta z_t^{(\alpha,t)}, \ldots, \delta z_{t+\ell-1}^{(\alpha,t)})$.

We define $P_1(\phi, t, \ell) = (P[\delta z_t^{(\phi,t)} = 1], \ldots, P[\delta z_{t+\ell-1}^{(\phi,t)} = 1])$ i.e., the sequence of probabilities of occurrence of 1 in the XOR differential keystream $\delta z^{(\alpha,t,\ell)}$, and define $P_2(\phi, t, \ell)$ as the $\frac{l(l-1)}{2}$ length real vector $(P[\delta z_t^{(\phi,t)} + \delta z_{t+1}^{(\phi,t)} = 1], \ldots, P[\delta z_t^{(\phi,t)} + \delta z_{t+\ell-1}^{(\phi,t)} = 1]$, $P[\delta z_{t+1}^{(\phi,t)} + \delta z_{t+2}^{(\phi,t)} = 1], \ldots, P[\delta z_{t+1}^{(\phi,t)} + \delta z_{t+\ell-1}^{(\phi,t)} = 1], \ldots$, $P[\delta z_{t+\ell-1}^{(\phi,t)} + \delta z_{t+\ell-2}^{(\phi,t)} = 1])$.

We also define $U_t = (w_{t,t+1}, \ldots, w_{t+l-2,t+l-1})$ of length $\frac{l(l-1)}{2}$, where $w_{t+i,t+j} = \delta z_{t+i}^{(\alpha,t)} + \delta z_{t+j}^{(\alpha,t)}$.

Now we find (α, T) by minimizing the quantity

$\min_{t_{min} \le t \le t_{max}, 0 \le j \le \mu-1} \Big(d\big(\delta z^{(\alpha,t,\ell)}, P_1(\phi_j, t, \ell)\big) + d\big(U_t, P_2(\phi_j, t, \ell)\big) \Big)$.

Algorithm 3: Probabilistic Prunning Algorithm

Input: $P_1(\phi_i, 0, \ell) = (a_{i,0}, \ldots, a_{i,\ell-1})$,
$P_2(\phi_i, 0, \ell) = (b_{i,0,1}, \ldots, b_{i,\ell-2,\ell-1})$
$\forall i \in [0, \mu-1]$

Output: Fault position α // `Fault injection Round: 0.`

```
min_0 = ∞
for i = 0 to μ − 1 do
    min = 0
    for j = 0 to ℓ − 1 do
        if δz_j^(φ_i.0) = 0 then
            min = min − log(1 − a_{i,j})
        else
            min = min − log(a_{i,j})
        for k = j + 1 to ℓ − 1 do
            if δz_j^(φ_i.0) + δz_k^(φ_i.0) = 0 then
                min = min − log(1 − b_{i,j,k})
            else
                min = min − log(b_{i,j,k})
    if min < min_0 then
        min_0 = min
        α = φ_i
```

3.1 Impact on the Grain Family

Our MLA gives much better success probability than combinatorial approach. The reason is that there are i, ϕ, such that $P[\delta z_i^{(\phi,0)} = 0] = \epsilon$, where $\epsilon \neq 0, 1$ is significantly different from $\frac{1}{2}$. For an example in Grain 128a (when authentication is mandatory), $P[\delta z_{34}^{(\{128\},0)} = 1] = \frac{3}{4}$. This information (not used earlier) can be used to improve the success probability.

For Grain v1, Fig 1 provides the distributions of $P[\delta z_i^{(\{\phi\},t)} = 1]$ for the faults $(\{0\}, 0)$ and $(\{80\}, 0)$. Note that the Ψ-signature of these faults matches.

Experiment with 10000 random key-IV shows that we can identify the correct fault location for Grain 128a when authentication is mandatory with success probability 0.99.

For Grain v1 success probability now becomes 1.0 for the first time. In [18], success probability of Grain v1 was reported as 0.99.

Experimental results for all versions of Grain are presented in Table 2. From the Table 2, it is clear we can find the round together with fault location easily when $t_{max} \le 30$. Approach of [18] for Grain 128a, when authentication is mandatory, fails as mentioned in [18, Page 14]. However, for this cipher we obtain significant success (Table 2).

We also report that, for Grain 128a, when authentication is mandatory and fault injection round is 0, 2-bit and 3-bit faults can be identified with probabilities 1.0 and 0.99 respectively.

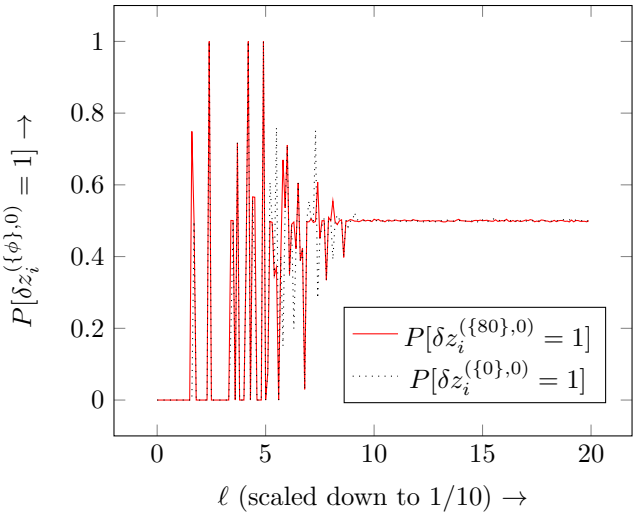


Fig. 1: Grain v1: Distributions of $P[\delta z_i^{(\{0\},0)} = 1]$, $P[\delta z_i^{(\{80\},0)} = 1]$, $0 \leq i \leq 200$.

$\mathcal{C}$	t_{min}	t_{max}	Existing Results	Prob. of Identification
Grain v1	0	0	0.99 [18]	1.0
	0	5	-	0.94
	0	30	-	0.86
Grain 128	0	0	1.0 [18]	1.0
	0	5	-	0.89
	0	30	-	0.66
Grain 128a*	0	0	-	1.0
	0	5	-	0.86
	0	30	-	0.61
Grain 128a**	0	0	0.81 [18]	0.99
	0	5	-	0.76
	0	30	-	0.53

Table 2: Single bit fault location identification probability using Algorithm 3. $*$ and $**$ respectively means the versions of Grain 128 when authentication is forbidden and mandatory.

3.2 Impact on MICKEY 2.0

The stream cipher MICKEY 2.0 [3], designed by Steve Babbage and Matthew Dodd, has been selected as a part of eStream's final hardware portfolio. Like Grain v1, MICKEY is also a synchronous, bit-oriented stream cipher. After a TMD tradeoff attack [15] against the initial version of MICKEY (version 1), the designers proposed a new variant which is known as MICKEY 2.0. Detailed description of MICKEY 2.0 is available in [3].

Differential Fault Attacks on MICKEY were presented in [6, 16]. Later [7] provided an improvement on [6]. One of the main bottleneck in MICKEY is to identify the fault location (when fault injection affects a single round only).

Experimentally we observe that $P[\delta z_i^{(\{e\},0)} + \delta z_j^{(\{e\},0)} = 1]$ is approximately 0.5, $0 \leq i < j \leq \ell$, where the fault position $\{e\}$ corresponds to the register R and $\ell = 60$. So we can not use pairwise keystreams for pruning,

Using our probabilistic signature approach, we observe experimentally that if fault (single bit) injection round is known, one can identify the fault location with success probability 0.33 when the fault occurs in R register and 0.28 when the fault occurs in S register. These probabilities are much higher than the random guess $\frac{1}{100} = 0.01$. Experimental results are presented in Table 3.

Impact on Trivium Our kind of analysis naturally works efficiently on Trivium [8]. In [14], it has already been shown that single bit fault locations can be determined with certainty for this cipher. This is because

Register	t_{min}	t_{max}	Prob. of Identification
R	0	0	0.33
	0	5	0.07
	0	30	0.01
S	0	0	0.28
	0	5	0.07
	0	30	0.02

Table 3: Fault Location (single bit) identification probability using Algorithm 3 on MICKEY.

structure of Trivium is very simple. This is the reason, there is not much scope of improvement against Trivium given the fault attacks are already more or less optimal.

4 Internal State Recovery of the Grain Family of Ciphers with Fault at Random Rounds

After identifying the fault locations (Section 2, Section3), we initiate the process of recovering the state of the cipher. Now for Grain v1, Grain 128 and Grain 128a when authentication is forbidden, all the normal and faulty keystream bits are available to the attacker, but for Grain 128a when authentication is mandatory, keystream bits are available to the attacker only at the even rounds $i \in \{0, 2, 4, \ldots\}$. One should also note that if $(\alpha_0, t_0), \ldots, (\alpha_{m-1}, t_{m-1})$ are m faults injected to the cipher and if $0 \leq t_0 \leq \cdots \leq t_{m-1}$ holds, then information prior to the cipher round t_0 simply could be discarded. Thus we assume that, in the internal state recovery stage the adversary starts with the following information:

(A) m fault locations, given by $(\alpha_0, t_0), \ldots, (\alpha_{m-1}, t_{m-1})$, where $0 \leq t_0 \leq \cdots \leq t_{m-1}$. **(B)** normal (fault free) keystream $z = (z_0, \ldots, z_{r-1})$ of length r. **(C)** m faulty keystreams $z^0, \ldots, z^{m-1}$ each of length r. **(D)** the faulty keystream $z^j = (z_0^j, \ldots, z_{r-1}^j)$ occurred due to the fault (α_j, t_j). **(E)** For Grain v1, Grain 128 and Grain 128a when authentication is forbidden, normal and faulty keystream bits at all the rounds are known to the attacker. But for Grain 128a when authentication is mandatory, the normal keystream bits $z_1, z_3, \ldots$ and the faulty keystream bits $z_1^j, z_3^j, \ldots$ are not available.

The internal state will be recovered at the PRGA base round t_0. For that we shall consider the l rounds given by $\{t_0, \ldots, r-1\}$ i.e., $\{t_0, \ldots, t_0 + l - 1\}$ where $l = r - t_0$.

4.1 Generating Polynomial Equations

Let the fault free internal state at the round i ($\geq t_0$) be $\mathcal{S}_i = (X_i, Y_i)$ where $X_i = (x_i, \ldots, x_{i+n-1})$ and $Y_i = (y_i, \ldots, y_{i+n-1})$, the internal state at round $i = t_0$ being $\mathcal{S}_{t_0} = (x_{t_0}, \ldots, x_{t_0+n-1}, y_{t_0}, \ldots, y_{t_0+n-1})$. We treat each x_i and y_i as variables. Corresponding to each normal pre-output bit z_i, we introduce two new inner state variables x_{i+n}, y_{i+n} and obtain the following two equations: $y_{i+n} = f(Y_i), x_{i+n} = y_i + g(X_i)$. Further if z_i is a keystream bit available, we consider the additional equation $z_t = \sum_{i=0}^{n-1} b_i y_{t+i} + \sum_{j=0}^{n-1} a_j x_{i+j} + h(\mathcal{S}_i)$. In this case we are considering the $2n + 2l$ variables $x_{t_0}, \ldots, x_{t_0+n-1}, x_{t_0+n}, \ldots, x_{t_0+n+l-1}$, $y_{t_0}, \ldots, y_{t_0+n-1}, y_{t_0+n}, \ldots, y_{t_0+n+l-1}$.

Let us now consider a fault $(\alpha, t) \in \{(\alpha_0, t_0), \ldots, (\alpha_{m-1}, t_{m-1})\}$. Since the cipher device is re-keyed with the same key-IV, before each fault injection, due to the fault injection, if the faulty internal state at round i be $\mathcal{S}_i^{(\alpha,t)}$ then, $\mathcal{S}_t^{(\alpha,t)}(e) = \mathcal{S}_t(e) + 1, \ \forall e \in \alpha$ and $\mathcal{S}_t^{(\alpha,t)}(e) = \mathcal{S}_t(e), \ \forall e \in [0, n-1] \setminus \alpha$ whereas $\mathcal{S}_i^{(\alpha,t)} = \mathcal{S}_i$ for all $i \in [0, t-1]$.

Again corresponding to each pre-output bit z_i, we introduce two new (faulty) inner state variables $x_{i+n}^{\alpha,t}$, $y_{i+n}^{\alpha,t}$ ($i \geq t$) respectively for the NFSR and the LFSR and obtain more equations as above. For each fault (α, t), in this case we introduce $2(t_0 - t + l)$ new faulty inner state variables $x_{t+n}^{\alpha,t}, \ldots, x_{t_0+n+l-1}^{\alpha,t}$, $y_{t+n}^{\alpha,t}, \ldots, y_{t_0+n+l-1}^{\alpha,t}$.

Now the system of polynomial equations thus obtained are simply passed on to the SAT solver Cryptominisat 2.9.6 in SAGE 6.9.3 for extracting solution for the inner state variables $x_{t_0}, \ldots, x_{t_0+n-1}, y_{t_0}, \ldots, y_{t_0+n-1}$ at round $i = t_0$. The hardware platform is an HP Z800 workstation with 3 GHz Intel(R) Xeon(R) CPU.

4.2 Experimental Results on SAT Solving

Let $\mathcal{C}$ denotes any one of the ciphers Grain v1, Grain 128 and Grain 128a. For each SAT solving trial (with cutting number 4) we first generated an inner state at PRGA round 0 by choosing key-IV randomly. Then we simulated m random bit faults at random rounds between 0 to $\tau - 1$. For example, we generated single bit faults $(\{e\}, t)$ uniformly and independently by choosing $e \in \{0, \ldots, \eta - 1\}$ and $t \in \{0, \ldots, \tau - 1\}$.

Single Bit Faults					Time in Seconds		
$\mathcal{C}$	m	τ	l	NOE	MinTime	MaxTime	AvgTime
Grain v1	11	30	200	250	17.01	5643.01	317.14
	12	30	200	250	21.28	6406.76	277.21
Grain 128	4	30	300	250	22.10	6207.65	486.36
	5	30	300	250	9.72	954.06	46.42
Grain 128a*	5	30	300	250	20.55	6991.16	387.87
	6	30	300	250	17.37	1288.31	79.93
Grain 128a**	10	30	300	250	22.53	6758.41	22.53
	11	30	300	250	1648.72	22.16	182.85

Table 4: SAT Experiment Data for single bit faults. NOE: Number of Experiments performed.

Grain 128a				Time in Seconds		
m	τ	l	NOE	MinTime	MaxTime	AvgTime
10	30	300	1000	16.57	6820.56	353.52
11	30	300	1000	18.56	2498.04	163.05

Table 5: SAT solving results for Grain 128a when authentication is mandatory with 3-neighbourhood bit faults.

5 Conclusion

In this paper, we consider Differential Fault Attack (DFA) on Stream cipher. In DFA, attacker first needs to find the location as well as the round of the fault. After that, a system of equations are generated and we solve them using SAT solvers. In the SAT solver phase, getting the solutions is generally an automated approach. However, obtaining the locations and the timing information regarding the faults seem to be the most challenging task in the DFA type of attacks. In this paper, we concentrate on this problem. Our generalized frame work for probabilistic signature schemes provide improve results in success probability over the existing efforts and further we could identify the fault locations and timing in certain cases (for example Grain 128a, MICKEY 2.0), that had not been possible earlier. Our method can be used even in the scenario where certain portion of keystream bits may not be available. This scenario may happen when a stream cipher is used in authenticated encryption schemes. Thus such schemes in second round of CAESAR competition may be analyzed against differential fault attack using our approaches.

References

1. M. Ågren, M. Hell, T. Johansson and W. Meier. A New Version of Grain 128 with Authentication. Symmetric Key Encryption Workshop 2011, 2011.
2. M. Ågren, M. Hell, T. Johansson and W. Meier. Grain 128a: A New Version of Grain 128 with Optional Authentication. International Journal of Wireless and Mobile Computing, 5(1):48–59, 2011. This is the journal version of [1].
3. S. Babbage and M. Dodd. The stream cipher MICKEY 2.0. ECRYPT Stream Cipher Project Report. Available at `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf`.
4. S. Banik, S. Maitra and S. Sarkar. A Differential Fault Attack on the Grain Family of Stream Ciphers. CHES 2012, LNCS Vol. 7428, pp. 122–139, 2012.
5. S. Banik, S. Maitra and S. Sarkar. A Differential Fault Attack on the Grain Family under Reasonable Assumptions. INDOCRYPT 2012, LNCS, Vol. 7668, pp. 191–208, 2012.
6. S. Banik and S. Maitra. A Differential Fault Attack on MICKEY 2.0. CHES 2013, LNCS, Vol. 8086, pp. 215–232, 2013.
7. S. Banik, S. Maitra and S. Sarkar. Improved Differential Fault Attack on MICKEY 2.0. Journal of Cryptographic Engineering. `http://link.springer.com/article/10.1007%2Fs13389-014-0083-9`, 2014
8. C. De Cannière and B. Preneel. TRIVIUM - a stream cipher construction inspired by block cipher design principles. eSTREAM, ECRYPT Stream Cipher Project.

9. P. Dey and A. Adhikari. Improved Multi-Bit Differential Fault Analysis of Trivium. INDOCRYPT 2014, LNCS, Vol. 8885, pp. 37–52, 2014.
10. P. Dey, A. Chakraborty, A. Adhikari and D. Mukhopadhyay. Multi-Bit Differential Fault Analysis of Grain 128 with Very Weak Assumptions. DATE 2015. Cryptology ePrint Archive, Report 2014/654, 2014.
11. S. Faust, P. Mukherjee, D. Venturi and D. Wichs. Efficient Non-Malleable Codes and Key-Derivation for Poly-Size Tampering Circuits. Cryptology ePrint Archive: Report 2013/702, http://eprint.iacr.org/2013/702, EUROCRYPT 2014, LNCS, Vol. 8441, pp. 111–128, 2014.
12. M. Hell, T. Johansson, A. Maximov and W. Meier. A Stream Cipher Proposal: Grain 128. http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain128_p3.pdf, 2005.
13. 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.
14. M. Hojsík and B. Rudolf. Differential Fault Analysis of Trivium. FSE 2008, LNCS, Vol. 5086, pp. 158–172, 2008.
15. J. Hong and W. Kim. TMD-Tradeoff and State Entropy Loss Considerations of stream cipher MICKEY. INDOCRYPT 2005, LNCS, Vol. 3797, pp. 169–182, 2005.
16. S. Karmakar and D. Roy Chowdhury. Differential Fault Analysis of MICKEY Family of Stream Ciphers. Cryptology ePrint Archive, Report 2014/262, 2014.
17. SAGE: Open Source Mathematics Software. Available at http://www.sagemath.org/.
18. S. Sarkar, S. Banik and S. Maitra. Differential Fault Attack against Grain family with very few faults and minimal assumptions. IEEE Transactions on Computers, 99(PrePrints):1, 2014.

Appendix A: Description of the Grain Family of Ciphers and MICKEY 2.0

Algebraic Description of the Grain Family

Consider $a_i, b_i, c_i \in \{0,1\}$ for $i \in \{0,\ldots,n-1\}$. Any cipher in the Grain family consists of an n-bit LFSR and an n-bit NFSR (see Figure 2). The update function of the LFSR is given by the equation $y_{t+n} = f(Y_t) = \sum_{i=0}^{n-1} c_i y_{t+i}$, where $Y_t = (y_t, \ldots, y_{t+n-1})$ is an n-bit vector that denotes the LFSR state at the t^{th} clock interval 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 $x_{t+n} = y_t + g(X_t) = y_t + g(x_t, \ldots, x_{t+n-1})$. Here, $X_t = (x_t, \ldots, x_{t+n-1})$ is an n-bit vector that denotes the NFSR state at the t^{th} clock interval and g is a non-linear function of the NFSR state bits. It may happen very well that g is degenerate on some of its variables, i.e., all the NFSR bits may not contribute in the function g.

At the t^{th} clock interval the η-bit state ($\eta = 2n$) of the cipher is denoted by $\mathcal{S}_t = (X_t, Y_t)$ i.e., $\mathcal{S}_t = (x_t, \ldots, x_{t+n-1}, y_t, \ldots, y_{t+n-1})$. Thus for any $j \in \{0,\ldots,\eta-1\}$, by the j^{th} state bit, we mean the j^{th} NFSR bit if $j \in \{0,\ldots,n-1\}$ and mean the $(j-n)^{th}$ LFSR bit if $j \in \{n,\ldots,2n-1\}$.

The key-stream is produced by combining the LFSR and NFSR bits as $z_t = \sum_{i=0}^{n-1} b_i y_{t+i} + \sum_{i=0}^{n-1} a_i x_{t+i} + h(\mathcal{S}_t)$, where h is a non-linear Boolean function, and may be degenerate on some of the variables.

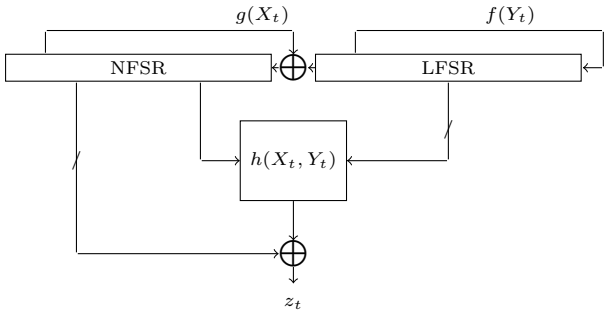


Fig. 2: Structure of Stream Cipher in Grain Family

	Grain v1	Grain 128	Grain 128a
n	80	128	128
m	64	96	96
Pad	FFFF	FFFFFFFF	FFFFFFFE
$f(\cdot)$	$y_{t+62} \oplus y_{t+51} \oplus y_{t+38} \oplus y_{t+23} \oplus y_{t+13} \oplus y_t$	$y_{t+96} \oplus y_{t+81} \oplus y_{t+70} \oplus y_{t+38} \oplus y_{t+7} \oplus y_t$	$y_{t+96} \oplus y_{t+81} \oplus y_{t+70} \oplus y_{t+38} \oplus y_{t+7} \oplus y_t$
$g(\cdot)$	$x_{t+62} \oplus x_{t+60} \oplus x_{t+52} \oplus x_{t+45} \oplus x_{t+37} \oplus x_{t+33} \oplus x_{t+28} \oplus x_{t+21} \oplus x_{t+14} \oplus x_t \oplus x_{t+63}x_{t+60} \oplus x_{t+37}x_{t+33} \oplus x_{t+15}x_{t+9} \oplus x_{t+60}x_{t+52}x_{t+45} \oplus x_{t+33} \oplus x_{t+28}x_{t+21} \oplus x_{t+63}x_{t+60} \oplus x_{t+21}x_{t+15} \oplus x_{t+63}x_{t+60} \oplus x_{t+52}x_{t+45}x_{t+37} \oplus x_{t+33} \oplus x_{t+28}x_{t+21}x_{t+15}x_{t+9} \oplus x_{t+52}x_{t+45}x_{t+37}x_{t+33} \oplus x_{t+28}x_{t+21}$	$y_t \oplus x_t \oplus x_{t+26} \oplus x_{t+56} \oplus x_{t+91} \oplus x_{t+96} \oplus x_{t+3}x_{t+67} \oplus x_{t+11}x_{t+13} \oplus x_{t+17}x_{t+18} \oplus x_{t+27}x_{t+59} \oplus x_{t+40}x_{t+48} \oplus x_{t+61} \oplus x_{t+65} \oplus x_{t+68}x_{t+84}$	$y_t \oplus x_t \oplus x_{t+26} \oplus x_{t+56} \oplus x_{t+91} \oplus x_{t+96} \oplus x_{t+3}x_{t+67} \oplus x_{t+11}x_{t+13} \oplus x_{t+17}x_{t+18} \oplus x_{t+27}x_{t+59} \oplus x_{t+40}x_{t+48} \oplus x_{t+61} \oplus x_{t+65} \oplus x_{t+68}x_{t+84} \oplus x_{t+88}x_{t+92}x_{t+93}x_{t+95} \oplus x_{t+22}x_{t+24}x_{t+25} \oplus x_{t+70}x_{t+78}x_{t+82}$
$h(\cdot)$	$y_{t+3}y_{t+25}y_{t+46} \oplus y_{t+3} y_{t+46}y_{t+64} \oplus y_{t+3}y_{t+46} \oplus x_{t+63} \oplus y_{t+25}y_{t+46}x_{t+63} \oplus y_{t+46}y_{t+64}x_{t+63} \oplus y_{t+3} y_{t+64} \oplus y_{t+46}y_{t+64} \oplus y_{t+64} \oplus x_{t+63} \oplus y_{t+25} \oplus x_{t+63}$	$x_{t+12}x_{t+95}y_{t+95} \oplus x_{t+12} y_{t+8} \oplus y_{t+13}y_{t+20} \oplus x_{t+95} y_{t+42} \oplus y_{t+60}y_{t+79}$	$x_{t+12}x_{t+95}y_{t+94} \oplus x_{t+12} y_{t+8} \oplus y_{t+13}y_{t+20} \oplus x_{t+95} y_{t+42} \oplus y_{t+60}y_{t+79}$
z_t	$x_{t+1} \oplus x_{t+2} \oplus x_{t+4} \oplus x_{t+10} \oplus x_{t+31} \oplus x_{t+43} \oplus x_{t+56} \oplus h$	$x_{t+2} \oplus x_{t+15} \oplus x_{t+36} \oplus x_{t+45} \oplus x_{t+64} \oplus x_{t+73} \oplus x_{t+89} \oplus y_{t+93} \oplus h$	$x_{t+2} \oplus x_{t+15} \oplus x_{t+36} \oplus x_{t+45} \oplus x_{t+64} \oplus x_{t+73} \oplus x_{t+89} \oplus y_{t+93} \oplus h$

Table 6: Exact description of the three ciphers following [1, 12].

Key Scheduling Algorithm (KSA). 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 0^{th} to the $(m-1)^{th}$ bits of the LFSR. The remaining m^{th} to $(n-1)^{th}$ bits of the LFSR are loaded with some fixed pad $P \in \{0,1\}^{n-m}$. Then, for the first $2n$ clocks, the key-stream bit z_t is XOR-ed to both the LFSR and NFSR update functions.

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

Note 1. For simplicity of analysis, we label the PRGA round as 0, from where the output keystream bit generation starts. Grain 128a supports two different modes of operation: with and without authentication and that depends on the IV, say $(v_0, \ldots, v_{m-1})$. Authentication is mandatory when $v_0 = 1$, and forbidden when $v_0 = 0$. Output keystream bits are generated at rounds $\{0, 1, 2, \ldots\}$ for Grain v1, Grain 128 and Grain 128a when authentication is forbidden. For Grain 128a when authentication is mandatory, output keystream bits are generated at rounds $\{0, 2, 4, \ldots\}$.

MAC Generation Algorithm (MGA) in Grain 128a. When Authentication is mandatory, Grain 128a [1, 2] considers generation of MAC. Here we follow the description given in [2].

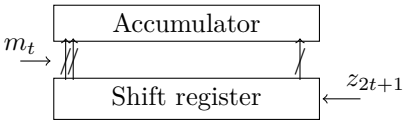


Fig. 3: Authentication

With $v_0 = 1$, the cipher picks every second keystream bit as output of the cipher after skipping the first 64 pre-output bits of PRGA rounds, i.e., in this case $z_0, z_2, z_4, \ldots$ is the generated keystream and the keystream bits $z_{-64}, z_{-63}, \ldots, z_{-1}$ and $z_1, z_3, z_5, \ldots$ are suppressed and are used for MAC generation.

Assume that we have a message of length L defined by the bits $m_0, \ldots, m_{L-1}$. 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 is denoted by $a_t^0, \ldots, a_t^{31}$ and $r_t, \ldots, r_{t+31}$ respectively. The accumulator is initialized through $a_0^j = z_j, -64 \leq j \leq -33$ and the shift register is initialized through $r_j = z_j, -32 \leq j \leq -1$. The shift register is updated as $r_{t+32} = z_{2t+1}$. The accumulator is updated as $a_{t+1}^j = a_t^j + m_t r_{t+j}$ for $0 \leq j \leq 31$ and $0 \leq t \leq L$. The final content of accumulator, $a_{L+1}^0, \ldots, a_{L+1}^{31}$ is used for authentication.

Brief Description of MICKEY 2.0

MICKEY 2.0 uses an 80-bit key and a variable length IV, the length of which may be between 0 and 80 bits. The physical structure of the cipher consists of two 100 bit registers R and S that exercise mutual control over each other's evolution. Let $r_0, r_1, r_2, \ldots, r_{99}$ denote the contents of the register R and $s_0, s_1, s_2, \ldots, s_{99}$ denote the contents of the register S. In order to describe the structure of the cipher and its working let us first define the following routines. Note that the description given here is based on [3].

Clocking register R Let $r_0, r_1, \ldots, r_{99}$ be the state of the register R before clocking, and let $r'_0, r'_1, \ldots, r'_{99}$ be the state of the register R after clocking. Define the integer array $RTAPS$ as follows

$$\begin{aligned} RTAPS = \{ & 0, 1, 3, 4, 5, 6, 9, 12, 13, 16, 19, 20, 21, 22, 25, 28, 37, 38, 41, 42, \\ & 45, 46, 50, 52, 54, 56, 58, 60, 61, 63, 64, 65, 66, 67, 71, 72, 79, 80, \\ & 81, 82, 87, 88, 89, 90, 91, 92, 94, 95, 96, 97\} \end{aligned}$$

Now define an operation

$$CLOCK\_R(R, INPUT\_BIT\_R, CONTROL\_BIT\_R)$$

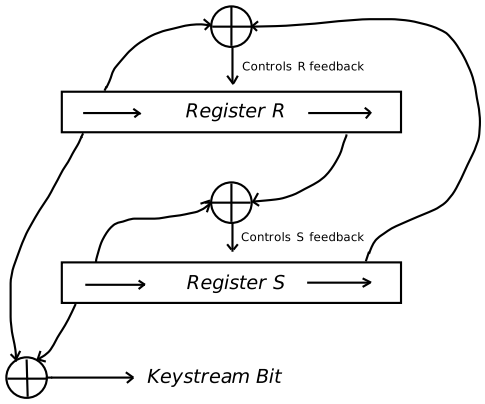


Fig. 4: The variable clocking architecture of MICKEY, as drawn in [3].

1. Define $FEEDBACK\_BIT = r_{99} + INPUT\_BIT\_R$
2. For $1 \leq i \leq 99 : r'_i = r_{i-1}$. $r'_0 = 0$.
3. For $0 \leq i \leq 99$: if $i \in RTAPS$, $r'_i = r'_i + FEEDBACK\_BIT$.
4. If $CONTROL\_BIT\_R = 1$:
   For $0 \leq i \leq 99 : r'_i = r'_i + r_i$

Clocking register *S* Let $s_0, s_1, \ldots, s_{99}$ be the state of the register S before clocking, and let $s'_0, s'_1, \ldots, s'_{99}$ be the state of the register S after clocking. Let $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{99}$ be intermediate variables. Define the four sequences $COMP0_i,\ 1 \leq i \leq 98$; $COMP1_i,\ 1 \leq i \leq 98$; $FB0_i,\ 0 \leq i \leq 99$ and $FB1_i,\ 0 \leq i \leq 99$ over GF(2) as in Table 7: Now define an operation

Table 7: The sequences $COMP0, COMP1, FB0, FB1$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$COMP0_i$		0	0	0	1	1	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0	1	0	1	0
$COMP1_i$		1	0	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	0	0	0	1	1	0	1
$FB0_i$	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1
$FB1_i$	1	1	1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0
i	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
$COMP0_i$	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1
$COMP1_i$	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	1
$FB0_i$	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1
$FB1_i$	0	1	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0
i	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
$COMP0_i$	0	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1
$COMP1_i$	0	0	0	1	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1
$FB0_i$	0	1	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
$FB1_i$	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1
i	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
$COMP0_i$	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	1	1	
$COMP1_i$	1	1	1	0	0	0	1	0	0	0	0	1	1	1	0	0	0	1	0	0	1	1	0	0	
$FB0_i$	1	1	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1	0	0	0
$FB1_i$	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1

$$CLOCK\_S(S, INPUT\_BIT\_S, CONTROL\_BIT\_S)$$

1. Define $FEEDBACK\_BIT = s_{99} + INPUT\_BIT\_S$
2. For $1 \leq i \leq 98 : \hat{s}_i = s_{i-1} + \big((s_i + COMP0_i) \cdot (s_{i+1} + COMP1_i)\big)$. $\hat{s}_0 = 0,\ \hat{s}_{99} = s_{98}$.
3. If $CONTROL\_BIT\_S = 0$:
   For $0 \leq i \leq 99$: $s'_i = \hat{s}_i + (FB0_i \cdot FEEDBACK\_BIT)$
   Else If $CONTROL\_BIT\_S = 1$:
   For $0 \leq i \leq 99$: $s'_i = \hat{s}_i + (FB1_i \cdot FEEDBACK\_BIT)$

The *CLOCK_KG* routine We define another operation

$$CLOCK\_KG(R, S, MIXING, INPUT\_BIT)$$

1. $CONTROL\_BIT\_R = s_{34} + r_{67}$, $CONTROL\_BIT\_S = s_{67} + r_{33}$
2. If $MIXING = 1$:
   $INPUT\_BIT\_R = INPUT\_BIT + s_{50}$
   Else If $MIXING = 0$:
   $INPUT\_BIT\_R = INPUT\_BIT$
3. $INPUT\_BIT\_S = INPUT\_BIT$
4. $CLOCK\_R(R, INPUT\_BIT\_R, CONTROL\_BIT\_R)$
5. $CLOCK\_S(S, INPUT\_BIT\_S, CONTROL\_BIT\_S)$

Working of the Cipher We will now describe the algorithm governing the functioning of the cipher. Let $K = k_0, k_1, \ldots, k_{79}$ be the 80 bit key used by the cipher. Let $IV = iv_0, iv_1, \ldots, iv_{v-1}$ be the v-bit IV ($0 \leq v \leq 80$). Then the cipher operates in the 4 stages as described below.

STAGE 1. IV loading
Initialize both R and S to the all-zero state.
For $0 \leq i \leq v-1$: $CLOCK\_KG(R, S, 1, iv_i)$

STAGE 2. Key loading
For $0 \leq i \leq 79$: $CLOCK\_KG(R, S, 1, k_i)$

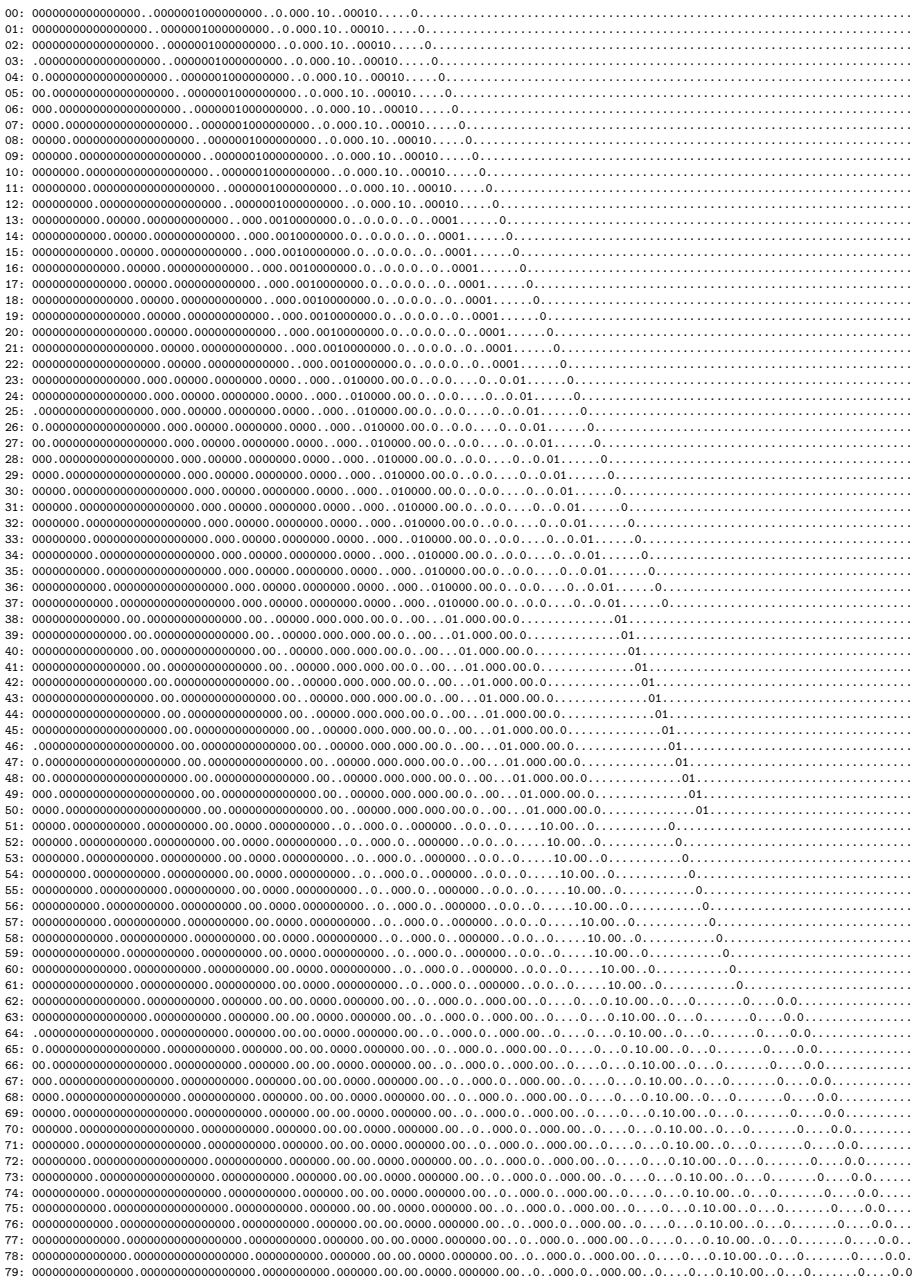
STAGE 3. Preclock Stage
For $0 \leq i \leq 99$: $CLOCK\_KG(R, S, 1, 0)$

STAGE 4. PRGA(Pseudo-Random stream generation algorithm)
$i \leftarrow 0$
While key-stream is required
$z_i = r_0 + s_0$
$CLOCK\_KG(R, S, 0, 0)$
$i \leftarrow i + 1$

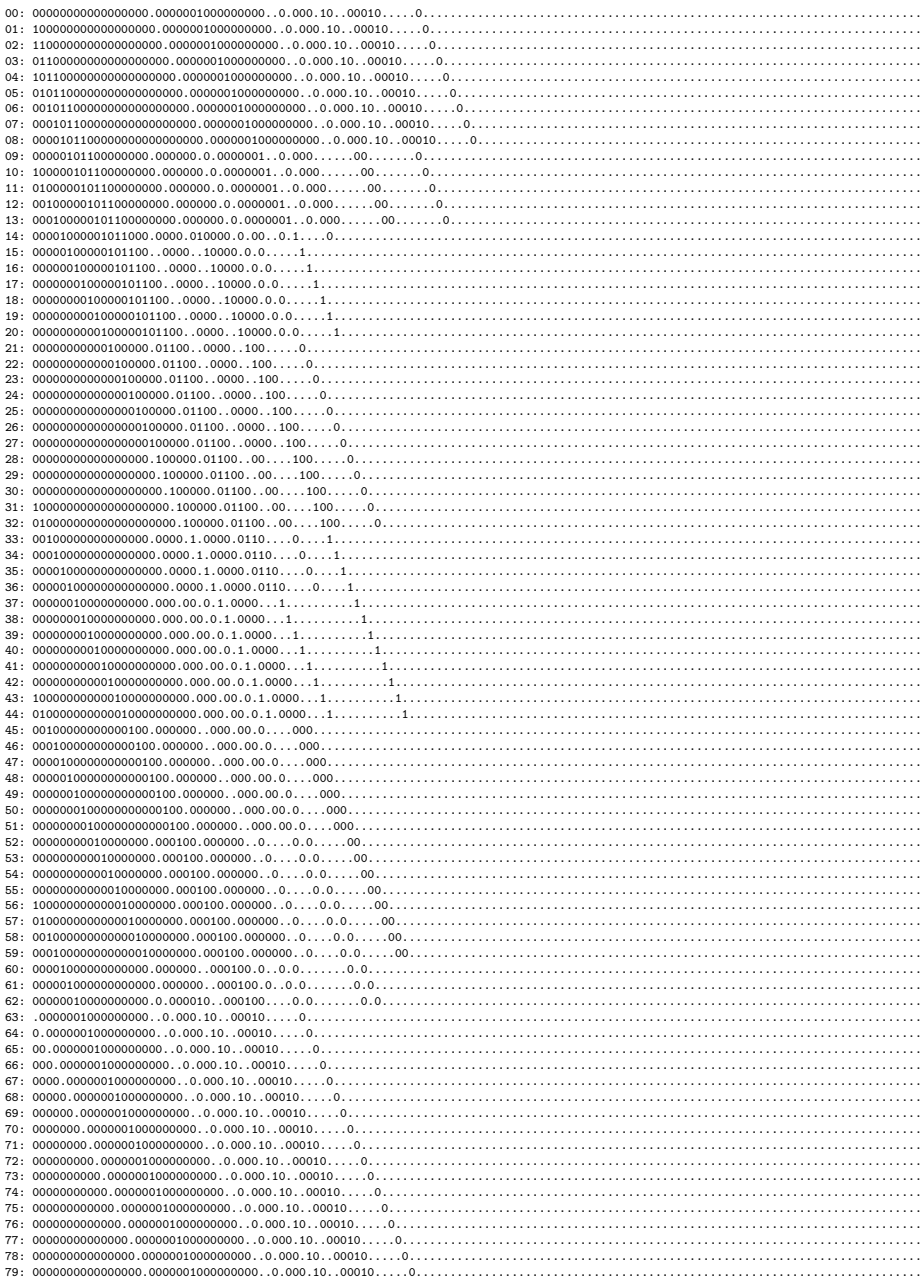
Appendix B: Single-bit faults in Grain v1

Signatures for single-bit faults in Grain v1

Faults in LFSR: Bits 0 to 79



Faults in NFSR: Bits 0 to 79



Appendix C: Illustrations

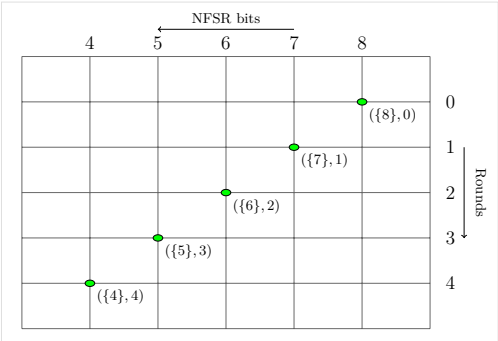


Fig. 5: Shifting of the faulty bit

	Grain v1	Grain 128	Grain 128a
NFSR tap bits	0, 1, 2, 4, 9, 10, 14, 15, 21, 28, 31, 33, 37, 43, 45, 52, 56, 60, 62, 63	0, 2, 3, 11, 12, 13, 15, 17, 18, 26, 27, 36, 40, 45, 48, 56, 59, 61, 64, 65, 67, 68, 73, 84, 89, 91, 95, 96	0, 2, 3, 11, 12, 13, 15, 17, 18, 22, 24, 25, 26, 27, 36, 40, 45, 48, 56, 59, 61, 64, 65, 67, 68, 70, 73, 78, 82, 84, 88, 89, 91, 92, 93, 95, 96
LFSR tap bits	0, 3, 13, 23, 25, 38, 46, 51, 62, 64	0, 7, 8, 13, 20, 38, 42, 60, 70, 79, 81, 93, 95, 96	0, 7, 8, 13, 20, 38, 42, 60, 70, 79, 81, 93, 94, 96

Table 8: NFSR and LFSR update and output taps