Multiple Fault Attack on PRESENT with a Hardware Trojan Implementation in FPGA

Jakub Breier and Wei He
Physical Analysis and Cryptographic Engineering,
Temasek Laboratories at Nanyang Technological University
Singapore
{jbreier, he.wei}@ntu.edu.sg

***Abstract*—Internet of Things connects lots of small constrained devices to the Internet. As in any other environment, communication security is important and cryptographic algorithms are one of many elements that we use in order to keep messages secure. It is necessary to use algorithms that do not require high computational power, lightweight ciphers are therefore an ideal candidate for this purpose. Since these devices work in various environments, it is necessary to test security of implementations of cryptographic algorithms.**

In this paper, we explore a possibility of attacking an ultra-lightweight cipher PRESENT by using a multiple fault attack. Utilizing the Differential Fault Analysis technique, we were able to recover the secret key with two faulty encryptions and an exhaustive search of 2^{16} remaining key bits. Our attack aims at four nibbles in the penultimate round of the cipher, causing faulty output in all nibbles of the output. We also provide a practical attack scenario by exploiting Hardware Trojan (HT) technique for the proposed fault injection in a Xilinx Spartan-6 FPGA.

***Keywords*-Fault Attack, PRESENT, FPGA, Hardware Trojan**

I. Introduction

Internet of Things brings new challenges into the security field. With interconnection of a huge number of small devices with constrained computational capabilities, there is a need to design algorithms and protocols simple enough to be run on such devices within a reasonable time frame yet still preserving high level of security. Lightweight cryptography provides algorithms that use operations fulfilling such requirements. It delivers adequate security and does not always lower the security-efficiency ratio [15]. Currently there are many lightweight cryptography algorithms available, providing various security levels and encryption speed [11]. For the further work we have chosen the PRESENT algorithm.

PRESENT is an ultra-lightweight block cipher, introduced by Bogdanov et al. in 2007 [4]. The algorithm was standardized in 2011, by the ISO/IEC 29192-2:2011 standard. It is based on SP-network, therefore uses three operations in each round – 4-bit non-linear substitution, bit permutation and `xor` with the key. The best cryptanalysis presented so far is a truncated differential attack on 26 out of 31 rounds presented in 2014 [3].

Fault attacks exploit a possibility to change the intermediate values in the algorithm execution so that it can radically reduce the key search space or even reveal the key. The first attack was proposed by Boneh, DeMillo and Lipton in 1996 [5], following by the practical attack by Biham and Shamir one year later [2].

Currently, fault analysis is a popular method to attack cryptographic implementations, utilizing clock/voltage glitch techniques, diode laser and ION beam irradiation, EM pulse, or hardware trojans. For attacking symmetric block ciphers, the most popular technique is Differential Fault Analysis (DFA), in which the fault is usually inserted in the last rounds of a cipher for observing differences between correct and faulty ciphertexts. Other techniques include Collision Fault Analysis (CFA), Ineffective Fault Analysis (IFA), Safe-Error Analysis (SEA) [7], [8]. We have chosen DFA as our attack technique, with inserting multiple faults in the penultimate round of PRESENT cipher.

In our work we present a novel multiple fault attack on PRESENT. By injecting four nibble-switch faults in the penultimate round we were able to recover the secret key with only two faulty ciphertexts and an exhaustive search of 2^{16} remaining bits of the key. In both faulty ciphertexts, it is necessary to flip different nibbles in order to produce a different fault mask in the last round. We provide a practical attack scenario by exploiting Hardware Trojan (HT) technique for fault injection in a Xilinx Spartan-6 FPGA.

The rest of the paper is organized as follows. Section II provides an overview of known fault attacks on PRESENT cipher and Section III describes this cipher in details. Our attack model is proposed in Section IV and HT implementation of our attack is described in Section V. Finally, Section VI concludes our work and provides motivation for further research.

II. Related Work

The first DFA attack on PRESENT was published in 2010 by G. Wang and S. Wang [22]. Their attack aimed at a single nibble and they were able to recover the secret key with the computational complexity of 2^{29} by using 64 pairs of correct and faulty cipher texts on average.

Zhao et al. [23] proposed a fault-propagation pattern based DFA and demonstrated this technique on PRESENT and PRINT ciphers. Their attack on PRESENT-80/128 uses 8 and 16 faulty cipher texts on average and reduces the master key search space to $2^{14.7}$ and $2^{21.1}$.

Gu et al. [12] used a combination of differential fault analysis and statistical cryptanalysis techniques to attack lightweight block ciphers. They tested their methodology on PRESENT-80 and PRINT-48. The attack on PRESENT is aimed at middle rounds of algorithm, using single random S-box and multiple S-boxes fault attack. The main outcome of the paper is an extension of the fault model from the usual fault model, which aims at ultimate or penultimate rounds, to the other rounds as well.

Bagheri et al. [1] presented two DFA attacks on PRESENT, the first one attacks a single bit and the second one attacks a single nibble of the intermediate state. They were able to recover the secret key with 18 faulty cipher texts on average, using the second attack.

The most efficient attack so far was proposed by Jeong et al. [13]. They used a 2-byte random fault model, attacking the algorithm state after round 28. For PRESENT-80, they needed two 2-byte faults and for PRESENT-128 they needed three 2-byte faults and an exhaustive search of $2^{22.3}$ on average.

Hardware Trojans have drawn much attention during the past decade due to their severity in security-sensitive embedded systems [9]. As an enormous network of diverse embedded devices, Internet of Things (IoT) is populated with a great number of ICs, to collect, encrypt, transmit, and store data. For each node inside an IoT system, a complete functional chip normally consists of sorts of IPs, and they are typically designed and manufactured by off-shore design houses or foundries. In theory, any parties involving into the design or manufacturing stages can make alterations in the circuits for malicious purposes [16] [18]. These tiny changes or extra logic can hide inside the system during the majority of its lifetime until a specific activation mechanism is awaken for pilfering secrets or impairing the main functionality.

As a typical stealthy modification to ICs, a Hardware Trojan (either named Trojan Horse) was intentionally integrated into embedded devices for disabling/destroying a system, leaking confidential information from side channels, or triggering critical faults [19] [17] [14]. HT can be implanted into the circuit at multiple stages with a stealthy nature. The post-manufacturing testing often fail to detect it since Trojan only influences the circuit under specific conditions [6]. At a proper future time, the Trojan can be activated. Unlikely to the counterpart Software Trojan (ST), HT cannot be removed by upgrading the software in each device. So HT is truly furtive and ineradicable which hence poses more serious threat to the system security, particularly to the cryptographic blocks inside the IoT system. HT basically consists of two components: (a) the trigger signal that performs to activate the inserted trojan, and (b) the payload that is affected by the trojan [20]. Actually, many solutions have been proposed for detecting the implanted trojans, such as the fine-grained optical inspections [21] and the *side-channel* based comparison with the fully trustworthy "golden chip". To increase the difficulties of HT detection, the trojan size is preferably to be as small as possible, *w.r.t* its host design. In this paper, a compact Trojan module is presented which serves to inject multiple faults into specific algorithmic points in PRESENT cipher, which makes the proposed multiple fault attack approach realistic in practices.

Table I
PRESENT S-BOX.

x	0	1	2	3	4	5	6	7
$S(x)$	C	5	6	B	9	0	A	D

x	8	9	A	B	C	D	E	F
$S(x)$	3	E	F	8	4	7	1	2

III. Overview of PRESENT Cipher

PRESENT is a symmetric block cipher, based on SP-network. It consists of 31 rounds, block length is 64 bits and it supports keys with lengths of 80 and 128 bits. Considering the usage purposes, authors recommend the 80 bit key length version. Each round consists of three operations: XOR with the round key, substitution by the 4-bit S-box (Table I), and bit permutation (Table II). At the end of the cipher, a post-whitening XOR with the key is performed, so there are 32 generated keys in total. The pseudo-code of the encryption process is stated below.

```
PRESENT Algorithm:
$generateRoundKeys()$
for $i := 1$ to $31$ step $1$ do
    $addRoundKey(STATE, K_i)$
    $sBoxLayer(STATE)$
    $pLayer(STATE)$
end
$addRoundKey(STATE, K_{32})$
```

Key schedule of 80-bit version of the algorithm is following. First, a secret key is stored in register K and represented as $k_{79}k_{78}\dots k_0$. Since the block length is 64 bits, only 64 leftmost bits are used for each round: $K_i = k_{63k_{62}\dots k_0}$. Therefore, at round i we have:

$$K_i = k_{63}k_{62}\dots k_0 = k_{79}k_{78}\dots k_{16}.$$

After every round, key register K is updated in a following way:

1) $[k_{79}k_{78}\dots k_1k_0] = [k_{18}k_{17}\dots k_{20}k_{19}]$
2) $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3) $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus RC$

Table II
PRESENT PERMUTATION LAYER.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

where RC is a round counter.

IV. Attack Model

PRESENT algorithm uses sixteen 4-bit S-boxes. The output of S-boxes is an input for the permutation layer. The attack exploits following properties of the algorithm:

- Output of one S-box is an input for four different S-boxes.
- Input of one S-box consists of outputs from four different S-boxes.
- There are four different groups of S-boxes:
  - The outputs of S-boxes 0-3 are inputs for S-boxes 0,4,8,12,
  - The outputs of S-boxes 4-7 are inputs for S-boxes 1,5,9,13,
  - The outputs of S-boxes 8-11 are inputs for S-boxes 2,6,10,14,
  - The outputs of S-boxes 12-15 are inputs for S-boxes 3,7,11,15.

If it is possible to corrupt the whole output of some S-box (flip four bits), the fault will spread into four S-boxes in the following round, affecting the same bit position in every S-box. Therefore if we aim at four S-boxes from distinct groups in round 30, the fault will be distributed to every S-box in round 31. This fact is depicted in Figure. 1.

A. Attack Steps

The attack steps are following:

1) The attacker inserts four 4-bit faults at the output of four S-boxes from distinct groups.
2) She computes difference tables according to the fault model. More specifically, the bit faults in $S^{31}_{i,j}$, where $i \in 0,..,15$ are S-boxes of round 31 and $j \in 0,1,2,3$ are bits of particular S-box are following:
   - Fault at $S^{30}_0, S^{30}_4, S^{30}_8, S^{30}_{12}$ will result to faults at $S^{31}_{i,0}$.
   - Fault at $S^{30}_1, S^{30}_5, S^{30}_9, S^{30}_{13}$ will result to faults at $S^{31}_{i,1}$.
   - Fault at $S^{30}_2, S^{30}_6, S^{30}_{10}, S^{30}_{14}$ will result to faults at $S^{31}_{i,2}$.
   - Fault at $S^{30}_3, S^{30}_7, S^{30}_{11}, S^{30}_{15}$ will result to faults at $S^{31}_{i,3}$.

   Those difference tables contain the fault mask (bit position of a fault) and the output mask, which is the difference of the S-box output of a correct and faulty input. As an example, for masks 1000 and 0100, Table III shows these values.
3) Using a formula 1, the attacker observes the output differences and she searches for possible S^{31} input candidates.
4) She repeats steps 1-3, inserting fault in different nibbles, which will result to attacking different S-box bits at S^{31}.
5) Shee compares the possible candidate sets from both attacks, the intersection of these sets gives exactly one candidate, which is the correct state before S^{31}.
6) After obtaining all the key nibbles of K^{31}, the attacker uses an exhaustive search on the rest of the key, so the search space is 2^{16}.

$$\Delta = P^{-1}(C') \oplus P^{-1}(C) \tag{1}$$

B. Attack Example

After obtaining the faulty ciphertext and inverting the last round permutation layer, it is possible to obtain information about the input of the substitution layer. Let us assume that we attacked the first nibble after the S-box operation in round 30. Therefore the faulty mask of inputs of S-boxes 0,4,8,12 in round 31 will be 1000. We can now observe the changes in the output of the algorithm. Table III shows every possible input and output of the S-box operation, together with the faulty one, after attacking the most significant bit. It is easy to see that in the worst case, we have narrowed the input candidates to four numbers. For example, for the output difference 1111, we have input candidates 0, 7, 8, 9. The notation is following:

- *I* is a correct input of the S-box.
- *I'* is a faulty input of the S-box.

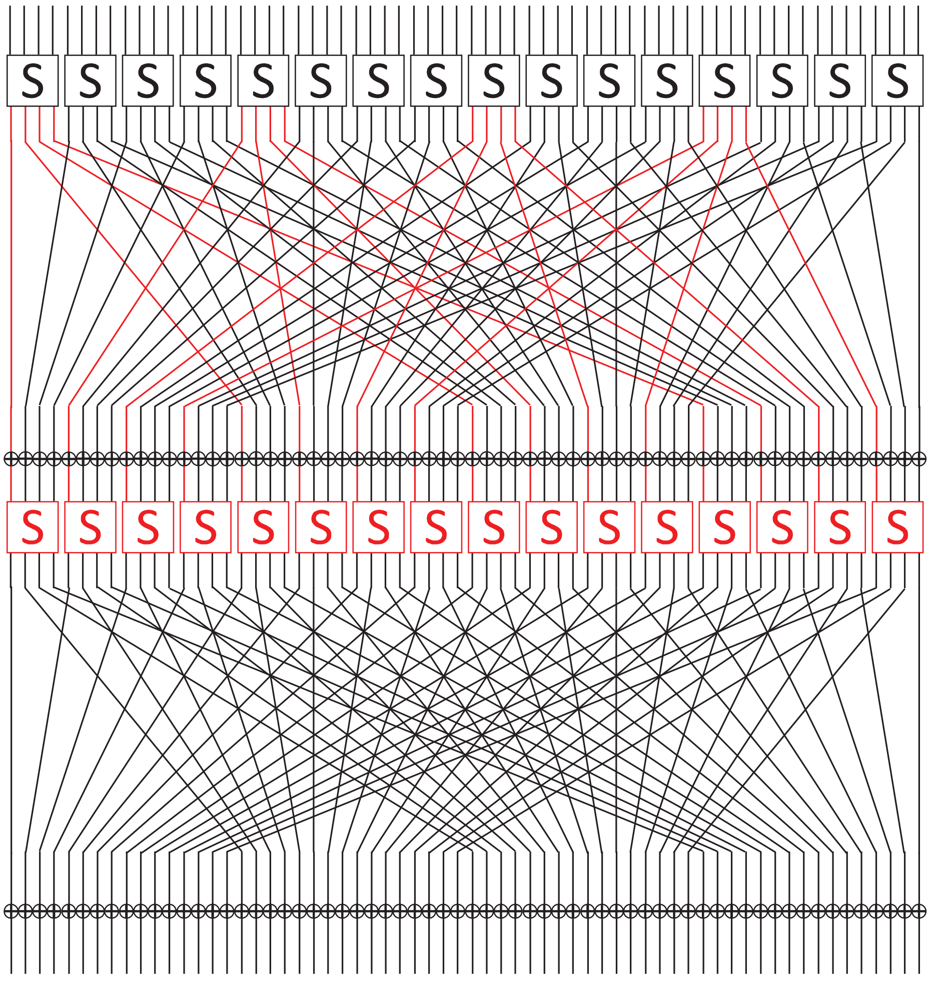
S S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S S

Figure 1. Fault propagation after attacking four S-boxes.

- O is a correct output of the S-box.
- O' is a faulty output of the S-box.
- Δ is the output difference between the correct and the faulty output.

The second step of the attack is to change the attacked nibble, so that it will affect another output nibble of the S-box belonging to the same group as the first one. Let us assume, we attacked the second nibble of the S-box output in round 30. Therefore the input of the same S-boxes in round 31 will be changed, but the faulty mask will be 0100 in this case. The outputs for this case are stated in table III. It is easy to see that the groups of the input values producing the same differences after the fault are not overlapping with the first attack, therefore we can determine the input value with certainty. For example if the output difference in this case would be 0101, the possible input candidates are 0, 1, 4, 5. The only common number for both attacks is 0, therefore it is the input value for the given S-box.

Using the simplified attack model with only one faulty nibble, it is possible to reveal the last round key with 8 faulty encryptions, since it is necessary to attack two distinct nibbles of each of four groups of S-boxes. If there is a possibility to inject multiple faults per encryption, the last round key could be revealed with 2 encryptions. In each run, the attacker would inject the fault into one nibble of the S-box output of four different groups, changing the nibbles after the first run. In both cases, an exhaustive search of 2^{16} is required to obtain the whole PRESENT-80 key.

V. Hardware Trojan for Practical Fault Injection

In this section, an FPGA implemented PRESENT cipher for the described multiple fault analysis approach will be detailed. The hardware realization relies on a Spartan-6 FPGA (XC6SLX4), soldered on Diligent Cmod-S6 commercial board [10]. We mounted the injections using the hardmacro-based Hardware Trojan (HT) [19] by inserting specially

Table III
FAULTS AND DIFFERENCES WITH FAULT MASKS 1000 AND 0100.

Mask 1000					Mask 0100				
I	I'	O	O'	Δ	I	I'	O	O'	Δ
0	8	C	3	1111	0	4	C	9	0101
1	9	5	E	1011	1	5	5	0	0101
2	A	6	F	1001	2	6	6	A	1100
3	B	B	8	0011	3	7	B	D	0110
4	C	9	4	1101	4	0	9	C	0101
5	D	0	7	0111	5	1	0	5	0101
6	E	A	1	1011	6	2	A	6	1100
7	F	D	2	1111	7	3	D	B	0110
8	0	3	C	1111	8	C	3	4	0111
9	1	E	5	1011	9	D	E	7	1001
A	2	F	6	1001	A	E	F	1	1110
B	3	8	B	0011	B	F	8	2	1010
C	4	4	9	1101	C	8	4	3	0111
D	5	7	0	0111	D	9	7	E	1001
E	6	1	A	1011	E	A	1	F	1110
F	7	2	D	1111	F	B	2	8	1010

Notation	
I	Original input
I'	Faulty input
O	Original output
O'	Faulty output
Δ	Difference between O and O'

created trojan modules into the algorithmic networks in FPGA scenario.

A. FPGA Scenario

Field Programmed Gate Array (FPGA) has been widely utilized in almost all digital/hybrid logic applications due to its rapid implementation, low cost and high performance. The major advantage of hardware implementation for cipher lies within its parallel computation networks that allow multiple logic chains computed in parallel, which results in high computational speed compared to the microcontroller scenarios. In our work, PRESENT cipher was implemented inside a compact Spartan-6 FPGA. The cipher is structured in loops with 16 4-bit S-boxes in parallel. The complete encryption is clocked with a global clock signal and the intermediate values from each round are stored in 128 1-bit registers. So a complete encryption in out implementation consists of 32 clock cycles, as seen in Figure 2.

B. Hardware Trojan for Fault Injection

HT typically requires some trigger signals to activate the inserted trojan modules. By the principle of the proposed multiple fault injection, a 1 bit signal of *trojan_trigger* is activated at the 30th encryption round when the fault perturbation is required. Since the injection point is at the output of the S-box, the specially devised trojan hardmacro for S-boxes 0, 4, 8, 12, and for S-boxes 1, 5, 9, 13 can be inserted during the chip fabrication or at the off-the-shelf stage. The payload is highlighted in the grey box of Figure 3. Signal of the *flip_trigger* is used to control the injection into different S-box groups. Since only 2 injection rounds are needed for the proposed fault attack, another 1 bit signal is enough. So there are totally 2 bit signals acting as the trigger in this solution. In Spartan-6 FPGA, every slice consists of 4 look-up tables (LUT), 8 multiplexers and 8 flip-flops. Even each LUT can either be used as one 6-input 1-output Boolean function or two 5-input 1-output functions, only 4 multiplexers in each slice has an external input that we have to use as the 1 input bit of the XOR gate and the input of multiplexer after the XOR gate. Therefore one slice can actually implement 4 HT modules, and 8 extra slices, *i.e.,* 4 Configurable Logic Blocks (CLBs), are sufficient for inserting all the 8 trojans. The states of the 2-bit trigger signals are given in Table IV. It is emphasized that the insertion of trojan can as well apply to different S-box groups, just obeying the principles explained in Section IV.

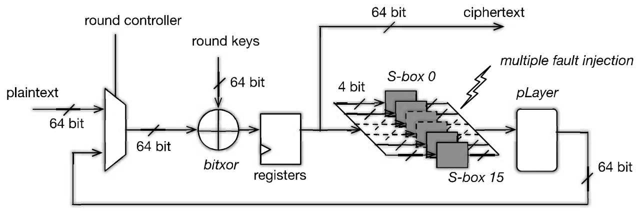

Figure 2. Parallel *Present* block cipher in FPGA.

Table IV
TROJAN TRIGGER STATE.

		trojan_trigger	
		'0'	'1'
flip_trigger	'0'	encryption	1st insertion
	'1'		2nd insertion

VI. CONCLUSIONS

In our paper we have proposed a multiple fault injection attack on PRESENT cipher, using DFA technique. By flipping four nibbles in the penultimate round we were able to obtain the secret key using two faulty ciphertext and an exhaustive search of a 2^{16} bits. We have implemented a hardware trojan, causing this type of fault on an FPGA implementation of PRESENT.

There are two other possible attack scenarios. The first one is flipping more than 4 bits, therefore the attack would affect

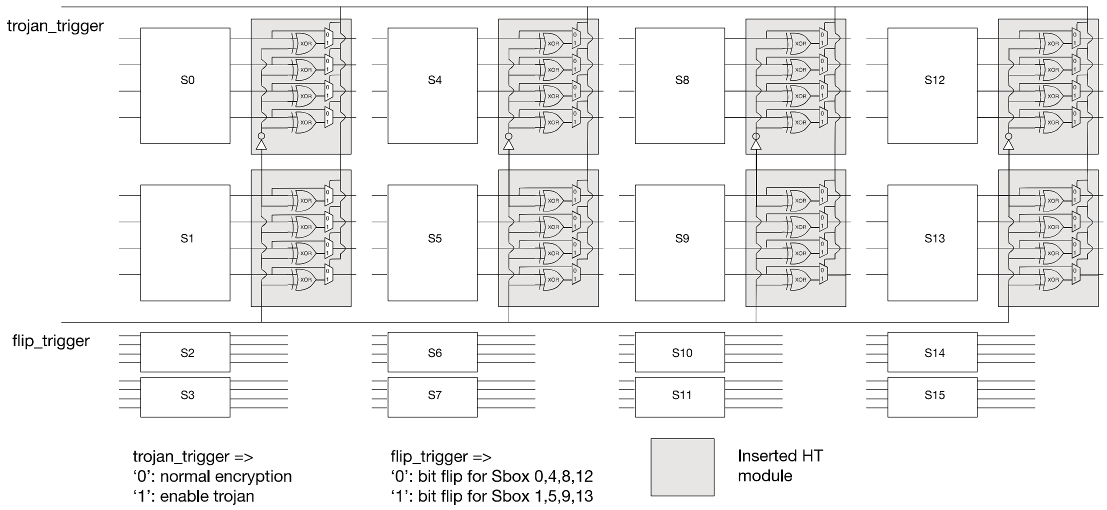

Figure 3. Trojans inserted into the signal paths after specific S-boxes.

the following nibble as well. In this case, more than one bit of particular S-box in round 31 will be affected. The attack can be still executed in the same way, only the differential table for the concrete fault mask has to be computed. It can be shown, that intersection of arbitrary two masks from two distinct differential tables gives exactly one candidate in every case, so it is possible to use multiple-bit fault masks.

The other scenario is flipping less than 4 bits in one nibble. In this case, the fault doesn't spread into every S^{31} S-box. The solution for this scenario is to use more faults, therefore the number of encryptions will increase.

The hardware trojan approach supporting the proposed multiple fault attack relies on the hardmacro based HT modules to be inserted into the outputs of specific S-boxes. In our tested Spartan-6 FPGA, only 4 CLBs are sufficient to implement the 8 trojan modules combined with 2 bit global trigger signal. Since the trojan blocks are inserted into the signal path of specific S-box outputs, without altering the main functionality of the ciphers, the trojan can be mounted at almost all design stages, such as front-end HDL coding or netlist alteration, as well as back-end layout or sub-gate manipulation.

In the subsequent work, we would like to focus on the realistic trojan insertion using the proposed multiple fault analysis into the security modules of an IoT scenario, and the related HT detection technique will also be emphasized.

References

[1] N. Bagheri, R. Ebrahimpour, and N. Ghaedi. New differential fault analysis on PRESENT. *EURASIP Journal on Advances in Signal Processing*, 2013(1):1–10, 2013.

[2] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In J. Kaliski, BurtonS., editor, *Advances in Cryptology CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer Berlin Heidelberg, 1997.

[3] C. Blondeau and K. Nyberg. Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In E. Oswald and P. Q. Nguyen, editors, *Eurocrypt 2014*, volume 8441 of *Lecture Notes in Computer Science*. Springer-Verlag, 2014.

[4] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin Heidelberg, 2007.

[5] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, pages 37–51, Berlin, Heidelberg, 1997. Springer-Verlag.

[6] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik. Hardware trojan insertion by direct modification of fpga configuration bitstream. *Design & Test, IEEE*, 30(2):45–54, 2013.

[7] C. Clavier. Attacking Block Ciphers. In M. Joye and M. Tunstall, editors, *Fault Analysis in Cryptography*, Information Security and Cryptography, pages 19–35. Springer Berlin Heidelberg, 2012.

[8] J.-L. Danger, S. Guilley, P. Hoogvorst, C. Murdica, and D. Naccache. A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards. *Journal of Cryptographic Engineering*, 3(4):241–265, 2013.

[9] U. O. Defense. Defense science board task force on high performance microchip supply. *Washington, DC*, pages 2005–02, 2005.

[10] Digilent. Cmod s6 reference manual, xilinx. *Inc., October*, Jan 30, 2014.

[11] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrion, J. Großschädl, and A. Biryukov. Triathlon of lightweight block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/209, 2015. http://eprint.iacr.org/2015/209.pdf.

[12] D. Gu, J. Li, S. Li, Z. Ma, Z. Guo, and J. Liu. Differential fault analysis on lightweight blockciphers with statistical cryptanalysis techniques. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, pages 27–33, Sept 2012.

[13] K. Jeong, Y. Lee, J. Sung, and S. Hong. Improved differential fault analysis on present-80/128. *International Journal of Computer Mathematics*, 90(12):2553–2563, 2013.

[14] Y. Jin, N. Kupp, and Y. Makris. Experiences in hardware trojan design and implementation. In *Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on*, pages 50–57. IEEE, 2009.

[15] M. Katagi and S. Moriai. Lightweight cryptography for the internet of things. *Sony Corporation*, 2008.

[16] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. *LEET*, 8:1–8, 2008.

[17] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson. Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 382–395. Springer, 2009.

[18] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, and W. Clay. Process reliability based trojans through nbti and hci effects. In *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, pages 215–222. IEEE, 2010.

[19] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. 2010.

[20] M. Tehranipoor and C. Wang. *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

[21] R. Torrance and D. James. The state-of-the-art in semiconductor reverse engineering. In *Proceedings of the 48th Design Automation Conference*, pages 333–338. ACM, 2011.

[22] G. Wang and S. Wang. Differential Fault Analysis on PRESENT Key Schedule. In *Computational Intelligence and Security (CIS), 2010 International Conference on*, pages 362–366, Dec 2010.

[23] X. Zhao, S. Guo, T. Wang, F. Zhang, and Z. Shi. Fault-propagate pattern based DFA on PRESENT and PRINT cipher. *Wuhan University Journal of Natural Sciences*, 17(6):485–493, 2012.