# Fault Analysis of Grain Family of Stream Ciphers

Sandip Karmakar (sandip1kk@gmail.com) and Dipanwita Roy Chowdhury (drc@cse.iitkgp.ernet.in)
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, India

***Abstract*—In this paper, we present fault attack on Grain family of stream ciphers, an eStream finalist. The earlier fault attacks on Grain work on LFSR whereas our target for fault induction is the NFSR. Our attack requires a small number of faults to be injected; 150 only for Grain v1 and only 312 and 384 for Grain-128 and Grain-128a, respectively. The number of faults are much lesser than the earlier reported fault attacks; 1587 for Grain-128 and 1831 for Grain-128a.**

Keywords: Stream Cipher, Grain-128, Grain v1, Grain 128a, Side Channel Attack, Fault Attack, NFSR Fault Attack

## I. Introduction

Fault attack is a practical form of side channel attack. Here, the adversary induces faults in states or operations of the cipher and analyze fault-free and faulty ciphertexts/keystreams to break the system. Both block ([10], [11]) and stream ([16], [17], [18]) ciphers have been shown *very* weak under fault analysis. However, practically inducing faults according to certain models are still challenging and widely researched area, with few successes ([19], [20]).

The final portfolio of eStream [1] includes three hardware-efficient and four software-efficient stream ciphers [3]. Grain [13] proposed by Martin Hell et al. is one of the three hardware based ciphers enlisted in the portfolio. A mathematical attack based on dynamic cube attack [12] and few fault attacks ([4], [7], [8], [9]) are only known weaknesses of Grain. These attacks employ a reasonable fault model, induces faults cipher state and analyses fault-free and faulty keystreams to deduce secret $key$ of the cipher using small number of faults and few minutes of computation in practical scenario.

The reported fault analysis ([4], [7], [8], [9]) target LFSR of Grain. The first fault attack on Grain family targeting LFSR was [4], which targets Grain-128. While [7], [8], [9] extends this work to other ciphers of the Grain family, Grain v1 and Grain128a targeting LFSR. In [9] the authors have discussed a few modifications so as to adapt these attacks under less restricted assumptions, such as incorporating multiple bit faults etc.

In this paper, faults are assumed to be induced at the NFSR of Grain. As already mentioned most of the research on fault analysis of Grain family of stream ciphers targets LFSR. This opens the question if only protecting LFSR will secure it from fault attacks. We show that the NFSR needs also be protected. Since, this means that both LFSR and NFSR can be targeted, the adversary essentially works on a less restricted fault model. LFSR and NFSR targeted attacks need to be modified to make it work on the entire state. We follow certain algorithms to deduce secret $key$ of the cipher following fault induction respecting our assumed model. The attack is applied and reported here against Grain v1, Grain-128 and Grain-128a. We require reasonable number of faults and solution of at most $2^{nd}$ degree equations (mostly linear) to break the systems. A preliminary version of this work is published in [6] which analyzes Grain-128 only, the complete fault analysis of all the members of Grain family, Grain v1, Grain-128 and Grain-128a is presented in this paper.

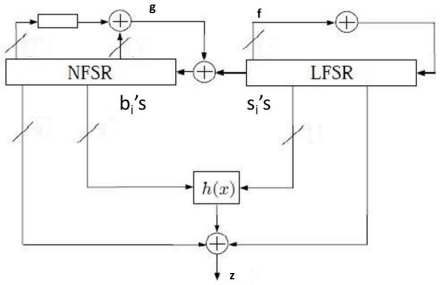


Fig. 1. Design of Grain family of ciphers

This paper is organized as follows. Following this introduction, section $II$ briefly discusses the specification of Grain family. We present our fault model in section $III$. The proposed attack against Grain family is presented in section $IV$. Section $V$ summarizes performance analysis. Finally, section $VI$ concludes the paper.

## II. Background

In this section, we briefly discuss the specification of the Grain family of stream ciphers.

Grain family of stream ciphers were introduced with Grain v1 ([14]). Grain-128 ([13]) is the latest member of the family.

The structure of Grain family of stream ciphers is shown in figure 1. Two left shift registers one linear, LFSR and one nonlinear, NFSR of equal length, $n$ store internal state of the cipher. The LFSR and NFSR of Grain v1 is of 80 bits, while both Grain-128 and Grain-128a are of 128 bits. The LFSR and NFSR are updated by linear function $f$ and nonlinear function $g$, respectively. A nonlinear filter function $h$ along with few linear terms, both defined with input bits from both NFSR and LFSR produces the output keystream $z_i$ at $i^{th}$ cycle. Throughout this paper $+$ refers to $+$ modulo 2.

For Grain v1 these functions are defined as,

$$\begin{aligned} f &= s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} \\ &\quad + s_{i+13} + s_i \\ g &= s_i + b_{i+62} + b_{i+60} + b_{i+52} + b_{i+45} \\ &\quad + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+14} \end{aligned} \tag{1}$$

$$
\begin{aligned}
&+b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + \\
&b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} \\
&+b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} \\
&+b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \\
&+b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \\
&+b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21} \qquad (2)
\end{aligned}
$$

$$
\begin{aligned}
h &= s_{i+25} + b_{i+63} + s_{i+3}s_{i+64} + s_{i+46}s_{i+64} + s_{i+64}b_{i+63} \\
&+s_{i+3}s_{i+25}s_{i+46} + s_{i+3}s_{i+46}s_{i+64} + s_{i+3}s_{i+46}b_{i+63} \\
&+s_{i+25}s_{i+46}b_{i+63} + s_{i+46}s_{i+64}b_{i+63} \qquad (3)
\end{aligned}
$$

$$
\begin{aligned}
z_i &= b_{i+1} + b_{i+2} + b_{i+4} + b_{i+10} + b_{31} \\
&+b_{43} + b_{56} + h \qquad (4)
\end{aligned}
$$

For Grain-128 these functions are defined as,

$$
\begin{aligned}
f &= s_i + s_{i+7} + s_{i+38} + s_{i+70} \\
&+s_{i+81} + s_{i+96} \qquad (5) \\
g &= s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} \\
&+b_{i+3}b_{i+67} + b_{i+11}b_{i+13} \\
&+b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} \\
&+b_{i+61}b_{i+65} + b_{i+68}b_{i+84} \qquad (6) \\
h &= b_{i+12}s_{i+8} + s_{i+13}s_{i+20} + b_{i+95}s_{i+42} \\
&+s_{i+60}s_{i+79} + b_{i+12}b_{i+95}s_{i+95} \qquad (7) \\
z_i &= b_{i+2} + b_{i+15} + b_{i+36} + b_{i+45} + b_{i+64} \\
&+b_{i+73} + b_{i+89} + h + s_{i+93} \qquad (8)
\end{aligned}
$$

Grain-128a was proposed to adapt Grain-128 in authentication [15]. For Grain-128a these functions are defined as,

$$
\begin{aligned}
f &= s_i + s_{i+7} + s_{i+38} + s_{i+70} \\
&+s_{i+81} + s_{i+96} \qquad (9) \\
g &= s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} \\
&+b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} \\
&+b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} \\
&+b_{i+68}b_{i+84} + b_{i+88}b_{i+92}b_{i+93}b_{i+95} \\
&+b_{i+22}b_{i+24}b_{i+25} \\
&+b_{i+70}b_{i+78}b_{i+82} \qquad (10) \\
h &= b_{i+12}s_{i+8} + s_{i+13}s_{i+20} + b_{i+95}s_{i+42} \\
&+s_{i+60}s_{i+79} + b_{i+12}b_{i+95}s_{i+94} \qquad (11) \\
y_i &= s_{i+93} + b_{i+2} + b_{i+15} + b_{i+36} + b_{i+45} \\
&+b_{i+64} + b_{i+73} + b_{i+79} + h \qquad (12) \\
z_i &= y_{64+2i} \qquad (13)
\end{aligned}
$$

It can be seen that in Grain-128a all output bits are not available for inspection, every second bit following $64^{th}$ output bit ($y_i$) are actually visible to the user. This certainly complicates the attack.

An *initialization* phase is carried out before generation of keystream bits. The $n$ bit key, $k = (k_0, k_2, \ldots, k_n)$ and the $m$ bit initialization vector $IV = (IV_0, IV_2, \ldots, IV_{m-1})$ are loaded in the NFSR and the LFSR respectively as, $b_i = k_i, 0 \leq i \leq n$ and $s_i = IV_i, 0 \leq i \leq m$, rest of the LFSR bits, $(s_m, s_{m+1}, \ldots, s_{n-1})$ are loaded with 1. $m$ has value 64 for Grain v1 and 96 for Grain-128 and Grain-128a. During initialization, the cipher is run for $2n$ rounds without producing any keystream. The output bit, $z_i$ is XOR-ed with feedback bit of both the LFSR and the NFSR.

## III. Fault Analysis Model

Our fault model creates faults in the NFSR. The following features are required for the attack.

1) The adversary is able to induce faults at $random$ positions of the NFSR of the Grain implementation (hardware or software). Hence, exact fault position is not known beforehand.
2) The fault affects *exactly one* bit of the NFSR at any cycle of operation. So, the fault amounts to flipping *exactly one bit* of the NFSR of the implementation.
3) A fault to an NFSR bit can be *reproduced* at any cycle of operation, once, it is created.
4) The attacker is able to determine and control the cycles of operation of the implementation, i.e., the *timing of the implementation is under control* of the attacker.

Flipping exactly one bit of the NFSR may seem to be a strong assumption, but can be achieved by triggering laser shots through the I/O signal for hardware implementations ([19], [20]). Also reproducing a fault at a particular location may seem difficult, but keeping parameters intact this can *possibly* be achieved.

## IV. Fault Analysis of Grain Family of Stream Ciphers

We inject faults at the NFSR of Grain ciphers and use differences in normal ($z_{normal}$)and faulty ($z_{faulty}$) keystreams to determine state of the cipher (i.e. $b_0, b_1, \ldots, b_n, s_0, s_1, \ldots, s_n$) at a target cycle $T$ (referred to as *base point*) of operation following initialization. Output difference of faulty and fault-free keystream at iteration $t$ after $T$ is denoted by $\delta^t$, i.e., $z^t_{normal} + z^t_{faulty} = \delta^t$.

We will follow the following five steps:

1) Determine fault position in the NFSR.
2) Pre-compute fault traces.
3) Determine NFSR bits, $b_0, b_1, \ldots, b_n$.
4) Determine LFSR bits, $s_0, s_1, \ldots, s_n$.
5) Invert states from cycle $T$ to 0 to obtain the *key*.

The details of each step is described next.

### A. Determine Fault Location in the NFSR

**Basic Idea**: The output $z_i$ are given by equations 4, 8 and 13. A fault induced in the NFSR will according to equation of $z_i$ produce different $\delta^t$ depending on the fault location at different $t$. For example, for Grain-128, faulting $b_{89}$ *will produce* $\delta^t = 1$ for $t = 0, 16, 25$ etc. *The observation is that a linear term of* $z_i$ *will always produce* $\delta^t = \delta(z_i) = 1$*, while nonlinear terms lead to* $\delta^t = \delta(z_i) = non-constant \neq 1$ *for some input values following standard finite difference rules, when fault is moved to corresponding bits in the term, without affecting other bits in* $z_i$ *in the process.* It turns out that this constant $\delta^t$ pattern is unique

**Algorithm 1** FormDeltaPattern(f)

$stateSize \leftarrow n$;
`Grain v1`
$outputPos[] \leftarrow \{1, 2, 4, 10, 31, 43, 56, 63\}$;
`Grain-128 and Grain-128a`
$outputPos[] \leftarrow \{2, 15, 36, 45, 64, 73, 79\}$;
**for** f=0 to $stateSize$ **do**
  $\sigma_f = \Phi$
  **for** $linearTerm \in outputPos$ **do**
    **if** $f \geq linearTerm$ **then**
      $mov \leftarrow (f - linearTerm)$;
      **if** $onlyContains(outputPos, mov, linearTerm, f)$ **then**
        $\sigma_f \leftarrow \sigma_f \cup \{mov\}$;
      **end if**
    **end if**
  **end for**
**end for**

for faults in NFSR bits. This gives us algorithm 1 to find this unique pattern versus fault locations. During simulation, we check the distance($mov$) of linear terms of $z_i$ from induced fault locations and check that no other $z_i$ input is corrupt ($onlyContains$), thus determining the pattern of $\delta^t$. Algorithm $onlyContains(array, movement, location, f)$ (algorithm 7) returns $true$ if *only location* is corrupt after $movement$ cycles of operation, among the elements of $array$ for fault at $f$ at *base point*. The $\delta^t = 1$, $(t = 0, 1, \ldots, n)$ pattern for faults at different NFSR positions is shown in tables I, III and II.

Once $\sigma_i$'s are known, during online attack phase, using algorithm 2 we determine the fault location by varying $IV$ *only* randomly and reproducing the fault. For Grain-128a, the corresponding visible cycles of constant output are also given alongside $\delta(y_i)$. Note that no fault location can be obtained up to location 67.

### B. Pre-compute Fault Traces

The purpose of this phase is to store all *possible* indices of corrupted $b$-bits at cycle $t$ following fault injection at location $f$.

**Basic Idea**: Algorithm 3 is used to store fault traces at $t^{th}$ cycle from fault induction at location $f$ ($FaultTraces(f)[t]$). The idea here is that the a corruption in $b$ bit positions of the feedback (equation of $b_{i+n-1}$) ($FeedbackPositions$) *may* corrupt $b_{i+n-1}$ next cycle. Also, a corruption at $b_i$ shifts left following cycle. Other locations remain fault-free. Note that $s$ bits remain uncorrupted throughout by design.

### C. Determine $NFSR$ Bits

We now exploit the difference of fault-free and faulty output bits to determine values of $NFSR$ bits. Both feedback and output equations are used in this phase. It is observed that output bit equation, $z$ (equations 4, 8, 15) has nonconstant monomials with $b$-bits that always contain $s$-bits. Hence, linear

**Algorithm 2** DetermineFaultlocation()

$FaultLocation \leftarrow \{\}$;
$NumIVs \leftarrow 0$;
$inc \leftarrow 100$;
**while** size($FaultLocation$) $\neq 1$ **do**
  $NumIVs \leftarrow NumIVs + inc$;
  **for** $i = 0$ **to** $NumIVs$ **do**
    initialize Grain with random IV, with that fixed key;
    fault at the same location;
    form $\delta^t$, $t = 0, 1, 2, \ldots, n$;
  **end for**
  **for** $i = 0$ **to** n **do**
    $FaultLocation \leftarrow \{\}$
    **if** all positions of $\delta^t = 1 \ \forall \ j = 0, 1, 2, \ldots, NumIVs$ is in $\sigma_i$ **then**
      $FaultLocation \leftarrow FaultLocation \cup \{i\}$
    **end if**
  **end for**
**end while**
**return** $FaultLocation$

TABLE I
FAULT LOCATION VS. $\delta^t = 1$ ($\sigma$), FAULT LOCATIONS 0 TO 79 OF GRAIN V1

| *Fault Loc.* | $t =$ | *Fault Loc.* | $t =$ |
|---|---|---|---|
| 0 | | 40 | 39, 38, 36, 30, 9 |
| 1 | 0 | 41 | 40, 39, 37, 31, 10 |
| 2 | 1, 0 | 42 | 41, 40, 38, 32, 11 |
| 3 | 2,1 | 43 | 42, 41, 39, 33, 12, 0 |
| 4 | 3, 2, 0 | 44 | 43, 42, 40, 34, 13, 1 |
| 5 | 4, 3, 1 | 45 | 44, 43, 41, 35, 14, 2 |
| 6 | 5, 4, 2 | 46 | 45, 44, 42, 36, 15, 3 |
| 7 | 6, 5, 3 | 47 | 46, 45, 43, 37, 16, 4 |
| 8 | 7, 6, 4 | 48 | 47, 46, 44, 38, 17, 5 |
| 9 | 8, 7, 5 | 49 | 48, 47, 45, 39, 18, 6 |
| 10 | 9, 8, 6, 0 | 50 | 49, 48, 46, 40, 19, 7 |
| 11 | 10, 9, 7, 1 | 51 | 50, 49, 47, 41, 20, 8 |
| 12 | 11, 10, 8, 2 | 52 | 51, 50, 48, 42, 21, 9 |
| 13 | 12, 11, 9, 3 | 53 | 52, 51, 49, 43, 22, 10 |
| 14 | 13, 12, 10, 4 | 54 | 53, 52, 50, 44, 23, 11 |
| 15 | 14, 13, 11, 5 | 55 | 54, 53, 51, 45, 24, 12 |
| 16 | 15, 14, 12, 6 | 56 | 55, 54, 52, 46, 25, 13, 0 |
| 17 | 16, 15, 13, 7 | 57 | 56, 55, 53, 47, 26, 14, 1 |
| 18 | 17, 16, 14, 8 | 58 | 57, 56, 54, 48, 27, 15, 2 |
| 19 | 18, 17, 15, 9 | 59 | 58, 57, 55, 49, 28, 16, 3 |
| 20 | 19, 18, 16, 10 | 60 | 59, 58, 56, 50, 29, 17, 4 |
| 21 | 20, 19, 17, 11 | 61 | 60, 59, 57, 51, 30, 18, 5 |
| 22 | 21, 20, 18, 12 | 62 | 61, 60, 58, 52, 31, 19, 6 |
| 23 | 22, 21, 19, 13 | 63 | 62, 61, 59, 53, 32, 20, 7, 0 |
| 24 | 23, 22, 20, 14 | 64 | 63, 62, 60, 54, 33, 21, 8, 1 |
| 25 | 24, 23, 21, 15 | 65 | 64, 63, 61, 55, 34, 22, 9, 2 |
| 26 | 25, 24, 22, 16 | 66 | 65, 64, 62, 56, 35, 23, 10, 3 |
| 27 | 26, 25, 23, 17 | 67 | 66, 65, 63, 57, 36, 24, 11, 4 |
| 28 | 27, 26, 24, 18 | 68 | 67, 66, 64, 58, 37, 25, 12, 5 |
| 29 | 28, 27, 25, 19 | 69 | 68, 67, 65, 59, 38, 26, 13, 6 |
| 30 | 29, 28, 26, 20 | 70 | 69, 68, 66, 60, 39, 27, 14, 7 |
| 31 | 30, 29, 27, 21, 0 | 71 | 70, 69, 67, 61, 40, 28, 15, 8 |
| 32 | 31, 30, 28, 22, 1 | 72 | 71, 70, 68, 62, 41, 29, 16, 9 |
| 33 | 32, 31, 29, 23, 2 | 73 | 72, 71, 69, 63, 42, 30, 17, 10 |
| 34 | 33, 32, 30, 24, 3 | 74 | 73, 72, 70, 64, 43, 31, 18, 11 |
| 35 | 34, 33, 31, 25, 4 | 75 | 74, 73, 71, 65, 44, 32, 19, 12 |
| 36 | 35, 34, 32, 26, 5 | 76 | 75, 74, 72, 66, 45, 33, 20, 13 |
| 37 | 36, 35, 33, 27, 6 | 77 | 76, 75, 73, 67, 46, 34, 21, 14 |
| 38 | 37, 36, 34, 28, 7 | 78 | 77, 76, 74, 68, 47, 35, 22, 15 |
| 39 | 38, 37, 35, 29, 8 | 79 | 78, 77, 75, 69, 48, 36, 23, 16 |

TABLE III
FAULT LOCATION VS. $\delta^t = 1$ ($\sigma$)OF GRAIN-128

| *Fault Location* | $t =$ | *Fault Location* | $t =$ |
|---|---|---|---|
| 0 | | 1 | 39,55,70,75,86,91,106,110,111,122,127 |
| 2 | 0,40,56,71,76,87,92,107,111,112,123,128 | 3 | 1,41,57,72,77,88,93,108,112,113,124 |
| 4 | 2,42,58,73,78,89,94,109 | 5 | 3,43,59,74,79,90,95,110 |
| 6 | 4,44,60,75,80,91,96,111 | 7 | 5,45,61,76,81,92,97,112 |
| 8 | 6,46,62,77,82,93,98,113 | 9 | 7,47,63,78,83,94,99,114 |
| 10 | 8,48,64,79,84,95,100,115 | 11 | 9,49,65,80,85,96,101,116 |
| 12 | 10,50,66,81,97,102 | 13 | 11,51,67,82,98,103 |
| 14 | 12,52,68,83 | 15 | 0,13,53,69,84 |
| 16 | 1,14,54,70,85 | 17 | 2,15,55,71,86 |
| 18 | 3,16,56,72,87 | 19 | 4,17,57,73,88 |
| 20 | 5,18,58,74,89 | 21 | 6,19,59,75,90 |
| 22 | 7,20,60,76,91 | 23 | 8,21,61,77,92 |
| 24 | 9,22,62,78,93 | 25 | 10,23,63,79,94 |
| 26 | 11,24,64,80,95 | 27 | 12,25,39,55,65,75,81,86,96 |
| 28 | 13,26,40,56,66,87,97 | 29 | 14,27,41,57,67,88,98 |
| 30 | 15,28,42,58,68,89,99 | 31 | 16,29,43,59,69,90,100 |
| 32 | 17,30,44,60,70,91,101 | 33 | 18,31,45,61,71,92,102 |
| 34 | 19,32,46,62,72,93,103 | 35 | 20,33,47,63,73,94,104 |
| 36 | 0,21,34,48,64,74,95,105 | 37 | 1,22,35,49,65,75,96,106 |
| 38 | 2,23,36,50,66,76,97,107 | 39 | 3,24,37,51,67,77,98,108 |
| 40 | 4,25,38,52,68,78,99,109 | 41 | 5,26,53,79 |
| 42 | 6,27,54,80 | 43 | 7,28,55,81 |
| 44 | 8,29,56,82 | 45 | 9,30,57,83 |
| 46 | 10,31,58,84 | 47 | 11,32,59,85 |
| 48 | 12,33,60,86 | 49 | 13,34,61,87 |
| 50 | 14,35,62,88 | 51 | 15,36,63,89 |
| 52 | 16,37,64,90 | 53 | 17,38,65,91 |
| 54 | 18,39,66,92 | 55 | 19,40,67,93 |
| 56 | 20,41,68,94 | 57 | 21,39,42,70,75,110 |
| 58 | 22,40,43,71,76,111 | 59 | 23,41,44,72,77,112 |
| 60 | 24,42,45,73,78 | 61 | 25,43,46,74,79 |
| 62 | 26,44,47,80 | 63 | 27,45,48,81 |
| 64 | 0,28,46,49,82 | 65 | 1,29,47,50,83 |
| 66 | 2,30,48,51,84 | 67 | 3,31,49,52,85 |
| 68 | 4,32,50,53 | 69 | 5,51,54 |
| 70 | 6,52,55 | 71 | 7,53,56 |
| 72 | 8,54,57 | 73 | 0,9,55,58 |
| 74 | 1,10,56,59 | 75 | 2,11,57,60 |
| 76 | 3,12,58,61 | 77 | 4,13,59,62 |
| 78 | 5,14,60,63 | 79 | 6,15,61,64 |
| 80 | 7,16,62,65 | 81 | 8,17,63,66 |
| 82 | 9,18,64,67 | 83 | 10,19,65,68 |
| 84 | 11,20,66,69 | 85 | 12,21,67 |
| 86 | 13,22,68 | 87 | 14,23,69 |
| 88 | 15,24,70 | 89 | 0,16,25,71 |
| 90 | 1,17,26,72 | 91 | 2,18,27,73 |
| 92 | 3,19,28,39,55,70,74,75,86,91 | 93 | 4,20,29,40,56,71,75,76,87,92 |
| 94 | 5,21,30,41,57,72,76,77,88,93 | 95 | 6,22,31,42,58,73,77,78,89,94 |
| 96 | 7,23,32,43,59,74,78,79,90,95 | 97 | 8,24,39,44,55,60,75,79,80,91,96 |
| 98 | 9,25,40,45,56,61,76,80,81,92,97 | 99 | 10,26,41,46,57,62,77,81,82,93,98 |
| 100 | 11,27,42,47,58,63,78,82,83,94,99 | 101 | 12,28,43,48,59,64,79,83,84,95,100 |
| 102 | 13,29,44,49,60,65,80,84,85,96,101 | 103 | 14,30,45,50,61,66,81,85,86,97,102 |
| 104 | 15,31,46,51,62,67,82,86,87,98,103 | 105 | 16,32,47,52,63,68,83,87,88,99,104 |
| 106 | 17,33,48,53,64,69,84,88,89,100,105 | 107 | 18,34,49,54,65,70,85,89,90,101,106 |
| 108 | 19,35,50,55,66,71,86,90,91,102,107 | 109 | 20,36,51,56,67,72,87,91,92,103,108 |
| 110 | 21,37,52,57,68,73,88,92,93,104,109 | 111 | 22,38,53,58,69,74,89,93,94,105,110 |
| 112 | 23,39,54,59,70,75,90,94,95,106,111 | 113 | 24,40,55,60,71,76,91,95,96,107,112 |
| 114 | 25,41,56,61,72,77,92,96,97,108,113 | 115 | 26,42,57,62,73,78,93,97,98,109,114 |
| 116 | 27,43,58,63,74,79,94,98,99,110,115 | 117 | 28,44,59,64,75,80,95,99,100,111,116 |
| 118 | 29,45,60,65,76,81,96,100,101,112,117 | 119 | 30,46,61,66,77,82,97,101,102,113,118 |
| 120 | 31,47,62,67,78,83,98,102,103,114,119 | 121 | 32,48,63,68,79,84,99,103,104,115,120 |
| 122 | 33,49,64,69,80,85,100,104,105,116,121 | 123 | 34,50,65,70,81,86,101,105,106,117,122 |
| 124 | 35,51,66,71,82,87,102,106,107,118,123 | 125 | 36,52,67,72,83,88,103,107,108,119,124 |
| 126 | 37,53,68,73,84,89,104,108,109,120,125 | 127 | 38,54,69,74,85,90,105,109,110,121,126 |

TABLE II
FAULT LOCATION VS. $\delta^{y_t} = 1$ ($\sigma$), FAULT LOCATIONS 68 TO 127 OF GRAIN-128A

| *Fault Loc.* | *t: Visible Cycle* | *Fault Loc.* | *t : Visible Cycle* |
|---|---|---|---|
| 68 | 66:1 | 69 | |
| 70 | 68:2 | 71 | |
| 72 | 70:3 | 73 | |
| 74 | 72:4 | 75 | |
| 76 | 74:5 | 77 | |
| 78 | 76:6, 66:1 | 79 | |
| 80 | 78:7, 68:2 | 81 | 66:1 |
| 82 | 80:8, 70:3 | 83 | 68:2 |
| 84 | 82:9, 72:4 | 85 | 70:3 |
| 86 | 84:10, 74:5 | 87 | 72:4 |
| 88 | 86:11, 76:6 | 89 | 74:5 |
| 90 | 88:12, 78:7 | 91 | 76:6 |
| 92 | 90:13, 80:8 | 93 | 78:7 |
| 94 | 92:14, 82:9 | 95 | 80:8 |
| 96 | 94:15, 84:10 | 97 | 82:9 |
| 98 | 96:16, 86:11 | 99 | 84:10 |
| 100 | 98:17, 88:12 | 101 | 86:11 |
| 102 | 100:18, 90:13, 66:1 | 103 | 88:12 |
| 104 | 102:19, 92:14, 68:2 | 105 | 90:13 |
| 106 | 104:20, 94:15, 70:3 | 107 | 92:14 |
| 108 | 106:21, 96:16, 72:4 | 109 | 94:15 |
| 110 | 108:22, 98:17, 74:5 | 111 | 96:16, 66:1 |
| 112 | 110:23, 100:18, 76:6 | 113 | 98:17, 68:2 |
| 114 | 112:24, 102:19, 78:7 | 115 | 100:18, 70:3 |
| 116 | 114:25, 104:20, 80:8 | 117 | 102:19, 72:4 |
| 118 | 116:26, 106:21, 82:9 | 119 | 104:20, 74:5 |
| 120 | 118:27, 108:22, 84:10 | 121 | 106:21, 76:6 |
| 122 | 120:28, 110:23, 86:11 | 123 | 108:22, 78:7 |
| 124 | 122:29, 112:24, 88:12 | 125 | 110:23, 80:8 |
| 126 | 124:30, 114:25, 90:13 | 127 | 112:24, 82:9 |

**Algorithm 3** FaultTrace(f)

```
Grain-128
FeedbackPositions ← {0, 3, 11, 13, 17, 18, 26, 27, 40,
48, 56, 59, 61, 65, 67, 68, 84, 91, 96};
Grain-128a
FeedbackPositions ← {
0,26,56,91,96,3,67,11,13,17,18,27,59,40,48,61,
65,68,84,88,92,93,95,22,24,25,70,78,82 };
Grain v1
feedBackPositions ←
{62, 60, 52, 45, 37, 33, 28, 21, 14, 9, 0, 63, 15};
FaultTrace[0] ← {f};
for i = 0 to 127 do
  FaultTrace[i] ← {};
  for element ∈ FaultTrace[i − 1] do
    FaultTrace[i] ← FaultTrace[i] ∪ {element - 1}
    if element ∈ FeedbackPositions then
      FaultTrace[i] ← FaultTrace[i] ∪ {n-1}
    end if
  end for
end for
```

equations in $b$-bits are not obtainable. So we utilize linear $b$ terms of $z$ and feedback equation $b_n$ (equations 2, 6 and 10) to determine values of $b$-bits.

**Basic Idea**: The idea is to move the induced fault to $b_p$ or $b_q$ if $b_p b_q$ is a term in $b_{i+n}$ by $movement$ iterations. *The observation is that if $b_n$ is not faulted through any other feedback tap (seen by consulting $FaultTraces$ table) at $T$ + movement cycle, $\delta^{movement} = b_q$ or $b_p$, respectively.* However, if fault into $b_n$ is due to multiple linear or nonlinear taps of $g$, we get, $\delta^{movement}$ = a polynomial in $b$-bits = $P$. This polynomial will be affine if at most degree 2 monomials were corrupt. Thus, essentially we can have an affine difference in feedback. The target is now to move this difference uncorrupted to $z$.

We utilize linear $b$ terms of $z$ for this purpose. That is the feedback difference is moved to a linear $b$ term of $z$, which is different for different versions of Grain. Note that here also we need to be sure that corruption of $z$ comes through this term only (using $onlyContains$).

So, we use $FaultTraces$ table first time during feedback through degree-2 monomial and the second time for output through linear $b$ terms. We can determine $P$ (Left Hand Side) during simulation, the actual output difference $\delta^t$ is obtained during online phase. $P$ will now equal the difference of output bits ($\delta^t$) after the $movement$ to corresponding linear $b$ term. These $b$ values have actually moved from another locations. So, if we need to move the fault $c$ cycles from the original location to effect $b_p$ of $b_n$, we obtained $b_{n+c}$ at *base point*. It can be executed for each degree-2 monomial in $g$ and each linear term in $z$.

This procedure is summarized and parameterized according to equations in algorithm 4. Algorithm 5 constructs $P$ throughout the process. Note that fault need to be moved from $f$, after $movement$ shifts, to $feedbacklocation\_b$. It output a single $b$ position or a linear equation in $b$ according as single or multiple taps corrupt $b_n$.

For Grain-128, the *base point* fault locations and number of bits obtained from that location in single $b$-bit is tabulated in table IV. The $NFSR$ bits obtained for Grain-128a are tabulated in table V. The equations obtained for Grain v1 are tabulated in table VI.

**Grain-128:**The number of faults which give linear equations is, 125. Single bit $P$'s obtainable are $b_3, b_4, \ldots, b_{127}$. Other 3 bits, $b_0, b_1, b_2$ need to be brute-forced. On an average of $\frac{0\times3+1\times8+2\times23+3\times31+4\times26+5\times37}{128} = 3.40$ $b$-bits can be obtained from a single fault at the NFSR and 56 faults are required to determine state bits, $b_0, b_1, \ldots, b_{127}$ of the NFSR. Number of induced faults can be reduced by injecting them at consecutive cycles. For example, from the fault at $b_{67}$ at base point, we can obtain value of bit $b_3$. A fault at the previous cycle at $b_{67}$ will give value of $b_3$ at that cycle, which is the value of $b_2$ at *base point*.

**Grain-128a:**In algorithm 4 note the check $(movement - 64)\%2 == 0$ for Grain-128a. This guarantees that corresponding output bit is *visible*. However, this means we need the value of $(movement - 64)/2$, which gives the cycle of the corresponding output bit. In this case it turns out that all the output bits are at cycle 0 for the equations obtained in table V. So, for faults at 7 locations of NFSR we obtain 7 $b$ bits

**Algorithm 4** Determine$NFSR$Bits(f)

```
Grain-128
feedBack_b[] ← {0, 26, 56, 91, 96, 3, 11, 13, 17, 18, 27, 40,
48, 59, 61, 65, 67, 68, 84}
Double_feedBack_b[] ←
{3, 11, 13, 17, 18, 27, 40, 48, 59, 61, 65, 67, 68, 84}
Double_feedBack_b_corr[] ←
{67, 13, 11, 18, 17, 59, 48, 40, 27, 65, 61, 3, 84, 68}
Single_output_b[] ← {2, 15, 36, 45, 64, 73, 89}
Grain-128a
feedBack_b[] ←
{0, 26, 56, 91, 96, 3, 67, 11, 13, 17, 18, 27, 59, 40, 48, 61, 65,
68, 84, 88, 92, 93, 95, 22, 24, 25, 70, 78, 82}
Double_feedBack_b[] ← {3, 11, 17, 27, 40, 61, 68}
Double_feedBack_b_corr[] ← {67, 13, 18, 59, 48, 65, 84}
Single_output_b[] ← {2, 15, 36, 45, 64, 73, 89}
Grain v1
feedBack_b[] ←
{62, 60, 52, 45, 37, 33, 28, 21, 14, 9, 0, 63, 15}
Double_feedBack_b[] ← {63, 37, 15}
Double_feedBack_b_corr[] ← {60, 33, 9}
Single_output_b[] ← {1, 2, 4, 10, 31, 43, 56}
movement ← 0
for i = 0 to length(Double_feedBack_b) do
  if f ≥ Double_feedBack_b[i] then
    movement ← (f − Double_feedBack_b[i])
    eqnLHS ← ConstructFeedbackDiffEqn(f,
    Double_feedBack_b[i] − f,
    Double_feedBack_b[i])
    for j = 0 to length(Single_output_b) do
      movement ← movement + 127 −
      Single_output_b[j];
      movement ← disp + 79 − Single_output_b[j];
      //Grain v1
      if condition then
        Grain-128
        condition = onlyContains(feedBack_b,
        movement, Single_output_b[j], f)
        && (Double_feedBack_b_corr[i] +
        movement ≤ 127)
        Grain-128a
        condition = (movement − 64)%2 == 0
        && OnlyCorrupt(feedBack_b, movement,
        Single_output_b[j], f)
        && (Double_feedBack_b_corr[i] +
        movement ≤ 127)
        Grain v1
        condition = OnlyCorrupt(feedBack_b,
        movement, Single_output_b[j], f)
        && (Double_feedBack_b_corr[i] +
        movement ≤ 79)
        obtained eqnLHS = δ^movement
        Grain-128a
        Cycle = (movement − 64)/2;
      end if
    end for
  end if
end for
```

**Algorithm 5** ConstructFeedbackDiffEqn($f, movement,$ $feedbacklocation$)

```
linearTerms = {};
for element ∈ FaultTrace(f)[movement] do
  for i = 0 to length(feedBack_b) do
    if element == feedBack_b[i] then
      if element == Double_feedBack_b[k] for some
      k then
        linearTerms = linearTerms ∪
        {Double_feedBack_b_corr[k]};
      else
        linearTerms = linearTerms ∪ {1};
      end if
    end if
  end for
end for
return XOR of linearTerms;
```

at base point, say, $T$. Among the fault location 68 is only identifiable, so, 1 fault at $b_{68}$ gives value of 1 $b$ bit at cycle=$T$, corresponding to faults at $T - disp$ cycle. Therefore, a fault at 68 in cycle $T - disp - 1$, gives value of $b_{84}$ at cycle $T - 1$ which equals $b_{83}$ at cycle $T$. This explains column 2 of table V with indices between 0 and 127. Note that $i$ of table VI may be both positive and negative. It is seen experimentally that $b_i$ for $0 \leq i \leq 84$ can be obtained in this way for Grain-128a. Rest 43 bits require a positive shift $T - disp + i$, the argument being similar. $b$ bits are obtained from difference of this visible output in fault-free and faulty forms. Therefore, after this step we are able to find values of all $b$ bits.

**Grain v1:** It is verified experimentally that there are fault-positions which could give linear equations involving $b$ bits only (table VI). Therefore, a fault at $b_{15}$ in cycle $T - disp - 1$, gives value of $b_9$ at cycle $T - 1$ which equals $b_8$ at cycle $T$. This explains column 2 of table VI with indices between 0 and 79 similar to Grain-128a and Grain-128. Note that $i$ of table VI may be both positive and negative like Grain-128a. It is seen experimentally that with positive $i$, $b_0$ to $b_{33}$ may be determined, while with negative $i$, up to $b_{33+36} = b_{69}$ can be obtained. Rest 10 $b$ bits are to be determined brute-force, with complexity $O(2^{10})$.

### *D. Determine $LFSR$ bits*

**Basic Idea**: Here we need to obtain equations involving $s$-bits alone or $s$ and $b$ bits. The idea is to target terms of $z$ containing both $s$ and $b$ indices. Now move fault to the one of the $b$ locations in the term and find output difference as an equation involving $s$ and $b$ bits, provided other terms are not corrupted.

We explain it using Grain-128. The induced fault is propagated to any of the locations $b_{12}$ or $b_{95}$ without corrupting other $b$-bits of $z_i$ (equation 8). It is an equation of the form,

TABLE IV
FAULT LOCATION VS. NFSR BITS OBTAINED FOR GRAIN-128

| *Fault Location* | *NFSR Bits Obtained* | *Fault Location* | *NFSR Bits Obtained* |
|---|---|---|---|
| $b_0$ | | $b_{43}$ | $b_{41}, b_{44}, b_{45}, b_{51}, b_{75}$ |
| $b_1$ | | $b_{44}$ | $b_{42}, b_{45}, b_{46}, b_{52}, b_{76}$ |
| $b_2$ | | $b_{45}$ | $b_{43}, b_{46}, b_{47}, b_{53}, b_{77}$ |
| $b_3$ | $b_{67}$ | $b_{46}$ | $b_{44}, b_{47}, b_{48}, b_{54}, b_{78}$ |
| $b_4$ | $b_{68}$ | $b_{47}$ | $b_{45}, b_{48}, b_{49}, b_{55}, b_{79}$ |
| $b_5$ | $b_{69}$ | $b_{48}$ | $b_{40}, b_{46}, b_{49}, b_{56}, b_{80}$ |
| $b_6$ | $b_{70}$ | $b_{49}$ | $b_{41}, b_{47}, b_{50}, b_{57}, b_{81}$ |
| $b_7$ | $b_{71}$ | $b_{50}$ | $b_{42}, b_{48}, b_{51}, b_{58}, b_{82}$ |
| $b_8$ | $b_{72}$ | $b_{51}$ | $b_{43}, b_{49}, b_{52}, b_{59}, b_{83}$ |
| $b_9$ | $b_{73}$ | $b_{52}$ | $b_{44}, b_{50}, b_{53}, b_{60}, b_{84}$ |
| $b_{10}$ | $b_{74}$ | $b_{53}$ | $b_{45}, b_{51}, b_{54}, b_{61}, b_{85}$ |
| $b_{11}$ | $b_{13}, b_{75}$ | $b_{54}$ | $b_{46}, b_{52}, b_{55}, b_{62}, b_{86}$ |
| $b_{12}$ | $b_{14}, b_{76}$ | $b_{55}$ | $b_{47}, b_{53}, b_{56}, b_{63}, b_{87}$ |
| $b_{13}$ | $b_{11}, b_{15}, b_{77}$ | $b_{56}$ | $b_{48}, b_{88}$ |
| $b_{14}$ | $b_{12}, b_{16}, b_{78}$ | $b_{57}$ | $b_{49}, b_{89}$ |
| $b_{15}$ | $b_{13}, b_{17}, b_{79}$ | $b_{58}$ | $b_{50}, b_{90}$ |
| $b_{16}$ | $b_{14}, b_{18}, b_{80}$ | $b_{59}$ | $b_{27}, b_{51}$ |
| $b_{17}$ | $b_{15}, b_{18}, b_{19}, b_{81}$ | $b_{60}$ | $b_{28}, b_{52}$ |
| $b_{18}$ | $b_{16}, b_{17}, b_{19}, b_{20}, b_{82}$ | $b_{61}$ | $b_{29}, b_{53}, b_{65}$ |
| $b_{19}$ | $b_{17}, b_{18}, b_{20}, b_{21}, b_{83}$ | $b_{62}$ | $b_{30}, b_{54}, b_{66}$ |
| $b_{20}$ | $b_{18}, b_{19}, b_{21}, b_{22}, b_{84}$ | $b_{63}$ | $b_{31}, b_{55}, b_{67}$ |
| $b_{21}$ | $b_{19}, b_{20}, b_{22}, b_{23}, b_{85}$ | $b_{64}$ | $b_{32}, b_{56}, b_{68}$ |
| $b_{22}$ | $b_{20}, b_{21}, b_{23}, b_{24}, b_{86}$ | $b_{65}$ | $b_{33}, b_{61}, b_{69}$ |
| $b_{23}$ | $b_{21}, b_{22}, b_{24}, b_{25}, b_{87}$ | $b_{66}$ | $b_{34}, b_{62}, b_{70}$ |
| $b_{24}$ | $b_{22}, b_{23}, b_{25}, b_{26}, b_{88}$ | $b_{67}$ | $b_3, b_{35}, b_{63}, b_{71}$ |
| $b_{25}$ | $b_{23}, b_{24}, b_{26}, b_{27}, b_{89}$ | $b_{68}$ | $b_4, b_{36}, b_{64}, b_{84}$ |
| $b_{26}$ | $b_{24}, b_{27}, b_{28}, b_{90}$ | $b_{69}$ | $b_5, b_{37}, b_{65}, b_{85}$ |
| $b_{27}$ | $b_{25}, b_{28}, b_{29}, b_{59}, b_{91}$ | $b_{70}$ | $b_6, b_{38}, b_{66}, b_{86}$ |
| $b_{28}$ | $b_{26}, b_{29}, b_{30}, b_{60}, b_{92}$ | $b_{71}$ | $b_7, b_{39}, b_{67}, b_{87}$ |
| $b_{29}$ | $b_{27}, b_{30}, b_{31}, b_{61}, b_{93}$ | $b_{72}$ | $b_8, b_{40}, b_{68}, b_{88}$ |
| $b_{30}$ | $b_{28}, b_{31}, b_{32}, b_{62}, b_{94}$ | $b_{73}$ | $b_9, b_{41}, b_{69}, b_{89}$ |
| $b_{31}$ | $b_{29}, b_{32}, b_{33}, b_{63}, b_{95}$ | $b_{74}$ | $b_{10}, b_{42}, b_{70}, b_{90}$ |
| $b_{32}$ | $b_{30}, b_{33}, b_{34}, b_{64}, b_{96}$ | $b_{75}$ | $b_{11}, b_{43}, b_{71}, b_{91}$ |
| $b_{33}$ | $b_{31}, b_{34}, b_{35}, b_{65}, b_{97}$ | $b_{76}$ | $b_{12}, b_{44}, b_{72}, b_{92}$ |
| $b_{34}$ | $b_{32}, b_{35}, b_{36}, b_{66}, b_{98}$ | $b_{77}$ | $b_{13}, b_{45}, b_{73}, b_{93}$ |
| $b_{35}$ | $b_{33}, b_{36}, b_{37}, b_{67}, b_{99}$ | $b_{78}$ | $b_{14}, b_{46}, b_{74}, b_{94}$ |
| $b_{36}$ | $b_{34}, b_{37}, b_{38}, b_{68}, b_{100}$ | $b_{79}$ | $b_{15}, b_{47}, b_{75}, b_{95}$ |
| $b_{37}$ | $b_{35}, b_{38}, b_{39}, b_{69}, b_{101}$ | $b_{80}$ | $b_{16}, b_{48}, b_{76}, b_{96}$ |
| $b_{38}$ | $b_{36}, b_{39}, b_{40}, b_{70}, b_{102}$ | $b_{81}$ | $b_{17}, b_{49}, b_{77}, b_{97}$ |
| $b_{39}$ | $b_{37}, b_{40}, b_{41}, b_{71}, b_{103}$ | $b_{82}$ | $b_{18}, b_{50}, b_{78}, b_{98}$ |
| $b_{40}$ | $b_{38}, b_{41}, b_{42}, b_{48}, b_{72}$ | $b_{83}$ | $b_{19}, b_{51}, b_{79}, b_{99}$ |
| $b_{41}$ | $b_{39}, b_{42}, b_{43}, b_{49}, b_{73}$ | $b_{84}$ | $b_{52}, b_{68}, b_{80}, b_{100}$ |
| $b_{42}$ | $b_{40}, b_{43}, b_{44}, b_{50}, b_{74}$ | $b_{85}$ | $b_{53}, b_{69}, b_{81}, b_{101}$ |
| $b_{86}$ | $b_{54}, b_{70}, b_{82}, b_{102}$ | $b_{87}$ | $b_{55}, b_{71}, b_{83}, b_{103}$ |
| $b_{88}$ | $b_{56}, b_{72}, b_{84}, b_{104}$ | $b_{89}$ | $b_{57}, b_{73}, b_{85}, b_{105}$ |
| $b_{90}$ | $b_{58}, b_{74}, b_{86}, b_{106}$ | $b_{91}$ | $b_{75}, b_{87}, b_{107}$ |
| $b_{92}$ | $b_{76}, b_{88}, b_{108}$ | $b_{93}$ | $b_{77}, b_{89}, b_{109}$ |
| $b_{94}$ | $b_{78}, b_{90}, b_{110}$ | $b_{95}$ | $b_{79}, b_{91}, b_{111}$ |
| $b_{96}$ | $b_{80}, b_{92}, b_{112}$ | $b_{97}$ | $b_{81}, b_{93}, b_{113}$ |
| $b_{98}$ | $b_{82}, b_{94}, b_{114}$ | $b_{99}$ | $b_{83}, b_{95}, b_{115}$ |
| $b_{100}$ | $b_{84}, b_{96}, b_{116}$ | $b_{101}$ | $b_{85}, b_{97}, b_{117}$ |
| $b_{102}$ | $b_{86}, b_{98}, b_{118}$ | $b_{103}$ | $b_{87}, b_{99}, b_{119}$ |
| $b_{104}$ | $b_{88}, b_{100}, b_{120}$ | $b_{105}$ | $b_{89}, b_{101}, b_{121}$ |
| $b_{106}$ | $b_{90}, b_{102}, b_{122}$ | $b_{107}$ | $b_{91}, b_{103}, b_{123}$ |
| $b_{108}$ | $b_{92}, b_{104}, b_{124}$ | $b_{109}$ | $b_{93}, b_{105}, b_{125}$ |
| $b_{110}$ | $b_{94}, b_{106}, b_{126}$ | $b_{111}$ | $b_{95}, b_{107}, b_{127}$ |
| $b_{112}$ | $b_{96}, b_{108}$ | $b_{113}$ | $b_{97}, b_{109}$ |
| $b_{114}$ | $b_{98}, b_{110}$ | $b_{115}$ | $b_{99}, b_{111}$ |
| $b_{116}$ | $b_{100}, b_{112}$ | $b_{117}$ | $b_{101}, b_{113}$ |
| $b_{118}$ | $b_{102}, b_{114}$ | $b_{119}$ | $b_{103}, b_{115}$ |
| $b_{120}$ | $b_{104}, b_{116}$ | $b_{121}$ | $b_{105}, b_{117}$ |
| $b_{122}$ | $b_{106}, b_{118}$ | $b_{123}$ | $b_{107}, b_{119}$ |
| $b_{124}$ | $b_{108}, b_{120}$ | $b_{125}$ | $b_{109}, b_{121}$ |
| $b_{126}$ | $b_{110}, b_{122}$ | $b_{127}$ | $b_{111}, b_{123}$ |

TABLE V
FAULT LOCATION VS. NFSR BITS OBTAINED FOR GRAIN-128A

| *Fault Location* | *NFSR Bit Obtained* $i^{th}$ *Shift from T* |
|---|---|
| 3 | $b_{67}$ |
| 11 | $b_{13}$ |
| 17 | $b_{18-i}$ |
| 27 | $b_{59-i}$ |
| 40 | $b_{48-i}$ |
| 61 | $b_{65-i}$ |
| 68 | $b_{84-i}$ |

TABLE VI
FAULT LOCATION VS. NFSR BITS OBTAINED FOR GRAIN V1

| *Fault Location* | *NFSR Bit Obtained* $i^{th}$ *Shift from T* |
|---|---|
| 15 | $b_{9-i}$ |
| 37 | $b_{33-i}$ |

$s_p + s_q b_r = \delta^t$. So, if this $b_r$ bit is 0, we have an equation in $s_p$, otherwise, we have a linear equation in $s$-bits.

Algorithm 6 describes and conditionally parameterizes the process.

**Grain-128:**Only 66 faults (at locations $12, 13, \ldots, 44$, $95, 96, \ldots, 127$) in the NFSR gives equations. 33 equations on an average will involve a single $s$-bit. Other equations are stored. The equations obtained from faults in the NFSR are tabulated in table VII. LFSR bits are updated according to a *linear* feedback relation (equation 1). So, all $s$-bits at any cycle $t$ after/before the *base point* can be written as linear combinations of the *base point* LFSR bits $s_0, s_1, \ldots, s_{127}$. So, essentially we need 128 *linearly independent* equations from faults at different cycles after/before the *base point*. We obtain equations involving LFSR bits following algorithm 6 after inducing faults at later/earlier cycles of operation. This process is continued till we obtain 128 linearly independent equations involving 128 $s$-bits at the base point. On an average, $(128/33) * 66 = 256$ faults need to be injected to obtain all LFSR bits. 128 linearly independent equations can be solved through Gaussian elimination in time $128^3 = O(2^{21})$.

**Grain-128a:**Here, the terms involving $b$ as well as $s$ bits in $z_i$ are exactly same as Grain-128. The equations in $LFSR$ bits obtained from faults in $NFSR$ bit will be same as that tabulated in table VIII following algorithm 6. Again, the corresponding visible output bit is tabulated alongside.

Since, all $b$ bits are known these equations are essentially linear equations at base point say, $T$. So, a fault at $T - movement - 1$ cycle at location 102 gives the value of $s_8 + b_{95}s_{94}$ at cycle $T-1$, which is equal to $s_{8-1} + b_{95-1}s_{94-1}$ at cycle $T$. This is tabulated alongside in table VIII $2^{nd}$ column with indices between 0 and 127. It is seen experimentally that this yields 128 linear equations in $s$ bits when both $+$ and $-$ shifts from $T$ are allowed with 256 faults in NFSR following same argument as Grain-128. It can be solved with about $2^{21}$ time complexity using Guassian elimination.

**Grain v1:**The terms involving $b$ as well as $s$ bits are, $s_{i+64}b_{i+63}$, $s_{i+3}s_{i+46}b_{i+63}$, $s_{i+25}s_{i+46}b_{i+63}$ and $s_{i+46}s_{i+64}b_{i+63}$. Hence, $\delta^t = z_t + z_t^f$ with a bit-flip at $b_{i+63}$ after *movement* gives the nonlinear equation for some

**Algorithm 6** Determine$LFSR$Bits(f)

```
Grain-128, Grain-128a
Output_b_double_s[] = {12, 95};
Output_s_double_b[] = {8, 42};
outputPos[] = {12, 95, 2, 15, 36, 45, 64, 73, 89};
Grain v1
Output_b_double_s[] = {63};
Output_s_double_b[] = {64};
outputPos[] = {1, 2, 4, 10, 31, 43, 56, 63};
movement ← 0;
for i = 0 to 1 do
  movement ← f − Output_b_double_s[i];
  if condition then
    Grain-128
    condition = OnlyContains(outputPos, movement,
    Output_b_double_s[i], f) &&
    (Output_s_double_b[i] + movement <= 127) &&
    (Output_b_double_s[1−i]+movement) <= 127 &&
    (95 + movement) <= 127
    Grain-128a
    condition = OnlyContains(outputPos, movement,
    Output_b_double_s[i], f) &&
    (Output_s_double_b[i] + movement <= 127) &&
    (Output_b_double_s[1−i]+movement) <= 127 &&
    (95 + movement) <= 127 && (movement-64)%2
    ==0
    && (movement-64)/2 ¿ 0
    Grain v1
    condition = OnlyContains(outputPos, movement,
    Output_b_double_s[i], f) &&
    (Output_s_double_b[i] + movement <= 79) &&
    (Output_b_double_s[1 − i] + movement) <= 79
    Grain v1
    obtained equation,
    1 + s64 + s46(s25 + s64)
    = δ^movement
    Grain-128, Grain-128a
    obtained equation,
    s_Output_s_double_b[i]+movement +
    b_Output_b_double_s[1−i]+movement·s95+movement
    = δ^movement
  end if
end for
```

**Algorithm 7** onlyContains(array, movement, location, f)

```
for element ∈ FaultTrace(f)[movement] do
  for i = 0 to length(array) do
    if element != location && element == array[i]
    then
      return false;
    end if
  end for
end for
return true;
```

TABLE VII
FAULT LOCATION VS. LFSR, NFSR EQUATIONS OBTAINED FOR GRAIN-128

| *Fault Location* | *Equations* | *Fault Location* | *Equations* |
|---|---|---|---|
| $b_{12}$ | $s_8 + b_{95}.s_{95}$ | $b_{95}$ | $s_{42} + b_{12}.s_{95}$ |
| $b_{13}$ | $s_9 + b_{96}.s_{96}$ | $b_{96}$ | $s_{43} + b_{13}.s_{96}$ |
| $b_{14}$ | $s_{10} + b_{97}.s_{97}$ | $b_{97}$ | $s_{44} + b_{14}.s_{97}$ |
| $b_{15}$ | $s_{11} + b_{98}.s_{98}$ | $b_{98}$ | $s_{45} + b_{15}.s_{98}$ |
| $b_{16}$ | $s_{12} + b_{99}.s_{99}$ | $b_{99}$ | $s_{46} + b_{16}.s_{99}$ |
| $b_{17}$ | $s_{13} + b_{100}.s_{100}$ | $b_{100}$ | $s_{47} + b_{17}.s_{100}$ |
| $b_{18}$ | $s_{14} + b_{101}.s_{101}$ | $b_{101}$ | $s_{48} + b_{18}.s_{101}$ |
| $b_{19}$ | $s_{15} + b_{102}.s_{102}$ | $b_{102}$ | $s_{49} + b_{19}.s_{102}$ |
| $b_{20}$ | $s_{16} + b_{103}.s_{103}$ | $b_{103}$ | $s_{50} + b_{20}.s_{103}$ |
| $b_{21}$ | $s_{17} + b_{104}.s_{104}$ | $b_{104}$ | $s_{51} + b_{21}.s_{104}$ |
| $b_{22}$ | $s_{18} + b_{105}.s_{105}$ | $b_{105}$ | $s_{52} + b_{22}.s_{105}$ |
| $b_{23}$ | $s_{19} + b_{106}.s_{106}$ | $b_{106}$ | $s_{53} + b_{23}.s_{106}$ |
| $b_{24}$ | $s_{20} + b_{107}.s_{107}$ | $b_{107}$ | $s_{54} + b_{24}.s_{107}$ |
| $b_{25}$ | $s_{21} + b_{108}.s_{108}$ | $b_{108}$ | $s_{55} + b_{25}.s_{108}$ |
| $b_{26}$ | $s_{22} + b_{109}.s_{109}$ | $b_{109}$ | $s_{56} + b_{26}.s_{109}$ |
| $b_{27}$ | $s_{23} + b_{110}.s_{110}$ | $b_{110}$ | $s_{57} + b_{27}.s_{110}$ |
| $b_{28}$ | $s_{24} + b_{111}.s_{111}$ | $b_{111}$ | $s_{58} + b_{28}.s_{111}$ |
| $b_{29}$ | $s_{25} + b_{112}.s_{112}$ | $b_{112}$ | $s_{59} + b_{29}.s_{112}$ |
| $b_{30}$ | $s_{26} + b_{113}.s_{113}$ | $b_{113}$ | $s_{60} + b_{30}.s_{113}$ |
| $b_{31}$ | $s_{27} + b_{114}.s_{114}$ | $b_{114}$ | $s_{61} + b_{31}.s_{114}$ |
| $b_{32}$ | $s_{28} + b_{115}.s_{115}$ | $b_{115}$ | $s_{62} + b_{32}.s_{115}$ |
| $b_{33}$ | $s_{29} + b_{116}.s_{116}$ | $b_{116}$ | $s_{63} + b_{33}.s_{116}$ |
| $b_{34}$ | $s_{30} + b_{117}.s_{117}$ | $b_{117}$ | $s_{64} + b_{34}.s_{117}$ |
| $b_{35}$ | $s_{31} + b_{118}.s_{118}$ | $b_{118}$ | $s_{65} + b_{35}.s_{118}$ |
| $b_{36}$ | $s_{32} + b_{119}.s_{119}$ | $b_{119}$ | $s_{66} + b_{36}.s_{119}$ |
| $b_{37}$ | $s_{33} + b_{120}.s_{120}$ | $b_{120}$ | $s_{67} + b_{37}.s_{120}$ |
| $b_{38}$ | $s_{34} + b_{121}.s_{121}$ | $b_{121}$ | $s_{68} + b_{38}.s_{121}$ |
| $b_{39}$ | $s_{35} + b_{122}.s_{122}$ | $b_{122}$ | $s_{69} + b_{39}.s_{122}$ |
| $b_{40}$ | $s_{36} + b_{123}.s_{123}$ | $b_{123}$ | $s_{70} + b_{40}.s_{123}$ |
| $b_{41}$ | $s_{37} + b_{124}.s_{124}$ | $b_{124}$ | $s_{71} + b_{41}.s_{124}$ |
| $b_{42}$ | $s_{38} + b_{125}.s_{125}$ | $b_{125}$ | $s_{72} + b_{42}.s_{125}$ |
| $b_{43}$ | $s_{39} + b_{126}.s_{126}$ | $b_{126}$ | $s_{73} + b_{43}.s_{126}$ |
| $b_{44}$ | $s_{40} + b_{127}.s_{127}$ | $b_{127}$ | $s_{74} + b_{44}.s_{127}$ |

determinable $t$,

$$\begin{aligned} s_{i+64} + s_{i+3}s_{i+46} + s_{i+25}s_{i+46} + s_{i+46}s_{i+64} &= \delta^t \\ s_{i+64} + s_{i+46}(s_{i+3} + s_{i+25} + s_{i+64}) &= \delta^t \end{aligned}$$

This equation becomes linear if $s_{i+46}$ is known, otherwise, we have few small nonlinear equations in $s$ variables. The equations in $LFSR$ bits obtained from faults in $NFSR$ bit is tabulated in table IX. As in the case of $NFSR$ changing cycles of fault injection by $i$, gives a displacement of indices of bit-positions of equations by $i$, with indices between 0 and 79, this explains column 2 of table IX. Total number of faults required on average is, $80/5 * 5 = 80$, as 80 equations are needed to solve 80 variables. The obtained small nonlinear equations can be solved using Grobner basis mathod, XL etc. with complexity at most $O(80^3) < O(2^{21})$, with about 160 faults required. Thus the complexity of the attack remains same as Grain-128 and Grain-128a.

### *E. Inverting Internal States*

Till this phase we have successfully obtained full internal state of Grain ciphers at *base point*i.e.,

TABLE VIII
FAULT LOCATION VS. LFSR,NFSR BITS OBTAINED FOR GRAIN-128A

| *Fault Location* | *NFSR Bit Obtained /$i^{th}$ Shift from T* | *Output Cycle* |
|---|---|---|
| 78 | $s_{8-i} + b_{95-i}s_{94-i}$ | 1 |
| 80 | $s_{8-i} + b_{95-i}s_{94-i}$ | 2 |
| 82 | $s_{8-i} + b_{95-i}s_{94-i}$ | 3 |
| 84 | $s_{8-i} + b_{95-i}s_{94-i}$ | 4 |
| 86 | $s_{8-i} + b_{95-i}s_{94-i}$ | 5 |
| 88 | $s_{8-i} + b_{95-i}s_{94-i}$ | 6 |
| 90 | $s_{8-i} + b_{95-i}s_{94-i}$ | 7 |
| 92 | $s_{8-i} + b_{95-i}s_{94-i}$ | 8 |
| 94 | $s_{8-i} + b_{95-i}s_{94-i}$ | 9 |
| 96 | $s_{8-i} + b_{95-i}s_{94-i}$ | 10 |
| 98 | $s_{8-i} + b_{95-i}s_{94-i}$ | 11 |
| 100 | $s_{8-i} + b_{95-i}s_{94-i}$ | 12 |
| 102 | $s_{8-i} + b_{95-i}s_{94-i}$ | 13 |
| 104 | $s_{8-i} + b_{95-i}s_{94-i}$ | 14 |
| 106 | $s_{8-i} + b_{95-i}s_{94-i}$ | 15 |
| 108 | $s_{8-i} + b_{95-i}s_{94-i}$ | 16 |
| 110 | $s_{8-i} + b_{95-i}s_{94-i}$ | 17 |
| 112 | $s_{8-i} + b_{95-i}s_{94-i}$ | 18 |
| 114 | $s_{8-i} + b_{95-i}s_{94-i}$ | 19 |
| 116 | $s_{8-i} + b_{95-i}s_{94-i}$ | 20 |
| 118 | $s_{8-i} + b_{95-i}s_{94-i}$ | 21 |
| 120 | $s_{8-i} + b_{95-i}s_{94-i}$ | 22 |
| 122 | $s_{8-i} + b_{95-i}s_{94-i}$ | 23 |
| 124 | $s_{8-i} + b_{95-i}s_{94-i}$ | 24 |
| 126 | $s_{8-i} + b_{95-i}s_{94-i}$ | 25 |

TABLE IX
FAULT LOCATION VS. LFSR,NFSR BITS OBTAINED FOR GRAIN V1

| *Fault Location* | *NFSR Bit Obtained /$i^{th}$ Shift from T* |
|---|---|
| 63 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 64 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 65 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 66 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 67 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 68 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 69 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 70 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 71 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 72 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 73 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 74 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 75 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 76 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 77 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |
| 78 | $1 + s_{64-i} + s_{46-i}(s_{25-i} + s_{64-i})$ |

$(s_0^T, s_1^T, \ldots, s_n^T, b_0^T, b_1^T, \ldots, b_n^T)$ are known. Now, we describe the procedure of obtaining $key$ by inverting this state. This method is similar to that of [4]. We denote by $f^t$ and $g^t$ the feedback values at $t^{th}$ cycle.

**Step 1:** When, $t > 2n$, let, $(s_0^{t+1}, s_1^{t+1}, \ldots, s_n^{t+1}, b_0^{t+1}, b_1^{t+1}, \ldots, b_n^{t+1})$ be the full internal state of Grain after $(t+1)$ cycles of operation. Then, according to Grain specification (refer to section 2),

$$s_i^{t+1} = s_{i+1}^t; i = 0, 1, \ldots, n-2; \quad (14)$$
$$s_{n-1}^{t+1} = f^t; \quad (15)$$
$$b_i^{t+1} = b_{i+1}^t; i = 0, 1, \ldots, n-2; \quad (16)$$
$$b_{n-1}^{t+1} = g^t; \quad (17)$$

TABLE X
FAULTS REQUIRED AND COMPUTING COST

| Ciphers | *Grain v1* | *Grain-128* | *Grain-128a* |
|---|---|---|---|
| Online Cost | *(#Faults,Cost)* | *(#Faults,Cost)* | *(#Faults,Cost)* |
| Location | (0,11) | (0,14) | (0,14) |
| Pre-computation | (0, $2^{12}$) | (0, $2^{15}$) | (0, $2^{15}$) |
| NFSR | (70, $2^{10}$) | (56, 128) | (128, 128) |
| LFSR | (80, $< 2^{21}$) | (256, $2^{21}$) | (256, $2^{21}$) |
| Inversion | (0, $r(T)$) | (0, $r(T)$) | (0, $r(T)$) |
| Total | (150, $< 2^{21}$) | (312, $2^{21}$) | (384, $2^{21}$) |

The above equations can be rewritten as follows.

$$s_{i+1}^t = s_i^{t+1}; i = 0, 1, \ldots, n-2; \quad (18)$$
$$s_0^t = s_n^{t+1} + f^t + s_0^t; \quad (19)$$
$$b_{i+1}^t = b_i^{t+1}; i = 0, 1, \ldots, n-2; \quad (20)$$
$$b_0^t = b_{n-1}^{t+1} + g^t + b_0^t; \quad (21)$$

These equations hold as both $f$ and $g$ respectively contain $s_0^t$ and $b_0^t$ in all the three versions. Thus we are able to invert the full state up to initialization.

**Step 2:** When, $t <= 2n$, the output bits, $z^t$ are fed back at $s_{n-1}^{t+1}$ and $b_{n-1}^{t+1}$. Now, can in all three cases be written in terms of variables,
$s_i^{t+1}, i = 0, 1, \ldots, n-1$ and
$b_i^{t+1}, i = 0, 1, \ldots, n-1$. For example, $s_{20}^t = s_{21}^{t+1}$ and $b_{26}^t = b_{27}^{t+1}$ etc.. Hence, once $z^t$ is computed,

$$s_0^t = s_{n-1}^{t+1} + f^t + s_0^t + z^t; \quad (22)$$
$$b_0^t = b_{n-1}^{t+1} + g^t + b_0^t + z^t; \quad (23)$$

Thus, again being able to invert during initialization also.

This way inverting from iteration $2n$ to 0, we get back secret $key$ in $b_0, b_1, \ldots, b_{n-1}$.

## V. COMPLEXITY, LIMITATIONS AND ACHIEVEMENTS

In this section, we measure the cost of our proposed attack against three Grain ciphers. This complexity is approximated by the average computing expenses of the process. We also estimate the average number of fault inductions required to completely break the system. Table X summarizes the cost. In this table $T$ refers to the base point of the attack, $r(T)$ represents the cost of reversing Grain for $T$ cycles.

1) *Fault Location Determination*: It is divided into formation of $\sigma$ table during simulation (algorithm 1) and determination of fault locations in online mode (algorithm 2). The off-line mode requires a space overhead of $128 * 128 = 2^{14}$ for Grain-128 and Grain-128a, and $80 * 80 = 1600$ for Grain v1. Following $\sigma$ formation, location determination is of logarithm complexity giving cost of about 14 units of computations at most. No online faults are required in this phase.
2) *Pre-computation of Fault Traces*: This phase stores precomputed traces for all $n$ fault-locations for the following $2n$ cycles. Hence, space required in $n*2n*n = 2n^3$. Time complexity of this phase is $n * 2n = 2n^2$ Grain

TABLE XI
COMPARISON OF COST OF FAULT ATTACK AGAINST GRAIN

| Paper | Cipher | Fault Model | Faults |
|---|---|---|---|
| [4] | Grain-128 | Similar | 1587 |
| [8] | Grain-128a | Similar | 1831 |
| Current | Grain v1 | - | 150 |
| Current | Grain-128 | - | 312 |
| Current | Grain-128a | - | 384 |

round operations. This is an off-line operation. Online fault induction is not required in this phase also.

3) *Determining NFSR Bits*: We have faulted a particular location in online mode and determined linear equations by executing algorithm 4 in off-line mode. About $56$ faults were required for Grain-128 and $128$ units of computation. For Grain-128a each fault yields one equation. Allowing shifts, total number of online faults required is $128$ and total computing cost is $128$ units. Grain v1 follows same pattern as Grain-128a, requiring $70$ online faults and $2^{10}$ units of computation for brute-force.
4) *Determining LFSR Bits*: We obtained linear or small nonlinear equations in this phase. Storing the equations require linear space. Grain-128 and Grain-128a due to same set of utilized equations, requires equal complexity, which is $256$ average online faults and computing of about $2^{21}$ units. Grain v1 requires $80$ faults and less than that amount of computing.
5) *Inverting States*: Inverting the state is constant computation per round and storage of $2n$ bits. Hence, $r(T)$ units of computing is required and constant space is required for this phase irrespective of the version.

It follows from table X, that Grain-128a is the strongest of the three ciphers.

Clearly, the attack can be adapted to Grain-like ciphers. Higher minimum degree $g$, $z$ will increase attack complexity. Multiple consecutive faults are a more realistic scenario, exploiting this situation may lead to better attacks. Allowing small higher degree equations may also decrease the complexity.

Table XI compares costs of fault attack against Grain presented in literatature with that presented here. It can be noted that in comparison requirements of number of faults is comparatively less in this paper.

In [5], the authors have proposed another attack on Grain family with minimal assumptions. They use SAT solver and non-reproduction of faults. The attack targets both LFSR and NFSR of the cipher. It is claimed that the attack takes 10 or less faults to break the system. However, one strong assumption in the paper is rekeying of the cipher. Ideally, an attacker does not have access to the pins for inputting key into the system. Also the overhead of using a SAT solver is not negligible.

## VI. CONCLUSION

In this paper, we have described a fault analysis on the eStream cipher Grain family of stream ciphers. The earlier fault attacks on Grain target LFSR of the cipher. We have presented a fault attack that faults NFSR. The attack is applied successfully against three Grain ciphers, Grain v1, Grain-128a and Grain-128. It has been seen that a few number of faults (at most 125% of state size) are required which is much lesser than earlier works. Space and time requirements are also minimal. We comment that this attack can be easily adapted on Grain-like ciphers.

## REFERENCES


[1] The eStream project. *"http://www.ecrypt.eu.org/stream/"*.
[2] Mukesh Agrawal, Sandip Karmakar, Dhiman Saha, and Debdeep Mukhopadhayay. Scan Based Side Channel Attacks on Stream Ciphers and their Counter-measures. *Progress in Cryptology - INDOCRYPT 2008*, 5365/2008:226–238, 2008.
[3] Steve Babbage, Christophe De Canniere, Anne Canteaut, Carlos Cid, Henri Gilbert, Thomas Johansson, Matthew Parker, Bart Preneel, Vincent Rijmen, and Matthew Robshaw. The eStream Portfolio. *"http://www.ecrypt.eu.org/stream/portfolio.pdf"*.
[4] Alexandre Berzati, Cecile Canovas, Guilhem Castagnos, Blandine Debraize, Louis Goubin, Aline Gouget, Pascal Paillier, and Stephanie Salgado. Fault Analysis of Grain-128. *Hardware-Oriented Security and Trust, IEEE International Workshop on*, 0:7–14, 2009.
[5] Santanu Sarkar and Subhadeep Banik and Subhamoy Maitra Differential Fault Attack against Grain family with very few faults and minimal assumptions *Cryptology ePrint Archive, Report 2013/494* 2013 http://eprint.iacr.org/.
[6] Karmakar, Sandip and Roy Chowdhury, Dipanwita. Fault Analysis of Grain-128 by Targeting NFSR. *Progress in Cryptology AFRICACRYPT 2011*, 298-315, 2011.
[7] Banik, Subhadeep and Maitra, Subhamoy and Sarkar, Santanu. A Differential Fault Attack on the Grain Family of Stream Ciphers. *Cryptographic Hardware and Embedded Systems CHES 2012*, 122–139, 2012.
[8] Banik, Subhadeep and Maitra, Subhamoy and Sarkar, Santanu. A Differential Fault Attack on Grain-128a using MACs. *SPACE 2012*, 2012.
[9] Banik, Subhadeep and Maitra, Subhamoy and Sarkar, Santanu. A Differential Fault Attack on the Grain Family under Reasonable Assumptions. *Indocrypt 2012*, 191–208, 2012.
[10] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. *Crypto 97*, 1294 of LNCS:513–525, 1997.
[11] Johannes Blomer and Jean-Pierre Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard. *Financial Cryptography*, 2742:162–181, 2003.
[12] Itai Dinur and Adi Shamir. Breaking Grain-128 with Dynamic Cube Attacks. *Cryptology ePrint Archive: Report 2010/570*.
[13] Martin Hell, Thomas Johansson, and Willi Meier. A Stream Cipher Proposal: Grain-128. *eSTREAM, ECRYPT Stream Cipher Project*, 2006.
[14] Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. *eSTREAM, ECRYPT Stream Cipher Project*, 2006.
[15] Martin Agren, Martin Hell, Thomas Johansson, and Willi Meier. A New Version of Grain-128 with Authentication. *SKEW 2011*.
[16] Jonathan J. Hoch and Adi Shamir. Fault Analysis of Stream Ciphers. *CHES 2004*, 3156/2004 of LNCS:1–20, 2004.
[17] Michal Hojsk and Bohuslav Rudolf. Differential Fault Analysis of Trivium. *FAST SOFTWARE ENCRYPTION*, 5086/2008 of LNCS:158–172, 2008.
[18] Aleksandar Kircanski and Amr M. Youssef. Differential Fault Analysis of Rabbit. *SELECTED AREAS IN CRYPTOGRAPHY*, 5867/2009:197–214, 2009.
[19] Sergei P. Skorobogatov. Optically Enhanced Positionlocked Power Analysis. *CHES 2006*, 4249 of LNCS:61–75, 2006.
[20] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. *CHES 2002*, 2523 of LNCS:2–12, 2002.