# Error Detection Architectures for Ring Polynomial Multiplication and Modular Reduction of Ring-LWE in $\frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$ Benchmarked on ASIC

Ausmita Sarker , *Student Member, IEEE*, Mehran Mozaffari Kermani , *Senior Member, IEEE*, and Reza Azarderakhsh , *Member, IEEE*

*Abstract*—**Ring learning with error (ring-LWE) within lattice-based cryptography is a promising cryptographic scheme for the post-quantum era. In this article, we explore efficient error detection approaches for implementing ring-LWE encryption. For achieving accurate operation of the ring-LWE problem and thwarting active side-channel attacks, error detection schemes need to be devised so that the induced overhead is not a burden to deeply embedded and constrained applications. This article, for the first time, investigates error detection schemes for both stages of the ring-LWE encryption operation, i.e., ring polynomial multiplication and modular reduction. Our schemes exploit recomputing with encoded operands, which successfully counter both natural faults (for the stuck-at model). We implement our schemes on an application-specific integrated circuit. As performance metrics show hardware overhead, our schemes prove to be low complexity with high error coverage. The proposed efficient architectures can be tailored and utilized for post-quantum cryptographic schemes in different usage models with diverse constraints.**

*Index Terms*—**Application-specific integrated circuit (ASIC), cryptographic engineering, ring learning with error (ring-LWE), ring polynomial multiplication (RPM).**

## NOMENCLATURE

| | |
|---|---|
| ASIC | Application-specific integrated circuit. |
| CED | Concurrent error detection. |
| DFIA | Differential fault intensity analysis. |
| FHE | Fully homomorphic encryption. |
| FPGA | Field-programmable gate array. |
| GE | Gate equivalent. |
| LWE | Learning with error. |
| NTT | Number theoretic transform. |
| REScO | Recomputing with scaled operands. |
| RESO | Recomputing with shifted operands. |
| RESwO | Recomputing with swapped operands. |
| RENO | Recomputing with negated operands. |
| ring-LWE | Ring learning with error. |
| RPM | Ring polynomial multiplication. |
| SHE | Somewhat homomorphic encryption. |
| VLSI | Very-large-scale integration. |

## I. INTRODUCTION

**L**ATTICE-BASED cryptography is popular for its resistance against known quantum algorithms, as its security incorporates worst-case hardness of lattice problems [1]. Ideal lattices have revolutionized post-quantum cryptography by providing realizable execution, higher efficiency, and low parameter size. LWE [2] is one of the most versatile worst-case lattice problems and allows us to completely pull out the lattice interpretation, resulting in an extremely simple scheme. Ring-LWE [3] is one of the most explored and studied lattice-based cryptographic schemes, introducing even more efficient encryption scheme than the standard lattice problems [4], practically realizable, and efficient for hardware implementation [5], [6], among post-quantum cryptosystems.

Ring-LWE emerges as a promising post-quantum cryptosystem to employ at limited-resource environments. Besides encryption and key generation, FHE [7], and SHE [8], two emerging groundbreaking techniques to secure cloud data rely on ring-LWE for efficient and advanced operations.

RPM is an integral part of a number of emerging post-quantum cryptographic algorithms and various noncryptographic applications. RPM is the most rigorous computation for ring-LWE, FHE, SHE, and a number of other cryptographic architectures. Thus, designing an efficient RPM architecture will certainly improve the performance of these state-of-the-art cryptosystems. RPM has versatile applications outside the cryptographic area. Erasure coding [9], a strategy to reconstruct corrupt data, uses RPM to ensure cost-effectiveness and less complexity. Ensuring the privacy of electronic medical record [10] or multiparty communication [11], along with many other applications [12]–[15], apply efficient realization of RPM. Consequently, a robust and efficient RPM will be much beneficial in terms of time and hardware complexities.

Ring-LWE involves addition and multiplication over a polynomial ring, where multiplication is the most rigorous operation and is computed using NTT [16], a robust and efficient construction [17]–[19], with smaller key lengths. Thus, efficient and fault-free modular multiplication of NTT is crucial to both high-speed and secure operation. Error detection architectures

for both multiplication and modular reduction operations of NTT will enhance the security of current ring-LWE cryptosystems to a great scale.

Previous works have been performed on error detection schemes on several cryptosystems [20]–[24]. The research in [20] focused on different aspects of tweakable enciphering schemes (TES), including implementations on hardware and software platforms, algorithmic security, and applicability to sensitive security-constrained usage models on TES. The work in [21] challenged the traditional use of fault coverage for uniformly distributed faults as a metric for evaluating the security of CED against differential fault analysis (DFA). In [22], the security of logic encryption against side-channel attacks was evaluated. The problem of exploitable fault characterization in the context of DFA attacks on block ciphers was addressed in [23]. [24] identified the weaknesses in the infection mechanism of the countermeasure that could be exploited by attacks, which change the flow sequence. This article proposes suitable randomization to reduce success probabilities of attacks, which change the flow sequence and develop a fault-tolerant implementation of the countermeasure. While these works are based on classical cryptosystems, we have very few literature works on error detection for post-quantum cryptosystems. The major contribution of this article is that we applied error detection schemes on post-quantum cryptosystems, unlike these previous works based on classical cryptosystems. Our error detection schemes are applied on RPM and modular reduction. Although the previous works have explored error detection on hash-based secure signature [25] and NTT of lattice-based cryptosystems [26], both of which are post-quantum cryptosystems, this article, for the first time, explores error detection schemes on RPM and modular reduction architectures, both integral to any lattice-based cryptosystems.

In this article, we propose error detection schemes of both RPM and modular reduction block, as different ring-LWE architectures use different moduli, depending on the security level and application. The main contributions of this article are as follows.

1) We introduce error detection schemes for RPM with several modulo $q$ architectures within the ring $R = \frac{\mathbb{Z}_q[x]}{x^n+1}$. Among the merits of the proposed schemes is that they are platform oblivious.
2) The proposed error detection schemes are RESO and RESwO. We apply both these schemes to different modulo $q$ architectures, where they could detect the faults injected with high error coverage.
3) We also introduce error detection schemes for the RPM architecture, RENO, a subset of REScO with different performance and implementation metrics and efficiency. These approaches add very little hardware overhead, which is advantageous to incorporate in deeply embedded systems.
4) The proposed error detection schemes are assessed, and the results show the error coverage. We implement our schemes on an ASIC, using Synopsys Design Compiler and a 65-nm standard cell library, to derive the implementation and performance metrics.

The rest of this article is organized as follows. Section II recaps the theoretical background of the RPM technique and ring-LWE encryption. Section III discusses our motivation and the proposed error detection schemes for ring-LWE architectures. Section IV summarizes our hardware implementation results. Finally, Section V concludes this article.

## II. PRELIMINARIES

### A. Ring Polynomial Multiplication

In this article, we have considered polynomial in the ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$. The irreducible polynomial inside this ring is represented as $f(x)$ with degree of $n$. Let two polynomials in this ring be $a(x)$ and $b(x)$. The multiplication of $a(x)$ and $b(x)$ is derived as

$$a(x) \cdot b(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j x^{i+j} \mod f(x). \qquad (1)$$

Here, we use the case presented in [3]. $f(x)$ is an irreducible polynomial, where $f(x) = x^n + 1$. Here, $n$ is a power of 2, $p$ is a prime number, and $p \equiv 1 \mod 2n$. From the properties of irreducible polynomial, we can write $x^n \equiv -1 \mod f(x)$. Using this value of $x^n$ in (1), we derive the polynomial multiplication as

$$c(x) = a(x) \cdot b(x)$$

$$= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (-1)^{\lfloor \frac{i+j}{n} \rfloor} a_i b_j x^{i+j \bmod n} \mod f(x). \qquad (2)$$

### B. Ring-LWE Encryption Scheme

Public-key encryption and signatures are essential for constructing lattice-based cryptosystems. Difficulty of ring-LWE problems is the measure of their security, comparable to the worst-case lattice problems [3]. Ring-LWE provides both encryption and portions of the signature scheme of ideal lattices, within a short key space, resulting in faster algebraic operations. The cryptographic schemes of the ring-LWE problem perform addition and multiplication over $R = \frac{\mathbb{Z}[x]}{x^n+1}$ and $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$, where $q$ is a prime number and $n$ is power of 2. Such problems need one to decide whether the samples $(a_1, t_1), \ldots, (a_m, t_m) \in R_q \times R_q$ are chosen uniformly random, or each $t_i = a_i s + e_i$, where $s, e_1, \ldots, e_m$ have small coefficients from the (1-D) discrete Gaussian distribution $D_\sigma$, with standard deviation $\sigma$ and mean 0, to attain best entropy/standard deviation ratio [6].

In the following, we describe the steps of the encryption scheme. The NTT of polynomial $a$ is denoted as $\tilde{a}$.
1) *Key generation stage GEN(a):* Two error polynomials $\tilde{r}_1$ and $\tilde{r}_2$ are sampled from $D_\sigma$, and let $\tilde{p} = \tilde{r}_1 - \tilde{a}.\tilde{r}_2 \in R_q$. The public key is the polynomial pair $(\tilde{a}, \tilde{p})$ and the secret key is $\tilde{r}_2$.
2) *Encryption stage ENC($\tilde{a}, \tilde{p}, M$):* The input message $M \in \{0, 1\}^n$, is encoded into a polynomial $\tilde{M} =$ encode $(M) \in R$, by multiplying each message bit by $\lfloor (q/2) \rfloor$. The ciphertext can be obtained as $\tilde{c}_1 = \tilde{a}\tilde{e}_1 + \tilde{e}_2$ and $\tilde{c}_2 = \tilde{p}\tilde{e}_1 + \tilde{e}_3 + \tilde{M}$, where $\tilde{e}_1, \tilde{e}_2$, and $\tilde{e}_3 \in R$ are three error polynomials, sampled from $D_\sigma$.
3) *Decryption stage DEC($\tilde{c}_1, \tilde{c}_2, \tilde{r}_2$):* Inverse NTT will recover $\tilde{M}$ using $\tilde{M} = \text{INTT}(\tilde{r}_2\tilde{c}_1 + \tilde{c}_2)$. Decoding of $M$ from $\tilde{M}$ can be found elementwise, using the following

rule: if $\tilde{M}[i] \in (-\lfloor (q/4) \rfloor, \lfloor (q/4) \rfloor)$, then $M[i] = 0$, else $M[i] = 1$, for $0 < i < n - 1$.

A number of combinations of $(n, q, \sigma)$ have been explored in previous work. The research works in [1] and [3] have proposed $(256, 4093, 8.35)$ and $(214, 16\,381, 7.37)$ as medium- and high-security parameter sets, respectively. Here, medium and high security correspond to the hardness of breaking an AES-128 and AES-256 bit block cipher, respectively. The works in [6] and [18] adopt the parameter sets to $(256, 7861, 11.31/\sqrt{2\pi})$ and $(512, 12289, 12.18/\sqrt{2\pi})$ as medium- and high-security parameters, compared to AES-128 and AES-256, respectively.

## III. Proposed Error Detection Scheme

### A. Fault Model

In fault attacks (intentional and malicious fault injections), preferably, single-bit faults using the stuck-at model are injected. We considered malicious faults; thus, we added the faults in the input signals of the circuit. We used AND and OR logic gates at the input of the multiplier to simulate stuck-at zero and stuck-at one faults, respectively, for both $a$ and $b$. For example, we perform AND operation between logic zero and a bit of the multiplier input, suppose $a$, to simulate 1-bit stuck-at zero fault at input $a$. Our schemes are also applicable to natural (accidental) faults, which can occur anywhere, starting from the inputs, the components, the outputs, or even in the internal signals. By repeatedly comparing the erroneous and error-free outputs, the last subkey is derived, and eventually, the secret key is compromised (noting the technological constraints, an attacker may not be able to inject a single stuck-at fault. Therefore, multiple bits might be flipped). We note that the stuck-at fault model (both single and multiple) is able to model both natural and malicious faults and thus is utilized throughout this article to achieve this twofold goal of the proposed schemes [27]. This is one of the reasons that adjacent stuck-at faults need to be considered in fault models as well. We note that such fault models consider both malicious faults and also natural faults based on stuck-at faults considered in this article.

### B. Error Detection Schemes for RPM

In this article, we present efficient error detection architectures for polynomial ring multiplication within ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$. The proposed schemes can be applied to general polynomials or operands, not confined to a subset or special cases of polynomials. Previous work in [28] has presented shift operation for the coefficients of one of the operands, as a countermeasure. This method perfectly worked for their model, where one of the operands of the RPM was ternary polynomial.

For a general polynomial case, if we rewrite (2) in matrix form, the multiplication within $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$ can be expressed as

$$
\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ . \\ . \\ . \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 & -a_{n-1} & . & . & . & -a_1 \\ a_1 & a_0 & & & . & -a_2 \\ a_2 & a_1 & & & . & -a_3 \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & . \\ a_{n-1} & a_{n-2} & . & . & . & a_0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ . \\ . \\ . \\ b_{n-1} \end{pmatrix}. \quad (3)
$$

Shifting the coefficients of $a(x)$ produces a very complex circuitry, and decoding the shifted message is practically impossible with low overhead. As a result, we do not utilize shifting operation for general polynomial within $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$, although it worked smoothly for the case of [28].

Besides shifting, research in [28] has also applied the checksum method as the fault detection technique. Actual and predicted checksums are compared to verify if the data are intact. As one of the polynomials for ring multiplication in this research work is ternary polynomial, the checksum of one of the multiplication operands and that of an intermediate computation are theoretically equal. Nonetheless, for the proposed ring multiplication here, the following is derived for checksum $C_s$:

$C_s = \sum_{k=0}^{n-1} c_k = (a_0 b_0 - a_{n-1} b_1 - a_{n-2} b_2 - \cdots - a_1 b_{n-1}) + (a_1 b_0 + a_0 b_1 - a_{n-1} b_2 - \cdots - a_2 b_{n-1}) + (a_2 b_0 + a_1 b_1 + a_0 b_2 - \cdots - a_3 b_{n-1}) + \cdots + (a_{n-1} b_0 + a_{n-2} b_1 + a_{n-3} b_2 + \cdots + a_0 b_{n-1}) = a_0(b_0 + b_1 + b_2 + \cdots + b_{n-1}) + a_1(b_0 + b_1 + b_2 + \cdots - b_{n-1}) + a_2(b_0 + b_1 + \cdots - b_{n-2} - b_{n-1}) + \cdots + a_{n-1}(b_0 - b_1 - \cdots - b_{n-1})$.

Additionally, we have derived the interleaved checksum, where we add the even and odd coefficients of the product of multiplication. The results are given as follows:

$\text{Int}_e = \sum_{k=0,2,4\ldots}^{n-1} c_k = (a_0 b_0 - a_{n-1} b_1 - a_{n-2} b_2 - \cdots - a_1 b_{n-1}) + (a_2 b_0 + a_1 b_1 + a_0 b_2 - \cdots - a_3 b_{n-1}) + \cdots + (a_{n-1} b_0 + a_{n-2} b_1 + a_{n-3} b_2 + \cdots + a_0 b_{n-1}) = a_0(b_0 + b_2 + \cdots + b_{n-1}) + a_1(b_1 + \cdots + b_{n-2} - b_{n-1}) + a_2(b_0 + b_2 + \cdots - b_{n-2}) + \cdots + a_{n-1}(b_0 - b_1 - b_3 - \cdots - b_{n-2})$,

$\text{Int}_o = \sum_{k=1,3,5\ldots}^{n-1} c_k = (a_1 b_0 + a_0 b_1 - a_{n-1} b_2 - \cdots - a_2 b_{n-1}) + (a_3 b_0 + a_2 b_1 + a_1 b_2 - \cdots - a_4 b_{n-1}) + \cdots + (a_{n-2} b_0 + a_{n-3} b_1 + a_{n-4} b_2 + \cdots - a_{n-1} b_{n-1}) = a_0(b_1 + b_3 + b_5 + \cdots + b_{n-2}) + a_1(b_0 + b_2 + \cdots - b_{n-3}) + a_2(b_1 + b_3 + \cdots - b_{n-1}) + \cdots + a_{n-1}(-b_2 - b_4 - \cdots - b_{n-1})$.

where $\text{Int}_e$ and $\text{Int}_o$ are even and odd interleaved checksums, respectively.

Both checksum and interleaved checksum will incur high area overhead, as there is no efficient approach that can minimize the cost of the circuit. The checksum presented in [28] can be applied to ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n-1}$; however, it is not efficient for our ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$. Moreover, the checksum operation of convolution multiplication block in [28] requires no multiplication operation, whereas the checksum in our RPM architecture requires $n$ modular multiplication units. Multiplication is an expensive operation that incurs high area overhead, which makes checksum an unsuitable scheme for RPM. As a result, we introduce recomputing schemes, which will provide us error detection with low cost.

*1) RPM Architecture:* In this article, we propose error detection schemes for the RPM within $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$. However, our scheme is applicable to another polynomial ring multiplication construction: $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n-1}$. The aforementioned multiplication can be expressed as follows:

We utilize a multiplication (modulo $p$) circuit to compute $a(x) \cdot b(x)$. The coefficients of elementwise multiplications are either positive and negative as shown through preceding partial products. We can explain this using (2), where the term $(-1)^{\lfloor \frac{i+j}{n} \rfloor}$ decides whether the coefficients are positive or negative. When $i + j < n, \lfloor \frac{i+j}{n} \rfloor = 0$, then $(-1)^0 = 1$, making the coefficients positive and *vice versa*. The function is given as

follows:

$$(-1)^{\lfloor \frac{i+j}{n} \rfloor} = \begin{cases} 1, & \text{when } i+j < n \text{ or } \lfloor \frac{i+j}{n} \rfloor = 0 \\ -1, & \text{when } i+j \geqslant n \text{ or } \lfloor \frac{i+j}{n} \rfloor = 1 \end{cases}. \quad (4)$$

We require a module capable of performing both the addition and subtraction of two operands. To achieve that, we use a multiplexing adder/subtractor unit. The selector of multiplexer is the term $\lfloor \frac{i+j}{n} \rfloor$. According to (4), the module acts as a mod $p$ adder and as a mod $p$ subtractor when $\lfloor \frac{i+j}{n} \rfloor$ is 0 and 1, respectively.

In our schemes, we utilize the aforementioned multiplication module to ensure smooth operation. At first, we apply REScO and compare the decoded product of multiplication with the output. Afterward, as an economical and low-power subset of scaling, we recompute the multiplication by negating one or both of the operands and compare the decoded message with the output. The latter approach adds very little area overhead, utilizing RPM for both the rings efficiently. We get the final result of the computation in $n$ cycles, as each coefficient is computed in parallel. Depending on objectives for error coverage and overhead, our schemes can be tailored to negate as well as scale one or both of the operands of multiplication, for a high-error-coverage error detection scheme.

*2) Proposed Error Detection Scheme Through Recomputing:* Error detection codes might be generally inefficient and expensive for general polynomial ring multiplication. With a view to making such detection schemes faster and cheaper, we have utilized a recomputing method that scales one or both operands of multiplication as encoding operation. REScO is a modified architecture of RPM, where we insert a multiplexer, a multiplier, and dividers. The selector of the multiplexer, normal/REScO, determines whether it performs original RPM or REScO. In the latter case, one of the operands, e.g., $b$, is scaled with a factor $k$, and we get the encoded operand $e(x) = k \cdot a(x) \cdot b(x)$. For the decoding process, we have to apply multiplicative inversion mod $p$ of the factor, $k$, i.e., $d(x) \equiv k^{-1} \cdot e(x) \bmod p$, where $d(x)$ is the decoded output. Thus, we have to select $k$, carefully, to avoid cases of the nonexisting multiplicative inverse. To achieve that goal, $k$ has to be a nonzero integer, where $\gcd(k, p) = 1$.

In terms of error coverage, REScO is effective in countering faults. However, it incurs substantial hardware overhead from the costly arithmetic operations, e.g., multiplier and divider modules, required during encoding and decoding stages. Inserting many of dividers in the implementation makes it expensive, power consuming, and slow. To solve this situation, we explore a more efficient and low-power subset of REScO, i.e., RENO.

In RENO, we recompute by negating one or both of the operands of multiplication. Negating can be inferred as multiplying with $-1$; hence, RENO is a special case of REScO. During encoding, the selector of the additional multiplexer chooses between normal and RENO operations, as shown in Fig. 1. In the case of RENO, the coefficients of the operands, given by (4), are swapped, using an inverter. Mathematically, $i+j < n, \lfloor \frac{i+j}{n} \rfloor = 0$, becomes 1 after inversion, making the coefficients negative and *vice versa*. Therefore, we get the encoded operand $e(x)$ as $e(x) = -a(x) \cdot b(x)$. During the decode stage, let any of the coefficients of decoded output $d(x)$ be $d_i$, where $d_i \equiv (p - e_i) \bmod p \equiv -e_i \bmod p$. We use $n$ number of subtractors in this stage of RENO. As adder/subtractor modules
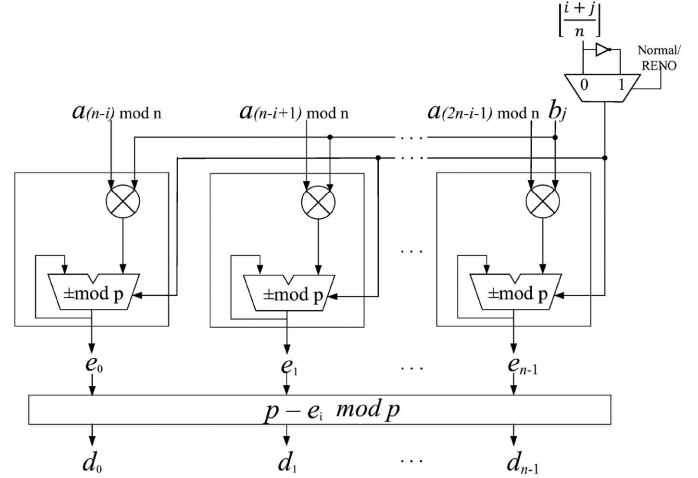


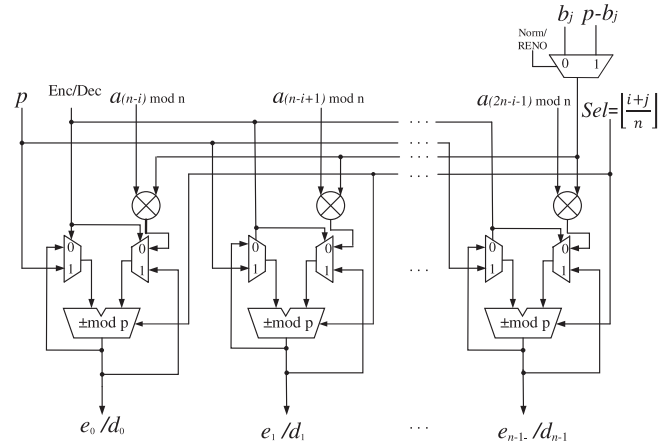Fig. 1. Hardware architecture of the proposed RENO.



Fig. 2. Hardware architecture of the modified RENO.

are inexpensive, compared to multiplier and divider modules of REScO, RENO provides an efficient and low-overhead-error detection method. We would like to emphasize that REScO and RENO are not two different techniques. RENO is a subset of REScO, where we are scaling the operands with $-1$.

The architecture of RENO in Fig. 1 can detect transient faults correctly. However, this architecture can only detect permanent faults present in the multiplexing adder/subtractor module, while failing to detect such faults in the operands or in any other section of the architecture. To resolve this issue, we modify Fig. 1 and negate one of the operands since the beginning of the computation, as presented in Fig. 2. The normal/RENO multiplexer will now select normal operand $b_j$ or negated operand $(p - b_j)$. As a result, the operands are negated at the input stage, and that enables this scheme to detect both permanent and transient faults in operands as well as entire architecture satisfactorily. Moreover, the structure in Fig. 2 reduces the hardware overhead by removing the $(p - e_i) \bmod p$ box and adding two multiplexers before the feedback structure. Using the selectors Enc/Dec, the multiplexers either perform encoding by elementwise multiplication of $a$ and negated $b$, i.e., $(p - b_j)$, or subtract $e_i$ from $p$, giving the decoded output $d_i \equiv (p - e_i) \bmod p$. We perform

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

SARKER *et al.*: ERROR DETECTION ARCHITECTURES FOR RPM AND MODULAR REDUCTION OF RING-LWE
5

| $H_1$ | $N_1$ | $R_1$ | $N_2$ | $\cdots$ | $R_n$ | |
|---|---|---|---|---|---|---|
| $H_2$ | | $N_1$ | $R_1$ | $\cdots$ | $N_n$ | $R_n$ |

Fig. 3. Pipelined scheduling for data path of the proposed schemes.

computations on the operands in two runs: the first run ($run1$) deals with normal computation, and the second run ($run2$) deals with RENO. We use the selector Enc throughout $run1$ and the first $n$ clock cycles of $run2$. In contrast, Dec is selected only in the $(n+1)$th cycle of RENO, in order to complete the decoding of negated operands. In such a manner, we eliminate $n$ number of subtractors by performing the $(p - e_i)$ operation through the already existing adder/subtractor modules. As the hardware overheads of multiplexers are considerably lower than these modules, modified RENO costs even less than regular RENO, while ensuring higher error coverage. We utilize an error detection flag, which is logic OR operation of comparisons for every column. Even if only one of the columns of Fig. 2 has erroneous output, the flag will be set to 1, and we can detect the error.

One can modify RENO by negating both of the operands. In this case, the input operands are $(p - a_i)$ and $(p - b_i)$, instead of $a$ and $b$. As multiplication of two negative terms gives a positive result, there is no need for decoding. Negating both input operands requires more involved encoding and hardware overhead (as seen in Section IV). Nevertheless, in some platforms, absence of decoding stage might compensate for this excess circuitry.

Two unlikely cases may appear during assertion of permanent or long transient faults. One event can be "masking," in which the output is not erroneous, even if a fault exists in the intermediate logic. Such cases are excluded because the circuit masks the faults and these are not translated to errors. The second instance is a rare case, where all the entries of operands $a_i$ and $b_i$ are zero. RENO cannot detect these errors, because negating any zero value will keep it unaltered. However, applying all the input bits to a logic OR gate can be a secondary measure to detect such a case. We would like to emphasize that this would be equivalent to multiplying two zero polynomials, which is an unlikely case.

*3) Ameliorating the Throughput Overhead Through Pipelining:* The delay overhead we took into account is the critical path delay, where the critical path is the path that incurs the highest delay. As our error detection is a time redundancy technique, the total time of a recomputed architecture will be twice of an original architecture deteriorating the throughput, if no measure is in place to compensate such shortcoming. Such absence of pipelining will degrade the throughput drastically, which can be improved by applying subpipelining. Subpipelining will increase the frequency to make sure that the design throughput is close to the original architecture. This will incur slightly higher area overhead, which can be overlooked as we are achieving low throughput degradation of the error detection approach. We insert registers in locations, which will, in turn, break the timing paths into approximately equal halves. We denote the two halves of the pipelined stages as $H_1$ and $H_2$. According to Fig. 3, our scheduling order of normal ($N_i$) and recomputed ($R_i$) operations is shown, where $1 \leq i \leq n$, with $n$ being the number of cycles

in the original nonpipelined approach. We compute $R_i$ and $N_i$ at the same cycle but in different pipelined stages, whereas in the next cycle, $N_{i+1}$ and $R_i$ are computed.

### C. Error Detection Schemes for Ring-LWE Architecture

To construct the ring-LWE encryption architecture, based on preliminaries presented in this article, we utilize a DSP-enabled schoolbook polynomial multiplier, along with a modular reduction block. Here, we emphasize on two sets of parameters, i.e., $(n, q, \sigma) = (214, 16\ 381, 7.37)$ and $(512, 12\ 289, 12.18/\sqrt{2\pi})$, both being high-security parameters. Resemblance between the reduction method of $(n, q, \sigma) = (214, 16\ 381, 7.37)$ and $(256, 4093, 8.35)$ makes our scheme easily modifiable to apply to the other parameter sets [29]. In contrast, $(n, q, \sigma) = (256, 7681, 11.31/\sqrt{2\pi})$ and $(512, 12289, 12.18/\sqrt{2\pi})$ both use the shift–addition–multiplication–subtraction–subtraction (SAMS2) technique for modular reduction in the research works of [18] and [30]. Thus, our error detection scheme presented through such parameter sets is also applicable to the former.

Choosing the proper value of $q$ varies upon the level of security and efficient modular reduction and is based on the property of the modulus, e.g., Fermat number or a large prime number. This article, for the first time, explores error detection schemes within modular operations.

Section III-B2 introduces error detection schemes using recomputing for multiplication operation, i.e., RPM. In Sections III-C1 and III-C2, we explore error detection schemes for modular reduction operations. In Fig. 1, we have seen mod$p$ block, where we can apply our modular reduction operations, based on the value of $p$. Our error detection schemes on modular reduction can be used in any compatible architecture, not being limited to RPM only.

*1) Error Detection Scheme for Polynomial Multiplier and $q = 16\ 381$:* In this construction, we use a DSP-based schoolbook polynomial multiplication scheme, followed by the modulo $q$ operation. For $q= 16\ 381$, it is found that $2^{14} \bmod 16\ 381 = 3$. As a result, the inputs of the DSP blocks are 14 bits in length, and the product can be written as $x_{27...0} = 2^{14} x_{27...14} + x_{13...0} = 3 x_{27...14} + x_{13...0} = (x_{27...14} << 1) + (x_{27...14}) + x_{13...0}$, where left shift is denoted by $<<$. The modular operation reduces the result within $[0, 16\ 380]$, requiring two modulo $q$ operations at most, which is performed by the modulo $q$ reducer block of Fig. 4. In contrast, the DSP block computes the unsigned multiplication through $(AB + C)$. In the case of signed multiplication, i.e., multiplication with a negative number, $(D - A)B + C$ is performed, where $D = q$.

In this article, we propose two variants of recomputing schemes, which we apply to the most rigorous computation of ring-LWE encryption operation, i.e., the entire DSP as well as the modular $q$ reducer block. Fig. 4 shows RESO, in which two of the input operands are shifted to left by 1 bit. Another approach is RESwO, where two of the input operands are swapped. In the former approach, we insert a multiplexer that controls either the normal mode or the RESO mode of operation through the select pin Norm/RESO. In a Norm operation, we get the usual multiplier output with mod $q$ reduction, whereas the RESO mode performs left shift of both operands $A$ and $C$, giving the
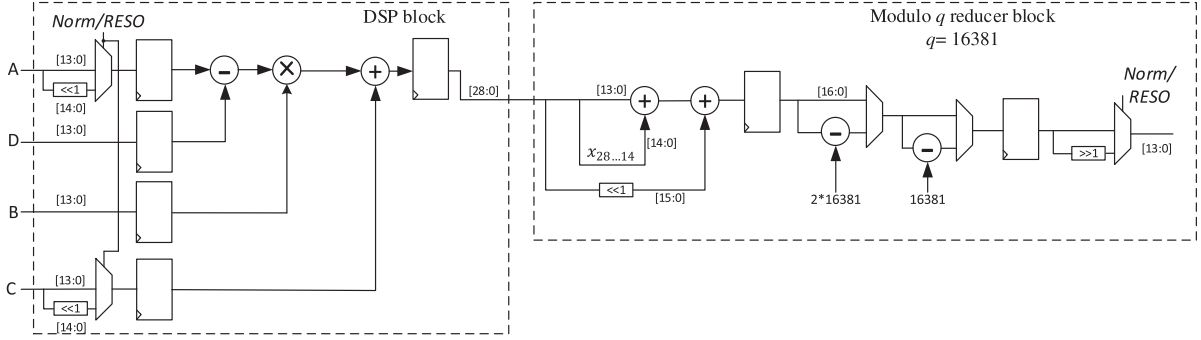
Fig. 4.    Proposed construction of schoolbook $\log_2 q \times \log_2 q$ bit multiplier for $q = 16\,381$.



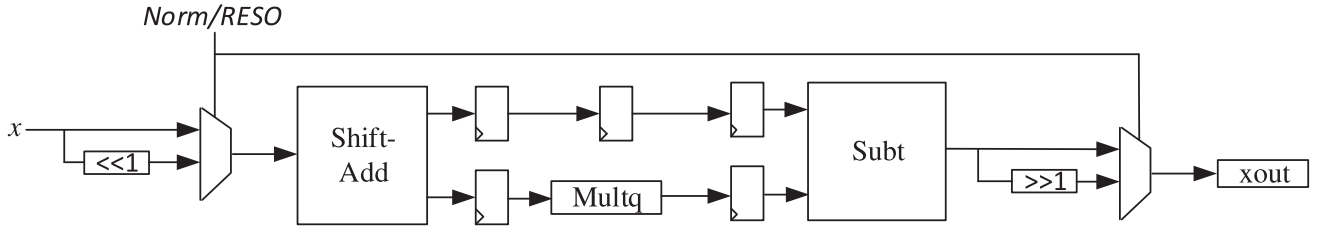Fig. 5.    Proposed SAMS2 construction for error detection in modular reduction.

multiplier output as $2AB + 2C = 2(AB + C)$. The output of the modulo $q$ block is shifted to the right by 1 bit, which will provide $AB + C$, in a fault-free scenario. The outputs of both the rounds are compared, and any discrepancy between the results detects the presence of faults in the architecture. RESwO can also be applied in a similar manner, which will detect both permanent and transient faults with less overhead.

*2) Error Detection Scheme for the SAMS2 Approach and $q = 12\,289$:* Modular reduction operations for a number of values are computationally less efficient than the former values we explored, yet such values of $q$ are famous and widely used in SHE and other cryptographic applications. The works of [18] and [30] apply the values of $q = 7681$ and $12\,289$ and use SAMS2 for faster modular reduction operation.

In the SAMS2 approach, we explain the Norm mode for $q = 12\,289$, where the input contains 14 bits, as $2^{14} \equiv 2^{12} - 1$ mod $12\,289$. In the Shift-Add block, we approximate quotient $t$ of $xout = x - tq$ as $x >> 14 + x >> 16 + x >> 18 + x >> 20 + x >> 22 + x >> 24 + x >> 26$, which is a combination of shift and addition operation, based on [30]. From this value of $t$, we use the $Multq$ block to find the product $tq$. However, the $Multq$ block is a combination of left shifts and addition, resulting in a much efficient scheme, compared to a multiplier. In the last block, i.e., $Subt$, the subtraction between $xin$ and multiples of $q$ from $q$ to $7q$ are performed in parallel, providing a much faster reduction due to simultaneous calculations. Taking the least positive number between $xin - tq$ and the results of the above subtractions, the $Subt$ block works in a loop until the output is not lower than $q$.

In this article, we present the error detection architecture of SAMS2 operation through the RESO mode of the multiplier (see Fig. 5). We apply RESO as encoding and decoding operations of the input and the output, respectively. As the entire SAMS2

operation is linear, applying a left shift at the input stage and a right shift at the output stage should retain the same $xout$ as the Norm mode. Thus, comparing the values of both rounds of operations provides us the error detection if both values of $xout$ fail to coincide.

## IV. ERROR COVERAGE AND ASIC ASSESSMENTS

In this section, we present the results of our error simulations and ASIC assessments using the Synopsys Design Compiler and VHDL with TSMC 65-nm for two security levels and two of our architectures to assess the overhead. Using 65-nm ASIC synthesis, we also present the overhead of the presented constructions for the case studies of moderate- and high-security levels, i.e., for $(n = 256, \ p = 1\,049\,089)$, and $(n = 512, \ p = 4\,206\,593)$, respectively. We have also chosen the third set of parameters $(n = 1024, \ p = 536\,903\,681)$ based on SHE [8]. The benchmarking is performed for the error detection architectures (for two proposed schemes, i.e., Prop. 1: Negating both operands and Prop. 2: Negating one operand, respectively) and also for the original construction.

### A. Fault Simulations

We evaluated the error detection capability of the proposed work based on fault-injection simulation coded in VHDL. Our schemes are through recomputing with encoded operands, a time redundancy technique, which can detect faults by repeating computation. Such time redundancy techniques can detect both permanent and transient faults, as the error in arithmetic logic unit can be detected by comparing $f(x)$ with the $\text{decode}(\text{encode}(f(x))$, given that $x$ is the input of the function $f$. When a transient fault occurs in the input, the decoded output

TABLE I
ERROR COVERAGE AND BINARY CLASSIFICATION TESTS

| Error Coverage | TPR | FNR | Precision | Sensitivity |
|---|---|---|---|---|
| 99.999% | 99.999% | 0.001% | 100% | 99.999% |

becomes decode(encode($f(x')$), with $x'$ being the faulty input. Such discrepancy in the decoded output with respect to the expected output $f(x)$ will detect the fault injection. In our error detection scheme in RPM, the encode is RENO and the construction is self-decoding. To further verify this proof, we injected transient faults in the input $b$ in Fig. 2 only during $run1$. For 65 536 cases, we found that our scheme can detect transient faults for 100% of the times.

To simulate permanent faults, we injected three types of stuck-at faults, i.e., single, 2-bit, and multiple-bit faults for over $10^{12}$ cases, all injected at the input state of the algorithm. An attacker may not be successful at flipping exactly 1 bit to collect sensitive information due to technological constraints, which led us to consider multiple stuck-at faults. The faults that we consider are stuck at 0 and stuck at 1. The schemes provide close to 100% error coverage (reservation on the comparator is explained in the following) for these three cases. The simulation results are confirming that the schemes can detect both transient and permanent faults satisfactorily. We assume that the comparators are hardened, i.e., the comparators are fault free and not compromised. We can harden the comparators in two ways, first by using large transistors that are resistant to faults. An alternative is using triple modular redundancy (TMR) [31]. In such a technique, a module is replicated three times, and the output is extracted from a majority voter. The assumption here is that the voter is immune to faults, which can be achieved by using tristate buffers as voters instead of SRAMs. Selective TMR [32] is another variant to ensure fault detection in comparators, with higher efficiency and lower area overhead than TMR. If none of the comparator hardening techniques are applied and the comparators are compromised, the error coverage will be 50%, while using the TMR techniques will increase the error coverage to 100% [32].

Our schemes provide close to 100% error coverage; our schemes are highly suitable in terms of detecting fault injection on the cryptosystems with high accuracy (see Table I). To further explain, we explored the rate of missed detection, also known as the false negative rate (FNR), which is 0.001%. This can be explained by masking of error. In such situation, the presence of one defect hides the presence of another defect. Our schemes have no false positives (FP) (false alarms), as such cases are relevant where error detection is done in mid-stages of error detection constructions, where detected faults are masked. All faults are correctly detected, and there has not been a case where a fault-free condition was flagged as a faulty one, resulting in zero FP detection and 0% false positive rate (FPR) of our schemes. We can deduce the true positive rate (TPR) of our schemes from TPR% = $(1 - \text{FNR})$%, to be 99.999%. Moreover, our schemes have 100 precision, where we define precision as the ratio of the number of true positives to the number of all positive, i.e., precision = $\frac{\text{TP}}{\text{TP+FP}}$, where FP is explained before and TP is the total number of true positive detection.

Sensitivity is the measure to correctly detect faults where faults are actually injected, i.e., sensitivity = $\frac{\text{TP}}{\text{TP+FN}}$, which, in turn, is equal to TPR. In this article, we define precision as ratio of the number of true positives to the number of all positive, i.e., precision = $\frac{\text{TP}}{\text{TP+FP}}$. The precision of our schemes is 100%. To find the receiver operating characteristics (ROC), one has to plot TPR against FPR with respect to varying threshold values. As the output of comparators shows either a high or low flag for fault and fault-free output, respectively, we do not require various values of threshold in our scenario. Considering the cases in which the comparison units are hardened, the resulting ROC curve is almost a vertical graph, as the horizontal axis denotes FPR and vertical axis denotes TPR. The 99.999% TPR and 0% FPR are the reasons behind such outcome. The simulation results are architecture and platform oblivious, which is presented in Table I for the Artix-7 FPGA device in Xilinx Vivado software.

### B. ASIC Comparison for Error Detection in the RPM Module

As shown in Table II, the area [in terms of $\mu m^2$ which can be converted to kilo gate equivalent (kGE), which is the normalized area for two-input NAND gate by dividing the column numbers by $1.41 \times 10^3$], delay (which is indication of maximum working frequency), and power consumption at the frequency of 20 MHz are tabulated. The original architecture denotes where no error detection schemes were applied. We also note that negating one of the input operands requires more involved encoding and hardware overhead, compared to negation both input operation. We have performed our simulations for register transfer level (RTL) and technology-independent constructions to make sure our results are oblivious of platform (different FPGA families and different ASIC technologies). We laid out the implementation of our schemes on the Synopsys Design Compiler. We would like to mention that we did not fabricate the chip; however, the proposed schemes are platform and implementation oblivious. The implementations have preserved the hierarchy, and the error detection construction has not been affected in optimization, and the efforts for area and delay have been set to medium.

Here, we note that the RENO operation in Section III-B2 and RESO operation in Sections III-C1 and III-C2 are compatible. We explored the efficient schemes in each of the architectures. RENO is computationally more efficient than RESO, in the case of the RPM architecture, as only one mod $p$ negation block suffices the RENO operation. In contrast, to apply RESO on RPM, we require to add left shift blocks to all of the input operands, which will require $(n + 1)$ left shift blocks at the input as we are feeding the coefficients of $a$ in parallel and $n$ right shift block at the output. Thus, RESO incurs much higher area overhead than RENO, making RENO a better recomputing scheme for RPM. However, applying RENO in the modular reduction operation further complicates the mod $q$ negation inside the modulo $q$ reducer block for Fig. 4 and the $Subt$ block for Fig. 5, as they are already a series of negation operation. Consequently, adding another mod $q$ negation will result in a discrepancy in the reduction operation. Moreover, to apply RESO, we only use two left shift blocks and one right shift block in Fig. 4, which is much cheaper than mod $q$ negation units.

We would like to finalize this section by noting that the proposed architectures are oblivious of the standard cell library and hardware platform. Therefore, we expect similar results in overheads on FPGA and ASIC libraries. We also note that

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

8                                                                                                                              IEEE TRANSACTIONS ON RELIABILITY

TABLE II
IMPLEMENTATION RESULTS FOR ASIC TSMC 65-NM OF RPM ARCHITECTURE (PROP. 1: NEGATING BOTH OPERANDS, PROP. 2: NEGATING ONE OPERAND)

| Architecture | Area ($\mu m^2$) | Delay/Frequency | Power (mW) at 20 MHz |
|---|---|---|---|
| Original ($n = 256$, $p = 1049089$) | 260,055 (184 kGE) | 25.2 ns (39.6 MHz) | 5.3 mW |
| Original ($n = 512$, $p = 4206593$) | 539,766 (382 kGE) | 26.5 ns (37.7 MHz) | 10.7 mW |
| Original ($n = 1024$, $p = 536903681$) | 1,097,602 (778 kGE) | 28.0 ns (35.7 MHz) | 23.9 mW |
| Prop. 1 ($n = 256$, $p = 1049089$) | 290,846 (11.5%) (206 kGE) | 33.9 ns (34.5%) (29.5 MHz) | 5.9 mW (11.3%) |
| Prop. 1 ($n = 512$, $p = 4206593$) | 589,111 (11.1%) (417 kGE) | 34.7 ns (32.8%) (28.8 MHz) | 12.8 mW (19.6%) |
| Prop. 1 ($n = 1024$, $p = 536903681$) | 1,254,009 (14.3%) (889 kGE) | 36.3 ns (28.6%) (27.5 MHz) | 26.3 mW (10.1%) |
| Prop. 2 ($n = 256$, $p = 1049089$) | 311,056 (19.6%) (220 kGE) | 28.6 ns (13.5%) (34.9 MHz) | 6.1 mW (15.1%) |
| Prop. 2 ($n = 512$, $p = 4206593$) | 608,223 (12.7%) (431 kGE) | 29.5 ns (11.3%) (33.8 MHz) | 12.4 mW (15.8%) |
| Prop. 2 ($n = 1024$, $p = 536903681$) | 1,262,960 (19.6%) (895 kGE) | 33.8 ns (20.7%) (29.6 MHz) | 26.7 mW (11.7%) |

the throughput and frequency degradations can be alleviated through pipelining at the expense of added hardware overhead. However, our fault model was stuck-at faults, whereas FPGAs deal with other faults, e.g., bit-flip in configuration memory cell, which may result in different error coverage. DFIA that is a combination of differential power analysis and fault injection concepts has gained much attention in the recent past. The biased fault models range from low intensity to higher ones in previous works. Our aforementioned proposed fault detection schemes have capabilities to detect these biased faults. Finally, previous studies of [28] introduced efficient countermeasures against fault attacks on NTRUEncrypt. However, RPM in NTRUEncrypt [28] is a special case, and it is not applicable for encrypting other systems, for example, the ring-LWE in [17]. Moreover, we do not utilize shifting operation for general polynomial within $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$, although it worked smoothly for the case of [28]. We also note that the presented error detection schemes in this article for RPM in the ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$ are not confined to this ring and can be incorporated into a number of other constructions, such as the ring $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n-1}$.

## V. CONCLUSION

In this article, we proposed error detection schemes, i.e., REScO and RENO, a subset of REScO with different performance and implementation metrics and efficiency. Additionally, we employed RESO and RESwO to a number of ring-LWE architectures and modular reduction stages, which can be applied to most well-known modulo operations. These approaches added very little hardware overheads, which was advantageous to incorporate in deeply embedded systems. We benchmarked the proposed architectures to assess their ability to detect transient and permanent faults. Moreover, we implemented the proposed error detection architectures on ASIC, and our results showed that the proposed error detection architectures can be feasibly utilized for RPM in the rings $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n+1}$ and $\mathbb{R} = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^n-1}$. We note that our scheme is suitable for the required performance and implementation metrics for constrained applications.

## REFERENCES

[1] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*, Berlin, Germany: Springer, 2009, pp. 147–191.

[2] O. Regev, "On lattices, learning with errors, random linear codes, cryptography," in *Proc. Annu. ACM Symp. Theory Comput.*, 2005, pp. 84–93.

[3] V. V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *J. ACM*, vol. 60, no. 6, pp. 1–35, Nov. 2013.

[4] N. Gottert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *Proc. 14th Int. Workshop Cryptographic Hardware Embedded Syst.*, Sep. 2012, pp. 512–529.

[5] T. Poppelmann and T. Guneysu, "Towards practical lattice-based public key encryption on reconfigurable hardware," in *Proc. 20th Int. Conf. Sel. Areas Cryptographic*, 2013, pp. 68–85.

[6] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *Proc. 16th Int. Workshop Cryptographic Hardw. Embedded Syst.*, 2014, pp. 371–391.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[8] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proc. ACM Workshop Cloud Comput. Secur.*, 2011, pp. 113–124.

[9] J. Detchart and J. Lacan, "Polynomial ring transforms for efficient XOR-based erasure coding," in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 604–608.

[10] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: Ensuring privacy of electronic medical records," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 103–114.

[11] A. Ben-David, N. Nisan, and B. Pinkas, "FairplayMP: A system for secure multi-party computation," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2008, pp. 257–266.

[12] J. Bos *et al.*, "CRYSTALS—Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2018, pp. 353–367.

[13] S. Streit and F. De Santis, "Post-quantum key exchange on ARMv8-A: A new hope for NEON made simple," *IEEE Trans. Comput.*, vol. 67, no. 11, pp. 1651–1662, Nov. 2018.

[14] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two party computation is practical," in *Proc. Int. Conf. Theory Appl. Cryptol. Inform. Secur.*, 2009, pp. 250–267.

[15] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. Annu. Conf. Adv. Cryptol.*, 2011, pp. 505–524.

[16] J. M. Pollard, "The fast Fourier transform in a finite field," *Math. Comput.*, vol. 25, pp. 365–374, 1971.

[17] D. D. Chen *et al.*, "High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 62, no. 1, pp. 157–166, Jan. 2015.

[18] C. P. Rentería-Mejía and J. Velasco-Medina, "High-throughput ring-LWE cryptoprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2332–2345, Aug. 2017.

[19] T. Oder, T. Schneider, T. Poppelmann, and T. Guneysu, "Practical CCA2-secure and masked ring-LWE implementation," *IACR Trans. Cryptographic Hardw. Embedded Syst.*, vol. 2018, pp. 142–174, 2018.

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

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

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

[23] S. Saha, U. Kumar, D. Mukhopadhyay, and P. Dasgupta, "An automated framework for exploitable fault identification in block ciphers," *J. Cryptographic Eng.*, vol. 9, no. 3, pp. 203–219, 2019.

[24] S. Patranabis, A. Chakraborty, and D. Mukhopadhyay, "Fault tolerant infective countermeasure for AES," *J. Hardw. Syst. Secur.*, vol. 1, no. 1, pp. 3–17, 2017.

[25] M. M. Kermani, R. Azarderakhsh, and A. Aghaie, "Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC,"*ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 2, 2019, Art. no. 59.

[26] A. Sarker, M. M. Kermani, and R. Azarderakhsh, "Hardware constructions for error detection of number-theoretic transform utilized in secure cryptographic architectures" *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 3, pp. 738–741, Mar. 2019.

[27] L. Breveglieri, I. Koren, and P. Maistri, "An operation-centered approach to fault detection in symmetric cryptography ciphers," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 534–540, May 2007.

[28] A. Kamal and A. Youssef, "Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks," *J. Cryptographic Eng.*, vol. 3, no. 4, pp. 227–240, May 2013.

[29] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *Proc IEEE Int. Symp. Circuits Syst.*, 2014, pp. 2796–2799.

[30] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede, "Efficient ring-LWE encryption on 8-bit AVR processors," in *Proc. Int. Workshop Cryptographic Hardw. Embedded Syst.*, 2015, pp. 663–682.

[31] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Develop.*, vol. 6, no. 2, pp. 200–209, Apr. 1962.

[32] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pp. 2957–2969, Oct. 2004.

**Mehran Mozaffari Kermani** (Senior Member, IEEE) received the B.Sc. degree from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively, all in electrical and computer engineering.

He joined the Advanced Micro Devices in 2011 as a Senior ASIC/Layout Designer, Integrating Sophisticated Security/Cryptographic Capabilities into accelerated processing. In 2012, he joined the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, as a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Research Fellow. From 2013 to 2017, he was an Assistant Professor with the Rochester Institute of Technology, Rochester, NY, USA. In 2017, he joined the Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA.

Dr. Kermani was a recipient of the prestigious NSERC Postdoctoral Research Fellowship in 2011 and the Texas Instruments Faculty Award (Douglas Harvey) in 2014. He is an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Embedded Computing Systems*, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS, and the Guest Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations 2016 and 2017. He was the lead Guest Editor for the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING for special issues on security. He has been the Technical Program Committee Member for a number of conferences, including the IEEE International Symposium on Hardware Oriented Security and Trust (Publications Chair), ACM Conference on Computer and Communications Security (Publications Chair), the Design Automation Conference, Design, Automation and Test in Europe Conference and Exhibition, International Workshop on Radio Frequency Identification: Security and Privacy Issues, International Workshop on Lightweight Cryptography for Security and Privacy, the International Workshop on the Arithmetic of Finite Fields, Workshop on Fault Diagnosis and Tolerance in Cryptography, and IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems.

**Reza Azarderakhsh** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Western University, London, ON, Canada, in 2011.

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

Dr. Azarderakhsh was a recipient of the Natural Sciences and Engineering Research Council Postdoctoral Research Fellowship while working with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON. He was the Guest Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations 2016 and 2017. He was also the Guest Editor for the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS for special issue on security. He is an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS.

**Ausmita Sarker** (Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2016. She is currently working towards the Ph.D. degree with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA.

Her current research interests include cryptographic hardware, post-quantum cryptography, and embedded systems security.