Efficient and Reliable Error Detection Architectures of Hash-Counter-Hash Tweakable Enciphering Schemes

MEHRAN MOZAFFARI-KERMANI, University of South Florida
REZA AZARDERAKHSH, Florida Atlantic University
AUSMITA SARKER, University of South Florida
AMIR JALALI, Florida Atlantic University

Through pseudorandom permutation, tweakable enciphering schemes (TES) constitute block cipher modes of operation which perform length-preserving computations. The state-of-the-art research has focused on different aspects of TES, including implementations on hardware [field-programmable gate array (FPGA)/ application-specific integrated circuit (ASIC)] and software (hard/soft-core microcontrollers) platforms, algorithmic security, and applicability to sensitive, security-constrained usage models. In this article, we propose efficient approaches for protecting such schemes against natural and malicious faults. Specifically, noting that intelligent attackers do not merely get confined to injecting multiple faults, one major benchmark for the proposed schemes is evaluation toward biased and burst fault models. We evaluate a variant of TES, i.e., the Hash-Counter-Hash scheme, which involves polynomial hashing as other variants are either similar or do not constitute finite field multiplication which, by far, is the most involved operation in TES. In addition, we benchmark the overhead and performance degradation on the ASIC platform. The results of our error injection simulations and ASIC implementations show the suitability of the proposed approaches for a wide range of applications including deeply embedded systems.

CCS Concepts: • **Hardware** → *Application specific integrated circuits*; *Hardware reliability screening*;

Additional Key Words and Phrases: Application-specific integrated circuit (ASIC), low complexity, reliability, tweakable enciphering schemes

ACM Reference format:
Mehran Mozaffari-Kermani, Reza Azarderakhsh, Ausmita Sarker, and Amir Jalali. 2018. Efficient and Reliable Error Detection Architectures of Hash-Counter-Hash Tweakable Enciphering Schemes. *ACM Trans. Embed. Comput. Syst.* 17, 2, Article 54 (January 2018), 19 pages.
https://doi.org/10.1145/3159173

1 INTRODUCTION

Tweakable enciphering schemes (TES) have been used in sensitive usage models through length preservation of the resulting ciphertext which is not only the function of the plaintext but also

This work has been supported by the U.S. federal agency award 60NANB16D245 granted from U.S. Department of Commerce, National Institute of Standards and Technology (NIST) to Mehran Mozaffari Kermani and Reza Azarderakhsh.
Authors' addresses: M. Mozaffari-Kermani and A. Sarker, 4202 E. Fowler Avenue, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, emails: mehran2@usf.edu, asarker@mail.usf.edu; R. Azarderakhsh and A. Jalali, 777 Glades Road EE 403, Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431-0991; emails: {razarderakhsh, ajalali2016}@fau.edu.

https://doi.org/10.1145/3159173

that of a "tweak." Bandwidth-efficient network protocols and security-retrofitting of old communication protocols are among the sensitive applications utilizing TES. Applications such as disk-sector encryption (in which sectors can be used as tweaks, where in Halevi et al. (2003), it has been proposed that the tweaks do not have to be stored explicitly) utilize such length preservation characteristic to achieve efficient data confidentiality as detailed in the IEEE Security in Storage Working Group (SISWG) P1619 (SISWG 2017).

For an n-bit block cipher E with the key space κ defined as $E : \kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, E enciphers n strings. Block cipher modes of operation (such as TES) provide security for confidentiality-providing block ciphers. Modes of operation extend the notion of block ciphers to call block ciphers multiple times to encrypt long messages through length preservation. It is noted that TES behaves as pseudorandom permutation, which is used extensively to provide security for low-level disk encryption, introducing variability in ciphertexts through tweaks. We also note that tweaks are public quantities used in the state-of-the-art to increase variability of block cipher outputs (see, for instance, Liskov et al. (2002)). The tweak in a TES serves the same purpose of increasing variability in the ciphertext.

A number of TES constructions have been introduced in the literature (for instance, Chakraborty and Sarkar (2008), Halevi (2004, 2007), Halevi and Rogaway (2004), Chakraborty et al. (2013, 2017), Mancillas-Lopez et al. (2010), Sarkar (2009), Bhaumik and Nandi (2015), and Peyrin and Seurin (2016)). Among the three types of TES (some of which constitute a universal hash function, i.e., a Wegman-Carter type polynomial hash), we focus on one variant (Hash-Counter-Hash) which involves multiplications in finite fields whose implementations have been the center of research attention. For instance, and to name a few, HCH, HCTR, and XCB fall under the Hash-Counter-Hash type. It is noted that it invokes two polynomial hash functions and a layer of encryption in-between. It includes not only block cipher calls but also finite field multiplications (one block cipher call and two $GF(2^n)$ field multiplications per message block).

Taking the Advanced Encryption Standard (AES) into account, there have been prominent efforts to make the structures of cryptographic algorithms more compact, e.g., expanded over an area of 2,400 gate equivalent (Moradi et al. 2011). Lightweight block ciphers such as KATAN and KTANTAN (Cannière et al. 2009), PICCOLO (Shibutani et al. 2011), SEA (Standaert et al. 2006), LED (Guo et al. 2011), lightweight efforts for CLEFIA (Beaulieu et al. 2015), and Simon and Speck (Beaulieu et al. 2013) are among the other state-of-the-art to reach lightweight structures. Nevertheless, error detection architectures to thwart natural and malicious faults need to be also used for such primitives.

Not only the error detection constructions need to be effective in terms of error coverage (the ratio of the number of detected errors and the entire number of injections) but the resulting architectures and the added footprints need to be practically applicable in different contexts. Concurrent error detection (CED) techniques (in which the errors are detected and the design functions as normal concurrently to avoid disruption), e.g., hardware redundancy (adding additional hardware to detect errors through duplication), information redundancy (redundancy in information to detect errors, e.g., through error detecting codes), time redundancy (recompute to detect errors), and hybrid redundancy (using inverses of the algorithms or those of their sub-parts to get to the original input and compare), have been widely used to architect reliable hardware for the AES and other cryptographic algorithms (Yen and Wu 2006; Di Natale et al. 2009; Mozaffari-Kermani and Reyhani-Masoleh 2008, 2010; Mozaffari-Kermani and Azarderakhsh 2013, 2015; Maistri and Leveugle 2008; Guo et al. 2015; Yasin et al. 2015; Karaklajic et al. 2013; Mozaffari-Kermani et al. 2014, 2016; Bayat-Sarmadi et al. 2010; Guo and Karri 2013).

We note that we base our constructions on the polynomial hashing structures; nevertheless, other structures such as BRW polynomials (the work in Bernstein (2007) based on earlier work

by Rabin and Winograd (1972)) can be utilized. In this article, we present error detection architectures that are oblivious of the underlying finite field multiplications. This is one of the major contributions of this article as it allows using different constructions for the two $GF(2^n)$ field multiplications used per message block. This not only gives freedom in choosing the developed implementations based on overhead tolerance, but can be tailored for different usage models. We also note that error detection in cryptographic architectures is of paramount importance because of two main complications that can arise: (a) Natural and malicious errors can simply cause malfunctioning and erroneous output, making the architectures of no use; and (b) malicious faults can be injected in order to derive the secrets through side-channel information leaked through comparing the original, fault-free output and the error-prone one.

With regard to the above, we propose parallel constructions through which not only transient faults are detected but permanent faults can be pinpointed as well. This is a characteristic that regular time redundancy schemes do not have. Burst and biased fault models are also considered in our simulations. Our contributions in this article are summarized as follows:

- We present customizable swapped entries for a recomputation (CSER) scheme for error detection. In this scheme, swapping is performed in order to be able to detect both transient and permanent faults (the former is used in fault attacks and the latter can happen in VLSI architectures). It is customizable as we do not have to get confined to swapping specific entries in the recomputation round to eventually decode and compare. We propose reliable hardware architectures for TES based on Hash-Counter-Hash constructions. This choice has been motivated by more involved constructions for this variant compared to the ones which do not have finite field multiplications through polynomial hashing. Obliviousness of the underlying multiplication architectures is one of the main features of the proposed efficient error detection scheme.
- Noting that intelligent attackers would do better than injecting random faults, the proposed architectures are benchmarked for the ability to detect transient and permanent faults by performing fault injection simulations through biased and burst fault models. The results of our error simulations show high error coverage for such TES constructions.
- Finally, we implement the architectures on the application-specific integrated circuit (ASIC) platform to compare the performance and implementation metrics with the original TES constructions. The results show that the proposed designs have acceptable overheads with very high error coverage. The area, delay, and throughput overheads/degradations are also proven to be acceptable.

The rest of the article is organized as follows: In Section 2, preliminaries are presented. Section 3 presents our proposed design for reliable architectures. In Section 4, fault injection simulations are performed to determine the error detection capabilities of the proposed architectures. Moreover, we benchmark our presented work by implementing our designs on ASIC in this section. Finally, conclusions are made in Section 5.

2 PRELIMINARIES

We present a brief description of our notations in what follows, noting that more details on the Hash-Counter-Hash TES variant are presented in the next section. Moreover, we present preliminaries of the schemes for error detection.

2.1 Notations and Brief Details on the Hash-Counter-Hash TES Variant

For an n-bit block cipher E with the key space κ (which cannot be the null space) defined as $E : \kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, E enciphers n strings. A TES is a function such that $E : \kappa \times \tau \times \mu \rightarrow \mu$

where τ, μ are the tweak and message spaces, respectively. The TES encryption can be denoted by ${E^T}_K(.)$ and very similar to block ciphers, here and for the TES, we have the decryption denoted by ${D^T}_K(.)$, where if $\beta = {E^T}_K(\varpi)$, then $\varpi = {D^T}_K(\beta)$.

Depending on the block ciphers used, the sizes of the relevant spaces in the TES would be different. Although the AES has three key sizes with the fixed datapath of 128 bits, other block ciphers vary in datapath length, e.g., KATAN and KTANTAN (Cannière et al. 2009), PICCOLO (Shibutani et al. 2011), SEA (Standaert et al. 2006), LED (Guo et al. 2011), lightweight efforts for CLEFIA (Beaulieu et al. 2015), and Simon and Speck (Beaulieu et al. 2013). For specific applications, the message space is defined as $\mu = \{0,1\}^{mn}$ with m, n being the block cipher datapath length and the number of plaintexts to be processed, respectively.

As we mentioned, in polynomial hashing, in addition to the block cipher key, we use a hash subkey to be processed by field multiplications. We note that for the datapath of size 128 bits, one would use field multiplications in $GF(2^{128})$ using irreducible polynomials of degree 128, e.g., $f(x) = x^{128} + x^7 + x^2 + x + 1$.

Finally, let us briefly remark on the cost of computations through polynomial hashing. Depending on the specifics of the inherent algorithm, the general computation $\Delta_{poly} = m_1 \times H^{n-1} + m_2 \times H^{n-2} + m_3 \times H^{n-3} + \cdots + m_{n-1} \times H + m_n$ is performed (note that this would change slightly for different TES types/modes, and the pluses denote modulo-2 addition). For such construction, we can use Horner's rule which leads to n multiplications and a number of XOR gates.

2.2 Schemes for Error Detection

Schemes for error detection can be classified into four types, i.e., hardware, time, information, and hybrid redundancy.

- —Hardware redundancy is based on duplicating the function and detecting faults by comparing the outputs of two copies.
- —In time redundancy, the function is computed twice on the same input and the results are compared. A variation of time redundancy, double-data-rate (DDR), computes the function on both clock edges to speed up the computation. We note that during the recomputation, the entries could be encoded (e.g., shifted), using different types of hardware to compute the data in the computation and recomputation, detecting both transient and permanent faults.
- —Error detecting codes, such as parity, systematic robust codes, or cyclic redundancy check, are used in the third type, i.e., information redundancy.
- —In hybrid redundancy, an operation, round, or the entire algorithm is followed by the inverse counterparts, and the result is compared with the input. Sub-expression sharing and pipelining could reduce the overhead in this scheme.

3 PROPOSED RELIABLE SCHEMES

In this section, we present the proposed CSER error detection schemes for the Hash-Counter-Hash TES variant. In this type of TES, the underlying operations are multiplications in finite fields. Let us present the proposed error detection constructions in detail as follows. We note that such TES constructions use a variant of the Wegman-Carter polynomial hash combined with a counter mode of operation.

3.1 Error Detection Constructions of Hash-Counter-Hash: Mode HCH

The proposed CSER error detection schemes for HCH (we also note that this algorithm can be modified for the specific application of disk encryption and have thus named it in some works as

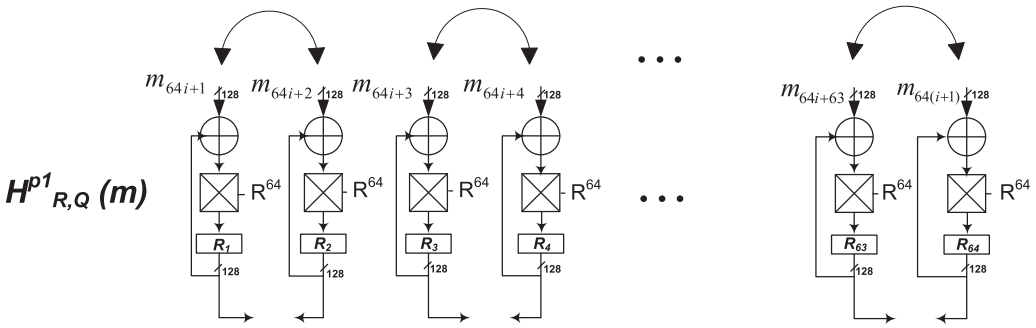

Fig. 1. Proposed error detection approach for $H^{p1}_{R,Q}(m)$ of HCH through swapping the adjacent entries.

HCHfp; however, we use the general name of HCH for brevity) involve parallelization of the hardware implementations of the above formulation. One of the most important aspects of the proposed scheme is that it is not dependent on the approach for performing the underlying finite field multiplications, e.g., bit-serial, bit-parallel, digit-level, composite-field based, Karatsuba-Ofman (KO) with different levels, and the like. In other words, merely proposing signature-based schemes, e.g., parity prediction, not only confines us to a given multiplier implementation and thus limits the applicability of the entire TES to specific usage models with given performance and implementation objectives, but it is also geared toward random error detection which, as mentioned before, is not an aggressive approach to model intelligent attackers.

The hash function $H_{R,Q}(m) = Q + m_1 \times R^{\pi-1} + m_2 \times R^{\pi-2} + \cdots + m_\pi \times R$ can be written in parallel assuming q branches of parallel constructions as

$$H_{R,Q}(m) = Q + m_1 \times \underbrace{R^q \times \cdots \times R^q}_{\frac{\pi}{q}-1\ times} \times R^{q-1} + \cdots + m_k \times \underbrace{R^q \times \cdots \times R^q}_{\frac{\pi}{q}-1\ times} \times R^{q-k} + \cdots +$$

$$m_{q-1} \times \underbrace{R^q \times \cdots \times R^q}_{\frac{\pi}{q}-1\ times} \times R + m_q \times \underbrace{R^q \times \cdots \times R^q}_{\frac{\pi}{q}-1\ times} + m_{q+1} \times \underbrace{R^q \times \cdots \times R^q}_{\frac{\pi}{q}-2\ times} \times R^{q-1} + \cdots + m_\pi \times R. \quad (1)$$

Although different numbers of parallel branches q can be used here, if the powers of $R = E_K(T)$ can be implemented efficiently, one could decrease the complexity of constructions; thus, it is advised that we use powers of 2 (exponentiation through squaring). Such exponentiations lead to low-complexity finite field realizations (squaring can be implemented with low complexity using field multiplications). The proposed CSER error detection schemes act on the three-phase parallel computations realized above.

We have shown two variants for error detection in Figures 1 and 2 for the case study of $q = 64$ parallel branches. As seen in these two figures, 64 parallel branches are built using finite field multiplications and a feedback structure to derive the output of this phase. Before presenting different variants of the proposed CSER, we would like to present different fault models we pledge to detect through the proposed remedies. First, we would detect both single and multiple stuck-at faults. These are detected with high error coverage as assessed through our mounted error injection simulations presented in this article. In addition, we consider adjacent burst faults, occurring in the adjacent field multipliers. We also consider biased fault models which are used in a subset of fault attacks. Finally, we detect both transient and permanent faults. As seen in the following, depending on the error coverage achieved for the latter models, the overhead (or performance degradation) of the proposed CSER can be tailored.

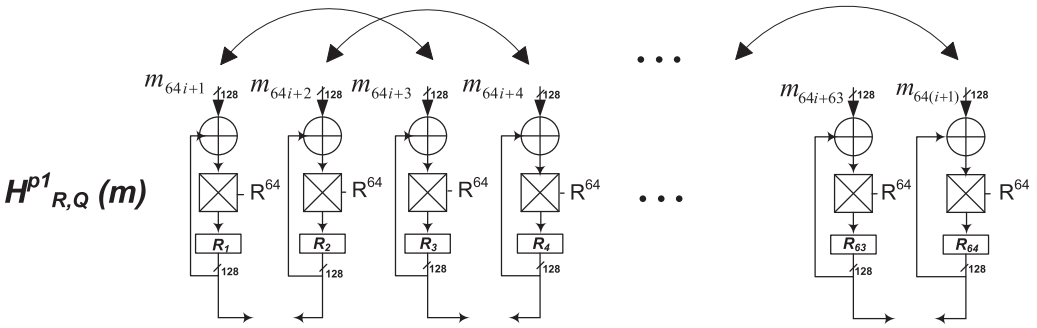

Fig. 2. Proposed error detection approach for $H^{p1}_{R,Q}(m)$ of HCH through swapping the interleaved entries.

In Figure 1, the proposed CSER has been applied to two adjacent branches, and in Figure 2, the scheme is applied to interleaved branches. These are two examples of applying the proposed scheme. Moreover, we note that not only the number of branches can be changed for having a compromise between area/power and performance, but the way we swap the entries can be tailored based on the fault models considered (hence the name CSER is much more suitable for the proposed schemes). For instance, in Figure 1, the faults in individual multipliers are detected (both permanent and transient faults as we swap the entries and compare the results of two runs without the need to perform decoding which will result in lower overhead). Moreover, as we show through error injection simulations, we cover both single and multiple faults. However, if adjacent multipliers are affected (burst or targeted or biased faults), one may use the construction proposed in Figure 2 for error detection so that swapping is done for interleaved branches. We would like to emphasize that being customizable is one of the main characteristics of the proposed scheme, i.e., depending on the usage model, one could use different variants of the presented CSER.

Let us denote the above architectures for Phase 1 (out of three phases). Let us present these three phases in detail. In Phase 1 (denoted by p1), we perform

$$
\begin{aligned}
H^{p1}_{R,Q}(m) = \underbrace{\{\cdots\{\{m_1 \times R^q + m_{q+1}\} \times R^q + m_{2q+1}\} \times R^q + \cdots\}}_{H^{p1}_{R,Q}(m_{iq+1}):\frac{\pi}{q}-1\ times} + \\
\underbrace{\{\cdots\{\{m_2 \times R^q + m_{q+2}\} \times R^q + m_{2q+2}\} \times R^q + \cdots\}}_{H^{p1}_{R,Q}(m_{iq+2}):\frac{\pi}{q}-1\ times} + \\
\underbrace{\{\cdots\{\{m_3 \times R^q + m_{q+3}\} \times R^q + m_{2q+3}\} \times R^q + \cdots\}}_{H^{p1}_{R,Q}(m_{iq+3}):\frac{\pi}{q}-1\ times} + \cdots + \\
\underbrace{\{\cdots\{\{m_{q-1} \times R^q + m_{2q-1}\} \times R^q + m_{3q-1}\} \times R^q + \cdots\}}_{H^{p1}_{R,Q}(m_{iq+q-1}):\frac{\pi}{q}-1\ times} + \\
\underbrace{\{\cdots\{\{m_q \times R^q + m_{2q}\} \times R^q + m_{3q}\} \times R^q + \cdots\}}_{H^{p1}_{R,Q}(m_{(i+1)q}):\frac{\pi}{q}-1\ times}. \qquad (2)
\end{aligned}
$$

We note that the branch results of Phase 1 are entries to Phase 2. As seen from the formulation, the number of iterations depends on the number of input blocks as well as the number of parallel branches q. Increasing the number of parallel branches would result in faster computations and

better throughput at the expense of more area and power consumption (one can fine-tune the architectures for the efficiency and energy consumption suitable for specific usage models). Finally, we note that the number of iterations in Phase 1 is far larger compared to those for combined Phases 2 and 3 as discussed in this section.

We would like to closely consider different variants of our CSER scheme and assess their effectiveness in detecting permanent and transient faults. With respect to Phase 1, let us categorize the scheme into two variants based on the extent of recomputation.

(1) One could perform the CSER scheme after $l < \frac{\pi}{q} - 1$ cycles. In other words, we store the result of the part of Phase 1 performed after l cycles in a register, i.e.,

$$
\begin{aligned}
H^{p1}_{R,Q}(m,l) = & \underbrace{\{\cdots\{\{m_1 \times R^q + m_{q+1}\} \times R^q + m_{2q+1}\} \times R^q + \cdots\}}_{l\ times} + \\
& \underbrace{\{\cdots\{\{m_2 \times R^q + m_{q+2}\} \times R^q + m_{2q+2}\} \times R^q + \cdots\}}_{l\ times} + \\
& \underbrace{\{\cdots\{\{m_3 \times R^q + m_{q+3}\} \times R^q + m_{2q+3}\} \times R^q + \cdots\}}_{l\ times} + \cdots + \\
& \underbrace{\{\cdots\{\{m_q \times R^q + m_{2q}\} \times R^q + m_{3q}\} \times R^q + \cdots\}}_{l\ times}. \qquad (3)
\end{aligned}
$$

Then, after the completion of Phase 1, we perform the recomputation for a fraction of the required number of cycles needed to get the final result, i.e., until the $(l < \frac{\pi}{q} - 1)$th cycle. For instance, following Figure 2 for interleaved swapping, we have the following:

$$
\begin{aligned}
H^{p1}_{R,Q}(m,l) = & \underbrace{\{\cdots\{\{m_3 \times R^q + m_{q+3}\} \times R^q + m_{2q+3}\} \times R^q + \cdots\}}_{l\ times} + \\
& \underbrace{\{\cdots\{\{m_4 \times R^q + m_{q+4}\} \times R^q + m_{2q+4}\} \times R^q + \cdots\}}_{l\ times} + \\
& \underbrace{\{\cdots\{\{m_1 \times R^q + m_{q+1}\} \times R^q + m_{2q+1}\} \times R^q + \cdots\}}_{l\ times} + \cdots + \\
& \underbrace{\{\cdots\{\{m_{q-2} \times R^q + m_{2q-2}\} \times R^q + m_{3q-2}\} \times R^q + \cdots\}}_{l\ times}. \qquad (4)
\end{aligned}
$$

Through simulations, we have confirmed that permanent and long transient faults (single, biased/burst, and multiple) can be detected with very high coverage with the tested l cycles. Moreover, the merit here is that throughput degradation is not as severe as recomputing all the way. In other words, through simulations, we have derived that even for a small fraction of cycles, i.e., $l \ll \frac{\pi}{q} - 1$ cycles, we reach the error coverage of close to 100%. However, the major drawback here is that it is possible that we get transient faults (some natural faults and, especially, malicious faults are of such nature) which cannot be detected in some cases. Depending on the performance and implementation metrics overhead/degradation, one might utilize this variant; however, as mentioned, there are loopholes left for the attackers this way if such throughput gain is preferred. We also note that we do not need to decode the output of CSER and that is a major benefit.

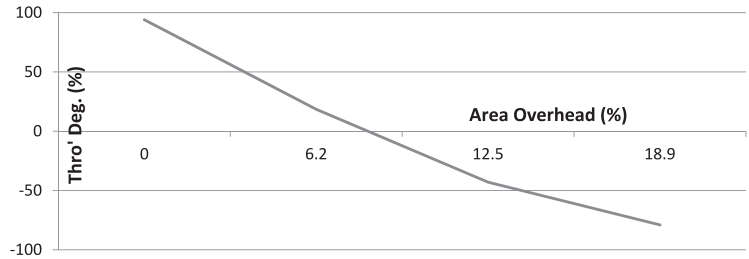

Fig. 3. An example of throughput alleviation for up to three stages of pipeline registers.

(2) In the second variant, we perform the CSER scheme for the entire Phase 1, i.e., we get the following, respectively, for the two runs:

$$H_{R,Q}^{p1}\left(m,\frac{\pi}{q}-1\right)=\underbrace{\{\cdots\{\{m_1\times R^q+m_{q+1}\}\times R^q+m_{2q+1}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+$$
$$\underbrace{\{\cdots\{\{m_2\times R^q+m_{q+2}\}\times R^q+m_{2q+2}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+$$
$$\underbrace{\{\cdots\{\{m_3\times R^q+m_{q+3}\}\times R^q+m_{2q+3}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+\cdots+$$
$$\underbrace{\{\cdots\{\{m_q\times R^q+m_{2q}\}\times R^q+m_{3q}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}, \tag{5}$$

and

$$H_{R,Q}^{p1}\left(m,\frac{\pi}{q}-1\right)=\underbrace{\{\cdots\{\{m_3\times R^q+m_{q+3}\}\times R^q+m_{2q+3}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+$$
$$\underbrace{\{\cdots\{\{m_4\times R^q+m_{q+4}\}\times R^q+m_{2q+4}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+$$
$$\underbrace{\{\cdots\{\{m_1\times R^q+m_{q+1}\}\times R^q+m_{2q+1}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}+\cdots+$$
$$\underbrace{\{\cdots\{\{m_{q-2}\times R^q+m_{2q-2}\}\times R^q+m_{3q-2}\}\times R^q+\cdots\}}_{\frac{\pi}{q}-1\ times}. \tag{6}$$

At the expense of throughput degradation, the error detection is extended in this variant to transient faults in the last $\frac{\pi}{q}-1-l$ cycles as well. We note that it is possible that we sub-pipeline the finite field multipliers in Figures 1 and 2, for instance into two stages, so that we increase the working frequencies and decrease the throughput degradations. This remedy would be at the expense of added area for the pipeline registers. We have presented Figure 3 to show that starting with no-pipeline stage to three pipeline stages, the area overhead increases for a given finite field multiplier, resulting in throughput alleviations. We note that although negative degradations can be achieved, the latency of the architectures increases as we add more stages which might not be acceptable for some applications.

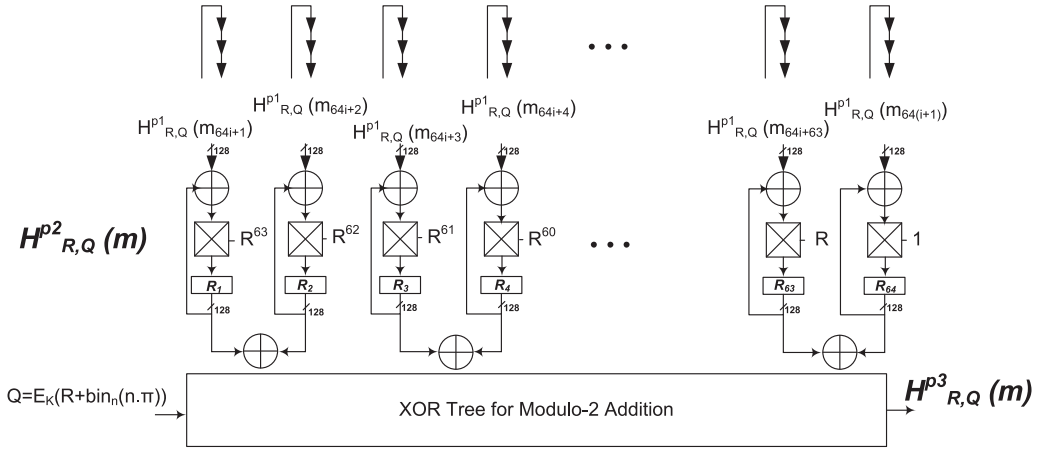

Fig. 4. Proposed error detection approach for Phases 2 and 3 of HCH, i.e., $H^{p2}_{R,Q}(m)$ and $H^{p3}_{R,Q}(m)$.

The output of Phase 1, i.e., $H^{p1}_{R,Q}(m)$, is used as follows in Phase 2 (denoted by p2):

$$H^{p2}_{R,Q}(m) = H^{p1}_{R,Q}(m_{iq+1}) \times R^{q-1} + H^{p1}_{R,Q}(m_{iq+2}) \times R^{q-2} + H^{p1}_{R,Q}(m_{iq+3}) \times R^{q-3} + \cdots + H^{p1}_{R,Q}(m_{iq+q-1}) \times R + H^{p1}_{R,Q}(m_{(i+1)q}). \quad (7)$$

There have been previous works (Mozaffari Kermani et al. 2012; Satoh et al. 2009) for performing efficient constructions for the counterpart of this phase in Galois Counter Mode (GCM). Basically, either the subkey exponentiations in GCM are calculated as is, or they are derived through the underlying low-cost exponentiations for hash subkey to powers of 2 (we do not differentiate these two cases in this article). Both such cases constitute only a very small fraction of the iterations needed to compute HCH. Finally, in Phase 3 (denoted by p3), we have $H^{p3}_{R,Q}(m) = Q + H^{p2}_{R,Q}(m)$. Figure 4 shows the proposed CSER scheme for Phases 2 and 3. As seen in this figure, we perform the recomputation without swapping the entries in these two phases (shown by multiple consecutive vertical arrows on top of Figure 4). These two phases constitute a very small fraction of cycles needed for the entire output derivation. Thus, not only simple recomputation adds very slightly to performance overhead but because we detect the permanent faults in the previous phase, not swapping the entries does not deteriorate the "quality" of error detection. We also use a counter mode of operation in HCH. For instance, for an n-bit string denoted by S, a key K and π blocks of plaintexts and ciphertexts, $A_1 - A_\pi$, the counter mode is defined as $Ctr_{K,S}(A_1 - A_\pi) = (A_1 + E_K(S + bin_n(1)), \ldots, A_\pi + E_K(S + bin_n(\pi)))$.

In this mode of TES, a specific method is used for handling the tweak. Moreover, an extra encryption operation is used before the counter mode compared to the HCTR mode. For the message blocks $m_1 - m_\pi$, we have the hash function $H_{R,Q}(m) = Q + m_1 \times R^{\pi-1} + m_2 \times R^{\pi-2} + \cdots + m_\pi \times R$. We note that given a block cipher, R, Q are defined as $R = E_K(T)$ and $Q = E_K(R + bin_n(n\pi))$. Moreover, multiple finite field multiplications are needed as presented above to reach the final result.

Let us finalize this section by practically considering two extremes of implementing the architectures. Our scheme is not confined to only polynomial with repeated exponentiations or only Horner's rule. Let us have an example for $q = 2$ in which the presented architectures would be converted to just two parallel branches (toward Horner's), a mix of both schemes. At the

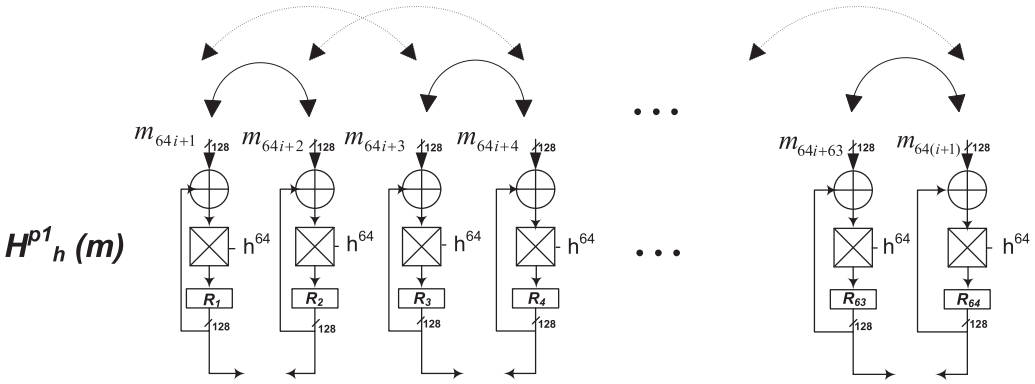

Fig. 5. Proposed error detection approach for Phase 1 of HCTR, i.e., $H_h^{p1}(m)$ (note that we have shown the first $\frac{\pi}{q}-1$ cycles, leaving one cycle in the two leftmost branches for Phase 2).

expense of losing throughput and performance, we decrease the area of the constructions, noting that the eventual architecture depends on the applications. For the applications which need high-performance architectures, moving toward repeated exponentiations would lead to parallel constructions for increasing the throughput; however, for low-area and lightweight applications, the proposed scheme can be fine-tuned toward Horner's rule. We would like to emphasize that a compromise of these two extremes is usually preferred to achieve the respective efficiency required (which takes into account both the area and the throughput). Finally, the formulations for $q = 2$ for Phase 1 would be

$$H_{R,Q}^{p1}(m) = \underbrace{\{\cdots\{\{m_1 \times R^2 + m_3\} \times R^2 + m_5\} \times R^2 + \cdots\}}_{H_{R,Q}^{p1}(m_{2i+1}):\frac{\pi}{2}-1\ times} +$$
$$\underbrace{\{\cdots\{\{m_2 \times R^2 + m_4\} \times R^2 + m_6\} \times R^2 + \cdots\}}_{H_{R,Q}^{p1}(m_{2i+2}):\frac{\pi}{2}-1\ times}.$$

3.2 Error Detection Constructions of Hash-Counter-Hash: Modes HCTR and XCB

In what follows, we present the error detection schemes for two modes of Hash-Counter-Hash, i.e., modes HCTR and XCB.

For the HCTR mode, we propose using a two-phase (Phase 1: p1 and Phase 2: p2) error detection scheme based on CSER. Assuming q branches of parallel constructions, we write the computations in both phases:

$$H_h(m) = m_1 \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}\ times} \times h + \cdots + m_k \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} \times h^{q-k+2} + \cdots + m_{q+1} \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} \times h +$$
$$m_{q+2} \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} + \cdots + m_\pi \times h^2 + bin_n(|\ m\ |) \times h. \quad (8)$$

In Phase 1, the two sets of entries for the runs can be asserted as follows (we have shown both adjacent and interleaved swapping in Figure 5; however, in what follows, we only present that of interleaved swapping for the sake of brevity). One can either compare the register contents (a) after l cycles or (b) all the way after $\frac{\pi}{q}-1$ cycles. The former would be acceptable for permanent and long transient faults (single, biased/burst, and multiple) but cannot detect specific shorter transient

faults (which are detected through the latter approach). However, the former could achieve very low throughput degradation, whereas the latter needs deep sub-pipelining for that which increases the latency. In the first run and for the first $\frac{\pi}{q} - 1$ cycles, we have

$$H_h^{p1}(m) = \underbrace{\{\cdots\{\{m_1 \times h^q + m_{q+1}\} \times h^q + m_{2q+1}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+1}): \frac{\pi}{q}-1\ times} + \underbrace{\{\cdots\{\{m_2 \times h^q + m_{q+2}\} \times h^q + m_{2q+2}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+2}): \frac{\pi}{q}-1\ times} + \underbrace{\{\cdots\{\{m_3 \times h^q + m_{q+3}\} \times h^q + m_{2q+3}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+3}): \frac{\pi}{q}-1\ times} + \cdots + \underbrace{\{\cdots\{\{m_{q-1} \times h^q + m_{2q-1}\} \times h^q + m_{3q-1}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+q-1}): \frac{\pi}{q}-1\ times} + \underbrace{\{\cdots\{\{m_q \times h^q + m_{2q}\} \times h^q + m_{3q}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{(i+1)q}): \frac{\pi}{q}-1\ times}. \quad (9)$$

Considering $l < \frac{\pi}{q} - 1$ and for the interleaved swapping variant, we have

$$H_h^{p1}(m) = \underbrace{\{\cdots\{\{m_3 \times h^q + m_{q+3}\} \times h^q + m_{2q+3}\} \times h^q + \cdots\}}_{l\ times} + \underbrace{\{\cdots\{\{m_4 \times h^q + m_{q+4}\} \times h^q + m_{2q+4}\} \times h^q + \cdots\}}_{l\ times} + \underbrace{\{\cdots\{\{m_1 \times h^q + m_{q+1}\} \times h^q + m_{2q+1}\} \times h^q + \cdots\}}_{l\ times} + \cdots + \underbrace{\{\cdots\{\{m_{q-3} \times h^q + m_{2q-3}\} \times h^q + m_{3q-3}\} \times h^q + \cdots\}}_{l\ times} + \underbrace{\{\cdots\{\{m_{q-2} \times h^q + m_{2q-2}\} \times h^q + m_{3q-2}\} \times h^q + \cdots\}}_{l\ times}. \quad (10)$$

We present Figure 6 for the error detection of HCTR in Phase 2 for $q = 64$. As seen in this figure, the output of Phase 1, i.e., $H_h^{p1}(m)$, is used as follows in Phase 2 (denoted by p2).

$$H_h^{p2}(m) = H_h^{p1}(m_{iq+1}) \times h^q \times h + H_h^{p1}(m_{iq+2}) \times h^q + H_h^{p1}(m_{iq+3}) \times h^{q-1} + \cdots + H_h^{p1}(m_{iq+q-1}) \times h^3 + H_h^{p1}(m_{(i+1)q}) \times h^2. \quad (11)$$

Unlike the previous mode of TES, here, we have one phase (Phase 2) as seen in Figure 6. Recomputations are shown by multiple consecutive vertical arrows on top of Figure 6. We note that this phase constitutes a very small fraction of cycles needed for the the entire output derivation.

In HCTR, apart from the pre- and post-computation operations, the main operation is based on finite field multiplications, described above. Such construction is defined as $H_h(m) = m_1 \times h^{\pi+1} + m_2 \times h^{\pi} + \cdots + m_{\pi} \times h^2 + bin_n(|\ m\ |) \times h$, where h is the hash subkey mentioned before, and $bin_n(|\ m\ |)$ is the n-bit binary representation of length of m.

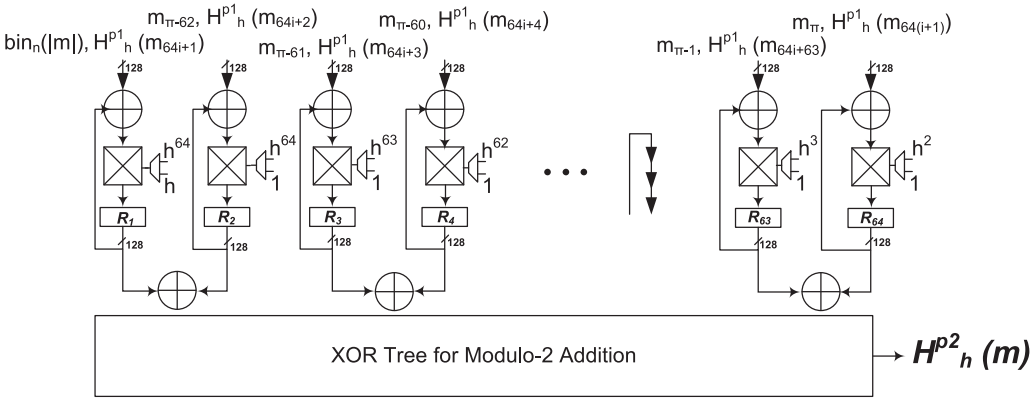

Fig. 6. Proposed error detection approach for Phase 2 of HCTR, i.e., $H_h^{p2}(m)$.

Finally, for the XCB mode, for q branches of parallel constructions, we have:

$$h_h(m,T) = m_1 \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}\ times} \times h^2 + \cdots + m_k \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} \times h^{q-k+3} + \cdots + m_{q+1} \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} \times h^2$$
$$+ m_{q+2} \times \underbrace{h^q \times \cdots \times h^q}_{\frac{\pi}{q}-1\ times} \times h + \cdots + m_\pi \times h^3 + T \times h^2 + (bin_{n/2}(|\ P\ |) \parallel bin_{n/2}(|\ T\ |)) \times h. \quad (12)$$

The proposed error detection for XCB has two phases as follows. In Phase 1 (whose figure is not shown for the sake of brevity), the entries for the two runs are asserted, where for the normal operation for the first $\frac{\pi}{q} - 1$ cycles we have

$$h_h^{p1}(m,T) = \underbrace{\{\cdots\{\{m_1 \times h^q + m_{q+1}\} \times h^q + m_{2q+1}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+1},T):\frac{\pi}{q}-1\ times} +$$
$$\underbrace{\{\cdots\{\{m_2 \times h^q + m_{q+2}\} \times h^q + m_{2q+2}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+2},T):\frac{\pi}{q}-1\ times} +$$
$$\underbrace{\{\cdots\{\{m_3 \times h^q + m_{q+3}\} \times h^q + m_{2q+3}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+3},T):\frac{\pi}{q}-1\ times} + \cdots +$$
$$\underbrace{\{\cdots\{\{m_{q-1} \times h^q + m_{2q-1}\} \times h^q + m_{3q-1}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{iq+q-1},T):\frac{\pi}{q}-1\ times} +$$
$$\underbrace{\{\cdots\{\{m_q \times h^q + m_{2q}\} \times h^q + m_{3q}\} \times h^q + \cdots\}}_{H_h^{p1}(m_{(i+1)q},T):\frac{\pi}{q}-1\ times}. \quad (13)$$

Moreover, considering $l < \frac{\pi}{q} - 1$ and for interleaved swapping, we reach similar entries as for the HCTR. Furthermore, the two choices are to compare the register contents after l cycles or after $\frac{\pi}{q} - 1$ cycles. In the second phase, as shown in Figure 7 for $q = 64$, we have

$$H_h^{p2}(m,T) = H_h^{p1}(m_{iq+1},T) \times h^q \times h^2 + H_h^{p1}(m_{iq+2},T) \times h^q \times h + H_h^{p1}(m_{iq+3},T) \times h^q +$$
$$\cdots + H_h^{p1}(m_{iq+q-1},T) \times h^4 + H_h^{p1}(m_{(i+1)q},T) \times h^3. \quad (14)$$

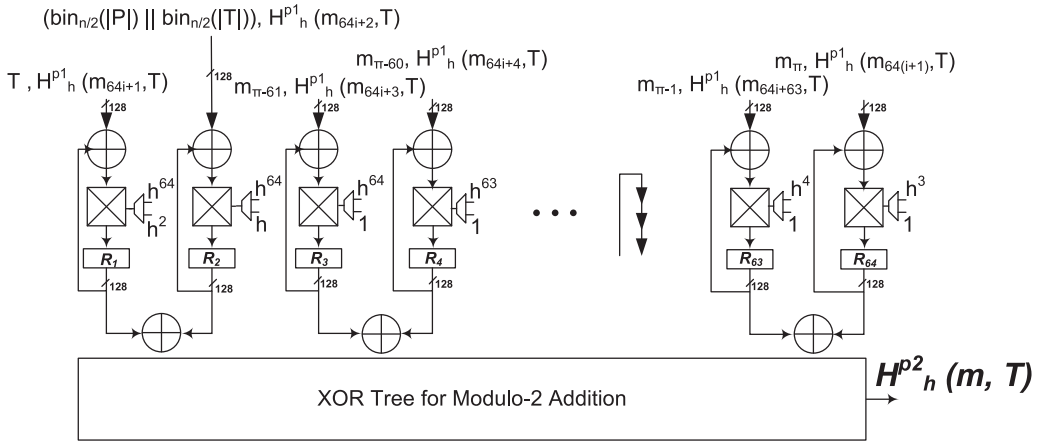

Fig. 7. Proposed error detection approach for Phase 2 of XCB, i.e., $H_h^{p2}(m, T)$.

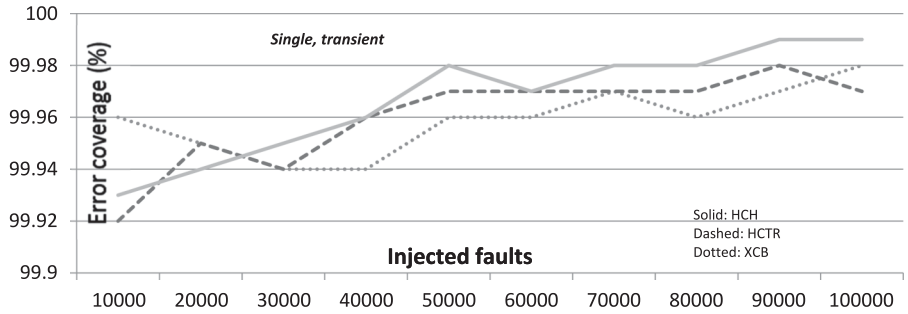

Fig. 8. Error coverage of three TES modes for single, transient faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

Apart from minor differences for the counter, the main contrast compared to the other modes is that, here, two hash subkeys are used for the computations through finite field multiplications. $h_h(m, T)$ is the polynomial hash variant of this mode and is defined as

$$h_h(m, T) = m_1 \times h^{\pi+2} + m_2 \times h^{\pi+1} + \cdots + m_\pi \times h^3 + T \times h^2 + (bin_{n/2}(|\ P\ |) \parallel bin_{n/2}(|\ T\ |)) \times h, \quad (15)$$

where T is an n-bit string.

4 LFSR-BASED ERROR INJECTION SIMULATIONS AND ASIC IMPLEMENTATIONS

The proposed error detection architectures for TES have been simulated after injecting faults. It is noted that for each of three modes (HCH, HCTR, and XCB), we have considered the most involved operations, i.e., polynomial hashing, for the case study of $q = 64$. The proposed architectures have the capability of detecting both permanent and transient faults (this covers both natural and malicious faults), and it has been assessed in our error injection simulations. For transient fault injection, simulations are done in two stages. First, we have injected the faults in just the first round. Then, we have injected the faults in just the second round (hence, the faults are considered as transient although a sub-set of such cases, i.e., long transient faults, are also covered here).

For simulations (the results are shown in Figures 8–13), VHDL has been used. The fault models used to test the proposed architectures are created using linear-feedback shift registers (LFSRs)

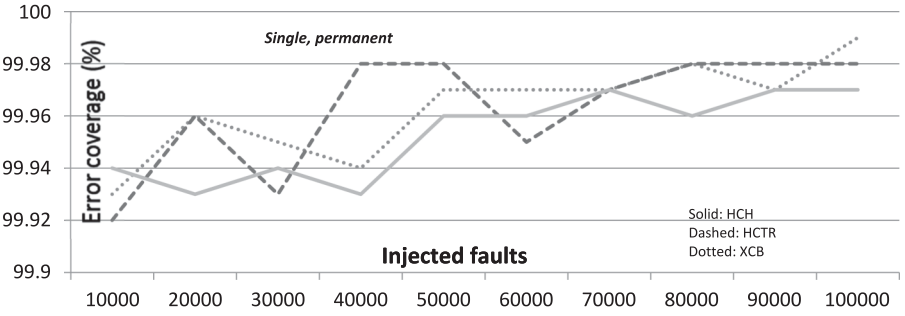

Fig. 9. Error coverage of three TES modes for single permanent faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

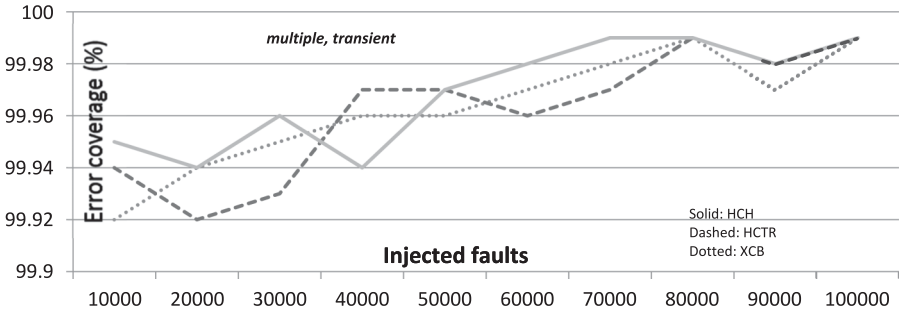

Fig. 10. Error coverage of three TES modes for multiple transient faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

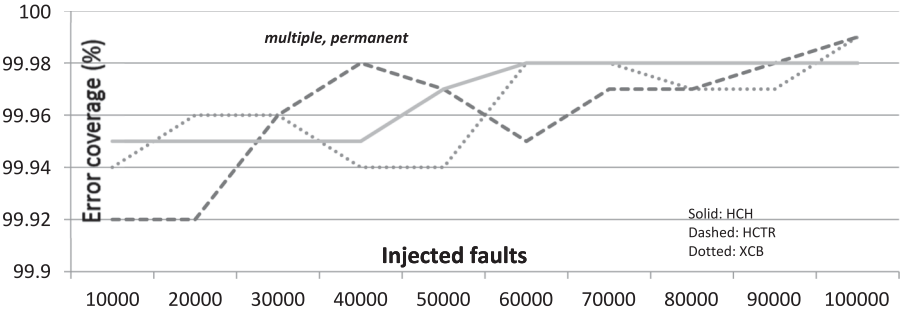

Fig. 11. Error coverage of three TES modes for multiple permanent faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

to generate pseudo-random fault vectors that can flip random bits in the output of the gates (and at random intervals). The j-bit LFSRs used here are j-bit registers with polynomials for maximum taps. This is achieved using multiplexers whose select signals are driven using LFSRs, thus randomizing the selection of faulty bits (coming from other LFSRs) and correct bit, i.e., the actual results. We have employed an LFSR to randomize the position and the value of the injected faults (used for multiple fault injection). As mentioned before, an intelligent attacker fault model is far more than just multiple random faults. Thus, we have considered single, multiple, and biased faults. We note that for biased faults, we have chosen to inject faults in the six leftmost branches of the 64-branch polynomial hashing architecture. Nevertheless, the proposed error detection schemes considering

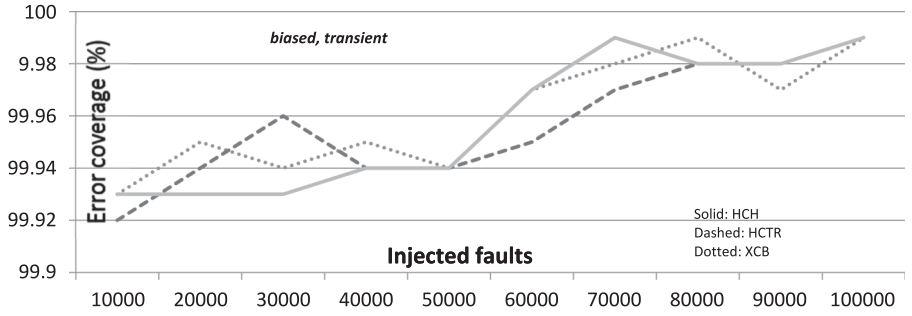

Fig. 12. Error coverage of three TES modes for biased transient faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

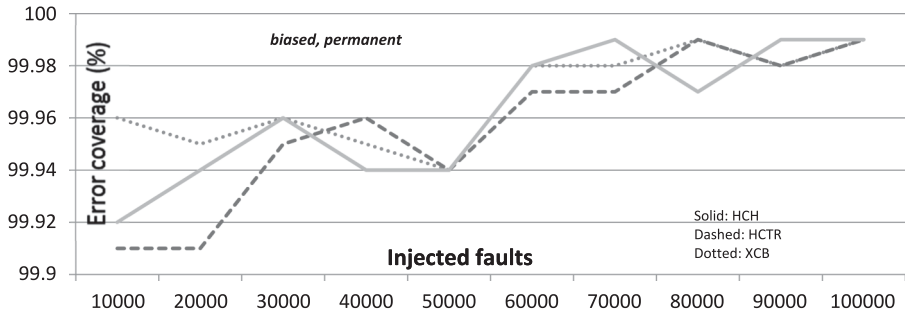

Fig. 13. Error coverage of three TES modes for biase, permanent faults (solid line: HCH; dashed line: HCTR; and dotted line: XCB) for up to 100,000 injections.

burst and biased faults are not confined to such selection, and our proposed approaches are capable of detecting the respective biased faults with counterpart models as well.

We have considered deep sub-pipelining the architectures and applying the proposed schemes for the entire operation (in Phase 1, we have $\frac{\pi}{q} - 1$ cycles). This has been performed instead of using the proposed CSER schemes merely for a portion of cycles in Phase 1 (gaining lower degradation in throughput at the expense of being vulnerable to a sub-set of transient faults). Moreover, as mentioned before, Phase 1 constitutes the majority of cycles, e.g., for $q = 64$ and $\pi = 2^{10}$ as a typical entry, depending on the TES mode, we only need two or three cycles for Phase 2 (or Phase 3 when applicable), whereas Phase 1 needs 2^{19} cycles.

Overall, up to 100,000 faults are injected and the error indication flags are observed. A counter is set to count the number of faults detected. We have also tested our proposed architectures using the single-bit fault model (as the most typical natural fault model) and the single-byte fault model (as the most practical fault model for fault attacks). The results of the former and other cases presented in Figures 8–13 do not include the latter case; however, for the latter case, our simulations show that we also get very close to 100% error coverage. One reason that we cannot detect very small percentages of faults is that we inject faults into the original architecture and the comparison XOR gate (final XOR gate which compares the first/second round results) at the same time (this is in analogy with predicted/actual signature comparisons). Such complications can arise from asserting faults to the comparison unit and can deteriorate the error detection scheme status. For biased faults (and in order to make the error detection schemes less effective), the attackers can have, as part of the default fault sets, those affecting the voters. In order to alleviate the status of

Table 1. ASIC 65-nm Benchmark of the Proposed Schemes for $q = 64$ and $\pi = 2^{10}$ Blocks with One-Stage Sub-Pipelining

Polynomial hashing (PH)	Area (kGE)	Overhead	Throughput (Gbps)	Degradation	Efficiency ($\frac{Gbps}{MGE}$)	Degradation
Original PH of HCH	2,645	5.1%	5.125[1]	13.5%	1.94	18.0%
Proposed scheme	2,781		4.434		1.59	
Original PH of HCTR	2,790	4.1%	4.987	12.9%	1.79	16.8%
Proposed scheme	2,903		4.342		1.49	
Original PH of XCB	2,783	3.8%	4.996	10.1%	1.80	13.9%
Proposed scheme	2,890		4.490		1.55	

[1]For our experiments, $\pi = 2^{10}$ has been chosen; nevertheless, if we select just $\pi = 2^7 = 128$ blocks, we achieve throughputs of 24–29Gbps. This would also affect the efficiency of constructions. However, we note that the intention here is to have a fixed original algorithm and benchmark the overheads and degradations.

the schemes in such cases, we need hardening of the voters (which is typically of low cost), e.g., through fault tolerance techniques such as triple modular redundancy.

Let us consider two phases for possible attacks. For "active" phase, where the attacker actually injects faults, the proposed error detection scheme is effective with high error coverage (this is the case for biased faults as well as in the case in which the respective complications are considered for the voter). For the "passive" (or "listen") phase, where the attacker passively listens to the node activities in the architectures to find the most effective attack nodes, let us consider two cases: (i) that the attacker could "listen" to the system before attacking, gathering as much information as possible, and base the attack on the most "active" areas (as one option, to attack the "voter") or the least "active" areas (as another option, could contain, e.g., initialization data). In this former case, the proposed schemes are effective as we protect the entire architecture independent of biasness; however, for the two exceptions of voter and initialization memory, hardening needs to be in place. (ii) The latter case is to mask such side-channel information about the node activities in the circuit, for example, through making the power consumptions in transitions equal, using architecture masking (threshold implementation in hardware) or logic families including dual-rail pre-charge (DRP) logic styles that consume an equal amount of power for every transition of a node in a circuit.

This section also presents the overhead incurred while applying the proposed error detection schemes on the ASIC platform. We have chosen to implement sub-quadratic Karatsuba-Ofman (KO) finite field multiplier (with one step, noting that the proposed scheme is oblivious of the multiplication architecture) as the underlying polynomial hashing operation; nonetheless, such choice does not confine the proposed schemes. In fact, being oblivious of the underlying multipliers is one of the advantages of the proposed schemes. Moreover, the architectures of polynomial hashing, the most involved sub-set for TES modes, have been implemented, noting that we refrain from presenting the error detection schemes for the block ciphers within, as these have been discussed in detail in the previous work. However, the presented benchmark is also oblivious of the block cipher used. The very high error coverage shown in Figures 8–13 is almost consistent across fault models. We also note that the fluctuations seen in these figures eventually get settled to the final values.

The results of our syntheses for the three modes discussed in this article using 65-nm CMOS technology are presented in Table 1. The ASIC 65-nm benchmark of the proposed schemes for $q = 64$ and $\pi = 2^{10}$ blocks with one-stage sub-pipelining is presented in this table. We note that as seen in Table 1, we use 64 parallel constructions for the three variants presented in this article, i.e., 64 multipliers are used in parallel. This could be condensed to, for instance, 32 or 16 multipliers, for lightweight architectures at the expense of deterioration in performance. The architectures have been coded in VHDL as the design entry to the Synopsys Design Compiler.

In synthesis, we have used default optimization efforts for both speed and area for the original and error detection architectures. The hierarchy has been maintained; nevertheless, using directives such as compile_ultra would also be possible if performed across all designs (obliviousness of the synthesis methodology is also hypothesized in our evaluations as we have not based the architectures on any resources instantiated from the library of the technology used in ASIC). We have performed one-stage deep sub-pipelining for the architectures of error detection for the three TES modes (HCH, HCTR, and XCB) proposed in the previous section. Sub-pipelining can be performed through different approaches. In order to make the process closer to ideal, we have also used the "retiming" option of Synopsys to slightly adjust the registers used for performance boost, taking into account both the multiplication and the XOR tree delays to have the entire delay ideally halved. Register retiming performs optimization of sequential logic by moving registers through logic boundaries to optimize timing with minimum area impact for designs that already contain registers. However, even through retiming, this might not be possible. We would like to emphasize that the presented results are independent of the platform, and similar results are expected for other standard-cell ASIC libraries.

To conclude the section, we provide a comparison with the case study of Simon/Speck to evaluate the overheads/degradations of the presented schemes with respect to the ones provided for these lightweight block ciphers. As seen in Table 1, we get 3.8%–5.1% area overhead, 10.1%–13.5% throughput degradation, and 13.9%–18.0% efficiency degradation for the three TES modes. The proposed scheme in Ahir et al. (2017) is based on recomputing with rotated operands for lightweight block ciphers Simon and Speck. For Simon, the overheads for delay, area, and throughput are roughly 3%–13%, 30%, and 3%–11%, respectively. We note that the efficiency degradation is also roughly over 30%. For Speck, the overheads for delay, area, and throughput are roughly 3%–6%, 11%, and 3%–6%, respectively. We note that the efficiency degradation is also roughly 14%–16%. The overhead comparisons show reasonable results as per these benchmarks.

Some important notes are also presented as follows regarding the implementations:

- The performance metrics of both iterative feedback and iterative parallel constructions, specifically throughput and efficiency, depend on the number of blocks processed. In other words, if we use a lower number of blocks processed, we get higher throughput and efficiency, as the meaningful output bits are received at the output in a shorter number of cycles and, thus, amount of total time. For our experiments, $\pi = 2^{10}$ has been chosen; nevertheless, if we select just $\pi = 2^7 = 128$ blocks, we achieve throughputs of 24–29Gbps. This would also affect the efficiency of constructions. However, we note that the intention here is to have a fixed original algorithm and benchmark the overheads and degradations.
- We have used KO multiplier which is sub-quadratic. This has been chosen to benchmark the metrics and can be modified (noting the obliviousness of the architectures of the type of multipliers chosen) to KO with deeper steps, KO with sub-pipelined constructions, or quadratic complexity faster multipliers. We would like to emphasize that such changes would not affect the used schemes, although they would affect the complexity and the performance of the original algorithms as well as the error detection one to lower/higher figures.

With such performance and implementation overheads/degradations, the proposed approaches achieve high error coverage (see Figures 8–13), making the hardware architectures of such variants of TES more reliable.

5 CONCLUSIONS

In this article, we have proposed reliable and efficient error detection architectures for different TES variants. We have proposed efficient approaches for protecting such schemes against natural

and malicious faults (noting that intelligent attackers do not merely get confined to injecting multiple faults, one major benchmark for the proposed schemes is evaluation toward biased and burst fault models). We have valuated a variant of TES, i.e., the Hash-Counter-Hash scheme, which involves polynomial hashing, constituting finite field multiplication which, by far, is the most involved operation in TES. In addition, we have benchmarked the overhead and performance degradation of the proposed architectures on the ASIC platform. The implementation results show that the overheads incurred by the proposed architectures are acceptable. The proposed architectures can be reliably and efficiently used and further tailored by customizing the architectures based upon the requirements in terms of reliability, security, and overhead tolerance. As seen from our benchmarks, we get 3.8%–5.1% area overhead, 10.1%–13.5% throughput degradation, and 13.9%–18.0% efficiency degradation for the three TES modes, achieving high error coverage, making the hardware architectures of such variants of TES more reliable.

REFERENCES

P. Ahir, M. Mozaffari Kermani, and R. Azarderakhsh. 2017. Lightweight architectures for reliable and fault detection Simon and Speck cryptographic algorithms on FPGA. *ACM Transactions on Embedded Computer Systems* 16, 4 (2017), 109:1–109:17.

S. Bayat-Sarmadi, M. Mozaffari Kermani, and A. Reyhani-Masoleh. 2014. Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 7 (2014), 1105–1109.

R. Beaulieu, D. Shors, J. Smith, S. T. Clark, B. Weeks, and L. Wingers. 2013. The Simon and Speck families of block ciphers. In *Proc. Cryptology ePrint Archive*, Report no. 2013/404.

R. Beaulieu, D. Shors, J. Smith, S. T. Clark, B. Weeks, and L. Wingers. 2015. Simon andSpeck: Block ciphers for the internet of things. In *Proc. Cryptology ePrint Archive*, Report no. 2015/585.

D. Bernstein. 2007. Polynomial evaluation and message authentication. Retrieved January 2018 from https://cr.yp.to/antiforgery/pema-20071022.pdf.

R. Bhaumik and M. Nandi. 2015. An inverse-free single-keyed tweakable enciphering scheme. In *Proc. ASIACRYPT*. 159–180.

C. D. Cannière, O. Dunkelman, and M. Knezevic. 2009. KATAN & KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Proc. Cryptographic Hardware and Embedded Systems*. 272–288.

D. Chakraborty and P. Sarkar. 2008. HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory* 54, 4 (2008), 1683–1699.

D. Chakraborty, C. Mancillas-Lopez, F. Rodriguez-Henriquez, and P. Sarkar. 2013. Efficient hardware implementations of BRW polynomials and tweakable enciphering schemes. *IEEE Transactions on Computers* 62, 2 (2013), 279–294.

D. Chakraborty, C. Mancillas-Lopez, and P. Sarkar. 2017. Disk encryption: Do we need to preserve length? *Journal of Cryptographic Engineering*, 1–21. DOI: 10.1007/s13389-016-0147-0

X. Guo and R. Karri. 2013. Recomputing with permuted operands: A concurrent error detection approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 10 (2013), 1595–1608.

X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri. 2015. Security analysis of concurrent error detection against differential fault analysis. *Journal on Cryptographic Engineering* 5, 3 (2015), 153–169.

J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. 2011. The LED block cipher. In *Proc. Cryptographic Hardware and Embedded Systems*. 326–341.

S. Halevi. 2004. EME: Extending EME to handle arbitrary-length messages with as sociated data. In *Proc. INDOCRYPT*. 315–327.

S. Halevi. 2007. Invertible universal hashing and the TET encryption mode. In *Proc. Advances in Cryptology-Ann. Int. Cryptology Conf. (CRYPTO)*. 412–429.

S. Halevi and P. Rogaway. 2003. A tweakable enciphering mode. In *Proc. Advances in Cryptology-Ann. Int. Cryptology Conf. (CRYPTO)*. 482–499.

S. Halevi and P. Rogaway. 2004. A parallelizable enciphering mode. In *Proc. CT-RSA*. 292–304.

IEEE Security in Storage Working Group (SISWG) P1619. 2017. PRP Modes Comparison IEEE p1619. Retrieved May 2017 from http://siswg.net/, IEEE Computer Society.

D. Karaklajic, J.-M. Schmidt, and I. Verbauwhede. 2013. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 12 (2013), 2295–2306.

M. Liskov, R. L. Rivest, and D. Wagner. 2002. Tweakable block ciphers. In *Proc. Advances in Cryptology-Ann. Int. Cryptology Conf. (CRYPTO)*. 31–46.

P. Maistri and R. Leveugle. 2008. Double-Data-Rate computation as a countermeasure against fault analysis. *IEEE Transactions on Computers* 57, 11 (2008), 1528–1539.
C. Mancillas-Lopez, D. Chakraborty, and F. Rodriguez-Henriquez. 2010. Reconfigurable hardware implementations of tweakable enciphering schemes. *IEEE Transactions on Computers* 59, 11 (2010), 1547–1561.
A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. 2011. Pushing the limits: A very compact and a threshold implementation of AES. In *Proc. Advances in Cryptology*. 69–88.
M. Mozaffari-Kermani and A. Reyhani-Masoleh. 2008. A lightweight concurrent fault detection scheme for the AES S-Boxes using normal basis. In *Proc. LNCS Cryptographic Hardware and Embedded Systems (CHES)*. 113–129.
M. Mozaffari-Kermani and A. Reyhani-Masoleh. 2010. Concurrent structure independent fault detection schemes for the advanced encryption standard. *IEEE Transactions on Computers* 59, 5 (2010), 608–622.
M. Mozaffari-Kermani and A. Reyhani-Masoleh. 2012. Efficient and high-performance parallel hardware architectures for the AES-GCM. 2012. *IEEE Transactions on Computers* 61, 8 (2012), 1165–1178.
M. Mozaffari-Kermani and R. Azarderakhsh. 2013. Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA. *IEEE Transactions on Industrial Electronics* 60, 12 (2013), 5925–5932.
M. Mozaffari-Kermani and R. Azarderakhsh. 2015. Reliable hash trees for post-quantum stateless cryptographic hash-based signatures. In *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*. 103–108.
M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi. 2014. Fault-resilient lightweight cryptographic block ciphers for secure embedded systems. *IEEE Embedded Systems* 6, 4 (2014), 89–92.
M. Mozaffari Kermani, R. Azarderakhsh, and A. Aghaie. 2016. Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC. *ACM Transactions Embedded Computing Systems* 16, 2 (2016), 59:1–59:19.
G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre. 2009. A reliable architecture for parallel implementations of the advanced encryption standard. *J. Electronic Testing: Theory and Applications* 25, 4 (2009), 269–278.
T. Peyrin and Y. Seurin. 2016. Counter-in-Tweak: Authenticated encryption modes for tweakable block ciphers. In *Proc. Advances in Cryptology*. 33-63.
M. O. Rabin and S. Winograd. 1972. Fast evaluation of polynomials by rational preparation. *Communications on Pure and Applied Mathematics* 25 (1972), 433–458.
P. Sarkar. 2009. Tweakable enciphering schemes using only the encryption function of a block cipher. Retrieved January 2018 from https://eprint.iacr.org/2009/216.pdf.
A. Satoh, T. Sugawara, and T. Aoki. 2009. High-performance hardware architectures for Galois Counter Mode. *IEEE Transactions on Computers* 58, 7 (2009), 917–930.
K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. 2011. Piccolo: An ultra-lightweight blockcipher. In *Proc. Cryptographic Hardware and Embedded Systems*. 342–357.
F. X. Standaert, G. Piret, N. Gershenfeld, and J. J. Quisquater. 2006. SEA: A scalable encryption algorithm for small embedded applications. In *Proc. Smart Card Research and Advanced Applications*. 222–236.
M. Yasin, B. Mazumdar, S. Subidh Ali, and O. Sinanoglu. 2015. Security analysis of logic encryption against the most effective side-channel attack: DPA. In *Proc. DFTS*. 97–102.
C. H. Yen and B. F. Wu. 2006. Simple error detection methods for hardware implementation of advanced encryption standard. *IEEE Transactions on Computers* 55, 6, 720–731.

Received May 2017; revised August 2017; accepted November 2017