Fault Diagnosis Schemes for Low-Energy Block Cipher Midori Benchmarked on FPGA

Anita Aghaie, *Student Member, IEEE,* Mehran Mozaffari Kermani, *Senior Member, IEEE,* and Reza Azarderakhsh, *Member, IEEE*

***Abstract*— Achieving secure high-performance implementations for constrained applications such as implantable and wearable medical devices are a priority in efficient block ciphers. However, security of these algorithms is not guaranteed in the presence of malicious and natural faults. Recently, a new lightweight block cipher, Midori, has been proposed that optimizes the energy consumption besides having low latency and hardware complexity. In this paper, fault diagnosis schemes for variants of Midori are proposed. To the best of the authors' knowledge, there has been no fault diagnosis scheme presented in the literature for Midori to date. The fault diagnosis schemes are provided for the nonlinear S-box layer and for the round structures with both 64-bit and 128-bit Midori symmetric key ciphers. The proposed schemes are benchmarked on a field-programmable gate array and their error coverage is assessed with fault-injection simulations. These proposed error detection architectures make the implementations of this new low-energy lightweight block cipher more reliable.**

***Index Terms*— Fault diagnosis, field-programmable gate array (FPGA), Midori block cipher, reliability.**

I. Introduction

LIGHTWEIGHT cryptography plays an essential role for achieving high security with low area and low energy consumption in many sensitive applications such as secure embedded systems, wireless nanosensors, radio-frequency identification (RFID) tags, and implantable and wearable medical devices. Such an efficiency is more critical in energy-constrained applications such as implantable medical devices in which replacing discharged batteries with power-inefficient architectures is a burden due to the required surgeries to remove these batteries [1]. In addition, tiny computing devices such as RFID tags and sensors need efficient block ciphers because of their small area and limited source power [2]. Such a need for efficiency is fulfilled by lightweight block ciphers, which provide high security level, low energy consumption, and low hardware complexity. It is noted that the Advanced Encryption Standard (AES) has been optimized in terms of area and power consumption [3].

Midori provides acceptable security level with optimal energy consumption. The S-boxes of Midori are different from those of the AES and other lightweight block ciphers. Furthermore, Midori has two types of bijective 4-bit S-boxes that are more energy efficient than the 8-bit ones. It is noted that Midori, like other lightweight block ciphers, accepts optimal cell permutation matrices and uses the most efficient maximum distance separable (MDS) matrices due to low implementation overheads and increasing immunity against several attacks [4], [5].

Error detection in crypto architectures has been the center of attention in [6]–[13]. The prior work has focused on various time and hardware redundancy approaches (including the approaches that are dependent or oblivious of the implementation platform and the algorithm architecture). However, in the case of Midori, to the best of the authors' knowledge, there is no prior work. The merit of the proposed approaches in this paper compared with that of the approaches presented before for lightweight block ciphers is twofold. First, we present both logic-gate-based and lookup table (LUT)-based error detection schemes for the two types of the S-boxes in Midori, which gives freedom to the designers to choose the implementation strategy based on the implementation and performance metric requirements and the platform to implement. Second, for the MixColumn operation, we have examined to achieve to have low-overhead detection approaches, by performing design space explorations before math not as an afterthought. Such careful investigations to have a combined original implementation and error detection architecture has not been performed in previous state-of-the-art approaches.

The performed simulation results show high error coverage (the percent of ratio of the number of detected errors to the number of injected faults) for the presented schemes. Using the proposed approaches, the error detection structures are capable of detecting the injected faults with high coverage (transient and permanent as well as single, multiple, and adjacent faults). We note that permanent faults, e.g., stuck-at faults, are caused by VLSI manufacturing defects (and of course if the intention is to break the entire device, such faults can be injected at runtime). There are well-established automatic test pattern generation based testing techniques to identify these faults [14]. On the other hand, "long transient faults" can lead to information leakage [15]. Simple time redundancy cannot detect long transient faults that last for

Manuscript received June 7, 2016; revised September 20, 2016 and November 8, 2016; accepted November 21, 2016. Date of publication December 14, 2016; date of current version March 20, 2017. This work was supported by the U.S. Department of Commerce, National Institute of Standards and Technology under the U.S. federal agency under Award 60NANB16D245.

A. Aghaie and M. Mozaffari Kermani are with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: aa6964@rit.edu; m.mozaffari@rit.edu).

R. Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: razarderakhsh@fau.edu).

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

Digital Object Identifier 10.1109/TVLSI.2016.2633412

the normal computation and recomputation, and attackers have successfully injected long transient faults to break this countermeasure [15]. Through field-programmable gate array (FPGA) implementations using the Xilinx Virtex-7 family, it is shown that the overheads of the proposed architectures are acceptable for resource-constrained applications.

This paper is organized as follows. In Section II, preliminaries related to the Midori block cipher are presented. The proposed error detection approaches are presented in Section III. In Section IV, the results of the fault injection simulations are explained. Furthermore, through FPGA implementations, the overheads are benchmarked. Eventually, conclusions are made in Section V.

II. Preliminaries

Midori consists of two parts, i.e., data processing and key scheduling modules. The plaintext input and the ciphertext output, which are 64 bits or 128 bits in width, are divided into 4-bit and 8-bit cells, respectively. This is also performed for the whitening key (WK) and round keys (RK_i). The round keys are used as the input to the main functions of the algorithm, and the WKs are modulo-2 added with the input and output of the entire encryption or decryption operation. Two variants of Midori, Midori64 and Midori128, are a 64-bit block cipher and a 128-bit block cipher with the same key length of 128 bits corresponding to 16 and 20 number of rounds, respectively.

Midori uses the following 4×4 array state:

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

in which the size of each cell is 4 bits and 8 bits for Midori64 and Midori128, respectively. Midori applies bijective S-boxes, Sb_0 and Sb_1, with a 4-bit structure and the involution property, which are used in Midori64 and Midori128, respectively. Midori128 utilizes four different 8-bit S-boxes, SSb_0, SSb_1, SSb_2, and SSb_3. Each of these 8-bit S-boxes consists of two 4-bit Sb_1 with permutation input and output structures, which are described in more detail in [4]. The S-boxes are utilized in each round function and apply the following four operations to the state matrix.

1) *SubCell (S):* The 4-bit and 8-bit S-boxes are used for each element of the state S in Midori64 and Midori128 in parallel. We have $s_i \leftarrow Sb_0[s_i]$ for Midori64 and $s_i \leftarrow SSb_{(imod4)}[s_i]$ for Midori128, where $0 \leq i \leq 15$.
2) *ShuffleCell (S):* Each byte of the state is derived as follows:

$$(s_0, s_1, \ldots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8).$$

3) *MixColumn (S):* Midori utilizes an involutive binary matrix M, applied to every $4m$-bit column of the state S, i.e., $(s_i, s_{i+1}, s_{i+2}, s_{i+3})^T \leftarrow M \times (s_i, s_{i+1}, s_{i+2}, s_{i+3})^T$ and $i = 0, 4, 8, 12$.
4) *KeyAdd (S, RK_i):* The ith n-bit round key RK_i is modulo-2 added to the state S.

Before the first round, an additional KeyAdd operation is applied, and in the last round, the ShuffleCell and MixColumn operations are omitted. The data processing part of Midori consists of its encryption and decryption that perform the mentioned round function for a specific number of rounds except the last round. The decryption is performed through the same sequence of the mentioned round function with a difference of the added InvShuffleCell.

III. Proposed Error Detection Schemes

In this section, the error detection approaches of sub-blocks in the Midori encryption and decryption are proposed.

A. Proposed Approaches for the S-Box Variants

In the hardware implementations of Midori, two approaches can be used for realizing the S-boxes, i.e., LUT-based and logic-gate-based implementations. The LUT-based S-boxes have advantages such as good performance and disadvantages such as having high area and power consumption. On the other hand, the latter approach typically has less area and power consumption.

Our proposed signature-based error detection approach is not confined to a special signature. However, for the sake of clarity, we present two examples, i.e., parity-based and interleaved parity-based approaches.

We can store predicted parities (or interleaved parities) of elements from the S array in LUTs. The scheme for the S-boxes Sb_0 and Sb_1 is based on deriving the predicted parities of the S-boxes using LUTs, as shown in Table I. For each element of S-boxes, we modulo-2 add all bits. Then, we store the result as a parity bit in an extended LUT with 5-bit elements (note that one extra bit is added to each 4-bit entry). Thus, the new protected state would consist of 16 5-bit elements that can be stored in FPGA block memories or pipelined distributed LUTs. An example would be to derive the parity of the first element of Sb_0, which is $\{c\}_{16} = \{1100\}_2$, which is zero.

The other signature-based error detection scheme is based on interleaved parity bits that are proposed in order to protect the nonlinear S-boxes. Interleaved-parity-based schemes are able to detect burst faults, i.e., adjacent multiple faults. Such faults happen in both natural defects and malicious fault attacks. In this scheme, we compute the interleaved parity bits of the 4-bit bijective S-boxes Sb_0 and Sb_1 in hexadecimal form, as shown in Table I. We have derived such parities by the modulo-2 addition of odd bits and even bits with each other separately. Similarly, these 2-bit interleaved parities along with 4-bit elements of each state are stored as 6-bit elements in memories. An example would be to derive the interleaved parity of the first element of Sb_0, which is $\{c\}_{16} = \{1100\}_2$, which is 11 (modulo-2 adding the odd and even bits separately).

We have also derived the formula for logic-based implementations of the two S-boxes Sb_0 and Sb_1, respectively. Suppose the inputs to the S-boxes are a, b, c, and d and the 4-bit outputs are a', b', c', and d'.

For Sb_0, we have derived $a' = \bar{c}\bar{a} \vee \bar{c}\bar{b} \vee \bar{a}\bar{d}$, $b' = \bar{d}\bar{a} \vee bc \vee acd$, $c' = bd \vee \bar{a}b \vee \bar{a}d$, and $d' = c(\bar{a} \vee \bar{b}) \vee d(\bar{a}b \vee a\bar{b})$.

TABLE I
(INTERLEAVED PARITY, PARTY) OF 4-bit BIJECTIVE S-BOXES Sb_0 AND Sb_1 IN HEXADECIMAL FORM

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Sb_0	(11,0)	(00,0)	(10,1)	(11,0)	(01,1)	(01,1)	(00,0)	(10,1)	(10,1)	(11,0)	(01,1)	(00,0)	(00,0)	(10,1)	(01,1)	(11,0)
Sb_1	(01,1)	(00,0)	(00,0)	(11,0)	(01,1)	(10,1)	(00,0)	(10,1)	(10,1)	(00,0)	(11,0)	(01,1)	(11,0)	(10,1)	(01,1)	(11,0)

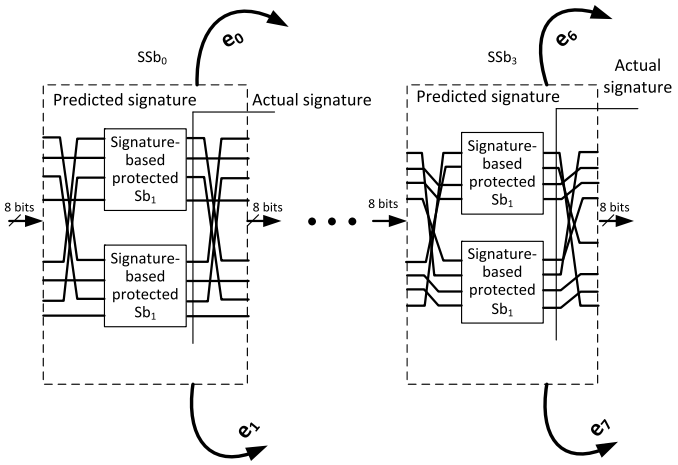

Fig. 1. Derivation of the error indication flags for the S-boxes in Midori128.

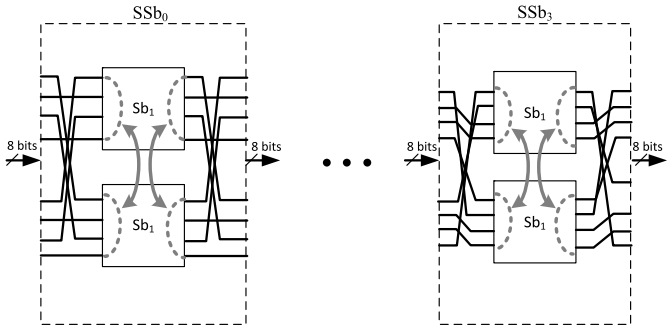

Fig. 2. Proposed RESI scheme for Midori128.

Furthermore, for Sb_1, $a' = a\bar{c} \vee a\bar{b} \vee \bar{a}\bar{d}b$, $b' = a\bar{d}\bar{c} \vee bc \vee \bar{d}b \vee \bar{a}\bar{d}c$, $c' = cd \vee \bar{b}ad \vee \bar{a}b$, and $d' = c\bar{a} \vee c\bar{b} \vee \bar{b}\bar{d}$, where $\vee$ represents an OR gate.

We would like to emphasize that other possible signatures can also be utilized. Here are two examples for parity-based and interleaved parity-based approaches.

The following equations show the predicted parity formulations for the two S-boxes Sb_0 and Sb_1, respectively. We have denoted the predicted parities in these formulations by a *"hat"* sign, i.e., $\hat{P}$, $\hat{P}_{Sb_0} = \bar{d}ac \vee b\bar{a}\bar{c} \vee \bar{b}\bar{d}(c \vee a) \vee db(\bar{a} \vee \bar{c})$ and $\hat{P}_{Sb_1} = \bar{b}(acd \vee \bar{c}\bar{d} \vee \bar{a}\bar{d}) \vee \bar{a}(bd \vee \bar{d}\bar{c}) \vee abc\bar{d}$.

For the interleaved parity-based scheme, we have derived the parities for odd bits and even bits in each of the S-boxes. For Sb_0, we have $\hat{P}^{(0)}_{Sb_0} = \bar{b}(\bar{a}c \vee \bar{b}\bar{d}) \vee d(cb \vee a\bar{c})$ and $\hat{P}^{(1)}_{Sb_0} = \bar{a}\bar{c}(b \vee \bar{d}) \vee \bar{b}d(\bar{a}c \vee a\bar{c}) \vee ac(\bar{d} \vee b)$. Furthermore, for Sb_1, we have $\hat{P}^{(0)}_{Sb_1} = bd \vee dc\bar{a} \vee a\bar{b}\bar{d} \vee a\bar{c}\bar{d}$ and $\hat{P}^{(1)}_{Sb_1} = c\bar{b} \vee \bar{a}\bar{c}\bar{d}$.

Different S-boxes are applied in the variants of Midori, for instance, Midori128 applies four different 8-bit S-boxes SSb_i, $0 \leq i \leq 3$. To keep the involution property of S-boxes, each output bit permutation is derived as the inverse of the corresponding input bit permutation. The structure of Midori and the proposed fault diagnosis schemes are presented in Fig. 1. Fig. 1 shows that four 8-bit outputs of these S-boxes are taken of specific permutation order (two of the S-boxes are omitted for the sake of brevity). Through the comparison of actual and predicted parities, we have error indication flags for each Sb_1 in S-boxes of SSb_i, as shown in Fig. 1 (e_0–e_7). Moreover, both aforementioned parity bits such as single parity and interleaved parity bit have been utilized to create error indication flags. Eventually, one can OR the flags to have a final error indication flag that alters of any faults detected in SSb_i.

1) Recomputing With Swapped Inputs: We use the method of recomputing with swapped inputs (RESI), as shown in Fig. 2, for Midori128 (part of the S-box block is shown for the sake of brevity). This method is a subset to the approaches presented in [16]. In this approach, we have swapped the inputs to the S-boxes Sb_1 in each of the four 8-bit S-boxes SSb_i, i.e., the first four inputs are asserted to the second S-box Sb_1 and the next 4-bit inputs go to the first one, as shown in Fig. 2. Then, if the output of each Sb_1 is swapped, it gives the correct results. Finally, we compare the swapped outputs with actual outputs to detect not only transient faults but also permanent faults. It is noted that the order of permutation of inputs for each SSb_i is different and swapping would be specific for each of the 8-bit S-boxes. In the proposed scheme, which is based on recomputations, we do not change the original algorithm; nevertheless, we perform the recomputation for detecting the errors; thus, no change is made in the original Midori computation and the overall structure for the original algorithm is intact. Therefore, algorithmic security is not affected in the proposed method as the datapath would use the output of the original Midori algorithm.

B. Fault Diagnosis of ShuffleCell and KeyAdd

The signature derivation for fault detection in ShuffleCell, such as parity, would be straightforward and can be realized free in hardware due to just rewiring of the elements of 4×4 array state (for instance, parity of inputs is equal to parity of outputs because rewiring does not affect the computation of parities). We need error detection mechanisms for ShuffleCell (an attacker may try to inject fault by violating setup time for these paths); yet, through using signatures, e.g., parity or interleaved parities, the predicted signatures are equal to the actual signatures of the prior transformation, and that reduces the cost for error detection.

The other operation, KeyAdd, consists of modulo-2 addition of the ith n-bit round key RK_i with the state S. In this operation, the signature inputs, i.e., state and round key, are modulo-2 added to derive the signature of output for each round. Suppose the output of KeyAdd is denoted by O and inputs are S and RK_i, $0 \leq i \leq 14$ and $0 \leq i \leq 18$ for Midori64 and Midori128, respectively. Denoting signatures

by "Sig.", and the predicted signatures of the output O, which is the function of two inputs by $\hat{\text{Sig.}}_{(O)}(S, \text{RK}_i)$, we have $\hat{\text{Sig.}}_{(O)}(S, \text{RK}_i) = \text{Sig.}(S) \oplus \text{Sig.}(\text{RK}_i)$.

C. Proposed Design for Fault Detection in MixColumn

Let us denote the input state of MixColumn as S and the output state as S'. Then, we have the following for this operation:

$$S' = M \times S \Longrightarrow \begin{pmatrix} s'_0 & s'_4 & s'_8 & s'_{12} \\ s'_1 & s'_5 & s'_9 & s'_{13} \\ s'_2 & s'_6 & s'_{10} & s'_{14} \\ s'_3 & s'_7 & s'_{11} & s'_{15} \end{pmatrix} = \begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix} \times \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \quad (1)$$

where each element of the input or output state matrix would be 4 bits and 8 bits for Midori64 and Midori128, respectively.

In the two Midori variants, the linear layers consist of the two mentioned operations, ShuffleCell and MixColumn, that are applied over GF(2^4) and GF(2^8) for the 64-bit Midori and 128-bit Midori, respectively. As mentioned for the MixColumn operation, Midori utilizes an involutive binary matrix M, as defined before. For the matrix M, there could be typically three types of 4×4 matrices, i.e., involutive MDS (M_A), noninvolutive MDS (M_B), and involutive almost MDS (M_C) matrices [4]

$$M_A = \begin{pmatrix} 1 & 2 & 6 & 4 \\ 2 & 1 & 4 & 6 \\ 6 & 4 & 1 & 2 \\ 4 & 6 & 2 & 1 \end{pmatrix}$$

$$M_B = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

$$M_C = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Among these matrices, involutive almost MDS (M_C) has been applied more in various lightweight ciphers such as PRINCE due to its efficiency. Furthermore, M_C has low diffusion speed and a small number of active S-boxes in each round and has led to increase in the immunity against linear and nonlinear attacks. In the proposed fault detection schemes, the objective is to evaluate these three matrices to possibly add a new aspect on how efficient these are when fault diagnosis approaches are used.

For this operation, we present three error detection schemes as detailed in the following.

1) Scheme 1 (Column Signatures): In the first scheme, we propose modulo-2 addition of the state elements of each column of the output matrix (S'). The theorem is that the result is equal to modulo-2 addition of the state elements of each column of the input matrix (S). Since the modulo-2 addition of each column of matrix M in all of three types of matrices is equal to "1," fault diagnosis through this approach is efficiently performed for the three matrices. In general, for $0 \leq i \leq 3$, we have $s'_i = m_i s_0 + m_{i+4} s_1 + m_{i+8} s_2 + m_{i+12} s_3$.

Let us modulo-2 add the first column of the state output matrix: $s'_0 + s'_1 + s'_2 + s'_3 = (m_0 + m_1 + m_2 + m_3)s_0 + (m_4 + m_5 + m_6 + m_7)s_1 + (m_8 + m_9 + m_{10} + m_{11})s_2 + (m_{12} + m_{13} + m_{14} + m_{15})s_3$. Moreover, one can derive that each of the coefficients of the input elements is equal to 1, e.g., $m_0 + m_1 + m_2 + m_3$ is equal to "1." For example, in the case of M_A and for Midori64 that consists of 4-bit elements in the states, we have $\{1\}_{16} + \{2\}_{16} + \{6\}_{16} + \{4\}_{16} = \{1\}_{16}$. Consequently, modulo-2 addition of each of the output matrix columns is equal to that of the columns of input matrix, i.e., $s'_0 + s'_1 + s'_2 + s'_3 = s_0 + s_1 + s_2 + s_3$. For both variants of Midori, one can derive four 4-bit (Midori64) or 8-bit (Midori128) signatures, which can eventually be compared with the actual ones to derive the respective error indication flags.

2) Scheme 2 (Low-Overhead Union Signature): The second scheme is through modulo-2 addition of all the elements of the output state (union signature), i.e., $s'_0 + s'_1 + \cdots + s'_{14} + s'_{15} = (m_0 + m_1 + m_2 + m_3)s_0 + (m_4 + m_5 + m_6 + m_7)s_1 + (m_8 + m_9 + m_{10} + m_{11})s_2 + (m_{12} + m_{13} + m_{14} + m_{15})s_3 + (m_0 + m_1 + m_2 + m_3)s_4 + \cdots + (m_{12} + m_{13} + m_{14} + m_{15})s_{15}$. It is derived that each of these coefficients, e.g., $m_0 + m_1 + m_2 + m_3$, is equal to "1" for the aforementioned matrices. As the proof, the binary values of the following hex entries are all equal to "1": $\{1\}_{16} + \{2\}_{16} + \{6\}_{16} + \{4\}_{16} = \{0\}_{16} + \{1\}_{16} + \{1\}_{16} + \{1\}_{16} = \{2\}_{16} + \{1\}_{16} + \{1\}_{16} + \{3\}_{16} = \{1\}_{16}$. Therefore, the modulo-2 addition of all output elements in the state is equal to that of all input elements in the state (as a nibble or a byte for Midori64 and Midori128, respectively), i.e., $\sum_{i=0}^{15} s_i = \sum_{i=0}^{15} s'_i$. In this approach, we have the less number of signatures that leads to lower overhead.

3) Scheme 3 (Interleaved Signatures): The third scheme is through predicting interleaved signatures. We prove that for each of the two random rows of M_C, this is a viable approach, whereas it is not a suitable scheme for the other two matrices presented before. Let us, through an example, detail on this scheme. Let us modulo-2 add two even-row elements of the state output state, i.e., rows 0 and 2, as $s'_0 + s'_2 = (m_0 + m_2)s_0 + (m_4 + m_6)s_1 + (m_8 + m_{10})s_2 + (m_{12} + m_{14})s_3$, and two odd-row elements, i.e., rows 1 and 3, as $s'_1 + s'_3 = (m_1 + m_3)s_0 + (m_5 + m_7)s_1 + (m_9 + m_{11})s_2 + (m_{13} + m_{15})s_3$. The derived results are two 4-bit or 8-bit predicted interleaved signatures in each column. As computed for the matrices before just in the case of M_C, the interleaved signatures for each column in the input and output states are equal. It is interesting that the modulo-2 addition of each of the two random rows of M_C leads to coefficients of "1" in the aforementioned discussions, e.g., the modulo-2 addition of the first and third rows of M_C is equal to "1010," proving $s'_0 + s'_2 = s_0 + s_2$; moreover, the modulo-2 addition of the second and fourth rows is "0101" and thus $s'_1 + s'_3 = s_1 + s_3$. However, the two other matrices, i.e., M_A and M_B, do not have this property.

Midori can utilize three 4×4 MDS matrices for the MixColumn transformation; we have compared these three

different variants to motivate that error detection needs to be considered as a design factor. In other words, the inventors of Midori have investigated the matrices used in MixColumn to reach the best efficiency for Midori, when the structures have similar algorithmic security. Different categories of the MixColumn transformations are designed based on a wide pool of criteria that can be made smaller by considering the error detection criterion during the design. The fact that the proposed error detection scheme through interleaved signatures is merely efficient for M_C and the other two proposed schemes can efficiently be applied to all three types of matrices in MixColumn further motivates the urgency of having error detection as a design factor not an afterthought.

D. Proposed Approach for Key Schedule

As mentioned before, for both variants of Midori, a 128-bit secret key (K) is applied; however, in the case of Midori64, the key is denoted as two 64-bit subkeys K_0 and K_1 and the WK is derived through modulo-2 addition of these 64-bit subkeys. In the key schedule of Midori64, the round key is derived through $\mathrm{RK}_i = K_{(\mathrm{imod2})} \oplus \alpha_i$, where $0 \leq i \leq 14$, while in Midori128, WK $= K$ and $\mathrm{RK}_i = K \oplus \beta_i$, $0 \leq i \leq 18$, in which $\beta_i = \alpha_i$ for $0 \leq i \leq 14$. The constants are in the form of 4×4 binary matrices, which are modulo-2 added to the LSB of the round key nibble in Midori64 and round key byte in Midori128. Therefore, the signature of the round key matrix is either intact (if round constant element is "0") or inverted (if round constant element is "1"): $\mathrm{Sig.}(\hat{\mathrm{RK}}_i) = \mathrm{Sig.}(K) \oplus \mathrm{Sig.}(\beta_i/\alpha_i)$.

For the sake of brevity, we go over $\alpha_0 = \beta_0, \alpha_{14} = \beta_{14}$, and β_{18} and we do not analyze all the matrices for the constant values; nonetheless, similar approaches can be used for them

$$\alpha_0 = \beta_0 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\alpha_{14} = \beta_{14} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\beta_{18} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

We present two examples for the error detection approach of round key operation. First, let us denote the input key by K and the output round key by RK_i. Then, we have the following for this operation in Midori128 with constant β_i, where $0 \leq i \leq 18$, $\mathrm{RK}_i = K \oplus \beta_i$, and rk_j, k_j, and β^j are the matrices entries for these states, and each element of input or output key matrix would be 4 bits and 8 bits for Midori64 and Midori128, respectively.

In the first example, we derive 16 signatures, where each signature is for the round key elements, $rk_j = k_j \oplus \beta^j$, where each k_j is a nibble as $k_j^3 k_j^2 k_j^1 k_j^0$ for Midori64 or a byte as $k_j^7 k_j^6 k_j^5 k_j^4 k_j^3 k_j^2 k_j^1 k_j^0$ for Midori128. One signature

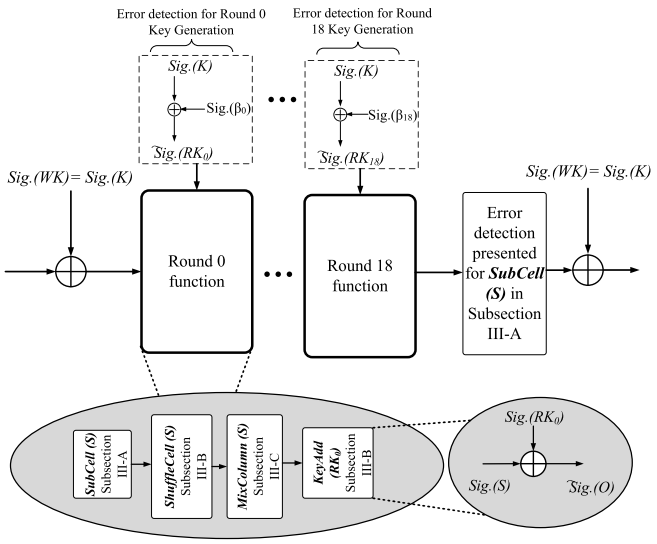

Fig. 3. Proposed error detection architecture for Midori128.

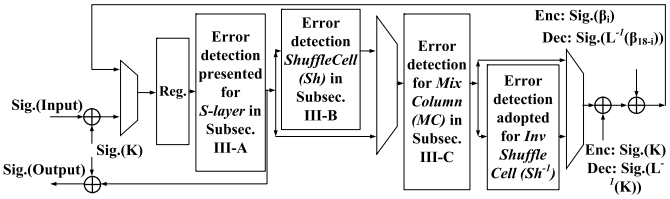

Fig. 4. Error detection architectures for the combined encryption/decryption unit for Midori128.

is derived for each element, for instance, for Midori128, $\mathrm{Sig.}(\hat{rk}_j) = \mathrm{Sig.}(k_j^7 k_j^6 k_j^5 k_j^4 k_j^3 k_j^2 k_j^1 (k_j^0 \oplus \beta^j))$, where $0 \leq j \leq 15$, and we have intact or inverted elements of the round key matrix.

The second scheme is based on the modulo-2 addition of the entire elements in the state matrix to create just one signature for error detection. We have one signature as $\mathrm{Sig.(rk)} = \sum_{j=0}^{15} k_j + \sum_{j=0}^{15} \beta^j$, for instance, for the three matrices $\alpha_0 = \beta_0, \alpha_{14} = \beta_{14}$, and β_{18} in this scheme, and we have the modulo-2 addition of all elements of each matrice as "0," "1," and "0," respectively. Therefore, for RK_0 and RK_{18}, the input signatures are intact, while for RK_{14}, it is inverted.

E. Overall Presented Architecture

This section is finalized by presenting the overall structures of the presented error detection schemes. The mentioned error detection structures of encryption of Midori128, which consists of 20 rounds with a cell size of 8 bits, are depicted in Fig. 3. The encryption function of this variant consists of the round function and key generation in which the last round has just the SubCell operation and the WK is modulo-2 added just in the first and last steps. As seen in Fig. 3, we have shown the respective subsections in which we have proposed the error detection schemes for different operations. The predicted signatures of each operation in round function are illustrated in Fig. 3 as proposed in the aforementioned sections.

Moreover, Midori's encryption and decryption signature-based error detection architectures are also presented in

Fig. 4. As seen in Fig. 4, the signatures of the input and the key ("Sig." Input and "Sig." Key) are derived and processed through a loop-like architecture, asserted to a multiplexer (MUX) and a register (as shown in Fig. 4). The encryption and decryption include similar operations except for the ShuffleCell (Sh) operation in encryption that is replaced with InvShuffleCell (Sh^{-1}) operation for decryption. The error detection scheme for InvShuffleCell (Sh^{-1}) operation can be adopted based on the explanations in Section III-B. Moreover, the key generation process is changed in decryption as shown, i.e., $L^{-1}(K)$ instead of K and the corresponding signatures "Sig." ($L^{-1}(K)$) and "Sig." (L), and, correspondingly, the ith round constant is replaced by $L^{-1}(\beta_{18-i})$ instead of β_i in Fig. 4.

We would like to emphasize that the granularity at which the comparisons between the generated and the predicted signatures are to be made has direct effects on reliability, false alarm resiliency, and overhead of both performance and implementation metrics. Let us go over three cases.

1) One can choose to have the check points for the entire encryption/decryption by deriving a "black-box" signature for these processes. In such a case, we have less overhead at the expense of the possibility of encountering masking for the error indication flags.
2) The false alarm ratios though are the lowest for this case, as we do not use fine-tuned multiple signatures, one may use the error indication flags of the transformations separately, where the formulations presented in this paper are used for each of the check points. In such a scheme, error coverage is higher at the expense of more overhead and the possibility of false alarms.
3) Finally, one may choose to have finer granularity, where each of the 4-bit S-boxes of Midori's 8-bit S-box are checked separately (or, for instance, the columns of the MixColumn transformation are checked individually). At the expense of higher overhead and higher error coverage, such a scheme may lead to higher false alarms ratios.

To finalize this section, let us provide three examples on the usage models of different signatures. Simple parity codes are capable of detecting odd faults, including single stuck-at faults, which are the ideal cases for fault attacks. Nevertheless, their effectiveness could be limited if fault attacks are mounted intelligently to circumvent such protections. Moreover, VLSI defects could result in burst faults whose detection is not possible through such signatures. Burst faults, e.g., adjacent faults, are detected through interleaved parties at the expense of more overhead. A third usage model would be contrasting the signature-based diagnosis approach, which uses linear codes that can (always) detect random errors of small multiplicity (and can never detect some other errors), which is diverse from an architecture based on robust codes that can detect (with probability) any error. These two solutions have two different goals, the first gives reliability and the second gives hardware security (against fault attacks).

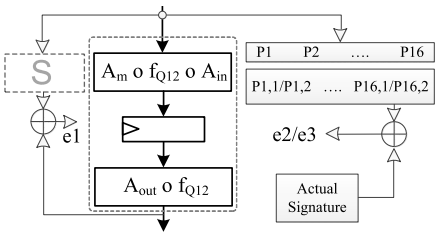

Fig. 5. Error detection architectures for TI of the Midori.

F. Error Detection for Threshold Implementation of Midori

For countering power analysis attacks, viable approaches are used, e.g., masking (prone to glitches in hardware) and its viable variant, threshold implementation (TI) for first order [17], [18] (and its extension to higher order attacks [19]). Protecting lightweight block ciphers against such attacks incurs performance and complexity overheads, which are undesirable.

The S-box of Midori is affine equivalent to the cubic class C_{266}: S has been previously introduced and can be designed through the following approach [5]: $A_{\text{out}} \circ Q_{12} \circ A_m \circ Q_{12} \circ A_{\text{in}}$, where A_{out}: $\{0A1B82934E5FC6D7\}_{16}$, A_m: $\{84B70C3F95A61D2E\}_{16}$, A_{in}: $\{8A02DF57CE469B13\}_{16}$, and Q_{12}: $\{0123456789CDEFAB\}_{16}$. We note that as defined in [5], f_{Q12} is used in three instances, preceded by A_{in} and followed by A_m. After a set of registers, three instances of f_{Q12} are followed by A_{out}. In what follows, we present two schemes that are depicted in Fig. 5 as well.

1) Scheme 1: In this scheme, knowing that the entire $A_{\text{out}} \circ Q_{12} \circ A_m \circ Q_{12} \circ A_{\text{in}}$ produces the S-box, one can use an S-box in parallel with TI to detect errors through comparison. In comparison with the naive approach of duplicating the entire $A_{\text{out}} \circ Q_{12} \circ A_m \circ Q_{12} \circ A_{\text{in}}$, here, we get high error coverage, no false alarms, and low overhead.

2) Scheme 2: Through deriving signatures (for instance, parity and interleaved parity), one can perform error detection with lower overhead, at the expense of lower error coverage. The error indication flags are derived through comparing the predicted signatures of S (stored in the memory, for instance, or derived through logic gates) and the actual signatures, which are functions of the output of the entire $A_{\text{out}} \circ Q_{12} \circ A_m \circ Q_{12} \circ A_{\text{in}}$.

G. Proposed Approaches in the Presence of Biased Fault Attacks

A subset of fault attacks, differential fault intensity analysis (see [20]–[22]), combines the idea of differential power analysis with fault injection principles to obtain biased fault models, whose merit is that same fault in both the original and redundant computations can be injected, more practically, where not all faults occur with equal probability. Practically, the attacker is interested in using as few faults as possible (preferably single faults with different intensities) to reduce the effort. Previous works argue that the single-bit (more likely in low fault intensity), two-bit, three-bit, and four-bit (more likely in higher intensities) biased fault models can be used to simulate variation of fault intensity. In addition,

fault categories presented in [22] and [23] include single-bit upset (SBU), single-byte double-bit upset, single-byte triple-bit upset (SBTBU), single-byte quadruple-bit upset (SBQBU), other single byte (OSB) faults, and multiple byte (MB) faults, the former four corresponding to single/two/three/four-bit models.

The proposed approaches in this paper based on error detecting codes, column signatures, and RESI are able to thwart a number of such fault models. Specifically, SBUs and SBTBUs are detected fully through the approaches based on error detecting codes and column signatures, using parities. Moreover, through interleaved parities, in addition to burst faults, some SBTBUs, SBQBUs, OSBs, and MBs are detected. Error detecting codes and column signatures (parities) can also detect OSBs and MBs, detailed in the next section through simulations. Furthermore, RESI can detect errors with a relatively high error coverage, presented in the next section.

Multibyte faults cannot be used practically for attacking time redundancy countermeasure implementations, e.g., RESI, and single-byte fault models are the only viable option for the attackers [23]. We note that, however, the presented countermeasures based on RESI could fail to detect the occurrence of a fault as long as the adversary could inject the same fault in both the original and redundant computations (biased fault model makes it easier). The proposed RESI architecture (see Fig. 2) can be used in conjunction with encoding schemes that nullify the effect of the bias in the fault model by fault space transformation (if two equivalent faults f_0 and f_1 are injected into the output registers, we use a mapping that transforms the fault space), thwarting both these attack schemes, similar to the schemes used in [23].

IV. Error Simulation and FPGA Benchmark

The error coverage assessment and overhead benchmark of the error detection structures are presented in this section.

A. Error Coverage

Most internal faults are modeled by transient random faults. By relying on simulations, error coverage through multiple stuck-at fault injections is evaluated for Midori. The results of our performed simulations are for both transient faults and permanent internal faults. We consider both single and multiple stuck-at faults because these model both natural and malicious faults (the reason is that natural faults are usually multiple, and although single stuck-at faults are the ideal cases for the attackers, due to technological constraints, multiple faults occur in reality). The single-bit errors in the nibbles (Midori64) or bytes (Midori128) occurring at the outputs of the linear or nonlinear components of Midori are detected by the presented signature-based error detection approaches. The error coverage of the proposed schemes for these single stuck-at faults is 100%; therefore, no simulation is required for these cases. The analytic reason is that odd faults are detected using the proposed approaches, as the inherent property of the signatures used, and single stuck-at faults are their subset. We anticipate that the proposed approaches would not detect all of the potential fault attacks, but the proposed architectures would make it more difficult to mount, e.g., analytically, more than 99.99% of the faults injected are detected. The reason is that multiple signatures are used for different subparts of the architectures that alarm the errors (the error coverage of interleaved parity, for instance, considering different transformations and rounds is $100 \times (1 - (0.5)^{2m})\%$, where m is the total number of use cases).

TABLE II
Error Coverage of the Proposed Schemes for Midori128

Type of faults	Injected faults	Detected faults	Error coverage
Stuck-at zero	10,000	9,910	99.10%
	100,000	99,890	99.89%
Stuck-at one	10,000	9,909	99.09%
	100,000	99,782	99.78%

The results of our performed simulations are for both transient faults and permanent internal faults. Midori128 encryption has been considered as reference for the fault injection simulations. Through applying two fault injection experiments, i.e., 10 000 and 100 000 faults (diverse in terms of the type of the fault, its location, and its count), error indication flags are monitored and the detected errors are counted for the encryption operation. The results of the performed simulations in Table II show that as the number of injection points is increased, higher error coverage is obtained. We would like to note two points on our experiment methods.

1) We have used linear-feedback shift registers (LFSRs) to inject the faults, when random multiple faults are required, where the location, the type, and the number of faults are chosen by LFSRs with maximum tap polynomials.
2) Starting with injecting 10 000 faults using such a method and through LFSRs for different assertions of the inputs, we have increased the number of injections to get closer to more realistic error coverages. This has been done up to 100 000 injections using the aforementioned method, and as seen in Table II, the change in the error coverage, although slight, hints to what we realistically would get through an exhaustive search.

Midori's S-boxes and MixColumn operations consume much of the area and power consumption of the block cipher (compared with ShuffleCell and KeyAdd). Therefore, these are the prominent operations for protecting against injected faults. We have presented in this paper three schemes for each of these operations, i.e., parity/interleaved parity/swapping the inputs for the S-boxes, and element-wise, column-wise, and matrix-wise signatures for MixColumn. Alleviating the error coverage of Midori can be performed through using the schemes with higher error coverage, i.e., swapping the inputs for the S-boxes combined with the element-wise error detection approach for MixColumn, at the expense of higher overhead incurred. Such an improvement would be a compromise between the reliability requirements and overhead tolerance.

B. Implementation Results

The results of the FPGA overhead assessments of our proposed schemes are presented in this section. Through this

TABLE III
FPGA IMPLEMENTATION RESULTS FOR THE ORIGINAL MIDORI128 ENCRYPTION AND ITS PROPOSED ERROR DETECTION SCHEME ON VIRTEX-7 FPGA FOR xc7vx330t

Architecture	Area (occupied slices)	Delay (ns) / Frequency (MHz)	Power (mW)	Throughput (Gbps)	Efficiency (Mbps/slices)
Midori128 (LUT-based)	155	2.70 (370.37 MHz)	340	47.41	305.9
Signature-based error detection for LUTs	161 (3.9%)	2.81 (4.07%) (355.87 MHz)	367 (7.9%)	45.55 (3.9%)	282.9 (7.5%)
Midori128 (logic-based)	157	2.79 (358.42 MHz)	349	45.88	292.2
Signature-based error detection for logic-based	171 (8.9%)	3.01 (7.88%) (332.22 MHz)	396 (13.5%)	42.52 (7.3%)	248.6 (14.9%)

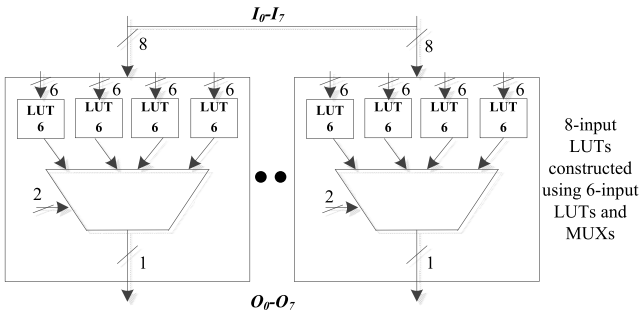

Fig. 6. Structure of Midori128 eight-input S-box implemented using Virtex-7 six-input LUTs.

benchmarking, the overheads (degradation) are derived for various metrics of the presented schemes. The ISE version 14.7 and Virtex-7 FPGA family (device: xc7vx330t) have been utilized for the FPGA implementations. VHDL has been used as the design entry for the original and the error detection structures. As seen in Fig. 6, we have also shown the structure of the eight-input S-box of Midori128 using six-input LUTs (six LUTs in Fig. 6) and MUXs, leading to 32 of 6 LUTs.

We present the results for two of the proposed schemes in this paper tabulated in Table III. Based on the simulation results in this section, these overheads are added for the error coverage of very close to 100%. The proposed fault diagnosis approaches provide high error coverage at the expense of acceptable overheads, making the hardware architectures of Midori more reliable. We would like to emphasize that it is expected to have similar overheads for other FPGA families and also the designs on application-specific integrated circuit.

We finalize this section by investigating differential fault attacks on a similar lightweight block cipher to Midori, PRESENT, to study potential fault attacks on this cipher as well. It is noted that many prior works have been done on fault attacks of standardized ciphers such as the AES and the DES [25]–[27]. In such attacks on lightweight ciphers like PRESENT, the attacker compares the correct ciphertext and the faulty one (caused by faulty operation) to obtain the secret key [24]. Mounting such attacks can be based on single-bit faults, single-nibble faults, or multiple-nibble faults, which are adopted into the key schedule algorithm during the encryption operation. According to such attacks on PRESENT with the SPN structure (or even for the functions in CLEFIA which is a Feistel network), potential attacks on Midori variants can be mounted (which have round-based architectures similar to PRESENT), to recover the subkeys in the rounds [24], [28], [29]. In differential fault attacks, the attackers try to inject the faulty nibble into a specific round in order to discover the secret key by examining a group of correct and faulty ciphertexts. After that, the attack is based on solving a number of fault equations that are obtained through the propagation of the injected faults through the remainder of the encryption structure. We anticipate that differential fault attacks, similarly, can be mounted on Midori. Finally, we would like to note that Midori and other block ciphers include controller architectures when practically implemented. Such control blocks are also susceptible to natural and malicious faults (see [30]) and the remedies proposed based on programmable state flip-flops.

V. CONCLUSION

In this paper, we have presented, for the first time, the fault diagnosis approaches for the energy-efficient lightweight block cipher Midori. For the S-boxes within Midori, we have derived and implemented both LUT-based and logic gate variants, and proposed fault diagnosis schemes that can be tailored based on the reliability and overheard objectives. The MixColumn operation has been examined to achieve a number of schemes, and the selection of the matrices within has been carefully done to have low-overhead detection approaches. Through FPGA implementations using the Xilinx Virtex-7 family, it has been shown that the overheads of the proposed architectures are acceptable for resource-constrained applications. More reliable architectures for Midori are achieved through the proposed detection schemes and they can be tailored based on the objectives in terms of reliability and overhead tolerance.

ACKNOWLEDGMENT

This work was performed under the U.S. federal agency award 60NANB16D245 granted from the U.S. Department of Commerce, National Institute of Standards and Technology.

REFERENCES

[1] M. Mozaffari-Kermani, M. Zhang, A. Raghunathan, and N. K. Jha, "Emerging frontiers in embedded security," in *Proc. Conf. VLSI Design*, Jan. 2013, pp. 203–208.
[2] T. Eisenbarth, S. Kumar, C. Paar, and L. Uhsadel, "A survey of lightweight-cryptography implementations," *IEEE Des. Test Comput.*, vol. 24, no. 6, pp. 522–533, Jun. 2007.
[3] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *Proc. EUROCRYPT*, May 2011, pp. 69–88.
[4] S. Banik *et al.*, "Midori: A block cipher for low energy (extended version)," in *Proc. Cryptol. ePrint Arch.*, 2015, pp. 411–436.
[5] A. Moradi and T. Schneider. (2016). *Side-Channel Analysis Protection and Low-Latency in Action-Case Study of PRINCE and Midori*. [Online]. Available: https://eprint.iacr.org/2016/481.pdf

[6] M. Mozaffari Kermani, R. Azarderakhsh, C.-Y. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 995–1003, May 2014.
[7] S. Ali, X. Guo, R. Karri, and D. Mukhopadhyay, "Fault attacks on AES and their countermeasures," in *Secure System Design Trustable Computing*. Springer, 2016, pp. 163–208.
[8] X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1595–1608, Oct. 2013.
[9] M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie, "Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC," *ACM Trans. Embedded Comput. Syst.*, to be published.
[10] 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.
[11] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptograph. Eng.*, vol. 5, no. 3, pp. 153–169, 2015.
[12] R. Karri, G. Kuznetsov, and M. Goessel, "Parity-based concurrent error detection of substitution-permutation network block ciphers," in *Proc. Cryptograph. Hardw. Embedded Syst.*, 2003, pp. 113–124.
[13] M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2804–2812, Dec. 2015.
[14] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, vol. 17. New York, NY, USA: Springer, 2000.
[15] G. Canivet, P. Maistri, R. Leveugle, J. Clédière, F. Valette, and M. Renaudin, "Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA," *J. Cryptol.*, vol. 24, no. 2, pp. 247–268, 2011.
[16] X. Guo and R. Karri, "Invariance-based concurrent error detection for advanced encryption standard," in *Proc. DAC*, 2012, pp. 573–578.
[17] S. Nikova, V. Rijmen, and M. Schläer, "Secure hardware implementation of nonlinear functions in the presence of glitches," *J. Cryptol.*, vol. 24, no. 2, pp. 292–321, 2011.
[18] E. Boss, V. Grosso, T. Güneysu, G. Leander, A. Moradi, and T. Schneider, "Strong 8-bit Sboxes with efficient masking in hardware," in *Proc. LNCS Cryptograph. Hardw. Embedded Syst. (CHES)*, 2016, pp. 171–193.
[19] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *Proc. ASIACRYPT*, 2014, pp. 326–343.
[20] N. F. Ghalaty, B. Yuce, and P. Schaumont, "Analyzing the efficiency of biased-fault based attacks," *IEEE Embedded Syst. Lett.*, vol. 8, no. 2, pp. 33–36, Jun. 2016.
[21] N. F. Ghalaty, B. Yuce, M. Taha, and P. Schaumont, "Differential fault intensity analysis," in *Proc. FDTC*, 2014, pp. 49–58.
[22] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for AES," in *Proc. COSADE*, 2015, pp. 189–203.
[23] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. H. Nguyen. (2015). *Using State Space Encoding to Counter Biased Fault Attacks on AES Countermeasures*. [Online]. Available: https://eprint.iacr.org/2015/806.pdf
[24] G. Wang and S. Wang, "Differential fault analysis on PRESENT key schedule," in *Proc. Comput. Intell. Secur.*, Dec. 2010, pp. 362–366.
[25] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proc. CRYPTO*, Dec. 1997, pp. 513–525.
[26] G. Piret and J.-J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *Proc. CHES*, 2003, pp. 77–88.
[27] 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.
[28] S. S. Ali and D. Mukhopadhyay, "Improved differential fault analysis of CLEFIA," in *Proc. FDTC*, Aug. 2013, pp. 60–70.
[29] A. Kiss, J. Krämer, and A. Stüber, "On the optimality of differential fault analyses on CLEFIA," in *Proc. MACIS*, 2015, pp. 181–196.
[30] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor, "AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs," in *Proc. Design Autom. Conf. (DAC)*, 2016, Art. no. 89.

Anita Aghaie (S'15) received the B.Sc. degree in electrical and computer engineering from the Isfahan University of Technology, Isfahan, Iran, in 2013. Currently, she is pursuing the graduate degree with the Rochester Institute of Technology, Rochester, NY, USA, under the supervision of Prof. M. Mozaffari-Kermani and under the co-supervision of Prof. R. Azarderakhsh.

Her current research interests include cryptographic engineering, fault diagnosis and tolerance in digital systems, ASIC/FPGA design, and computer arithmetic.

Mehran Mozaffari Kermani (S'00–M'11–SM'16) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively.

Dr. Mozaffari-Kermani is currently the Publications Chair for the HOST conference. He joined the Advanced Micro Devices, Markham, Ontario, Canada, as a Senior ASIC/Layout Designer, integrating sophisticated security/cryptographic capabilities into accelerated processing. In 2012, he joined the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, as an NSERC Post-Doctoral Research Fellow.

Dr. Kermani has been a TPC Member for a number of conferences including DAC, DATE, RFIDSec, LightSec, WAIFI, FDTC, and DFT. He was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011 and the Texas Instruments Faculty Award (Douglas Harvey) in 2014. Currently, he is an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, *ACM Transactions on Embedded Computing Systems*, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I. He is a Guest Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the Special Issue on Emerging Embedded and Cyber Physical System Security Challenges and Innovations (2016 and 2017). He was the lead Guest Editor of the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING for special issues on security.

Reza Azarderakhsh (M'12) received the B.Sc. degree in electrical and electronic engineering and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2011.

He joined the Department of Electrical and Computer Engineering, University of Western Ontario, as a Limited Duties Instructor, in 2011. He has been an NSERC Post-Doctoral Research Fellow with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON. His current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography.

Dr. Azarderakhsh was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2012. Currently, he is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I. He is a Guest Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the Special Issue on Emerging Embedded and Cyber Physical System Security Challenges and Innovations (2016 and 2017). He is also a Guest Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS for the Special Issue on Emerging Security Trends for Biomedical Computations, Devices, and Infrastructures (2015 and 2016).