Countermeasures of Side Channel Attacks on Symmetric Key Ciphers Using Cellular Automata

Sandip Karmakar and Dipanwita Roy Chowdhury

Indian Institute of Technology, Kharagpur

Abstract. Side Channel Attacks (SCA) are one of the most effective means in breaking symmetric key ciphers. Generally, SCA exploits the side-channel leakages output by the implementations of ciphers or introduces defects in the system to analyze them. A number of countermeasures have been proposed to strengthen/remedy implementations of ciphers against SCA. However, none of the countermeasures, to our knowledge, are good enough towards its goal ([16], [19], [3]). In this paper, we emphasis on the necessity of randomness in designing countermeasures against SCA and propose Cellular Automata (CA) based system to thwart SCA. Our countermeasure is also analyzed against popular SCA, such as, differential power attack (DPA), scan-chain based attacks (SC-SCA) and fault attacks (FA).

1 Introduction

Side Channel Attacks (SCA) are cryptographic attacks on implementations of ciphers. This kind of attacks analyze the side channel leakages of the implementations of the ciphers, such as, power consumption, electromagnetic radiation, scan-chains etc. to deduce information about the key of the cipher. Fault attacks are a kind of SCA which inject faults into the implementation of a cipher in order to obtain information about the secret key exploiting the correct and faulty ciphertexts. It is shown in literature ([4], [10], [9], [13], [22], [1], [5]) that a wide variety of symmetric ciphers are vulnerable against SCA. The most important aspect of SCA is that it is completely practical i.e. it can be implemented easily in real-life applications and completes within few hours. Most block and stream ciphers have been shown to be broken by SCA. Hence, the necessity to resist SCA is obvious. A large number of countermeasures are introduced in literature aiming to thwart SCA. Most of the countermeasures are however costly and shown to be analyzable. Countermeasures using random masks are particularly effective ([16], [19], [3]).

Cellular Automata are a self-evolving system of cells which updates itself per cycle following a rule embedded into it. Linear Cellular Automaton (CA) is known for its ability to generate pseudorandom sequences needed for various applications like VLSI testing and coding theory [21]. CA shows very good pseudo-randomness ([21], [14]). Several researchers have exploited this pseudorandomness of CA to

G.C. Sirakoulis and S. Bandini (Eds.): ACRI 2012, LNCS 7495, pp. 623–632, 2012.

cryptography. The cryptanalysis of linear CA-based cryptographic techniques [18] show that nonlinearity is needed for cryptographic applications. However, nonlinear CA shows high correlation. The 3-neighbourhood nonlinear rule 30 CA has long been considered a good pseudo-random generator and studied for cryptography [20]. It passed various statistical tests for pseudorandomness with good results, until Willi Meier and Othmar Staffelbach proposed an attack, exploiting its high correlation, on pseudorandom sequences generated by rule 30 CA [17], which would break any such system of 300 cells with a complexity of about 2^{19}. Another attack on rule 30 CA is also reported in [15]. These findings show that for cryptography, the data stream generated by CA needs to satisfy additional properties for pseudorandom sequence generation [14].

In this paper, the non-linear hybrid CA introduced in [14] is used as the random sequence generator. We test different cryptographic properties of the CA along with d-monomial to conclude that it is cryptographically robust. These cryptographically robust sequences are used as a random mask for resisting the side channel attacks. We then present the masking scheme. Finally, we analyze the masking scheme against different SCA.

This paper is organized as follows. Following the introduction, Section 2 presents basic definitions and notations regarding Cellular Automata and the related cryptographic terms. It also discusses the presently known countermeasures against SCA. In Section 3, we introduce the random masking scheme using CA and discuss its performance. Finally, the paper is concluded in Section 4.

2 Preliminaries

In this section, we present the basic definitions of CA and also of the cryptographic properties. The existing countermeasures against SCA are also briefly outlined in this section.

2.1 Cellular Automata Related Definitions

Definition 1. *Cellular Automata: A cellular automaton is a finite array of cells. Each cell is a finite state machine $C = (Q, f)$ where Q is a finite set of states and f is a mapping $f : Q^n \to Q$. The mapping f, is called local transition function. n is the number of cells the local transition function depends on. On each iteration of the CA each cell updates itself with respective f.*

Adjacent cells of a cell are called the neighbourhood of CA. A 1-dimensional CA, whose rule depends on left and right neighbour and the cell itself is called a 3-*neighbourhood CA*. Similarly, if each cell depends on 2 left and 2 right neighbours and itself only, it is called 5-*neighbourhood CA*. A CA whose cells depend on 1 left and 2 right neighbouring cells is called a 4-*neighbourhood right skew CA*. A *left skewed* 4-*neighbourhood CA* can be defined similarly.

Definition 2. *Rule: The local transition function for a* 3*-neighbourhood CA cell can be expressed as follows:*

$$q_i(t+1) = f[q_i(t), q_{i+1}(t), q_{i-1}(t)]$$

where, f denotes the local transition function realized with a combinational logic, and is known as a rule of CA [7]. Here, $q_i(t)$ represents the value of the i^{th} cell after t iterations. The decimal value of the truth table of the local transition function is defined as the rule number *of the cellular automaton.*

For one dimensional 3-neighbourhood CA the definitions of some rules are given below:

Rule 30: $f = q_{i-1}(t) \oplus (q_{i+1}(t) + q_i(t))$, where $+$ is the Boolean 'OR' operator and $\oplus$ is the Boolean 'XOR' operator.
Rule 60: $f = q_{i-1}(t) \oplus q_i(t)$.
Rule 90: $f = q_{i-1}(t) \oplus q_{i+1}(t)$.

A CA whose local transition function is same accross the cells is called *uniform CA*. A CA whose local transition function is *not* same for all the cells is a *hybrid CA*. Hybrid CA may be constructed by choosing different linear rules or by choosing different linear and nonlinear rules over the automaton. A CA whose first and last cells are connected to 0 is called *null-boundary* CA.

A CA whose local transition function consists of only 'XOR' operator is called a *linear CA*. A CA whose at least one local transition function consists of 'AND'/'OR' in addition to 'XOR' is *nonlinear CA*. For example, rule, $f = q_{i-1}(t) \oplus q_{i+1}(t)$ employed in each cell is a linear CA and $f = q_{i-1}(t).q_{i+1}(t)$ employed in each cell is a nonlinear CA, where, $q_{i-1}(t)$ and $q_{i+1}(t)$ denotes left and right neighbours of the i^{th} cell at t^{th} instance of time. A uniform CA each of whose transition function is, $f = q_{i-1}(t) \oplus (q_{i+1}(t) + q_i(t))$ is a *rule* 30 *uniform CA*.

2.2 Cryptographic Terms and Primitives

We now present definitions of related cryptographic terms and properties used in this paper.

Definition 3. *Pseudorandom Sequence: An algorithmic sequence is pseudorandom if it cannot be distinguished from a truly random sequence by any efficient (polynomial time) probabilistic procedure or circuit.*

A variable or its negation (x or $\bar{x}$) is called a literal. Any number of 'AND'-ed literals is called a *conjunction*. For example, $x.y.\bar{z}$ is a *conjunction*.

Definition 4. *Algebraic Normal Form: Any* Boolean function *can be expressed as* XOR *of* conjunctions *and a* Boolean constant, True *or* False. *This form of the* Boolean function *is called its* Algebraic Normal Form (ANF).

Every *Boolean function* can be expressed in *ANF*. As an example, $f(x_1, x_2, x_3) = (x_1 \oplus x_2).(x_2 \oplus x_3)$ is not in *ANF*. Its *ANF* representation is, $f(x_1, x_2, x_3) = x_1.x_2 \oplus x_1.x_3 \oplus x_2 \oplus x_2.x_3$.

Definition 5. *Algebraic Degree: The maximum number of literals in any* conjunction *of* ANF *of a* Boolean function *is called its* degree. *Ciphers expressible or conceivable as a* Boolean function *have* algebraic degree *which is the same as the* degree *of the* ANF *of the* Boolean function.

Thus, $f(x_1, x_2) = x_1 \oplus x_2 \oplus x_1.x_2$ has algebraic degree 2.

2.3 Existing Countermeasures against SCA

In this section, we briefly discuss existing proposed countermeasures against SCA.

The countermeasures presented in literature can be broadly divided in two categories:

1. Full/Partial Duplication of the Cipher Implementation.
2. Random Masking of the Cipher Output.

Full/Partial Duplication of the Cipher Implementation. The most popular and effective countermeasure proposed against SCA is the full or partial duplication of the cipher in its implementations ([2], [6], [3], [16], [19]). In this category, duplicate copies of the cipher is kept in the implemented system. Therefore, within the cipher there are always two ciphertexts (full/partial). A correct output is generated only when both the implementations match in output. This kind of countermeasure is particularly effective against fault attacks. The three major countermeasures proposed are discussed below:

- **Full Duplication:** The cipher implementation duplicates the cipher in two. Both the implementations are run simultaneously. A correct ciphertext is generated only if both the outputs of the implementations match. Clearly, a fault attack against such countermeasure is highly unlikely to succeed as in case of defective computations both the implementations are unlikely to give same result. DPA may however be more effective against such countermeasure. SC-SCA is unaffected by this algorithm. However, this method is costly.
- **Partial/Full Encrypt/Decrypt:** Here, the generated ciphertext from the encryption system is fully/partially decrypted through the decryption system to see for a match at an intermediate level or the plaintext. A correct output is generated only after a match. FA and DPA can be resisted using this technique, but, SC-SCA is not affected.
- **Diffusion of Defect:** Fault attacks are effective against symmetric key ciphers mainly due the localization of defect in the computation of the ciphertext. Hence, a technique to increase the complexity of FA is to diffuse the defect in computation of the cipher. In this scenario, the the output of both the implementations are XOR-ed and run through a diffusion layer before actually generating the ciphertext. A 0 difference only generates the correct ciphertext. This clearly makes FA difficult. It also increases the complexity of DPA. SC-SCA remains unaffected through this countermeasure.

Random Masking. Random masking is a strong form of countermeasure ([11], [8]) against SCA targeting mainly against DPA and SC-SCA. There are many variants of masking in literature. Generally, random masking also uses some form of duplication of cipher computation. However, SC-SCA will be protected by a simple masking of the scan_out line with no . FA or DPA however, requires full duplication in general. A generic view is to mask the two computations of ciphertexts using some randomly generated masks and output correct ciphertext when at some equality. An SC-SCA may be protected using masking by giving the adversary the masked output. FA may be resisted by testing the equality of the two masked outputs. Clearly, DPA can also be thwart by an appropriate random mask. We provide an example of the random masking scheme next.

Example: Consider cryptosystem S_c. A mask generator M_g. A duplicate of the cryptosystem will be present in the system, S_c'. At any instant consider, the output from S_c and S_c' be the ciphertexts, C and C'. The mask generator M_g generates two masks per correct ciphertext, M and M' (Fig. 1). The countermeasure logic then does the following:

$a = (C + M) + M'$
$b = (C' + M') + M$
if $a == b$ **return** C
else return 0
where, + is the bitwise-XOR operator.

Note that, this masking scheme prevents the exact computation of C and C' from being leaked thus resisting DPA, also, the computation of a and b is symmetric which also complicates DPA. A fault attack is also difficult due to this masking scheme as, considering S_c or S_c' or M_g being faulted at random storage places or operations used in these systems, it is highly unlikely that the equality $a == b$ will hold. It can be mentioned that a lot of security of the system depends on M_g being a good pseudorandom generator. Otherwise, FA may easily find a bypass by faulting S_c or S_c' only or both. Pseudorandomness of M and M' also secures the system against DPA as the added random noise into the computation prevents predicting exact leakage of the system. Non-random masks are therefore, weakness of the system.

Although effective, most of the countermeasures have been shown to be vulnerable. The masking based countermeasure is particularly effective as it is simple to implement and much more effective against any SCA compared to the duplication based countermeasures which are mainly targeted to FA. In the following subsection we will emphasis on the necessity of randomness of the masking function and present a masking scheme using CA which we analyze against FA, DPA and SC-SCA.

3 Random Masking Using Cellular Automata

In this section, we introduce a countermeasure against SCA on symmetric key ciphers. As may be seen from the earlier description of existing countermeasures

against SCA, the random masking scheme is an elegant and secure methodology in resisting SCA. However, most of the security of the scheme depends on the random sequence generator used in masks. The requirements of the generation of mask-bits are, security, low-cost hardware and high throughput. CA is a good choice for low-cost compact implementations and security. We discuss the CA based pseudo-random generator for masking next.

Our countermeasure assumes that there is a duplication of the original cipher structure. The masking bits are generated using the pseudo-random sequence generator M_g. Two sets of bits, M and M' are generated per output bit sequence. The design and security of M_g is described in the following two subsections. The masking scheme then works as depicted next.

$a = M' \star (C \star M)$;
$b = M \star (C' \star M')$;
if $(a == b)$ *return C;*
else return 0;

where, C and C' are the two ciphertexts generated by the two cipher instances. $\star$ is a commutative operator generally taken as $\oplus$. Note that, this scheme eliminates the possibility of FA as, C and C' are not the same in that case. The complexity of DPA attacks increase as the masking introduces randomness into the power traces. SC-SCA also gets complicated as the random masks are also part of scan-chain. The operator $\star$ may be operators other than $\oplus$, such as, . or a combination of other operators.

Table 1. ANF of CA Rules used in Ruleset 5

Rule #	ANF	Linear?
30	$(x_2.x_3) \oplus x_1 \oplus x_2 \oplus x_3$	No
60	$x_1 \oplus x_2$	Yes
90	$x_1 \oplus x_3$	Yes
120	$x_1 \oplus (x_2.x_3)$	No
150	$x_1 \oplus x_2 \oplus x_3$	Yes
180	$x_1 \oplus x_2 \oplus (x_2.x_3)$	No
210	$x_1 \oplus x_3 \oplus (x_2.x_3)$	No
240	x_1	Yes

3.1 Design of M_g

M_g is a random sequence generator the output of which masks the ciphertexts. The random sequence generator M_g is constructed using a hybrid nonlinear cellular automata (CA) [21]. It is an CA consisting of cells with rules $30, 60, 90, 120, 150, 180, 210, 240$ spaced alternatively. The actual functions these rules operate on is given in table 1. We use a null-boundary CA of length 128 for this design. The input to the CA is a 128 bit key specified by the developer.

At each iteration the CA is run for two cycles to generate two masks M and M'. The masks are then operated with the two generated ciphertexts, C and C' as discussed earlier (Fig. 1).

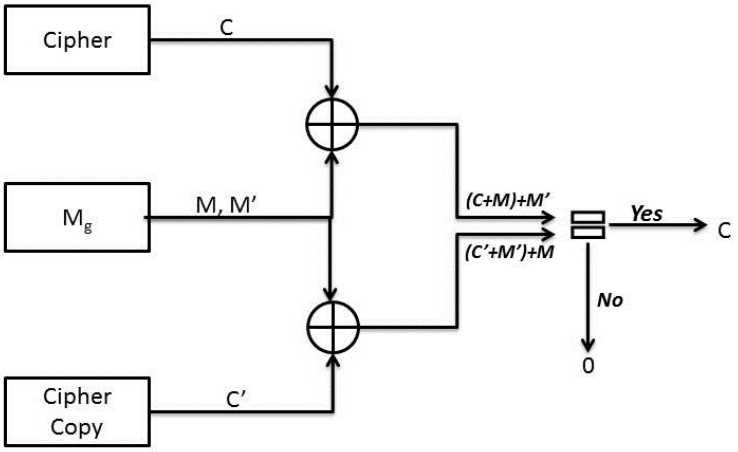

Fig. 1. The Pseudo-random Generator M_g

We present the cryptographic properties of the hybrid CA in the following subsection.

Cryptographic Properties of M_g. The hybrid CA used in construction of M_g, is studied in [14], it is called ruleset 5 by the authors. We briefly discuss its characteristics here.

Ruleset 5 is tested over three iterations for cryptographic properties like, balancedness, nonlinearity, resiliency and algebraic degree. It is also tested against d-monomial test [12]. The results are tabulated in tables 2 and 3.

Table 2. Cryptographic Properties of Ruleset 5

Iteration	Balancedness	Nonlinearity	Resiliency	Degree
1	Balanced	2	2	2
2	Balanced	8	2	3
3	Balanced	32	2	4

It can be seen that over all the iterations, the CA generates balanced output, has a fast nonlinearity growth. Resiliency of the CA is constant, it has good algebraic degree growth rate. Also, results of d-monomial test is satisfactory. Hence, it may be claimed that it will work as a good pseudo-random sequence generator.

Table 3. *d*-Monomial Characteristics of Hybrid Ruleset 5 CA [14]

	Number of n^{th} degree terms			
Rules	*1*	*2*	*3*	*4*
Ideal	*1,2,3*	*1,5,10*	*0,5,52*	*0,2,52*
Ruleset 5	3,2,4	1,3,5	0,2,6	0,0,3

3.2 Performance

M_g is a simple 128-bit masking generator. The generation of masks requires only 2 cycles of operation of the CA. Hence, it is fast in operation. The hardware of M_g requires 128 bit registers, 192 XOR and 64 AND gates.Obviously, the design requires competitive hardware. Therefore, it can be said that the performance of M_g is fast with competitive hardware overhead.

3.3 Security

We briefly discuss the security achieved by M_g in this subsection.

Scan Based Side Channel Attacks. Scan-chains allow the adversary to observe states of the chip-registers. Due to the presence of 128-bit CA registers, the scan-chain observation now has to derive $n+128$ bits in order to completely break the system. Though scan-attacks may possibly be able to obtain the states of the cipher and the M_g registers, it may be possible to change operation of the system such that in test mode through scan-chain the adversary is actually able to obtain only the scrambled version of the cipher states. The modification may be done as follows:

if (`test_mode`)
return $(C \star M) \star M'$

This way the adversary needs to guess 128 bits of M, leading to a complexity of 2^{128*n} to break the cipher. Note that the 128 registers of M_g need not be disconnected from scan-chain while testing.

Fault Attacks. Introducing faults into the system, may due to the countermeasure induce faults in three parts, cipher state bits, duplicate cipher state bits and the masking generator M_g. Due to the masking scheme, the probability to actually get a faulty ciphertext is, 2^{128*2} due to the two masks, M and M' used in the scheme. Evidently, the countermeasure strengthens the cipher against fault attacks.

Power Attacks. According to the scheme, power attacks now have to take care of the 128-bit registers of M_g. Since, the AND and XOR gates of M_g will

introduce disturbances in the power traces of the cipher instead of having only the cipher implementation, essentially the adversary has to break a system of $2n + 128$ bits. Therefore, the attack is expected to be difficult. Further, most of the rules used in the CA are linear, therefore, power traces in breaking the mask generator are expected to be flat. Hence, it may be claimed that the countermeasure is safe against power attacks.

4 Conclusion

In this paper we have proposed a cellular automata based masking generator to countermeasure SCA on symmetric key ciphers. The masking generator is built using a non-linear hybrid CA. It is shown to be cryptographically robust. We have further analyzed the security of the countermeasure against popular SCA like, fault attacks, differential power attacks and scan-based side channel attack. The countermeasure does not induce much hardware overhead to the implementations and is fast in operation.

References

1. Agrawal, M., Karmakar, S., Saha, D., Mukhopadhyay, D.: Scan Based Side Channel Attacks on Stream Ciphers and Their Counter-Measures. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 226–238. Springer, Heidelberg (2008)
2. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
3. Barenghi, A., Breveglieri, L., Koren, I., Pelosi, G., Regazzoni, F.: Countermeasures against fault attacks on software implemented AES: effectiveness and cost. In: Proceedings of the 5th Workshop on Embedded Systems Security, WESS 2010, pp. 7:1–7:10. ACM, New York (2010)
4. 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, vol. 0, pp. 7–14 (2009)
5. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
6. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
7. Pal Chaudhuri, P., Roy Chowdhury, D., Nandi, S., Chattopadhyay, S.: CA and its Applications: A Brief Survey. Additive Cellular Automata - Theory and Applications 1 (1997)
8. Coron, J.-S., Goubin, L.: On Boolean and Arithmetic Masking against Differential Power Analysis. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 231–237. Springer, Heidelberg (2000)
9. Hojsík, M., Rudolf, B.: Differential Fault Analysis of Trivium. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 158–172. Springer, Heidelberg (2008)

10. Hojsík, M., Rudolf, B.: Floating Fault Analysis of Trivium. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 239–250. Springer, Heidelberg (2008)
11. Itoh, K., Takenaka, M., Torii, N.: DPA Countermeasure Based on the "Masking Method". In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 440–456. Springer, London (2002)
12. Saarinen, M.J.O.: Chosen-IV Statistical Attacks on e-Stream Stream Ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, pp. 5–19 (2006)
13. 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)
14. Karmakar, S., Mukhopadhyay, D., Roy Chowdhury, D.: *d*-Monomial Tests of Nonlinear Cellular Automata for Cryptographic Design. In: Bandini, S., Manzoni, S., Umeo, H., Vizzari, G. (eds.) ACRI 2010. LNCS, vol. 6350, pp. 261–270. Springer, Heidelberg (2010)
15. Koc, C.K., Apohan, A.M.: Inversion of Cellular Automata Iterations. IEE Proceedings on Computers and Digital Techniques 144(5), 279–284 (1997)
16. Malkin, T.G., Standaert, F.-X., Yung, M.: A Comparative Cost/Security Analysis of Fault Attack Countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 159–172. Springer, Heidelberg (2006)
17. Meier, W., Staffelbach, O.: Analysis of Pseudo Random Sequences Generated by Cellular Automata. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 186–199. Springer, Heidelberg (1991)
18. Paterson, K.G., Blackburn, S.R., Murphy, S.: Theory and Applications of Cellular Automata in Cryptography. IEEE Transactions on Computers 46(5) (1997)
19. Sere, A.A., Iguchi-Cartigny, J., Lanet, J.-L.: Automatic detection of fault attack and countermeasures. In: Proceedings of the 4th Workshop on Embedded Systems Security, WESS 2009, pp. 7:1–7:7. ACM, New York (2009)
20. Wolfram, S.: Cryptography with Cellular Automata. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 429–432. Springer, Heidelberg (1986)
21. Wolfram, S.: Random Sequence Generation by Cellular Automata. Advances in Applied Mathematics 7, 123–169 (1986)
22. Yang, B., Wu, K., Karri, R.: Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard. In: ITC 2004: Proceedings of the International Test Conference, Washington, DC, USA, pp. 339–344 (2004)