A Differential Fault Attack on the Grain Family of Stream Ciphers

Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar

Applied Statistics Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700 108, India
{s.banik_r,subho}@isical.ac.in, sarkar.santanu.bir@gmail.com

Abstract. In this paper we study a differential fault attack against the Grain family of stream ciphers. The attack works due to certain properties of the Boolean functions and corresponding choices of the taps from the LFSR. The existing works, by Berzati et al. (2009) and Karmakar et al. (2011), are applicable only on Grain-128 exploiting certain properties of the combining Boolean function h. That idea could not easily be extended to the corresponding Boolean function used in Grain v1. Here we show that the differential fault attack can indeed be efficiently mounted for the Boolean function used in Grain v1. In this case we exploit the idea that there exists certain suitable α such that $h(\mathbf{x}) + h(\mathbf{x} + \alpha)$ is linear. In our technique, we present methods to identify the fault locations and then construct set of linear equations to obtain the contents of the LFSR and the NFSR. As a countermeasure to such fault attack, we provide exact design criteria for Boolean functions to be used in Grain like structure.

Keywords: Fault Attacks, Countermeasures, Grain v1, Grain-128, Grain-128a, LFSR, NFSR, Stream Cipher.

1 Introduction

The Grain v1 stream cipher is in the hardware profile of the eStream portfolio [1] that has been designed by Hell, Johansson and Meier in 2005 [15]. It is a synchronous bit oriented stream cipher, although it is possible to achieve higher throughput at the expense of additional hardware. The physical structure of Grain is simple as well as elegant and it has been designed so as to require low hardware complexity. Following certain attacks on the initial design of the cipher, the modified versions Grain v1 [15], Grain-128 [16] and Grain-128a [2] were proposed after incorporating certain changes. Analysis of this cipher is an area of recent interest in as evident from numerous cryptanalytic results related to this family [3–5, 8–13, 19, 20, 22, 23, 27, 28].

Fault attacks are known to be very efficient against stream ciphers in general, and have received a lot of attention in recent cryptographic literature [6, 7, 17, 18, 21]. For differential fault attack scenario in stream ciphers, the attacker is allowed to inject faults in the internal state. Then by analyzing the difference in the faulty

E. Prouff and P. Schaumont (Eds.): CHES 2012, LNCS 7428, pp. 122–139, 2012.

and the fault-free keystreams, one should be able to deduce the complete or partial information about the internal state/secret key. The most common method of injecting faults is by using laser shots or clock glitches [24, 25]. Though the fault attacks usually rely on optimistic assumptions and study the cipher in a model that is weaker than the original version, they are not unrealistic as evident from literature. In this paper too, the model we study is a follow up of existing state-of-the-art literature [5, 19]. A detailed justification of the feasibility of such fault model is presented in [5, Section IIIB]. Before proceeding further, let us now present the fault model.

1. Similar to [5], we consider that the attacker is able to reset the system with the original Key-IV and start the cipher operations again. The work [19] requires a different assumption, where the IVs need to be modified in each initialization.
2. The attacker can inject a fault at any one random bit location of the LFSR or NFSR. As a result of the fault injection, the binary value in the bit-location (where the fault has been injected) is toggled. The attacker is not allowed to choose the location where he wants to inject the fault. However, as assumed in both [5, 19] the fault in any bit may be reproduced at any later stage of operation, once injected.
3. Similar to [5], we inject faults in the LFSR only, whereas the NFSR has been used for fault injection in [19].
4. The attacker has full control over the timing of fault injection, i.e., it is possible to inject the fault precisely at any stage of the cipher operation.

Our Contribution. Grain-128 has been successfully cryptanalyzed by employing fault attacks [5, 19]. However, Grain-v1 employs Boolean function of different kind, and thus such fault attacks may not immediately work against this cipher. In this paper we have tried to explore a generic fault attack on the structure of the Grain family of stream ciphers and thus in particular the idea works for Grain v1 too. The works presented in [5, 19] exploited the fact that the Boolean function g in Grain-128 is quadratic and the function h has only one cubic term other than the quadratic terms. This is not the scenario in Grain v1, where the Boolean functions are of more complicated structure in their Algebraic Normal Form. We point out that there are still problems in the choice of such functions as in Grain v1 [15] and suggest suitable choice instead of that one.

The novel idea of our fault attack is based on certain specific observations related to the output Boolean function h. For Grain v1, h is a 5-variable function with the differential property that $h(s_0, s_1, s_2, s_3, s_4) + h(1 + s_0, 1 + s_1, s_2, s_3, 1 + s_4) = s_2$. This helps us in determining one of the LFSR bits of the internal state and repeating this several times we get the complete LFSR state. Then we further note that h can be written as $s_4 \cdot u(s_0, s_1, s_2, s_3) + v(s_0, s_1, s_2, s_3)$, where $u(s_0, s_1, s_2, s_3) + u(s_0, 1 + s_1, s_2, 1 + s_3) = 1$. This helps us in determining the NFSR bits. To highlight our contribution in this paper, we like to refer to the following comment from [19]:

> "The attack may be extended to Grain-like ciphers with higher degree feedback functions and output functions. However, determining fault locations can be a challenging task if linear terms are removed from output bit expression. Higher degree feedback functions and output functions will however certainly increase the attack complexity as mostly nonlinear equations will be obtained."

We show that the complexity of the attack is not exactly related to the degree of the output functions. A q-variable Boolean function (say, $h : \{0,1\}^q \rightarrow \{0,1\}$) with high degree can also be attacked in a similar manner if there exists certain suitable $\alpha \in \{0,1\}^q$ such that $h(\mathbf{x}) + h(\mathbf{x} + \alpha)$ is linear. That is, higher degree functions may not increase the attack complexity as linear equations may actually be available using clever techniques instead of nonlinear ones.

An integral part of any fault attack is to identify the register location where the fault has been injected. We outline a novel technique of identifying the fault location in the LFSR by using an optimal length Signature vector technique.

We also point out that there exists a pool of 5-variable Boolean functions which are of matching parameters as proposed in [15] and possess additional properties that help in resisting the kind of differential fault attack that we describe here. That is, we present specific countermeasures to this kind of fault attack that relies on proper choice of the nonlinear combining Boolean function.

Organization of this paper. In this section, we proceed with the details of the Grain family (in particular Grain v1). Next, in Section 2, we present a broad description of the actual attack. The implementation of the attack on Grain v1 along with the fault location identification routine is explained in Section 3. The countermeasure corresponding to this attack with respect to proper choice of Boolean functions is described in Section 4. Section 5 concludes the paper.

We abuse the + notation for Boolean XOR, i.e., GF(2) addition as well as standard arithmetic addition. However, that will be clear from the context.

1.1 Brief Description of Grain Family

The exact structure of the Grain family is explained in Figure 1. It consists of an n-bit LFSR and an n-bit NFSR. Certain bits of both the shift registers are taken as inputs to a combining Boolean function, whence the keystream is produced. The update function of the LFSR is given by the equation $y_{t+n} = f(Y_t)$, where $Y_t = [y_t, y_{t+1}, \ldots, y_{t+n-1}]$ is an n-bit vector that denotes the LFSR state at 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 $x_{t+n} = y_t + g(X_t)$. Here, $X_t = [x_t, x_{t+1}, \ldots, x_{t+n-1}]$ 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.

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

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

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

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

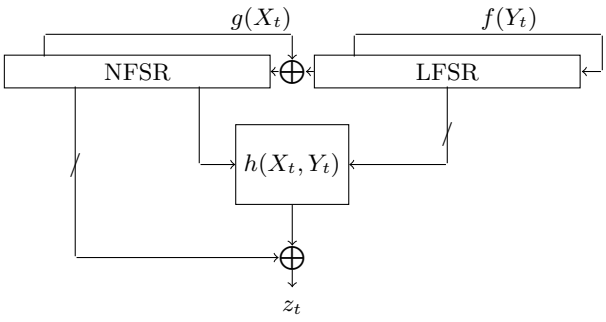

Fig. 1. Structure of Stream Cipher in Grain Family

One may note that given any arbitrary state and the information about its evolution (the number of clocks in KSA or PRGA), one can calculate the corresponding state S_0^K at the beginning of the KSA. This is because the state update functions in both the KSA and PRGA in the Grain family are one-to-one and invertible. Hence one can construct the KSA^{-1} routine that given an input $2n$ bit vector denoting the internal state of the cipher at the end of the KSA, returns the $2n$ bit vector giving internal state of the cipher at the beginning of the KSA. One can similarly describe a PRGA^{-1} routine that inverts one round of the PRGA.

As we will consider Grain v1 for our attack description, let us describe it now. In Grain v1, the size of Key is $n = 80$ bits and the IV is of size $m = 64$ bits. The pad used in the KLA is $P =$ `0xFFFF`. The LFSR update rule is given by $y_{t+80} = y_{t+62} + y_{t+51} + y_{t+38} + y_{t+23} + y_{t+13} + y_t$. The NFSR state is updated as

$x_{t+80} = y_t + g(x_{t+63}, x_{t+62}, x_{t+60}, x_{t+52}, x_{t+45}, x_{t+37}, x_{t+33}, x_{t+28}, x_{t+21}, x_{t+15}, x_{t+14}, x_{t+9}, x_t)$, where $g(x_{t+63}, x_{t+62}, x_{t+60}, x_{t+52}, x_{t+45}, x_{t+37}, x_{t+33}, x_{t+28}, x_{t+21}, x_{t+15}, x_{t+14}, x_{t+9}, x_t)$

$= x_{t+62} + x_{t+60} + x_{t+52} + x_{t+45} + x_{t+37} + x_{t+33} + x_{t+28} + x_{t+21} + x_{t+14} + x_{t+9}$
$+x_t + x_{t+63}x_{t+60} + x_{t+37}x_{t+33} + x_{t+15}x_{t+9} + x_{t+60}x_{t+52}x_{t+45} + x_{t+33}x_{t+28}x_{t+21}$
$+x_{t+63}x_{t+45}x_{t+28}x_{t+9} + x_{t+60}x_{t+52}x_{t+37}x_{t+33} + x_{t+63}x_{t+60}x_{t+21}x_{t+15}$
$+x_{t+63}x_{t+60}x_{t+52}x_{t+45}x_{t+37} + x_{t+33}x_{t+28}x_{t+21}x_{t+15}x_{t+9}$
$+x_{t+52}x_{t+45}x_{t+37}x_{t+33}x_{t+28}x_{t+21}$.

The output keystream is produced by combining the LFSR and NFSR bits as $z_t = \bigoplus_{a \in A} x_{t+a} + h(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_{t+63})$, where

$A = \{1, 2, 4, 10, 31, 43, 56\}$ and $h(s_0, s_1, s_2, s_3, s_4) = s_1 + s_4 + s_0s_3 + s_2s_3 + s_3s_4 + s_0s_1s_2 + s_0s_2s_3 + s_0s_2s_4 + s_1s_2s_4 + s_2s_3s_4$.

2 Broad Idea of the Generic Differential Fault Attack

In this section we will describe the generic fault attack idea on any cipher with the physical structure of the Grain family, i.e. a cipher in which there is an n-bit LFSR driving an n-bit NFSR. The LFSR and NFSR are being updated by feedback functions f, g respectively and the output keystream bit at each round is generated by an output function of the internal state, i.e., a function of certain locations from both the LFSR and the NFSR. The main nonlinear part of the output function is the 5-variable function h and we study this function carefully.

For this, let us first describe a few issues related to Boolean functions. The readers may have a look at [26] and the references therein for detailed background on Boolean functions. A q-variable Boolean function is a mapping from the set $\{0,1\}^q$ to the set $\{0,1\}$. Apart from the truth table, another important way to represent a Boolean function is by its Algebraic Normal Form (ANF). A q-variable Boolean function $h(x_1, \ldots, x_q)$ can be considered to be a multivariate polynomial over $GF(2)$. This polynomial can be expressed as a sum of products representation of all distinct k-th order products ($0 \leq k \leq q$) of the variables. More precisely, $h(x_1, \ldots, x_q)$ can be written as

$$a_0 + \bigoplus_{1 \leq i \leq q} a_i x_i + \bigoplus_{1 \leq i < j \leq q} a_{ij} x_i x_j + \cdots + a_{12\ldots q} x_1 x_2 \ldots x_q,$$

where the coefficients $a_0, a_{ij}, \ldots, a_{12\ldots q} \in \{0,1\}$. This is the ANF representation of h. The number of variables in the highest order product term with nonzero coefficient is called the *algebraic degree*, or simply the degree of h and denoted by $deg(h)$. Functions of degree at most one are called affine functions. Given the above background, let us present the following definition.

Definition 1. *Consider a q-variable Boolean function F. A non-zero vector $\alpha \in \{0,1\}^q$ is said to be an affine differential of F if $F(\mathbf{x}) + F(\mathbf{x} + \alpha)$ is an affine function. A Boolean function is said to be affine differential resistant if it does not have any affine differential.*

We propose to recover the secret key used in the cipher by observing and analyzing the difference between the fault-free and faulty keystreams. Our attack algorithm attempts to recover the internal state of the cipher after the completion of KSA (or equivalently when PRGA is about to begin). Since both the PRGA and the KSA of Grain family is completely invertible, one can then run the KSA^{-1} routine to determine the secret key K. As we have pointed out earlier, one can obtain a set of linear equations if there exist affine differentials corresponding to the function h and one such corresponding affine function should be on the variables that come from the locations of the LFSR only.

Given this background, the attack will comprise of the following steps:

1. The attacker is allowed to reset the cipher with the original Key-IV and restart cipher operations.
2. The attacker can inject a fault at any one random bit location of the LFSR. As a result of the fault injection, the binary value in the bit-location (where the fault has been injected) is toggled. The attacker is not allowed to choose the location of the LFSR where he wants to inject the fault. However, the fault in any LFSR bit may be reproduced at any later stage of operation, once injected.
3. Initially the attacker injects a fault (may be more than one in a few cases) in a randomly chosen position of the LFSR and identifies the fault location by comparing the original (fault-free) and faulty keystream.
4. The attacker has full control over the timing of fault injection, i.e., it is possible to inject the fault precisely at any stage of the cipher operation. Thus, knowing the fault location, (i) it is possible to restart the cipher operations with the original Key-IV, (ii) inject further faults in the same location (in our case either two or four faults) at specific PRGA rounds.
5. In this case, by comparing the original (fault-free) and faulty keystream in certain PRGA rounds, we obtain linear equations with respect to the LFSR state bits at the beginning of the PRGA. We run the fault attack suitable number of times so that we have several such linear equations and solving them we get the LFSR state. It is possible to obtain the linear equations (and thus to solve them efficiently) due to certain property of the Boolean function h.
6. Similarly as above, comparing the original (fault-free) and faulty keystream in certain other PRGA rounds, we obtain linear equations with respect to the NFSR bits at certain PRGA round and thus get back the NFSR state at the beginning of the PRGA. One can then run the KSA^{-1} routine to determine the secret key K.

In the next section we detail this algorithm with respect to Grain v1.

3 Differential Fault Analysis: Case Study with Grain v1

Our attack is generic and it works for any version of the Grain family. For Grain-128 our attack works in a similar broad framework as in [5], though the

exact details of the signatures, construction of linear equations and the way of exploiting the Boolean functions need to follow the method we describe below. Since the existing works [5, 19] will not work on Grain v1 due to comparatively complicated output function h, we concentrate on this version as a case study to explain our novel approach. Further, we would like to point out that to the best of our knowledge there is no existing fault attack on Grain v1 available in literature. Moreover, our attack strategy works on any generic Grain like structure and points out the importance of properly choosing the Boolean functions and the LFSR, NFSR locations that will be fed into the functions.

3.1 Obtaining the Location of the Fault

Our attack model assumes that the attacker is allowed to toggle the value at exactly one random location of the LFSR. The attacker, however can not explicitly choose the location where the fault is to be injected. In order for the attack to succeed, it is very important that it will be possible to identify the location of the LFSR where the fault has been induced.

Some Definitions and Notations. Let $S_0 \in \{0,1\}^{160}$ be the initial state of the Grain v1 PRGA, and S_{0,Δ_ϕ} be the initial state resulting after injecting fault in LFSR location $\phi \in [0,79]$. Let $Z = [z_0, z_1, \ldots, z_l]$ and $Z^\phi = [z_0^\phi, z_1^\phi, \ldots, z_l^\phi]$ be the first l keystream bits produced by S_0 and S_{0,Δ_ϕ} respectively. The task for the fault location identification routine is to determine the fault location ϕ by analyzing the difference between Z and Z^ϕ. Initially we have taken the value of $l = 80$. After describing the fault location identification strategy in detail, we will study the value of l more critically.

We define an 80 bit vector E_ϕ over GF(2) whose i^{th} element $E_\phi(i)$ is the logical XNOR (complement of XOR) of the i^{th} elements of Z and Z^ϕ, i.e., $E_\phi(i) = 1 + z_i + z_i^\phi$ (here + should be interpreted as $\oplus$). Since S_0 can have 2^{144} values (each arising from a different combination of the 80 bit key and 64 bit IV, rest 16 padding bits are fixed), each of these choices of S_0 may lead to different patterns of E_ϕ. The bitwise logical AND of all such vectors E_ϕ is denoted as the Signature vector Sgn_ϕ for the fault location ϕ.

The Sgn_ϕ Pattern. Note that whenever $Sgn_\phi(i)$ is 1 this implies that the i^{th} keystream bit produced by S_0 and S_{0,Δ_ϕ} is equal for all choices of S_0. Calculating the Signature vectors by this method is a computationally infeasible task. We will describe a method to calculate them efficiently as below.

For Grain v1, two initial states of the PRGA $S_0, S_{0,\Delta_{79}} \in \{0,1\}^{160}$ which differ only in the 79^{th} position of the LFSR, produce identical output bits in 68 specific positions among the initial 80 keystream bits produced during the PRGA. If an input differential is introduced in the 79^{th} LFSR position, then at all rounds numbered $k \in [0,79] \setminus \{15, 33, 44, 51, 54, 57, 62, 69, 72, 73, 75, 76\}$, the

difference exists in positions that do not provide input to the Boolean function h and hence at these clocks the keystream bit produced by the two states are essentially the same. At all other clock rounds the difference appears at positions which provide input to h. Hence the keystream produced at these clocks may be different. Following the explanation given above, we can write Sgn_{79} in hexadecimal notation, Sgn_{79} = `FFFE FFFF BFF7 EDBD FB27`, which has $80 - 12 = 68$ many 1's and rest 0's.

Generalizing the above idea, for two PRGA initial states $S_0, S_{0,\Delta_\phi} \in \{0,1\}^{160}$ which differ only in the ϕ^{th} LFSR location, an analysis of the differential trails shows that out of the first 80 keystream bits produced by them, the bits at a certain fixed rounds are guaranteed to be equal. Thus by performing the above analysis for all fault locations ϕ ($0 \leq \phi \leq 79$), it is possible to calculate all the Signature vectors. A table containing the vectors for each fault location ϕ is available in Table 1.

Table 1. Fault Signature Vectors Sgn_ϕ for $0 \leq \phi \leq 79$ in hexadecimal notation for Grain v1

ϕ	Sgn_ϕ	ϕ	Sgn_ϕ	ϕ	Sgn_ϕ	ϕ	Sgn_ϕ
0	FFFF 3F7F CB93 A080 0000	20	FFFF BEFF F3B7 F4A9 3808	40	FFFE DFFF B3EE ED31 3B40	60	FFFD FFBF EDEF F93A 7E52
1	FFFF 9FBF E5C9 D040 0000	21	FFFF DF7F F9DB FA54 9C04	41	FFFF 6FFF D9F7 7698 9DA0	61	FFFE FFDF F6F7 FC9D 3F29
2	FFFF CFDF F2E4 E820 0000	22	FFFF EFBF FCED FD2A 4E02	42	FFFF B7FF ECFB BB4C 4ED0	62	FFFF 7FEF DB7B F64E 9D90
3	7FFF E7EF F972 7410 0000	23	FFFF 77DF DE72 F694 2501	43	FFFF DBFF F67D DDA6 2768	63	FFFF BFF7 EDBD FB27 4EC8
4	BFFF F3F7 FCB9 3A08 0000	24	FFFF BBEF EF39 7B4A 1280	44	FFFF EDFF FB3E EED3 13B4	64	7FFF DFFB F6DE FD93 A764
5	DFFF F9FB FE5C 9D04 0000	25	7FFF DDF7 F79C BDA5 0940	45	FFFF F6FF FD9F 7769 89DA	65	BFFF EFFD FB6F 7EC9 D3B2
6	EFFF FCFD FF2E 4E82 0000	26	BFFF EEFB FBCE 5ED2 84A0	46	7FFF FB7F FECF BBB4 C4ED	66	DFFF F7FE FDB7 BF64 E9D9
7	F7FF FE7E FF97 2741 0000	27	DFFF F77D FDE7 2F69 4250	47	BFFF FDBF FF67 DDDA 6276	67	EFFF FBFF 7EDB DFB2 74EC
8	FBFF FF3F 7FCB 93A0 8000	28	EFFF FBBE FEF3 97B4 A128	48	DFFF FEDF FFB3 EEED 313B	68	F7FF FDFF BF6D EFD9 3A76
9	FDFF FF9F BFE5 C9D0 4000	29	F7FF FDDF 7F79 CBDA 5094	49	EFFF FF6F FFD9 F776 989D	69	FBFF FEFF DFB6 F7EC 9D3B
10	FEFF FFCF DFF2 E4E8 2000	30	FBFF FEEF BFBC E5ED 284A	50	F7FF FFB7 FFEC FBBB 4C4E	70	FDFF FF7F EFDB 7BF6 4E9D
11	FF7F FFE7 EFF9 7274 1000	31	FDFF FF77 DFDE 72F6 9425	51	FBFF 7FDB DFF2 74FC A40B	71	FEFF FFBF F7ED BDFB 274E
12	FFBF FFF3 F7FC B93A 0800	32	FEFF FFBB EFEF 397B 4A12	52	FDFF BFED EFF9 3A7E 5205	72	FF7F FFDF FBF6 DEFD 93A7
13	FFDF 7FF9 DBFA 549C 0400	33	FF7F FFDD F7F7 9CBD A509	53	FEFF DFF6 F7FC 9D3F 2902	73	FFBF FFEF FDFB 6F7E C9D3
14	FFEF BFFC EDFD 2A4E 0200	34	FFBF FFEE FBFB CE5E D284	54	FF7F EFFB 7BFE 4E9F 9481	74	FFDF FFF7 FEFD B7BF 64E9
15	FFF7 DFFE 76FE 9527 0100	35	FFDF FFF7 7DFD E72F 6942	55	FFBF F7FD BDFF 274F CA40	75	FFEF FFFB FF7E DBDF B274
16	FFFB EFFF 3B7F 4A93 8080	36	FFEF FFFB BEFE F397 B4A1	56	FFDF FBFE DEFF 93A7 E520	76	FFF7 FFFD FFBF 6DEF D93A
17	FFFD F7FF 9DBF A549 C040	37	FFF7 FFFD DF7F 79CB DA50	57	FFEF FDFF 6F7F C9D3 F290	77	FFFB FFFE FFDF B6F7 EC9D
18	FFFE FBFF CEDF D2A4 E020	38	FFFB 7FFE CFBB B4C4 ED00	58	FFF7 FEFF B7BF E4E9 F948	78	FFFD FFFF 7FEF DB7B F64E
19	FFFF 7DFF E76F E952 7010	39	FFFD BFFF 67DD DA62 7680	59	FFFB FF7F DBDF F274 FCA4	79	FFFE FFFF BFF7 EDBD FB27

Steps for Location Identification. As mentioned above, the task for the fault identification routine is to determine the value of ϕ given the vector E_ϕ. For any element $V \in \{0,1\}^l$ define the set $B_V = \{i : 0 \leq i < l,\ V(i) = 1\}$. Now define a relation $\preceq$ in $\{0,1\}^l$ such that for 2 elements $V_1, V_2 \in \{0,1\}^l$, we will have $V_1 \preceq V_2$ if $B_{V_1} \subseteq B_{V_2}$.

Now we check the elements in B_{E_ϕ}. By definition, these are the PRGA rounds i during which $z_i = z_i^\phi$. By the definition of Signature vector proposed above, we know that for the correct value of ϕ, $B_{Sgn_\phi} \subseteq B_{E_\phi}$ and hence $Sgn_\phi \preceq E_\phi$. So our strategy would be to search all the Signature vectors and formulate the candidate set $\Psi_0 = \{\psi : 0 \leq \psi \leq 79,\ Sgn_\psi \preceq E_\phi\}$. If $|\Psi_0|$ is 1, then the single element in Ψ_0 will give us the fault location ϕ. However, this may not necessarily be the case always. If Ψ_0 has more than one element, we will be unable to decide conclusively at this stage.

In such a scenario we reset the cipher with the original Key-IV and this time apply the fault at the same location ϕ at the beginning of the 80^{th} PRGA round and record the next 80 keystream bits $Z^{\phi}(80) = [z^{\phi}_{80}(80), z^{\phi}_{81}(80), \ldots, z^{\phi}_{159}(80)]$, where $z^{\phi}_{i}(t)$ denotes the i^{th} keystream bit produced due to a fault on LFSR location ϕ at the beginning of PRGA round t. Let the corresponding fault-free bits be denoted by $Z(80) = [z_{80}, z_{81}, \ldots, z_{159}]$. Now reformulate and recalculate the vector E_{ϕ} so that its i^{th} element is the logical XNOR of the i^{th} elements of $Z(80)$ and $Z^{\phi}(80)$. We now search over the Signature vectors in the candidate set Ψ_0 and narrow down the set of possible candidates to $\Psi_1 = \{\psi : \psi \in \Psi_0,\ Sgn_{\psi} \preceq E_{\phi}\}$. Clearly, $|\Psi_1| \leq |\Psi_0|$, and so if $|\Psi_1| = 1$ then the fault location ϕ is the single element in Ψ_1. If not, we repeat the above process for another round, i.e. reset and apply the fault at the PRGA round 160 etc. If after k rounds of this process, $|\Psi_{k-1}| = 1$, then the single element in Ψ_{k-1} gives us the desired location ϕ.

Length of Signature Vector. In the idea given above, we have considered the length of the signature vector $l = 80$. It may be noted that that fault identification routine is also possible if we increase or decrease the length of the signature vector. So what guidelines must be followed to choose an optimal signature length. Intuitively the following considerations seem to be useful.

1. The signature vector must be long enough so as to uniquely identify the fault location applying one or more faults.
2. The length of the signature vector must be such that the average number of faults required to identify the fault location can be minimized.

We shall see how each of the above considerations affect the choice of the length l of the Signature vector. For example, by simply looking at the Signature vectors (one may refer to Table 1), one can deduce for sure that l can not be less than or equal to 16, otherwise $\phi = 0, 1, 2, 19, 20, 21, 22, 23, 24, 41, 42, 43, 44, 45, 61, 62, 63$ will have the same Signature vectors. We will give a better bound on l in the following Lemma.

Lemma 1. *The LFSR fault location can not be uniquely identified if the length of the signature vector Sgn_{ϕ} is less than or equal to* 44.

Proof. Take $l = 44$. Studying the Signature vectors, one can check that $Sgn_{40} =$ `FFFE DFFF B3E` and $Sgn_{79} =$ `FFFE FFFF BFF`. Note that, for all locations $i \in [0, 43]$ such that $Sgn_{40}(i) = 1$, the value of $Sgn_{79}(i)$ is also 1. This implies that $Sgn_{40} \preceq Sgn_{79}$. Now consider the case with the fault location $\phi = 79$. Then by the definition of the signature vector we have $Sgn_{79} \preceq E_{\phi}$. Since $\preceq$ is a partial order on $\{0, 1\}^l$, this implies that $Sgn_{40} \preceq E_{\phi}$ and so whenever $\phi = 79$ the fault location identification routine will never be able to narrow down the set of possible candidates Ψ_k to only $\{79\}$ for any value of k. It is easy to check that the same argument holds for any $l < 44$. □

Whenever $l \geq 45$ a simple exhaustive search through the Signature vectors for all fault locations, will show that $Sgn_{\phi_1} \not\preceq Sgn_{\phi_2}$ for any two fault locations

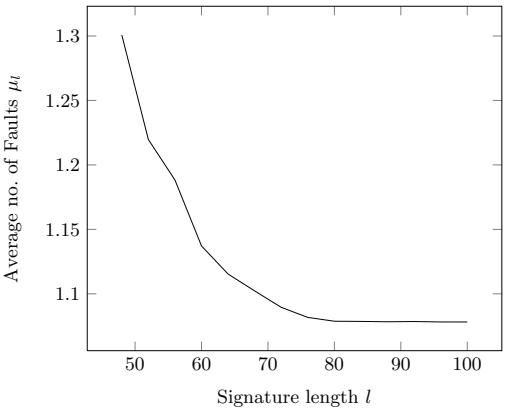

Fig. 2. Average number of faults vs Length of Signature

$0 \leq \phi_1 \neq \phi_2 \leq 79$. Further, we have to choose some $l \geq 45$ so that the average number of faults, for determining the fault location uniquely, can be minimized. Finding, this optimal value of l mathematically is a difficult task, and hence we choose to determine the optimal value by performing computer simulations. By taking the average over 2^{20} uniformly randomly chosen Key-IV pairs for Grain v1, for every signature length $l \geq 45$ we get the curve of Average number of faults μ_l vs Length of Signature l given in Figure 2.

We can see that after $l = 80$, $\mu_l = 1.08$ becomes almost constant for increasing values of l. So the length of the Signature vector has been chosen to be 80 bits.

A similar analysis for Grain-128 shows that the minimum Signature length must be greater than or equal to 62. For $l = 128$, the value of μ_l is around 1.001.

3.2 Determining the LFSR Internal State

Once the fault location ϕ has been identified we can proceed towards determining the LFSR internal state at the beginning of the PRGA. Depending on the value of ϕ we do one of the following.

- If $0 \leq \phi \leq 37$, we disregard the faulty keystream bits, and reset the cipher and look to hit another LFSR location.
- If $38 \leq \phi \leq 41$, we reset the cipher and apply faults at the location ϕ at the beginning of PRGA rounds $0, 20$ and record the faulty keystream bits at certain specific PRGA rounds. We then reset the cipher and look to hit another LFSR location.
- If $42 \leq \phi \leq 79$, we reset the cipher and apply faults at the location ϕ at the beginning of PRGA rounds $0, 20$ and record the faulty keystream bits at certain specific PRGA rounds. We reset the cipher again and apply faults at the location ϕ at the beginning of PRGA rounds $204, 224$ and record the

faulty keystream bits at certain other specific PRGA rounds. We then reset the cipher and look to hit another LFSR location
– We continue this process till all LFSR locations 38 to 79 have been hit.

We would like to point out that each double fault (injected at PRGA rounds $0, 20$ or $204, 224$) yields one linear equation in the initial LFSR state bits of the PRGA. By injecting 2 faults in the 4 LFSR locations 38 to 41 and 4 faults in the 38 LFSR locations 42 to 79, we obtain a set of 80 independent linear equations in the initial LFSR state bits, which can be solved to get the entire LFSR state at the start of the PRGA. The faulty keystream bits recorded in this phase will be again used to recover the NFSR internal state as will be explained in Section 3.3. Before describing the attack in detail let us state the following symbolic notations that we shall be using henceforth.

Some Notations

1. $S_t = [x_0^t, x_1^t, \ldots, x_{79}^t \quad y_0^t, y_1^t, \ldots, y_{79}^t]$ is used to denote the internal state of the cipher at the beginning of round t of the PRGA. Thus x_i^t (y_i^t) denotes the i^{th} NFSR (LFSR) bit at the start of round t of the PRGA. When $t = 0$, we use $S_0 = [x_0, x_1, \ldots, x_{79} \quad y_0, y_1, \ldots, y_{79}]$ to denote the internal state for convenience.
2. $S_t^{\phi}(t_1, t_2)$ is used to denote the internal state of the cipher at the beginning of round t of the PRGA, when a fault has been injected in LFSR location ϕ at the beginning of the t_1^{th} and the t_2^{th} PRGA round.
3. $z_i^{\phi}(t_1, t_2)$ denotes the keystream bit produced in the i^{th} PRGA round, after faults have been injected in LFSR location ϕ at the beginning of the t_1^{th} and the t_2^{th} PRGA round. z_i is the fault-free i^{th} keystream bit.

Beginning the Attack. We start by making the following observation about the output Boolean function h in Grain v1: $h(s_0, s_1, s_2, s_3, s_4) + h(1 + s_0, 1 + s_1, s_2, s_3, 1 + s_4) = s_2$. Hence h is not affine differential resistant. Note that s_0, s_1, s_2, s_3 correspond to LFSR locations $3, 25, 46, 64$ respectively and s_4 corresponds to the NFSR location 63. This implies that if two internal states S and S_{Δ} be such that they differ in LFSR locations $3, 25$ and NFSR location 63 and in no other location that contributes inputs to the output keystream bit, then the difference of the keystream bit produced by them will be equal to the value in LFSR location 46. Getting differentials at exactly these 3 locations may be difficult by injecting a single fault, but may be achieved if we faulted the same LFSR location twice, as will be explained by the following lemma.

Lemma 2. *If a fault is injected in LFSR location* $38 + r$ *(*$0 \leq r \leq 41$*), at the beginning of the PRGA rounds* λ *and* $\lambda + 20$ *(*$\lambda = 0, 1, \ldots$*), then in round number* $55 + \lambda + r$ *of the PRGA, the faulty internal state* $S_{55+\lambda+r}^{38+r}(\lambda, \lambda + 20)$ *and the fault-free internal state* $S_{55+\lambda+r}$ *will differ in LFSR locations* $3, 25$ *and NFSR location* 63 *and in none of the other* 9 *tap locations that contributes to the output keystream bit.*

Proof. The proof requires the analysis of the differential trail of the successive PRGA rounds. A differential Δ introduced in LFSR location $38+r$ $(0 \leq r \leq 41)$, at the beginning of rounds λ and $\lambda+20$ of the PRGA, will certainly reside on the LFSR locations $3, 25$ and NFSR location 63 at the beginning of round $55+\lambda+r$ of the PRGA. The differential also does not affect any other location involved in the computation of the output keystream bit in round $55+\lambda+r$. □

The above lemma implies that if $\lambda = 0$, i.e., if faults are injected at the beginning of the PRGA and round 20 at location $38+r,\ 0 \leq r \leq 41$ of the LFSR, then in the PRGA round $55+r$ we will have

$$z_{55+r} + z_{55+r}^{38+r}(0, 20) = y_{46}^{55+r} \quad \forall r \in [0, 41].$$

Now since the NFSR does not influence the LFSR during the PRGA, y_{46}^{55+r} is a linear function of the initial LFSR bits $y_0, y_1, \ldots, y_{79}$ for all $0 \leq r \leq 41$. For example, by analyzing the LFSR we have $y_{46}^{55} = y_3 + y_{16} + y_{21} + y_{26} + y_{34} + y_{41} + y_{44} + y_{54} + y_{59} + y_{65} + y_{72}$.

So in this process, we obtain 42 linear equations in the original LFSR bits $y_0, y_1, \ldots, y_{79}$. We need another 38 equations such that the resulting 80 equations are linearly independent. We have attempted to find the remaining 38 equations by resetting the cipher and then introducing faults later in the PRGA. If we let $\lambda = 204$, i.e., if double faults were introduced in LFSR locations $42+r$ with $0 \leq r \leq 37$ at the beginning of the PRGA rounds 204 and 224, then by the previous analysis it may be deduced that

$$z_{263+r} + z_{263+r}^{42+r}(204, 224) = y_{46}^{263+r} \quad \forall r \in [0, 37].$$

This provides us with another 38 equations. We have observed that these equations are linearly independent. Writing these equations in matrix notation, we have $LY = W$. The rows of the matrix L is defined by the linear functions $y_{46}^{55}, y_{46}^{56}, \ldots, y_{46}^{96}, y_{46}^{263}, \ldots, y_{46}^{300}$. Further, $Y = [y_0\ y_1\ \ldots\ y_{79}]^T$ and W is the column vector defined as follows

$$W(r) = z_{55+r} + z_{55+r}^{38+r}(0, 20) \quad 0 \leq r \leq 41,$$
$$W(42+r) = z_{263+r} + z_{263+r}^{42+r}(204, 224) \quad 0 \leq r \leq 37.$$

Since the matrix L and its inverse can be pre-computed beforehand, the vector $Y = L^{-1}W$ can be calculated immediately after applying the faults and calculating W.

Note that for the second round of fault injections the choice of fault locations $42 \leq \phi \leq 79$ and PRGA rounds $204, 224$ is by no means unique. By searching over various values of λ, one may be able to obtain a set of linearly independent equations for other choices of fault locations and PRGA rounds.

Remark 1. If the function h in Grain v1 had been affine differential resistant, then such linear equations could not have been formed. Instead one had to consider a set of nonlinear equations to get Y. As referred in [19], solving such nonlinear equations is more challenging task and in that case the fault attack we

describe here would have been less efficient. The method works in a similar manner for Grain-128 and Grain-128a. For example, the output function in Grain-128 is of the form $h(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) = s_0s_1 + s_2s_3 + s_4s_5 + s_6s_7 + s_0s_4s_8$, where s_0 and s_4 corresponds to NFSR variables. One can check that for any $\alpha \in \{001000000, 000100000, 000000100, 000000010\}$, $h(\mathbf{x}) + h(\mathbf{x} + \alpha)$ is a linear function of LFSR variables only.

3.3 Determining the NFSR Internal State

Once the LFSR internal state of the initial PRGA round is known, one can then proceed to determine the NFSR internal state. In [4] it was shown, that this could have been done efficiently for the initial version of the cipher i.e. Grain v0. After the attack in [4] was reported, the designers made the necessary changes to Grain v1, Grain-128 and Grain-128a so that for these new ciphers, determining the NFSR state form the knowledge of the LFSR state was no longer straightforward. In order to determine the NFSR bits, we look into the decomposition of the Boolean function h in more detail. The attack we will describe in this section can be mounted due to the following observations on the Grain output function h.

A. $h(\cdot)$ can be written in the form $s_j \cdot u(\cdot) + v(\cdot)$ where s_j corresponds to a variable which takes input from an NFSR tap location;
B. There exists a differential β such that $u(\mathbf{s}) + u(\mathbf{s} + \beta) = 1$;
C. $v(\mathbf{s}) + v(\mathbf{s} + \beta)$ = a function of variables that takes input from LFSR locations only.

For Grain v1, $h(s_0, s_1, s_2, s_3, s_4) = s_4 \cdot u(s_0, s_1, s_2, s_3) + v(s_0, s_1, s_2, s_3)$, where $u(s_0, s_1, s_2, s_3) = 1 + s_3 + s_0s_2 + s_1s_2 + s_2s_3$, and $v(s_0, s_1, s_2, s_3) = s_1 + s_0s_3 + s_2s_3 + s_0s_1s_2 + s_0s_2s_3$. Thus we note that (i) u, v are functions on the LFSR bits only, (ii) $u(s_0, s_1, s_2, s_3) + u(s_0, 1 + s_1, s_2, 1 + s_3) = 1$ and (iii) $v(s_0, s_1, s_2, s_3) + v(s_0, 1 + s_1, s_2, 1 + s_3) = 1 + s_0 + s_2$. Hence h satisfies all the properties listed above.

The fault-free keystream bit at the t^{th} round can now be rewritten as $z_t = \bigoplus_{a \in A} x_a^t + x_{63}^t \cdot u(y_3^t, y_{25}^t, y_{46}^t, y_{64}^t) + v(y_3^t, y_{25}^t, y_{46}^t, y_{64}^t)$. Consider two internal states S_t and $S_{t,\Delta}$ which differ in the LFSR locations 25 and 64 and in no other location, that provides input to h. If z_t and $z_{t,\Delta}$ are the keystream bits produced by S_t and $S_{t,\Delta}$ in that round, then using the previous observation we can see that

$$z_t + z_{t,\Delta} = x_{63}^t + v(y_3^t, y_{25}^t, y_{46}^t, y_{64}^t) + v(y_3^t, 1 + y_{25}^t, y_{46}^t, 1 + y_{64}^t).$$

Let $c_t = \left[v(y_3^t, y_{25}^t, y_{46}^t, y_{64}^t) + v(y_3^t, 1 + y_{25}^t, y_{46}^t, 1 + y_{64}^t)\right]$. Since the LFSR internal state is already available, c_t can be computed immediately, and hence the difference of the two keystream bits plus the value of c_t gives us the value at the NFSR location 63 at round t of the PRGA. In the next Lemma, we shall investigate when this differential pattern in the internal state is obtained by employing the same fault injection strategy in the previous subsection.

Lemma 3. *Let $S_0, S_1, S_2, \ldots$ be the successive internal states of the PRGA for Grain v1. Then the faulty state $S_t^{\phi}(0, 20)$ will differ from S_t at LFSR locations $25, 64$ and none of the other 10 tap locations that feed the output function for the following values of ϕ, t: (i) $\phi = 51 + r$, $t = 91 + r$ for $0 \leq r \leq 28$, (ii) $\phi = 62 + r$, $t = 55 + r$ for $0 \leq r \leq 17$, (iii) $\phi = 62 + r$, $t = 75 + r$ for $0 \leq r \leq 15$.*

Proof. The proof follows from an analysis of the differential trails of Grain v1 PRGA, and is similar to the proof for Lemma 2. □

The Lemma essentially implies that if faults are injected at the beginning of the PRGA and round 20 at location $51 + r$ of the LFSR ($0 \leq r \leq 28$), then in the PRGA round $91 + r$ we will have

$$z_{91+r} + z_{91+r}^{51+r}(0, 20) + c_{91+r} = x_{63}^{91+r} \quad \forall r \in [0, 28].$$

Also, the following equations hold:

$$z_{55+r} + z_{55+r}^{62+r}(0, 20) + c_{55+r} = x_{63}^{55+r} \quad \forall r \in [0, 17],$$

$$z_{75+r} + z_{75+r}^{62+r}(0, 20) + c_{75+r} = x_{63}^{75+r} \quad \forall r \in [0, 15].$$

Since the LHS of all the above equations are known, we can therefore calculate the value of the NFSR location 63 for all PRGA rounds 55, 56, ..., 72, 75, 76, ..., 119. Because of the shifting property of the NFSR, the equations $x_i^j = x_{i-1}^{j+1} \quad \forall i \in [1, 79]$ hold. Therefore knowing $x_{63}^{55}, x_{63}^{56}, \ldots, x_{63}^{72}, x_{63}^{75}, x_{63}^{76}, \ldots, x_{63}^{119}$ is equivalent to knowing $x_{15}^{103}, x_{16}^{103}, \ldots, x_{32}^{103}, x_{35}^{103}, x_{36}^{103}, \ldots, x_{79}^{103}$, i.e., we now know 63 out of the 80 NFSR state bits of S_{103}.

Finding the Remaining Bits. Any bits of the NFSR internal state not found out in the previous subsection could be obtained by performing an exhaustive search over them. However, if h is such that both u, v are functions on the LFSR bits only then the attack can be further simplified. Since the function h in Grain v1 satisfies this property, we proceed to determine the remaining 17 NFSR bits of S_{103}. These may be found by a combination of solving equations and guesswork. Since the 80 LFSR bits of S_0 have already been found in the previous section, one can efficiently calculate the 80 LFSR bits of S_{103} by running the Grain v1 PRGA routine. This is because the LFSR evolves independently during the PRGA. Then, by observing the fault-free output keystream bits we can write the following equations:

$$z_{102+\gamma} = x_{0+\gamma}^{103} + x_{1+\gamma}^{103} + x_{3+\gamma}^{103} + x_{9+\gamma}^{103} + x_{30+\gamma}^{103} + x_{42+\gamma}^{103} + x_{55+\gamma}^{103} + u_{102+\gamma} x_{62+\gamma}^{103} + v_{102+\gamma},$$

for $\gamma = 0, 1, \ldots, 14$, where $u_i = u(y_3^i, y_{25}^i, y_{46}^i, y_{64}^i)$ and $v_i = v(y_3^i, y_{25}^i, y_{46}^i, y_{64}^i)$. Since the LFSR initial state is known, u_i, v_i are available. Consider the set of 15 equations given above. In the last equation it can be seen that x_{14}^{103} is the only unknown and hence its value may be easily calculated. Once x_{14}^{103} is known, x_{13}^{103} becomes the only unknown in the 14^{th} equation and its value too may be

immediately calculated. Backtracking in this manner one can calculate upto x_5^{103} from the 6^{th} equation. At this point we have calculated the value of 73 NFSR bits of S_{103}. The 5^{th} equation is

$$z_{106} = x_4^{103} + x_5^{103} + x_7^{103} + x_{13}^{103} + x_{34}^{103} + x_{46}^{103} + x_{59}^{103} + u_{106}x_{66}^{103} + v_{106}.$$

This equation has two unknowns x_4^{103} and x_{34}^{103} and so the value of either unknown can not be calculated conclusively. Similarly the 4^{th} equation has two unknowns x_3^{103} and x_{33}^{103}. If we try out all the possibilities of $x_{34}^{103}, x_{33}^{103}$ then the value of the remaining 5 unknowns may be calculated uniquely. So we do an exhaustive search over the 2 bits (4 possible candidates) for S_{103}. The correct S_{103} may be found out by observing the keystream bits $z_{103}, z_{104}, \ldots$, as required. We eliminate any candidate S_{103} vector that does not produce the required keystream bit sequence. This routine thus gives us the entire S_{103} vector. Note that in order to recover the NFSR state one does not have to inject any additional faults other than those already injected to determine the LFSR state.

Remark 2. If the function h in Grain v1 were such that it could not be decomposed into u and v as above, then the attack would not have been as straightforward. The attack here is efficient as u and v are of certain nice structures and their inputs are from LFSR bits only. The LFSR bits are already known after the recovery of the LFSR bits and that helps in recovering the NFSR state easily. It can be checked that the output function of Grain-128 and Grain-128a also follows properties **(A), (B), (C)** given at the beginning of this section and thus renders them vulnerable to this attack.

3.4 Finding the Secret Key and Complexity of the Attack

It is known that the KSA and PRGA routines in the Grain family are invertible. Once we have all the bits of S_{103}, by running the inverse PRGA routine 103 times, we obtain the initial PRGA state S_0. Thereafter, by running the inverse KSA routine one can recover the secret key.

The attack complexity directly depends on the number of fault experiments to be performed such that all of locations in $[38, 79]$ of the LFSR are covered. To have this, the expected number of fault experiments is $80 \cdot \sum_{i=1}^{42} \frac{1}{i} \approx 344$. In each fault experiment, the fault identification routine requires μ_l faults and simulation results show that the expected value of μ_l is 1.08. Further depending on the LFSR location hit, during the attack phase, one needs to inject 2 or 4 extra faults for determining the internal state. Therefore, the expected number of faults that our attack needs is $344 \times (1.08) + 4 \times 38 + 2 \times 4 \approx 2^{9.05}$.

To determine the internal state, we have to perform one matrix multiplication, and solve a set of 78 linear equations and then exhaustively search over 2 variables. After that, 103 invocations of the PRGA^{-1} routine and a single invocation of the KSA^{-1} routine are needed to determine the Secret Key.

Thus the dominant time/memory consuming process in our attack is the multiplication of $L^{-1}W$ which requires around 80×80 bits to store L^{-1} and

$80^2 \approx O(2^{12.6})$ bit operations to calculate the product. Further storing the Sgn_ϕ patterns also requires 80×80 bits as described in Table 1.

As stated before, this is the first reported fault attack on Grain v1. Two fault attacks [5, 19] have been reported against Grain-128 and that is the reason direct comparison is not possible. However, one may note that our resource requirements are either favorable or comparable to that of [5, 19].

4 Countermeasure: Choice of Proper Boolean Function

In [5], it was suggested that one of the methods to prevent such fault attacks was to keep two identical implementations of both the shift registers in the cipher hardware. Naturally this needs additional hardware.

One important question here is what are the reasons such that the fault attack can be efficiently implemented. We have already seen that the source of the weakness lies with the output Boolean function h. Our attack is possible as there exists the vector $\alpha = [1, 1, 0, 0, 1]$ such that $h(\mathbf{s}) + h(\mathbf{s} + \alpha)$ is an affine Boolean function. This function h, used in Grain v1, is clearly not affine differential resistant. In [15], the designers clearly specify the reasons for choosing this particular output function.

> "This filter function is chosen to be balanced, correlation immune of the first order and has algebraic degree 3. The nonlinearity is the highest possible for these functions, namely 12."

In view of the fault attack presented here, we need affine differential resistant functions with the same parameters. One may refer to [26] to have many such functions in the class of rotation symmetric Boolean functions and we describe the ANF of one of those as below:

$F(s_0, s_1, s_2, s_3, s_4) = s_0s_1 + s_1s_2 + s_2s_3 + s_3s_4 + s_4s_0 + s_0s_2 + s_1s_3 + s_2s_4 + s_3s_0 + s_4s_1 + s_0s_1s_3 + s_1s_2s_4 + s_2s_3s_0 + s_3s_4s_1 + s_4s_0s_2$. This function can be realized with a few extra logic gates as below. The gate count is presented as per the calculation of [15].

	Gate Requirement					
	NAND2	NAND3	NAND4	NAND5	NAND6	Gate Count
h [15]	8	1	9	1	0	30
F (our)	8	0	10	2	1	38

Proper cryptographic choice of h with possibly higher number of variables with efficient implementation in terms of low gate counts is an important open question. Further, we should also note that the decomposition of h in u, v that possess properties **(A), (B), (C)** given in Section 3.3 helps in mounting an efficient fault attack. We further note that the function F described above does not satisfy property **(B)** if s_4 is the only variable that takes input from an NFSR location. This implies that even if the initial LFSR state of the PRGA is made known to the attacker, the attacker will be unable to apply the attack given in Section 3.3 to the function F.

5 Conclusion

In this paper we have described a differential fault attack that works on all the versions of Grain. Such attacks were studied earlier on Grain-128 in [5, 19]. However, the attacks could not be mounted on Grain v1 due to the different structure of the output function $h(\cdot)$. Here we show that the function of Grain v1 too has some weakness in terms of having affine differentials. By this we mean that there exists certain suitable α such that $h(\mathbf{x})+h(\mathbf{x}+\alpha)$ is linear. Our attack works due to this observation and corresponding choices of the taps from the LFSR. That is, from a general perspective, the differential fault attack can be mounted on Grain like structures even with Boolean functions of higher degree. We also provide examples of functions that are affine differential resistant and suggest use of such functions in Grain family as a countermeasure. Our work provides clear direction in choosing the output Boolean function and its inputs from the locations of the LFSR and the NFSR.

Acknowledgments The authors like to thank the Centre of Excellence in Cryptology, Indian Statistical Institute for relevant support towards this research.

Reference

1. The ECRYPT Stream Cipher Project. eSTREAM Portfolio of Stream Ciphers (revised on September 8, 2008)
2. Ågren, M., Hell, M., Johansson, T., Meier, W.: A New Version of Grain-128 with Authentication. In: Symmetric Key Encryption Workshop 2011. DTU, Denmark (2011)
3. 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. In: SHARCS - Special-purpose Hardware for Attacking Cryptographic Systems (2009)
4. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of Grain. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 15–29. Springer, Heidelberg (2006)
5. Berzati, A., Canovas, C., Castagnos, G., Debraize, B., Goubin, L., Gouget, A., Paillier, P., Salgado, S.: Fault Analysis of Grain-128. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 7–14 (2009)
6. Berzati, A., Canovas-Dumas, C., Goubin, L.: Fault Analysis of Rabbit: Toward a Secret Key Leakage. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 72–87. Springer, Heidelberg (2009)
7. Blömer, J., Seifert, J.P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
8. Bjørstad, T.E.: Cryptanalysis of Grain using Time/Memory/Data tradeoffs, v1.0 (February 25, 2008), `http://www.ecrypt.eu.org/stream`.
9. De Cannière, C., Küçük, Ö., Preneel, B.: Analysis of Grain's Initialization Algorithm. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 276–289. Springer, Heidelberg (2008)

10. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 327–343. Springer, Heidelberg (2011)
11. Dinur, I., Shamir, A.: Breaking Grain-128 with Dynamic Cube Attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)
12. Englund, H., Johansson, T., Turan, M.S.: A Framework for Chosen IV Statistical Analysis of Stream Ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
13. Fischer, S., Khazaei, S., Meier, W.: Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
14. Fredricksen, H.: A survey of full length nonlinear shift register cycle algorithms. SIAM Rev. 24, 195–221 (1982)
15. Hell, M., Johansson, T., Meier, W.: Grain - A Stream Cipher for Constrained Environments. ECRYPT Stream Cipher Project Report 2005/001 (2005), http://www.ecrypt.eu.org/stream
16. Hell, M., Johansson, T., Maximov, A., Meier, W.: A Stream Cipher Proposal: Grain-128. In: IEEE International Symposium on Information Theory, ISIT 2006 (2006)
17. Hoch, J.J., Shamir, A.: Fault Analysis of Stream Ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004)
18. Hojsík, M., Rudolf, B.: Differential Fault Analysis of Trivium. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 158–172. Springer, Heidelberg (2008)
19. Karmakar, S., Roy Chowdhury, D.: Fault Analysis of Grain-128 by Targeting NFSR. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 298–315. Springer, Heidelberg (2011)
20. Khazaei, S., Hassanzadeh, M., Kiaei, M.: Distinguishing Attack on Grain. ECRYPT Stream Cipher Project Report 2005/071 (2005), http://www.ecrypt.eu.org/stream
21. Kircanski, A., Youssef, A.M.: Differential Fault Analysis of Rabbit. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 197–214. Springer, Heidelberg (2009)
22. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSR-based Cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
23. Lee, Y., Jeong, K., Sung, J., Hong, S.: Related-Key Chosen IV Attacks on Grain-v1 and Grain-128. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 321–335. Springer, Heidelberg (2008)
24. Skorobogatov, S.P.: Optically Enhanced Position-Locked Power Analysis. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 61–75. Springer, Heidelberg (2006)
25. Skorobogatov, S.P., Anderson, R.J.: Optical Fault Induction Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)
26. Stanica, P., Maitra, S.: Rotation symmetric Boolean functions - Count and cryptographic properties. Discrete Applied Mathematics (DAM) 156(10), 1567–1580 (2008)
27. Stankovski, P.: Greedy Distinguishers and Nonrandomness Detectors. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 210–226. Springer, Heidelberg (2010)
28. Zhang, H., Wang, X.: Cryptanalysis of Stream Cipher Grain Family. IACR Cryptology ePrint Archive 2009: 109 (2009), http://eprint.iacr.org/2009/109