# Reliable Hardware Architectures for Cryptographic Block Ciphers LED and HIGHT

Srivatsan Subramanian, Mehran Mozaffari-Kermani, *Senior Member, IEEE*, Reza Azarderakhsh, *Member, IEEE*, and Mehrdad Nojoumian

***Abstract*—Cryptographic architectures provide different security properties to sensitive usage models. However, unless reliability of architectures is guaranteed, such security properties can be undermined through natural or malicious faults. In this paper, two underlying block ciphers which can be used in authenticated encryption algorithms are considered, i.e., light encryption device and high security and lightweight block ciphers. The former is of the Advanced Encryption Standard type and has been considered area-efficient, while the latter constitutes a Feistel network structure and is suitable for low-complexity and low-power embedded security applications. In this paper, we propose efficient error detection architectures including variants of recomputing with encoded operands and signature-based schemes to detect both transient and permanent faults. Authenticated encryption is applied in cryptography to provide confidentiality, integrity, and authenticity simultaneously to the message sent in a communication channel. In this paper, we show that the proposed schemes are applicable to the case study of simple lightweight CFB for providing authenticated encryption with associated data. The error simulations are performed using Xilinx Integrated Synthesis Environment tool and the results are benchmarked for the Xilinx FPGA family Virtex-7 to assess the reliability capability and efficiency of the proposed architectures.**

***Index Terms*—Authenticated encryption, high security and lightweight (HIGHT), light encryption device (LED), reliability.**

## I. Introduction

TO PROVIDE different security properties efficiently, lightweight cryptographic implementations on different hardware platforms have been emerged due to the advancement of constrained devices. These nodes require low-complexity implementations over small chip area and consume low amount of energy. Nevertheless, the Advanced Encryption Standard (AES), the current symmetric-key cryptography standard, may not achieve such tight necessities in terms of performance and implementation metrics. Thus, lightweight security mechanisms through low-complexity implementations of cryptographic algorithms are needed. We note that there have been constant, prominent efforts to realize the AES lightweight, an example for which is a 128-bit AES that was developed over an area of 2400 gate equivalent [1]. It is noted that the AES architecture in [1] has been toward considerable area reductions; nonetheless, it is still considered a burden for resource-constrained applications, such as radio-frequency identification tags, nano-sensor nodes, and applications such as implantable and wearable medical devices. Furthermore, the inability of the AES to adapt to the varying level of security needed by different devices might be inefficient in case lower number of bits need to be protected.

In recent years and based on the above motivation, a number of lightweight block ciphers have been proposed, e.g., KATAN and KTANTAN [2], PICCOLO [3], and PRESENT [4], SEA [5], light encryption device (LED) [6], Simon and Speck [7]–[10], Midori [11], high security and lightweight (HIGHT) [12], etc. Based on such ciphers, an open competition for a new authenticated encryption algorithm [13] has been initiated and will identify a portfolio of authenticated ciphers that offer advantages over AES-Galois counter mode and are suitable for widespread adoption. These algorithms, e.g., simple lightweight CFB (SILC) [14] that employs authenticated encryption with associated data, use encryption/decryption blocks as underlying structures in which the aforementioned block ciphers can be used. Authenticated encryption provides authenticity and privacy to the data by first converting the plaintext to ciphertext and an authentication tag, message authentication code (MAC).

It is imperative to note that although cryptographic algorithms, e.g., authenticated encryption which preserves authenticity and confidentiality of the message sent by the sender, provide different security mechanisms, natural and malicious faults can undermine such purpose. Let us go over different fault models we have considered in this paper. We consider both single and multiple stuck-at faults because both are relevant with respect to intentional and natural faults, i.e., fault attackers prefer to ideally be able to inject single faults but, in reality, due to lack of technological advancements, multiple faults might occur, whose protection mechanisms are required.

Manuscript received June 7, 2016; revised September 30, 2016 and December 8, 2016; accepted January 15, 2017. Date of publication January 31, 2017; date of current version September 14, 2017. The work of M. Mozaffari-Kermani and R. Azarderakhsh were supported by the U.S. Department of Commerce, National Institute of Standards and Technology through the U.S. Federal Agency Award under Grant 60NANB16D245. This paper was recommended by Associate Editor S. Hu.

S. Subramanian and M. Mozaffari-Kermani are with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: sxs2476@rit.edu; m.mozaffari@rit.edu).

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

M. Nojoumian is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: mnojoumian@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/TCAD.2017.2661811

0278-0070 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Moreover, natural faults can be of single nature, e.g., single event upsets or multiple defects. Furthermore, we consider both transient and permanent faults. The attackers typically inject transient faults to gain as much information as they desire without breaking the cryptosystems; however, natural defects could be of permanent nature; thus, we consider them as well.

Various types of fault injection mechanisms, such as temperature attacks, optical attacks, electromagnetic fault injection, and the respective countermeasures to such attacks are presented to date. Concurrent error detection (CED) techniques have been widely used to architect reliable hardware for the cryptographic algorithms [15]–[23] (including a number of schemes, e.g., hardware/information/time/hybrid redundancy). Hardware redundancy makes use of extra hardware to process the same input twice to match the two outputs. Information redundancy schemes have a number of variants, e.g., robust codes [24]. Time redundancy technique has a number of schemes, e.g., recomputing with shifted operands [25], [26], recomputing with rotated operands (RERO) [27], and recomputing with permuted operands [28]. The hybrid redundancy scheme is given in [29]–[31], where different improvements in the architecture have been proposed.

In this paper, we consider two block ciphers which can be used as part of SILC, i.e., LED [6], an AES-based block cipher and HIGHT [12]. LED has 64-bit block length and uses 64-bit key length and reuses the *S*-box of PRESENT block cipher [4]. HIGHT is of generalized Feistel type network and has 64-bit block length and 128-bit key length. HIGHT is suitable for embedded CPUs that are used in the nano-sensor network systems.

In this paper, we propose error detection approaches for block ciphers LED and HIGHT, considering the reliability and performance metrics objectives. Signature-based approaches are used in conjunction with the proposed error detection schemes based on recomputing with encoded operands to achieve high efficiency, while maintaining high error coverage. The organization of this paper is as follows. In Section II, preliminaries are provided. Section III presents our proposed error detection scheme. In Section IV, through fault-injection simulations, we evaluate the error coverage capabilities of the proposed scheme. Moreover, in this section, we benchmark the overheads of the proposed scheme on Xilinx FPGA families. Finally, conclusions are made in Section V.

## II. Preliminaries

In this section, we briefly explain the block ciphers LED and HIGHT. Then, in the next section, the proposed error detection approaches are presented.

### A. LED Block Cipher

LED is a 64-bit block cipher. It is of the AES type and is a substitution permutation network (SPN). LED has a nonce length $l_N$ of 8 bytes and tag length $\tau$. It reuses *S*-box of PRESENT block cipher and has a 64-bit key (LED block cipher can be used as one of the underlying block ciphers $E_K$ for SILC, as a usage model). The input plaintext which is of 64-bit length is arranged in a $4 \times 4$ array matrix called cipher *state* matrix. The cipher *state* matrix of the LED block cipher along with the key matrix are governed by an irreducible polynomial in GF($2^4$). The implemented LED block cipher uses a 64-bit key. Both the cipher *state* matrix and the key are arranged in $4\times4$ matrix in the form of 16 four-bit nibbles. Each entity in the cipher *state* matrix and the key matrix is of 4-bit length.

The first operation is the addition of round key denoted by addRoundKey(*state*, $K_i$). This operation is performed on cipher *state* matrix and key matrix $K_i$. LED block cipher has a second operation called *step* function, that is responsible for providing enhanced security. This operation comprises of four iterative rounds for encryption in a sequential manner. In LED, each round consists of sequential set of four operations. For the 64-bit key array matrix, addRoundKey and *step* operations are repeated for eight times. The LED cipher has 32 rounds of iteration for encrypting the plaintext. The sequence of operations that are carried out on the output of the first step are AddConstants, SubCells, ShiftRow, and MixColumnsSerial, more details are presented in the next section.

### B. HIGHT Block Cipher

HIGHT is a 64-bit block cipher with 128-bit key length proposed. HIGHT is a variant of Feistel network and it has 32 iterative rounds to complete the encryption process. It has 64-bit plaintext and 128-bit master key as its inputs and a 64-bit ciphertext as its output. It has been claimed that the hardware implementation of HIGHT is more efficient than the AES by justifying that, it had consumed 3048 gates in 0.25 $\mu$m technology.

The plaintext input is represented by $P = P_7||\ldots P_1||P_0$, where $P_0 \ldots P_7$ are each of eight bits length, and $'||'$ denotes concatenation operation and the plaintext *P* is of 64 bits length. For each round $i = 0, \ldots, 32$, the 64-bit intermediate values are represented by $X_i = X_{i,7}||\ldots X_{i,1}||X_{i,0}$. The 64-bit ciphertext output is represented as $C = C_7||\ldots||C_1||C_0$ and the master key which is of 128 bits length is denoted as $\mathrm{MK} = \mathrm{MK}_{15}||\ldots||\mathrm{MK}_0$, more details are presented in the next section when we go over the proposed error detection schemes.

## III. Proposed Error Detection Approaches

In this section, the error detection schemes used for detecting the transient and permanent faults in the LED and HIGHT block ciphers are presented.

### A. Motivations

In what follows, we present the motivations in presenting the approaches for error detection of LED and HIGHT block ciphers. Then, in the next sections, the proposed methods are presented. The CED techniques have been widely used to architect reliable hardware for the cryptographic algorithms [15]–[23] (including, diverse schemes, e.g., hardware/information/time/hybrid redundancy). Let us review and analyze the existing fault diagnosis schemes here. The use of variants of parity for error detection is effective;

nevertheless, one needs to utilize/tailor such approaches so that they are flexibly used, depending on the error detection capability requirements and overhead tolerance (we have proposed such flexibility when applicable in this paper). Nonuniform error detection due to using parity can be alleviated using robust codes at the expense of relatively higher overhead. If the high area/power overhead is the burden to the usage model, one can alleviate that through the time redundancy approaches, deteriorating the performance metrics. We have considered in this paper variants of time redundancy mechanisms which can detect both permanent and transient faults.

The high overhead of hardware redundancy (area/delay overhead of duplication is roughly 100%), the high delay overhead and inability to detect permanent faults of conventional time redundancy (which is usually the case for hardware failures), the need for flexibility in terms of overhead and reliability compromise for information redundancy, and the need for appropriate choice for the specific techniques within hybrid redundancy pool have been motivations for the proposed work.

This is the first work presenting efficient error detection approaches for LED and HIGHT. We have closely considered two main criteria in choosing such approaches.

1) Applicability of the approaches to the specific algorithms needs to be considered closely; for instance, while the FPGA realization of the *S*-boxes through look-up tables is efficient (which also affects the error detection, as presented in this paper for LED), one may choose to realize them on ASIC through logic-gate approaches (which needs realization of signatures as logic gates instead of storing them in memories).
2) The level of granularity of the check points can be dynamically chosen, which is dependent on the overhead tolerance and the reliability requirements for such ciphers.

In proposing the error detection techniques here, we have considered both of these. The merit of the proposed approaches is as follows.

1) The proposed algorithms are oblivious of the implementation platform (unlike some previous works which use specific resources of platforms).
2) They can be tailored based on the reliability/overhead compromise, e.g., the scheme for MixColumnsSerial of LED.
3) They are flexible in terms of overhead of metrics, and the reliability goals (alternate approaches are proposed to ensure such requirements are achieved, e.g., the recomputing with encoded operands scheme of HIGHT).

## B. Error Detection for LED Block Cipher

The overall flow and the top view of the error detection for the LED block cipher is presented here, followed by the details on the approaches. We have chosen using signature-based schemes for LED, especially because of the structures of SubCells and MixColumnSerial, for which signature-based schemes provide not only flexibility in error detection but also achieve high error coverage. The overall error detection architecture is composed of the predicted/actual signatures and their comparisons to derive the error indication flags (the derived predicted signatures of these four operations are compared with the actual ones to get the error indication flags).

*1) AddConstants and ShiftRow:* In AddConstants operation, the round constant matrix is given by the round constant matrix of

$$\begin{bmatrix} 0 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ 1 & (\mathrm{rc}_2||\mathrm{rc}_1||\mathrm{rc}_0) & 0 & 0 \\ 2 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ 3 & (\mathrm{rc}_2||\mathrm{rc}_1||\mathrm{rc}_0) & 0 & 0 \end{bmatrix}.$$

The round constants are given by six round-specific bits, i.e., $\mathrm{rc}_0, \mathrm{rc}_1, \mathrm{rc}_2, \mathrm{rc}_3, \mathrm{rc}_4$, and $\mathrm{rc}_5$. For each round, the value of round constants are made zero and the values of the six bits are updated before the AddConstants function. This round constant matrix is bit-wise XORed with the output of the *state* matrix from the addRoundKey operation and the value of the *state* matrix is updated.

The second column of the round constant matrix depends on the round in which the matrix is used. For instance, in Round 1, the value of the round constants are $\mathrm{rc}_5 = 0$, $\mathrm{rc}_4 = 0, \mathrm{rc}_3 = 0, \mathrm{rc}_2 = 0$, and $\mathrm{rc}_0 = 1$. For the proposed signature-based approach, we can derive the predicted signatures of such a matrix, e.g., the predicted parity of the last two columns are zero and the one for the second column is also zero. The reason for such derivation is that $(\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) \oplus (\mathrm{rc}_2||\mathrm{rc}_1||\mathrm{rc}_0) \oplus (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) \oplus (\mathrm{rc}_2||\mathrm{rc}_1||\mathrm{rc}_0)$ leads to always zero value verifying that the predicted parity is zero. For the first column, modulo-2 adding of the elements to derive the predicted parities also results in zero. Thus, the predicted parity vector for the round constant matrix is $\hat{P} = [0, 0, 0, 0]$ (we use hat notations for predicted signatures). Thus, we can conclude that the derivation of the predicted parity is free in hardware for the AddConstants step.

*a) Interleaved parity for burst faults:* One can also derive the interleaved parities of the AddConstants step to account for burst faults. Burst faults are adjacent faults that can affect the output in case of both malicious and natural faults. Let us denote on the round constant matrix the rows through which we derive two interleaved parities, i.e., even/odd rows as

$$\begin{bmatrix} 0 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ \mathbf{1} & (\mathrm{rc}_\mathbf{2}||\mathrm{rc}_\mathbf{1}||\mathrm{rc}_\mathbf{0}) & \mathbf{0} & \mathbf{0} \\ 2 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ \mathbf{3} & (\mathrm{rc}_\mathbf{2}||\mathrm{rc}_\mathbf{1}||\mathrm{rc}_\mathbf{0}) & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

By modulo-2 adding such rows, we derive the interleaved parity vectors as $\hat{P_1} = \hat{P_2} = [2, 0, 0, 0]$. The reason for such derivation is that for the last three columns, the values are canceled once modulo-2 added and for the first column we have $\{0\}_{16} + \{2\}_{16} = \{1\}_{16} + \{3\}_{16}$.

The ShiftRow operation shifts the rows of the resulting *state* matrix from the SubCells operation with respect to its row number. If the operation is performed on the first row, then the first row is shifted to the left by 1 position and likewise for the second row, the contents of the *state* matrix are shifted by two positions. For the ShiftRow step, derivation of the predicted signatures is straightforward, e.g., parity prediction is free in hardware as rewiring does not change the predictions.

TABLE I
(INTERLEAVED) PREDICTED PARITY OF THE *S*-BOX FOR THE LED BLOCK CIPHER

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\hat{P}_{s[x]}$ | (0) | (0) | (0) | (1) | (0) | (0) | (0) | (1) | (0) | (1) | (0) | (1) | (1) | (1) | (1) | (1) |
| $\hat{P}_{s[x]}$ (*Int.*) | (11) | (00) | (11) | (01) | (11) | (00) | (00) | (10) | (11) | (01) | (00) | (10) | (01) | (10) | (01) | (10) |

*2) SubCells:* The SubCells operation updates the value of the *state* matrix by replacing the contents of the *state* matrix, according to the *S*-Box. For detecting the errors in *S*-Box, we devise a signature-based scheme illustrated through two case studies, i.e., parity prediction and interleaved (Int.) parity prediction as shown in Table I. In this method, the output bits of the *S*-box are XORed (and for the case of interleaved parity, odd and even bits are modulo-2 added separately) and the output of the *S*-box is appended by one bit (two bits for the interleaved case). For example, if the output is $\{C\}_{16}$ which is $\{1100\}_2$, that means the predicted parity of is 0 and the interleaved parity pair is 11 (see Table I).

*3) MixColumnsSerial:* MixColumnsSerial uses a hardware-friendly matrix known as MDS, that is given by (this matrix has been changed from the original construction in [32])

$$M = \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix}.$$

The updated *state* matrix from the previous operation is multiplied with the $M$ matrix and the resulting *state* matrix is updated column-wise.

For presenting our signature-based scheme, let us denote the input and output *state* matrices as

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_a & a_b \\ a_c & a_d & a_e & a_f \end{bmatrix}, R = \begin{bmatrix} r_0 & r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 & r_7 \\ r_8 & r_9 & r_a & r_b \\ r_c & r_d & r_e & r_f \end{bmatrix}.$$

Each entity in the input and output *state* matrices is a four-bit nibble. In the MixColumnsSerial step, each column of the output matrix $R$ is the product of the matrix $M$ and the *state* matrix $A$. As a case study, to compute $r_0$, the first element of the resultant matrix $R$, denoting the bits of the elements of $A$ as $a_{ij}$ for $i$th bit $j$th element, we have (using the irreducible polynomial utilized for reductions, and not presenting the details for the sake of brevity): $r_0 = 4.a_0 + a_4 + 2.a_8 + 2.a_c = \cdots = x^3[a_{2c} + a_{28} + a_{14} + a_{30}] + x^2.[a_{10} + a_{40} + a_{24} + a_{38} + a_{3c}] + x.[a_{10}+a_{20}+a_{34}+a_{18}+a_{48}+a_{1c}+a_{4c}]+1.[a_{20}+a_{44}+a_{1c}+a_{18}]$.

One can derive the formulas for the column signatures of the MixColumnsSerial by modulo-2 adding those for each column, e.g., the first column for $r_0$, $r_4$, $r_8$, and $r_c$, whose details are not presented for the sake of brevity. Adding modulo-2 the first column of matrix $M$, one can derive the following signature: $r_0 \oplus r_4 \oplus r_8 \oplus r_c = (4 \oplus 8 \oplus B \oplus 2).a_0 + (1 \oplus 6 \oplus E \oplus 2).a_4 + (2 \oplus 5 \oplus A \oplus F).a_8 + (2 \oplus 6 \oplus 9 \oplus B).a_c = 5.a_0 + B.a_4 + 2.a_8 + 6.a_c$.

This can be generalized to other columns and thus we have the followings for the second to fourth columns: $r_1 \oplus r_5 \oplus r_9 \oplus r_d = 5.a_1 + B.a_5 + 2.a_9 + 6.a_d$, $r_2 \oplus r_6 \oplus r_a \oplus r_e = 5.a_2 + B.a_6 + 2.a_a + 6.a_e$, $r_3 \oplus r_7 \oplus r_b \oplus r_f = 5.a_3 + B.a_7 + 2.a_b + 6.a_f$.

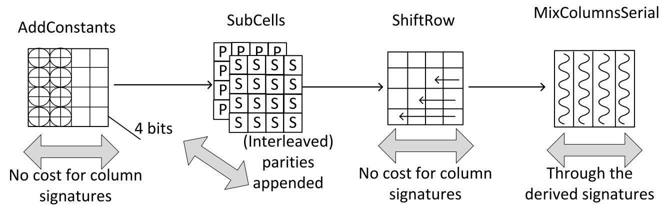
AddConstants
SubCells
ShiftRow
MixColumnsSerial
4 bits
No cost for column signatures
(Interleaved) parities appended
No cost for column signatures
Through the derived signatures

Fig. 1. Proposed fault diagnosis scheme for the LED block cipher.

As seen in Fig. 1, we have denoted the four operations in LED by the fault diagnosis mechanisms for different sub-blocks. As seen in this figure, the AddConstants parity derivation is cost-free in hardware and that of ShiftRow follows the same details. One can use the derivations for SubCells operations in Table I as well as those for MixColumnsSerial. The overall error detection architecture is composed of the predicted and actual signatures and their comparisons to derive the error indication flags. In more details, the derived predicted signatures of these four operations are compared (XORed) with the corresponding, actual ones to get the error indication flags (which can be ORed to derive one flag).

To have a compromise between the reliability objectives and the error detection capability, for the transformations AddConstants, ShiftRow, MixColumnsSerial, and SubCells, one can utilize different signatures, e.g., as seen in this section and Table I, simple signatures (parities) can be utilized with low overhead and limited error detection capability (to odd number of faults). Instead, if reliability objectives are prevalent, interleaved parities can be utilized (to account for burst faults) at the expense of higher overhead.

Let us use the example of AddConstants step to compare different cases for such compromise. The predicted parity vector for the round constant matrix is $\hat{P} = [0, 0, 0, 0]$ whose derivation is free in hardware. However, by modulo-2 adding specific rows, e.g., even/odd rows in

$$\begin{bmatrix} 0 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ \mathbf{1} & (\mathrm{rc}_\mathbf{2}||\mathrm{rc}_\mathbf{1}||\mathrm{rc}_\mathbf{0}) & \mathbf{0} & \mathbf{0} \\ 2 & (\mathrm{rc}_5||\mathrm{rc}_4||\mathrm{rc}_3) & 0 & 0 \\ \mathbf{3} & (\mathrm{rc}_\mathbf{2}||\mathrm{rc}_\mathbf{1}||\mathrm{rc}_\mathbf{0}) & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

we derive the interleaved parity vectors as $\hat{P_1} = \hat{P_2} = [2, 0, 0, 0]$. Such a compromise gives the architects the ability to select the error detection approaches based on the overhead tolerance and reliability requirements.

### C. Error Detection for HIGHT Block Cipher

In this section, we first present the error detection schemes for the "initial/final transformations and round function" through two pair two rail checker (TPTRC). Then, an alternate method based on recomputation is presented.

The TPTRC scheme, inherently, is able to detect both transient and permanent faults. Transient single stuck-at faults are among the ideal cases for fault attackers; however, in practice, multiple (and adjacent) faults occur. These are detected using the presented scheme and more details are presented in the next section. 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); is diverse from an architecture based on robust codes which can detect (with probability) any error. We also go over an alternate approach based on recomputation with encoded operands which has lower area complexity and power consumption, at the expense of lower performance. The choice among these two schemes depends on the overhead tolerance for specific applications. The latter scheme, similar to the former one, detects both transient and permanent faults. We note that permanent faults need to be detected as they might occur through very large scale integration defects; however, the attackers are not interested to mount the attacks through such damaging faults.

*1) Initial/Final Transformations and Round Function:* The 64-bit plaintext is given as the input to the initial transformation function. It uses the plaintext and whitening keys $\mathrm{WK}_3, \mathrm{WK}_2, \mathrm{WK}_1, \mathrm{WK}_0$ as its input and generates the output as intermediate values $X_0 = X_{0,0}, \ldots, X_{0,6}, X_{0,7}$ for the first round. The whitening keys $\mathrm{WK}_3, \mathrm{WK}_2, \mathrm{WK}_1$, and $\mathrm{WK}_0$ are fed by the key scheduling algorithm. The initial transformation algorithm consists of XOR operations denoted by $\oplus$ and modular addition represented by $\boxplus$ (note that for decryption, this is replaced by subtraction $\boxminus$).

In the final transformation, the input is the output of the last round $X_{32}$ and the other inputs are the whitening keys $\mathrm{WK}_7, \mathrm{WK}_6, \mathrm{WK}_5, \mathrm{WK}_4$. The output generated by this algorithm is the ciphertext, given by $C$, that is of 64 bits length and it is concatenated and represented as $C_0 || \ldots C_6 || C_7$.

The round function is an iterative process for the 32 rounds and it plays a vital role in providing enhanced security. The operations carried out in the round function are modular addition and XOR operation. The input of the round function is the output of the previous round $X_i$ and the four SubKeys $\mathrm{SK}_{4i+3}, \mathrm{SK}_{4i+2}, \mathrm{SK}_{4i+1}, \mathrm{SK}_{4i}$ generated per each round. The round function algorithm generates the 64-bit output $X_{i+1}$ and uses the auxiliary functions $F_0$ and $F_1$ to compute the output $X_i = X_{i+1,0} || \ldots X_{i+1,6} || X_{i+1,7}$ concatenation of eight bytes each. The auxiliary functions use the operation $x^{<<<1}$, which is a representation of 1-bit left rotation of the 8-bit value $x$. The output of the rotated value is bitwise XORed or modulo-2 added depending on the input. The output of the last round is fed to the input of the final transformation. We note that $F_0(x) = x^{<<<1} \oplus x^{<<<2} \oplus x^{<<<7}$ and $F_1(x) = x^{<<<3} \oplus x^{<<<4} \oplus x^{<<<6}$.

For modular addition operation, we choose self-checking carry select adder because they are fast and have relatively low complexity (for detecting both permanent and transient stuck-at faults). Generally, it is well known that carry select adders contain two ripple carry adders and multiplexers. In addition to that hardware resource, as shown in Fig. 2, this self-checking carry select adder uses XNOR gates and TPTRC

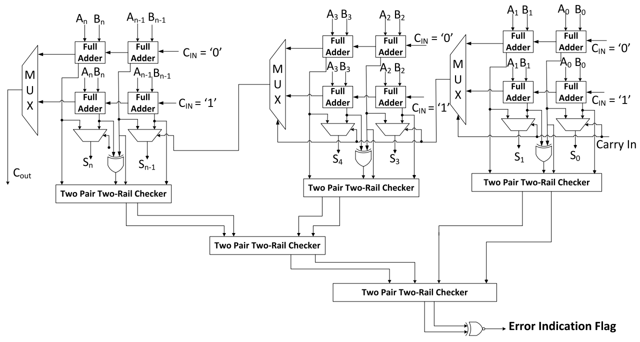

Fig. 2. Error detection in HIGHT through self-checking carry select adder.

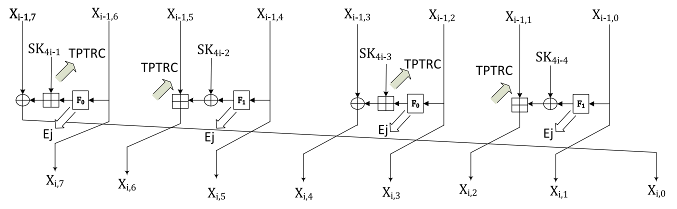

Fig. 3. Error detection for the round function sub-block for HIGHT.

as an additional hardware resource to detect any single stuck at fault [33]. As seen in Fig. 2, the size of the adder is arbitrary up to $n$-bits. Here, for our addition operation, the operands are of 64 bits size and, hence, we use 64-bit adder to perform the operations. The adder consists of cascaded ripple carry adders that can add up to two bits at a time and then the value of the carry out is rippled through a multiplexer. This carry out acts as the value of the actual carry-in to the next set of cascaded ripple carry adders. The carry select adder precomputes the values of the sum bits before knowing the value of the actual carry-in. Once the value of the actual carry-in is known, the appropriate sum bits and carry-out of the carry select adder is given as the output by the carry select adder.

Self-checking multiplexers and TPTRC are used in the proposed architectures. The TPTRC has two pairs of inputs $(x_0, y_0)$ and $(x_1, y_1)$. In the fault-free condition, the input pairs to the TPTRC are complementary to each other, i.e., $x_0 = y_0'$ and $x_1 = y_1'$. If there is no fault, then the output pair of the TPTRC are also complementary to each other, i.e., $t = t_0'$. Finally, the output pair of TPTRC are given to the XNOR gate and if there is a fault, the error indication flag is raised to high. The valid input code words for the TPTRC are 10 and 01. In case of a single stuck at fault in any one of the internal path of the ripple carry adder, the input pairs to the TPTRC will be 11 and 00. This will result in a nonvalid output and error indication flag is raised. Such an adder can be used to detect faults in the HIGHT block cipher.

A variant of $n$-bit model of the self-checking carry select adder based on dual rail encoding has been proposed in [34]. This variant adder includes an additional circuitry of AND gates in addition to the hardware used in Fig. 2. To compute the sum bits for the $n$-bit adder in a dual rail form with valid codewords (10 and 01), AND gates are used in Fig. 3. If $S_{0n}$ is the sum bit computed by the full adder for carry-in of "0" and $S_{1n}$ is the sum bit computed by full adder for carry-in of "1,"

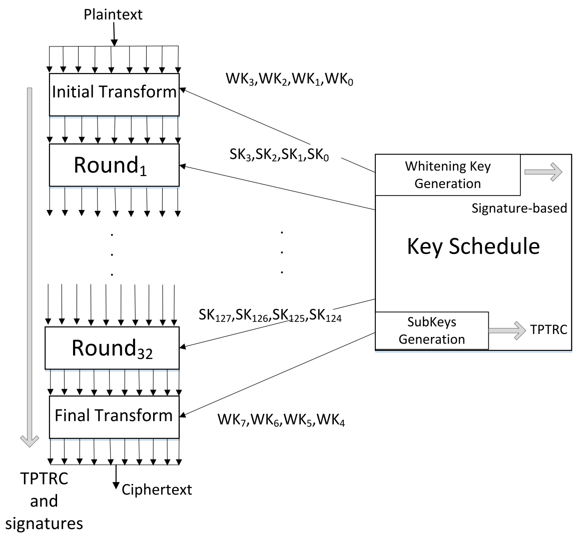

Fig. 4. Error detection for the entire encryption of HIGHT block cipher.

then XNOR operation is performed between sum-bit $S_{0n}$ and all the lower sum-bits. The output of XNOR gate is fed to $x_0$ in the TPTRC input pair $(x_0, y_0)$, and $y_0$ is connected to $S_{1n}$. Similarly, the other input pair $(x_1, y_1)$ to TPTRC is connected in such a way that $x_1$ is connected from the output of XNOR gate and $S_{0n-1}$. The sum bit $S_{0n-1}$ is computed by the full adder for $n-1$ inputs with a carry-in of 0. The sum bit $S_{1n-1}$ is connected to $y_1$ of TPTRC. The input pairs are in dual-rail form, that is, the input pairs will be always complementary to the TPTRC, i.e., $x_0 = y_0'$ and $x_1 = y_1'$.

Finally, as seen in Fig. 3, we have shown the fault diagnosis of the round function of HIGHT. As seen in this figure, for the auxiliary functions $F_0$ and $F_1$ within round function to compute the output $X_i = X_{i+1,0}|| \ldots X_{i+1,6}||X_{i+1,7}$, we use signature-based schemes to derive the error indication flags. In more details, the derived predicted signatures of the operations are compared (XORed) with the corresponding, actual ones to get the error indication flags (which can be ORed to derive one flag). Specifically, for the modular addition, the presented TPTRC approach is utilized (see Fig. 3); furthermore, for the auxiliary functions $F_0$ and $F_1$, we use signature-based schemes. Thus, for the modular addition operation, the proposed scheme based on Fig. 2 is utilized. Finally, Fig. 4 presents the entire error detection approach for the encryption of HIGHT.

*2) Alternate Approach (Recomputing With Encoded Operands):* CED can be performed through recomputing with encoded operands in such a way that the operations are computed twice, one for the normal operands and one for the encoded operands. If an $n$-bit operand is encoded, e.g., rotated left or right by $k$ bits during the recomputation step, no bit is lost and the scheme utilizes the sizes of adders, ALUs, and registers increased only by 1 bit. Such an alternate approach, similar to the previous one, detects both transient and permanent faults in the HIGHT block cipher. Using such a recomputation, one can efficiently detect $k \bmod n$ consecutive logical errors and $k \bmod((n+1)-1)$ in arithmetic operations, where $n$ is the length of arithmetic operations and $k$ bits is the number of bits to be rotated. We have employed such an approach for the HIGHT block cipher. As the operations involved in the HIGHT block cipher are addition mod GF($2^8$) and XOR, such an alternate scheme is efficiently applicable as shown through our error simulations and FPGA implementations.

Let $\phi$ and $\phi'$ be the $n$-bit rotation and "back-rotation" functions, respectively. Let $\theta$ be the input to the arithmetic function $f$, such that $f(\theta)$ is the output of the arithmetic function. The aim is to satisfy $\phi'(f(\phi(\theta)) = f(\theta)$. The operands in the HIGHT block cipher are run twice. For the first run, the original operands are passed and the result is computed and stored in a register. For the second run, rotated operands (right or left cyclic shift) are computed and the result is compared with the first result that is stored in a register. If there is not a match, then error is detected. For addition operations, to ensure the correctness of the carry-in for $k+1$th bit and carry out from $n-1$th bit, we add an extra bit to the most significant bit (*) position using an $n+1$ adder. This bit is always set to 0 (or stuck at zero). Thus, as a result of rotation, the logic $n-1$th bit does not interfere with the logic 0th bit in the rotated position as well as in the normal position. During normal computation, the most significant bit is 0; however, during the recomputation step, the most significant bit is the $i$th bit. Because the most significant bit is always stuck at zero, during recomputation, the carry-out from the $i$th bit is given to the carry-in of the $i+1$th bit. In normal computation, as the value of the carry-out is always 0, it does not change the $i$th bit.

In what follows, we use the case study of SILC to show how the proposed approaches can be utilized; nevertheless, this does not confine the presented schemes. SILC has a fixed block length $n$ and uses a block cipher $E$ such that $\kappa_E \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. We note that HIGHT and LED can both be used for such a construction. SILC comprises of two algorithms for encryption (SILC-$\varepsilon_K$) and decryption (SILC-$D_K$). The encryption SILC-$\varepsilon_K$ algorithm consists of three subroutines HASH, ENC, and PRF. The subroutines are called in a sequential manner, i.e., the output of one subroutine is fed to the input of the successive subroutine. The decryption comprises of the same subroutines as the encryption algorithm, except that an additional comparison between the tag computed in the decryption algorithm and encryption algorithm is performed. The first step in the decryption algorithm is the computation of $V$ (which is the result of hashing the inputs through the key). The next step is the computation of MAC in the decryption algorithm $T^*$(which is the PRF, using the key, of the hash result and the ciphertext). The third step is the comparison of the tags, and if they do not match, the algorithm returns the result $\perp$, meaning that the authentication is failed. The final step in the decryption algorithm is retrieving the original plaintext $M$.

*3) Key Schedule:* The key schedule for HIGHT comprises of two algorithms: 1) whitening key generation and 2) SubKeys generation. The main idea behind the key schedule algorithm is to preserve the master key. Whitening key generation algorithm is a subroutine that generates the eight whitening keys necessary for the initial/final transformations. The SubKey generation subroutine generates 128 SubKeys and supplies four SubKeys per each round. For Round 1, four

SubKeys $SK_3, SK_2, SK_1, SK_0$ and for round 32, the SubKeys $SK_{127}, SK_{126}, SK_{125}$, and $SK_{124}$ are generated by this subroutine. For SubKeys generation algorithm, the generation of 128 7-bit constants from $\delta_0$ to $\delta_{127}$ is done by the constant generation module. All the operations are carried out over GF($2^8$) governed by the "connection" polynomial $x^7 + x^3 + 1$. The hardware used for the constant generation is a linear feedback shift register (LFSR). The initial state of the 7-bit LFSR $'h'$ is fixed at $(1011010)_2$.

Fault diagnosis of whitening key generation is straightforward as it is rewiring in hardware, e.g., one can use signature-based schemes to have low-complexity error detection. For SubKeys generation and its internal constant generation module, other that logic operations, modular addition is utilized whose fault diagnosis has been discussed previously.

One can use the proposed algorithms in this paper to derive the error indication flags of the underlying block ciphers in SILC. Let us consider two scenarios: if the error indication flags are raised, an incorrect ciphertext and tag is expected. In this case, not only the derived tag in decryption would be faulty, but a faulty ciphertext is transmitted to the receiver side. However, the tags match as authentication is not compromised (same ciphertext is used to derive both tags). In the second scenario, let the HASH or PRF functions are faulty, in which case the tags at the receiver side do not match and the algorithm returns the result $\perp$. In both cases, the error indication flags correctly alarm an incorrect ciphertext.

The work in [35] and [36] presents differential fault analysis (DFA) attacks on HIGHT and LED. The authors used data leaked through random byte model (covered in our simulations for HIGHT) or random bit (for LED) to deduce the secret key (transient fault simulations are also presented in the next section). For instance, the sketch for attacking HIGHT includes three phases: 1) collection of right ciphertext and faulty ciphertexts; 2) the computation of the candidates of subkeys in select rounds and whitening key in the final transformation; and 3) the recovery of the 128-bit secret key from the candidates of subkeys and whitening key. If such attacks are successful by bypassing, for instance, the RERO scheme, we make a small architectural addition to our proposed scheme in order to detect such type of DFA attacks. Since the fault injections are made at the input of a round, we compare the input subcipher in each round (starting from second round) with that generated in previous round. Any discrepancies will be indicated by the error indication flag. Should the attacker try to inject faults in the subcipher in the previous round itself, the previously proposed RERO scheme will detect such an attack. Thus, the RERO and the suggested addition should be able to protect the ciphers against permanent and transient faults and make the DFA attacks more difficult; however, we do not claim that it will be able to detect all types of DFA attacks, for instance, [37], [38].

## IV. Error Simulations and FPGA Implementations

In this section, we present the error simulations and FPGA implementations for LED and HIGHT block ciphers to benchmark their effectiveness.

To benchmark the effectiveness of the proposed schemes for both ciphers, we utilize LFSRs to inject stuck-at faults. The purpose of using LFSRs is to generate pseudo-random fault patterns for every clock cycle. The outputs of the LFSRs are XORed with a random bit and sent as the feedback to the input. In this way, random single and multiple stuck-at faults are injected in the design. We use the polynomial $x^6 + x^3 + 1$ for the implementation of the LFSRs. The LFSRs are used to inject 10 000 faults in LED block cipher and the fault coverage is found to be very close to 100% from the simulation results (this is the case for both parity and interleaved parity signatures for multiple errors; however, burst errors are detected only by the proposed interleaved parities). In HIGHT and for recomputing with encoded operands approach, we have used the LFSRs with the same polynomials to inject the faults. In the first run, the actual operands are sent for computation of the output. In the second run, the operands that are rotated to the right are used to compute the output. In the third run, the output from the last round is rotated left and then compared with the output from the first run. If they do not match, then error indication flags are raised. The modular adders used in the round function module are also injected with single and multiple faults. By the use of LFSRs, around 10 000 single and multiple faults are injected in the HIGHT architecture, where, the results from the simulations for the overall structure shows a very high fault coverage of close to 100% (again, this is the case for multiple random faults, and depending on specific fault models, we might get a bit of divergence in the error detection capabilities).

The proposed methods, being for reliability, can deal with permanent and transient faults. To make sure we cover a good number of fault models, through injecting 10 000 faults in the architecture of HIGHT, we have also investigated transient faults, one/two/three-bit faults, byte faults, and two-byte adjacent faults. The simulations have been performed separately and the results show that:

1) for transient faults, we have 9989 detected faults leading to the error coverage of around 99.9%;
2) for one-bit faults, we have 9854 detected faults leading to the error coverage of around 98.5%;
3) for two-bit faults, we have 9890 detected faults leading to the error coverage of 98.9%;
4) for three-bit faults, we have 9988 detected faults leading to the error coverage of around 99.9%;
5) for one-byte faults, we have 9976 detected faults leading to the error coverage of around 99.8%;
6) for two-byte faults, we have 9980 detected faults leading to the error coverage of 99.8%. We note that although these figures are very high, tailoring the proposed error detection schemes, one may achieve higher error coverage if the overhead is tolerated.

In this section, we also present the area, power consumption, and delay overhead results for LED and HIGHT block ciphers through the FPGA implementations. The benchmarking is done for both the original and fault detection architectures of LED and HIGHT block ciphers. The FPGA implementations are carried out on Xilinx family Virtex-7 with target device 7vx330tffg1157-3.

TABLE II
FPGA IMPLEMENTATIONS OF LED ON VIRTEX-7
(TARGET DEVICE: 7VX330TFFG1157-3)

| Architecture | Area (slices) | Delay ($ns$) | Power ($mW$) |
|---|---|---|---|
| LED-original | 178 | 5.841 | 2.93 |
| LED-signature | 217 (21.9%) | 5.914 (1.2%) | 3.67 (25.2%) |

TABLE III
FPGA IMPLEMENTATIONS OF HIGHT ON VIRTEX-7
(TARGET DEVICE: 7VX330TFFG1157-3)

| Architecture | Area (slices) | Delay ($ns$) | Power ($mW$) |
|---|---|---|---|
| HIGHT-original | 191 | 2.686 | 2.15 |
| HIGHT-proposed | 252 (31.9%) | $\sim$2.686 | 2.17 (0.01%) |

The error detection schemes for LED are shown as LED-signature in Table II. The area in terms of number of slices, delay, and power consumptions are derived for Virtex-7 by implementing the design at 100 MHz frequency. The overheads of the error detection architectures are shown in the parentheses in this table (using Xilinx Integrated Synthesis Environment 14.7 version).

The results from the FPGA implementations of HIGHT-original and fault detection are shown in Table III. The area overhead for the fault detection design is due to the inclusion of error detecting adders used for the recomputed operation. From the results, we can infer that there are negligible power and delay overheads for the architecture as well as acceptable area overhead (we note that such overheads can be adjusted based on the objectives, for instance, avoiding subpipelining could be a compromise between area/power and throughput).

The throughput degradations are also derived for the original and fault detection architectures. For the signature-based schemes, the throughput degradation is negligible, i.e., for LED we have 342.407 Mb/s for the original and 338.180 Mb/s for the fault detection structures leading to the degradation of 1.23. For the HIGHT algorithm, one can perform subpipelining to account for the inherent reduction in throughput. Utilizing one-stage subpipelining, at the cost of registers added, similar throughput to the original architecture is derived.

There has not been any prior work done on error detection methods for these ciphers to the best of our knowledge. Mozaffari-Kermani *et al.* [39] presented fault diagnosis of Pomaranch cipher. They have used bit-interleaved scheme for error detection. We compare the overheads of Pomaranch with the proposed scheme. The combined area and throughput overhead for Pomaranch is 35.5%. The proposed schemes have combined area and delay overheads of 23% for LED and 31.9% for HIGHT, respectively. Since the architecture of Pomaranch and presented fault detection scheme is a lot different than the proposed method, the differences in the overheads are reasonably justified.

## V. CONCLUSION

In this paper, signature-based and recomputing with encoded operand-based approaches are presented for the LED and HIGHT block ciphers. Formulas for the linear and non-linear sub-blocks of the LED block cipher are presented, tailoring which one can achieve the required reliability and overheads. We have also applied subpipelining to overcome the inherent throughput degradation of the proposed approaches for the HIGHT block cipher. Such SPN-based and Feistel network-based block ciphers can be used as part of the authenticated encryption mechanisms, such as SILC. Through fault-injection analysis, it has been shown that the error coverage is close to 100%. Moreover, through FPGA implementations, we have shown that acceptable overheads are achieved for both ciphers. Based on the reliability requirements and available resources, one may utilize the proposed error detection schemes for making the hardware implementations of LED and HIGHT algorithms more reliable.

## REFERENCES

[1] 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. Adv. Cryptol.*, Tallinn, Estonia, 2011, pp. 69–88.

[2] C. D. Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN—A family of small and efficient hardware-oriented block ciphers," in *Proc. Cryptograph. Hardw. Embedded Syst.*, Lausanne, Switzerland, 2009, pp. 272–288.

[3] K. Shibutani *et al.*, "Piccolo: An ultra-lightweight blockcipher," in *Proc. Cryptograph. Hardw. Embedded Syst.*, Nara, Japan, 2011, pp. 342–357.

[4] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Proc. Cryptograph. Hardw. Embedded Syst.*, Vienna, Austria, 2007, pp. 450–466.

[5] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, "SEA: A scalable encryption algorithm for small embedded applications," in *Proc. Smart Card Res. Adv. Appl.*, Tarragona, Spain, 2006, pp. 222–236.

[6] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED block cipher," in *Proc. Cryptograph. Hardw. Embedded Syst.*, Nara, Japan, 2011, pp. 326–341.

[7] R. Beaulieu *et al.*, "Simon and Speck: Block ciphers for the Internet of Things," *Cryptol. ePrint Archive*, vol. 2015, p. 585, 2015.

[8] R. Beaulieu *et al.*, "The SIMON and Speck families of block ciphers," *Cryptol. ePrint Archive*, vol. 2013, p. 404, 2013.

[9] A. Biryukov, A. Roy, and V. Velichkov, "Differential analysis of block ciphers SIMON and Speck," in *Proc. Fast Softw. Encryption*, London, U.K., 2014, pp. 546–570.

[10] S. Sun *et al.*, "Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers," in *Proc. Adv. Cryptol.*, Kaohsiung, Taiwan, 2014, pp. 158–178.

[11] S. Banik *et al.*, "Midori: A block cipher for low energy (extended version)," in *Proc. Cryptol. ePrint Archive*, vol. 2015, p. 1142, 2015.

[12] D. Hong *et al.*, "HIGHT: A new block cipher suitable for low-resource device," in *Proc. CHES*, Yokohama, Japan, 2006, pp. 46–59.

[13] *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. Accessed on Feb. 2017. [Online]. Available: https://competitions.cr.yp.to/caesar.html

[14] *SILC: SImple Lightweight CFB*. [Online]. Available: https://competitions.cr.yp.to/round2/silcv2.pdf

[15] C.-H. Yen and B.-F. Wu, "Simple error detection methods for hardware implementation of Advanced Encryption Standard," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 720–731, Jun. 2006.

[16] G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre, "A reliable architecture for parallel implementations of the Advanced Encryption Standard," *J. Electron. Testing Theory Appl.*, vol. 25, nos. 4–5, pp. 269–278, Aug. 2009.

[17] 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.

[18] 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.

[19] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A low-power high-performance concurrent fault detection approach for the composite field S-box and inverse S-box," *IEEE Trans. Comput.*, vol. 60, no. 9, pp. 1327–1340, Sep. 2011.

[20] 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.

[21] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *Proc. DFTS*, Amherst, MA, USA, 2015, pp. 97–102.

[22] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A structure-independent approach for fault detection hardware implementations of the Advanced Encryption Standard," in *Proc. IEEE Workshop Fault Diagnosis Tolerance Cryptography*, Vienna, Austria, Sep. 2007, pp. 47–53.

[23] R. Karri, G. Kuznetsov, and M. Goessel, "Parity-based concurrent error detection of substitution-permutation network block ciphers," in *Proc. Cryptograph. Hardw. Embedded Syst.*, Cologne, Germany, 2003, pp. 113–124.

[24] M. Karpovsky, K. J. Kulikowski, and A. Taubin, "Robust protection against fault-injection attacks on smart cards implementing the Advanced Encryption Standard," in *Proc. Depend. Syst. Netw.*, Florence, Italy, 2004, pp. 93–101.

[25] M. Mozaffari-Kermani and R. Azarderakhsh, "Reliable hash trees for post-quantum stateless cryptographic hash-based signatures," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Amherst, MA, USA, Oct. 2015, pp. 103–108.

[26] J. H. Patel and L. Y. Fung, "Concurrent error detection in ALUs by recomputing with shifted operands," *IEEE Trans. Comput.*, vol. C-31, no. 7, pp. 589–595, Jul. 1982.

[27] J. Li and E. E. Swartzlander, "Concurrent error detection in ALUs by recomputing with rotated operands," in *Proc. Defect Fault Tolerance VLSI Syst.*, Dallas, TX, USA, 1992, pp. 109–116.

[28] 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.

[29] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1509–1517, Dec. 2002.

[30] A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in *Proc. CHES*, Washington, DC, USA, 2008, pp. 100–112.

[31] J. Rajendran, H. Borad, S. Mantravadi, and R. Karri, "SLICED: Slide-based concurrent error detection technique for symmetric block ciphers," in *Proc. HOST*, Anaheim, CA, USA, 2010, pp. 70–75.

[32] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. (2012). *The LED Block Cipher*. [Online]. Available: https://eprint.iacr.org/2012/600.pdf

[33] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Self-checking carry-select adder design based on two-rail encoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 12, pp. 2696–2705, Dec. 2007.

[34] M. A. Akbar and J. A. Lee, "Comments on 'self-checking carry-select adder design based on two-rail encoding,'" *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2212–2214, Jul. 2014.

[35] Y. Lee, J. Kim, J. H. Park, and S. Hong, "Differential fault analysis on the block cipher HIGHT," in *Future Information Technology, Application, and Service*. Amsterdam, The Netherlands: Springer, 2012, pp. 407–416.

[36] P. Jovanovic, M. Kreuzer, and I. Polian, "A fault attack on the LED block cipher," in *Proc. Int. Workshop (COSADE)*, Darmstadt, Germany, 2012, pp. 120–134.

[37] N. F. Ghalaty, B. Yuce, and P. Schaumont, "Differential fault intensity analysis on PRESENT and LED block ciphers," in *Proc. Int. Workshop (COSADE)*, Berlin, Germany, 2015, pp. 174–188.

[38] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for AES," in *Proc. Int. Workshop (COSADE)*, Berlin, Germany, 2015, pp. 189–203.

[39] 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.

**Srivatsan Subramanian** received the B.E. degree in electrical and electronics engineering from the St. Joseph's College of Engineering, Chennai, India, in 2013. He is currently pursuing his graduate studies with the Electrical and Microelectronic Engineering Department, Rochester Institute of Technology, Rochester, NY, USA, under the supervision of Prof. M. Mozaffari-Kermani and co-supervision of Prof. R. Azarderakhsh.

His current research interests include cryptographic hardware systems, fault tolerant computing, and low-power field-programmable gate array design.

**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.

He joined Advanced Micro Devices 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 a Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellow. He is currently with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA.

Dr. Mozaffari-Kermani 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. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, and the Guest Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations in 2016 and 2017. He was the Lead Guest Editor for the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING for special issues on security. He has been the TPC Member for a number of conferences, including Hardware Oriented Security and Trust (Publications Chair), Design Automation Conference, Design, Automation and Test in Europe.

**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 a Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Research Fellow with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. 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 NSERC Post-Doctoral Research Fellowship in 2012. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS. He is the Guest Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations in 2016 and 2017. He is also the Guest Editor for the IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS for the special issue of Emerging Security Trends for Biomedical Computations, Devices, and Infrastructures in 2015 and 2016.

**Mehrdad Nojoumian** received the Ph.D. degree in computer science from the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada, in 2012.

He held an academic position with Southern Illinois University Carbondale, Carbondale, IL, USA, before joining Florida Atlantic University, Boca Raton, FL, USA, in 2015. He was a Visiting Scholar with New York University, New York, NY, USA, and Centrum Wiskunde & Informatica, Amsterdam, The Netherlands, in 2011. His current research interests include applied cryptography, security and privacy, trust management, and game theory and interdisciplinary research on the intersection of computer and social sciences, such as economics and psychology.

Dr. Nojoumian was a recipient of the Prestigious Natural Sciences and Engineering Research Council of Canada Alexander Graham Bell Canada Graduate Scholarships and UW President's Graduate Scholarship.