TINA: TMVP Initiated Novel Accelerator for Lightweight Ring-LWE-based PQC

Tianyou Bao, *Student Member, IEEE*, Pengzhou He, *Student Member, IEEE*,
Shi Bai, and Jiafeng Xie, *Senior Member, IEEE*

***Abstract*—Post-quantum cryptography (PQC) has recently garnered significant attention across various communities. Alongside the ongoing standardization process for general-purpose PQC algorithms by the National Institute of Standards and Technology (NIST), the research community is actively exploring the realm of lightweight PQC schemes. Ring-Binary-Learning-with-Errors (RBLWE)-based encryption scheme (RBLWE-ENC) is a promising lightweight PQC candidate suitable for Internet-of-Things (IoT) and edge computing applications. The parameters of the RBLWE-ENC, however, do not favor deploying typical fast algorithms like number theoretic transform (NTT). In this paper, therefore, we propose to design a Toeplitz Matrix-Vector Product (TMVP) Initiated Novel Accelerator (TINA) for RBLWE-ENC. We innovatively used TMVP (a subquadratic-complexity fast algorithm for polynomial multiplication) to derive the significant arithmetic operation of RBLWE-ENC into a new form for high-performance operation. This novel formulation culminates in the development of a comprehensive accelerator known as TINA. Through implementation and comparative analysis, we demonstrate the efficiency gains achieved by our proposed accelerator. To the authors' best knowledge, this is the first report on the TMVP strategy initiated RBLWE-ENC accelerator. The findings of this work is expected to provide valuable references in the ongoing advancement of lightweight PQC development.**

***Index Terms*—Fast algorithm, hardware accelerator, high-performance, lightweight post-quantum cryptography, Ring-Binary-Learning-with-Errors, Toeplitz Matrix-Vector Product**

I. Introduction

Nearly all currently deployed public-key cryptographic algorithms, such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC), are vulnerable to a sufficiently large quantum computers, due to Shor's algorithm [1]. Therefore, the pursuit of an alternative approach known as the post-quantum cryptography (PQC) has gained significant attention across diverse communities. The National Institute of Standards and Technology (NIST) is currently engaged in a standardization effort focused on general-purpose PQC algorithms [2]. At the same time, the research community is actively seeking lightweight PQC solutions tailored for specific applications, including Internet-of-Things (IoT) deployments and edge computing servers [3], [4], as underscored in the recent National Science Foundation (NSF) Secure and Trustworthy Cyberspace Principal Investigators' Meeting 2022 (SaTC PI Meeting'22) [5].

Manuscript received August 15, 2023. (corresponding author: Jiafeng Xie).

T. Bao, P. He, and J. Xie are with the Department of Electrical and Computer Engineering, Villanova University, Villanova, PA, 19085 USA (e-mail: {tbao,phe,jiafeng.xie}@villanova.edu).

S. Bai is with the Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL, 33431 USA (e-mail: sbai@fau.edu).

Among the candidates for PQC, lattice-based cryptography (LBC) stands out, with many LBC schemes built upon the foundation of the Learning-with-Error (LWE) problem [6]. A commonly used variant of LWE is Binary-LWE (BLWE), which admits worst-case to average-case hardness to standard lattice problems [7], and enjoys a more efficient implementation due to the use of binary errors [8], [9]. Its structured variant, RBLWE-ENC is based on the average-case hardness of the RBLWE (Ring-Binary-Learning-with-Errors) problem [9]. Extensive security analyses and evaluations have confirmed its promise for lightweight applications [10].

Following the recent trend in PQC, more attention has been made to efficient hardware implementation of RBLWE-ENC. On the one hand, a cohort of resource-constrained use cases, exemplified by IoT and edge devices, necessitates a compact RBLWE-ENC implementation. Conversely, applications such as IoT servers boast more abundant resources, encompassing platforms like field-programmable gate arrays (FPGAs), enabling the deployment of robust high-performance accelerators. In the former scenario, a possible approach involves adopting a point-wise processing strategy with a small area but at the cost of extended latency [11], [12]. On the other hand, the latter scenario demands an accelerator of elevated performance while still upholding yet with decent resource usage. Our paper focuses on the efficient design of the RBLWE-ENC accelerator for the latter scenario, specifically targeting a complete accelerator (with a sampler), which has yet to be done in the literature.

Existing Work. The main arithmetic operation of RBLWE-ENC is the polynomial multiplication over ring [8]. Accordingly, starting from the introduction of RBLWE-ENC, several efforts have been used to obtain an efficient implementation of the polynomial multiplication for RBLWE-ENC, especially on the hardware platforms. Some of the case example efforts include: The first full-hardware work was reported in [13]. The subsequent work [4] and [14] introduced optimized structures. A further improvement was reported in [15]. More recently, high-performance implementations were presented in [16], [17]. Other work in the field also include [11]. Strictly speaking, these existing work are all based on the schoolbook-based method with a computational-complexity of $\mathcal{O}(n^2)$. Very recently, a Karatsuba algorithm initiated accelerator for RBLWE-ENC was presented in [18], which was the first try in the field to use a fast algorithm to reduce the computational complexity to $O(\frac{3n^2}{4})$. Its corresponding architecture, however, is still relatively large, and thus there still exists room for further exploration and breakthrough.

TABLE I: Notations for This Paper

Notations for RBLWE-ENC	
a	public parameter (integer polynomial)
r_1, r_2	binary polynomials (r_2: secret key)
e_1, e_2, e_3	binary errors (binary polynomials)
m	message
n	scheme size
q	modulus
$f(x)$	ring polynomial ($f(x) = x^n + 1$)
Notations for deriving the proposed algorithm & accelerator	
Z_0, Z_1	TMVP input matrix-vectors (binary field)
T_0, T_1, T_2	TMVP main matrices (binary field)
V_0, V_1	TMVP output matrix-vectors (binary field)
G, P	binary polynomial
W, D, Q	integer polynomial

Challenges. Though the design in [18] is still not that ideal, using a fast algorithm for RBLWE-ENC seems to be an inevitable trend. On the other hand, however, there exist three aspects of challenges related to the accelerator design for RBLWE-ENC. First of all, the parameter settings of RBLWE-ENC are generally smaller than the regular Ring-LWE-based schemes. Hence, it does not favor deploying typical fast algorithms such as number theoretic transform (NTT) [19]. Secondly, the direct deploying of other fast algorithms like the Karatsuba algorithm needs significant implementation efforts on the output processing due to the involved arithmetic factors such as x or $(x^2 - x)$ and hence is hard to be popularized for general usage. Meanwhile, because of the unequaled coefficient size in the polynomial multiplication of RBLWE-ENC, the addition-related pre-computing operations will increase the small-size coefficient-related processing bit-width, which might offset the gain from deploying fast algorithm [18]. Lastly, all the existing designs are basically based on the main arithmetic operation of RBLWE-ENC and have yet to consider the overall operations, including the sampler (not a trivial effort).

Motivation and Main Contributions. It is noticed that Toeplitz Matrix-Vector Product (TMVP) is a relatively new technique for polynomial multiplication over the binary field [20], [21], [22], [23]. Recently, TMVP has also been deployed for polynomial multiplication used in other types of PQC schemes like [24]. Compared with the Karatsuba algorithm, we noticed that TMVP-based polynomial multiplication involves simpler final output processing as it is based on matrix-vector product-related operations. This work is an extension of [24] that we propose to design a **T**MVP **I**nitiated **N**ovel **A**ccelerator (TINA) for RBLWE-ENC. Major contributions include:

- We have innovatively used TMVP to derive a new algorithm based on the major arithmetic operation of RBLWE-ENC for high-performance computation.
- We have mapped the proposed algorithm into an efficient hardware accelerator (TINA) with the help of several new algorithm-to-architecture design techniques.
- We have conducted thorough implementation and comparison to show that the proposed accelerator has superior performance over the state-of-the-art designs and is suitable for high-performance, lightweight applications.

Note that TMVP-deployed BRLWE-ENC implementation has yet to be explored in the literature. Besides, the proposed accelerator has included the sampler and related phases' operations and thus is more complete than the existing designs.

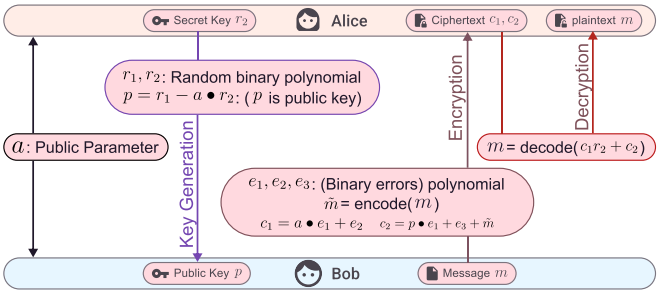

Fig. 1: Three phases for RBLWE-ENC.

The rest of the paper is organized as follows. The preliminary knowledge is introduced in Section II. The proposed algorithm is described in Section III. In Section IV, we present the proposed TINA. The proposed accelerator is evaluated in Section V. Section VI gives the conclusion.

II. Preliminary Knowledge

Notations. There are two groups of notations used in this paper: one for the operations of RBLWE-ENC and another for the deriving the proposed algorithm based on TMVP method. Details of these notations are listed in Table I.

A. RBLWE-ENC

Overview. RBLWE is a structured variant of BLWE [7], [9], which uses the ring $\mathbb{Z}_q[x]/(x^n + 1)$ (one element in the ring is expressed as the polynomial of degree $(n-1)$ with integer coefficients modulo q, and the other has merely binary coefficients). RBLWE-ENC consists of three phases: *key generation*, *encryption*, and *decryption* [9] (see Fig. 1).

- **Key generation**. Note that a is a g public parameter shared by Alice and Bob; r_1 and r_2 are randomly selected binary polynomials (r_2 is the secret key). After the operation of $p = r_1 - a \cdot r_2$, p is sent to Bob (r_1 is discarded then).
- **Encryption**. After receiving the public key from Alice, Bob uses three error (binary) polynomials e_1, e_2, and e_3 to produce the two polynomials (ciphertext) of c_1 and c_2. Message $\tilde{m}$ is computed by the operation that each coefficient of the input message m (n-bit binary polynomial) is multiplied with $q/2$ (respectively). c_1 and c_2 will be sent back to Alice.
- **Decryption**. Alice finally uses the secret key r_2 to recover the original message m. Note that there is a threshold decoder function involved with the final decryption process, i.e., it will produce a binary value of '1' if the coefficient is in the range of $(q/4, 3q/4)$, otherwise it will return '0'.

Note that though the authors of [4] have proposed to use an inverted RBLWE-ENC, where the integer coefficients of one polynomial are selected from the inverted range of $(-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor - 1)$ (facilitate two's complement representation), the three phases of Fig. 1 remain almost the same as the previous. Therefore, in this paper, we still use the major arithmetic operations in Fig. 1 while adopting this strategy.

Security Level. The BLWE problem admits a worst-case hardness of the standard lattice problem [7], [25],while the

RBLWE-ENC is based on the average hardness of the RBLWE problem [9]. Using a structured variant such as Ring-LWE and Module-LWE is a common approach in the NIST PQC standardization process [2]. Concretely, the security level of RBLWE-ENC has been estimated in [9] and then re-examined in [10]. Recent research [25] has also confirmed BLWE's security hardness, which again indirectly supported RBLWE-ENC being a promising scheme for lightweight applications.

Parameter Settings. There exist two widely used parameter sets for the RBLWE-ENC, namely $(n, q) = (256, 256)$ and $(n, q) = (512, 256)$. In this work, we focus on implementation for such two parameter sets as they have been used extensively in the research community to conduct implementation benchmark for lightweight applications [4], [18], [12]. In particular, the parameter set of $(n, q) = (512, 256)$ has been examined in [10] which has a quantum/classic security of 140/190 bits.

B. TMVP

Definition. Let us define an $n \times n$ Teoplitz matrix as $T = [t_{i,j}]_{0 \le i,j \le n-1}$, where $t_{i,j} = t_{i-1,j-1}$ [20], [22]. Define $V = (V_0, V_1)$ as an $n \times 1$ column vector (V_0 and V_1 are $\frac{n}{2} \times 1$ column vectors) and T_0, T_1, and T_2 as $\frac{n}{2} \times \frac{n}{2}$ Toeplitz matrices. Finally, we have the following expressions [20]

$$\begin{aligned} Z = \begin{bmatrix} Z_0 \\ Z_1 \end{bmatrix} &= \begin{bmatrix} T_0 & T_2 \\ T_1 & T_0 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \\ &= \begin{bmatrix} T_0(V_0 + V_1) + (T_2 + T_0)V_1 \\ T_0(V_0 + V_1) + (T_1 + T_0)V_0 \end{bmatrix}, \end{aligned} \tag{1}$$

where we can further define four components, i.e., component matrix point (CMP), component vector point (CVP), pointwise multiply (PWM), and reconstruction (R): $\mathrm{CMP}(T) = (T_2 + T_0, T_0, T_0 + T_1)$, $\mathrm{CVP}(V) = (V_1, V_0 + V_1, V_0)$, $P = \mathrm{PWM}(\mathrm{CMP}(T), \mathrm{CVP}(V)) = (P_0, P_1, P_2)$, and $Z = \mathrm{R}(P) = (P_0 + P_1, P_1 + P_2)$ (for $P_0 = (T_2 + T_0)V_1$, $P_1 = T_0(V_0 + V_1)$, and $P_2 = (T_1 + T_0)V_0$). Therefore, the complexity of original matrix-vector product ($\mathcal{O}(n^2)$) is reduced to $\mathcal{O}(3(\frac{n}{2})^2)$ (three sub-matrix-vector products).

The operation of (1) can be iteratively applied to the original matrix-vector product to achieve subquadratic-complexity. So far, according to the authors' best knowledge, there is no TMVP-based design reported for RBLWE-ENC.

III. TINA: Algorithmic Formulation

As seen from Fig. 1, polynomial multiplication is the major arithmetic operation among all three phases of RBLWE-ENC. This section thus focuses on the formulation of the proposed algorithm for the polynomial multiplication of RBLWE-ENC. Note that this algorithm is extended from [24].

Extension to the Integer Field. Define $t'_{i,j}$ as an integer as well as v'_i. We then have

$$\begin{bmatrix} Z'_0 \\ Z'_1 \end{bmatrix} = \begin{bmatrix} T'_0(V'_0 + V'_1) + (T'_2 - T'_0)V'_1 \\ T'_0(V'_0 + V'_1) + (T'_1 - T'_0)V'_0 \end{bmatrix}, \tag{2}$$

where $t'_{i,j}$ is denoted by the two's complement form. Comparing with (1), one can see that the only difference is that two subtractions are now used to replace the additions in CMP.

Preparation. The proposed algorithm is derived as follows.

Definition 1. Define: $W = \sum_{i=0}^{n-1} w_i x^i$, $D = \sum_{i=0}^{n-1} d_i x^i$, and $G = \sum_{i=0}^{n-1} g_i x^i$, where g_i is the binary coefficient and d_i and w_i are coefficients of 8-bit over ring ($\log_2 q = 8$), and W is the product polynomial. We have

$$\begin{aligned} W =& DG \bmod f(x) = \sum_{i=0}^{n-1} d_i x^i \bmod f(x), \\ =& \sum_{i=0}^{n-1} d_i (Gx^i \bmod f(x)), \end{aligned} \tag{3}$$

where $f(x) = x^n + 1$.

Definition 2. Define $G^{[0]} = Gx^0 \bmod f(x)$, $G^{[1]} = Gx^1 \bmod f(x)$, ..., and $G^{[n-1]} = Gx^{n-1} \bmod f(x)$.

Thus, we can have $G^{[0]} = g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1}$, $G^{[1]} = G^{[0]} x \bmod f(x) = -g_{n-1} + g_0 x + \cdots + g_{n-2} x^{n-1}$, ..., $G^{[N-1]} = G^{[n-2]} x \bmod f(x) = -g_1 - g_2 x - g_3 x^2 - \cdots + g_0 x^{n-1}$, for $f(x) = x^n + 1$ and $x^n \equiv -1$.

Thus, the original polynomial multiplication of (3) becomes

$$W = \sum_{i=0}^{n-1} d_i G^{[i]}, \tag{4}$$

which can be transferred into a matrix-vector product of

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{n-1} \end{bmatrix} = \begin{bmatrix} g_0 & -g_{n-1} & \cdots & -g_1 \\ g_1 & g_0 & \cdots & -g_2 \\ \vdots & \vdots & \ddots & \vdots \\ g_{n-1} & g_{n-2} & \cdots & g_0 \end{bmatrix} \times \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \end{bmatrix}. \tag{5}$$

Definition 3. Define (5) as $[W] = [G] \times [D]$, where $[W]$, $[G]$, and $[D]$ are $n \times 1$, $n \times n$, and $n \times 1$ matrices, respectively. Define again the elements in the matrix as $[W]_{i,1}$, $[G]_{i,j}$, and $[D]_{i,1}$ ($1 \le i, j \le n$), i.e., $[G]_{1,1} = g_0$, $[G]_{2,1} = g_1$, etc.

We can thus follow (2) to rewrite (5) as

$$\begin{bmatrix} W_0 \\ W_1 \end{bmatrix} = \begin{bmatrix} G_0 & G_2 \\ G_1 & G_0 \end{bmatrix} \times \begin{bmatrix} D_0 \\ D_1 \end{bmatrix}, \tag{6}$$

where $[W_i]$ and $[D_i]$ are $\frac{n}{2} \times 1$ vectors ($0 \le i \le 1$), i.e., $[W_0] = [w_0\ w_1\ \cdots\ w_{n/2-1}]^T$, $\cdots$, $[D_1] = [d_{n/2}\ \cdots\ d_{n-1}]^T$; $[G_j]$ are $\frac{n}{2} \times \frac{n}{2}$ Toeplitz matrices ($0 \le j \le 2$), e.g.,

$$[G_0] = \begin{bmatrix} g_0 & -g_{n-1} & \cdots & -g_{n/2+1} \\ g_1 & g_0 & \cdots & -g_{n/2+2} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n/2-1} & g_{n/2-2} & \cdots & g_0 \end{bmatrix}. \tag{7}$$

Note that $[G_2] = -[G_1]$, then we have

$$\begin{aligned} \begin{bmatrix} W_0 \\ W_1 \end{bmatrix} &= \begin{bmatrix} G_0 & -G_1 \\ G_1 & G_0 \end{bmatrix} \times \begin{bmatrix} D_0 \\ D_1 \end{bmatrix} \\ &= \begin{bmatrix} G_0(D_0 + D_1) + (G_1 + G_0)(-D_1) \\ G_0(D_0 + D_1) + (G_1 - G_0)D_0 \end{bmatrix}. \end{aligned} \tag{8}$$

Considerations. The direct hardware implementation of (8) would consume huge resource usage and thus is considered impractical. Besides that, similar to the Karatsuaba method [18], TMVP-based approach also involves the pre-computing related additions, e.g., $(G_1 + G_0)$ in (8), which increases the bit-width to 2-bit (not in favor of iterative deploying).

Proposed Strategy. Based on the above considerations, we propose a *novel mathematical derivation strategy* for the polynomial multiplication of RBLWE-ENC to achieve hardware implementation-friendly, relatively low-complexity, and high-performance operation. In detail, we plan to: (i) derive the three $\frac{n}{2}$-size matrix-vector products into accumulation forms to obtain high-speed computation yet with relatively small complexity; (ii) determine a suitable computational sequence to minimize resource usage (implementation-friendly); (iii) output these accumulation components to produce the final result with low resource usage. Overall, we plan to have the following three steps to derive the algorithm.

Step-I. Without loss of generality, we can consider $[G_0][D_0 + D_1]$ first (the same principle applies to the other two $\frac{n}{2} \times \frac{n}{2}$ matrix-vector products as well). One can see that $[D_0 + D_1]$ is still an $\frac{n}{2} \times 1$ matrix-vector such that we have

$$\begin{aligned}[G_0][D_0 + D_1] =& [G_0]_{:,1}[D_0 + D_1]_{1,1} + \\ & [G_0]_{:,2}[D_0 + D_1]_{2,1} + \cdots + \\ & [G_0]_{:,n/2}[D_0 + D_1]_{n/2,1} \\ =& \sum_{j=1}^{n/2} [G_0]_{:,j}[D_0 + D_1]_{j,1},\end{aligned} \tag{9}$$

which turns the original matrix-vector product into an accumulation format, i.e., $n/2$ elements of $[G_0]_{:,j}$ are respectively multiplied with the matched $[D_0 + D_1]_{j,1}$ and then through $n/2$ cycles of accumulations to produce the final output.

Likewise, we can also have

$$\begin{aligned}[G_1 + G_0][-D_1] =& \sum_{j=1}^{n/2} [G_1 + G_0]_{:,j}[-D_1]_{j,1}, \\ [G_1 - G_0][D_0] =& \sum_{j=1}^{n/2} [G_1 - G_0]_{:,j}[D_0]_{j,1}.\end{aligned} \tag{10}$$

Therefore, all three $n/2$-size matrix-vector products of (8) have become forms of accumulations, where each accumulation unit takes $n/2$ cycles to produce the output.

Benefits. This type of accumulation brings multiple benefits: (i) the $n/2$ elements of one certain column of the main matrix, e.g., $j = 1$ for $[G_0]_{:,j}$, are exactly the $n/2$ elements of the first column of $[G_0]$ (from left), which is easy to be implemented on hardware platform; (ii) the corresponding value from vector-matrix (e.g., $[D_0+D_1]_{j,1}$) is processed one by one as required by the accumulation, which can also be easily implemented with minimized resources (see Section IV); (iii) the values of the related component after the additions (subtraction equals addition under two's complement representation), e.g., $[G_1 + G_0]$ & $[G_1 - G_0]$, still maintain small-size as TMVP is not iteratively deployed (which potentially reduces area usage).

Step-II. It becomes clear that the accumulated results from (9) and (10) need to be added to produce the final output of (8). To maintain a small critical-path implementation, we decided to process these accumulations separately and then add the corresponding results together[1]. In this case, we have

$$\begin{aligned}[W_0] &= [G_0(D_0 + D_1)] + [(G_1 + G_0)(-D_1)], \\ &= \sum_{j=1}^{n/2} [G_0]_{:,j}[D_0 + D_1]_{j,1} + \sum_{j=1}^{n/2} [G_1 + G_0]_{:,j}[-D_1]_{j,1},\end{aligned} \tag{11}$$

and

$$\begin{aligned}[W_1] &= [G_0(D_0 + D_1)] + [(G_1 - G_0)D_0] \\ &= \sum_{j=1}^{n/2} [G_0]_{:,j}[D_0 + D_1]_{j,1} + \sum_{j=1}^{n/2} [G_1 - G_0]_{:,j}[D_0]_{j,1}.\end{aligned} \tag{12}$$

Step-III. Lastly, we propose to process operations of (11) and (12) in parallel. This is because: (a) related elements, such as $[G_1 + G_0]_{:,j}$ and $[G_1 - G_0]_{:,j}$, can be produced through operations on elements from one whole column of the original matrix $[G]$, which reduces potential implementation cost; (b) the final output actually rests on three different accumulation units, i.e., additions are still needed after accumulations, which requires that these accumulations can be executed simultaneously for both high-performance and low-complexity implementation.

Finally, for practical implementation, the accumulated results from three parallel accumulation units are delivered out in serial such that the actual output elements of $[W_0]$ and $[W_1]$ are also collected in serial[2]. This setup also allows low resource usage on the output hardware component design.

The Proposed TMVP-based Polynomial Multiplication Algorithm. The proposed algorithm is thus concluded as:

Algorithm 1: Proposed TMVP-based polynomial multiplication algorithm for RBLWE-ENC

Input : G and D are integer polynomials. // the actual bit-width of the coefficients follow ***Definition 1***;
Output: $W = GD \bmod (x^n + 1)$;

Initialization step
make ready the inputs G and D;
$[\overline{Z_0}] = [\overline{Z_1}] = [\overline{Z_2}] = 0$; // $[\overline{Z_0}]$, $[\overline{Z_1}]$, and $[\overline{Z_2}]$ are $\frac{n}{2} \times 1$ matrices
Main step
for $i = 1$ *to* $n/2$ **do**
  for $j = 1$ *to* $n/2$ **do**
    $[\overline{Z_0}]_{:,1} = [\overline{Z_0}]_{:,1} + [G_0]_{:,j}[D_0 + D_1]_{j,1}$;
    $[\overline{Z_1}]_{:,1} = [\overline{Z_1}]_{:,1} + [G_1 + G_0]_{:,j}[-D_1]_{j,1}$;
    $[\overline{Z_2}]_{:,1} = [\overline{Z_2}]_{:,1} + [G_1 - G_0]_{:,j}[D_0]_{j,1}$;
  end
  $[W_0]_{i,1} = [\overline{Z_0}]_{i,1} + [\overline{Z_1}]_{i,1}$;
  $[W_1]_{i,1} = [\overline{Z_0}]_{i,1} + [\overline{Z_2}]_{i,1}$;
end
Final step
obtain all the coefficients of output W;

which fully fulfills the objectives of the proposed mathematical derivation design strategy.

[1]This step is different from that in [24], where related additions are executed first and then accumulations.

[2]This part is also different from the algorithm of [24], where the output values are processed in parallel.

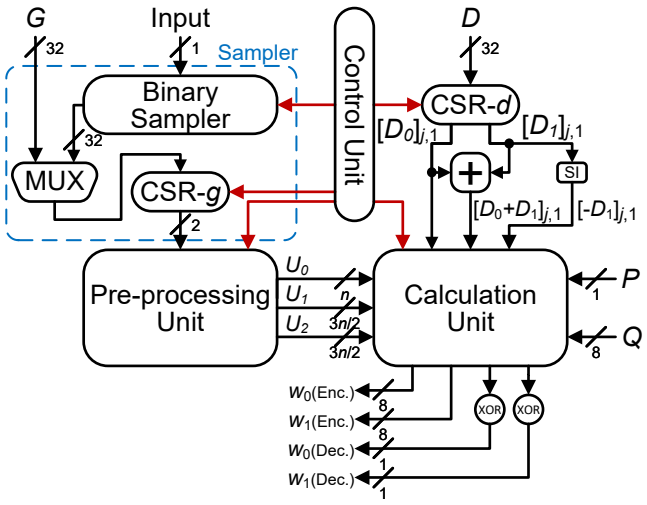

Fig. 2: The proposed accelerator (TINA) for RBLWE-ENC. Note: $[U_0] = [G_0]$, $[U_1] = [G_0 + G_1]$, $[U_2 = [G_1 - G_0]]$ (applies to Figs. 4, 6, and 7).

Besides that, as seen from Fig. 1, the arithmetic operation in the major phases of RBLWE-ENC also involves the addition with other polynomials (one polynomial for decryption and two polynomials for encryption). Meanwhile, a constant error also needs to be added according to the suggestion to a recent report in [17]. The related components (including the decryption decoder) for these operations are all included in the actual hardware implementation (the following section).

IV. TINA: Proposed Hardware Accelerator for RBLWE-ENC

This section focuses on the design process of the proposed accelerator (TINA). The overall architectural overview of TINA for RBLWE-ENC is shown in Fig. 2, which is built from the proposed Algorithm 1 through several novel algorithm-to-architecture mapping techniques.

Overall Description. The proposed accelerator of Fig. 2 contains four main units, namely the sampler, the pre-processing unit, the calculation unit (including the input processing for D), and the control unit. While the sampler is responsible for generating required random binary numbers to be used for the computation involved within certain operational phases of RBLWE-ENC (i.e., encryption/decryption), the pre-processing unit is in charge of producing necessary input signals for the proposed TMVP-based computation (Algorithm 1) within the following unit. The calculation unit focuses on the execution of the main computation steps of Algorithm 1, including the accumulation steps (the input preparation for D is also included) and related output delivery. Finally, the control unit is designed mainly to produce all needed signals for the proper operation of TINA. The details of these units and related hardware design techniques are described below. Note that we assume both G and D inputs are from outside of regular 32-bit (could be other regular bits as well).

Sampler. The sampler is an important component for a complete RBLWE-ENC accelerator, as the binary polynomials (including the secret key) are generated from the binary sampler. So far, however, there are no available reports about the hardware-implemented RBLWE-ENC structures deploying a proper binary sampler to execute related complete operations.

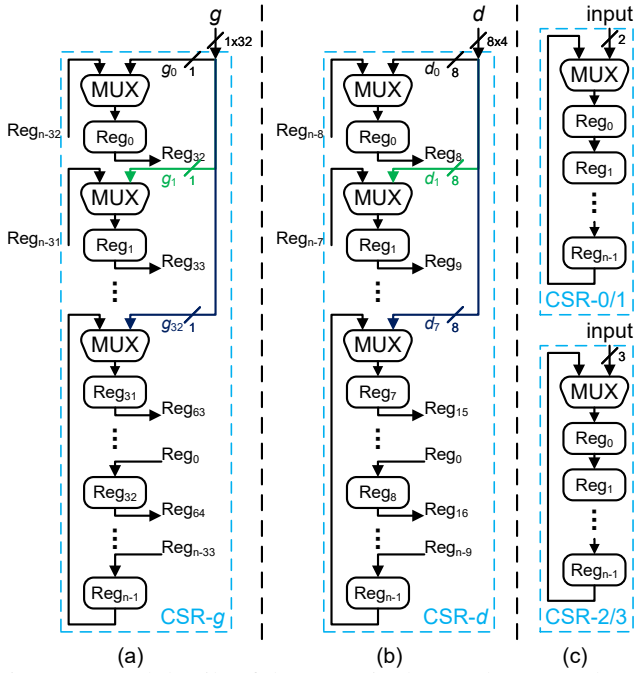

Fig. 3: Internal details of the CSRs in the accelerator (values in the registers are initial loadings), where CSR-0/1 and CSR-2/3 are used in the pre-processing unit, as shown in Fig. 4.

The **major challenge** of designing a sampler component for RBLWE-ENC accelerator lies in two aspects: (i) the sampler can generate true random binary numbers while maintaining a relatively small area usage; (ii) the need for a proper wrapper to transfer the generated random binary numbers into correct signals for following steps of computation.

For the first aspect of the challenge, in our proposed accelerator design, we have deployed an open-accessed true random number generator (TRNG) as the binary sampler [26]. This sampler is highly optimized as it can generate one 32-bit binary value per cycle while maintaining small area usage. The interested readers can refer to the original paper for detailed internal structure and related source code. Note that a 1-bit input is needed to activate the TRNG to produce the output.

While for the second aspect of the challenge, we have designed a novel serial-in parallel-out circular-shift register (CSR) to transfer the 32-bit output of the TRNG into n number of regular 1-bit signals. As shown in Fig. 3, there are in total n number of registers and 32 MUXes. In the loading stage, the selection of the 32 MUXes are set to the right channel so that the 32 output bits from the sampler can be serially loaded into the registers. After that, the selection signals of these MUXes are switched to the left channel such that the values loaded in the registers are circularly shifted once per cycle. Note that due to the processing bit-width (32-bit), the output of Register-0 (Reg_0) is connected to the input of Register-32 (Reg_{32}), similar to the other registers. Finally, the outputs of two registers (two bits) are connected to the outside, namely Reg_0 and $\text{Reg}_{\frac{2}{n}-1}$, to coordinate with the following computation unit to form the needed signals from $[G_0]$ and $[G_1]$ according to Algorithm 1.

Pre-processing Unit. This computation unit, as shown in

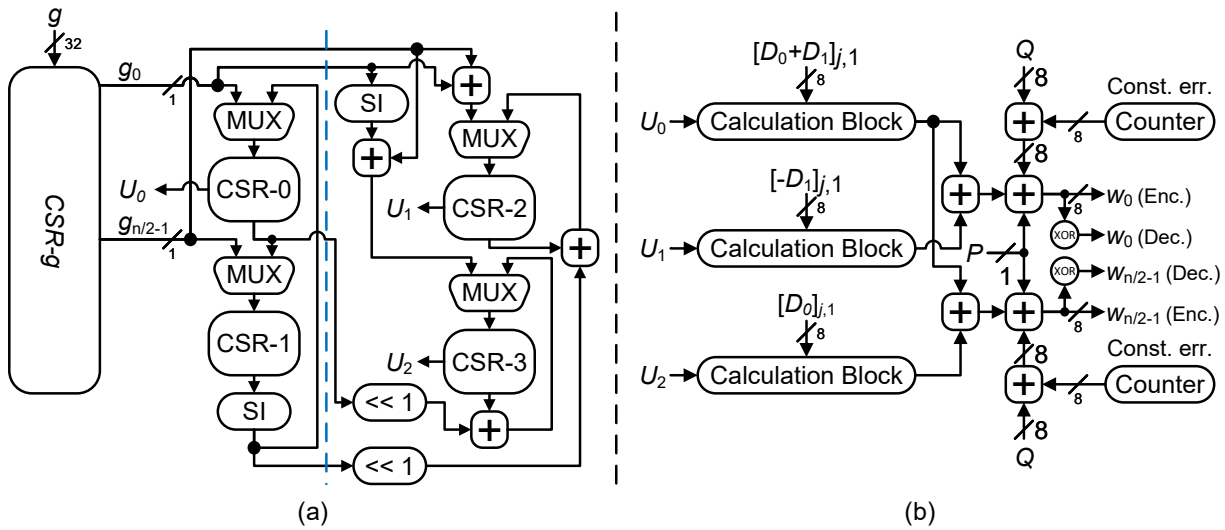

Fig. 4: The internal structures of the pre-processing unit and calculation unit.

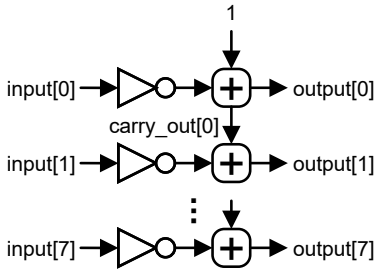

Fig. 5: The internal structures of the sign inverter.

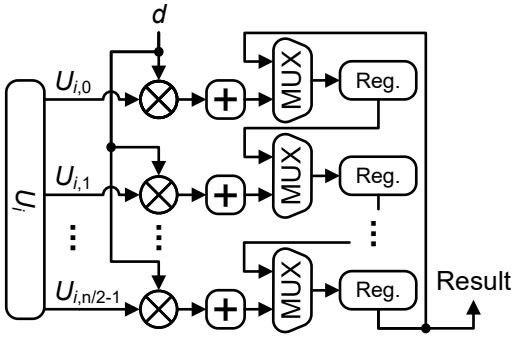

Fig. 6: Internal details of the calculation block.

Fig. 4, is another major component for the proposed RBLWE-ENC accelerator. According to the procedures shown in Lines 5-7 of Algorithm 1, we can consider first the two important operations involved here: the two corresponding elements from matrix $[G_0]$ and matrix $[G_1]$ need to be added together to be multiplied with one coefficient from $[-D_1]$ (Line 6); while the same elements from matrix $[G_0]$ and matrix $[G_1]$ are subtracted (actually can be seen as the addition with another element with sign inverted) and then multiplied with the related coefficient from $[D_0]$ (Line 7). Based on Step-III of the **Proposed Mathematical Derivation Strategy** in Section III, these two operations should be executed at the same time.

Multi-CSR Processing Technique. To realize the needed arithmetic operations, we have proposed a multi-CSR processing technique. As shown in Fig. 4 (a), we have used four CSRs, four MUXes, four adders, and two sign inverters (SIs) to realize the needed operations. The internal structures of these CSRs are shown in the dotted boxes of Fig. 3 (c), respectively, where the only difference is the processing bit-width. As specified in Step-I of Section III (paragraph of "**Benefits**"), the first columns (from left) of matrices $[G_0]$ and $[G_1]$ are actually the first column of $[G]$, two MUXes connecting with CSR-0 and CSR-1 can function to: (i) load with respective initial values; (ii) circular shifting the values in two CSRs to produce correct elements for each column of $[G]$ (with the help of SI); (iii) produce needed values from $[G_0]$ and $[G_1]$, respectively, based on the switching of MUXes. In this case, the output of CSR-0, CSR-2, and CSR-3 are corresponding values from $[G_0]$, $[G_0 + G_1]$, and $[G_1 - G_0]$, respectively. Note that the values stored in CSR-0 and CSR-1 are in the range of $[-1, 1]$ (because of the existence of SI), which become $[-2, 2]$ in CSR-2 and CSR-3 due to the involved addition/subtraction operations (as shown in Fig. 4(a), a bit-extension cell is needed when the value is transferred from CSR-0/CSR-1 to CSR-2/CSR-3). Finally, we want to mention that SI comprises eight inverters followed by the same number of half-adders (see Fig. 5), following the sign inversion method for the two's complement representation.

Overall, the proposed multi-CSR processing technique produces needed column elements for $[G_0]$, $[G_1 + G_0]$, and $[G_1 - G_0]$, respectively (we have used $[U_0], [U_1]$, and $[U_2]$ in Fig. 2), for the computation executed in the following unit.

Calculation Unit. The calculation unit mainly executes the accumulation operation of Lines 5-7 of Algorithm 1 (including the input processing for D) as well as the final output delivery (Lines 9-10 in Algorithm 1). To fully accomplish the related operations, we have used another new CSR for D to generate the needed $[D_0]_{j,1}$, $[-D_1]_{j,1}$, and $[D_0 + D_1]_{j,1}$, respectively, according to Algorithm 1. As shown in Fig. 3(b), the new CSR receives the 32-bit input and then transfers the whole data into an 8-bit style. Then, the $[D_0]_{j,1}$ and $[D_1]_{j,1}$ from the CSR-d are used as the two inputs of an adder to produce the needed $[D_0]_{j,1}$, $[D_0 + D_1]_{j,1}$, and $[-D_1]_{j,1}$, respectively, to be fed into the calculation unit.

While for the accumulation-related operation, as shown in Fig. 4 (b), there are in total three calculation blocks involved

TABLE II: Major Area-Time Complexities for The Proposed RBLWE-ENC Accelerator (TINA) and The Existing Designs

Design	AND	Adder[!]	Register[!]	MUX[!]	Extra input/output resources	Latency[1]	Complexity	CD?*
[13][2]	–	2n[1]	n	n	three $\log_2 q$-bit n-size SRs	n	$O(n^2)$	N
[4][2]	n	n	n	$n+1$	three $\log_2 q$-bit n-size SRs	n	$O(n^2)$	N
[11]	n	n	n	n	three $\log_2 q$-bit n-size SRs	n	$O(n^2)$	N
[16] Arch.-I	n	n	n	n	two $\log_2 q$-bit n-size SRs	n	$O(n^2)$	N
[17]	n	n	n	n	three $\log_2 q$-bit n-size SRs	n	$O(n^2)$	N
[18] $u=1$	$2(n-1)$	$3(n-1)+3$	$\frac{9n}{2}-3$	$3n+5$	two 2-bit $\frac{n}{2}$-size SRs[3]	$\frac{n}{2}$	$O(\frac{3n^2}{4})$	N
TINA	–	$\frac{3}{2}(n-1)+9$	$\frac{3n}{2}+1$	$\frac{3n}{2}+6^{\#}$	see Figs. 3 and 4	$\frac{n}{2}$	$O(\frac{3n^2}{4})$	Y

Note: we follow the existing designs' reporting styles to list the complexities based on the decryption phase (particularly the timing-complexity). The latency listed here refers to the major computation cycles based on the decryption phase.
We have also listed specific input/output processing resources (if available). SR: shift-register.
[!]: The adders, registers, and MUXes listed are all $\log_2 q$-bit.
*: CD: complete design (a complete accelerator including sampling for both encryption and decryption operations).
[1]: This structure needs n $\log_2 q$-bit subtractors and $\log_2 q$-bit adders.
[2]: These two designs belong to the parallel-in parallel-out structures (but related input/output processing resources were not specified). For [13], maybe three $\log_2 q$-bit n-size shift-registers (or multiple BRAMs) are needed. For [4], two $\log_2 q$-bit n-size shift-registers and one $\log_2 q$-bit n-size output buffer are needed, based on the re-implementation in [15].
[3]: This design also needs two 1-bit $\frac{n}{2}$-size SRs and one 2-bit $\frac{n}{2}$-size SR, which is equivalent to two 2-bit $\frac{n}{2}$-size SRs.
#: The MUXes used for multipliers (Fig. 7) are also included and calculated here (equivalent number).
Note that TINA needs two 2-bit $\frac{n}{2}$-size SRs and two 3-bit $\frac{n}{2}$-size SRs in the pre-processing unit, as well as one CSR-g (1-bit) and one CSR-d (8-bit). All these SRs are shown in Figs. 2, 3, and 4.

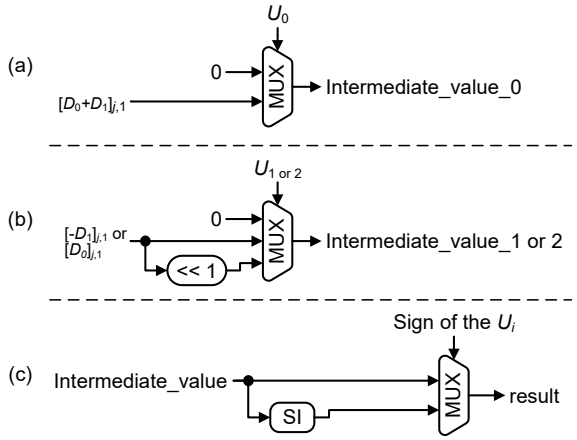

Fig. 7: Internal details of the multipliers.

within this unit, corresponding with the three $\frac{n}{2} \times \frac{n}{2}$ matrix-vector products of Algorithm 1 (Lines 5-7), respectively. The internal structure of the calculation block is shown in Fig. 6, which includes $\frac{n}{2}$ number of point-wise multipliers processing in parallel followed by the same number of adders, MUXes, and Registers (Reg.). Note that we have used MUX-based point-wise multipliers, where the size of the MUX is determined by the input bit-width, as shown by the example in Fig. 7. For the first calculation block, we used the MUX-based point-wise multiplier in Fig. 7(a); for the rest two calculation blocks, the MUX-based point-wise multipliers are the same as that in Fig. 7(b). After the intermediate value is calculated, the second MUX determines whether the result is positive or negative to produce the final result, as shown in Fig. 7(c). Besides that, followed by the point-wise multiplier are an adder, a MUX, and a register, where the adder and register form the accumulation unit with the help of the MUX. After $\frac{n}{2}$ cycles of accumulation, the values stored in the registers can be serially delivered out again with the help of these MUXes.

Apart from the three calculation blocks, there are still four adders needed to execute the final addition with the constant error (Const. err.) as recommended by the recent report [17]. Meanwhile, the final additions with the corresponding coefficients of two polynomials (namely P and Q) are needed to produce correct encryption/decryption results. Note that an XOR (decoder) attached with the two most significant bits of the output value produces the decryption output [13].

Control Unit. A control unit is required to coordinate all the components/cells in the accelerator of Fig. 2 to function in a proper manner. The working status of the architecture of Fig. 2 can be split into three stages, namely loading, computing, and delivery, which can be realized by a finite state machine (FSM). The loading stage simply refers to the loading of the sampler output or input coefficients of G & D into the CSRs; while the computing stage denotes the computation of the required operations according to mainly Lines 5-7 of Algorithm 1 to produce the correct output; and the delivery stage refers to the final output processing (Lines 9-10). Finally, the control unit generates these signals to determine the accelerator's operational phase, e.g., encryption/decryption.

Overall Operation. The proposed accelerator of Fig. 2 requires only $n/2$ cycles of accumulation to generate the output result, benefiting from the proposed TMVP-based polynomial multiplication algorithm as well as related design techniques.

For decryption, as shown in Fig. 2, G and D are 32-bit inputs. G needs $256/32 = 8$ cycles to complete the preloading stage after stored in CSR-g through the first MUX. At the same time, D is directly stored in CSR-d, which takes a total of $256/32 \times 8 = 32$ cycles. The following $n/2$ cycles are the warm-up phase of the pre-processing unit (Fig. 4(a)), and CSR-g is in the circularly-shifting status (the outputs attached to registers loading initial g_0 and $g_{n/2-1}$ are delivered out). Note that at this time, these four MUXs in Fig. 4(a) all turn on the left input. The next $n/2$ cycles belong to the calculation phase, and these four MUXes all turn on the right input to produce U_0, U_1, and U_2, respectively. Meanwhile, $[D_0]_{j,1}$, $[-D_1]_{j,1}$, and $[D_0 + D_1]_{j,1}$ are produced from CSR-d. Adding the results of the first calculation block and the second calculation block generates the intermediate values of W_0 (serial format); while adding the results of the first and

third calculation blocks produces the intermediate values of W_1 (in serial). These two intermediate values, plus the serial-fed P, Q, and constant error, produce the final result.

The encryption operation is similar to the decryption operation. However, the overall calculation process needs to be performed twice to produce ciphertext c_1 and c_2 (see Fig. 1). In the first round, G, D, and P are used as inputs, while in the second round, G, D, P, and Q are used as inputs.

The key generation phase involves the same operation as the decryption phase in the calculation part but is slightly different in the data input setup. In the beginning, the MUX in the sampler unit chooses the data from the sampler as input and passes it to CSR-g. Note that the binary sampler generates r_2 in the preloading stage and outputs r_1 in the calculation stage, which is directly connected to the P port of the calculation unit (only 1 bit of the data is used each cycle). Also, the input for D now becomes $-a$ (connecting with Fig. 1).

V. Evaluation: Complexity & Comparison

This section presents a comprehensive evaluation of the proposed accelerator, from theoretical complexity to implementation comparison, to showcase the efficiency of TINA.

Complexity Analysis. As described in Section IV, the proposed TINA contains a sampler unit, a pre-processing unit, a calculation unit (including the input processing resources for D), and a control unit. While the actual number of resource usage cannot be exactly calculated due to the sophisticated setup for different phases' operations, we have listed the major area-time complexities of TINA in Table II along with the existing ones.

As seen from Table II, TINA overall has smaller area-time complexities than the existing ones. Though the rough estimated resource usage of TINA, such as the number of adders, is about ≥ 1.5 times than the majority of the existing ones (except [13] and [18]), the computational latency of TINA is only half of those ones, which indicates that TINA has a bigger chance to obtain smaller area-time complexities. In particular, when comparing with the very recent Karatsuba-based accelerator [18], TINA actually has better smaller resource usage (e.g., significantly less number of adders and registers). Even when the resource usage of input/output is considered, TINA still has an advantage in area usage (see Table V later). Most importantly, TINA has a complete hardware setup for encryption & decryption operations (with sampling), which is missing in all existing ones.

Implementation. To further evaluate the actual performance of the proposed TMVP-based RBLWE-ENC accelerator, we have coded the constructed accelerator of Fig. 2 with VHDL. The binary sampler is adopted from the open-source code [26]. Finally, we have used the Vivado 2020.2 to synthesize and implement it on the targeted AMD-Xilinx Virtex-7 XC7V2000t and Kintex-7 XC7K325t devices. Note that we have followed the existing designs like [4], [11], [18] to select the parameter sets, i.e., $(n, q) = (256, 256)$ and $(n, q) = (512, 256)$.

Area Complexity. The area usage for the designed accelerator is listed in Table III, where the frequency is set as 200MHz. One can notice that the pre-processing unit and calculation unit occupy the majority of resource usage of the accelerator, namely 16,496 LUTs and 12,858 FFs. With the combination of other components, the whole accelerator occupies 17,187 LUTs and 13,272 FFs (or equivalent 5,132 slices).

TABLE III: Area-Complexity for The Proposed TINA (Virtex-7). The Clock Frequency Constraint is Set As 200MHz.

Building block	LUTs	FFs	Slices
$n = 512$			
Sampler unit	31	32	5*
Pre-processing unit	7,458	2,565	3,003*
Calculation unit	9,038	10,293	4,517*
Other blocks	119	99	*
Whole accelerator	17,158	13,501	5,132
(% of overall FPGA)	1.40	0.55	1.68

Note: no DSPs and BRAMs usage for the proposed accelerator.
*: The slice number reported by Vivado involves overlapping calculations.

TABLE IV: Time-Complexity for The Proposed Accelerator (TINA) for Different Security Levels of RBLWE-ENC, Where The Clock Frequency Constraint is Set to 200MHz.

n	Cycles/Time (μs)		
	KeyGen	Encryption	Decryption
256	448/2.24	1,120/5.60	448/2.24
512	896/4.48	1,840/9.20	896/4.48

Timing Results. The time-complexity of the proposed hardware accelerator, in terms of the number of cycles and related computational time, with respect to different security levels of RBLWE-ENC, are calculated and listed in Table IV, where the frequency constraint is set as 200MHz. For instance, for $n = 512$, the implemented accelerator requires 896, 1,840, and 896 cycles to execute the key generation, encryption, and decryption operations, respectively, which is equivalently 4.48μs, 9.20μs, and 4.48μs, respectively.

Comparison With The Existing Implementations. We have also listed the area-time complexities of the proposed accelerator along with the existing implementations for comparison, including the available Karatsuba-based design of [18] and other high-speed ones [13], [4], [15], [16], [17]. Note that: (i) the designs of [14], [11] were implemented for Intel devices ([18] has already shown its efficiency over them); (ii) designs like [12] belong to the low-speed category with very long latency cycles. We thus do not include them for comparison.

Comparison Consideration. Due to the fact that the existing designs were mainly built on the major arithmetic operation of the decryption phase of RBLWE-ENC, these implementations hence did not include as many resources as the proposed one (which has a complete setup for different phases' operations). Therefore, a balanced consideration is needed when TINA is compared with the existing ones.

As shown in Table V, when comparing with the existing RBLWE-ENC implementations, the proposed accelerator has significantly outperformed the existing designs of [13], [4], in aspects of both area-delay product (ADP) and throughput.

When comparing with [15], we want to mention that this existing design has a simple structure with small input processing resources (which executes the operation of one polynomial multiplication and one polynomial addition). Hence, a direct comparison is not feasible in this case. But as indicated in [16] that its proposed architectures have better area-time

TABLE V: Comparison of FPGA Implementation Performance (AMD-Xilinx Devices)

Design	n	Phase*	Device	LUT	FF	Slice	Fmax	Latency[1]	Delay	ADP[2]	Throughput[3]	CD?
[13]	256	Dec.	Spartan-6	6,728	6,813	1,874	101	262	2,594	4,861	0.099	N
[4][$]	256	Dec.	Virtex-7	5,153	2,151	1,701	261	257	985	1,675	0.260	N
[15][4] $u=1$	256	Dec.	Virtex-7	3,600	2,568	1,146	415	256	617	707	0.415	N
[16] Arch.-I	256	Dec.	Virtex-7	5,324	6,469	1,781	357	256	717	1,277	0.360	N
[17]	256	Dec.	Virtex-7	7.6k	6.2k	2.3k	514	259	504	1,159	0.508	N
[18] $u=1$	256	Dec.	Virtex-7	12,360	7,163	3,501	395	128	324	1,135	0.790	N
TINA	256	Dec.	Virtex-7	10,264	6,737	3,030	339	128	378	1,145	0.678	Y
TINA	256	Enc.	Virtex-7	10,264	6,737	3,030	339	256	776	2,289	0.339	Y
[15][4] $u=1$	256	Dec.	Kintex-7	3,600	2,568	1,134	394	256	650	737	0.394	N
[16] Arch.-I	256	Dec.	Kintex-7	5,159	6,467	1,963	347	256	738	1,449	0.340	N
[18] $u=1$	256	Dec.	Kintex-7	12,360	7,163	3,606	384	128	333	1,202	0.679	N
TINA	256	Dec.	Kintex-7	10,225	6,697	3,114	367	128	367	1,142	0.698	Y
TINA	256	Enc.	Kintex-7	10,225	6,697	3,114	367	256	733	2,283	0.349	Y
[4][$]	512	Dec.	Virtex-7	10,285	4,249	3,289	263	513	1,951	6,417	0.262	N
[15][4] $u=1$	512	Dec.	Virtex-7	7,184	5,128	2,208	399	512	1,283	2,833	0.399	N
[16] Arch.-I	512	Dec.	Virtex-7	11,123	12,851	3,668	357	512	1,434	5,260	0.357	N
[17]	512	Dec.	Virtex-7	15k	12.3k	4.6k	470	515	1,096	5,042	0.470	N
[18] $u=1$	512	Dec.	Virtex-7	24,739	14,407	6,871	392	256	1,306	8,974	0.784	N
TINA	512	Dec.	Virtex-7	20,365	13,617	5,994	303	256	845	5,066	0.606	Y
TINA	512	Enc.	Virtex-7	20,365	13,617	5,994	303	512	1,690	10,131	0.303	Y
[18] $u=1$	512	Dec.	Kintex-7	24,736	14,406	6,949	380	256	674	4,684	0.760	N
TINA	512	Dec.	Kintex-7	20,334	13,594	6,067	309	256	829	5,028	0.618	Y
TINA	512	Enc.	Kintex-7	20,334	13,594	6,067	309	512	1,657	10,055	0.309	Y

CD: complete design, i.e., a complete accelerator (including sampling) for all three phases' operations.
*: The existing designs were mostly designed according to the arithmetic operation in the decryption phase. Hence we use this phase to conduct the calculations. Nevertheless, we have added the encryption phase's performance for TINA as it can operate in both phases. Dec.: decryption; Enc.: encryption.
$: We used the re-implemented data from [15], which was justified in [15].
1: Latency cycles, denote the major computation time of executing the decryption phase.
2: ADP=#Slice×delay (Dec.) $\times 10^3$, where Delay=critical-path×latency. Unit for Fmax: MHz. Unit for delay: ns.
3: Throughput = n/Delay.
4: The design of [15] did not involve the input resource usage (except one 1-bit n-size SR). Hence, its comparison with TINA needs more balanced consideration. The same principle applies to the other existing designs since the proposed accelerator has a more complete structural/operational setup than the other designs as well as more input processing resources.

complexities than [15], we conclude that TINA actually has better actual ADP than [15] (as indicated in Table V that TINA has smaller ADP than [16]). Moreover, TINA also has a more complete structure and operational setup than [15] and [16].

When compared with [17], the proposed TINA has comparable ADP but with a much higher throughput. The design of [17] has smaller area usage than TINA, but the primary reason is that [17] had fewer input processing resources and internal components than the proposed accelerator. Therefore, the actual area-time complexities of [17] are much larger than TINA. Finally, we want to mention again that TINA is a complete accelerator while [17] was designed mainly for the major arithmetic operation of the decryption phase (no sampler and related components for all phases' complete operations).

As seen from Table V, the design of [18] has a very comparable ADP and throughput with TINA. Again, as we mentioned above, [18] did not have a more complete structural and operational setup than TINA, the actual area-time complexities of [18] are much larger than the proposed accelerator. Besides that, we also want to mention TINA's other advantages over [18]: (i) TINA has a smaller area occupation than [18], so TINA is more suitable for lightweight high-performance applications; (ii) TINA is based on TMVP, which involves much easier output setup than [18] and ease of popularization of the proposed technique (see the comparison of Fig. 2 with Fig. 5 of [18]); (iii) TINA can operate in all phases of RBLWE-ENC, which the design of [18] does not possess. Overall, we conclude that TINA has superior performance than [18].

Discussion. As the major focus of this paper is to develop a novel fast-algorithm-initiated complete accelerator for RBLWE-ENC, we compare its performance only with the existing RBLWE-ENC designs. Besides that, as the existing designs were also using shift registers for input processing (though not fully), it is also very appropriate that the proposed accelerator is competing with the similar design style structures in the literature (memory-based design can be our future option to reduce the input/output processing cost). Nevertheless, we want to mention that the existing designs like [4], [18] have demonstrated their structures' efficiency over the other types of PQC implementations, including Kyber and Saber [18]. Following the foundation of this work, we hope a more comprehensive comparison can be made in the future when RBLWE-ENC is developed into a more mature, lightweight PQC scheme.

Future work can also be more focused on: (a) developing more efficient fast algorithms for non-NTT style polynomial multiplication in RBLWE-ENC; (b) finding more effective strategies to implement the PQC scheme when employing fast algorithms; (c) conducting a series of security-related analyses on accelerator implementations.

Other Works. We also want to mention other PQC works in the field [27], [28], [29], [30], [31]. Due to the scheme difference, we do not explicitly discuss them here. Nevertheless, these designs are important to the hardware PQC design field.

VI. Conclusion

In this paper, we propose a novel TMVP-based RBLWE-ENC accelerator. Firstly, we have derived the polynomial multiplication of the RBLWE-ENC into the proposed TMVP-based algorithm. Then, we built the proposed accelerator based on the proposed algorithm along with the help of several novel design techniques. Finally, a detailed evaluation process, including complexity analysis and implementation, has shown that the proposed accelerator has significantly better balanced area-time complexities than the state-of-the-art solutions. To the authors' best knowledge, this is the first report on the RBLWE-ENC accelerator with complete operational phases. The research outcome of this work is expected to generate significant impacts on lightweight PQC development as well as its further standardization.

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[2] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, *et al.*, "Status report on the second round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2020.

[3] 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 Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 383–394, 2021.

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

[5] "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," p. slides page 4, 2022.

[6] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[7] D. Micciancio and C. Peikert, "Hardness of SIS and LWE with small parameters," in *Annual cryptology conference*, pp. 21–39, Springer, 2013.

[8] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pp. 1–23, Springer, 2010.

[9] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proceedings of the 2nd ACM international workshop on IoT privacy, trust, and security*, pp. 2–9, 2016.

[10] F. Göpfert, C. van Vredendaal, and T. Wunderer, "A hybrid lattice basis reduction and quantum search attack on LWE," in *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*, pp. 184–202, Springer, 2017.

[11] B. J. Lucas, A. Alwan, M. Murzello, Y. Tu, P. He, A. J. Schwartz, D. Guevara, U. Guin, K. Juretus, and J. Xie, "Lightweight hardware implementation of binary ring-lwe pqc accelerator," *IEEE Computer Architecture Letters*, vol. 21, no. 1, pp. 17–20, 2022.

[12] K. Shahbazi and S.-B. Ko, "An optimized hardware implementation of modular multiplication of binary ring lwe," *IEEE Transactions on Emerging Topics in Computing*, 2023.

[13] A. Aysu, M. Orshansky, and M. Tiwari, "Binary ring-lwe hardware with power side-channel countermeasures," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1253–1258, IEEE, 2018.

[14] 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 *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1279–1284, IEEE, 2021.

[15] J. Xie, P. He, X. Wang, and J. L. Imana, "Efficient hardware implementation of finite field arithmetic $ab + c$ for binary Ring-LWE based post-quantum cryptography," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1222–1228, 2021.

[16] 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 8, pp. 3297–3307, 2022.

[17] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 8, pp. 1007–1019, 2022.

[18] P. He, Y. Tu, J. Xie, and H. Jacinto, "KINA: Karatsuba initiated novel accelerator for Ring-Binary-LWE (RBLWE)-based post-quantum cryptography," *IEEE Trans. Very Large-Scale Integration (VLSI) Systems*, pp. 1–13, 2023.

[19] J. M. Pollard, "The fast fourier transform in a finite field," *Mathematics of computation*, vol. 25, no. 114, pp. 365–374, 1971.

[20] H. Fan and M. A. Hasan, "A new approach to subquadratic space complexity parallel multipliers for extended binary fields," *IEEE Transactions on Computers*, vol. 56, no. 2, pp. 224–233, 2007.

[21] M. Cenk, C. Negre, and M. A. Hasan, "Improved three-way split formulas for binary polynomial and toeplitz matrix vector products," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1345–1361, 2012.

[22] C.-Y. Lee and P. K. Meher, "Area-efficient subquadratic space-complexity digit-serial multiplier for type-ii optimal normal basis of $gf(2\{m\})$ using symmetric tmvp and block recombination techniques," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 12, pp. 2846–2855, 2015.

[23] C.-Y. Lee and P. K. Meher, "Speeding up subquadratic finite field multiplier over gf (2m) generated by trinomials using toeplitz matrix-vector with inner product formula," in *2011 Fifth International Conference on Genetic and Evolutionary Computing*, pp. 232–236, IEEE, 2011.

[24] P. He and J. Xie, "Novel implementation of high-performance polynomial multiplication for unified KEM saber based on TMVP design strategy," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pp. 1–8, IEEE, 2023.

[25] D. Micciancio, "On the hardness of learning with errors with binary secrets (2018)," *URL: http://cseweb. ucsd. edu˜ daniele/papers/BinLWE. pdf*.

[26] D. B. Thomas and W. Luk, "FPGA-optimised uniform random number generators using LUTs and shift registers," in *2010 International Conference on Field Programmable Logic and Applications*, pp. 77–82, IEEE, 2010.

[27] C. P. Renteria-Mejia and J. Velasco-Medina, "High-throughput ring-lwe cryptoprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2332–2345, 2017.

[28] Z. Chen, Y. Ma, T. Chen, J. Lin, and J. Jing, "High-performance area-efficient polynomial ring processor for crystals-kyber on fpgas," *Integration*, vol. 78, pp. 25–35, 2021.

[29] Y. Tu, P. He, Ç. K. Koç, and J. Xie, "LEAP: Lightweight and efficient accelerator for sparse polynomial multiplication of HQC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[30] T. Wang, C. Zhang, P. Cao, and D. Gu, "Efficient implementation of dilithium signature scheme on fpga soc platform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 9, pp. 1158–1171, 2022.

[31] A. Aysu, B. Yuce, and P. Schaumont, "The future of real-time security: Latency-optimized lattice-based digital signatures," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 3, pp. 1–18, 2015.

Tianyou Bao received his MS degree in Computer Science from Georege Washington University in 2021. He is currently working towards a Ph.D. degree in Computer Engineering at Villanova University. His research interests include post-quantum cryptography, hardware security, and application of machine learning in security systems.

Pengzhou He received his BS degree in Intelligence Science and Technology from University of Science and Technology Beijing in 2019. He is currently working towards a Ph.D. degree in Computer Engineering at Villanova University. His research interests include post-quantum cryptography, hardware security, high-performance IoT devices, and application of machine learning in security systems.

Shi Bai received his Ph.D. in Computer Science from the Australian National University in 2012. He is currently an Associate Professor at the Department of Mathematical Sciences at Florida Atlantic University. His research interests include post-quantum cryptography, algorithmic number theory and analysis of algorithms.Dr. Bai has served as technical committee member for reputed conferences such as Asiacrypt, CT-RSA, PQCrypto and etc. His research has been supported by the NSF CAREER Award, NSF collaborative grant, NIST grant, and CyberFlorida capacity building grant.

Jiafeng (Harvest) Xie (SM'20) received the M.E. and Ph.D. from Central South University and University of Pittsburgh, in 2010 and 2014, respectively.

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

Dr. Xie has served as technical committee member for many reputed conferences such as HOST, ICCAD, and DAC. He served as Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS previously. Currently, he is serving as Associate Editor for MICROELECTRONICS JOURNAL, IEEE ACCESS, and IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS. 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.