Reliable Hash Trees for Post-quantum Stateless Cryptographic Hash-based Signatures

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

***Abstract*—The potential advent of quantum computers in coming years has motivated security researchers to start developing resistant systems capable of thwarting future attacks, i.e., developing post-quantum cryptographic approaches. Hash-based, code-based, lattice-based, multivariate-quadratic-equations, and secret-key cryptography are all potential candidates, the merit of which is that they are believed to resist both classical and quantum computers and applying "Shor's algorithm"—the quantum-computer discrete-logarithm algorithm that breaks classical schemes—to them is infeasible. In this paper, we propose reliable and error detection hash trees for stateless hash-based signatures which are believed to be one of the prominent post-quantum schemes, offering security proofs relative to plausible properties of the hash function. We note that this work on the emerging area of reliable, error detection post-quantum cryptography, can be extended and scaled to other approaches as well. We also note that the proposed approaches make such schemes more reliable against natural faults and help protecting them against malicious faults. We propose, benchmark, and discuss fault diagnosis methods for this post-quantum cryptography variant choosing a case study for hash functions, and present the simulations and implementations results to show the applicability of the presented schemes. The presented architectures can be tailored for different reliability objectives based on the resources available, and would initiate the new research area of reliable, error detection post-quantum cryptographic architectures.**

***Index Terms*—Error detection, hash-based signatures, post-quantum cryptography, reliability.**

I. INTRODUCTION

With the potential development of quantum computers in the future, post-quantum cryptography needs to get much developed and advanced so that the transition is safe and secure. In other words, after the advent of such powerful systems, cryptographic protocols and architectures such as Rivest-Shamir-Adleman (RSA) and elliptic curve digital signature algorithm (ECDSA) can be compromised and, thus, post-quantum security mechanisms need to be well in-place to thwart post-quantum attacks. Indeed RSA and ECDSA which are perceived today as being small and fast, can be potentially broken in small polynomial time (and scaling up to secure parameters is infeasible) by "Shor's algorithm" [1].

Inapplicability of Shor's quantum algorithm to post-quantum approaches such as hash-based, code-based, lattice-based, multivariate-quadratic-equations, and secret-key cryptography and the fact that another quantum algorithm, "Grover's algorithm," [2] is partially applicable yet, comparably very slow, make the aforementioned post-quantum cryptography mechanisms attractive. Hash-based signature schemes are, perhaps, viable, secure solutions (with small architectures and key sizes), where every signature scheme uses a cryptographic hash function. Relativity of security proofs to the properties of hash functions and obtained interesting results [3] suggest using hash-based signatures security in presence of quantum adversaries; a matter which is yet to be proven for many other post-quantum signature proposals.

Natural and malicious error detection is the concentration of a number of previous works in classical cryptography. An important reason leading to natural faults in the very-large-scale integration (VLSI) implementations is hardware failures, for instance, natural VLSI single event upsets, electromagnetic waves, or external radiations. In cryptographic hardware and embedded systems, the possibility of mounting active side-channel analysis attacks through fault injections, commonly referred to as fault attacks, necessitates error detection schemes capable of thwarting such attacks. Boneh *et al.* [4] first introduced how to recover secret keys of RSA and the Discrete-Logarithm-Based crypto-systems through fault analysis attacks. For cryptographic architectures such as the Advanced Encryption Standard (AES), much research has been carried out to achieve fault-immune structures, see, for instance, [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] (also, refer to [15] for reliable architectures for lightweight cryptography). Moreover, concerning the finite field arithmetic architectures, various concurrent error detection (CED) multipliers have been proposed [16], [17], [18], [19], [20] to provide reliability mechanisms for crypto-systems.

This work presents error detection schemes for the inner architectures of hash-based signatures for post-quantum resistivity. In this paper, we present two different mechanisms, i.e., structure-dependent and -oblivious detection schemes, and the reliability and fault resilience approaches of such signatures are presented. We note that this choice among other post-quantum mechanisms has been due to the aforementioned security and key-size reasons, yet it does not confine the proposed methods to be applicable to other post-quantum approaches. We refrain proposing error detection *"stateful"* signature mechanisms, i.e., reading a secret key/message and generating a signature and an updated secret key, as if the secret key update fails, security disintegrates [21], [22]. *Therefore, stateless post-quantum hash-based signatures (using parameters that provide 2^{128} security against quantum attacks, practical usage models, and low-cost implementations) are considered and the respective diagnosis approaches are presented*. We focus on the full binary hash tree constructions

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

R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA (e-mail: rxaeec@rit.edu).

978-1-4799-8606-4/15/$31.00

within such signature schemes to provide error detection approaches. Due to technical constraints, an attacker may not be able to inject a single-bit fault; therefore, in practice, multiple faults occur and this is considered in the (transient and permanent) fault model assessed throughout this paper. The ratios of the errors detected are different depending on the error detection methods taken. Although we focus on VLSI defects, the high error coverage of the presented schemes would increase the difficulty for potential fault attackers. Through error simulations, the error coverage for the proposed schemes are derived and it is shown that with high error coverage, reliable architectures are devised.

The remainder of this paper is organized as follows. Preliminaries related to post-quantum hash-based signatures are presented in Section II. In Section III, the proposed error detection approaches are presented. The results of the fault injection simulations and the implementations are presented in Section IV. Finally, Section V concludes this paper.

II. Preliminaries

In this section, we present preliminaries related to hash-based signatures and their stateless variants.

Hash-based signatures proposed through Lamport scheme [23] include public key consisting of two hash outputs for secret inputs. In Merkle scheme (commonly referred to as Merkle tree) [24], the process starts with a one-time signature scheme (OTS) and continues through authenticating 2^h key pairs using a binary hash tree of height h. The merit of this approach is its small signatures and secret/public keys; nevertheless, the key generation and signature time are exponential in h (recent, practical solutions resolve this problem [25], [26]).

Stateless hash-based signature schemes (see [21], [27] for instance) have been introduced and used for providing hash-based security mechanisms. In [27], a binary certification tree built out of one-time signature keys is used. For this scheme, key generation requires a single OTS key generation. Signing takes $2n$ OTS key generations and n OTS signatures (can be done in reasonable time for secure parameters). Furthermore, the approach in [27] proposes randomized leaf selection, i.e., instead of applying a public hash function to the message, it selects an index randomly.

A. Dan Bernstein's SPHINCS Construction

In what follows, we present our reference stateless hash-based signature which is presented in [21] (please see [22] for the preliminary work). The functions in SPHINCS are:

- "Two short-input cryptographic hash functions $F : \{0,1\}^n \rightarrow \{0,1\}^n$ and $H : \{0,1\}^{2n} \rightarrow \{0,1\}^n$,
- One arbitrary-input randomized hash function $\mathcal{H} : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^m$ for $m = poly(n)$,
- A family of pseudorandom generators $G_\lambda : \{0,1\}^n \rightarrow \{0,1\}^{\lambda n}$,
- An ensemble of pseudorandom function families $\mathcal{F}_\lambda : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \{0,1\}^n$,
- A pseudorandom function family $\mathcal{F} : \{0,1\}^* \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$ that supports arbitrary input lengths."

Algorithm 1 Root computation.

Input: Leaf index i, leaf L_i, authentication path $Auth_i$=$(A_0, ..., A_{h-1})$ for L_i.
Output: Root node of tree that contains L_i.
Initialize: $P_0 \leftarrow L_i$.
[1] for $j = 0$ to h do
[2] If $\lfloor i/2^{j-1} \rfloor \equiv 0 \bmod 2$ then
[3] $P_j = H((P_{j-1} \,||\, A_{j-1}) \oplus Q_j)$.
[4] If $\lfloor i/2^{j-1} \rfloor \equiv 1 \bmod 2$ then
[5] $P_j = H((A_{j-1} \,||\, P_{j-1}) \oplus Q_j)$.
[6] end for.
Return P_h.

It is noted that full binary hash trees are utilized in the construction as well. In SPHINCS, a binary hash tree of height h has 2^h leaves which are n-bit strings $L_i, i \in [2^h - 1]$. Each node $N_{i,j}$ ($0 < j \leq h$ and $0 \leq i < 2^{h-j}$), of the tree stores an n-bit string. Moreover, root computation is performed as presented in Algorithm 1 [21]. In this algorithm, $Auth_i$ consists of all the sibling nodes of the nodes contained in the path from L_i to the root; moreover, XOR function is denoted as "$\oplus$" and concatenation as $||$ throughout the paper. Finally, we note that we have the construction for binary tree hash so that the values for internal nodes are $N_{i,j} = H((N_{2i,j-1} \,||\, N_{2i+1,j-1}) \oplus Q_j)$, where $Q_j \in \{0,1\}^{2n}$ are h masks. Our focus is hash-tree constructions; for more detailed introduction to SPHINCS including its parameters, message signer HORST, and the Winternitz one-time signature (WOTS$^+$), one can refer to [21].

B. ChaCha (Salsa's Variant)

In this paper, we utilize BLAKE hash algorithm [28] as the algorithm used in the hash-tree constructions which is constructed using the concepts of ChaCha [29], a variant of Salsa20 stream cipher [30] to increase the amount of diffusion per round. Dan Bernstein's SPHINCS construction [21] is based on hash ($\mathcal{H}$) BLAKE algorithm and utilizes ChaCha for deriving two short-input cryptographic hash functions (F and H) mentioned in the previous section. The core operation in BLAKE is ChaCha's quarter round (function $G(a, b, c, d)$) as follows:

$$\begin{aligned} a &\leftarrow a + b \\ d &\leftarrow (d \oplus a) \lll 16 \\ c &\leftarrow c + d \\ b &\leftarrow (b \oplus c) \lll 12 \\ a &\leftarrow a + b \\ d &\leftarrow (d \oplus a) \lll 8 \\ c &\leftarrow c + d \\ b &\leftarrow (b \oplus c) \lll 7, \end{aligned} \tag{1}$$

where in (1), $\lll$ denotes rotation towards the most significant bits. In addition, $+$ and $\oplus$ represent mod-2^{32} addition and 32-bit XOR operations, respectively.

ChaCha20 uses 10 iterations of the double round. Google has selected ChaCha20 along with Bernstein's Poly1305 message authentication code as a replacement for RC4 in

OpenSSL. As of 2014, almost all HTTPS connections made from Android devices to Google properties have used the new cipher suite; Google plans to make it available as part of the Android platform.

III. Proposed Error Detection Schemes

In this section, two different error detection approaches are proposed, i.e., hash algorithm-oblivious scheme in which the fault diagnosis is performed independent of the hash function used and also mechanisms for detection of errors in ChaCha stream cipher. We would like to emphasize that although we use ChaCha in the second scheme, the proposed methods are applicable with slight modifications to (a) similar stream ciphers such as Salsa20 and (b) hash functions which are based on ChaCha or its variants.

We refrain from presenting the double modular redundancy scheme for the sake of brevity as it is straightforward with impractical overhead of around 100% which is not efficient. The two error detection approaches presented in this paper are based on a new recomputing with swapped nodes (RESN) in the hash-tree constructions and combined signatures and recomputing with encoded operands (REEO) for ChaCha, whose details are presented in the following.

A. Recomputing with Swapped Nodes

The hash tree construction with the values for internal nodes as $N_{i,j} = H((N_{2i,j-1} \mid\mid N_{2i+1,j-1}) \oplus Q_j)$, where $Q_j \in \{0,1\}^{2n}$ are h masks and those for leaf nodes as $N_{i,0} = L_i$ is used in post-quantum cryptographic hash-based signature systems to eventually compute the root, i.e., $N_{0,h}$. We note that before applying the hash function to the concatenation of two child nodes to compute their parent, both child nodes are XORed with a randomly chosen mask, i.e., $Q_j \in \{0,1\}^{2n}$. In what follows, full binary trees, unbalanced trees, and their relevant discussions on their error detection are presented.

1) Full Hash Tree Constructions: In proposing error detection schemes for hash tree constructions, we devise hash algorithm-oblivious schemes which are capable of detecting errors for any utilized hash function. A clear advantage here is that this does not confine us to a specific hash algorithm while achieving high permanent and transient error coverage. We do not use regular time-redundancy schemes as well because of their inability in detecting permanent faults, instead, we investigate various recomputations applicable to hash trees.

Schemes such as recomputing with rotated/shifted operands are not applicable to hash trees. The reason is that the n-bit strings in the nodes, e.g., nodes $N_{2i,j-1}$ and $N_{2i+1,j-1}$, are concatenated and modulo-2 added with the respective randomly chosen mask, i.e., $Q_j \in \{0,1\}^{2n}$, and then hashed; thus, a shift or rotation in any of these nodes creates a new $2n$-bit value to be added (modulo-2) and hashed, whose output reconstruction is not readily known. Fig. 1 shows a hash tree construction and details on inapplicability of a chosen recomputation scheme, i.e., recomputing with rotated operands to left. As seen in this figure, typical internal nodes are shown along with hash function and mask modulo-2 addition, where

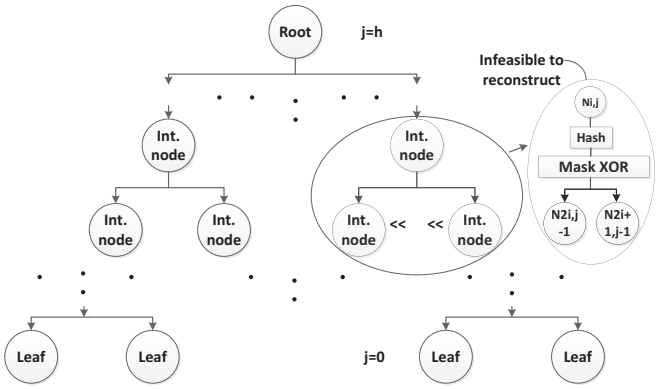

Figure 1. Inapplicability of recomputing with rotated operands for hash tree constructions.

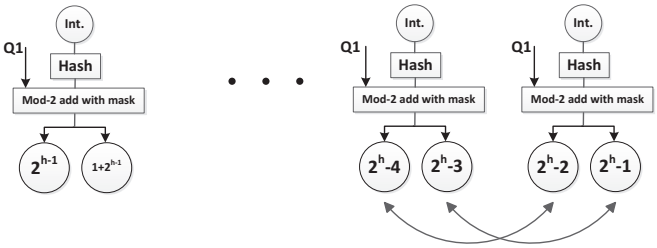

Figure 2. The high-level structure of the proposed RESN scheme.

rotated operands (strings) are shown by << symbol adjacent to the nodes.

Let us present our proposed scheme based on RESN by depicting the swapped strings in the leaf nodes. This is shown in Fig. 2. As seen in this figure, the leaf nodes (for simplicity, we denote such leaves as $\psi_{2^{(h-1)}}$ to ψ_{2^h-1}, shown by their indices only in Fig. 2) are shown and a typical swapped structure is depicted which can be modified based on the required specifications. Let us now prove that the proposed RESN structure is a viable solution for permanent and transient error detection of hash tree constructions. The swapped nodes in Fig. 2 are pairs (ψ_{2^h-2}, ψ_{2^h-1}) and (ψ_{2^h-4},ψ_{2^h-3}). As we have the same masks for deriving the internal node strings corresponding to $j = 1$, i.e., $Q_0 \in \{0,1\}^{2n}$, and then the hash algorithm acts the same on the result, we get the same output for both of the internal nodes, i.e., $N_{2^{(h-1)}-1,1} = H((\psi_{2^h-2} \mid\mid \psi_{2^h-1}) \oplus Q_0)$ and $N_{2^{(h-1)}-2,1} = H((\psi_{2^h-4} \mid\mid \psi_{2^h-3}) \oplus Q_0)$.

Finally, we note that the well-known Merkle tree, which does not include modulo-2 addition with the masks, can take advantage of the proposed RESN through swapping nodes, for instance, pairs (ψ_{2^h-2}, ψ_{2^h-1}) and (ψ_{2^h-4},ψ_{2^h-3}) can be swapped in Merkle trees to get the results of the internal nodes, i.e., $N_{2^{(h-1)}-1,1} = H(\psi_{2^h-2} \mid\mid \psi_{2^h-1})$ and $N_{2^{(h-1)}-2,1} = H(\psi_{2^h-4} \mid\mid \psi_{2^h-3})$.

2) Unbalanced Binary L-Trees: In case the number of bit strings in the verification key of the chosen OTS is not a power of 2, the resulting Merkle tree is unbalanced. Such unbalanced trees are denoted as L-Trees, where the usage model is confined to hash public keys, e.g., those in SPHINCS. The height of an L-Tree is $\lceil log_2 l \rceil$ and it needs $\lceil log_2 l \rceil$

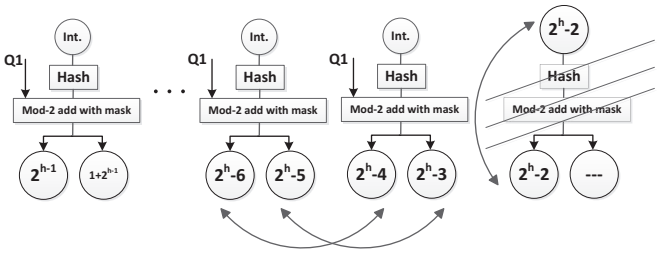

Figure 3. Error detection in unbalanced L-Trees through RESN.

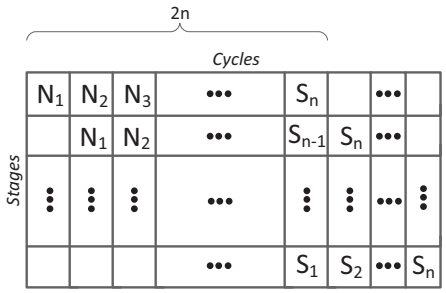

Figure 4. Throughput alleviation for the proposed scheme for hash tree constructions for normal strings (N) and swapped strings (S) of nodes for $\exists_n$ stages.

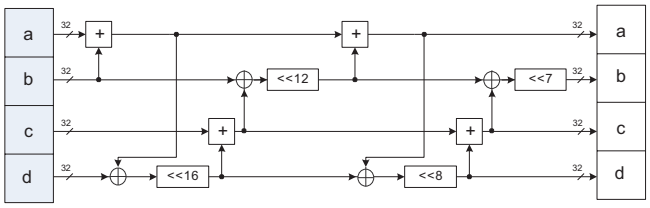

Figure 5. ChaCha's quarter-round operations.

bitmasks.

Error detection for L-Trees through the proposed RESN is possible; yet, one needs to be careful to correctly swap the nodes. We have shown such a structure in Fig. 3. The l leaves of an L-Tree are the elements of a WOTS$^+$ public key and the tree is constructed as full binary trees; however, a left node that has no right sibling (typically in the last row) is lifted to a higher level of the L-Tree until it becomes the right sibling of another node, as seen in Fig. 3. The node (leaf in this case) ψ_{2^h-2} without any right string is lifted up and the error detection is performed through RESN for pairs (ψ_{2^h-4}, ψ_{2^h-3}) and (ψ_{2^h-6},ψ_{2^h-5}) to derive he same output for both of the internal nodes, i.e., $N_{2^{h-1}-2,1} = H((\psi_{2^h-4} \,||\, \psi_{2^h-3}) \oplus Q_0)$ and $N_{2^{h-1}-3,1} = H((\psi_{2^h-6} \,||\, \psi_{2^h-5}) \oplus Q_0)$.

3) Discussions: The proposed RESN scheme is applicable to hash tree constructions for error detection of both permanent and transient faults. Other schemes such as recomputing with rotated or shifted operands (n-bit strings) do not provide efficient approaches for error detection of hash tree constructions as discussed in this section. One of the shortcomings of such recomputation is the decreased throughput of the computation (with the fixed applied frequency). To alleviate this problem, suppose one pipeline-register has been placed to sub-pipeline the structures to break the timing path to approximately equal halves. Let us denote the two halves of pipelined stages by $\exists_1$ and $\exists_2$. The original input is first applied to the architecture and in the second cycle, while the second half of the circuit executes the first input, the second input (see Fig. 4) or the encoded variant of the first input is fed to the first half of the circuit. This trend (which can be scaled to n stages as seen is Fig. 4) is consecutively executed for normal strings (N) and swapped strings (S) of nodes for $\exists_n$ stages (see Fig. 4). Such approach, although increases the latency which may not be admissible in some applications, ensures lower degradation in the throughput (and achieving higher frequencies) at the expense of more area overhead.

B. ChaCha's Error Detection Schemes

In what follows, we present the error detection approaches for ChaCha stream cipher which is used in BLAKE hash function of the hash tree construction. Fig. 5 shows the structure of ChaCha which is the main building block of BLAKE (the three steps for BLAKE compression function, i.e., initialization, round functions, and finalization, as well as the control signal for selecting the initialization or round function steps construct BLAKE). We note that based on our ASIC implementation evaluations, initialization and finalization steps constitute less than 5% of the area of BLAKE.

In what follows, we present two schemes for detection of errors in ChaCha; complementary scheme and REEO approach. In the former, we embed the inverse of the quarter round in the original structure; this is shown in Fig. 6. Specifically, for the 32-bit first and third entries, a and c, (out of the total of four entries to the left-hand side of Fig. 5), adder/subtractors are used instead of single adders (compare Fig. 5 with Fig. 6). In the hardware implementation of the adder/subtractor unit, adding is performed as normal, while subtraction is done by complementing an input and having carry-in as one, i.e., twos complement process. The 32-bit second and fourth entries in Fig. 6 (b and d) are also complemented by reverting the rotation operations (compare Fig. 6 with Fig. 5). Finally, in the right hand side, we obtain the outputs of the original, $a-d$, and the reverse algorithm, $a_r - d_r$. It is shown in the next section that with high error coverage for both transient and permanent faults, the proposed scheme results in acceptable hardware overhead. Finally, it is noted that the pipelined stages shown in this scheme (Fig. 6) result in much decrease in throughput degradation.

Our presented complementary scheme has high error coverage which is of benefit for reliability-constrained applications. Although taking advantage of our proposed embedding approach reduces the hardware overhead of this scheme, resource-constrained devices might just be able to tolerate low hardware overheads. For ChaCha and to get a low-overhead error detection approach, one can use the alternative REEO approach in which recomputations are performed through encoded entries (for instance, rotations). The proposed REEO-based scheme is shown in Fig. 7 (although we have shown the recomputing with rotated operands [nodes or leaves in the

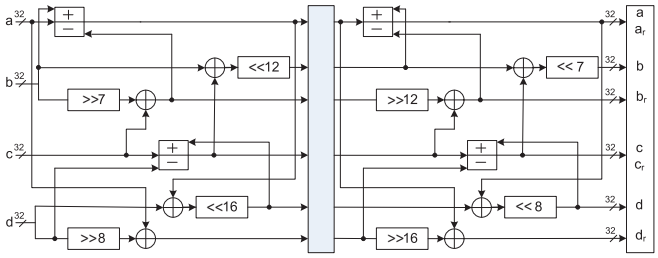

Figure 6. The proposed complementary scheme for ChaCha with pipelined stages.

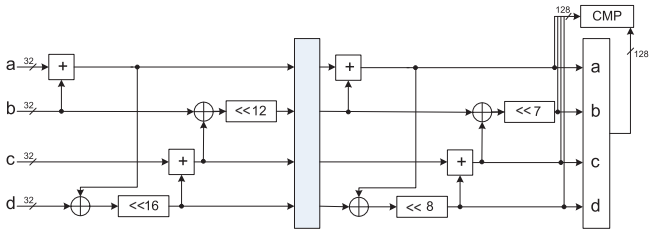

Figure 7. The proposed recomputation with encoded hash tree construction nodes (leaves) scheme for ChaCha.

hash-based construction], encoding schemes are not certainly confined to this approach). The pipelined original structure in this figure is executed two times with original and rotated inputs (for the four 32-bit inputs in Fig. 7). We note that for the XOR and rotation operations in Fig. 7, one would only need to rotate the original inputs. However, for modular addition, based on a modification we have made, a zero is inserted between the 32-bit rotated operands (A and B as examples) before addition and then after addition the middle bit is discarded to obtain C. It is worth-mentioning that after each addition, we require to discard the middle-bit (which is the carry-out of the last-bit addition of the non-rotated operands), because it can be non-zero. Finally, a comparison is performed as shown in Fig. 7 in which the original result is compared with back-rotated results.

IV. Error Simulations and ASIC Implementations

Throughout this paper, both single and multiple stuck-at faults have been considered. With respect to fault analysis attacks, due to the technological constraints, single stuck-at fault may not be applicable for an attacker to flip exactly only one bit to gain more information. Thus, multiple stuck-at faults are also considered in this paper. We consider the BLAKE-64 structure which instantiates a variant of the ChaCha block to construct a hash function used in hash trees. According to the fault model presented in this section, both stuck-at zero and stuck-at one faults are injected in multiple random locations. We consider the transient faults to occur at both runs independently. However, for permanent faults, each fault can occur at the same location for both runs with the same polarity. Therefore, the number of fault locations for this type is half of that for the transient one.

For single-bit fault injection, we generated 1,000,000 random inputs and for both types of stuck-at faults (we note that this number of random inputs was chosen to have a practical simulation process; however, single stuck-at faults were injected exhaustively). For multiple-bit fault injection simulations, the same number of injections were performed (a number of multiple-bit faults were injected per each random input) with random locations. Single stuck-at faults were detected 100% using both complementary and REEO approaches and for multiple stuck-at faults, we analyzed $2-8$ bit faults for transient and permanent types as well as random number of bits. The results show that for transient faults, the percent detection ratio is 99.88% to 99.94% and for permanent faults, we have the error coverage of 99.94% to 99.99%. The faults with more number of bits can be detected with higher ratios and the error detection capability for permanent multiple-bit faults is slightly higher.

This section also presents the results of our ASIC syntheses performed for the original and the error detection structures of the BLAKE structure (constructing a hash function used in hash trees such as L-Tree, full binary tree, Merkle tree, and the like) to benchmark the overheads induced. We note that ASIC is chosen based on the resources available to us (library and tools), and as our presented schemes are not dependent on the hardware platform; similar overheads are expected if FPGAs are utilized for the implementations. Through these ASIC syntheses, the overheads in terms of hardware and timing are derived. We have used the TSMC 65-nm standard-cell library for the original and the error detection structures and Synopsys Design Compiler [31].

To benchmark the performance of the proposed schemes, we have done implementations for the original and fault diagnosis schemes, whose results are presented in Table I. Moreover, based on the sub-pipelining approach we presented in this paper, one can alleviate the inherent performance degradations of the proposed structures. Specifically, with the expense of adding registers for deep sub-pipelining (for instance, one stage sub-pipelining in Table I), higher frequencies are achieved which make the degradations in throughput less intense. We note that in Table I, the areas (in terms of μm^2), maximum working frequencies (in terms of MHz), and throughputs (in terms of $Gbps$) have been obtained. In order to make the area results meaningful when switching technologies, we have provided the NAND-gate equivalency. This is performed using the area of a NAND gate in the utilized TSMC 65-nm CMOS library which is $1.41\mu m^2$.

In Table I, for the original BLAKE and the proposed schemes, the 4G BLAKE structure is utilized. In this structure, four parallel internal operations are performed simultaneously and twice. According to Table I, for the complementary scheme, the area overhead and throughput degradation are 33.1% and 14.5%, respectively. These are lower for the REEO-based scheme, i.e., 9.1% and 10.4%, respectively. We note that, as expected, the overheads for the complementary schemes are higher than those for REEO-based ones. However, the advantage of the complementary schemes is their slightly higher error coverage.

Table I
AREA OVERHEAD AND PERFORMANCE DEGRADATIONS OF THE PROPOSED SCHEMES.

Structure	Area [μm^2] (KGE)	Overhead	Frequency [MHz]	Throughput [$Gbps$]	Degradation
Original	79,772 (56.5)	-	307	9.6	-
Complementary	106,191 (75.3)	33.1%	538	8.2	14.5%
REEO[1]	87,012 (61.7)	9.1%	551	8.6	10.4%

1. One stage sub-pipelined architecture.

V. CONCLUSIONS

In this paper, two fault diagnosis approaches for hash-based post-quantum signatures are proposed. The merit of the proposed schemes is that they are a step-forward towards reliability and fault attack immunity of resistant hash trees in future potential post-quantum systems. We have developed a method for swapping nodes and leaves in binary hash trees, L-Trees, and Merkle trees to detect faults, where recomputations with rotated/shifted, and in general, encoded operands fail. Moreover, for the inner hash function, we have presented two diagnosis methods which are capable of reaching high error coverage with acceptable area overhead and performance degradation. Based on the reliability requirements and available resources, one may select the error detection schemes suitable for these architectures.

Finally, in this paper, we have considered hash-based signature schemes as one potential method for post-quantum cryptography. An intriguing future investigation is developing post-quantum resistant and reliable architectures through code-based, lattice-based, and multivariate-quadratic-equations, and ideally introducing a new metric which benchmarks post-quantum architecture reliability and fault attack immunity.

ACKNOWLEGEMENT

The authors would like to thank the reviewers for careful reading and valuable comments. This work has been supported partly by Texas Instruments award granted to the authors.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, 1997.

[2] Lov K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. ACM Symp. Theory of Computing,* 1996, pp. 212-219.

[3] F. Song, "A note on quantum security for post-quantum cryptography," in *Proc. Post-Quantum Cryptography*, 2014, pp. 246–265.

[4] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. Int. Conf. Eurocrypt*, 1997, pp. 37–51.

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

[6] T. G. Malkin, F. X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Workshop Fault Diagnosis and Tolerance in Cryptography*, Oct. 2006, pp. 159–172.

[7] G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre, "A reliable architecture for parallel implementations of the Advanced Encryption Standard," *J. Electronic Testing: Theory and Applications*, vol. 25, no. 4, pp. 269–278, Aug. 2009.

[8] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the Advanced Encryption Standard," *IEEE Trans. Comput*., vol. 59, no. 5, pp. 608–622, May 2010.

[9] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," to appear in *J. Cryptogr. Eng.*, 2015.

[10] P. Maistri and R. Leveugle, "Double-data-rate computation as a countermeasure against fault analysis," *IEEE Trans. Comput*., vol. 57, no. 11, pp. 1528-1539, Nov. 2008.

[11] X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1595-1608, Oct. 2013.

[12] 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 and Fault Tolerance in VLSI Systems (DFT)*, Oct. 2011, pp. 325-331.

[13] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Parity-based fault detection architecture of S-box for Advanced Encryption Standard," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Oct. 2006, pp. 572-580.

[14] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A lightweight high-performance fault detection scheme for the Advanced Encryption Standard using composite fields," *IEEE Trans. Very-large Scale Integr. (VLSI) Sys*., vol. 19, no. 1, pp. 85-91, 2011.

[15] M. Mozaffari Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925-5932, Dec. 2013.

[16] S. Fenn, M. Gossel, M. Benaissa, and D. Taylor, "On-line error detection for bit-serial multipliers in $GF(2^m)$," *J. Electronic Testing: Theory and Applications*, vol. 13, pp. 29–40, 1998.

[17] S. Bayat-Sarmadi and M. Anwar Hasan, "On concurrent detection of errors in polynomial basis multiplication," *IEEE Trans. Very-large Scale Integr. (VLSI) Sys*., vol. 15, no. 4, pp. 413–426, Apr. 2007.

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

[19] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "NREPO: Normal basis Recomputing with Permuted Operands," in *Proc. HOST,* 2014, pp. 118-123.

[20] M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *Embedded Systems Letters*, vol. 6, no. 4, pp. 89-92, 2014.

[21] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," in *Proc.* EUROCRYPT, 2015, pp. 368-397.

[22] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," *IACR Cryptology ePrint Archive,* 2014, pp. 1-30.

[23] L. Lamport, "Constructing digital signatures from a one way function," *Technical Report SRI-CSL-98, SRI International Computer Science Laboratory*, 1979.

[24] R. Merkle, "A certified digital signature," in *Proc. CRYPTO,* 1990, pp. 218–238.

[25] A. Hülsing, L. Rausch, and J. Buchmann, "Optimal parameters for XMSS," in *Proc. Security Engineering and Intelligence Informatics*, 2013, pp. 194–208.

[26] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS - a practical forward secure signature scheme based on minimal security assumptions," in *Proc. Post-Quantum Cryptography*, 2011, pp. 117–129.

[27] O. Goldreich, "Foundations of Cryptography," Vol. 2, Basic Applications. Cambridge University Press, Cambridge, UK, 2004.

[28] "BLAKE hash function", available: https://131002.net/blake/.

[29] D. J. Bernstein, "ChaCha, a variant of Salsa20", available: http://cr.yp.to/chacha/chacha-20080120.pdf, 2008.

[30] D. J. Bernstein, "The Salsa20 family of stream ciphers Salsa", available: http://cr.yp.to/snuffle/salsafamily-20071225.pdf, 2007.

[31] Synopsys, www.synopsys.com.