Reliable Architecture-Oblivious Error Detection Schemes for Secure Cryptographic GCM Structures

Mehran Mozaffari Kermani, *Senior Member, IEEE*, and Reza Azarderakhsh, *Member, IEEE*

***Abstract*—To augment the confidentiality property provided by block ciphers with authentication, the Galois Counter Mode (GCM) has been standardized by the National Institute of Standards and Technology. The GCM is used as an add-on to 128-bit block ciphers, such as the Advanced Encryption Standard (AES), SMS4, or Camellia, to verify the integrity of data. Prior works on the error detection of the GCM either use linear codes to protect the GCM architectures or are based on AES–GCM architectures, confining the mechanisms to the AES block cipher. Although such structures are efficient, they are not only confined to specific architectures of the GCM but might also not fully take advantage of the parallel architectures of the GCM. Moreover, linear codes have been shown to be potentially ineffective with respect to biased faults. In this paper, we propose algorithm-oblivious constructions through recomputing with swapped ciphertext and additional authenticated blocks, which can be applied to the GCM architectures using different finite field multipliers in $GF(2^{128})$. Such obliviousness for the proposed constructions used in the GCM gives freedom to the designers. We present the results of error simulations and application-specific integrated circuit implementations to demonstrate the utility of the presented schemes. Based on the overhead/degradation tolerance for implementation/performance metrics, one can fine-tune the proposed method to achieve more reliable architectures for the GCM.**

***Index Terms*—Application-specific integrated circuit (ASIC), Galois Counter Mode (GCM), $GF(2^{128})$ multiplier, reliability.**

NOMENCLATURE

ASIC	Application-specific integrated circuit.
FPGA	Field-programmable gate array.
GCM	Galois Counter Mode.
GE	Gate equivalent.
RESCAB	Recomputing with swapped ciphertext and additional authenticated blocks.

Manuscript received November 20, 2017; revised March 12, 2018 and June 3, 2018; accepted September 1, 2018. Date of publication December 12, 2018; date of current version November 26, 2019. The work of M. Mozaffari Kermani and R. Azarderakhsh was supported by the National Institute of Standards and Technology under the U.S. Federal Agency Award 60NANB16D245. Associate Editor: F. Belli. *(Corresponding author: Mehran Mozaffari Kermani.)*

M. Mozaffari Kermani is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: mehran2@usf.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/TR.2018.2882484

I. INTRODUCTION

STANDARDIZED by the National Institute of Standards and Technology, the GCM [1] is used in conjunction with block ciphers, such as the Advanced Encryption Standard (AES), Camellia, SMS4, and CLEFIA, through a hash function over a binary finite field, to provide authentication, augmented to confidential data.

There have been previous proposed architectures for the GCM in the literature. The FPGA bitstream encryption/authentication through the GCM, bit-sliced implementations for 64-bit Intel processors [2], and software-based implementations [3] are among the early works adopting the GCM. Moreover, GCM implementations, such as the ones in [4]–[9], provide the confidence to utilize the architectures for different secure applications. In addition, the algorithmic security of the GCM has been scrutinized in recent studies [10], [11].

The GCM provides authentication for encrypted/raw data; nevertheless, natural faults, e.g., through exposure to laser and cosmic ray particles, such as alpha and gamma rays, and malicious faults undermine its reliability. Previous works on the error detection of the GCM have either utilized linear codes, e.g., parity and cyclic redundancy check (CRC) [7], or presented different combined AES–GCM error detection architectures by carefully scrutinizing the implementation cycles [9].

In this paper, we present error detection schemes that are not confined to specific GCM architectures. We also verify the error coverage of the proposed approaches through error simulations for both transient and permanent multiple and biased faults. Our schemes constitute algorithm-oblivious constructions through RESCAB, which can be applied to the GCM architectures using different finite field multipliers in $GF(2^{128})$; for example, in our experiments, we have used a quadratic and a subquadratic multiplier to show the obliviousness of the proposed method. The obliviousness of our schemes comes from the fact they can be applied to different constructions (and are not confined to just one, such as schemes that derive predicted signatures). For instance, one can use different multipliers, different numbers of branches, different finite fields, such as polynomial basis or normal basis, and different modulo-2 addition trees. We benchmark the proposed architectures to assess their ability to detect transient and permanent faults by performing fault injection simulations. Moreover, we implement the proposed error detection architectures on an ASIC platform using 65-nm standard-cell library.

This paper is organized as follows. In Section II, we present the proposed error detection constructions for high-throughput

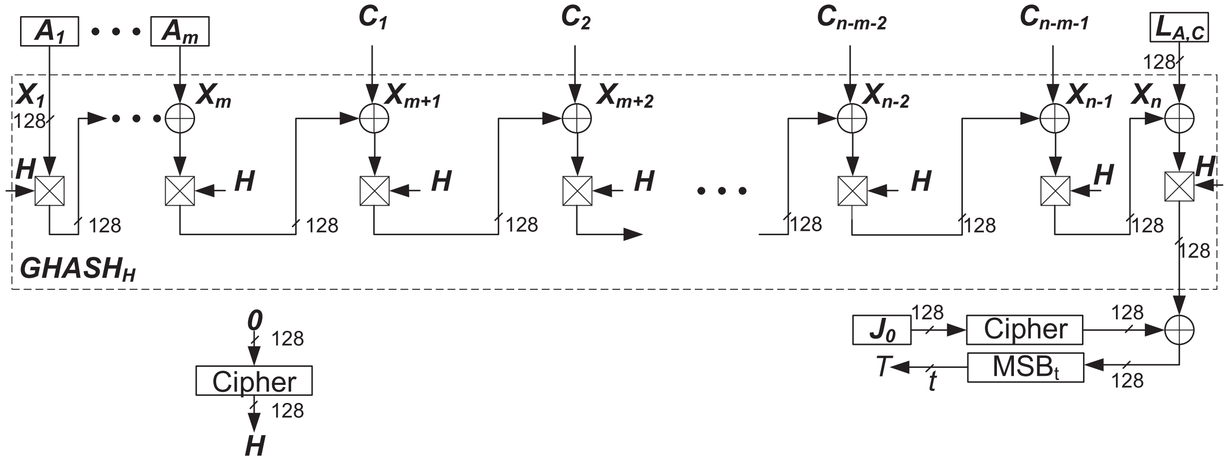

Fig. 1. GCM architecture including GHASH.

GCM. In this section, through case studies as well as through a general formula, we propose our efficient error detection construction. We, then, assess the proposed scheme on an ASIC, preceded by error injection simulations in Section III in order to show the efficacy of the approach proposed in this paper. Finally, conclusions are presented in Section IV.

II. Proposed Error Detection Constructions

In the GCM, the encrypted blocks are asserted in conjunction with additional potential raw data, as shown in Fig. 1, to the Galois Hash (GHASH) function, which is constructed by finite field multiplications with a fixed parameter [hash subkey (H)]. In Fig. 1, $A_1 - A_m$ are additional authenticated data (ADD) (not encrypted) to be verified for integrity. Moreover, $C_1 - C_{n-m-1}$ are the 128-bit ciphertext blocks derived through a given block cipher. Together with the ADD and an added 128-bit block for specifying the length $L_{A,C}$, we get n blocks of data, i.e., $X_1 - X_n$. The GCM can provide authentication assurance for additional data (of practically unlimited length per invocation) that are not encrypted. Such data are used to derive the authentication tag as part of the GCM. The plaintext and the additional authenticated data are the two categories of data that the GCM protects. We note that the GCM protects the authenticity of the plaintext and the AAD. The GCM also protects the confidentiality of the plaintext, while the additional authenticated data are protected in terms of authenticity. Such data might include addresses, ports, sequence numbers, protocol version numbers, and other fields that indicate how the plaintext should be treated.

The GHASH function calculates the following: $\sum_{i=1}^{n} X_i H^{n-i+1} = X_1 \times H^n \oplus X_2 \times H^{n-1} \oplus \cdots \oplus X_{n-1} \times H^2 \oplus X_n \times H$. Note that we use the symbol $\oplus$ for XOR operation or modulo-2 addition and the symbol $\times$ for multiplication in finite field. The hash subkey is generated by applying the 128-bit block cipher to the "zero" input block, i.e., $0 = (0, 0, \ldots, 0) \in GF(2^{128})$, as shown in Fig. 1. The arithmetic operations are done using the irreducible polynomial $f(x) = x^{128} + x^7 + x^2 + x + 1$. Eventually, the authentication tag *T* (with a length of *t* bits for the most significant part of the added output with the encrypted given input J_0) is derived as shown in Fig. 1. The following example shows a simple multiplication for GHASH in finite field.

Example: Suppose we get the polynomial for the 128-bit hash subkey (H) as $H(x) = x^{27} + x^{25} + x^{20} + x^4 + x + 1$. To find the polynomial representing $\beta(x) = (X_{n-1} \times H + X_n) \times H \bmod f(x)$, assuming the two 128-bit input blocks to GHASH as $X_{n-1} = x^{89} + x^{23} + x^{10}$ and $X_n = x^{93} + x^{24} + x^{10} + x$, one would need to perform a finite field multiplication followed by reduction using the irreducible polynomial. We note that through the irreducible polynomial, we have $x^{128} = x^7 + x^2 + x + 1$; thus, $x^{129} = x^8 + x^3 + x^2 + x$, $x^{130} = x^9 + x^4 + x^3 + x^2$, $x^{131} = x^{10} + x^5 + x^4 + x^3$, and the like. The final simplified answer for $\beta(x) = (X_{n-1} \times H + X_n) \times H \bmod f(x)$ will have these powers of x: 120, 118, 113, 94, 93, 91, 89, 77, 73, 64, 63, 60, 51, 50, 49, 44, 37, 35, 31, 30, 26, 24, 23, 22, 21, 17, 16, 15, 14, 13, 8, 5, 3, i.e., $\beta(x) = x^{120} + x^{118} + x^{113} + x^{94} + x^{93} + x^{91} + x^{89} + x^{77} + x^{73} + x^{64} + x^{63} + x^{60} + x^{51} + x^{50} + x^{49} + x^{44} + x^{37} + x^{35} + x^{31} + x^{30} + x^{26} + x^{24} + x^{23} + x^{22} + x^{21} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^8 + x^5 + x^3$.

A. Motivation for the Proposed Approach

Error detection constructions of the GCM are important because if there are errors in the output of the GCM, the respective tag would be calculated incorrectly and that would cause errors in providing authenticity. In other words, we note the following.

1) If the block cipher output is faulty, the resulting tag would be calculated based on that, causing an incorrect tag based on the erroneous output.
2) If the GHASH function is faulty, one would have a correct ciphertext with a wrong tag, which would cause failure in authenticity.
3) If both these outputs are erroneous, not only do we get an incorrect ciphertext but also an incorrect tag.

Linear codes used in [7] are effective, especially for special classes of fault models and also random faults. However, there are two main problems with using linear codes for the GCM. First, the formulations and error detection mechanisms would be confined to one architecture of the GCM, e.g., one type of multiplier in $GF(2^{128})$ (and, thus, would need to be revisited/implemented accordingly). Moreover, the designers need to have freedom in choosing the block ciphers paired with the GCM. In addition, we note that the parallel GCM architectures, e.g., presented in [5] and [8] (to alleviate the performance/throughput and, thus, the efficiency of the architectures), can be leveraged for error detection to achieve more efficient constructions.

To resolve the above-mentioned complications, in this paper, we present error detection schemes that are not confined to specific GCM architectures, whose details are presented in the following.

B. Error Detection for Parallel GHASH

The error detection construction proposed in this paper is based on the parallel high-throughput realization of the GCM in [8]. In other words, by implementing the GCM in parallel branches, one would have q parallel adders–multipliers for deriving the output of the GCM. Noting that $q = 2^j$, $1 \leq j \leq \lfloor \log_2 n \rfloor$, which leads to a lower number of clock cycles, let us present the following example for $q = 8, n = 24$:

$$\begin{gathered} cX_1 \times H^8 \times H^8 \times H^8 \oplus X_2 \times H^8 \times H^8 \times H^7 \\ \oplus \cdots \oplus X_8 \times H^8 \times H^8 \times H \oplus X_9 \times H^8 \times H^8 \\ \oplus X_{10} \times H^8 \times H^7 \oplus \cdots \oplus X_{24} \times H. \end{gathered}$$

We also achieve the following for the general case:
$X_1 \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q} \text{ times}} \oplus X_2 \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H^{q-1} \oplus \cdots \oplus X_i \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H^{q-i+1} \oplus \cdots \oplus X_q \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H \oplus X_{q+1} \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \oplus X_{q+2} \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-2 \text{ times}} \times H^{q-1} \oplus \cdots \oplus X_n \times H$. Here, only the exponentiations of the hash subkey to the powers of 2 are used for the exponentiations H^{q-i+1}, $1 \leq i \leq q$, to have low-complexity structures.

Our schemes constitute algorithm-oblivious constructions through RESCAB, which can be applied to the GCM architectures implemented using different finite field multipliers in $GF(2^{128})$. The RESCAB is based on asserting the input data blocks and swapping the blocks to get to the output. There are two advantages in the proposed scheme. First, the derivation of decoded output (after asserting the swapped inputs) to compare with the normal output is trivial. Second, the scheme is applicable to any number of branches $q = 2^j$, $1 \leq j \leq \lfloor \log_2 n \rfloor$, any type of finite field multiplication, and any combination of encrypted and raw data.

Process of error detection: The scheme proposed here is based on recomputation. Therefore, no additional circuitry, other than added registers for storing the results of cycles and also comparators, is used. This allows different implementations of the GCM to be utilized if needed, as opposed to the schemes based on added error detection units. Let us explain our scheme for $q = 2^j$, $1 \leq j \leq \lfloor \log_2 n \rfloor$. The output of the GCM can be derived by grouping the input blocks, i.e., we derive the following example for $q = 8, n = 24$:

$$\begin{gathered} X_1 \times H^8 \times H^8 \times H^8 \oplus X_9 \times H^8 \times H^8 \oplus \\ \cdots \oplus X_i \times H^8 \times H^8 \times H^{\overbrace{1+2+4+\cdots}^{8-i+1}} \oplus \\ X_{i+8} \times H^8 \times H^{\overbrace{1+2+4+\cdots}^{8-i+1}} \oplus \\ \cdots \oplus X_8 \times H^8 \times H^8 \times H \oplus \\ X_{16} \times H^8 \times H \oplus \cdots \oplus X_{24} \times H. \end{gathered}$$

We also achieve the following for the general case: $X_1 \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H^q \oplus X_{q+1} \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \oplus \cdots \oplus X_i \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H^{\overbrace{1+2+4+\cdots}^{q-i+1}} \oplus X_{i+q} \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-2 \text{ times}} \times H^{\overbrace{1+2+4+\cdots}^{q-i+1}} \oplus \cdots \oplus X_q \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-1 \text{ times}} \times H \oplus X_{2q} \times \underbrace{H^q \times \cdots \times H^q}_{\frac{n}{q}-2 \text{ times}} \times H \oplus \cdots \oplus X_n \times H.$

Phase 1: It is noted that because the number of input blocks n is large (combined additional authenticated data blocks and encrypted blocks), in very high percentage of the iterations, i.e., $\frac{n}{q} - 1$ out of $\frac{n}{q} - 1 + \log_2 q$ times, we have H^q as one of the inputs of the finite field multipliers used to derive GHASH, e.g., for $n = 2^{20}$ and $q = 8$, we have $\frac{n}{q} - 1 = (0.999997)[\frac{n}{q} - 1 + \log_2 q]$. We need $\log_2 q$ times multiplication cycles, in general, so that using the exponentiations to the powers of two, we can implement H^{q-1} for low-complexity realizations. If one stores the 128-bit output result of the GCM after $\frac{n}{q} - 1$ cycles and compares that with the result obtained for the swapped variant of the 128-bit input blocks (for instance, the input for the first multiplier now becomes the input of the kth multiplier for $2 \leq k \leq q$), then, after $\frac{n}{q} - 1$ cycles, the result must be the same, without the need for decoding. This low-complexity construction is another advantage of the proposed architecture.

Fig. 2 shows the construction of the proposed scheme for $q = 4$ parallel branches. The swapped inputs through the RESCAB are also shown by the arrows. We note that swapping the input blocks means asserting specific operands for each branch and, then, changing the asserted operands by swapping the inputs to the branches. Moreover, we use an intermediate register R_{int} to store the result after $\frac{n}{4} - 1$ cycles. We have also shown the architecture for $q = 32$ in Fig. 3. To explain in detail, we have used the simple example of $n = 32$. In this case and as shown in Fig. 3, swapping has been done in the order shown; nevertheless, such a construction is not confined to just this order (see Fig. 3). It is possible that we get transient faults (some natural faults and, especially, malicious faults are of such nature) that cannot

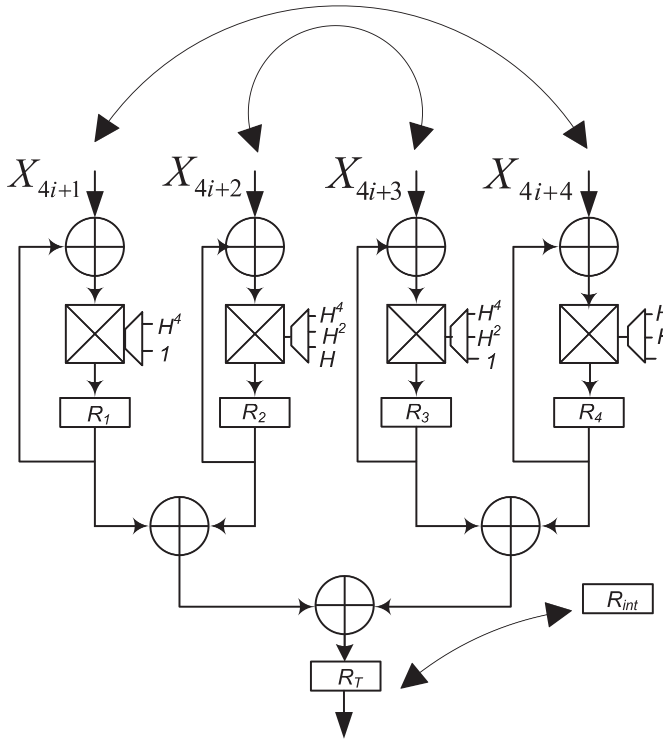

Fig. 2. Proposed RESCAB construction for GCM ($q = 4$).

be detected in some cases as we cover most but not all the iterations. The following simple example elaborates on the fact that as we increase the number of branches, the performance is increased but that depends on the area tolerance in specific applications.

Example: Comparing Figs. 2 and 3 for two parallel constructions, i.e., for $q = 4$ parallel branches in Fig. 2 and for $q = 32$ in Fig. 3, one notes that the area footprints in these two figures vary significantly. Let us examine in more detail what the implications of having 4 and 32 branches are with respect to area complexity. Fig. 2 needs four parallel multipliers as opposed to Fig. 3 in which 32 are needed. However, in the first phase, as discussed previously, for $n = 128$, and based on the number of cycles $\frac{n}{q} - 1$, 3 and 31 cycles are needed, respectively. The choice for these architectures depends on the area tolerance of the applications. For example, in implantable and wearable medical devices, one might use low-area footprints, whereas for game console security, performance is the bottleneck.

Phase 2: The remaining $\log_2 q$ iterations constitute a very small fraction of cycles needed for the entire output derivation, e.g., for $n = 2^{20}$ and $q = 8, 0.0003\%$. Thus, we propose a simple recomputation that adds very slightly to performance overhead. We note that in this case, the left-hand side of Fig. 3 is executed twice without swapping the operands. We detect the permanent faults in the previous phase through recomputation.

The proposed scheme can also be applied to Poly1305, which is a cryptographic message authentication code (can be used to verify the data integrity and the authenticity of a message) [12]. Google has selected Poly1305 (along with Bernstein's ChaCha20 symmetric cipher) as a replacement for RC4 in TLS/SSL [13]. Poly1305 uses 128-bit blocks, which are used as coefficients of a polynomial, evaluated modulo the prime number $2^{130} - 5$. Similar to the GCM, Poly1305 derives an authentication tag that is appended to the output of a block cipher, say the AES with key k, and gets verified at the receiver side. Briefly, the input messages construct the integers $c_1 - c_q$, and the tag is derived by multiplying these with a type of "key" r (similar to the hash subkey in the GCM) as follows: $(((c_1 r^q + c_2 r^{q\text{-}1} + +c_q r^1) \bmod 2^{130} - 5) + \text{AES}_k(n)) \bmod 2^{128}$. Similar to the RESCAB method for the GCM, here, exponentiations of r can be realized in hardware (and the input entries can be swapped) in parallel. We note that Poly1305 is efficiently implemented in software, and the adoption of our proposed scheme does not mean that we advocate the hardware implementations of Poly1305.

III. Error Simulations, ASIC Implementations, and Comparisons

In simulations and in theory, we note that the fundamental difference between security attacks and random faults is the intelligent-attacker assumption. However, due to limitations in practice, in real implementations, one needs to note that it is very difficult to control these variables due to the nature of the implementations. In this section, we present the results of our error injection simulations for both random and biased faults.

Error injection simulations: We have used linear-feedback shift register based injections for stuck-at faults for the GCM architecture with $q = 8$ parallel branches for two types of finite field multipliers, assessed after applying the RESCAB scheme. We have considered both unbiased and biased faults. We have injected 10 000–50 000 faults, and for the error injection of biased faults, we have considered having two adjacent branches (out of eight in our example) faulty.

Figs. 4 and 5 show the results of our simulations for permanent faults using bit-parallel finite field multiplier (quadratic); see the results depicted in Fig. 4 for random multiple faults (solid line) and biased faults (dotted line), and for the Karatsuba–Ofman subquadratic multiplier, see the results in Fig. 5. We have considered the case study of 2^{10} cycles for throughput degradation alleviation. Similarly, Figs. 6 and 7 show the results of our simulations through transient fault injections.

As shown in Figs. 4–7, the error coverage for the experimented injections is high. We note that the slight fluctuations seen in these figures do not follow a trend. This has been verified by the injection of up to 80 000 faults for the case study of permanent biased faults, and the results show a similar error coverage as Fig. 4. We have also increased the number of our simulation-based experiments by having different inputs and fault numbers/locations, leading to 100 000 instances. Similar to previous experiments, this confirms the accuracy of our error coverage results. Our proposed schemes can detect transient faults with high error coverage. The proposed schemes also provide high error coverage for permanent and long transient faults.

We would also like to discuss two unlikely cases that may appear during permanent or long transient faults. One event can be "masking," in which the output is not erroneous, even if a

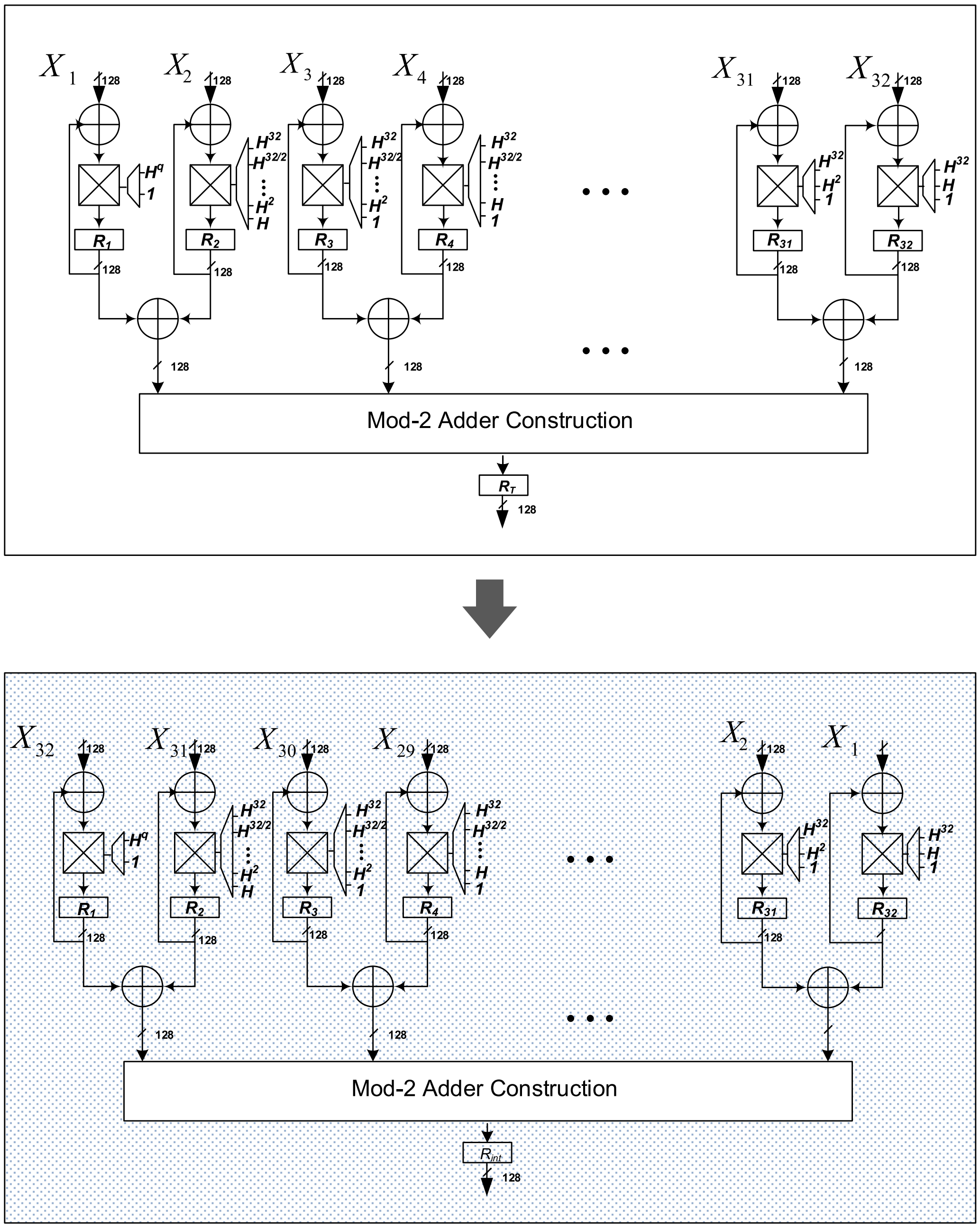

Fig. 3. Proposed RESCAB construction for GCM $q = 32$ and $n = 32$ cycles: Original operands (top subfigure) and swapped operands (bottom shaded subfigure).

fault exists in the intermediate logic. Such cases are excluded because the circuit masks the faults and these are not translated to errors. The second instance is a rare case where all the entries of operands are zero in each column. As swapping any zero value will keep it unaltered, this benchmark case cannot be detected, confirming the results of our simulations. However, applying all the input bits to a logic OR gate can be a secondary measure to detect the errors in this unlikely case.

Karatsuba–Ofman subquadratic multipliers have subquadratic complexity and are realized by converting the costly multiplications to additions at the cost of added delay. The bit-parallel construction, on the other hand, is much larger but considerably fast when implemented. We do not present the error detection schemes of the block ciphers in this paper as it has been studied in previous works [9]–[26].

In Fig. 8, we see a clear connection between the simulated architecture and the proposed scheme. As seen in this figure, for eight parallel branches and through adding the proposed approach, the transient and permanent faults (see Figs. 4–7) are injected through the approach presented in this section. Moreover, oblivious of the multiplier type, the simulations for Fig. 8 (see the results depicted in Figs. 4–7) show high error coverage.

Throughput alleviation: We propose two methods for alleviating the throughput of the proposed constructions. First, through deep subpipelining of the finite field multipliers, one can carefully assert/reorder the normal and swapped raw and encrypted

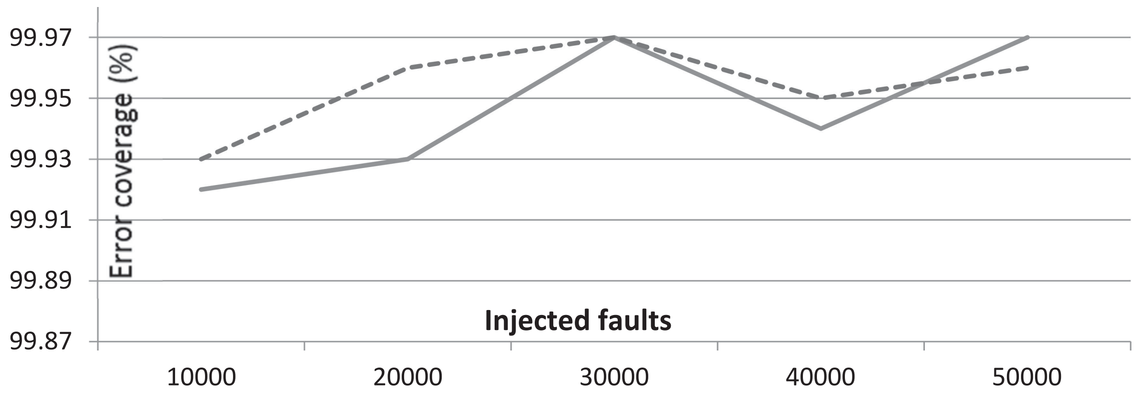

Fig. 4. Error injection simulation results using bit-parallel finite field multiplier (quadratic) in the GCM for $q = 8$: Permanent faults (solid line: multiple faults, dotted line: biased faults).

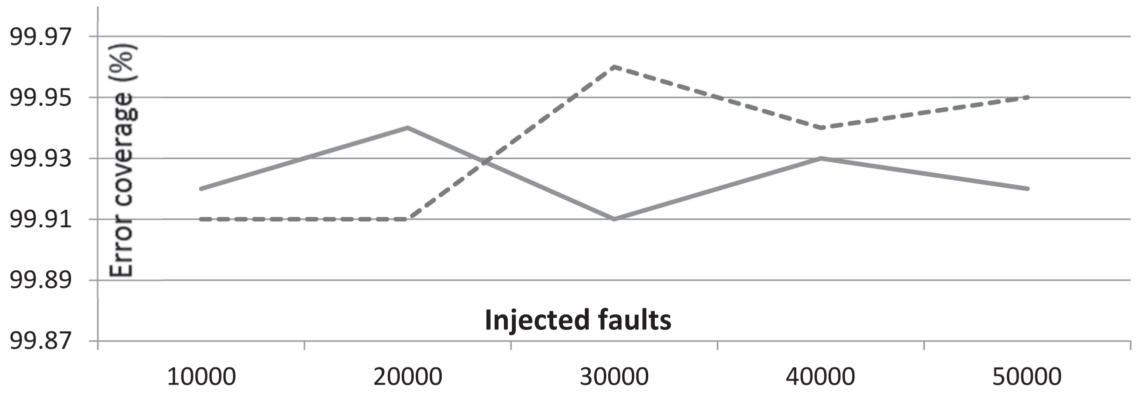

Fig. 5. Error injection simulation results using Karatsuba–Ofman subquadratic multiplier in the GCM for $q = 8$: Permanent faults (solid line: multiple faults, dotted line: biased faults).

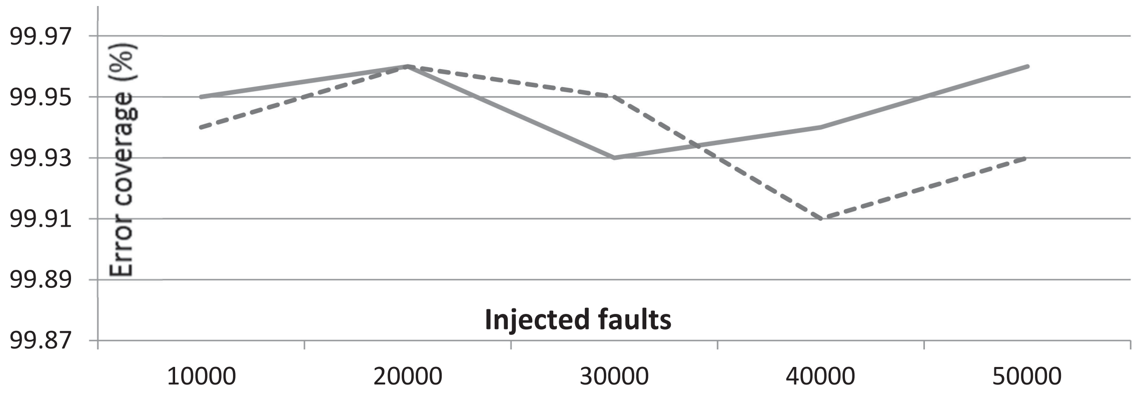

Fig. 6. Error injection simulation results using bit-parallel finite field multiplier (quadratic) in the GCM for $q = 8$: Transient faults (solid line: multiple faults, dotted line: biased faults).

data. This would increase the throughput of the RESCAB method at the expense of added hardware complexity.

The second remedy is to recompute with the swapped entries for just a fraction of the total cycles needed for deriving the GCM. Let us have an example to clarify this approach. In the proposed parallel approach, let us have $n = 2^{20}$ blocks and $q = 8$ parallel constructions (we get $\frac{n}{q} - 1 = (0.999997)[\frac{n}{q} - 1 + \log_2 q]$). In the original RESCAB, to achieve high error coverage, after the first $2^{17} - 1$ cycles, we compare the normal and encoded results. Based on the reliability requirements and throughput degradation tolerance, however, this can be lowered. For instance, the intermediate register can be populated with the result of normal and swapped operands after 2^{16} cycles to roughly halve the cycles needed before reaching the intermediate value to compare for error detection (to halve the throughput degradation as well). We note that such flexibility is an advantage of the proposed approach, noting the complications presented in the previous section.

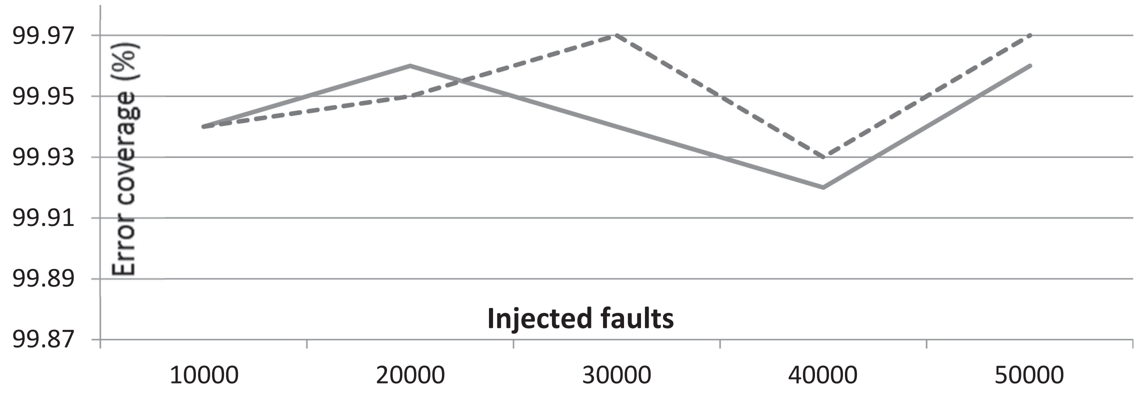

Fig. 7. Error injection simulation results using Karatsuba–Ofman subquadratic multiplier in the GCM for $q = 8$: Transient faults (solid line: multiple faults, dotted line: biased faults).

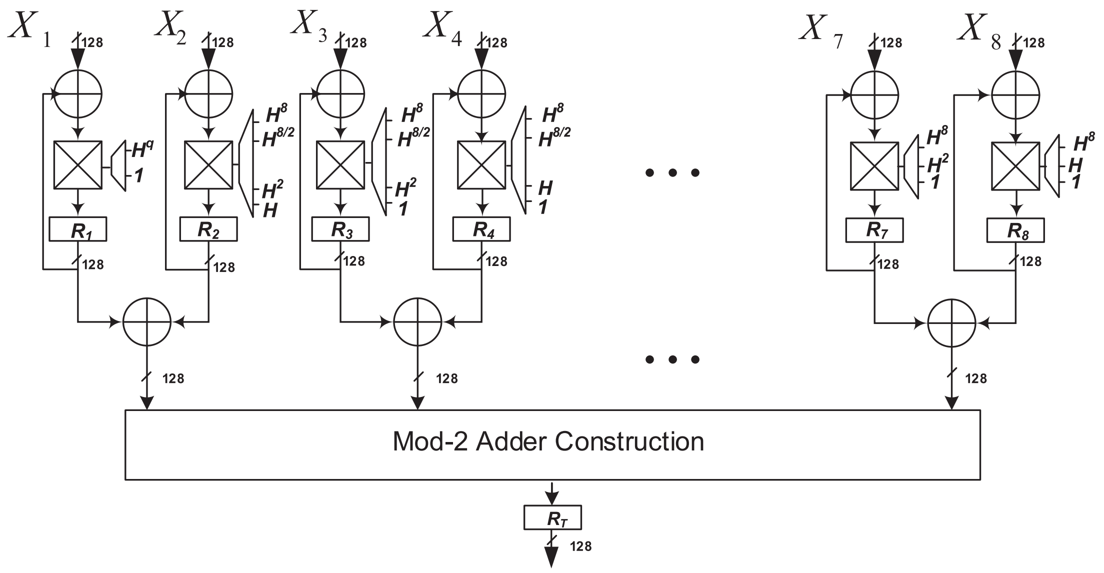

Fig. 8. Pictorial view of the simulated circuit for error detection of the GCM ($q = 8$).

TABLE I
ASIC 65-nm TSMC BENCHMARK OF THE PROPOSED SCHEMES FOR AES–GCM AND $n = 2^{30}$ BLOCKS

Scheme	Area (kGE), overhead	Thr' [kbps], overhead
Plain	≃918k, 4.9%	0.301, 102%
Deep sub-pipelining	≃934k, 6.7%	0.564, 7.7%
RESCAB 2^{25} cycles	≃918k, 4.9%	0.538, 11.9%
RESCAB 2^{10} cycles	≃918k, 4.9%	0.559, 8.5%

ASIC implementations: Using 65-nm ASIC synthesis, we also present the overhead of the presented constructions for the case study of AES–GCM. The benchmarking is performed for the error detection architectures using TSMC 65-nm library and Synopsys Design Compiler (presented in Table I for area [kilo gate equivalent (kGE), which is the normalized area for 2-input NAND gate] and respective throughputs).

In Table I, we have presented four schemes based on bit-parallel multipliers for the GCM with $q = 8$, i.e., variants of the proposed approach in the last three rows based on 1) one-stage deep subpipelining (highest error coverage and area overhead); 2) case study of 2^{25} cycles; and 3) case study of 2^{10} cycles. The former approach increases the area overhead (which is normally negligible for time redundancy) to decrease throughput degradation, whereas the latter cases have lower overhead but at the expense of covering permanent and a subset of transient faults. We have also presented in the first row the plain approach without subpipelining.

Comparisons: Let us consider linear codes, such as CRC, parities, and interleaved parities, for error detection of the GCM constructions. Although such structures are efficient, they have major drawbacks. They are not only confined to specific architectures for the ciphers within blockcipher-GCM (for example, if any of such codes are derived for the AES, obviously, they cannot be used for CLEFIA block cipher) but also to the very architecture in GHASH within the GCM for which such codes are derived (for example, the parities derived for error detection in polynomial basis quadratic multipliers cannot be utilized for composite field normal basis subquadratic multipliers). In contrast with the previous work, which either utilizes linear codes, e.g., parity and CRC [7] (confined to specific architectures of the GCM and considers random and not biased faults), or focuses on the AES–GCM, not taking advantage of the parallel constructions presented in [5] and [8] to fine-tune the schemes [9]. This paper is not limited to the types of finite field multipliers, can be tailored towards higher reliability or lower overhead, and also can be applied to different block ciphers, and considers biased fault models. Specifically, the two previous works on the error detection of the GCM have either utilized linear codes [7] or presented different combined AES–GCM error detection architectures by carefully scrutinizing the implementation cycles [9]. Let us start with the former. Linear codes used in [7] are effective especially for special classes of fault models and also random faults. However, there are two main problems with using linear codes for the GCM. First, the formulations and error detection mechanisms would be confined to one architecture of the GCM, e.g., one type of multiplier in $GF(2^{128})$ (and, thus, would need to be revisited/implemented accordingly). Using any other type of multiplier would result in modifications in the architectures proposed in such schemes. This is rather important as we do not want to get confined to specific types or architectures (and not being able to tailor the GCM architectures utilized in different usage models). The error coverage of the proposed scheme is very high (more than 99.9%), which is higher than different variants of the one in [7]: 48%, 74%, 87%, 92.5%, 96%, and 98%. Moreover, although much effective, for the latter, the designers need to have freedom in choosing the block ciphers paired with the GCM. In addition, we note that the parallel GCM architectures, e.g., presented in [5] and [8] (to alleviate the performance/throughput and, thus, the efficiency of the architectures), can be leveraged for error detection to achieve more efficient constructions. The proposed scheme in this paper alleviates the above-mentioned shortcomings.

IV. Conclusion

In this paper, we proposed, simulated through error injections, and implemented on ASIC our proposed schemes for the GCM. Our schemes constituted algorithm-oblivious constructions through RESCAB, which can be applied to the GCM architectures using different finite field multipliers in $GF(2^{128})$; for example, in our experiments, we used a quadratic and a subquadratic multiplier to show the obliviousness of the proposed approach. The proposed scheme is not limited to the types of finite field multipliers, can be tailored towards higher reliability or lower overhead, can be applied to different block ciphers, and considers biased fault models. Such obliviousness for the proposed constructions used in the GCM gives freedom to the designers. Through deep subpipelining, we reached an area overhead of 6.7% and a throughput degradation of 7.7%. Moreover, for RESCAB 2^{25} cycles (2^{10} cycles), we reached an area overhead of 4.9%, while throughput degradations are 11.9% and 8.5%, respectively. Based on the available resources, one may utilize the proposed error detection schemes for making the hardware implementations of the GCM more reliable.

References

[1] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," National Inst. of Standards Technol., Gaithersburg, MD, USA, Rep. NIST SP 800-38D, 2007.

[2] E. Käsper and P. Schwabe, "Faster and timing-attack resistant AES-GCM," in *Proc. 11th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2009, pp. 1–17.

[3] K. Jankowski and P. Laurent, "Packed AES-GCM algorithm suitable for AES/PCLMULQDQ instructions," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 135–138, Jan. 2011.

[4] S. Lemsitzer, J. Wolkerstorfer, N. Felbert, and M. Braendli, "Multi-gigabit GCM-AES architecture optimized for FPGAs," in *Proc. 9th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2007, pp. 227–238.

[5] A. Satoh, T. Sugawara, and T. Aoki, "High-performance hardware architectures for Galois Counter Mode," *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 917–930, Jul. 2009.

[6] B. Buhrow, K. Fritz, B. Gilbert, and E. Daniel, "A highly parallel AES-GCM core for authenticated encryption of 400 Gb/s network protocols," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs*, 2015, pp. 1–7.

[7] A. A. Kouzeh Geran and A. Reyhani-Masoleh, "A CRC-based concurrent fault detection architecture for Galois/Counter Mode (GCM)," in *Proc. IEEE 23rd Symp. Comput. Arithmetic*, 2016, pp. 24–31.

[8] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Efficient and high-performance parallel hardware architectures for the AES-GCM," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1165–1178, Aug. 2012.

[9] X. Guo and R. Karri, "Low-cost concurrent error detection for GCM and CCM," *J. Electron. Testing*, vol. 30, no. 6, pp. 725–737, 2014.

[10] S. Gueron, A. Langley, and Y. Lindell, "AES-GCM-SIV: Specification and analysis," *eprint 2017/168*, 2017, pp. 1–16.

[11] T. Iwata and K. Minematsu, "Stronger security variants of GCM-SIV," *eprint 2016/853*, 2016, pp. 1–24.

[12] D. Bernstein, "The Poly1305-AES message-authentication code," in *Proc. 12th Int. Conf. Fast Softw. Encryption*, 2005, pp. 32–49.

[13] "Google swaps out crypto ciphers in OpenSSL," *InfoSecurity*, Apr. 24, 2014.

[14] G. Xiaofei 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.

[15] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptographic Eng.*, vol. 5, no. 3, pp. 153–169, 2015.

[16] M. Yasin, B. Mazumdar, S. Subidh Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2015, pp. 97–102.

[17] 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.* (Special Issue on Embedded Device Forensics and Security: State of the Art Advances), vol. 16, no. 2, Apr. 2017, Art. no. 59.

[18] 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.

[19] M. Mozaffari Kermani, R. Azarderakhsh, C. 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.

[20] 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 Nanotechnol. Syst.*, Oct. 2015, pp. 103–108.

[21] 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.

[22] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Fault detection structures of the S-boxes and the inverse S-boxes for the Advanced Encryption Standard," *J. Electron. Testing: Theory Appl.*, vol. 25, no. 4, pp. 225–245, Aug. 2009.

[23] M. Mozaffari Kermani, R. Azarderakhsh, A. Sarker, and A. Jalali, "Efficient and reliable error detection architectures of Hash-Counter-Hash tweakable enciphering schemes," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 2, Apr. 2018, Art. no. 54.

[24] M. Mozaffari Kermani and A. Reyhani-Masoleh, “A high-performance fault diagnosis approach for the AES SubBytes utilizing mixed bases,” in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr.*, Nara, Japan, Sep. 2011, pp. 80–87.

[25] M. Mozaffari Kermani and A. Reyhani-Masoleh, “Reliable hardware architectures for the third-round SHA-3 finalist Grostl benchmarked on FPGA platform,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Vancouver, BC, Canada, Oct. 2011, pp. 325–331.

[26] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. P. Chakrabarti, “Fault space transformation: A generic approach to counter differential fault analysis and differential fault intensity analysis on AES-like block ciphers,”*IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 5, pp. 1092–1102, May 2017.

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 in electrical and computer engineering from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively.

In 2011, he joined the 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 an NSERC Postdoctoral Research Fellow. From 2013 to 2017, he was an Assistant Professor with the Rochester Institute of Technology, and since 2017, he has been with the Computer Science and Engineering Department, University of South Florida, Tampa, FL, USA.

Dr. Kermani was the recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Postdoctoral Research Fellowship in 2011 and the Texas Instruments Faculty Award (Douglas Harvey) in 2014. He is an Associate Editor for the IEEE TRANSACTIONS ON VLSI SYSTEMS, the *ACM Transactions on Embedded Computing Systems*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, and a Guest Editor for 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 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 Technical Program Committee (TPC) member for a number of conferences including IEEE International Symposium on Hardware Oriented Security and Trust (HOST) (Publications Chair), Design Automation Conference (DAC), Design, Automation and Test in Europe (DATE), RFID Security Conference (RFIDSec), Lightweight Security Conference (LightSec), International Workshop on the Arithmetic of Finite Fields (WAIFI), Fault Diagnosis and Tolerance in Cryptography (FDTC), and IEEE International Symposium on Defect and Fault Tolerance (DFT) in VLSI and Nanotechnology Systems (DFT).

Reza Azarderakhsh (M’11) received the Ph.D. degree in electrical and computer engineering from Western University, London, ON, Canada, in 2011.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Florida Atlantic University, Boca Raton, FL, USA. His research interests include finite field and its applications, elliptic curve cryptography, pairing-based cryptography, and postquantum cryptography.

Dr. Azarderakhsh was the recipient of the NSERC Postdoctoral Research Fellowship while working with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. He was the Guest Editor for 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 also the Guest Editor for the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS for the special issue on “Security.” He is an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS (TCAS-I). He is an I-SENSE Fellow.