FResCA: A Fault-Resistant Cellular Automata Based Stream Cipher

Jimmy Jose[1,2(✉)] and Dipanwita Roy Chowdhury[1]

[1] Crypto Research Laboratory, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India
{jimmy,drc}@cse.iitkgp.ernet.in
[2] Department of Computer Science and Engineering, National Institute of Technology Calicut, Calicut, India
jimmy@nitc.ac.in

Abstract. Grain is a stream cipher suitable for restricted hardware environments. The cipher is particularly vulnerable to fault attacks. It has been shown that fault injection in either the linear or nonlinear block of Grain can break the cipher. Fault attacks generally exploit the linear behaviour and the reversibility of the cipher states. Using Cellular Automata (CA), we propose a Grain-like cipher which is shown to be strong particularly against fault attacks.

Keywords: Fault analysis · Grain cipher · Stream cipher · Cellular automata

1 Introduction

Encryption techniques in general can be broadly classified into two, namely symmetric-key encryption (secret-key encryption) and asymmetric-key encryption (public-key encryption). Stream ciphers fall under symmetric-key encryption where the sender and the receiver share the same secret key. Stream ciphers may be classified into synchronous where the keystream depends only on the key and asynchronous where the keystream depends on both the key and the ciphertext.

The importance of stream ciphers stems from the fact that they are suitable for resource-constrained environments where computing power, memory, etc. are at a premium. Stream ciphers may be efficient in software meaning they need fewer instructions to execute or may be hardware efficient meaning they need less hardware circuitry.

eSTREAM [1], the ECRYPT Stream Cipher Project, was conceptualised to promote the design of efficient stream ciphers. The shortlisted ciphers under the the project falls into two categories. One set of ciphers are more suitable for software applications with high throughput requirements and the other set of ciphers are suitable for restricted hardware environments. Grain cipher falls under the second category and our interest is on 128-bit version of Grain known as Grain-128 which is described in detail in Subsect. 2.1. In this paper, reference to Grain implies Grain-128 version of the cipher unless otherwise stated.

S. El Yacoubi et al. (Eds.): ACRI 2016, LNCS 9863, pp. 24–33, 2016.
DOI: 10.1007/978-3-319-44365-2_3

Side channel attacks (SCA) are attacks that target the limitations in the physical implementation of the cryptosystem. They can be either active or passive. Fault attacks are active side channel attacks and in particular, they find relevance against stream ciphers [9]. Faults are injected into unknown bit positions in the cipher state and by tracking these faults, the state of the cipher is found. These attacks are suitable when the cryptosystem is not vulnerable to direct attacks. Some of the fault attacks that were proposed against Grain are fault injection in LFSR [4], fault injection in NFSR [12], and the attack which is applicable to Grain family of ciphers [3].

The immunity of a CA based Trivium-like stream cipher against fault attacks was shown [10] in ACRI 2014. In [7], a scalable stream cipher based on CA was proposed. In this paper, we propose a CA based Grain-like stream cipher FResCA which is resistant to fault attack. Analysis of its cryptographic strength is provided with a special emphasis on fault attacks.

The paper is organised as follows. In Sect. 2, we give a brief description of Grain and suitability of CA as better cryptographic primitive against fault attacks.The proposed cipher FResCA is described in Sect. 3. Section 4 discusses the security of the proposed cipher and Sect. 5 describes the cipher's strength against fault attacks. We conclude with Sect. 6.

2 Preliminaries

Boolean functions should have certain desirable cryptoproperties so that they can be employed in practical cryptosystems. A detailed discussion on Boolean functions and their cryptoproperties can be found in [14]. In this section, a brief description of Grain is provided followed by a discussion on CA's suitability as cryptographic primitive against fault attacks.

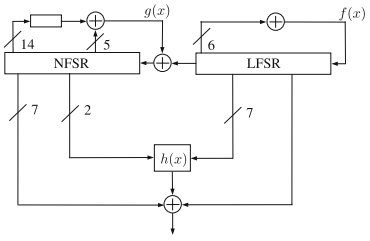

Fig. 1. Grain block diagram

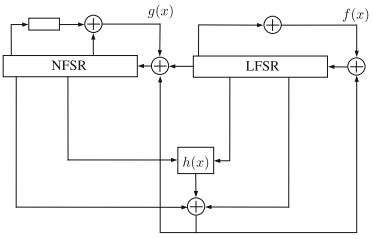

Fig. 2. Grain Initialisation

2.1 Grain-128 Description

Grain [8] has three blocks namely an LFSR, an NFSR, and an output function as shown in Fig. 1. The contents of the LFSR and NFSR are denoted by $(s_i, s_{i+1}, \cdots, s_{i+127})$ and $(b_i, b_{i+1}, \cdots, b_{i+127})$ respectively at clock i and

together determine the 256-bit state of the cipher. The LFSR feedback polynomial $f(x)$ updates the LFSR as

$s_{i+128} = s_i \oplus s_{i+7} \oplus s_{i+38} \oplus s_{i+70} \oplus s_{i+81} \oplus s_{i+96}$.

The NFSR feedback polynomial $g(x)$ together with s_i from LFSR updates the NFSR as

$$b_{i+128} = s_i \oplus b_i \oplus b_{i+26} \oplus b_{i+56} \oplus b_{i+91} \oplus b_{i+96} \oplus b_{i+3}b_{i+67} \oplus b_{i+11}b_{i+13} \oplus b_{i+17}b_{i+18} \oplus b_{i+27}b_{i+59} \oplus b_{i+40}b_{i+48} \oplus b_{i+61}b_{i+65} \oplus b_{i+68}b_{i+84}.$$

The nonlinear function h is defined as

$h = b_{i+12}s_{i+8} \oplus s_{i+13}s_{i+20} \oplus b_{i+95}s_{i+42} \oplus s_{i+60}s_{i+79} \oplus b_{i+12}b_{i+95}s_{i+95}$.

The output bit z_i is given as

$z_i = b_{i+2} \oplus b_{i+15} \oplus b_{i+36} \oplus b_{i+45} \oplus b_{i+64} \oplus b_{i+73} \oplus b_{i+89} \oplus h \oplus s_{i+93}$.

In the initialisation phase, the key $(k_0, \cdots, k_{127})$ and IV $(v_0, \cdots, v_{95})$ are loaded into the NFSR and LFSR as

$(b_0, \cdots, b_{127}) \leftarrow (k_0, \cdots, k_{127})$

$(s_0, \cdots, s_{127}) \leftarrow (v_0, \cdots, v_{95}, 1, \cdots, 1)$.

Then the cipher is iterated 256 times without producing the keystream in the initialisation phase as shown in Fig. 2. Instead, the keystream bits are fed back and XORed with the input of both the LFSR and NFSR. After the initialisation phase, these feedback paths are removed and keystream bits are available through the output line.

2.2 CA as Better Cryptographic Primitive Against Fault Attacks

CA can provide fast evolution and high nonlinearity if appropriate CA rules are employed. CA diffuse the state bits very fast and in a single cycle, every bit undergoes transformation. This parallel transformation forces the introduced fault to spread quickly and dissipate. So fault tracking becomes very difficult.

Most of the stream ciphers that are vulnerable to fault attacks use a reversible algorithm. That is, if we know the state of the cipher at any instant, the cipher can be run backwards until it reaches the initial state revealing the key which was used for the cipher initialisation. CA can be effectively employed in such a way that it is practically infeasible to reverse the cipher.

Since nonlinearity of CA based stream ciphers is quite high if appropriate CA rules are employed, it is very difficult to generate linear equations which can be solved to extract state bits of the cipher as done in a general fault attack. Thus, a CA based stream cipher can be designed such that it prevents fault attack.

Other than these, CA prevent correlation attacks too. CA based stream ciphers can provide fast initialisation as they achieve desirable values of cryptographic properties in less rounds. They can be designed in such a way that they are suitable in hardware as well as software.

3 FResCA Description

Our cipher model is Grain-like and has three blocks, namely nonlinear, linear, and a mixing function as shown in Fig. 3. Nonlinear block uses highly nonlinear

4-neighbourhood CA rule whereas linear block uses 3-neighbourhood maximum length CA and both are of 128-bit length. Third block performs nonlinear mixing and produces the output stream.

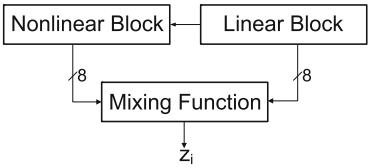

Fig. 3. FResCA block diagram

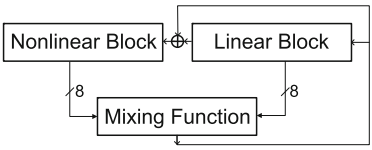

Fig. 4. FResCA Initialisation

3.1 Nonlinear Block

A study was conducted in [11] to explore the cryptographic properties of four-neighbourhood CA. In the paper, five good candidate rules were selected based on their cryptographic properties. The rules 43350 and 51510 are cryptographically stronger among the five rules. In the NIST test suite [2], rule 43350 performed better. So this rule is used in the nonlinear block which computes the state of the CA cell q_i at time $t+1$ as

Rule 43350: $q_i(t+1) = q_{i-2}(t) \oplus q_{i+1}(t) \oplus (q_{i-1}(t) + q_i(t))$,

where $q_{i-2}(t)$, $q_{i-1}(t)$, $q_i(t)$, and $q_{i+1}(t)$ represent the state of the two left neighbours, self, and right-neighbour respectively at time t of the left-skewed four-neighbourhood CA, $\oplus$ and $+$ represent XOR and OR operations respectively. Rule 43350 provides high nonlinearity to the nonlinear block and the nonlinearity increases rapidly with each iteration. The rule is balanced and exhibits good correlation immunity also.

3.2 Linear Block

Linear block uses a three-neighbourhood linear maximum-length CA. Two linear rules, rule 90 and 150 are used to realise the CA. These rules are defined as
Rule 90: $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$
Rule 150: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$

The bit positions 1 and 29 uses rule 150 and all other 126 positions use rule 90 to realise the maximum length CA.

3.3 Nonlinear Mixing

The nonlinear mixing is achieved by using NMIX function [5] which is defined for two n-bit inputs X, Y, and output Z as follows:
$z_i \leftarrow x_i \oplus y_i \oplus c_{i-1}$
$c_i \leftarrow x_0 y_0 \oplus \cdots \oplus x_i y_i \oplus x_{i-1} x_i \oplus y_{i-1} y_i$
and $x_{-1} = y_{-1} = c_{-1} = 0, 0 \leq i \leq n-1$.

We take 8-bits each from the nonlinear and linear block as input to NMIX and the most significant bit (MSB) is taken as the output from NMIX. We can see that all input bits are present in the computation of MSB. The output function is a 16-variable bent function of degree two and this mixing provides high nonlinearity.

3.4 Working of FResCA Cipher

FResCA is **F**ault-**Res**istant **C**ellular **A**utomata based Grain-like cipher. The cipher has an initialisation phase and a keystream generation phase. The output is suppressed in the initialisation phase which consists of 32 iterations. The number of iterations in initialisation phase is very less in comparison to 256 iterations in Grain. The 128-bit key and 128-bit IV are loaded to the nonlinear and linear blocks respectively. During this phase, the output is fed back to both the nonlinear and linear blocks as shown in Fig. 4. The output bit acts as the right neighbour for the rightmost linear cell and the XOR of the output bit and the leftmost bit of linear block acts as the right neighbour to the rightmost nonlinear cell in the initialisation phase. In each iteration, each bit in the nonlinear block changes its state according to the 4-neighbourhood CA rule 43350. In the linear block, the bits change according to 3-neighbourhood linear rule 90 except for bit locations 1 and 29 where rule 150 is used. Eight taps are taken from each of the nonlinear and linear blocks so that the number of input lines to the nonlinear mixing block is 16. The eight taps corresponds to the bit positions 1, 22, 43, 64, 65, 86, 107, and 128 in both the blocks. The tap positions are spaced equally other than the two central taps so that the output is influenced by all the bits in less iterations.

Thirty-two cycles are more than sufficient for all the 256 state bits to influence the input to the NMIX function and thereby influencing the output of NMIX, i.e., the keystream bit. So each keystream bit is influenced by all the 256 state bits. The first keystream bit z_1 (first 32 keystream bits are suppressed and are not available as output) involves 16 variables, 8 from the nonlinear block and 8 from the linear block.

If nonlinear bits are represented as $(b_1, \cdots, b_{128})$ and linear bits are represented as $(s_1, \cdots, s_{128})$, then z_1 is represented as

$z_1 = b_{128} \oplus s_{128} \oplus b_1 s_1 \oplus b_{22} s_{22} \oplus b_{43} s_{43} \oplus b_{64} s_{64} \oplus b_{65} s_{65} \oplus b_{86} s_{86} \oplus b_{107} s_{107} \oplus b_{86} b_{107} \oplus s_{86} s_{107}$.

The cryptographic properties of z_1 are measured. Nonlinearity, correlation-immunity, resiliency, and algebraic degree are 32256, 1, 1, and 2 respectively. Since we have resiliency value as 1, the Boolean function is balanced also.

The second keystream bit z_2, which is also suppressed, involves 41 variables (26 from the nonlinear block and 15 from linear block). The Boolean function corresponding to z_2 is

$$z_2 = b_{126} \oplus b_{127} \oplus b_{128} \oplus s_{127} \oplus (b_1 s_1) \oplus (b_1 s_2) \oplus (b_{105} b_{84}) \oplus (b_{105} b_{85}) \oplus (b_{105} b_{86}) \oplus (b_{105} b_{87}) \oplus (b_{105} s_{106}) \oplus (b_{105} s_{108}) \oplus (b_{106} b_{84}) \oplus (b_{106} b_{85}) \oplus (b_{106} b_{86}) \oplus (b_{106} b_{87}) \oplus (b_{106} s_{106}) \oplus (b_{106} s_{108}) \oplus (b_{107} b_{84}) \oplus (b_{107} b_{85}) \oplus (b_{107} b_{86}) \oplus (b_{107} b_{87}) \oplus (b_{107} s_{106}) \oplus (b_{107} s_{108}) \oplus (b_{108} b_{84}) \oplus (b_{108} b_{85}) \oplus (b_{108} b_{86}) \oplus (b_{108} b_{87}) \oplus (b_{108} s_{106}) \oplus (b_{108} s_{108}) \oplus (b_{127} b_{128}) \oplus (b_2 s_1) \oplus (b_2 s_2) \oplus (b_{20} s_{21}) \oplus (b_{20} s_{23}) \oplus (b_{21} s_{21}) \oplus (b_{21} s_{23}) \oplus (b_{22} s_{21}) \oplus (b_{22} s_{23}) \oplus (b_{23} s_{21}) \oplus (b_{23} s_{23}) \oplus (b_{41} s_{42}) \oplus (b_{41} s_{44}) \oplus (b_{42} s_{42}) \oplus (b_{42} s_{44}) \oplus (b_{43} s_{42}) \oplus (b_{43} s_{44}) \oplus (b_{44} s_{42}) \oplus (b_{44} s_{44}) \oplus (b_{62} s_{63}) \oplus (b_{62} s_{65}) \oplus (b_{63} s_{63}) \oplus (b_{63} s_{64}) \oplus (b_{63} s_{65}) \oplus (b_{63} s_{66}) \oplus (b_{64} s_{63}) \oplus (b_{64} s_{64}) \oplus (b_{64} s_{65}) \oplus (b_{64} s_{66}) \oplus (b_{65} s_{63}) \oplus (b_{65} s_{64}) \oplus (b_{65} s_{65}) \oplus (b_{65} s_{66}) \oplus (b_{66} s_{64}) \oplus (b_{66} s_{66}) \oplus (b_{84} s_{85}) \oplus (b_{84} s_{87}) \oplus (b_{85} s_{85}) \oplus (b_{85} s_{87}) \oplus (b_{86} s_{85}) \oplus (b_{86} s_{87}) \oplus (b_{87} s_{85}) \oplus (b_{87} s_{87}) \oplus (s_{106} s_{85}) \oplus (s_{106} s_{87}) \oplus (s_{108} s_{85}) \oplus (s_{108} s_{87}) \oplus (b_{105} b_{85} b_{86}) \oplus (b_{106} b_{107} b_{84}) \oplus (b_{106} b_{107} b_{85}) \oplus (b_{106} b_{107} b_{86}) \oplus (b_{106} b_{107} b_{87}) \oplus (b_{106} b_{107}\ s_{106}) \oplus (b_{106} b_{107} s_{108}) \oplus (b_{106} b_{85} b_{86}) \oplus (b_{107} b_{85} b_{86}) \oplus (b_{108} b_{85} b_{86}) \oplus (b_{21} b_{22} s_{21}) \oplus (b_{21} b_{22} s_{23}) \oplus (b_{42} b_{43} s_{42}) \oplus (b_{42} b_{43} s_{44}) \oplus (b_{63} b_{64} s_{63}) \oplus (b_{63} b_{64} s_{65}) \oplus (b_{64} b_{65} s_{64}) \oplus (b_{64} b_{65} s_{66}) \oplus (b_{85} b_{86} s_{85}) \oplus (b_{85} b_{86} s_{87}) \oplus (b_{106} b_{107} b_{85} b_{86}).$$

Algebraic degree increases from 2 to 4 in the second iteration itself and increases with each iteration. A Boolean function should have high algebraic degree for cryptographic security [6]. Presence of forty-one variables makes it impossible to compute the other crypto properties as the truth-table has 2^{41} entries.

4 Security of FResCA

We analyse the security of the cipher with respect to different attacks.

Meier-Staffelbach Attack Our cipher uses 4-neighbourhood CA in the nonlinear block. It has been shown in [11] that a certain class of 4-neighbourhood CA resist Meier-Staffelbach attack. The nonlinear rule in FResCA is from that class. So the cipher is strong against the attack.

Linear Attacks First keystream bit of our cipher has a nonlinearity of 32256. The nonlinearity increases with each iteration. Keystream bits are available only from 33rd iteration onwards and the nonlinearity will be much higher at that stage.

Correlation Attacks The nonlinear CA rule 43350 exhibits good correlation property [11]. Output from this block is combined with the output from the maximum length CA block using NMIX function [5] which guarantees correlation immunity and balancedness in the output. Thus correlation attacks can be ruled out.

Algebraic Attacks Ciphers having high algebraic degree in their Boolean function are difficult to attack algebraically. The rate of increase in algebraic degree is high with each iteration in our cipher. This prevents algebraic attacks.

Scan-Based Side Channel Attacks This attack will succeed if the cipher is reversible. The combination of nonlinear and linear CA rules makes our cipher robust against this attack.

Experimental Results Our cipher was run on NIST test suite [2]. Input to the NIST test suite was a file containing 0.1 billion keystream bits. NIST test suite was allowed to partition the input to 100 keystreams where each keystream contains 1 million bits.

Our cipher with rule 43350 as the 4-neighbourhood rule passed all the tests for different key-IV pairs. Rule 51510, which is also thought to be promising [11], failed in some tests (failures relatively low in number). The same key-IV pairs were used to generate keystream bits for NOCAS cipher [13] which needs 64 cycles in the initialisation phase whereas FResCA needs only 32. NOCAS failed in non-overlapping tests but passed all other tests in the test suite.

Strength of the cipher against fault attack is discussed separately in the next section.

5 Strength of FResCA Against Fault Attacks

Grain is shown to be vulnerable against fault attacks. It is shown in [4] that the cipher can be broken by inserting fault into the LFSR state. Later, fault injection in NFSR state [12] is also shown to be successful.

5.1 Injecting Fault into Linear Block of Grain

Initially, the attack [4] tries to find out the fault location by analysing the keystream difference bits. If d_i, z_i, and z'_i are respectively the i^{th} keystream difference bit, keystream bit, and keystream bit after fault injection, then $d_i = z_i \oplus z'_i$. Corresponding to each possible fault location i in the LFSR, a unique pattern in $\{d_i\}$ is found out.

The Boolean representation of Grain-128 output z_i contains $s_{i+13}s_{i+20}$ and $s_{i+60}s_{i+79}$ as terms. If any one of the four bits s_{13}, s_{20}, s_{60}, or s_{79} is faulted, then the output difference represents the value of an LFSR bit. As an example, if fault is injected at position 60, output difference is the value of s_{79}. In this way, each LFSR bit is revealed.

If we know the LFSR bits, we can generate linear equations involving NFSR bits from the regular keystream. The only exception is the involvement of the term $b_{i+12}b_{i+95}s_{i+95}$ and if $s_{i+95} = 0$, we get linear equation involving several bits of the current NFSR state. The keystream difference equations that are used to recover LFSR states can be reused here also.

Since Grain-128 algorithm is reversible, the cipher can be run backwards from the known current state to reach the initial state to reveal the key.

Prevention of the Attack in FResCA In our cipher, fault position cannot be found out as described in the attack. ΔGrain algorithm tries to find a unique pattern P_i corresponding to each each fault position i, $0 \leq i \leq 127$. The algorithm relies on the fact that the fault injected position will be represented by 1 while other bits are 0 and the availability of this 1 in the output through different taps at different instances of time provides the value of i. In our CA based linear

block, a single 1 in the register generates more 1's in different cell positions with each iteration (unlike in Grain where the single 1 shifts its position with each iteration) and we cannot generate unique patterns corresponding to each fault position.

In phase 3 of the attack, the algorithm (algorithm 3) which finds the number of known LFSR bits also will fail as it uses ΔLFSR as used in ΔGrain algorithm because there will be more 1's in our cipher as opposed to a single 1 in ΔLFSR. In the case of Grain, the number of LFSR bits that are recovered depends on the fault location and the number of times the cipher is clocked after the fault is injected. In our case, during subsequent clocking after fault injection, the fault will not be preserved in a single cell and gets mixed with more and more neighbours with each iteration. We can find out exactly one linear bit if the fault location is 86 or 107 as $s_{86}s_{107}$ is the only term involving linear bits in the keystream function $z = b_{128} \oplus s_{128} \oplus b_1 s_1 \oplus b_{22}s_{22} \oplus b_{43}s_{43} \oplus b_{64}s_{64} \oplus b_{65}s_{65} \oplus b_{86}s_{86} \oplus b_{107}s_{107} \oplus b_{86}b_{107} \oplus s_{86}s_{107}$.

If we know all the linear bits (s_i's), we can try to find nonlinear bits (b_i's). Then the only nonlinear term in z_i (keystream bit) is $b_{86}b_{107}$. We cannot make it linear as was done in Grain where the only nonlinear term was $b_{12}b_{95}s_{95}$ and if s_{95} was zero, the whole equation becomes linear. In Grain, more iterations will produce more linear equations (shown in Fig. 4 - Algorithm "CountEquations" [4]) whereas in our cipher, more iterations will not be fruitful as the fault start mixing with more and more neighbours in each iteration.

We need to compute the initial state to find the key thereby breaking the cipher. Use of CA and combination of nonlinear and linear CA rules to produce keystream bits make the computation of initial state from the known present state of the cipher difficult. In our case, finding present state itself is not possible.

5.2 Injecting Fault into Nonlinear Block of Grain

After initialisation phase in Grain, if fault is injected into NFSR, it cannot propagate to LFSR. This attack [12], just like the previous attack, also starts by finding out the fault injection location. Over a large number of key-IV combinations, the nonlinear b-bits will provide unique keystream difference sequence for fault injection at a particular location thereby revealing the fault location. To enhance the attack, a table named Fault Traces Table which contains the list of corrupted bit locations after t iterations on fault injection at location f is also created.

To find out nonlinear bits, the feedback equation for b_{128} is used which contains seven degree-2 terms containing only nonlinear b-bits of the form $b_m b_n$. Moving the fault to either b_m or b_n will provide the value of the other as the difference. To get more linear equations involving b-bits, we use linear b-terms in z_i. Feedback fault is moved to one of the single b-bit output taps to get either the value of b-bit or linear equation involving several b-bits.

The three terms $b_{i+12}s_{i+8}$, $b_{i+95}s_{i+42}$, $b_{i+12}b_{i+95}s_{i+95}$ in z_i are exploited to determine the LFSR bits. They provide either the s-bit value or provide linear equation involving s-bits. These equations are solved to get the LFSR state.

Like in the previous attack, after NFSR and LFSR state is obtained, Grain is run backwards to get the key.

Prevention of the Attack in FResCA When fault is injected in the nonlinear block of our cipher, the fault propagation largely depends on the nonlinear CA rule used and the effect of the fault is propagated to the neighbouring cells on both sides of the cell where fault was injected. Because of the fast diffusion of the introduced fault, we will not be able to find unique pattern σ_f as in algorithm 1 of the attack so that fault location can be found out. Moreover, the computed fault traces in our cipher looks random as the nonlinear CA rule determines how the fault traces are spread.

It is not possible to determine the nonlinear bits in our cipher like the NFSR bits in Grain. The attack against Grain uses equations corresponding to z_i and b_{128}. In our cipher, we have only output z_i as there is no feedback path (which corresponds to b_{128} in Grain) for the nonlinear block. In the output equation for Grain, there are 7 single b-bit output taps, namely $b_2, b_{15}, b_{36}, b_{45}, b_{64}, b_{73}$, and b_{89} that are exploited in this phase whereas our cipher has only one, i.e., b_{128}. We cannot move a fault into single bit output tap as done in Grain as it is a simple left shift in Grain with each iteration. In our cipher, the number of 1's will be more and controlling them to occupy specific cell locations is practically impossible. This phase of the attack consults Fault Traces Table twice, but in our case, fault traces cannot be constructed in a similar manner.

We try to find out linear block bits just like how LFSR bits are found out in Grain assuming that we have already found out the nonlinear block bits. This phase uses the terms in z_i representation which has both b and s, like $b_m s_n$. In Grain, the induced fault propagates to some specified locations in NFSR without corrupting other b-bits of z_i. In our cipher, it is very difficult to guarantee this as more and more bits get corrupted because of the higher diffusion of the fault which is the inherent nature of the CA.

Computation of initial state from the known present state of the cipher thereby breaking the cipher is very difficult because of the use of CA and combination of nonlinear and linear CA rules to produce keystream bits as described in the previous attack.

6 Conclusion

We have proposed a 4-neighbourhood CA based Grain-like stream cipher. Its initialisation is 8 times faster than Grain. We have shown that it is strong against different attacks, in particular, fault attacks. Experimental results confirm our claim for its robustness.

References

1. The Estream Project. http://www.ecrypt.eu.org/stream/. Accessed 31 Mar 2016
2. The NIST Statistical Test Suite. http://csrc.nist.gov/groups/ST/toolkit/rng/. Accessed 31 Mar 2016
3. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the grain family of stream ciphers. In: Proceedings of the CHES 2012-14th International Workshop, Leuven, Belgium, 9–12 September 2012
4. Berzati, A., Canovas, C., Castagnos, G., Debraize, B., Goubin, L., Gouget, A., Paillier, P., Salgado, S.: Fault analysis of grain-128. In: 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, pp. 7–14 (2009)
5. Bhaumik, J., RoyChowdhury, D.: Nmix: An ideal candidate for key mixing. In: SECRYPT 2009, Proceedings of the International Conference on Security and Cryptography, Milan, Italy, 7–10 July 2009, pp. 285–288 (2009)
6. Ding, C., Xiao, G., Shan, W.: The Stability Theory of Stream Ciphers. LNCS, 1st edn. Springer, Heidelberg (1991)
7. Ghosh, S., RoyChowdhury, D.: CASca: A CA based scalable stream cipher. In: Mohapatra, R.N., Chowdhury, D.R., Giri, D. (eds.) Mathematics and Computing. Springer Proceedings in Mathematics & Statistics, pp. 95–105. Springer, India (2015)
8. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: 2006 IEEE International Symposium on Information Theory, pp. 1614–1618 (2006)
9. 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)
10. Jose, J., Das, S., RoyChowdhury, D.: Inapplicability of fault attacks against triviumon a cellular automata based stream cipher. In: Proceedings of the Cellular Automata-ACRI 2014, Krakow, Poland, 22–25 September 2014, pp. 427–436 (2014)
11. Jose, J., RoyChowdhury, D.: Investigating four neighbourhood cellular automata as better cryptographic primitives. J. Discrete Math. Sci. Crypt. (to be published in 2016)
12. 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)
13. Karmakar, S., RoyChowdhury, D.: NOCAS: A nonlinear cellular automata based stream cipher. In: Automata 2011, Center for Mathematical Modeling, 21–23 November 2011, pp. 135–146. University of Chile, Santiago, Chile (2011)
14. Wu, C.K., Feng, D.: Boolean Functions and Their Applications in Cryptography. Advances in Computer Science and Technology, 1st edn. Springer, Heidelberg (2016)