KINA: Karatsuba Initiated Novel Accelerator for Ring-Binary-LWE (RBLWE)-Based Post-Quantum Cryptography

Pengzhou He, *Graduate Student Member, IEEE*, Yazheng Tu, *Graduate Student Member, IEEE*, Jiafeng Xie, *Senior Member, IEEE*, and H. S. Jacinto, *Member, IEEE*

***Abstract*— Along with the National Institute of Standards and Technology (NIST) post-quantum cryptography (PQC) standardization process, lightweight PQC-related research, and development have also gained substantial attention from the research community. Ring-binary-learning-with-errors (RBLWE), a ring variant of binary-LWE (BLWE), has been used to build a promising lightweight PQC scheme for emerging Internet-of-Things (IoT) and edge computing applications, namely the RBLWE-based encryption scheme (RBLWE-ENC). The parameter settings of RBLWE-ENC, however, are not in favor of deploying typical fast algorithms like number theoretic transform (NTT). Following this direction, in this work, we propose a Karatsuba initiated novel accelerator (KINA) for efficient implementation of RBLWE-ENC. Overall, we have made several coherent interdependent stages of efforts to carry out the proposed work: 1) we have innovatively used the Karatsuba algorithm (KA) to derive the major arithmetic operation of RBLWE-ENC into a new form for high-performance operation; 2) we have then effectively mapped the proposed algorithm into an efficient hardware accelerator with the help of a number of optimization techniques; and 3) we have also provided detailed complexity analysis and implementation comparison to demonstrate the superior performance of the proposed KINA, e.g., the proposed design with $u = 2$ involves 64.71% higher throughput and 15.37% less area-delay product (ADP) than the state-of-the-art design for $n = 512$ (Virtex-7). The proposed KINA offers flexible processing speed and is suitable for high-performance applications like IoT servers. This work is expected to be useful for lightweight PQC development.**

***Index Terms*— Karatsuba initiated novel accelerator (KINA), lightweight, polynomial multiplication, post-quantum cryptography (PQC), ring-binary-learning-with-errors (RBLWE).**

NOMENCLATURE

a	Public parameter (integer polynomial).
r_1, r_2, e_1, e_2, e_3	Binary polynomials (r_2: secret key; e_1, e_2, e_3: errors).
m	Message.
n	Scheme size.
q	Modulus.
$f(x)$	Ring polynomial ($f(x) = x^n + 1$).
u, v	$(n/2) = uv$ (u and v are integers).
B, W	Binary polynomial (algorithm derivation and hardware design).
D, T, Z	Integer polynomial (algorithm derivation and hardware design).

Manuscript received 24 April 2023; revised 22 June 2023; accepted 19 July 2023. Date of publication 21 August 2023; date of current version 27 September 2023. This work was supported by the Visiting Faculty Research Program at the Air Force Research Laboratory, Rome, NY, USA under Grant FA8750-20-3-1003. *(Corresponding authors: Jiafeng Xie; H. S. Jacinto.)*

Pengzhou He, Yazheng Tu, and Jiafeng Xie are with the Department of Electrical and Computer Engineering, Villanova University, Villanova, PA 19085 USA (e-mail: phe@villanova.edu; ytu1@villanova.edu; jiafeng.xie@villanova.edu).

H. S. Jacinto is with the Air Force Research Laboratory Information Directorate, Rome, NY 13441 USA (e-mail: h.jacinto@afrl.af.mil).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2023.3302289.

Digital Object Identifier 10.1109/TVLSI.2023.3302289

I. INTRODUCTION

IT HAS been proven that the current public-key cryptosystems such as Rivest Shamir Adleman (RSA) and elliptic curve cryptography (ECC) can be broken by Shor's algorithm [1], [2] operated on a large-scale quantum computer [1]. As it is predicted that the well-established quantum computer will be available in the not far future, the research community has already started designing next-generation cryptosystems [3], [4], [5], i.e., post-quantum cryptography (PQC). Indeed, the National Institute of Standards and Technology (NIST) already initiated the PQC standardization process [5], and the lattice-based PQC has been regarded as one of the most important categories of PQC schemes [5], [6].

Many of the lattice-based PQC are based on the learning-with-errors (LWE) problem [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. While the ongoing NIST PQC standardization process targets general-purpose applications [5], [6], there is also a need to develop lightweight PQC algorithms. This is also confirmed by the very recent National Science Foundation (NSF) Secure and Trustworthy Cyberspace Principal Investigators' Meeting 2022 (SaTC PI Meeting'22) that one of the future research directions is the "lightweight PQC" [18]. Fortunately, some preliminary works on lightweight lattice-based PQC have already been carried out. In an earlier article, it is shown that for the LWE problem based on binary errors [19], [20], [22], [23], [24], i.e., binary-LWE (BLWE), the hardness of the lattice problem still remains [19] and can be used to build lightweight PQC. Following this proof, the ring variant of BLWE, RBLWE, is introduced to obtain a smaller computational complexity than the regular Ring-LWE-based PQC [24]. RBLWE-based encryption scheme (RBLWE-ENC) is based on the average-case hardness of the RBLWE problem,

and detailed security analysis has shown that it is secure enough for lightweight applications [24].

A. Existing Works

After the initial introduction, Buchmann et al. [24] reported the software implementation results. While on the hardware platform: 1) the first hardware implementation of the RBLWE-ENC was released in [25]; 2) the second hardware design for RBLWE-ENC was reported in [26]; 3) a high-speed hardware structure (but incomplete) was then reported in [27]; 4) a compact RBLWE-ENC hardware architecture was presented in [28]; 5) a lookup table (LUT)-like method-based hardware architecture was reported in [29]; 6) another compact structure for RBLWE-ENC was presented in [30]; 7) a new high-speed hardware RBLWE-ENC was introduced in [31]; 8) a pair of low-speed and high-speed RBLWE-ENC hardware accelerators were presented in [32]; and 9) efficient hardware RBLWE-ENC architectures were also recently reported in [33], [34], and [35], respectively. Meanwhile, there also exist other types of implementations like the fault detection scheme of [36] (based on the high-speed structure in [26]). These reports represent the major works in the field.

B. Challenges

The main operation of RBLWE-ENC is a particular polynomial multiplication over the ring $\mathbb{Z}_q/(x^n+1)$, where one polynomial involves merely binary values and another polynomial consists of integer coefficients. This particular setup is not desirable for the direct deployment of a fast algorithm (e.g., Karatsuba) as the addition-related iterative operations will increase the small-size coefficient involved in processing bit-width, which might offset the gain from deploying a fast algorithm. Meanwhile, the parameter settings of the RBLWE-ENC are not in favor of employing another widely used fast algorithm, i.e., number theoretic transform (NTT) [37]. In fact, the existing implementations of RBLWE-ENC are all based on the schoolbook polynomial multiplication (complexity of $O(n^2)$, e.g., [25], [26], [27]). For resource-constrained applications, the schoolbook-based method may still be a good choice as it allows the basic point-wise operations to obtain compact implementation [32], [35]. While for high-performance applications like the Internet-of-Things (IoT) servers that contain enough resources (e.g., field-programmable gate array (FPGA) devices), we prefer to accelerate RBLWE-ENC based on a hardware implementation strategy as it not only offers high-speed operation but also provides opportunities to be further developed into specific integrated circuits. In this case, the schoolbook-based design strategy can be further improved to obtain better performance, i.e., better area-time complexities.

C. Major Contributions

Based on the aforementioned considerations, in this article, we propose to introduce a Karatsuba initiated novel accelerator (KINA) for efficient implementation of RBLWE-ENC. We have carried out three steps of efforts to finalize the proposed work (main contributions) as follows.

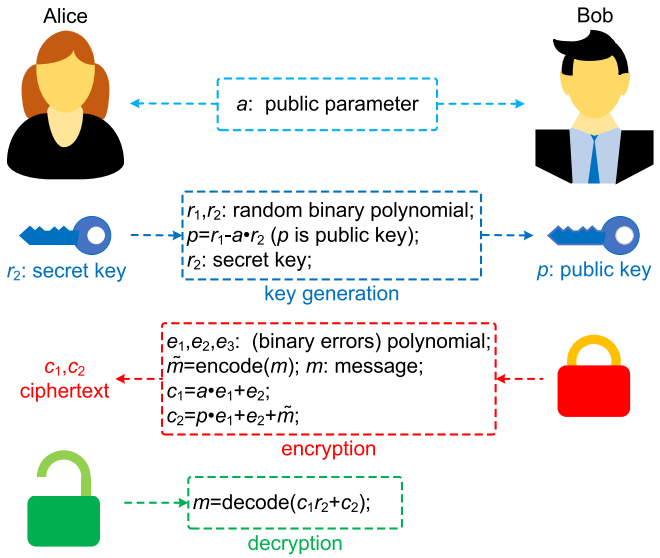

Fig. 1. Major phases of RBLWE-ENC.

1) We have used the Karatsuba algorithm (KA) to derive the polynomial multiplication of RBLWE-ENC into a new algorithm for high-speed processing.
2) We have then mapped the proposed polynomial multiplication algorithm into a new RBLWE-ENC hardware accelerator (KINA) with the help of a number of optimization techniques.
3) We have conducted thorough complexity analysis and comparison to confirm the efficiency of the proposed RBLWE-ENC accelerator (KINA).

Note that though KA-based polynomial multiplication is a standard technique for the LWE-based scheme, how to efficiently employ this technique to obtain high-speed processing of RBLWE-ENC has not been explored in the literature. To the authors' best knowledge, the proposed KINA is the first report about the KA-based RBLWE-ENC accelerator with flexible processing speed for different high-performance applications.

The rest of the article is organized as follows. Section II gives brief preliminaries. The proposed algorithm is presented in Section III. The hardware accelerator is introduced in Section IV. The complexity analysis and comparison are presented in Section V. Related works and future research are described in Section VI. The conclusion is given in Section VII.

II. Preliminaries

A. Ring-Binary-Learning-With-Errors-Based PQC

RBLWE-ENC consists of three main phases [24], [25], [26]: key generation, encryption, and decryption, as shown in Fig. 1. Major notations are also listed in Nomenclature.

1) *Key Generation:* The key generation is based on $p = r_1 - a \cdot r_2$, where p is the public key that will be sent to Bob (r_1 will then be discarded). In this phase, the secret and public keys have n and $n\log_2 q$ bits, respectively.
2) *Encryption:* The message binary polynomial m ($m = m_0 + \cdots + m_{n-1}x^{n-1}$) is firstly encoded into $\tilde{m}$ based on (1). Then, three binary polynomials (errors) e_1, e_2,

and e_3 will be used to produce the ciphertext c_1 and c_2 for Alice (the length of the ciphertext is $2n\log_2 q$ bits)

$$(m_0, \ldots, m_{n-1}) \rightarrow \sum_{i=0}^{n-1} m_i \left(\frac{q}{2}\right) x^i. \tag{1}$$

3) *Decryption:* In this phase, Alice recovers the encoded message (using secret key r_2) from the original m. Of course, a threshold decoder function [24] will be employed to generate the final output: the output will be "1" if the coefficient of the obtained value lies in the range of $(q/4, 3q/4)$, otherwise the outcome will be "0".

The recent report of [26] proposed an inverted RBLWE-based scheme, i.e., the coefficients of the polynomials are represented in the inverted range of $(-\lfloor (q/2) \rfloor, \lfloor (q/2) \rfloor - 1)$ such that all the modular operations can be performed naturally under the two's complement form. The three phases of Fig. 1 under this strategy remain the same, except $(m_0, \ldots, m_{n-1}) \rightarrow \sum_{i=0}^{n-1} m_i(-(q/2))x^i$, and the final decode function (opposite of the original one). In this article, we also adopt this strategy.

Security Level and Parameter Sets: BLWE with a restricted number of samples retains the worst case hardness of the LWE problem [19], while RBLWE-ENC is based on the average-case hardness of RBLWE. A relatively recent security analysis has estimated that RBLWE-ENC achieves 73/84 and 140/190 quantum/classic security bits for the parameter settings of $(n, q) = (256, 256)$ and $(n, q) = (512, 256)$, respectively [21], [22]. In this article, we follow the existing reports [22], [24] to use these parameter sets for possible lightweight applications.

B. KA: Karatsuba Algorithm (Binary Field)

The typical two-term KA-like method over binary field is as follows [38], [39], [40], [41] (where $A'_L = \sum_{i=0}^{(n/2)-1} a'_{2i} x^{2i}$, $A'_H = \sum_{i=0}^{(n/2)-1} a'_{2i+1} x^{2i}$, the same as B')

$$A' = A'_L + x A'_H, \quad B' = B'_L + x B'_H. \tag{2}$$

Then, define C' as the product of A' and B' such that

$$C' = \left\{(1+x)A'_L B'_L + \left(x^2 + x\right)A'_H B'_H + x\left[\left(A'_L + A'_H\right)\left(B'_L + B'_H\right)\right]\right\} \bmod f'(x) \tag{3}$$

where $C' = \sum_{i=0}^{n-1} c'_i x^i$ and a'_i, b'_i ,*and* $c'_i \in \{0, 1\}$ ($f'(x)$ is the field polynomial). Equation (3) can be iteratively applied to the polynomial multiplication to obtain subquadratic complexity.

III. KINA: From Mathematical Derivation to Strategy Formulation

A. Major Challenges

The iterative deployment of KA on the polynomial multiplication involves parallel computation and thus is not ideal for hardware implementation (the resource usage will be too large). Meanwhile, as mentioned in the Challenges of Section I, the small-size coefficient-involved processing bit-width will bring extra overhead even if we choose only a very small number of iterative deployments.

B. Overall Principle

We thus decided to use only the two-term decomposition of (3) so that the small-size coefficient-related computation will not cause large overhead. Still, the direct mapping of the two-term KA into hardware will incur large resource usage. Therefore, we propose to: 1) process all the input–output in a serial-in and serial-out format (practical for deploying in actual applications) and 2) compute the major arithmetic procedure in an accumulation format to save the resource usage (hardware implementation friendly). The steps below have strictly followed this principle.

C. Extension of KA to RBLWE-ENC

It is obvious that the main operation of the RBLWE-ENC is the polynomial multiplication, which can be defined as

$$T = BD \bmod f(x) \tag{4}$$

where $B = \sum_{i=0}^{n-1} b_i x^i$, $D = \sum_{i=0}^{n-1} d_i x^i$, and $T = \sum_{i=0}^{n-1} t_i x^i$ for d_i and t_i are integers in $\mathbb{Z}_q$ and $b_i \in \{0, 1\}$. Then, we have [follow (3)]

$$B = B_L + x B_H, \quad D = D_L + x D_H \tag{5}$$

where $B_L = \sum_{i=0}^{(n/2)-1} b_{2i} x^{2i}$, $B_H = \sum_{i=0}^{(n/2)-1} b_{2i+1} x^{2i}$, $D_L = \sum_{i=0}^{(n/2)-1} d_{2i} x^{2i}$, and $D_H = \sum_{i=0}^{(n/2)-1} d_{2i+1} x^{2i}$.

Then, we can have

$$\begin{aligned}
&(B_L + x B_H)(D_L + x D_H) \bmod f(x) \\
&= B_L D_L + x(B_L D_H + B_H D_L) + x^2 B_H D_H \bmod f(x) \\
&= B_L D_L + x[(B_L + B_H)(D_L + D_H) - B_L D_L - B_H D_H] \\
&\quad + x^2 B_H D_H \bmod f(x) \\
&= (1-x) B_L D_L + x[(B_L + B_H)(D_L + D_H)] \\
&\quad + \left(x^2 - x\right) B_H D_H \bmod f(x)
\end{aligned} \tag{6}$$

where the original n-size polynomial multiplication becomes the addition of three $n/2$-size subpolynomial multiplications.

D. Proposed Algorithmic Derivation

Define $B_M = B_L + B_H$ and $D_M = D_L + D_H$ such that (6) can be [42]

$$\begin{aligned}
T &= (1-x)T_L + \left(x^2 - x\right)T_H + x T_M \bmod f(x) \\
&= (1-x)T_L \bmod f(x) + \left(x^2 - x\right)T_H \bmod f(x) \\
&\quad + x T_M \bmod f(x)
\end{aligned} \tag{7}$$

where $T_L = B_L D_L$, $T_H = B_H D_H$, and $T_M = B_M D_M$.

1) Detailed Steps to Derive T_L: Let us consider T_L first,

$$T_L = B_L D_L \bmod f(x) \tag{8}$$

which can be rewritten as

$$\begin{aligned} T_L &= \left(b_0 + b_2x^2 + \cdots + b_{n-2}x^{n-2}\right) \\ &\quad \times \left(d_0 + d_2x^2 + \cdots + d_{n-2}x^{n-2}\right) \\ &= d_0\left(b_0 + b_2x^2 + \cdots + b_{n-2}x^{n-2}\right) \\ &\quad + d_2x^2\left(b_0 + b_2x^2 + \cdots + b_{n-2}x^{n-2}\right) \\ &\quad + \cdots \\ &\quad + d_{n-2}x^{n-2}\left(b_0 + b_2x^2 + \cdots + b_{n-2}x^{n-2}\right) \end{aligned} \tag{9}$$

which can be expressed as an equivalent matrix-vector product

$$\begin{aligned} [T_L] &= \begin{bmatrix} b_0 & & & & \\ b_2 & b_0 & & & \\ b_4 & b_2 & b_0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ b_{n-2} & b_{n-4} & b_{n-6} & \ddots & b_0 \\ & b_{n-2} & b_{n-4} & \ddots & b_2 \\ & & b_{n-2} & \ddots & b_4 \\ & & & \ddots & \vdots \\ & & & & b_{n-2} \end{bmatrix} \times \begin{bmatrix} d_0 \\ d_2 \\ d_4 \\ \vdots \\ d_{n-2} \end{bmatrix} \\ &= [B_L][D_L] \end{aligned} \tag{10}$$

which can be seen as a $(n-1) \times (n/2)$ circulant matrix-vector product (the blank parts are actually "0"s). For instance, the second column of $[B_L]$ (from left, $(n/2)$ nonzero elements) is the circularly shifted version of the first column (circularly downward by one position), so are the rest of the columns.

2) Computation Strategy for $[T_L]$*:* The direct implementation of (10), however, will involve too much resource usage, and hence we propose to compute the matrix-vector product of (10) into column-wise accumulations, i.e., the elements of one column of $[B_L]$ are multiplied with one corresponding element of vector-matrix $[D_L]$ and then accumulated with the next column-based similar operation (so on and so forth). As the nonzero elements in each column of $[B_L]$ are identical, we can hence share these coefficients during the accumulation process while the elements of $[D_L]$ can be fed in a serial format. Note that we can also process multiple columns of $[B_L]$ at the same time (with related elements of $[D_L]$) for higher speed applications.

There are in total $(n/2)$ columns in the matrix $[B_L]$, and we can thus define the first column (from left) of $[B_L]$ as $[B_L]_1$, the second column as $[B_L]_2, \ldots, [B_L]_{(n/2)}$. We also define the elements of vector-matrix $[D_L]$ as: $[D_L]_{1,1} = d_0$, $[D_L]_{1,2} = d_1, \ldots, [D_L]_{1,(n/2)} = d_{n-2}$. Thus, (10) can be

$$[T_L] = \sum_{i=1}^{\frac{n}{2}} [B_L]_i [D_L]_{1,i} \tag{11}$$

where (10) becomes the form of column-wise accumulations.

Define $(n/2) = uv$ (u and v are integers). Then, (11) becomes

$$[T_L] = \sum_{j=1}^{v} \sum_{i=1}^{u} [B_L]_{ju+i} [D_L]_{1,ju+i} \tag{12}$$

which has v groups of accumulation that each group has u items of $[B_L]_{ju+i}[D_L]_{1,ju+i}$. These u computations can be executed at the same time to speed up the overall processing.

3) Computation for $[T_H]$ *and* $[T_M]$*:* Similarly, we have

$$\begin{aligned} [T_H] &= \begin{bmatrix} b_1 & & & & \\ b_3 & b_1 & & & \\ \vdots & \vdots & \vdots & \ddots & \\ b_{n-1} & b_{n-3} & b_{n-5} & \ddots & b_1 \\ & b_{n-1} & b_{n-3} & \ddots & b_3 \\ & & b_{n-1} & \ddots & b_5 \\ & & & \ddots & \vdots \\ & & & & b_{n-1} \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_3 \\ d_5 \\ \vdots \\ d_{n-1} \end{bmatrix} \\ &= [B_H][D_H]. \end{aligned} \tag{13}$$

Similarly, one can easily follow the same strategy for $[T_L]$ and $[T_H]$ to compute $[T_M]$.

4) Final Recombination: As seen from (7), each term of the KA-deployed polynomial multiplication has factors such as $(1-x)$, x, and (x^2-x), which involves position-shifting-based additions and hence is not hardware implementation friendly (if we calculate all related coefficients at the same time). To save resource usage, we can arrange the coefficients of the final output of T to be delivered out in a serial format.

Specifically, we can define $T_L = \sum_{i=0}^{n-2} t_{L,i}x^{2i}$, $T_H = \sum_{i=0}^{n-2} t_{H,i}x^{2i}$, and $T_m = \sum_{i=0}^{n-2} t_{M,i}x^{2i}$.

Connecting with (7), we can have

$$\begin{aligned} &T_L(1-x) + T_H\left(x^2 - x\right) + T_M x \bmod f(x) \\ &= t_{L,0} + \left(t_{M,0} - t_{L,0} - t_{H,0}\right)x \\ &\quad + \left(t_{L,1} - t_{H,0}\right)x^2 + \left(t_{M,1} - t_{H,1} - t_{L,1}\right)x^3 \\ &\quad + \cdots \\ &\quad + \left(t_{H,n-2}\right)x^{2n-2} \bmod f(x) \end{aligned} \tag{14}$$

which can be substituted with $x^n \equiv -1$ to have

$$\begin{aligned} &T_L(1-x) + T_H\left(x^2 - x\right) + T_M x \bmod f(x) \\ &= \left(t_{L,0} - t_{L,\frac{n}{2}} - t_{H,\frac{n}{2}-1}\right) \\ &\quad + \left(-t_{L,0} + t_{L,\frac{n}{2}} + t_{H,\frac{n}{2}} - t_{H,0} + t_{M,0} - t_{M,\frac{n}{2}}\right)x \\ &\quad + \left(t_{L,1} - t_{L,\frac{n}{2}+1} + t_{H,0} - t_{H,\frac{n}{2}}\right)x^2 \\ &\quad + \cdots \\ &\quad + \left(-t_{L,\frac{n}{2}-2} + t_{H,n-2} - t_{H,\frac{n}{2}-2} + t_{M,\frac{n}{2}-2} - t_{M,n-2}\right)x^{n-3} \\ &\quad + \left(t_{L,\frac{n}{2}-1} + t_{H,\frac{n}{2}-2} - t_{H,n-2} + t_{M,\frac{n-2}{2}}\right)x^{n-2} \\ &\quad + \left(-t_{L,\frac{n}{2}-1} - t_{H,\frac{n}{2}-1} + t_{M,\frac{n}{2}-1}\right)x^{n-1} \\ &= t_0 + t_1x + \cdots + t_{n-1}x^{n-1} \end{aligned} \tag{15}$$

where it is found that the final output coefficients t_i are just the addition of the corresponding values of $t_{L,i}$ and/or $t_{H,i}$ and/or $t_{M,i}$. The actual computation process can be seen in the corresponding hardware component section in Section IV.

5) Final Algorithm Formulation: Based on the above mathematical derivation of (4)–(15), we can have the proposed KA-based polynomial multiplication algorithm as follows.

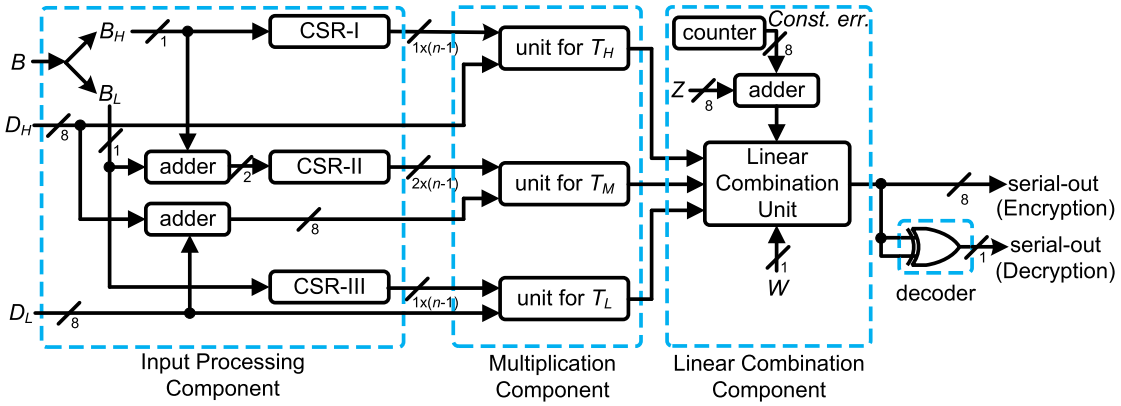

Fig. 2. Proposed RBLWE-ENC accelerator (KINA, $u = 1$), where Z and W denote the additional two polynomials (Z is an integer polynomial and W is a binary polynomial. CSR: circular shift-register. A constant error (Const. err.), delivered from a counter, is also needed when executing the addition with the integer polynomial Z, following the suggestion of [34].

Algorithm 1 Algorithm of KA-Based Polynomial Multiplication for RBLWE-ENC

Input : B, and D are polynomials; D has coefficients $\in \mathbb{Z}_q$ and B is a binary polynomial.
Output: $T = BD \bmod (x^n + 1)$.

Initialization step
1 Make ready the inputs B and D; // $B = B_L + B_H$ and $D = D_L + D_H$
2 $\overline{[T_1]} = \overline{[T_2]} = \overline{[T_3]} = 0$; // $\overline{[T_1]}$, $\overline{[T_2]}$, and $\overline{[T_3]}$ are the same size matrices as $[T_L]$, $[T_H]$, and $[T_M]$, respectively
Main step
3 **for** $j = 1$ *to* v **do**
4 **for** $i = 1$ *to* u **do**
5 $\overline{T_1} = \overline{T_1} + [B_L]_{ju+i}[D_L]_{1,ju+i}$; // see (12)
6 $\overline{T_2} = \overline{T_2} + [B_H]_{ju+i}[D_H]_{1,ju+i}$; // follow (12)
7 $\overline{T_3} = \overline{T_3} + [B_M]_{ju+i}[D_M]_{1,ju+i}$; // follow (12)
8 **end**
9 **end**
10 $[T_L] = \overline{[T_1]}$, $[T_H] = \overline{[T_2]}$, and $[T_M] = \overline{[T_3]}$;
Final step
11 $T = T_L(1-x)+T_H(x^2-x)+T_M x \bmod f(x)$; // follow (15) and deliver all the coefficients of T serially;

Where T_L, T_H, and T_M are processed in parallel to obtain high-performance computation. Note that for RBLWE-ENC, we also need to consider the operations related to other polynomials like the additions with an integer polynomial as well as the decoder function in the decryption phase. The following section will cover details of the hardware accelerator.

IV. KINA: Proposed RBLWE-Based PQC Accelerator

Following Algorithm 1 of Section III, we can have the proposed RBLWE-ENC accelerator (KINA) as described below. As shown in Fig. 1, the major arithmetic operation involved within RBLWE-ENC includes a polynomial multiplication followed by the addition with two polynomials (e.g., ciphertext c_2 in the encryption phase), which can be extended to the operations in other phases (the subtraction in the key generation can be easily realized by the hardware implementation). Thus, this major arithmetic operation, one polynomial multiplication along with the additions with two other polynomials, is used to construct the proposed accelerator. Meanwhile, we have also presented a higher speed version of KINA (when $u > 1$). Finally, we have also proposed several optimization techniques to further maximize the design efficiency.

A. KINA: Proposed RBLWE-ENC Accelerator

1) Architectural Overview: As shown in Fig. 2, the proposed RBLWE-ENC accelerator consists of three major components, namely the input processing component, the multiplication component, and the linear combination component. During the actual execution process, e.g., encryption phase or decryption phase, inputs B and D are firstly decomposed based on (5), and then loaded into the corresponding shift-register in the input processing component, i.e., D_L/D_H and B_L/B_H. Besides, two adders are needed to produce the corresponding B_M and D_M. After that, three computational units in the multiplication component, T_L, T_M, and T_H, take the processed coefficients as input and execute point-wise multiplications of the related coefficients as well as the accumulation of the matrix-vector products. When this multiplication step is executed, the three results are then delivered to the linear combination component to produce the final results according to Line-11 of Algorithm 1 in Section III (see (15) as well). The final result will be delivered out in a serial format until the whole computation process is finished. It is noted that the output for the encryption phase is 8 bit, while the output for decryption is 1 bit. The detailed internal structures and related functions of these components in Fig. 2 are described below.

2) Input Processing Component: The input processing component is responsible for loading and delivering the decomposed coefficients of B_L, B_H, D_L, and D_H, as well as the producing and delivering of correct coefficients of B_M and D_M to the following multiplication component. Overall, the input processing component consists of three circular shift-registers [CSRs, length of $(n-1)$] and two adders (different

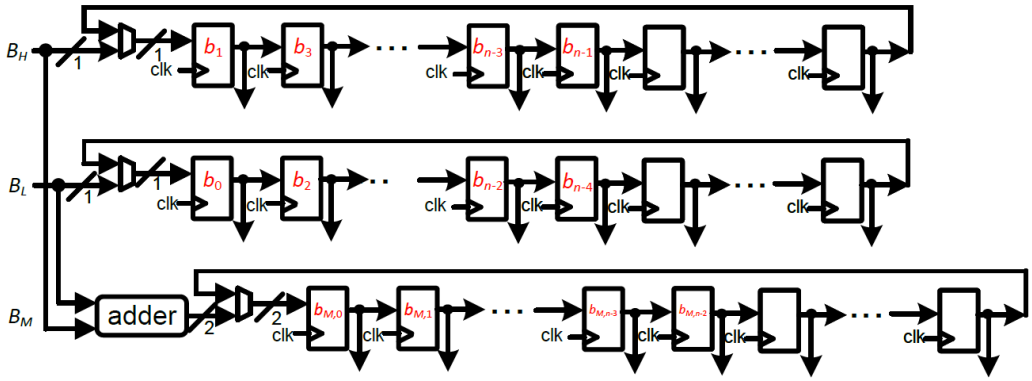

Fig. 3. Details of the CSRs for B_H, B_L, and B_M (from up), respectively.

sizes), as seen from Fig. 3. Overall, it takes $(n/2)$ cycles to load in coefficients of B_L and B_H, which work in parallel with the multiplication process. Meanwhile, the serially loaded coefficients of B_L and B_H will pass through a 1-bit adder to produce the corresponding coefficient of B_M to be loaded into the 2-bit CSR. When all the $(n/2)$ coefficients are loaded into the corresponding CSRs, the rest $(n/2)$ registers are still set as "0"s. The MUXes inside the CSRs then close the loop, and all the related coefficients will be circularly shifted, which function the same as the feature of "circularly downward-shifted columns" in $[T_L]$, $[T_H]$, and $[T_M]$ [see (10) and (13)]. Meanwhile, the values of D_L, D_H, and D_M are serially fed into the accelerator, which matches the column-based accumulation in Lines 5–7 of Algorithm 1. Note the output of all $(n-1)$ registers of three CSRs are connected to the related processing units in parallel.

3) Multiplication Component: The multiplication component consists of three parallel processing units for T_H, T_L, and T_M, respectively. The internal structures of the processing units for T_H/T_L and T_M are shown in Fig. 4(a) and (b), respectively. As the computation processes of T_L and T_H are very similar [see (10) and (13)], we just use one set of the internal structure to illustrate the detailed design. Basically, the $(n-1)$ bits from related CSRs are fed to $(n-1)$ sets of a point-wise multiplier followed by an accumulator, where the point-wise multiplier is executed by an 8-bit AND cell [Fig. 4(a)] and the accumulator contains an adder followed by a register (the output of the register is also used as another input of the adder to form the loop). Thus, after $(n/2)$ cycles' accumulations, the output of the register becomes $t_{L,i}$ or $t_{H,i}$ $(0 \le i \le n-2)$. Note that we have also inserted a 2-to-1 MUX in the middle of the accumulator such that these final output values of $t_{L,i}$ or $t_{H,i}$ $(0 \le i \le n-2)$ can be circularly shifted, which facilitates the delivering of the final output in a serial format (see the linear combination component). The internal structure of the processing unit of T_M is almost the same as T_L and T_H, except that one input to the point-wise multiplier has now become 2 bit, which can be realized by a MUX-based design as shown in Fig. 4(c). Three precalculated values (based on the input $d_{M,i}$, where $d_{M,i}$ represents the corresponding coefficients of D_M), i.e., "0," "$d_{M,i}$," and "$2d_{M,i}$," are attached to the MUX, and the result will be determined according to the 2-bit $b_{M,i}$ ($b_{M,i}$ is the corresponding coefficient of B_M). After being accumulated, the output of the T_M unit is delivered to the Linear Combination Unit together with the outcomes of the other two units for the calculation of the final result T. Note that the connected $(n-1)$ registers in the accumulators, through the insertion of MUXes, and functions as a shift-register to provide correct input signals for the following linear combination component to produce the correct output in a serial format.

Case Example: For a clear demonstration, we have also used a case example of $n = 8$. Connecting with (10), we have

$$[T_L] = \begin{bmatrix} b_0 & & & \\ b_2 & b_0 & & \\ b_4 & b_2 & b_0 & \\ b_6 & b_4 & b_2 & b_0 \\ & b_6 & b_4 & b_2 \\ & & b_6 & b_4 \\ & & & b_6 \end{bmatrix} \times \begin{bmatrix} d_0 \\ d_2 \\ d_4 \\ d_6 \end{bmatrix} = [B_L][D_L]. \quad (16)$$

Meanwhile, the output of the CSR for T_L is "$b_0, b_2, b_4, b_6, 0, 0, 0, 0$," which becomes "$0, b_0, b_2, b_4, b_6, 0, 0, 0$" in the next cycle, "$0, 0, b_0, b_2, b_4, b_6, 0, 0$" in the third cycle, and finally "$0, 0, 0, b_0, b_2, b_4, b_6$" in the last cycle. This process exactly matches the column-based accumulations of (10). Since each column of the matrix in (16) takes one cycle, the entire multiplication needs four cycles. Based on this case example, we can conclude that a total of $(n/2)$ cycles are needed for the multiplication component.

4) Linear Combination Component: The linear combination component is responsible for the calculation of the final result by combining the outputs of the three units in the multiplication component using linear operations (addition/subtraction) according to (14) and (15). As shown in Fig. 5, a set of sign inverters (SIs) and adders are used in this component to obtain the correct output. During the linear combination process at each cycle, corresponding values stored in the accumulators from the multiplication component are fed into the SIs and adders, respectively, according to the setup of Fig. 5 to obtain the desired output. Note the selection signals to the MUX are generated from the control unit. The linear combination component can be seen as the

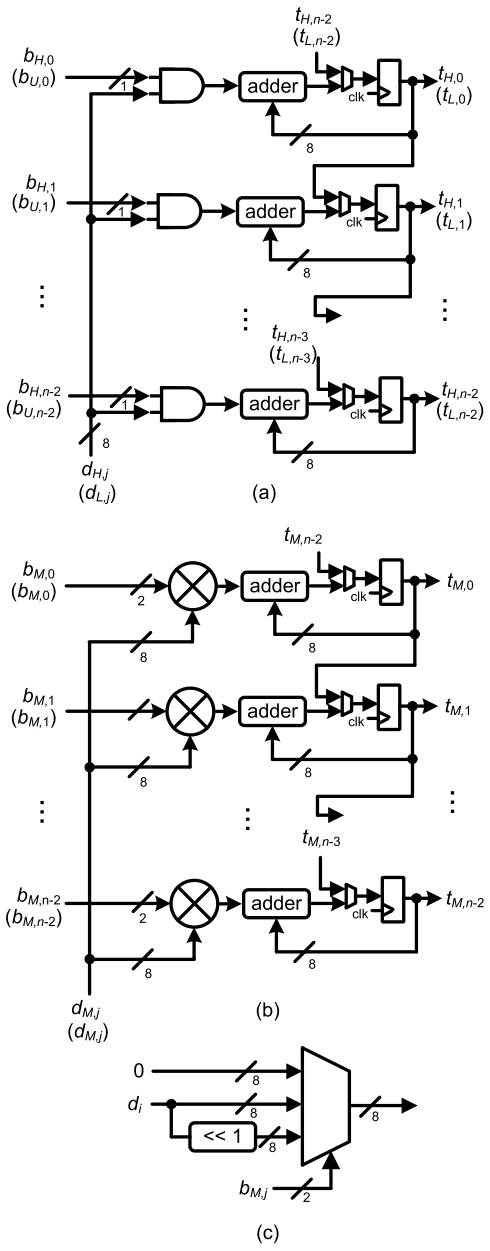

Fig. 4. (a) Processing unit for T_H or T_L. (b) Processing unit for T_M. (c) Point-wise multiplier for T_M.

output component, and it takes n cycles to deliver all output values.

Example of the computation inside the linear combination component: As seen from (15), e.g., the first term associated with $x^0 = 1$ is $(t_{L,0} - t_{L,(n/2)} - t_{H,(n/2)-1})$, which can be obtained through the SI and adders attached to $t_{L,0}$, $t_{L,(n/2)}$, and $t_{H,(n/2)-1}$ as well as related MUXes to deliver the desired coefficient, added with corresponding coefficients of $(Z +$ Cons. Err.) and W, to produce the final output t_0. Similarly, the other output values of T can be obtained through the coordination of SIs, adders, and MUXes.

5) Control Unit: Finally, a control unit is also required to coordinate the proper operation of the RBLWE-ENC accelerator. Specifically, this control unit is based on a finite state machine (FSM), where it involves five operational states,

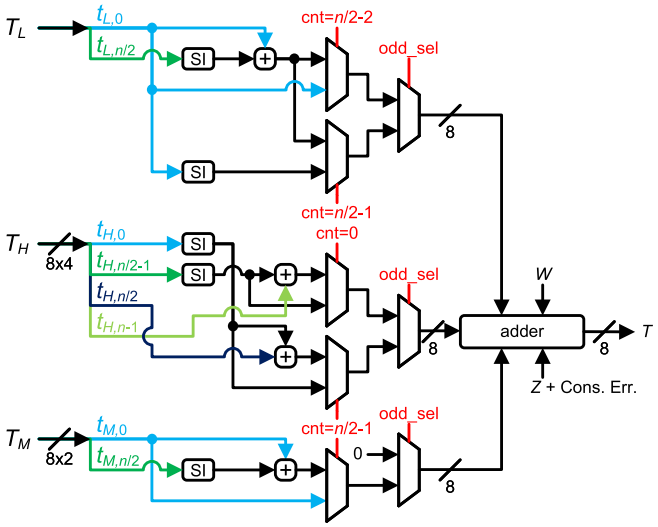

Fig. 5. Linear combination component, where SI denotes the sign inverter. Cons. Err.: constant error. The green and blue signals denote that they are connected with the corresponding registers in Fig. 4; while the red signals are control signals, e.g., "odd_sel" denotes that the MUX works in the lower channel when the odd order of output coefficient is selected to be produced when "odd_sel" = "1" (similar to the "cnt" control signals attached to the MUXes, e.g., "cnt = $N-1$" means that the MUX works in the lower channel when this counting signal is "1").

namely "clear/reset," "load," "compute," "output," and "done." During the "clear/reset" state, the signals and registers in the accelerator are cleared up for preparing the execution of the new task. Then, during the "load" state, all the signals are loaded into respective CSRs for the following "compute" state. While the following "compute" state executes the point-wise multiplication-based accumulation to obtain the desired result for T_L, T_H, and T_M. The next state is the "output" period, which executes the linear combination operation to obtain the desired output in a serial format following (15). After all the output coefficients t_i $(0 \leq i \leq n-1)$ are delivered, the accelerator will release the final "done" signal.

6) Overall Operation: After the proposed KINA loads the values of B_L and B_H into the CSRs ($(n/2)$ cycles), the related three units for T_L, T_M, and T_H need $(n/2)$ cycles to produce the correct output values (or decryption output) to be sent to the linear combination component for final serial outputting (n cycles). Overall, the computation time of the multiplication component can be viewed as the major latency of KINA.

B. KINA: Higher Speed Version

The RBLWE-ENC accelerator of Fig. 2 processes one column ($u = 1$) per cycle. For higher speed applications, however, we can increase the number of column-based accumulation, i.e., parallel processing based on larger u, to obtain a higher speed version of KINA.

The overall data flow of the proposed higher speed RBLWE-based accelerator is very similar to the basic version shown in Fig. 2. The coefficients of B and D are grouped into three groups, respectively, and are then fed into different multiplication units, the products go through the accumulation and permutation before they are sent to the linear combination component for the final calculation. The output size is also

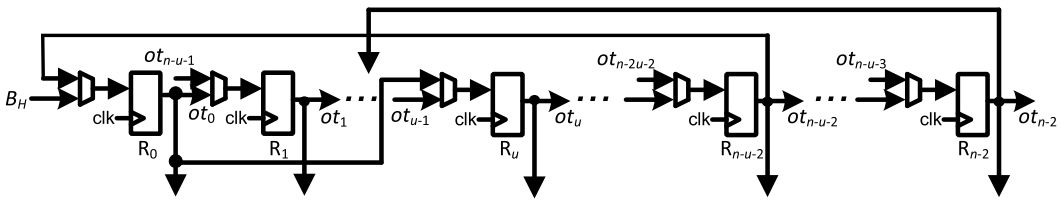

Fig. 6. Internal structure of the shift-register for B for the proposed higher speed KINA ("ot$_i$" denotes the output of the individual register).

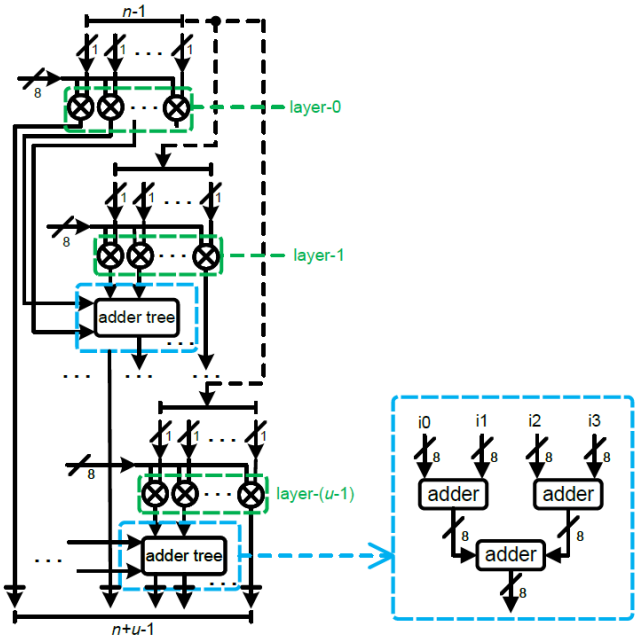

Fig. 7. Internal structure of the computation unit in the multiplication component, where each unit contains the adder trees (here we have used T_L and the input of B_L is 1-bit). Note that for T_M, we need to set the input (B_M) as 2-bit. Meanwhile, we have also demonstrated the internal structure for the adder tree for $u = 4$, which can be extended to different values of u.

8 or 1 bit for encryption/decryption phase. Nevertheless, slight modifications have been made in the three multiplication components as well as the shift-registers for D in the input processing component since u ($u = 2, 4, \ldots$) columns of B are involved in the calculation at the same cycle.

The modification made in the input processing component is that shift-registers for D are changed to u-parallel output instead of the original serial-out. This is because we need u of the coefficients delivered out at the same time for the parallel calculation. Also, the shift-registers do not shift one position per cycle—they shift u numbers every cycle, which is implemented by connecting the register to the uth one in front of its position in a circular format (and the internal structure is shown in Fig. 6). Another modification is made in the multiplication component. Unlike the one in the basic version of Fig. 2, which has only one layer of point-wise multiplier-based accumulator-sets, there are u layers of the multiplier accumulator-sets, where each set has $(n-1)$ point-wise multipliers and the products are sent to a special component, adder tree, for the addition (see Fig. 7). Each layer of point-wise multipliers takes $(n-1)$ coefficients of B and one coefficient of D as its inputs and calculates the products. A single adder tree takes u products as its inputs and calculates their summation as its output. After the multiplication and permutation are finished, the results will be delivered to the accumulators for the final linear combination. One unit of the multiplication component consists of $u \times (n-1)$ point-wise multipliers, u adder trees, and $(n+u-1)$ accumulators. Meanwhile, each adder tree contains $(n-2)$ adders.

Overall Operation of the Proposed Higher Speed KINA: The higher speed KINA, after all the values of B_L and B_H are loaded into the CSRs [$(n/2)$ cycles], the accelerator needs only $n/(2u)$ cycles to compute the T_L, T_M, and T_H at the cost of increased hardware usage in the multiplication component. The final output still needs n cycles (for the linear combination component). Thus, the major computation time of the higher speed KINA is $n/(2u)$ cycles.

V. Complexity and Comparison

A. Complexity Analysis

The area-time complexities of the proposed KINA (Fig. 2) are listed as follows.

1) The input processing component requires three CSRs (two CSRs, respectively, for B_L and B_H, contain $(n/2)$ 1-bit registers; one CSR for B_M needs $(n/2)$ 2-bit registers), one 1-bit adder, and one 8-bit adder.
2) The computation unit for T_H (and T_L) has $(n-1)$ AND gates, $(n-1)$ 8-bit adders, and $(n-1)$ 8-bit registers, while the computation unit for T_M has $(n-1)$ 3-to-1 MUXes, $(n-1)$ 8-bit adders, and $(n-1)$ 8-bit registers.
3) The linear combination component has 6 8-bit adders, 5 SIs, 8 8-bit MUXes, and an XOR gate. The proposed KINA takes $(n/2)$ cycles to execute related accumulations (multiplication component) and another n cycles to output the final results in a serial format.

For the higher speed design of KINA, the input processing component area consumption remains the same as the basic version. In the multiplication component, a total number of two $\times$ u $\times$ $(n-1)$ AND gates and u $\times$ $(n-1)$ MUXes are needed for the point-wise multiplications and 3 $\times$ u $\times$ $(n-1)$ 8-bit adders, as well as $3 \times (n-1) + \log_2 u \times (n-2)$ 8-bit registers, will be used for permutation and accumulation. The area usage of the linear combination component remains the same. Finally, the time required for the computation (multiplication component) decreases as u increases, i.e., $n/(2u)$ cycles.

The area-time complexities of the proposed designs (both the basic and higher speed versions), in terms of the number of AND gates, XOR gates, adders, MUXes, and latency cycles, are listed in Table I along with those of the existing high-speed designs of [25], [26], [31], [32], [33], and [34]. Note that

TABLE I
MAJOR AREA-TIME COMPLEXITIES FOR THE PROPOSED RBLWE-BASED ACCELERATOR AND THE STATE-OF-THE-ART STRUCTURES

Design	AND	Adder ($\widetilde{q}$-bit)	Register ($\widetilde{q}$-bit)	MUX ($\widetilde{q}$-bit)	Extra input/output resources	Latency*	Complexity
[25][2]	–	2n[1]	n	n	three $\widetilde{q}$-bit n-size SRs	n	$O(n^2)$
[26][2]	n	n	n	$n+1$	three $\widetilde{q}$-bit n-size SRs	n	$O(n^2)$
[32]	n	n	n	n	three $\widetilde{q}$-bit n-size SRs	n	$O(n^2)$
[33] Arch.-I	n	n	n	n	two $\widetilde{q}$-bit n-size SRs	n	$O(n^2)$
[34]	n	n	n	n	three $\widetilde{q}$-bit n-size SRs	n	$O(n^2)$
Prop. $u=1$	$2(n-1)$	$3(n-1)+3$	$\frac{9n}{2}-3$	$3n+5$	two 2-bit $\frac{n}{2}$-size SRs[3]	$\frac{n}{2}$	$O(\frac{3n^2}{4})$
Prop. $u>1$	$2u(n-1)$	$3u(n-1)+3$	$3\times(n-1)+\log_2 u\times(n-2)$	$3n+5$	two 2-bit $\frac{n}{2}$-size SRs[3]	$\frac{n}{2u}$	$O(\frac{3n^2}{4})$

$\widetilde{q}=\log_2 q$. $\frac{n}{2}=uv$.
The existing designs are mostly based on the structures for decryption phase, and hence we list the complexities estimated at this specific phase. Besides that, specific input/output processing resources are also specified, if they are available (SR: shift-register).
*: The latency cycles listed here refer to the major computation time based on the decryption phase, excluding the input loading and output delivery time.
[1]: The design needs n $\widetilde{q}$-bit subtractors and $\widetilde{q}$-bit adders.
[2]: These two designs have parallel-in parallel-out setup and the related input/output shift-registers are not specified. But from the designed structure, maybe three $\widetilde{q}$-bit n-size shift-registers are used (or multiple BRAMs) for [25]. Meanwhile, from the re-implemented result in [31], the design of [37] has two $\widetilde{q}$-bit n-size shift-registers and one $\widetilde{q}$-bit n-size output buffer.
[3]: The proposed KINA also needs two 1-bit $\frac{n}{2}$-size SRs and one 2-bit $\frac{n}{2}$-size SR, which equivalents to two 2-bit $\frac{n}{2}$-size SRs.

we have listed the major computation time (clock cycles) as the latency for all the designs, following what we have discussed in Section IV (which was also reported in these existing designs).

Note that we do not include the following designs since: 1) the design of [29] is a special design based on LUT-like method; 2) the structures of [28] and [30] belong to compact designs (similar to [43], which is a compact implementation of an approximate Ring-LWE based scheme); and 3) the designs in [31] and Architecture-II of [33] did not consider the input–output processing resources in structural design.

One can see that from Table I, the proposed designs overall have relatively larger area complexities than those of the existing ones because of the proposed Karatsuba-based design strategy that three processing components are processed in parallel. But as we consider the overall area-time complexities, it is very obvious that all the existing designs have a complexity of $O(n^2)$, while the proposed one can achieve $O((3n^2)/4)$. Finally, one has to mention that the listed complexities are based on theoretical estimation, and the corresponding implementation can reflect a more precise result.

B. FPGA Implementation Results and Related Comparison

1) Experimental Setup: While the comparison of area-time complexities listed in Table I is more on the theoretical side, there is a need for a more detailed comparison. Thus, we have implemented the proposed accelerator on the FPGA platform and the experiment is setup as follows: 1) we have coded the proposed RBLWE-ENC accelerator (KINA) with VHDL and have verified its functionality through ModelSim; 2) we have followed the existing strategies [25], [26], [31], [32], [33] to synthesize and implement the coded design on the Xilinx FPGAs (after place and route), i.e., Virtex-7 XC7V2000t and Kintex-7 XC7K325t devices, respectively, through Vivado 2020.2; 3) we have chosen the same parameter settings according to the existing designs of [25], [26], [31], [32], and [33], i.e., $(n,q)=(256,256)$ and $(n,q)=(512,256)$ ($\widetilde{q}=8$), which correspond to the quantum/classic security of 73/84 bits and 140/190 bits, respectively [22]; 4) the proposed accelerator also includes the third and fourth polynomials Z and W for operations of both encryption and decryption phases as well as related resources; 5) for a more general demonstration, we do not use the other available resources on the FPGA devices such as the block RAM (BRAM), etc.; 6) we have chosen $u=1$, $u=2$, $u=4$, $u=8$, and $u=16$ for the proposed KINA, respectively, to showcase the high-speed operational performance under different processing setups; and 7) the obtained area-time complexities, in terms of area usage (the number of (LUTs, registers (FFs), and slices), maximum frequency (Fmax, MHz), latency cycles, delay (critical-path $\times$ latency cycles), area-delay product (ADP), and throughput are all listed in Table II.

2) Discussion: From Table II, we can clearly see that the area consumption of the proposed KINA increases as u becomes bigger. This is because the number of adders and point-wise multipliers in the KINA is positively proportional to u, i.e., the number of involved adders and point-wise multipliers increases to execute the parallel calculation and permutation in the multiplication component. However, the latency drops significantly as u increases as more columns of $[D]$ ($[D_L]$, $[D_H]$, and $[D_M]$) can be involved in the calculation at the same time, and thus the number of cycles required for the multiplication process decreases rapidly. Finally, one can also find that the higher speed version of the proposed KINA has relatively better performance than the basic one. Meanwhile, in terms of the overall area-time complexities, the proposed design with $u=2$ obtains the best ADP among all the cases.

3) Comparison With the Existing RBLWE-ENC Implementations: The comparison with those of the state-of-the-art ones (i.e., [25], [26], [31], [32], [33] is seen in Table III). We have carefully considered the comparison setup, as described below. First of all, as the designs of [29] and [32] are reported on the Intel Straix-V device, we thus also obtained the performance of the proposed KINA ($u=2$) on the same Stratix-V device, as listed in Table IV. Secondly, we want to mention that some of the existing designs do not include the input processing component in the implementation, and hence we need careful

TABLE II
FPGA IMPLEMENTATION PERFORMANCE OF THE PROPOSED ACCELERATOR ON AMD-XILINX DEVICES*

Design	Device	LUT	FF	Slice	Fmax (MHz)	Latency[1]	Delay (ns)[2]	ADP[3]	Throughput[4]
$n = 256$									
Pro. ($u = 1$)	Virtex-7	12,360	7,163	3,501	395	128	324	1,135	0.790
Pro. ($u = 2$)	Virtex-7	20,269	7,204	5,723	340	64	188	1,077	1.362
Pro. ($u = 4$)	Virtex-7	31,157	37,845	13,675	376	32	85	1,164	3.012
Pro. ($u = 8$)	Virtex-7	58,532	86,801	26,751	364	16	44	1,176	5.818
Pro. ($u = 16$)	Virtex-7	91,056	135,690	40,221	297	8	27	1,083	9.481
Pro. ($u = 1$)	Kintex-7	12,360	7,163	3,606	384	128	333	1,202	0.679
Pro. ($u = 2$)	Kintex-7	20,381	2,208	5,660	347	64	184	1,044	1.391
Pro. ($u = 4$)	Kintex-7	31,349	37,839	13,258	398	32	80	1,066	3.200
Pro. ($u = 8$)	Kintex-7	57,665	86,619	19,565	293	16	55	1,067	4.655
Pro. ($u = 16$)	Kintex-7	91,026	135,736	40,393	231	8	35	1,414	7.314
$n = 512$									
Pro. ($u = 1$)	Virtex-7	24,739	14,407	6,871	392	256	1,306	8,974	0.784
Pro. ($u = 2$)	Virtex-7	40,668	14,447	11,098	333	128	385	4,267	1.332
Pro. ($u = 4$)	Virtex-7	60,148	75,733	24,997	360	64	178	4,449	2.876
Pro. ($u = 8$)	Virtex-7	116,022	173,781	51,552	289	32	111	5,696	4.634
Pro. ($u = 16$)	Virtex-7	182,279	272,044	78,910	219	16	73	5,760	7.014
Pro. ($u = 1$)	Kintex-7	24,736	14,406	6,949	380	256	674	4,684	0.760
Pro. ($u = 2$)	Kintex-7	40,497	14,346	10,932	309	128	414	4,628	1.237
Pro. ($u = 4$)	Kintex-7	58,680	75,669	25,973	309	64	207	5,380	2.485
Pro. ($u = 8$)	Kintex-7	115,936	173,781	49,003	305	32	105	5,141	4.876
Pro. ($u = 16$)	Kintex-7	220,310	271,894	74,728	218	16	73	5,484	7.010

*: All the related calculations for all designs are based on the decryption phase of the RBLWE-ENC (Dec.: decryption).
[1]: Latency cycles, refers to the major computation time, where the input loading and final output delivery are not included (following the existing styles).
[2]: Delay=critical-path×latency=latency/Fmax. [3]: ADP=#slice×delay (Dec.) $\times 10^3$. [4]: Throughput = n/Delay.

consideration when selecting the proper competing designs. For instance, we notice that the designs of [31] do not include the input processing component (reported area is smaller than the actual value, similar to Arch.-II of [33]), and we also consider that the designs of [33] have shown their efficiency over the ones in [31], we thus just list the result of [31] for discussion. Note Arch.-I of [33] is listed for actual comparison due to its complete setup in this aspect (see Fig. 4 of [33]).

4) Comparison Discussion: When comparing with the existing designs, although the area consumption of KINA is relatively high, the latency of the proposed design with different choices of u is much lower than the existing designs, e.g., it ($u = 2$) involves at least 53.92% less delay than the best of the existing designs on the Virtex-7 device for $n = 256$. Also, the ADP of the proposed design is at least 15.37% less than the existing design of [34] for $n = 512$ on the Virtex-7 device. Another noticeable advantage of the proposed design is the throughput, which refers to the performance of calculation over a unit period time. From Table III, we can see that the throughput of the proposed design ($u = 2$) is 2.17× to 5.24× than the existing designs on the Virtex-7 device. This similar situation happens on the Intel Stratix-V device, as shown in Table IV, where the proposed accelerator ($u = 2$) has significantly better ADP than [29] and [32] (we followed the existing designs to calculate the ADP). This indicates that the proposed design is extremely suitable for high-speed calculation, such as IoT servers.

5) Comparison With Software Implementation (CPU): To demonstrate the efficiency of the proposed accelerator, we have also measured the performance of the software implemented (coded in C language) RBLWE-ENC deploying the proposed KA strategy. The hardware setup is as follows: 1) we have used the microbenchmark support library from Google [47] as the benchmark library; 2) we have used the single core of AMD Ryzen Threadripper 3960× processor running at 3.8 GHz; 3) the testing was carried out on the Ubuntu 20.04 LTS OS on a virtual machine; and 4) we have used g++ 9.4.0 to compile the code and disabled the optimization flag. The software implementation of KA-deployed RBLWE-ENC (decryption) takes 145 762 ns (number of testing iterations is 4798) and 545 954 ns (number of testing iterations is 1282) for $n = 256$ and $n = 512$, respectively. From the obtained data, one can see that the proposed hardware KINA accelerator is much faster than the software implementation one and hence is preferred for practical deployment. Finally, we want to mention that the FPGA-based implementation can also be extended further as specific integrated circuits for potential applications, which the CPU implementation does not offer.

6) Discussion About the Performance With Other PQC: We have also listed other lattice-based schemes in Table V for a more comprehensive discussion. In particular, we have selected the available implementations of NIST PQC schemes for discussion: NewHope [44], Kyber [45], and Saber [46]. Note the existing designs mostly did not report the slice number, and we thus use the number of LUTs to calculate the ADP.

It is seen that the proposed KINA has significantly better area-time complexities than the existing designs. Besides that, we want to point out that the designs of [44] and [45] have used extra numbers of DSPs and BRAMs, and hence their actual area-time complexities are larger than the calculated ADP. Meanwhile, when comparing with the public-key scheme Saber of [46] (KA-based implementation as well), the proposed KINA not only has smaller area usage but also

TABLE III
COMPARISON OF FPGA IMPLEMENTATION PERFORMANCE (AMD-XILINX DEVICES)*

Design	Device	LUT	FF	Slice	Fmax (MHz)	Latency[1]	Delay (ns)[2]	ADP[3]	Throughput[4]
$n = 256$									
[25]	Spartan-6	6,728	6,813	1,874	101	262	2,594	4,861	0.099
[26]#	Virtex-7	5,153	2,151	1,701	261	257	985	1,675	0.260
[31]* $u = 1$	Virtex-7	3,600	2,568	1,146	415	256	617	707*	0.415
[31]* $u = 2$	Virtex-7	6,237	2,568	1,881	314	128	408	767*	0.627
[33]* Arch.-I	Virtex-7	5,324	6,469	1,781	357	256	717	1,277	0.360
[34]	Virtex-7	7.6k	6.2k	2.3k	514	259	504	1,159	0.508
Pro. ($u = 2$)	Virtex-7	20,269	7,204	5,723	340	64	188	1,077	1.362
[31]* $u = 1$	Kintex-7	3,600	2,568	1,134	394	256	650	737*	0.394
[31]* $u = 2$	Kintex-7	6,244	2,568	1,932	318	128	403	779*	0.635
[33]* Arch.-I	Kintex-7	5,159	6,467	1,963	347	256	738	1,449	0.340
Pro. ($u = 2$)	Kintex-7	20,381	2,208	5,660	347	64	184	1,044	1.391
$n = 512$									
[26]#	Virtex-7	10,285	4,249	3,289	263	513	1,951	6,417	0.262
[31]* $u = 1$	Virtex-7	7,184	5,128	2,208	399	512	1,283	2,833*	0.399
[31]* $u = 2$	Virtex-7	13,100	5,133	3,796	286	256	895	3,397*	0.572
[33]* Arch.-I	Virtex-7	11,123	12,851	3,668	357	512	1,434	5,260	0.357
[34]	Virtex-7	15k	12.3k	4.6k	469.7	515	1,096	5,042	0.470
Pro. ($u = 2$)	Virtex-7	40,668	14,447	11,098	333	128	385	4,267	1.332
Pro. ($u = 2$)	Kintex-7	40,497	14,346	10,932	309	128	414	4,628	1.237

*: The designs of [31] do not include the input processing component, and hence the reported area is smaller than its actual value (this is the same situation applies to the Arch.-II of [33]). Thus, we list the reported data of [31] here just for a brief discussion. Nevertheless, the Arch.-I of [33] has the complete setup on the input processing component (see Fig. 4 of [33]) and also consider that the designs of [33] have shown their efficiency over the ones in [31], we just list the Arch.-I of [33] here for a more complete comparison.
#: The authors of [31] have re-implemented the design in [26] (including all input processing shift-registers), we thus use the data from [31].
*: All the related calculations for all designs are based on the decryption phase of the RBLWE-ENC (Dec.: decryption).
[1]: Latency cycles, refers to the major computation time, where the input loading and final output delivery are not included (following the existing styles).
[2]: Delay=critical-path×latency=latency/Fmax. [3]: ADP=#Slice×delay (Dec.) $\times 10^3$. [4]: Throughput = n/Delay.

TABLE IV
COMPARISON OF FPGA IMPLEMENTATION PERFORMANCE (INTEL DEVICES)*

Design	Device	ALMs	Fmax (MHz)	Latency[1]	Delay (ns)[2]	ADP[3]	Throughput[4]
$n = 256$							
[29]	Stratix-V	4,495	321	258	133	3,612	0.32
[32] high speed	Stratix-V	4,446	379	257	257	3,013	0.38
Pro. ($u = 2$)	Stratix-V	6,186	282	64	133	820	1.93
$n = 512$							
[29]	Stratix-V	9,038	317	514	1,621	14,652	0.32
[32] high speed	Stratix-V	8,864	328	513	1,564	13,864	0.33
Pro. ($u = 2$)	Stratix-V	12,330	442	128	290	820	1.77

*: All the related calculations for all designs are based on the decryption phase of the RBLWE-ENC (Dec.: decryption). ALM: Adaptive logic module.
[1]: Latency cycles, refers to the major computation time, following the existing styles.
[2]: Delay=critical-path×latency=latency/Fmax. [3]: ADP=#ALMs×delay (Dec.) $\times 10^3$. [4]: Throughput = n/Delay.

involves much faster processing time. One may conclude that the proposed KINA is more suitable for high-performance lightweight applications where the processing speed is more desirable with relatively small resource usage.

Lastly, we also want to mention that both Kyber and Saber explore module rings in constructing the security keys and ciphertext (e.g., Kyber is built on the Module-LWE [49]). Meanwhile, Kyber has utilized a built-in NTT ciphertext structure to improve practical implementation performance [49], while Saber explores the small secret-key size and rounding techniques [50]. Still, it is encouraging to see that comparing the implementation result of KINA with Kyber (and even Saber) also highlights the potential efficiency of Ring-LWE, especially when the parameter (e.g., modulus q) is selected as small. Due to the module ring setup, Kyber may obtain better efficiency on area usage; Ring-LWE-based scheme, however, can achieve faster computation (which leads to better overall area-delay efficiency). For lightweight applications, where the security requirement is not that high, one may choose to use the Ring-LWE-based setting with small parameters (i.e., RBLWE in this article) to obtain better feasibility and efficiency.

7) Unique Features: Overall, the proposed KINA possesses two unique features: 1) this is the first RBLWE-based PQC accelerator based on KA and 2) the proposed accelerator provides flexible processing throughput, depending on the choices of u, for potential high-speed environments. These two unique features facilitate the deploying of the proposed KINA in various high-performance applications.

VI. RELATED WORKS AND FUTURE RESEARCH

While RBLWE is a relatively new ring variant of BLWE, which uses binary errors to achieve low-complexity implementation [19], [20], [22], [23], [24], there also exist works

TABLE V
DISCUSSION ABOUT THE PERFORMANCE WITH OTHER PQC IMPLEMENTATIONS

design	n	phase	device	LUT	FF	Slice	Fmax	DSP	BRAM	latency[1]	delay	ADP
NewHope [44]	512	Enc./Dec.	Artix-7	6,780	4,026	-	200	2	7	6.6k/2.5k	33.00/12.50	223,740/84,750
Kyber [45][2] server	512[1]	Enc./Dec.	Artix-7	7,412	4,644	2,126	161	3	2	5,079/-	30.50/-	226,066/-
Kyber [45][2] client	512[1]	Enc./Dec.	Artix-7	6,785	3,981	-	167	3	2	-/6,668	-/41.30	-/280,220
Pro. $u = 1$	512	Enc./Dec.	Artix-7	45,158	28,671	12,044	250	0	0	512/256	2.05/1.02	92,484/46,242
Saber [46][3]	512[3]	Enc./Dec.	UltraScale+	34,886	9,858	-	100	85	6	664/326	6.64/3.26	231,643/113,728
Pro. $u = 1$	512	Enc./Dec.	UltraScale+	50,118	28,750	7,871	505	0	0	512/256	1.01/0.51	50,842/25,421
Pro. $u = 2$	512	Enc./Dec.	UltraScale+	90,763	28,774	15,003	514	0	0	256/128	0.62/0.31	55,967/27,984

Unit for Fmax: MHz. Unit for delay: ns. Delay=critical-path×latency. ADP=#LUTs × delay. Since most of the existing designs do not report the slice number, we just use the number of LUTs to calculate the ADP.
[1]: Latency refers to encryption/decryption (encapsulation/decapsulation) cycles, respectively.
[2]: The design of [45] used server and client to execute encapsulation and decapsulation, respectively. We just listed the performance of security rank $k = 2$, which is an equivalent size of $n = 512$.
[3]: Public-key encryption scheme of Saber under the security rank of $k = 2$, which is an equivalent size of $n = 512$.

about the regular Ring-LWE-based PQC [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [48]. As Kyber and NewHope can be seen as important representatives of them, we do not compare with these regular Ring-LWE-based PQC.

The proposed KINA has constant time operation and hence is resistant to regular timing attacks [53]. While this article focuses mostly on the developing KA-initiated computation strategy for the major arithmetic operation of RBLWE-ENC (polynomial multiplication) as well as the overall hardware accelerator, the research on designing and implementing a key-encapsulation mechanism (KEM) of RBLWE-ENC and related works (such as side-channel attacks) is out of the scope of this article. Nevertheless, we want to mention that developing an efficient KEM version of RBLWE-ENC can be seen as one of our future research directions. Meanwhile, side-channel analysis and related countermeasures can also be extended further on the proposed accelerator.

Finally, we also want to emphasize that the research and development for lightweight PQC is still an under-explored area (as pointed out in the NSF SaTC PI Meeting'22 [18]), though the NIST PQC standardization has recently selected algorithms like Kyber for general-purpose usage [6]. Therefore, we hope the proposed work in this article can stimulate many follow-up investigations from the research community, e.g., scheme development, parameter selection, security analysis, implementation techniques, etc. Besides that, we also hope the proposed KINA design strategy can be extended for polynomial multiplication used in NIST-selected schemes like Falcon [51] and even Dilithium [52], where they are not bound with fast algorithms [6] (Kyber is built-in with NTT already [49]). As KINA provides a flexible and extensible way for accelerating large integer polynomial multiplications, it is natural to think about applying similar structures to these NIST PQC schemes. While emphasizing the plausible high-throughput and flexible processing, a couple of related works need to be done for such endeavoring, including modular reduction, point-wise multiplier, sampler design, etc.

VII. CONCLUSION

In this article, we propose an efficient RBLWE-ENC accelerator, KINA, on the hardware platform. The key contributions of this work include: 1) usage of KA to derive an efficient computation of the polynomial multiplication over ring, the major arithmetic operation of the RBLWE-ENC; 2) efficiently mapped the proposed algorithm into a new RBLWE-ENC accelerator, KINA (including the higher speed version); and 3) conducted analysis and comparison to show the efficiency of the proposed accelerator. It turns out to be: a) the proposed KINA is the first Karatsuba-based RBLWE-ENC accelerator that achieves a complexity of $O((3n^2)/4)$ and b) the proposed accelerator provides flexible processing capabilities. The proposed design strategy and implementation results are expected to help the further development of the RBLWE-based lightweight PQC scheme.

ACKNOWLEDGMENT

The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense, or of the United States Air Force. The appearance of external hyperlinks does not constitute endorsement by the United States Department of Defense (DoD) of the linked websites, or of the information, products, or services contained therein. The DoD does not exercise any editorial, security, or other control over the information found at these locations.

REFERENCES

[1] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 1–14.
[2] W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
[3] D. J. Bernstein and L. Tanja, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.
[4] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 147–191.
[5] *Post-Quantum Cryptography*. Accessed: 2016. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography
[6] *PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*. Accessed: 2022. [Online]. Available: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4
[7] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009.
[8] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2010, pp. 1–23.

[9] 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.

[10] D. Liu, C. Zhang, H. Lin, Y. Chen, and M. Zhang, "A resource-efficient and side-channel secure hardware implementation of ring-LWE cryptographic processor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1474–1483, Apr. 2019.

[11] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[12] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2013, pp. 68–85.

[13] A. Aysu, C. Patterson, and P. Schaumont, "Low-cost and area-efficient FPGA implementations of lattice-based cryptography," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, Jun. 2013, pp. 81–86.

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

[15] J. Howe, T. Oder, M. Krausz, and T. Güneysu, "Standard lattice-based key encapsulation on embedded devices," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 372–393, Aug. 2018.

[16] S. Bian, M. Hiromoto, and T. Sato, "Filianore: Better multiplier architectures for LWE-based post-quantum key exchange," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[17] T. Fritzmann and J. Sepúlveda, "Efficient and flexible low-power NTT for lattice-based cryptography," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2019, pp. 141–150.

[18] (2022). *National Science Foundation (NSF) 2022 Secure and Trustworthy Cyberspace Principal Investigators' Meeting (SaTC PI Meeting'22) Break Out Group Reports/Slides: Security in a Post-Quantum World, Slides Page 4*. [Online]. Available: https://cps-vo.org/group/satc-pimtg22/breakouts

[19] D. Micciancio and C. Peikert, "Hardness of SIS and LWE with small parameters," in *Proc. Annu. Cryptol. Conf.*, 2013, pp. 21–39.

[20] J. Buchmann, F. Gopfert, R. Player, and T. Wunderer, "On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack," in *Proc. Int. Conf. Cryptol. Afr.*, 2016, pp. 24–43.

[21] M. Liu and P. Nguyen, "Solving BDD by enumeration: An update," in *Topics in Cryptology—CT-RSA 2013*, vol. 7779, E. Dawson, Ed. Berlin, Germany: Springer, 2013, pp. 293–309.

[22] F. Gopfert, C. Vredendaal, and T. Wunderer, "A hybrid lattice basis reduction and quantum search attack on LWE," in *Proc. Int. Workshop Post-Quantum Cryptogr.* Cham, Switzerland: Springer, 2017, pp. 184–202.

[23] D. Micciancio, "On the hardness of learning with errors with binary secrets," *Theory Comput.*, vol. 14, no. 1, pp. 1–17, 2018.

[24] J. Buchmann, F. Göpfert, T. Göneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proc. 2nd ACM Int. Workshop IoT Privacy, Trust, Secur.*, May 2016, pp. 1–8.

[25] A. Aysu, M. Orshansky, and M. Tiwari, "Binary ring-LWE hardware with power side-channel countermeasures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1253–1258.

[26] S. Ebrahimi, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5500–5507, Jun. 2019.

[27] J. Xie, K. Basu, K. Gaj, and U. Guin, "Special session: The recent advance in hardware implementation of post-quantum cryptography," in *Proc. IEEE 38th VLSI Test Symp. (VTS)*, Apr. 2020, pp. 1–10.

[28] P. He, U. Guin, and J. Xie, "Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary ring-LWE post-quantum cryptography," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 383–394, Jun. 2021.

[29] J. Xie, P. He, and W. Wen, "Efficient implementation of finite field arithmetic for binary ring-LWE post-quantum cryptography through a novel lookup-table-like method," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 1279–1284.

[30] K. Shahbazi and S.-B. Ko, "Area and power efficient post-quantum cryptosystem for IoT resource-constrained devices," *Microprocessors Microsyst.*, vol. 84, Jul. 2021, Art. no. 104280.

[31] J. Xie, P. He, X. Wang, and J. L. Imaña, "Efficient hardware implementation of finite field arithmetic AB+CAB+C for binary ring-LWE based post-quantum cryptography," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1222–1228, Apr. 2022.

[32] B. J. Lucas et al., "Lightweight hardware implementation of binary ring-LWE PQC accelerator," *IEEE Comput. Archit. Lett.*, vol. 21, no. 1, pp. 17–20, Jan. 2022.

[33] J. L. Imaña, P. He, T. Bao, Y. Tu, and J. Xie, "Efficient hardware arithmetic for inverted binary ring-LWE based post-quantum cryptography," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 8, pp. 3297–3307, Aug. 2022.

[34] D. Xu, X. Wang, Y. Hao, Z. Zhang, Q. Hao, and Z. Zhou, "A more accurate and robust binary ring-LWE decryption scheme and its hardware implementation for IoT devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 8, pp. 1007–1019, Aug. 2022.

[35] P. He, T. Bao, J. Xie, and M. Amin, "FPGA implementation of compact hardware accelerators for ring-binary-LWE based post-quantum cryptography," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, no. 3, pp. 1–23, 2023.

[36] A. Sarker, M. M. Kermani, and R. Azarderakhsh, "Fault detection architectures for inverted binary ring-LWE construction benchmarked on FPGA," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 4, pp. 1403–1407, Apr. 2021.

[37] J. M. Pollard, "The fast Fourier transform in a finite field," *Math. Comput.*, vol. 25, no. 114, pp. 365–374, 1971.

[38] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Sov. Phys. Doklady*, vol. 7, no. 7, pp. 595–596, Jan. 1963.

[39] J. Xie, P. Meher, X. Zhou, and C. Lee, "Low register-complexity systolic digit-serial multiplier over $GF(2^m)$," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 773–783, Oct. 2018.

[40] J.-S. Pan, C.-Y. Lee, and P. K. Meher, "Low-latency digit-serial and digit-parallel systolic multipliers for large binary extension fields," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 12, pp. 3195–3204, Dec. 2013.

[41] P. L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 362–369, Mar. 2005.

[42] C.-Y. Lee and J. Xie, "Digit-serial versatile multiplier based on a novel block recombination of the modified overlap-free Karatsuba algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 1, pp. 203–214, Jan. 2019.

[43] D.-E.-S. Kundi, A. Khalid, S. Bian, C. Wang, M. O'Neill, and W. Liu, "AxRLWE: A multilevel approximate ring-LWE co-processor for lightweight IoT applications," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10492–10501, Jul. 2022.

[44] N. Zhang, B. Yang, C. Chen, S. Yin, S. Wei, and L. Liu, "Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 49–72, Mar. 2020.

[45] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 328–356, Feb. 2021.

[46] Y. Zhu et al., "LWRpro: An energy-efficient configurable crypto-processor for module-LWR," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1146–1159, Mar. 2021.

[47] *Benchmark: A Microbenchmark Support Library Google*. Accessed: 2022. [Online]. Available: https://google.github.io/benchmark/random_interleaving.html

[48] Y. A. Birgani, S. Timarchi, and A. Khalid, "Area-time-efficient scalable schoolbook polynomial multiplier for lattice-based cryptography," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 12, pp. 5079–5083, Dec. 2022.

[49] (2020). *Kyber*. [Online]. Available: https://pq-crystals.org/kyber/

[50] (2019). *Saber*. [Online]. Available: https://www.esat.kuleuven.be/cosic/pqcrypto/saber/

[51] (2021). *Falcon*. [Online]. Available: https://falcon-sign.info/

[52] (2022). *Dilithium*. [Online]. Available: https://pq-crystals.org/dilithium/resources.shtml

[53] T. Schneider, A. Moradi, and T. Guneysy, "ParTI–towards combined hardware countermeasures against side-channel and fault-injection attacks," in *Proc. Annu. Cryptol. Conf.*, 2016, pp. 302–332.

Pengzhou He (Graduate Student Member, IEEE) received the B.S. degree in intelligence science and technology from the University of Science and Technology, Beijing, China, in 2019. He is currently working toward the Ph.D. degree in computer engineering at Villanova University, Villanova, PA, USA.

His research interests include postquantum cryptography, hardware security, high-performance IoT devices, and application of machine learning in security systems.

Yazheng Tu (Graduate Student Member, IEEE) received the M.S. degree in computer science from Washington University, St. Louis, MO, USA, in 2020. He is currently working toward the Ph.D. degree in computer engineering at Villanova University, Villanova, PA, USA.

His research interests include lightweight postquantum cryptography, hardware security, and artificial intelligence (AI) in security systems.

Jiafeng (Harvest) Xie (Senior Member, IEEE) received the M.E. degree from Central South University, Changsha, China, in 2010, and the Ph.D. degree from the University of Pittsburgh, Pittsburgh, PA, USA, in 2014.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Villanova University, Villanova, PA, USA. His research interests include postquantum cryptographic engineering, hardware security, lightweight postquantum cryptography, and VLSI implementation of encrypting neural network systems.

Dr. Xie has served as a technical committee member for many reputed conferences such as the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), International Conference on Computer-Aided Design (ICCAD), and Design Automation Conference (DAC). He served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS previously. He received the IEEE ACCESS Outstanding Associate Editor 2019. He also received the IEEE Philadelphia Section Merrill Buckley Jr. Student Project Award 2022, AFRL Visiting Research Faculty Program (VFRP) Award 2022, and the Best Paper Award from IEEE HOST 2019. Currently, he is serving as an Associate Editor for *Microelectronics Journal*, IEEE ACCESS, and IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS.

H. S. Jacinto (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Boise State University, Boise, ID, USA, in 2020.

He worked on hardware acceleration and SoC design of secure protocols with Boise State University. From 2015 to 2017, he worked with Idaho National Laboratory and the Advanced Energy Laboratory, researching secure communications and sensor design for nuclear reactor monitoring. From 2018 to the present, he has been with the Air Force Research Laboratory, Quantum Information Science Group researching quantum information processing systems, integrated quantum photonics, and quantum control. His research focuses on quantum network cybersecurity, quantum informatics, and secure adaptive hardware antitamper and encryption technologies.