Strengthening SIMON Implementation Against Intelligent Fault Attacks

Jaya Dofe, *Student Member, IEEE*, Jonathan Frey, *Student Member, IEEE*, Hoda Pahlevanzadeh, *Student Member, IEEE*, and Qiaoyan Yu, *Member, IEEE*

***Abstract*—Driven by malicious intent, attackers are impelled to extract the cipher key and thus compromise the cryptosystem through fault attacks. Existing fault-detection methods can effectively detect random faults in the cipher implementation, but yield a high fault bypass rate (FBR) under intelligent fault attacks. To address this limitation, we propose a new microarchitecture to thwart fault attacks that place mathematically symmetric faults on the two encryption data paths. To further reduce the FBR for a new lightweight cipher SIMON, we propose a new countermeasure that integrates operand permutation and masking techniques. Closed-form expressions for depermutation and demasking in SIMON are provided in this letter. Our method was assessed under two fault attack scenarios (random and symmetric fault injections) with bit-flip, stuck-at-0, and stuck-at-1 fault models. Simulation results show that our method minimizes the FBR to zero with the fault attack scenarios of symmetric fault location and stuck-at-0 fault injections. Overall, the proposed method outperforms the existing fault-detection methods in multiple fault attack conditions, at the cost of 5% more area overhead than the most hardware-efficient fault detection method.**

***Index Terms*—Block cipher, cryptography, fault attacks, fault bypass rate, fault detection, security, simon.**

I. Introduction

SECURITY is a crucial concern for the development of trustworthy embedded systems. Cryptographic algorithms are commonly used to ascertain security of the system. Driven by malicious intent, attackers are impelled to extract the cipher key and thus compromise the embedded system. As embedded systems are typically hardware constrained, lightweight ciphers are preferred and effective fault-detection methods for lightweight ciphers are needed.

Fault attacks can compromise the cipher implementation and produce faulty ciphertexts for cryptanalysts to retrieve the secret key [1], [5], [7], [11], [13], [15]. White light, laser beams, voltage glitches, and temperature control are possible manipulation means to perform fault attacks [2]. Faults intentionally induced by fault attacks are different than random faults caused by a radioactive environment or manufacturing defects, in terms of fault location, injection timing, and faulty vector size. Consequently, existing fault-detection methods [3], [6], [8]–[12], [14] designed for random faults may not be sufficient to handle intelligent fault attacks, which place mathematically symmetric faults on the two encryption data paths.

In this letter, we highlight the limitation of existing fault-detection methods against intelligent fault attacks. To address the limitation, we propose a new microarchitecture to prohibit the generation of faulty ciphertexts. Moreover, we propose a new countermeasure that strengthens the SIMON [4] implementation, in order to reduce the availability of faulty ciphertexts caused by both random faults and fault attacks. As the cryptanalysis relies on faulty ciphertexts to retrieve the secret key, our countermeasure has the potential to increase the fault attack time and reduce the attack success rate.

Manuscript received June 08, 2015; accepted August 19, 2015. Date of publication September 08, 2015; date of current version December 01, 2015. This manuscript was recommended by A. Coskun.

The authors are with the Department of Electrical and Computer Engineering, University of New Hampshire, Durham, NH 03824 USA (e-mail: jhs49@wildcats.unh.edu; jpg73@wildcats.unh.edu; htr23@wildcats.unh.edu; qiaoyan.yu@unh.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2015.2477273

II. Preliminaries

A. Random Fault Injection Versus Fault Attacks

Fault attacks have a different nature, compared to random faults that typical fault-detection techniques handle. Random faults due to hardware defects and operating environment (e.g., radiation and temperature variation) are uncontrollable and unpredictable in terms of fault locations and occurrence moments. In contrast, fault attacks in cryptosystems are intentionally induced by the attacker to retrieve the key [1], [5], [7], [11], [13], [15]. Faults brought forth by fault attacks are controlled by the attacker and have predictable location and timing. For instance, the attacker prefers to inject faults at the last few rounds of AES [5], [8]. Another example, if the attacker targets the left half input of the penultimate round in SIMON, the key can be retrieved from the last round [16].

B. Limitation of Existing Countermeasures

Fault-detection methods are a typical countermeasure to ensure the integrity of the cryptographic process. When fault attacks become a security concern, conventional fault-detection methods were inherited naturally [2]. In the evaluation stage, the existing works [3], [6], [9]–[12], [14] only consider the scenario that faults randomly happen in the cipher implementation. As fault attacks are intentional, we need to rigorously consider that the attacker may inject *intelligent faults* that have potential to bypass the fault-detection mechanism. For instance, the fault-detection methods shown in Fig. 1 cannot detect the faults injected in the input, i.e. intermediate state registers. This is because both data paths receive the same faulty input, and no difference will be observed at the comparison stage. As a result, a faulty ciphertext can bypass the fault detection.

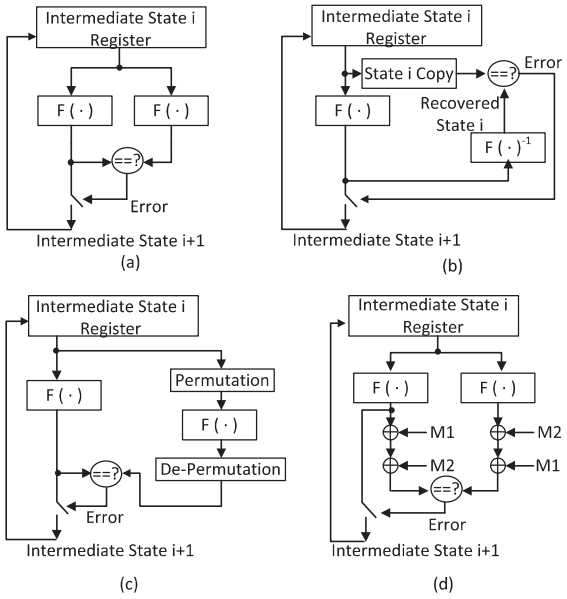

Fig. 1. Typical fault-detection methods against fault attacks. (a) DMR [3], (b) inverse function [11], (c) permutation [9], [10], and (d) masking. $\mathrm{F}(\cdot)$ represents one round function for the cryptographic algorithm.

Moreover, each method has its own limitation against mathematically symmetric fault injection (i.e., intelligent faults). We refer the symmetric faults to the two faults injected in the two data paths of the encryption unit in a way that the result of both faults will nullify the effect on the fault detection stage. In the double modular redundancy (DMR) method in Fig. 1(a), if the fault attack is performed at the same time and location on the two identical $\mathrm{F}(\cdot)$ functions, the DMR detection will fail. For a lightweight cipher like SIMON, the inverse function for fault detection in Fig. 1(b) is likely designed with a symmetric architecture. The inverse function method may fail too, if the recovered intermediate state i is corrected by another fault injection on the inverse data path before the comparison stage. In the work [9], [10], the intermediate state is permutated before $\mathrm{F}(\cdot)$, as shown in Fig. 1(c). Permutation can possibly fail because the permutated input bits may share the same logic value [10]. Because of the property of XOR, the masking scheme shown in Fig. 1(d) cannot detect the faults injected on the two $\mathrm{F}(\cdot)$ function, as long as the overall fault bits and injection locations are the same. This problem cannot be resolved even if the order of masking is extended [13].

III. Proposed Countermeasure Against Fault Attacks

To address the faulty ciphertexts bypassing problem, we propose a new microarchitecture and a new fault-attack countermeasure algorithm for SIMON.

A. Proposed Microarchitecture

Our new microarchitecture against fault attacks is shown in Fig. 2. The *compute* signal is used to select the plaintext or the intermediate state from the previous round iteration. The two data paths, the encryption $\mathrm{F}(\cdot)$ alone and the fault-detection $\mathrm{G}(\cdot)$ followed with $\mathrm{F}(\cdot)$, are executed simultaneously and their outputs (i.e., intermediate state) are fed to registers 1 and 2, respectively. The inverse fault-detection function $\mathrm{G}(\cdot)^{-1}$ and comparison of CWs are performed before the beginning of the next round. If the inconsistency between CW1 and CW2 is observed, the fault injected in the two data paths is detected and the faulty ciphertext is dropped. Thus, the faulty ciphertext is not available for cryptanalysis. By producing and savings two different CWs, our microarchitecture eliminates the risk originated from the use of a single intermediate state register shown in Fig. 1. Even if the attacker can manipulate the two registers at a certain degree, the chance to successfully bypass the fault detection will be significantly lower than the single register method.

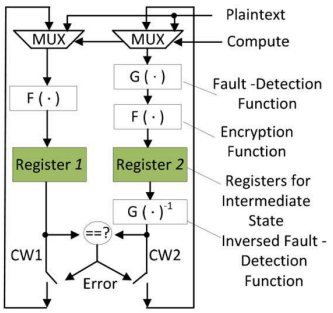

Fig. 2. Proposed fault detection microarchitecture for fault attacks.

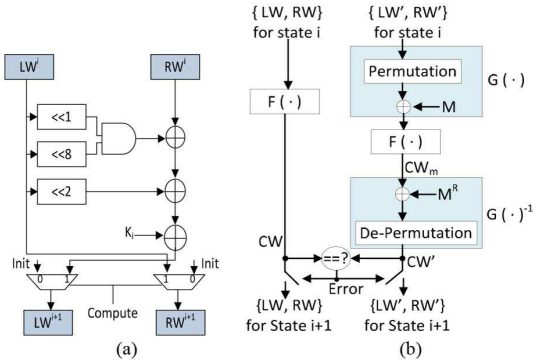

Fig. 3. Fault-detection for SIMON. (a) $\mathrm{F}(\cdot)$ for SIMON round function. (b) proposed fault-detection algorithm. '$\ll$' is a circular left shifting function. For mathematical analysis purpose, we exclude the registers here.

B. Fault-Attack Countermeasure Algorithm for SIMON

We apply the proposed microarchitecture to SIMON. As depicted in Fig. 3(a), the round function of the SIMON cryptographic algorithm is composed of three circular shifting, one bitwise AND, and three bitwise exclusive-OR functions. We propose to integrate permutation and masking techniques to address intelligent fault attacks in the iterative SIMON implementation as shown in Fig. 3(b). LW and RW represent the upper word and lower word, respectively, and K is the key. To recover the original intermediate state, we derive the depermutation and demasking functions for SIMON.

1) Derivation for Depermutation Function: We use two vectors $\mathrm{P} = [\mathrm{p}_0, \mathrm{p}_1, \ldots, \mathrm{p}_n]$, and $\mathrm{Q} = [\mathrm{q}_0, \mathrm{q}_1, \ldots, \mathrm{q}_n]$ as an example to introduce the process of permutation and depermutation. A circular left shifting ($\ll 3$) is adopted as a special permutation in the following example:

$$\begin{aligned}&(P \ll 3) \oplus (Q \ll 3)\\ &= (p_3 \hat{} q_3, \ldots, p_n \hat{} q_n, p_0 \hat{} q_0, p_1 \hat{} q_1, p_2 \hat{} q_2)\\ &= (P \oplus Q) \ll 3. \end{aligned} \quad (1)$$

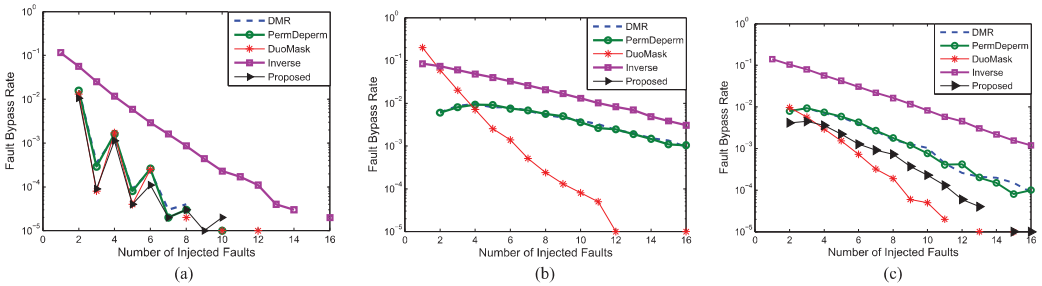

Fig. 4. Fault bypass rates for different fault-detection methods under fault attack scenario 1: random faults. (a) Bit-flip fault, (b) stuck-at-0 fault, and (c) stuck-at-1 fault. Fault bypass rate for some methods are not shown in the figures because of its fault bypass rate is zero.

If the operands P and Q are permutated by circular left shifting ($\ll 3$), the depermutation function to retrieve the original $P \oplus Q$ is just simply depermuting the output by circular right shifting ($\gg 3$). If we mask P by XORing a masking vector M before permutation, the permutation can be inversed as shown in (2). Certainly, we can generalize (2) by replacing "3" with any number. Also any bit reordering for permutation can be applied to (2)

$$\begin{aligned}&((P \oplus M) \ll 3) \oplus (Q \ll 3)\\ &= (P \ll 3) \oplus (M \ll 3) \oplus (Q \ll 3)\\ &= (P \oplus M \oplus Q) \ll 3.\end{aligned} \tag{2}$$

2) Derivation for Demasking Function: The SIMON round function is expressed in (3), where CW represents an intermediate state for the ciphertext

$$\begin{aligned}CW = &[(LW \ll 1) \,\&\, (LW \ll 8)] \oplus RW\\ &\oplus (LW \ll 2) \oplus K.\end{aligned} \tag{3}$$

As a masking vector is applied to LW before it goes through the encryption, we derive the demasking function in (4). This is not trivial because of the nonlinear function AND

$$\begin{aligned}CW_m = &[(LW_m \ll 1) \,\&\, (LW_m \ll 8)] \oplus RW\\ &\oplus (LW_m \ll 2) \oplus K\end{aligned} \tag{4}$$

in which, the masked intermediate state CW_m uses the masked input $LW_m = LW \oplus M$. We use the XOR and shifting property proved in (2) to simplify (4) and obtain (5)

$$\begin{aligned}CW_m &= [(C \oplus X) \,\&\, (D \oplus Y)] \oplus RW\\ &\quad \oplus (E \oplus Z) \oplus K\\ &= (CD \oplus RW \oplus K \oplus E)\\ &\quad \oplus (XD \oplus XY \oplus YC \oplus Z)\\ &= CW \oplus M^R\end{aligned} \tag{5}$$

where $\mathrm{C} = \mathrm{LW} \ll 1$, $\mathrm{D} = \mathrm{LW} \ll 8$, $\mathrm{E} = \mathrm{LW} \ll 2$, $\mathrm{X} = \mathrm{M} \ll 1$, $\mathrm{Y} = \mathrm{M} \ll 8$, and $\mathrm{Z} = \mathrm{M} \ll 2$.

By XORing the demasking vector $\mathrm{M^R}$ with $\mathrm{CW_m}$, we can retrieve the original CW. Note that $\mathrm{M^R}$ is not symmetric with M. The complete $\mathrm{G}(\cdot)^{-1}$ function is the integration of (1) and (5). The combination of masking and permutation increases the difficulty of performing a successful fault attack, as the mathematical calculation complexity and the time needed to produce a useful faulty ciphertext are increased.

IV. Simulation Results

We used a SIMON with a 64-bit plaintext and 96-bit key as a case study. We examined the fault bypass rate and hardware cost of different countermeasures applied to the SIMON round function. Our method was compared with four methods, DMR [3], inverse function [11], XORing two masking vectors (i.e. DuoMask), and permutation (i.e., PermDeperm) [9], [10]. The SIMON was implemented in an iterative fashion with Verilog HDL and synthesized in Synopsys Design Compiler with a TSMC 65 nm technology. The reported hardware cost was from the synthesized result. We also modeled the SIMON round function at gate level in MATLAB to assess the fault bypass rate. As the impact of fault attacks on the key scheduling function will be similar to the round function, we focus on the fault attacks on the round function in this letter.

A. Fault Bypass Rate

We define the fault bypass rate (FBR) as the ratio of the number of faulty ciphertexts that bypass the fault detection over the total number of test cases where the intermediate state is different than the fault-free version. This metric is important for cryptanalysis, because faulty ciphertexts are used to extract the key. The faster the attacker obtains the faulty ciphertext, the quicker the key can be retrieved. The minimization of the FBR can potentially cause a longer time to perform cryptanalysis. Three fault models (i.e., bit-flip, stuck-at-0 and stuck-at-1) and two fault attack scenarios are considered here. We also assume that the comparator circuit is fault-free [10].

1) Fault Scenario 1: Random Faults: Random faults means the faults are placed randomly on the fault-tolerant SIMON implementation. As shown in Fig. 4, FBR decreases as the number of faulty bits increases. The result shown in Fig. 4(a) is especially interesting. All methods except the inverse function one [11] obtain a better FBR for odd numbers of faults, as the effect of even number of faults on SIMON is cancelled through multiple XOR gates. Compared with other methods, our method minimizes FBR to 0% for stuck-at-0 faults [which is not shown in the logarithmically-scaled Fig. 4(b)]. For stuck-at-1 faults, our method is the second best if the number of faults exceeds three shown in Fig. 4(c). This is because our method uses more AND gates than DuoMask does. For bit-flip faults, our method achieves the best FBR if the number of faults is less than six.

2) Fault Scenario 2: Mathematically Symmetric Faults: Mathematically symmetric faults mean that the effect of the symmetric faults will be nullified before the comparator. As the structures of the fault detection methods in Figs. 1 and 3(b) are different, we varied the fault injection locations accordingly to

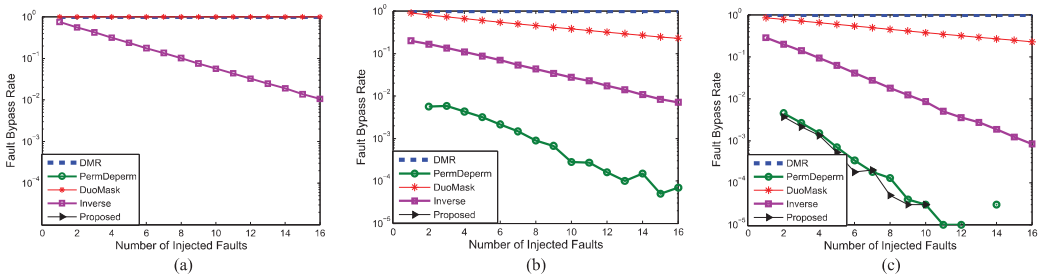

Fig. 5. Fault bypass rates for different countermeasure methods under fault attack scenario 2: faults on the symmetric location. (a) Bit-flip fault, (b) stuck-at-0 fault, and (c) stuck-at-1 fault. Fault bypass rate for some methods are not shown in the figures because of its fault bypass rate is zero.

TABLE I
COMPARISON OF AREA AND POWER CONSUMPTION

Cost / Design	Area (equivalent gates)	Total Power (mW)
DMR [3]	3393 (100%)	1.8837 (100%)
Inverse [11]	3351 (99%)	1.7875 (95%)
DuoMask	3739 (110%)	1.9347 (103%)
PermDeperm [9, 10]	3393 (100%)	1.8837 (100%)
Proposed	3539 (104%)	1.8994 (101%)

ensure the mathematical symmetry in the experiment. In this fault scenario, our method achieves the best FBR performance over other methods. As shown in Fig. 5(a), the PermDeperm and our method can successfully detect all faults. For the case of stuck-at-0 faults shown in Fig. 5(b), the proposed method outperforms the PermDeperm method because of our masking process. If the faults are stuck-at-1 shown in Fig. 5(c), our method cannot detect the faults injected in the AND gates due to logical masking, but it is still superior to other methods.

B. Area Cost and Power Consumption

The hardware cost for each method is shown in Table I. The area includes the complete implementation of the baseline SIMON and fault-detection methods. For fair comparison, we implemented the methods in [3], [9]–[11] with our new microarchitecture to address the faults injected in the intermediate state registers. PermDeperm [9] shifts the operand by hard wiring and thus does not cost extra hardware over DMR. Our method only increases the area overhead by 5% over the most hardware-efficient fault detection method (inverse [11]). Our method consumes slightly less area than DuoMask because of the single masking vector. As the maximum delay of the five designs is 2 ns, we set the clock frequency to 500 MHz. The proposed method increases the total power by 6% over the inverse function [11].

V. CONCLUSION

The security of a lightweight cipher implementation is crucial for embedded systems. To retrieve the secret key, attackers perform fault attacks to generate faulty ciphertexts for the follow-up cryptanalysis. The existing countermeasures for block ciphers are designed for random faults, rather than intelligent fault attacks, in which the attack intents to bypass the fault detection. In this letter, we differentiate fault attacks from the random fault injection, and highlight the limitation of the existing methods against mathematically symmetric fault attacks. We propose a new microarchitecture for fault attack detection to reduce the probability of faulty ciphertexts bypassing fault detection. For SIMON, we propose a new fault-attack countermeasure algorithm, which integrates the operand permutation and masking techniques. To the best of our knowledge, this is the first work that provides the closed-form expressions of the depermutation and demasking functions for SIMON. Our case study shows that the proposed method outperforms the existing methods in random and symmetric fault-attack scenarios, where the bit-flip or stuck-at-0 fault models are applied. The limitation of our method is 5% more area and 6% more power consumption over the most-efficient fault-detection method.

REFERENCES

[1] A. Moradi *et al.*, "A generalized method of differential fault attack against AES cryptosystem," in *Proc. CHES*, 2006, pp. 91–100.
[2] H. Bar-El *et al.*, "The Sorcerer's apprentice guide to fault attacks," Cryptology ePrint Archive, Tech. Rep. 2004/10, 2004.
[3] G. Di Natale *et al.*, "A reliable architecture for parallel implementations of the Advanced Encryption Standard," *J. Electron. Testing*, vol. 25, no. 4, pp. 269–278, 2009.
[4] A. Aysu, E. Gulcan, and P. Schaumont, "SIMON says: Break area records of block ciphers on FPGAs," *Embed. Syst. Lett.*, vol. 6, no. 2, pp. 37–40, Jun. 2014.
[5] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *Proc. WISTP*, 2011, pp. 224–233.
[6] M. Mozaffari Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron*, vol. 60, no. 12, pp. 5925–5932, Dec. 2013.
[7] J. Blomer and J. P. Seifert, "Fault based cryptanalysis of the advanced encryption standard (AES)," *FC 2003*, pp. 162–181, 2003, LNCS 2742.
[8] A. Barenghi *et al.*, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proc. IEEE*, vol. 100, no. 11, pp. 3056–3076, Nov. 2012.
[9] M. Mozaffari-Kermani *et al.*, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *IEEE Embed. Syst. Lett.*, vol. 6, no. 4, pp. 89–92, Dec. 2014.
[10] X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1595–1608, Oct. 2013.
[11] R. Karri *et al.*, "Concurrent error detection of fault-based side-channel cryptanalysis of 128-bit symmetric block ciphers," in *Proc. DAC'01*, pp. 579–585.
[12] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.
[13] V. Lomne, T. Roche, and A. Thillard, "On the need of randomness in fault attack countermeasures—application to AES," in *Proc. FDTC*, Sep. 2012, pp. 85–94.
[14] X. Guo *et al.*, "NREPO: Normal basis recomputing with permuted operands," in *Proc. HOST*, 2014, pp. 118–123.
[15] X. Guo *et al.*, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptographic Eng.*, vol. 5, no. 3, pp. 153–169, 2015.
[16] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, "Differential fault analysis on the families of SIMON and SPECK ciphers," in *Proc. FDTC*, 2014, pp. 40–48.