APAS: Application-Specific Accelerators for RLWE-based Homomorphic Linear Transformations

Song Bian, *Member, IEEE* Dur E Shahwar Kundi, *Member, IEEE* Kazuma Hirozawa,
Weiqiang Liu, *Senior Member, IEEE* Takashi Sato, *Senior Member, IEEE*

***Abstract*—Recently, the application of multi-party secure computing schemes based on homomorphic encryption in the field of machine learning attracts attentions across the research fields. Previous studies have demonstrated that secure protocols adopting packed additive homomorphic encryption (PAHE) schemes based on the ring learning with errors (RLWE) problem exhibit significant practical merits, and are particularly promising in enabling efficient secure inference in machine-learning-as-a-service applications. In this work, we introduce a new technique for performing homomorphic linear transformation (HLT) over PAHE ciphertexts. Using the proposed HLT technique, homomorphic convolutions and inner products can be executed without the use of number theoretic transform and the rotate-and-add algorithms that were proposed in existing works. To maximize the efficiency of the HLT technique, we propose APAS, a hardware-software co-design framework consisting of approximate arithmetic units for the hardware acceleration of HLT. In the experiments, we use actual neural network architectures as benchmarks to show that APAS can improve the computational and communicational efficiency of homomorphic convolution by 8× and 3×, respectively, with an energy reduction of up to 26× as compared to the ASIC implementations of existing methods.**

***Index Terms*—Homomorphic Encryption, Ring Learning with Errors, Secure Inference, Neural Networks, Homomorphic Linear Transformation, Number Theoretic Transform**

I. Introduction

Due to the powerful blind computing capability and succinct ciphertext representations, ring learning with errors (RLWE) based homomorphic encryption (HE) schemes are known to be efficient in constructing secure multi-party computation (MPC) protocols. As suggested by previous works [1], HE is extremely efficient in processing linear operations (e.g., matrix-vector products) when compared to other MPC protocols such as secret sharing [2], [3] or Yao's garbled circuit [4], [5]. As a result, over the past several years, significant advancements were observed regarding the efficient utilization of HE in various MPC applications, notably in the field of secure inference (SI) based on convolutional neural networks (NN) [1], [6].

This work was partially supported by JSPS KAKENHI Grant No. 20K19799, 20H04156, 20K21793, JST CREST Grant No. JPMJCR19K5, JST PRESTO Grant No. JPMJPR20M7, and National Natural Science Foundation of China under Grant No. 62022041, 61871216. The authors also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

S. Bian, K. Hirozawa, and T. Sato are with the Department of Communications and Computer Engineering, Kyoto University, Kyoto 606-8501, Japan (e-mail: sbian@easter.kuee.kyoto-u.ac.jp; takashi@easter.kuee.kyoto-u.ac.jp).

D.S. Kundi*, and W. Liu are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: dureshahwar@nuaa.edu.cn; liuweiqiang@nuaa.edu.cn). * She is also with National University of Sciences and Technology, Pakistan.

Unfortunately, the software-hardware co-design framework for HE-based MPC schemes remain somewhat unexplored. A number of works have discussed the general hardware design for RLWE-based (fully) HE schemes, most of which are proposed for the number-theoretic transform (NTT) based approaches [7]–[13], with a limited number of works that utilize the standard polynomial multiplication algorithms [14], [15]. However, we point out that most of the existing hardware accelerators either only focus on traditional cryptographic applications (e.g., public-key encryption schemes), or are proposed for general HE constructions without real-world application in mind. Since HE requires drastically different hardware platforms depending on the application, the benefits of existing hardware platforms are obscure. For example, in a recent work [11], a hardware architecture for homomorphically searching through a 2^{16}-record database is proposed, where the parameters are set to ensure a 96-bit security level. Nonetheless, it is not obvious how such search capability can be of practical merit, as most of the real-world databases contain far more records than 2^{16}. Similarly, in [12], the performance statistics of a single ciphertext multiplication are reported for extremely large RLWE parameters (e.g., $n = 2^{15}$ and q is a 1440-bit integer). While the hardware acceleration proves to be fruitful, it is not likely that practical applications can tolerate such level of performance overhead. Hence, while using a set of fictional parameters suffices to show the benefit of application-specific accelerators, abstract designs proposed in existing works cannot be readily deployed to real-world tasks.

In this work, we propose a new homomorphic linear transformation (HLT) technique for the efficient evaluation of NN-based SI schemes. Since the proposed HLT allows for simple arithmetic operations, we also devise APAS, a hardware-software co-design framework that dynamically instantiates application-specific hardware architectures. While it is commonly believed that NTT is an indispensable design element in implementing efficient HE operators [1], [11], [16], the proposed HLT directly performs linear transformation over the homomorphic ciphertexts, without the use of expensive NTT and homomorphic rotations. By adopting the proposed HLT technique, we can use dynamic approximate hardware architectures [17] in APAS to further improve the practical efficiency of NN-based SI. The main contributions of this work are summarized as follows.

- **Efficient Homomorphic Linear Transformation**: To the best of our knowledge, we are the first to formulate homomorphic operations over RLWE-based HE ciphertexts as linear transformations without the use of NTT. Our fundamental insight is that linear transformations over vector spaces preserve the decipherability of the RLWE ciphertext. A proof-of-concept implementation of our technique is available at [18].

- **Dynamic Hardware Accelerator for HE**: To the best of our knowledge, APAS represents the first approximate accelerator design for RLWE-based homomorphic computations. We find that per-network hardware design achieves around $2\times$ time and $6\times$–$9\times$ area reductions, along with a similar-level of power savings compared to existing fixed NTT hardware implementations.
- **Improving Secure Inference Efficiency**: Instead of fictional computational workloads, we use the convolution layers in a well-known binary convolutional NN (CNN) [19] as benchmarks for APAS. We are able to reduce the homomorphic convolution latency by $8\times$. We also achieve 10^6 (resp. $26\times$) energy reduction compared to the CPU (resp. application-specific hardware) implementation of the best-known existing works, while improving the prediction accuracy by roughly 4%.

The rest of this paper is organized as follows. First, in Section II, we provide preliminaries on the BFV cryptosystem, RLWE-based homomorphic convolution, and SI based on NN. Second, we introduce the proposed HLT technique in Section III. Third, we present the APAS framework in Section IV. Forth, we provide detailed analysis on the target application (i.e., NN-based SI), and compare our designed hardware with NTT-based approaches in Section V. Finally, Section VI concludes this work.

II. Preliminaries

A. Notations

Throughout the paper, we use the standard notation $\mathbf{a} \in \mathbb{Z}_q^n$ to refer to a vector $\mathbf{a}$ with dimension n where elements $a_i \in \mathbf{a}$ belong to $\mathbb{Z}_q$, the set of integers modulo q. Vectors that are transformed by NTT (resp., negacyclic NTT) are noted with a hat, e.g., $\hat{\mathbf{a}} = \mathsf{NTT}(\mathbf{a})$ (resp., $\hat{\mathbf{a}} = \mathsf{NegNTT}(\mathbf{a})$). Matrices are written in capital (e.g., A), and $\lg x$ is the shorthand for $\log_2 x$.

We provide the following list of commonly used variables and their brief definitions to enhance the readability of this work. The formal definitions of the variables can be found in later sections.

- p, q: p and q are moduli used in the HE schemes, where p is the plaintext modulus, and q is the ciphertext modulus.
- n, m, $\Phi_m(x)$: n and m are sometimes used interchangeably in this work, as they both dictate the lattice dimension. We have $n = \phi(m)$, where $\phi(\cdot)$ is the Euler's totient function. $\Phi_m(x)$ is the m-th cyclotomic polynomial. We also use $\mathcal{R}_q$ to denote the quotient ring $\mathcal{R}/q\mathcal{R}$, where $\mathcal{R} = \mathbb{Z}[x]/\langle\Phi_m(x)\rangle$ is the ring of polynomials modulo $\Phi_m(x)$.
- $\mathbf{u}$, $\mathbf{w}$: The input image and NN weight matrices that are flattened as vectors. The lengths of $\mathbf{u}$ and $\mathbf{w}$ are denoted as n_u and n_w, respectively. We also make use of $\eta = \min(n_u, n_w)$. $[\mathbf{u}]$ means the encrypted version of $\mathbf{u}$.
- c_i, c_o: The input and output number of channels for a particular NN convolution layer.
- $\mathbf{e}$, ε: We use $\mathbf{e}$ to depict the RLWE errors, and ε the approximation errors.
- $\cdot$, $*$, $\circ$: When dealing with vectors, $\cdot$ is the standard inner product operator, $*$ is the convolution operator, and $\circ$ is the Hadamard product operator.

B. Packed Homomorphic Encryption

We use the private-key version of the BFV [20] implemented in the SEAL homomorphic encryption library [21] as an illustration of a typical real-world realization of RLWE-based packed additive homomorphic encryption (PAHE) scheme. A BFV instance is parameterized by the tuple (n, p, q, χ_σ), where n is the lattice dimension, p is the plaintext modulus, q is the ciphertext modulus, and χ_σ is a discrete Gaussian distribution parameterized by σ. We use U_1 to denote the uniform distribution over the set $\{-1, 0, 1\}$, and U_q for the uniform distribution over $\{0, 1, \cdots, q-1\}$. BFV makes use of a pair of one-dimensional negacyclic NTT operators (NegNTT, $\mathsf{NegINTT}$) to efficiently compute polynomial multiplications.

- **Key Generation**: First, generate the secret key vector $\mathbf{s} \leftarrow U_1^n$, where all of the n elements in $\mathbf{s}$ are sampled from the distribution U_1. The secret key is kept private to the protocol participant (e.g., the client in Section II-D) who executes the encryption function.
- **Encryption**: Upon receiving a plaintext vector $\hat{\mathbf{u}} \in \mathbb{Z}_p^n$, compute $\mathbf{u} = \mathsf{NegINTT}(\hat{\mathbf{u}}) \in \mathbb{Z}_p^n$. Uniformly sample an integer vector $\mathbf{a} \leftarrow U_q^n$ and draw a vector $\mathbf{e} \leftarrow \chi_\sigma^n$. Compute
$$\mathbf{c}_1 = \mathsf{NegINTT}(\mathsf{NegNTT}(\mathbf{a}) \circ \mathsf{NegNTT}(\mathbf{s})) + \mathbf{e} + \mathbf{u} \cdot q/p, \tag{1}$$
where $\circ$ is the Hadamard product between two vectors. Let $\mathbf{c}_0 = -\mathbf{a}$. Release the output ciphertext $[\mathbf{u}] = (\mathbf{c}_0, \mathbf{c}_1)$ to public communication channels.
- **Decryption**: Upon receiving the ciphertext $[\mathbf{u}]$ from public channels, compute
$$\mathbf{u} = \lfloor \frac{p}{q}(\mathsf{NegINTT}(\mathsf{NegNTT}(\mathbf{c}_0) \circ \mathsf{NegNTT}(\mathbf{s})) + \mathbf{c}_1) \rceil \tag{2}$$
with the private secret key $\mathbf{s}$, where $\lfloor \cdot \rceil$ is the rounding operator.
- **Homomorphic Addition**: For two ciphertexts $[\mathbf{u}] = (\mathbf{c}_{u,0}, \mathbf{c}_{u,1})$ and $[\mathbf{v}] = (\mathbf{c}_{v,0}, \mathbf{c}_{v,1})$, output $[\mathbf{u} + \mathbf{v}] = [\mathbf{u}] \boxplus [\mathbf{v}] = (\mathbf{c}_{u,0} + \mathbf{c}_{v,0}, \mathbf{c}_{u,1} + \mathbf{c}_{v,1})$.
- **Homomorphic Hadamard Product**: The $\boxdot$ operator takes as inputs a ciphertext $[\mathbf{u}] = (\mathbf{c}_{u,0}, \mathbf{c}_{u,1})$ where

Algorithm 1 The InnerProd operator applied to a ciphertext vector $[\mathbf{u}]$ and a plaintext vector $\mathbf{v}$.

```
Require: $[\mathbf{u}]$, $\mathbf{w}$
1: $[\mathbf{v}] = [\mathbf{u}] \circ \mathbf{w}$
2: for $i = n/2, n/4, \cdots, 1$ do
3:     $[\mathbf{v}] = [\mathbf{v}] \boxplus \mathsf{rot}([\mathbf{v}], i)$
4: $[y] = [\mathbf{v}](0)$
5: return $[y]$
```

Algorithm 2 The Conv operator applied to a ciphertext vector $[\mathbf{u}]$ and a plaintext vector $\mathbf{v}$ using the input-packing technique proposed in [1]. Here $\mathbf{w}(i)$ refers to the i-th vector element in $\mathbf{w}$.

```
Require: $[\mathbf{u}]$, $\mathbf{w}$
1: $[\mathbf{v}] = [\mathbf{u}] \boxdot \mathbf{w}(0)$
2: for $i = 1, 2, \cdots, n-1$ do
3:     $[\mathbf{v}] = [\mathbf{v}] + \mathsf{rot}([\mathbf{u}], i) \boxdot \mathbf{w}(i)$
4: return $[\mathbf{v}]$
```

$\mathbf{u} = \mathsf{NegINTT}(\hat{\mathbf{u}})$ and a plaintext vector $\mathbf{w}$. First, we compute $\hat{\mathbf{w}} = \mathsf{NegNTT}(\mathbf{w})$, and output

$$[\hat{\mathbf{u}} \circ \hat{\mathbf{w}}] = [\mathbf{u}] \boxdot \hat{\mathbf{w}} = (\mathsf{NegNTT}(\mathbf{c}_{u,0}) \circ \hat{\mathbf{w}}, \mathsf{NegNTT}(\mathbf{c}_{u,1}) \circ \hat{\mathbf{w}}). \quad (3)$$

- **Homomorphic Rotation**: Upon input $[\mathbf{u}] \in \mathbb{Z}_q^{2\times n}$, let $\mathbf{u} = (u_0, u_1, \cdots, u_{n-1})$, $\mathsf{rot}([\mathbf{u}], k) = [(u_k, u_{k+1}, \cdots, u_{n-1}, u_0, \cdots, u_{k-1})]$ for $k \in \{0, \cdots, n-1\}$.

C. Homomorphic Operators over BFV

The BFV cryptosystem provides native support to $\mathsf{HomOps} = (\boxplus, \boxdot, \mathsf{rot})$ in a PAHE setting, and most open-source BFV libraries [22], [23] implement the above operators as library routines. On the other hand, other operators like homomorphic convolution and inner product need to be built upon the native operators. Over the past few years, some works have discussed how to efficiently leverage the packing ability of BFV to implement general linear operations, e.g., in [1], [6]. Here, we give example constructions of the inner product and the convolution operators to illustrate the design complexity.

Suppose we have a plaintext vector $\mathbf{w} \in \mathbb{Z}_p^n$ and an encrypted vector $[\mathbf{u}]$, where $\mathbf{u} \in \mathbb{Z}_p^n$ for the lattice dimension n and plaintext modulus p. Algorithms 1–2 respectively explain how the inner product and convolution operators can be implemented by only using HomOps. We observed that, the inefficiency in both algorithms stem from the fact that when multiple plaintext values are encrypted into a single ciphertext (e.g., $[\mathbf{u}]$ encrypts n plaintext integers in $\mathbf{u}$), interactions between the plaintext values are not allowed. In other words, current methods offer no direct ways of adding two plaintext values in a single ciphertext. As a result, both algorithms adopt the rotate-and-add strategy, which is complex from two perspectives: i) extra keys need to be generated (known as the Galois keys) to perform the computationally expensive key switching operations in rot, and ii) the actual run-time of the rotate-and-add strategy vary significantly for different vector and matrix dimensions. Consequently, as further discussed in Section II-D, the computational time for linear operations dominates the total runtime in NN-based SI, and the performance profile is extremely non-linear with respect to the input dimensions [24].

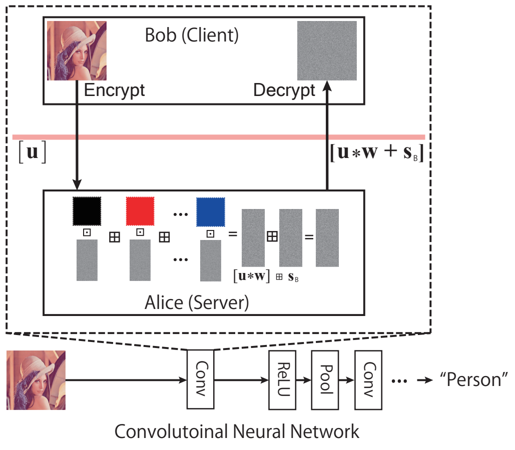

Bob (Client)
Encrypt
Decrypt
$[\mathbf{u}]$
$[\mathbf{u}*\mathbf{w} + \mathbf{s}_B]$
$[\mathbf{u}*\mathbf{w}] \boxplus \mathbf{s}_B$
Alice (Server)
Conv
ReLU
Pool
Conv
"Person"
Convolutoinal Neural Network

Fig. 1. An overview on a general CNN structure, where the convolution is implemented by the secure protocol proposed in [1].

D. Secure Inference and Related Works

While early explorations in NN-based SI generally adopts single-protocol approaches [6], [25]–[29], it was recently discovered that mix-protocol approaches [1], [24] combining HE and traditional MPC schemes achieve better accuracy and efficiency.

We provide a high-level sketch of the mix-protocol approach proposed in [1] in Fig. 1. A typical CNN has a number of linear (e.g., convolutional layers) and non-linear (e.g., ReLU activation) layers. Through the sequential evaluations of the layers, specific tasks, for example, recognizing people in images as shown in Fig. 1, can be achieved. Here, we only provide a brief summary on the protocol for homomorphic convolution layers, and more details on the complete secure inference protocol can be found in [1]. Bob as a client first flattens his two-dimensional input image $U \in \mathbb{Z}_p^{n_o \times n_i}$ into a vector $\mathbf{u} \in \mathbb{Z}_p^{n_u}$ in a raster-scan manner, where $n_u = n_i \cdot n_o$. Bob then encrypts the vector using Eq. (1), and sends $[\mathbf{u}]$ to Alice. Alice convolves the encrypted ciphertext $[\mathbf{u}]$ with a plaintext $\mathbf{w}$ according to Algorithm 2. After the homomorphic convolution, Alice homomorphically adds a random vector $\mathbf{s}_B$ to the convolution result (i.e., $[\mathbf{v}]$=$[\mathbf{u}*\mathbf{w}]$) to prevent weight leakage as $[\mathbf{v} + \mathbf{s}_B \bmod p]$ where p is the plaintext modulus. The randomized results are returned to Bob, completing one round of homomorphic convolution. We point out that the repeated computation of linear layers (e.g., convolution layers and fully-connected layers) generally amounts from 40% to as much as 90% of the total inference time in SI protocols [1], [24].

Except HE-based methods, we also see recent advances in secure inference protocols based on secret sharing (SS) [3], [16] and garbled circuits (GC) [5]. For MPC-based technique, it is reported that the online inference time can be significantly reduced. However, we point out that the total inference time in [16] remains the same as [1]. Meanwhile, purely garbled circuit (GC) based secure inference techniques, such as XONN [5], face a much more significant performance-accuracy trade-off. For example, while XONN can perform a round of secure inference within 3 seconds, the prediction accuracy is only 81.85%. The authors need to manually scale the neural network to obtain an accuracy of 86% where the inference latency inflates to 65.94 seconds [5]. As later demonstrated in Section V-B, using the proposed HLT technique, we can easily adopt properly quantized layers (that is not binary) to improve the prediction accuracy while keeping a low inference latency.

E. Hardware Implementation for RLWE-based HE

While a plethora of hardware architectures are proposed to accelerate the RLWE-based HE cryptosystems [7]–[9], [13], [30]–[32], to the best of our knowledge, most (if not all) designs integrate the NTT acceleration unit as a fundamental hardware element. The primary reason is that such platforms are general-purpose, i.e., they can accelerate any HE scheme (including fully HE) adopted in any application. Nevertheless, we point out that the performance for general-purpose HE systems is not yet practical. For example, the FHE-based secure inference protocol proposed in [6] takes 730 seconds to carry out a single inference over the CIFAR-10 dataset, while it only takes less than 50 seconds on the most recent mix-protocol approach [1], [16].

In contrast to existing works where the design efforts in the software and hardware layers are separated, we propose a software-hardware co-design approach, where the proposed HLT technique is designed to allow for an efficient hardware implementation, and the hardware accelerator is application-specific to HLT (e.g., we do not implement the NTT unit).

III. Homomorphic Linear Transformation Without NTT

As NTT-based HLT are complex and expensive to compute, a natural question to ask is, if we can avoid the rotate-and-add construction in the first place. In this section, we observe that, by using a set of modified homomorphic encryption and decryption functions, any HLT can be directly carried out without NTT and Algorithms 1–2.

A. The Ring Structure of the RLWE Ciphertext

Before delving into the actual cryptosystem construction, we first explain one of the main theoretical complexities that was hidden from the cryptosystem description in Section II-B. Instead of integer vectors, all of the operands involved in RLWE-based cryptosystems are polynomials in a quotient ring $\mathcal{R}_q$ defined by the m-th order cyclotomic polynomial $\Phi_m(x)$ for some $m \in \mathbb{Z}$. Here, $\mathcal{R}_q$ is isomorphic to $\mathbb{Z}_q^n$, where n is the degree of $\Phi_m(x)$ and is defined as $n := \phi(m)$. It is noted that this isomorphism is extremely important, as it lays the foundation for our proposed technique. Additions and multiplications between these RLWE ciphertexts then correspond to ring operations. For additions, the arithmetic procedure is the same, i.e., computing coordinate-wise summations modulo q. Meanwhile, for multiplications between two polynomials in $\mathcal{R}_q$, we need to reduce the product polynomial over $\Phi_m(x)$ and q. Depending on the exact instantiation of $\Phi_m(x)$ (i.e., the exact value of m), the reduction over $\Phi_m(x)$ varies in its algorithmic construction and computational efficiency.

When m is a power of 2, the multiplications of polynomials modulo the m-th cyclotomic polynomial $\Phi_m(x) = x^n + 1$ (where $m = 2n$) become negacyclic convolutions. Subsequently, negacyclic NTT is utilized according to the convolution theorem [33], i.e.,

$$\begin{aligned} &a(x) * b(x) \bmod x^n + 1 \\ &= \mathsf{NegINTT}(\mathsf{NegNTT}(a(x)) \circ \mathsf{NegNTT}(b(x))) \end{aligned} \tag{4}$$

for two (univariate) polynomials $a(x)$ and $b(x)$. Without NegNTT, polynomial multiplications can be directly computed using schoolbook algorithms, where the reduction modulo $x^n + 1$ results in alternating signs in the product summation [15]

$$a * b \bmod x^n + 1 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(-1)^{\lfloor (i+j)/n \rfloor} a_i b_j x^{i+j \bmod n}. \tag{5}$$

However, [15] did not discuss the case when n is not a power of two. In this work, in order to fully leverage the benefits of custom hardware, we consider quotient rings over $\Phi_m(x)$ where m is a *prime*. The prime-order cyclotomic polynomials are known to have the form $\Phi_m(x) = \sum_{i=0}^{m-1} x^i$. It is obvious that a reduction over $\sum_{i=0}^{m-1} x^i$ is different from a reduction modulo $x^n + 1$. In fact, the reduction of the product between two polynomials over prime-order Φ_m can be calculated in a similar (and actually simpler) way as

$$a * b \bmod \sum_{i=0}^{m-1} x^i = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j x^{i+j \bmod n}. \tag{6}$$

In what follows, we outline a unified RLWE cryptosystem that supports both power-of-two and prime order cyclotomic rings, avoiding the use of NTT.

B. Modified Cryptosystem

The key generation and encryption steps are mostly identical to that in Section II-B. The only difference is that we set $n = \phi(m)$ for some integer m. Therefore, $\mathbf{a}, \mathbf{e}$ and $\mathbf{s}$ will be integer vectors in $\mathbb{Z}_q^n = \mathbb{Z}_q^{\phi(m)}$. We also instantiate the m-th cyclotomic polynomial Φ_m.

- **Key Generation**: As in Section II-B, generate the secret key vector $\mathbf{s} \leftarrow U_1^n$. The secret key is kept private to the protocol participant who executes the encryption function.

- **Encryption**: Upon receiving the plaintext vector $\mathbf{u} \in \mathbb{Z}_p^n$, we directly compute
$$\mathbf{c}_1 = (\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot q/p) \bmod \Phi_m(x) \tag{7}$$
where $*$ is the convolution operator. Similar to Section II-B, we set $\mathbf{c}_0 = -\mathbf{a}$ and output the ciphertext $[\mathbf{u}] = (\mathbf{c}_0, \mathbf{c}_1)$ to public communication channels.
- **Decryption**: We use a slightly more general version of the RLWE decryption function. We assume that the input ciphertext $[\mathbf{u}]$ is of the form $[\mathbf{u}] = (C_0, \mathbf{c}_1)$, where $C_0 \in \mathbb{Z}_q^{n\times n}$ is some matrix. With secret key $\mathbf{s}$, compute
$$\mathbf{u} = \lfloor \frac{p}{q}\big((\mathbf{C}_0 \cdot \mathbf{s} + \mathbf{c}_1) \bmod \Phi_m(x)\big)\rceil \tag{8}$$
Observe that, when C_0 is the circulant matrix constructed by rotating the entries in $\mathbf{c}_0$, Eq. (8) correctly decrypts $[\mathbf{u}]$ because of the fact that cyclic (and negacyclic) convolutions are equivalent to multiplications by circulant matrices [33], [34], i.e.,
$$\mathsf{Circ}(\mathbf{c}_0) \cdot \mathbf{s} \equiv \mathbf{c}_0 * \mathbf{s} \equiv -\mathbf{a} * \mathbf{s} \bmod \Phi_m(x). \tag{9}$$
Here, Circ converts a vector into its appropriate circulant matrix form.
- **Homomorphic Addition**: The $\boxplus$ operator is exactly the same as in Section II-B.
- **Homomorphic Linear Transformation**: Upon a ciphertext $[\mathbf{u}] = (\mathbf{c}_{u,0}, \mathbf{c}_{u,1})$ where $\mathbf{u} \in \mathbb{Z}_p^n$ and some matrix $\mathfrak{W} \in \mathbb{Z}_p^{n\times n}$, we define the homomorphic linear transformation $\mathsf{HLT}_{\mathfrak{F}}$ and the homomorphic inner product operator $\boxdot$ as
$$[f(\mathbf{u})] = \mathsf{HLT}_{\mathfrak{F}}([\mathbf{u}]) = \mathfrak{W} \boxdot [\mathbf{u}] \bmod \Phi_m(x), \tag{10}$$
where $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ is some linear transformation that is essentially an automorphism over the vector space $\mathbb{Z}_p^n$. The relationship between $\mathsf{HLT}_{\mathfrak{F}}$, $\mathfrak{W}$, and $\boxdot$ is further discussed in Section III-D.

C. Security for the Modified Cryptosystem

First, it is pointed out that, in RLWE-based cryptosystems, the order of the cyclotomic polynomial (m) can have a negative security impact [35]. Fortunately, the analysis in [35] shows that, when m is a prime or a power of two, the underlying RLWE problem is no easier to solve than the standard LWE case. Therefore, while the proposed HLT technique works for any m, in this work, we only consider the power-of-two and the prime cases to ensure provable security.

The security of the modified cryptosystem can be easily deduced from the security of the original cryptosystem. Combined with the analyses in [35], we point out that the key generation and encryption steps in Eq. (1) and Eq. (7) are algebraically equivalent. Essentially, the view of the adversary on both the original and the modified cryptosystems remains the same, which is the encrypted ciphertext $(\mathbf{c}_0, \mathbf{c}_1)$ that is indistinguishable from uniformly random strings as guaranteed by the hardness of the LWE problem. Therefore, any successful attack on the modified cryptosystem will be a successful attack on the original cryptosystem. A formal proof for the following proposition is provided in Appendix A.

Proposition 1. *The cryptosystem specified in Section III-B is as secure as that specified in Section II-B against semi-honest adversaries.*

D. Main Theorem on Homomorphic Linear Transformation

As mentioned in Section III-A, RLWE ciphertexts reside in $\mathcal{R}_q$ which are isomorphic to $\mathbb{Z}_q^n$. On the other hand, the plaintext space in our modified cryptosystem is $\mathbb{Z}_p^n$, without any explicit algebraic structure. Taking the coefficient embedding defined in [36], we equivalently stated that polynomials in $\mathcal{R}_q$ form a $\mathbb{Z}_q$-module of rank n. Additionally, the module becomes an n-dimensional vector space V_q when q is a prime (i.e., $\mathbb{Z}_q$ becomes the finite field $\mathbb{F}_q$). Similarly, while we did not intentionally impose a structure on the plaintext space, the plaintext vectors form a natural $\mathbb{Z}_p$-module with rank n, and we also get a vector space V_p when p is a prime.

From the perspective of linear algebra, we know that any linear transformation $\mathfrak{F} : V_q \to V_q$ obeys the following rules.
$$\mathfrak{F}(\mathbf{u} + \mathbf{w}) = \mathfrak{F}(\mathbf{u}) + \mathfrak{F}(\mathbf{w}), \text{ and} \tag{11}$$
$$\mathfrak{F}(\mathbf{u}) = \mathfrak{W} \cdot \mathbf{u}, \tag{12}$$
where $\mathbf{u}, \mathbf{w} \in V_q$, and $\mathfrak{W} \in \mathbb{Z}_q^{n\times n}$. Note that when the linear transformation is applied to a matrix, we assume that it is applied to each column vector of the matrix. Subsequently, we have the following theorem.

Theorem 1. *Let p, q be primes. For any BFV ciphertext $[\mathbf{u}] = (\mathbf{c}_0, \mathbf{c}_1) \in \mathcal{R}_q^2$, let $\mathsf{HLT}_{\mathfrak{F}} : \mathcal{R}_q^2 \to \mathcal{R}_q^{n+1}$ be a function over the BFV ciphertext where $\mathfrak{F} : V_q \to V_q$ is any linear transformation. There exists a linear transformation $f : V_p \to V_p$ where it holds that*
$$\mathsf{Dec}\big(\mathsf{HLT}_{\mathfrak{F}}([\mathbf{u}])\big) = \mathsf{Dec}\big(\mathfrak{F}(\mathsf{Circ}(\mathbf{c}_0)), \mathfrak{F}(\mathbf{c}_1)\big) = f(\mathbf{u}) \tag{13}$$
except for a small probability δ (referred to as the decryption failure probability).

Proof. From the definition of the modified decryption function in Eq. (8), we have that
$$\begin{aligned}&\mathsf{Dec}\big(\mathfrak{F}(\mathsf{Circ}(\mathbf{c}_0)), \mathfrak{F}(\mathbf{c}_1)\big)\\ &= \lfloor \frac{p}{q}\big((\mathfrak{F}(\mathsf{Circ}(\mathbf{c}_0)) \cdot \mathbf{s} + \mathfrak{F}(\mathbf{c}_1)) \bmod \Phi_m(x)\big)\rceil.\end{aligned} \tag{14}$$
Observe that
$$\begin{aligned}\mathfrak{F}(\mathbf{c}_1) &= \mathfrak{F}(\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot q/p)\\ &= \mathfrak{F}(\mathbf{a} * \mathbf{s}) + \mathfrak{F}(\mathbf{e}) + \mathfrak{F}(\mathbf{u} \cdot q/p) \text{ from Eq. (11)}\\ &= \mathfrak{F}(\mathsf{Circ}(\mathbf{a}) \cdot \mathbf{s}) + \mathfrak{F}(\mathbf{e}) + \mathfrak{F}(\mathbf{u} \cdot q/p) \quad (15)\\ &= \mathfrak{F}(\mathsf{Circ}(\mathbf{a})) \cdot \mathbf{s} + \mathfrak{F}(\mathbf{e}) + \mathfrak{F}(\mathbf{u} \cdot q/p). \quad (16)\end{aligned}$$
Here, the derivation from Eq. (15) to Eq. (16) follows from Eq. (12) and the associativity of linear transformations, i.e.,
$$\begin{aligned}\mathfrak{F}\big(\mathsf{Circ}(\mathbf{a}) \cdot \mathbf{s}\big) &= \mathfrak{W} \cdot \big(\mathsf{Circ}(\mathbf{a}) \cdot \mathbf{s}\big)\\ &= \big(\mathfrak{W} \cdot \mathsf{Circ}(\mathbf{a})\big) \cdot \mathbf{s}\\ &= \mathfrak{F}(\mathsf{Circ}(\mathbf{a})) \cdot \mathbf{s},\end{aligned} \tag{17}$$

where $\mathfrak{W}$ is the matrix whose multiplication corresponds to the linear transformation $\mathfrak{F}$. Therefore, the decryption function in Eq. (14) becomes

$$\begin{aligned}&\lfloor\frac{p}{q}\big((\mathfrak{F}(\mathsf{Circ}(-\mathbf{a}))\cdot\mathbf{s}+\mathfrak{F}(\mathsf{Circ}(\mathbf{a}))\cdot\mathbf{s}\\&+\mathfrak{F}(\mathbf{e})+\mathfrak{F}(\mathbf{u}\cdot\frac{q}{p}))\bmod\Phi_m(x)\big)\rceil\\&=\lfloor\frac{p}{q}\big(\mathfrak{F}(\mathbf{e})+\mathfrak{F}(\mathbf{u}\cdot\frac{q}{p}))\bmod\Phi_m(x)\big)\rceil \qquad (18)\\&=\lfloor\big(\mathfrak{F}(\frac{p}{q}\cdot\mathbf{e})+\mathfrak{F}(\mathbf{u})\bmod\Phi_m(x)\big)\rceil\bmod p. \qquad (19)\end{aligned}$$

Following the same decryption analysis from standard RLWE cryptosystems (we provide a detailed decryption failure analysis in Section IV-B), Eq. (19) evaluates to $\big(\mathfrak{F}(\mathbf{u})\bmod\Phi_m(x)\bmod p\big)$ except for some small probability δ. Finally, simply take $f(\mathbf{u})=\big(\mathfrak{F}(\mathbf{u})\bmod\Phi_m(x)\bmod p\big)$, and we obtain the theorem. □

An immediate corollary that follows from Theorem 1 is as follows.

Corollary 1. *Let p,q be primes. For any BFV ciphertext $[\mathbf{u}]=(\mathbf{c}_0,\mathbf{c}_1)\in\mathcal{R}_q^2$ and any invertible matrix $\mathfrak{W}\in\mathbb{Z}_q^{n\times n}$, there exists a matrix $W\in\mathbb{Z}_p^{n\times n}$, where*

$$\mathsf{Dec}\left(\mathsf{HLT}_{\mathfrak{W}}([\mathbf{u}])\right)=\mathsf{Dec}\left(\mathfrak{W}\cdot\mathsf{Circ}(\mathbf{c}_0),\mathfrak{W}\cdot\mathbf{c}_1\right)=W\cdot\mathbf{u} \qquad (20)$$

except for a small probability δ.

Note that the $\cdot$ operator in Eq. (20) is the normal dot product defined in the vector space V_q. Since Corollary 1 follows trivially from the fact that any linear transformation over vector spaces corresponds to multiplication by some invertible matrix, we omit a formal proof.

Correctness for Decryption: From Eq. (19), it is observed that, for the decryption function to correctly round off the errors, $\mathfrak{F}(p/q\cdot\mathbf{e})\bmod\Phi_m(x)\le 1/4$ is needed. Since the size of $\mathbf{e}$ is random, the following probability is defined as the RLWE decryption failure probability.

$$\Pr[\mathfrak{F}(p/q\cdot\mathbf{e})\ge 1/4]<\delta \qquad (21)$$

where δ should be kept small. The exact value of δ can be dependent on the actual application, and we give more detailed analyses in Section IV-B.

E. Application to Homomorphic Convolution

First, we point out that, by Theorem 1, the following equation holds true

$$\mathsf{Dec}(\mathsf{HLT}_{\mathbf{w}}([\mathbf{u}]))=\mathsf{Dec}(\mathbf{w}*\mathsf{Circ}(\mathbf{c}_0),\mathbf{w}*\mathbf{c}_1)=\mathbf{w}*\mathbf{u}, \qquad (22)$$

for two vectors $\mathbf{u}\in\mathbb{Z}_p^{n_u}$ and $\mathbf{w}\in\mathbb{Z}_p^{n_w}$. Second, observe that, when the linear transformation $\mathfrak{F}$ is the convolution operator, we have the following equality

$$\mathbf{w}*\mathsf{Circ}(\mathbf{c}_0)=\mathsf{Circ}(\mathbf{w}*\mathbf{c}_0). \qquad (23)$$

Since we also know that inner products between circulant matrices and vectors become convolutions as noted in Eq. (9), we have that

$$\mathbf{w}*\mathsf{Circ}(\mathbf{c}_0)\cdot\mathbf{s}=\mathsf{Circ}(\mathbf{w}*\mathbf{c}_0)\cdot\mathbf{s}=\mathbf{w}*\mathbf{c}_0*\mathbf{s}. \qquad (24)$$

Therefore, we actually have

$$\mathsf{Dec}(\mathsf{HLT}_{\mathbf{w}}([\mathbf{u}]))=\mathsf{Dec}(\mathbf{w}*\mathbf{c}_0,\mathbf{w}*\mathbf{c}_1)=\mathbf{w}*\mathbf{u}. \qquad (25)$$

Basically, Eq. (25) is stating that a direct convolution over an RLWE ciphertext corresponds to the homomorphic convolution between $\mathbf{u}$ and $\mathbf{w}$, without the need of NTT and Algorithm 2. In what follows, we provide an alternative proof to Eq. (25) and take a closer look at why it holds true for different cyclotomic rings sketched in Section III-B.

1) Theoretical Explanation: Here, we first explain the homomorphic convolution operator when $n=\phi(m)$ is a power of two, and then provide a similar analysis when m is a prime. For the ease of presentation, all arithmetic operations in this section are performed with respect to the (polynomial) ring structure (e.g., for $a,b\in\mathcal{R}_q$, $a*b$ refers to $a*b\bmod\Phi_m(x),q$), except when explicitly stated.

Power-of-Two Case: When we use a power-of-two cyclotomic polynomial (i.e., in the form x^n+1) to instantiate our RLWE cryptosystem in Section III-B, by definition, the convolutions computed in Eq. (5) and Eq. (4) are equivalent. Then, we encrypt $\mathbf{u}\in\mathbb{Z}_p^{n_u}$ using Eq. (7) without the use of NTT as

$$[\mathbf{u}]=(-\mathbf{a},\mathbf{a}*\mathbf{s}+\mathbf{e}+\mathbf{u}\cdot q/p). \qquad (26)$$

Convolving $[\mathbf{u}]$ with a plaintext vector $\mathbf{w}\in\mathbb{Z}_p^{n_w}$ modulo the cyclotomic polynomial $\Phi_m(x)$ and some integer modulus q, we get

$$[\mathbf{u}]*\mathbf{w}=(-\mathbf{a}*\mathbf{w},\mathbf{a}*\mathbf{s}*\mathbf{w}+\mathbf{e}*\mathbf{w}+\mathbf{u}*\mathbf{w}\cdot q/p). \qquad (27)$$

Decrypting the result, we have

$$\begin{aligned}\mathsf{Dec}([\mathbf{u}]*\mathbf{w})&=\lfloor\frac{p}{q}(\mathbf{c}_0*\mathbf{w}*\mathbf{s}+\mathbf{c}_1*\mathbf{w})\rceil\\&=\lfloor\frac{p}{q}(-\mathbf{a}*\mathbf{s}*\mathbf{w}+\mathbf{a}*\mathbf{s}*\mathbf{w}+\mathbf{e}*\mathbf{w}+\mathbf{u}*\mathbf{w}\cdot\frac{q}{p}\rceil\\&=\lfloor\frac{p\cdot\mathbf{e}*\mathbf{w}}{q}+\mathbf{u}*\mathbf{w}\rceil. \qquad (28)\end{aligned}$$

When $\mathbf{e}$ is small enough in Eq. (28), the errors are rounded off, and $\mathbf{u}*\mathbf{w}$ is recovered. In other words, $[\mathbf{u}]*\mathbf{w}$ is actually a valid encryption of $[\mathbf{u}*\mathbf{w}]$, as stated in Eq. (25).

Therefore, we do not need the complex procedures in Algorithm 2 to homomorphically convolve two vectors. The plaintext vector can be directly convolved with the ciphertext vector, and we obtain the convolution result after decryption. We emphasize that the above procedure only remains correct when the following condition is met

$$\begin{aligned}&\mathbf{u}*\mathbf{w}=\mathbf{u}*\mathbf{w}\bmod\Phi_m,\text{ or equivalently}\\&n_u+n_w-1<\phi(m)=n \qquad (29)\end{aligned}$$

where the length of $\mathbf{u}$ is n_u, and that of $\mathbf{w}$ is n_w. We also stress that the convolution on the left hand side of

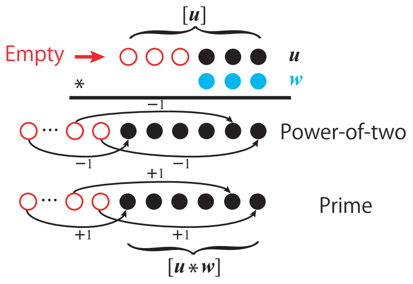


Fig. 2. Conceptual illustration of schoolbook based homomorphic convolution with prime and power-of-two cyclotomic polynomial reduction, where black dots are the coefficients in the ciphertext vector $[\mathbf{u}]$ that contains the actual values from $\mathbf{u}$, red circles are those that only contains errors, and blue dots are the plaintext weight vector $\mathbf{w}$.

Eq. (29) is *not* polynomial multiplication in the ring $\mathcal{R}_q$ (it is a normal convolution). As also listed in Eq. (29), an equivalent correctness condition is that the total length of the convolved result is less than $\phi(m) = n$. Under such condition, the convolution formula defined in Eq. (5) evaluates to

$$\mathbf{u} * \mathbf{w} \bmod x^n + 1 = \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_w-1} u_i w_j x^{i+j \bmod n} \tag{30}$$

where the negacyclic wrapping never occurs ($i + j$ is always less than n).

Prime Order Case: When the ring of choice is based on prime order cyclotomic polynomials, since we do not have the fast NTT-based Hadamard product operator, existing works offer no efficient methods of evaluating homomorphic convolutions between the encrypted vector $[\mathbf{u}]$ and a plaintext vector $\mathbf{w}$. However, the exact algorithm in the power-of-two example in Eq. (25) can also be performed over prime-order cyclotomic rings, with the same correctness condition specified in Eq. (29) (here, we are trying to avoid the cyclic wrap-around, instead of the negacyclic one when m is a power of two). Note that for m a prime, $\phi(m) = m - 1$. Being able to choose m as a prime integer significantly improve the practicality of RLWE-based HE, as the efficiency of HLT strongly depends on the size of the ciphertext polynomial.

2) Illustrative and Equivalent Matrix Demonstration: To better explain the proposed approaches for homomorphic convolution, we first sketch a toy example in Fig. 2. Here, the ciphertext $[\mathbf{u}]$ with a dimension of 6 contains a plaintext input $\mathbf{u}$ of length 3, where the upper three slots (red circles in Fig. 2) remains empty. The weight vector $\mathbf{w}$ also has 3 elements. We can see that when the length of the convolved result $\mathbf{u}*\mathbf{w}$ (5 in this case) is less than or equal to the length of the ciphertext (6 in the example), a ciphertext polynomial multiplied by a plaintext polynomial is equivalent to performing an internal convolution over the encrypted plaintext filling out the empty ciphertext slots. In terms of the ciphertext polynomial, the result has a total length of 8, where the higher-order 2 terms (containing only errors) will be wrapped around through reductions over Φ_m according to either Eq. (5) or (6).

Following the illustrative example, we now provide an equivalent derivation using the notations in Theorem 1 and Corollary 1 for Fig. 2. As mentioned, a circular convolution between $\mathbf{u}$ and $\mathbf{w}$ corresponds to the multiplication of $\mathbf{u}$ by a circulant matrix constructed from $\mathbf{w}$. Denote $\mathbf{d} = \mathbf{a} * \mathbf{s} + \mathbf{e}$, recall that $\mathbf{c}_1 = \mathbf{d} + \frac{q}{p}\mathbf{u}$. Let $d_i \in \mathbf{d}$, $u_i \in \mathbf{u}$, and $w_i \in \mathbf{w}$, we have

$$\begin{aligned}
&\mathfrak{W} \cdot \mathbf{c}_1 \\
&= \begin{bmatrix} w_0 & w_1 & w_2 & 0 & 0 & 0 \\ 0 & w_0 & w_1 & w_2 & 0 & 0 \\ 0 & 0 & w_0 & w_1 & w_2 & 0 \\ & 0 & 0 & w_0 & w_1 & w_2 \\ \pm w_2 & 0 & 0 & 0 & w_0 & w_1 \\ \pm w_1 & \pm w_2 & 0 & 0 & 0 & w_0 \end{bmatrix} \cdot \begin{bmatrix} d_0 + \frac{q}{p}u_0 \\ d_1 + \frac{q}{p}u_1 \\ d_2 + \frac{q}{p}u_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix}^T \qquad (31) \\
&= \mathbf{w} * (\mathbf{d} + \frac{q}{p}\mathbf{u}) \equiv \mathbf{w} * \mathbf{c}_1 \bmod \Phi_m(x) \qquad (32)
\end{aligned}$$

where the $\pm$ signs in $\mathfrak{W}$ stem from the cyclotomic reduction. The $\pm$ signs are positive when we have a prime-order cyclotomic reduction (i.e., m is prime), and negative when m is a power of two. By Corollary 1, Eq. (31) evaluates to $\mathbf{u} * \mathbf{w}$ after decryption.

Using terms defined in Theorem 1, we showed that when the plaintext linear transformation f is taken to be the cyclic convolution, we can construct a matrix $\mathfrak{W}$, where the multiplication by $\mathfrak{W}$ corresponds to the ciphertext transformation $\mathfrak{F}([\mathbf{u}]) \bmod \Phi_m(x) \bmod p$. Hence, in order to produce meaningful arithmetic operations over the ciphertexts that live in the designated cyclotomic rings, $\mathfrak{W}$ needs to be carefully constructed, with additional constraints such as Eq. (29) and the $\pm$ signs in Eq. (31).

In summary, we showed that homomorphic convolutions, a special case of the linear transformation in Theorem 1, can be carried out in an efficient manner without the use of NTT operators and Algorithm 2. In a practical sense, when a ciphertext of $[\mathbf{u}] \in \mathbb{Z}_q^{2\times n}$ is convolved with $\mathbf{w}$, according to Eq. (5), the operation has an overall complexity of $\mathcal{O}(n \cdot n_w) < \mathcal{O}(3 \cdot n \cdot \lg n)$ when $n_w \ll n$. In other words, our proposed homomorphic convolution has a time complexity that is less than the rotate-and-add approach in Algorithm 2 when n_w is small, as in the case of CNN filters.

In what follows, we instantiate the linear transformation technique under a packed homomorphic convolution setting in Section III-F, and a homomorphic inner product setting in Section III-G.

F. Packed Homomorphic Convolution

Since n needs to be quite large to ensure RLWE security, most existing works adopt the Smart-Vercauteren (SV) packing technique [37] to maximize the plaintext slot utilization of the RLWE ciphertext [1]. One of the main practical issue with the direct homomorphic convolution devised in Section III-E is that, since we do not apply the NTT-based SV packing technique, when both the input plaintext and the filter are extremely short (i.e., $n_u + n_w \ll \phi(m)$), we waste a large number of the $\phi(m)$-dimensional ciphertext space.

Through the use of Theorem 1, we show a provably correct construction to perform packed homomorphic convolution over a single ciphertext. We observe that, since the operator can be any linear transformation expressed through an invertible matrix, we can change the construction of $\mathfrak{W}$ to permit separate convolution on different region of the ciphertext. As an example, we wish to convolve two sets of inputs $\mathbf{u}_i, \mathbf{u}_j \in \mathbb{Z}_p^2$ (without loss of generality, we use a smaller example to save space) encrypted as a single ciphertext $\mathbf{c}_1 = \mathbf{a} * \mathbf{s} + \mathbf{e} + \frac{q}{p}(\mathbf{u}_i || 0 || \mathbf{u}_j || 0)$, where $||$ is the concatenation operator. Suppose also that we have filters $\mathbf{w}_i, \mathbf{w}_j \in \mathbb{Z}_p^2$. In this case, n needs to satisfy $n \geq 6$, since the two convolutions each can have a maximum length of 3 (convolution between two 2-dimensional vectors). To compute $\mathbf{u}_i * \mathbf{w}_i$ and $\mathbf{u}_j * \mathbf{w}_j$, consider the following matrix multiplication

$$\mathfrak{W} \cdot \mathbf{c}_1 = \begin{bmatrix} \mathfrak{W}_i & \mathbf{0} \\ \mathbf{0} & \mathfrak{W}_j \end{bmatrix} \cdot \begin{bmatrix} d_0 + \frac{q}{p} u_{i,0} \\ d_1 + \frac{q}{p} u_{i,1} \\ d_2 \\ d_3 + \frac{q}{p} u_{j,0} \\ d_4 + \frac{q}{p} u_{j,1} \\ d_5 \end{bmatrix}^T \tag{33}$$

$$= \begin{bmatrix} \mathbf{w}_i * (\mathbf{d}_i + \frac{q}{p} \mathbf{u}_i) & \mathbf{w}_j * (\mathbf{d}_j + \frac{q}{p} \mathbf{u}_j) \end{bmatrix} \tag{34}$$

where

$$\mathfrak{W}_i = \begin{bmatrix} w_{i,0} & w_{i,1} & 0 \\ 0 & w_{i,0} & w_{i,1} \\ \pm w_{i,1} & 0 & w_{i,0} \end{bmatrix}, \mathfrak{W}_j = \begin{bmatrix} w_{j,0} & w_{j,1} & 0 \\ 0 & w_{j,0} & w_{j,1} \\ \pm w_{j,1} & 0 & w_{j,0} \end{bmatrix},$$

$\mathbf{0}$ are 3×3 zero matrices, $\mathbf{d}_i = [d_0 \; d_1 \; d_2]$ and $\mathbf{d}_j = [d_3 \; d_4 \; d_5]$. Essentially, we are conducting two separate convolutions over two separate regions of the same ciphertext. The main observation here is that, the separate convolutions in Eq. (34) is equal to Eq. (33). Consequently, Corollary 1 guarantees that, as long as $\mathfrak{W}$ is an invertible matrix, Eq. (34) decrypts to the vector $[\mathbf{u}_i * \mathbf{w}_i \; \mathbf{u}_j * \mathbf{w}_j]$, which contains the two separate convolution results. Since $\mathfrak{W}$ is diagonal, it is invertible as long as both $\mathfrak{W}_i$ and $\mathfrak{W}_j$ are invertible. This condition can be easily satisfied as both $\mathbf{w}_i, \mathbf{w}_j$ are assumed to be non-zero (convolutions with all-zero vectors are generally meaningless).

As it turns out, for an n-dimensional ciphertext, the above technique allows us to embed $\lfloor n/(n_u + n_w - 1) \rfloor$ plaintext inputs into a single ciphertext.

G. Application to Inner Product

Theorem 1 actually allows for arbitrary linear transformation to be carried out on RLWE ciphertexts, without the rotate-and-add approach in Algorithm 1. For brevity, we only provide an abstract analysis to inner products as a demonstration of our idea.

Here, let $\mathbf{u}$ be a n_u-dimensional vector in $\mathbb{Z}_p^{n_u}$ encrypted as $(\mathbf{c}_0, \mathbf{c}_1) = (-\mathbf{a}, \mathbf{a} * \mathbf{s} + \mathbf{e} + \frac{q}{p}\mathbf{u})$ (where $\mathbf{u}$ is properly zero-padded) as in Section III-F, and W a $n_u \times n_w$ matrix in $\mathbb{Z}_p^{n_u \times n_w}$. We can construct an $(n \times n)$-dimensional matrix $\mathfrak{W}$ as

$$\mathfrak{W} \cdot \mathbf{c}_1 \tag{35}$$

$$= \begin{bmatrix} W & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \cdot \begin{bmatrix} d_0 + \frac{q}{p} u_0 \\ \vdots \\ d_{n_u - 1} + u_{n_u - 1} \\ d_{n_u} \\ \vdots \\ d_{n-1} \end{bmatrix}^T = \begin{bmatrix} W \cdot (\mathbf{d}_{n_u} + \mathbf{u}) \\ \mathbf{d}_{n - n_u} \end{bmatrix} \tag{36}$$

where we define $\mathbf{d}_{n_u}$ to be $[d_0 \; \cdots \; d_{n_u - 1}]$, $\mathbf{d}_{n - n_u}$ as $[d_{n_u} \; \cdots \; d_{n-1}]$, and I is the identity matrix with dimension $((n - n_u) \times (n - n_w))$. After decryption, Corollary 1 guarantees that the output is $W \cdot \mathbf{u}$, which is precisely the inner product between $\mathbf{u}$ and W. We note that the proposed technique is particularly useful when W is a sparse matrix (e.g., in the case of sparse circulant matrices for short convolutions).

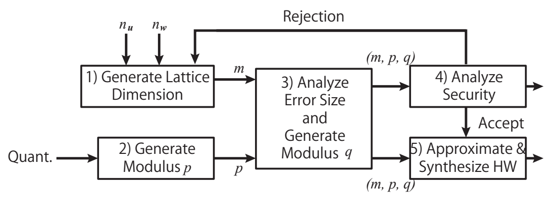

Rejection
n_u
n_w
1) Generate Lattice Dimension
m
3) Analyze Error Size and Generate Modulus q
(m, p, q)
4) Analyze Security
Accept
Quant.
2) Generate Modulus p
p
5) Approximate & Synthesize HW
(m, p, q)

Fig. 3. The APAS framework that automatically generates RLWE parameters and the corresponding approximate hardware architectures.

IV. Application-Specific Hardware for Homomorphic Convolution

The theoretical results in Section III standout when application-specific hardware is adopted. For CPUs, basic arithmetic operations run roughly in the same amount of time and energy, with or without the use of NTT. Conversely in this section, we show that, without NTT, we can significantly reduce the complexity of the underlying hardware. We propose APAS, a hardware-software co-optimization framework to automatically generate application-specific hardware architectures for the corresponding linear transformation. We use homomorphic convolution (based on the proposed technique) as a demonstration, since this operation dominates the runtime of NN-based SI schemes [1]. However, our framework applies generally to all types of HLTs.

A. Overview of APAS

The central question to APAS is how hardware architectures can be synthesized to maximize the efficiency of the homomorphic convolution computations involved in SI. As Fig. 3 shows, the framework utilizes five subcomponents in achieving this goal. We first sketch a general flow of the framework, and then discuss some of the important steps. Since APAS is designed for instantiating hardware accelerators used in homomorphic convolution, the inputs to the framework are the discrete sequences to be convolved. As before, we assume that the inputs to APAS are two one-dimensional vectors $\mathbf{u} \in \mathbb{Z}_{p_u}^{n_u}$ and $\mathbf{w} \in \mathbb{Z}_{p_w}^{n_w}$.

1) **Dimension Generation**: Under the condition in Eq. (29), we set $n = \min(\mathsf{NextPow2}(n_u + n_w - 1), \mathsf{NextPrime}(n_u + n_w - 1))$. Basically, $\mathsf{NextPow2}$ finds the smallest power-of-two that is larger than $n_u + n_w - 1$, and $\mathsf{NextPrime}$ the smallest prime satisfying the same requirement. If the next power-of-two is smaller than the next prime, we take $n = \mathsf{NextPow2}(n_u + n_w - 1)$, and $m = 2n$. Otherwise, we have $n = \mathsf{NextPrime}(n_u+n_w-1)$, and $n = m-1$ in this case. In particular, we observe that many images have dimensions that are of a power of two (e.g., the CIFAR-10 dataset is of dimension 32×32), which makes the convolved results slightly longer than a power-of-two vector. In such case, we can adopt prime-order cyclotomic polynomials in instantiating the RLWE cryptosystem.

2) **Plaintext Modulus Generation**: Generating the plaintext modulus p is easy, as p only needs to be large enough to contain the convolved results. In other words, we need to ensure that $p \geq u * w$, where $u * w$ is computed according to Eq. (30). For a multi-channel NN convolution with c_i input channels, this translates to a correctness requirement of

$$p \geq c_i \cdot \max(u) \cdot \max(w) \cdot \eta. \tag{37}$$

where $\eta = \min(n_u, n_w)$. Note we do not require $p \equiv 1 \bmod n$ for n a power of two as in most existing NTT-based approaches [1], [38]. However, p does need to be a prime to ensure that $\mathbb{Z}_p$ is isomorphic to the prime-order finite field $\mathbb{F}_p$.

3) **Error Analysis and Ciphertext Modulus Generation**: Analyzing the error accumulation behavior with a fixed dimension m and plaintext modulus p is one of the central contributions of APAS, and we will provide further details in Section IV-B. Roughly speaking, we estimate the size of $\mathbf{e} * \mathbf{w}$ in Eq. (28), and generate the ciphertext modulus q to ensure correct decryption. Note that similar to p, q also does not need to split over n.

4) **Security Analysis**: In this work, we use the framework given in [39] to provide the equivalent LWE bit security. When the security requirement is not fulfilled, we reject the parameter sets and re-generate m and q. This process is repeated until parameters that satisfy both error and security constraints are generated.

5) **Hardware Synthesis**: Given q, n_w and the error analysis, we can instantiate approximate hardware units to maximize the energy efficiency of APAS. Note that the architecture is dynamic, i.e., for different q and n_w, we can have drastically different architectures. We give concrete example architectures in Section IV-C.

B. Error Analysis

Following the analysis in Section III-E and [40], for an approximate evaluation of the homomorphic convolution in Eq. (27), the error behavior can be formulated as

$$\begin{aligned}&(\mathsf{Approx}(-\mathbf{a}) * \mathbf{w}, \mathsf{Approx}(\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot q/p) * \mathbf{w}) \\ &= ((-\mathbf{a} + \boldsymbol{\varepsilon}_0) * \mathbf{w}, (\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot q/p + \boldsymbol{\varepsilon}_1) * \mathbf{w}).\end{aligned} \tag{38}$$

for some approximation error vectors $\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1$. During decryption, we compute

$$\begin{aligned}&\lfloor \frac{p}{q}(-\mathbf{a} * \mathbf{s} * \mathbf{w} + \boldsymbol{\varepsilon}_0 * \mathbf{s} + (\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot \frac{q}{p} + \boldsymbol{\varepsilon}_1) * \mathbf{w}) \rceil \\ &= \lfloor \frac{p}{q}(\boldsymbol{\varepsilon}_0 * \mathbf{s} + \boldsymbol{\varepsilon}_1 + \mathbf{e}) * \mathbf{w} + \mathbf{u} * \mathbf{w} \rceil.\end{aligned} \tag{39}$$

The decryption remains correct as long as

$$\Pr[|\frac{p}{q}(\boldsymbol{\varepsilon}_0 * \mathbf{s} + \boldsymbol{\varepsilon}_1 + \mathbf{e}) * \mathbf{w})| \geq 1/4] < \delta. \tag{40}$$

Here, we assert that Eq. (40) is a soft bound, in the sense that $\delta > 0$ can be arbitrary depending on the application. For example, in a typical CNN-based inference scheme, prediction accuracy ranges from mid 80% to 99%. As a result, even if 0.1% of the ciphertext cannot be correctly decrypted, the resulting accuracy degradation is hardly noticeable as demonstrated in Section V-B. Hence, in this work, we set $\delta = 10^{-3}$.

Let $n = \phi(m)$ as mentioned. Substituting a and b in Eq. (5) with $\mathbf{e}$ and $\mathbf{w}$, we see that the l-th coefficient θ_l for x^l after the convolution and reduction over a power-of-two cyclotomic polynomial can be written as

$$\theta_l = \sum_{i=0}^{n-1} (-1)^{(n-l-1+i)/n} w_i e_{(n-i+l) \bmod n}, \tag{41}$$

and we can simply remove the alternating -1 term in case of prime-order cyclotomic polynomial. Since the alternating sign (in the case of power-of-two Φ) becomes uniform once we take a sum (e.g., in the decryption process) over l, we only need to study the norm bound for the product distribution $\sum_{i=0}^{n-1} w_i \cdot e_{(n-i+l) \bmod n}$, which is basically the inner product between $\mathbf{w}$ and a permuted version of $\mathbf{e}$. Notice that the summation in Eq. (41) is actually taken over $\eta = \min(n, n_w)$. In CNN, it is generally the case that n_w (the length of the filter) is much smaller than the lattice dimension n. Therefore, the size of θ_l can also be kept small.

We adopt the powerful Chernoff-Cramér inequality to derive an upper bound on the product distributions for all terms in Eq. (40). The reason that simpler bounds, such as Lemma 2.2 in [41] used in [38], [40], cannot be easily borrowed is that many of the operands involved do not follow well-defined distributions (e.g., a discrete Gaussian distribution). In the experiment, we show that Chernoff-Cramér gives us bounds that are sharp enough for practical purposes.

Concretely, the Chernoff-Cramér inequality is stated as follows.

Theorem 2. *(Theorem D.1 in [42]) Let ψ be a distribution over $\mathbb{R}$ and let $X_0, \cdots, X_{\eta-1}$ be independent and identically* *distributed variables drawn from ψ, with mean μ. Then, for any t such that $M_{\psi}(t) < \infty$, it holds that*

$$\Pr\left[\sum_{i=0}^{\eta-1} X_i \geq \eta\mu + \beta\right] \leq \exp(-\beta t + \eta \ln(M_{\psi}(t))), \quad (42)$$

where $M_{\psi}(t)$ is the moment generating function defined as

$$M_{\psi}(t) := \mathbb{E}[\exp(t(\psi - \mathbb{E}[\psi]))] \quad (43)$$

Since Theorem 2 holds for any bounded distribution, we can apply the theorem to the random variable $X_{we} = w \cdot e$ or $X_{w\boldsymbol{\varepsilon}} = w \cdot \varepsilon$. Here, we define w and ε to be random variables coming from the weight and approximate error distributions, respectively. Therefore, the summation of X over η is precisely the inner product $\mathbf{w} \cdot \boldsymbol{\varepsilon}$. Finally, we note that the L_2 bound on the convolved result $\boldsymbol{\varepsilon}_0 * \mathbf{w} * \mathbf{s}$ follows a similar derivation, where we can apply the Chernoff-Cramér inequality twice, first on $\boldsymbol{\varepsilon}_0 * \mathbf{w}$ and then on $(\boldsymbol{\varepsilon}_0 * \mathbf{w}) * \mathbf{s}$.

While existing studies have proposed methods for bounding intrinsic RLWE [42] and approximation errors [40], errors generated by homomorphic evaluation of arbitrary linear transformation cannot be bounded using some presumed distribution without a specific target application. In NN-based SI schemes, the weight quantizer (denoted as p_w) is generally much smaller than p (e.g., $p_w = 2$ in binary networks [5], [19]) because p needs to be large enough to contain the convolved results between $\mathbf{u}$ and $\mathbf{w}$ (and therefore $p > p_w$ strictly). We provide a more concrete example and asymptotic analysis in Section V-C. Nevertheless, for our application, we can safely assume that the weights are distributed according to some sub-Gaussian distribution over the range of its valuation. We note that existing NTT-based approaches cannot benefit from this small-operand properties of the CNN filters. As noted in [1], after NTT, even binary operands become uniformly distributed over the entire plaintext space, i.e., p.

In the experiments, we provide a concrete example and the resulting theoretical bounds on the size of the errors. In addition, we also simulate the actual product error distribution and decryption failure probability (i.e., the left hand side in Eq. (40)) to show that a simulation-based approach acquires (slightly) tighter error bounds.

C. Approximate Hardware Architecture

The main difficulty in designing the hardware accelerator in APAS is that, the hardware architectures vary significantly from network to network. For example, in recent works [16], [24], [43], neural architecture search (NAS) is conducted in searching the best NN for SI. The resulting network architecture has highly non-uniform parameterizations, where both the filter sizes and quantization factors vary from layer to layer. In this section, we adopt a network-specific approach in the design and implementation of approximate hardware for NN-based SI.

The approximate hardware architecture used in APAS is sketched in Fig. 4. The approximate multiplier shares similarities with existing designs in [40], where the $\lg q$-bit input

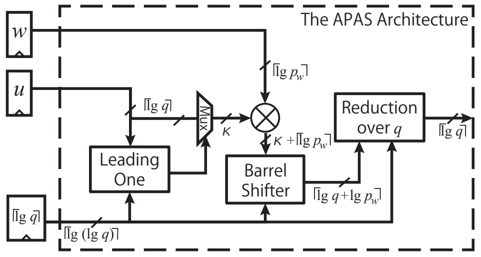


Fig. 4. A high-level abstraction of the approximate hardware architecture adopted in APAS.

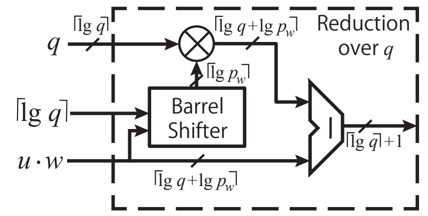


Fig. 5. The modified Barrett reduction unit for a fixed bit-width modulus q. It is noted that for the subtractor, the upper operand is the subtrahend, and lower operand represents the minuend.

$u \in \mathbf{u}$ is approximated by a κ-bit integer. Here, $\kappa = \lceil \lg q \rceil - k$ and k is the approximation factor. The leading one detection unit feeds the position of the first one in the input to a Mux, where the Mux selects the κ-bit integer following the leading one from the input number. The approximated number is multiplied by a $\lceil \lg p_w \rceil$-bit input $w \in \mathbf{w}$. First, we note that $\lceil \lg p_w \rceil$ is assumed to be small relative to the RLWE plaintext modulus $\lceil \lg p \rceil$ where $\lceil \lg p \rceil = \lceil \lg (p_u \cdot p_w) \rceil$ (e.g., $\lceil \lg p_w \rceil$ can be as small as two bits in binary neural networks, while $\lceil \lg p \rceil = 13$ bits to contain the convolved results). Therefore, in extreme cases, we do not need a full-fledged multiplier, but only one or two κ-bit approximate adders. Next, the most important observation we make is that, $\lceil \lg q \rceil$ is dynamic from layer to layer. When the current $\lceil \lg q \rceil$ is drastically smaller than the maximum $\lceil \lg q \rceil$ in the entire NN, we can save powers by gate clocking the registers for the upper bits in all operands.

1) Modified Barrett Reduction Unit: One of the main differences between [40] and Fig. 4 is the need to reduce the product over q, since q is generally not selected to be a power of two in RLWE based cryptosystems. In addition, we observe that q, as well as $\lg p_w$, are not fixed in APAS. Therefore, the proposed modular reduction unit, shown in Fig. 5, adopts the Barrett reduction algorithm [44] with a dynamic modulus construction. For some input product $u \cdot w$, the reduction is carried out by calculating

$$y = (u \cdot w) - q \cdot \big(((u \cdot w) \cdot \gamma) \gg \ell\big) \quad (44)$$

where $\gamma = \lfloor 2^{\ell}/q \rfloor$, and $u \cdot w$ is the product to be reduced. Benefiting from the fully custom hardware, we can set $\ell = \lceil \lg q \rceil$. We omit a formal proof, but only point out that, in the original Barrett reduction algorithm, the reduced result is in the range of $[0, 2q]$. However, in our modified architecture, we

TABLE I
NEURAL ARCHITECTURES USED FOR PARAMETER INSTANTIATION

Layer Type	Filter Dimension	Quantization
CR	$(3 \times 192 \times 5 \times 5)$	(8, 4)
BCR	$(192 \times 160 \times 1 \times 1)$	(1, 1)
BCR	$(160 \times 96 \times 1 \times 1)$	(1, 1)
PL	(3×3)	
BCR	$(96 \times 192 \times 5 \times 5)$	(1, 1)
BCR	$(192 \times 192 \times 1 \times 1)$	(1, 1)
BCR	$(192 \times 192 \times 1 \times 1)$	(1, 1)
PL	(3×3)	
BCR	$(192 \times 192 \times 3 \times 3)$	(1, 1)
BCR	$(192 \times 192 \times 1 \times 1)$	(1, 1)
CR	$(192 \times 10 \times 1 \times 1)$	(8, 4)
PL	(8×8)	
Accuracy on CIFAR-10: $85.9 \pm 0.1\%$		

always know that

$$\big(q \cdot ((u \cdot w) \gg \ell)\big) \leq u \cdot w \leq \big(2q \cdot ((u \cdot w) \gg \ell)\big). \quad (45)$$

Consequently, y will always be in $\mathbb{Z}_q$, and we do not need further range adjustments as in the original Barrett reduction algorithm.

V. EXPERIMENTS AND COMPARISONS

A. Section Layout and Experimental Setup

In this section, we first perform accuracy and weight distribution studies on a NN architecture in Section V-B. Through these experiments, we are able to derive concrete error bounds for the homomorphic evaluations, and subsequently instantiate parameters for APAS in Section V-C. Finally, we synthesize the instantiated hardware accelerators in APAS, and compare our performance with the-state-of-the-art homomorphic convolution efficiencies in Section V-D.

In the experiment, we use the binary NN (BNN) proposed in [19] to demonstrate the effectiveness of APAS. The adopted neural architectures are listed in Table I, where CR indicates a convolution with a point-wise ReLU layer, BCR a binarized CR, and PL the average pooling layer. Quantization shows the input and weight quantization factors, respectively, and dimensions are sketched in the order of input channels, filter channels, filter width and filter height. Compared to manually-crafted NN, BNN exhibits higher levels of layer-to-layer parameter variations. For example, the quantization varies from 2-bit to 12-bit (8-bit inputs with 4-bit weights). Hence, the application-specific dynamic architectures proposed in Section IV-C become especially useful.

The approximate hardware architectures for the respective parameter sets are synthesized on a 65 nm low-power process node using Synopsis Design Compiler [45] and a Xilinx Virtex-7 (XC7V585T-3) FPGA device.

B. Neural Architecture and Weight Distribution

In this section, we first examine the accuracy performance and weight distributions in the trained NN using the BNN architecture depicted in Table I. The prediction accuracy is

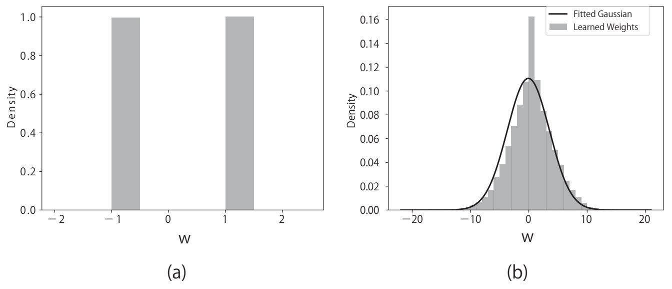


Fig. 6. Distributions of weights in (a) the third BCR layer with binary weights and (b) the first CR layer with 4-bit fixed-point weights in the trained BNN.

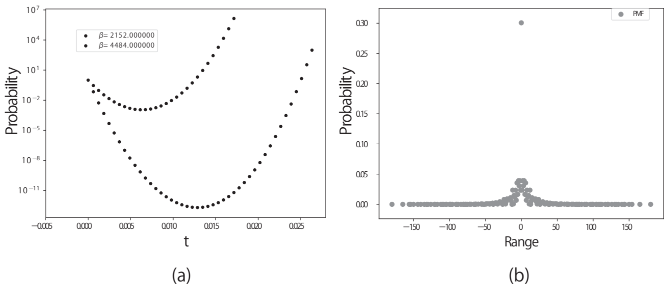


Fig. 7. (a) The probability mass function defined for the product distribution $w \cdot e$. (b) The Chernoff-Cramér bounds. At the convex point, we have the minimum β that guarantee $\sum_{i=0}^{n-1} X \leq n\mu + \beta$, except for the corresponding small probability.

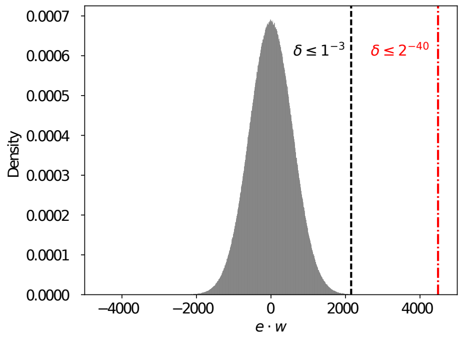


Fig. 8. The simulated product distribution of $c_{\text{in}} \cdot \mathbf{w} * \mathbf{e}$ where the convolution is carried out using the proposed homomorphic convolution technique.

TABLE II
PROPOSED PARAMETER SETS

Parameter	m	p	$\lg q$	k	Security
BNN_{min}	661	12289	27-bit	4	128-bit
BNN_{mid}	1024	12289	27-bit	4	129-bit
BNN_{max}	1049	1228943	36-bit	5	167-bit
DArL [38]	2048	12289	55-bit	-	$\geq$ 157-bit
Gazelle [1]	2048	5324801	60-bit	-	128-bit

86.1%, which is recorded on the CIFAR-10 dataset using the PyTorch framework [19], [46].

To study the impact of decryption failure probability on the NN prediction accuracy, we randomly select 0.1% of the convolved results in each and every convolution layers

TABLE III
SYNTHESIZED HARDWARE ARCHITECTURE ON ASIC AND FPGA PLATFORMS

65-nm Low Power Technology Node			
Architecture	Delay (ns)	Area (NAND)	Power (μW)
BNN$_{\max}$	3.4	4530	243
Exact 36-bit	6.4	15515	327
DArL (55-bit NTT)	7.1	31260	580
Gazelle (60-bit NTT)	7.3	40061	782

Xilinx Virtex-7 (XC7V585T-3)			
Architecture	Frequency (MHz)	Area (LUT/Reg/Slice)	Power (mW)
BNN$_{\max}$	222	390/147/146	343
Exact 36-bit	240	1051/103/344	377
DArL (55-bit NTT)	245	1932/135/592	464
Gazelle (60-bit NTT)	235	5712/384/1827	689

to fail the decryption in the forward propagation. As shown in Table I, the prediction accuracy from an average of ten inference runs (with different random seeds) is $85.9 \pm 0.1\%$, which is only slightly lower than the accuracy of a normal inference (86.1%) without decryption failure. However, such accuracy degradation is generally considered marginal in most NN-based inference schemes, as the normal inference accuracy can also vary by a similar amount because of the initial weight randomness and the stochastic optimization process. Furthermore, we note that, even with decryption failures, our network is still more accurate than the neural architectures adopted in many existing works [1], [5] (e.g., Gazelle only has an accuracy of 81.6% on the CIFAR-10 dataset).

Next, we investigate the distribution of weights in the trained NN. In Table I, we observe that there are four basic types of convolution layers: 12-bit quantized 5×5, binary 5×5, binary 1×1 and binary 3×3 with varying filter channels. The distribution of weights in the first CR layer and third BCR layer in Table I are shown in Fig. 6(a) and (b), respectively. Immediately, we see that, in the BCR layer, the weight distribution approaches the Bernoulli distribution with parameter $1/2$ extremely well. In addition, when the range of valuation is enlarged to 4-bit (from -15 to $+15$ in Fig. 6(b)), we can see that the overall weight distribution approximates a Gaussian, which means that large weight values (e.g., 15) are much less likely to appear than smaller ones (e.g., 1). As later described in the error estimation step, the centered weight distribution and the short filter size generate a very small amount of errors. As a result, we end up with much smaller ciphertext sizes compared to existing works.

C. Parameter Instantiation

To illustrate how parameters can be instantiated (and the derived bounds on evaluation errors), we follow the five steps prescribed in the APAS framework (in Fig. 3) for the first CR layer in the BNN architecture as shown in Table I.

Step 1) We generate the lattice dimension. Since the input dimension is $1,024 = 32 \times 32$, we can choose the smallest dimension that can contain the convolved results while guaranteeing the security of the homomorphic encryption scheme. Here, m is taken to be 1,049.

Step 2) We generate the plaintext modulus given n_w and n_u. Since the inputs are 9-bit and weights are 5-bit with sign, according to Eq. (37), p can be as small as $1,228,800 = 3 \cdot 2^9 \cdot 2^5 \cdot 25$, which is the maximum attainable value during the calculation of a convolution between a c_i-channel signed 8-bit input vector with a signed 4-bit filter of length $n_w = 25$. Note that p needs to be a prime to ensure that $\mathbb{Z}_p$ is the field $\mathbb{F}_p$, so we take p to be a 21-bit prime $1,228,943$ that is the smallest prime larger than $1,228,800$.

Step 3) With p and m, we analyze the size of the errors generated during homomorphic evaluation using Theorem 2. As proposed in Section IV-B, we can precisely define the PMF of this evaluation error when the weight distribution is known. As a demonstration, we plot the PMF of $\mathbf{e} * \mathbf{w}$ in Fig. 7(b), where $\mathbf{e}$ is a vector of the discrete Gaussian errors from the RLWE cryptosystem, and $\mathbf{w}$ the NN weights. Using the PMF sketched in Fig. 7(b), we can obtain the precise probability of $\Pr\left[\sum_{i=0}^{m-1} X_i \geq m\mu + \beta\right]$ in Eq. (42) for any given β. The minimum β's that guarantee probability below 10^{-3} and 2^{-40} are presented in Fig. 7(a). When $\beta = 2,152$, we have $\Pr\left[\sum_{i=0}^{\eta-1} X_i \geq \eta\mu + \beta\right] \leq 10^{-3}$. By setting q accordingly (i.e., $q/4p \geq \sum_{i=0}^{\eta-1} X_i$), we can then ensure that $\delta \leq 10^{-3}$. To confirm the above theoretical derivation, we simulated the size of the product $c_{\text{in}} \cdot \mathbf{w} * \mathbf{e}$ in Fig. 8 showing the respective bound for $\delta \leq 10^{-3}$ and $\delta \leq 2^{-40}$. Since the simulated bound for $\delta \leq 10^{-3}$ is $1,930$ (slightly better than the theoretical bound), we conclude that our theoretical error derivation is quite sharp with respect to the simulated one and can be used to capture the error characteristics for asymptotically small values of δ. Although a simulated approach is tighter, it can only provide bounds on relatively large δ, i.e., ones where simple Monte-Carlo simulation finishes in finite time.

Step 4) In this case, m is selected to meet the security constraint, and we can safely proceed. If m is too small, we follow the protocol in Fig. 3 and return to Step 1).

Step 5) After generating the ciphertext modulus q, we can explore the permitted approximation factor k by integrating the approximation error into the above error analysis procedure.

The generated variables and moduli become the BNN$_{\max}$ parameter set as listed in Table II, along with parameters for other types of layers. We demonstrate three sets of parameters for different layer configurations. BNN$_{\min}$ is for short convolution binary input quantizers, BNN$_{\text{mid}}$ for longer convolution also with small operands, and BNN$_{\max}$ for the largest parameter configuration that is able to handle a 8-bit 32×32 input convolved with a 4-bit 5×5 filter using a single ciphertext. The proposed HLT technique results in asymptotically smaller level of error amplification (up to 14 bits) compared to existing works. In particular, the error amplification in APAS is $\mathcal{O}(p_w \cdot \sqrt{c_i \cdot n_w})$, while the techniques in [1], [38] ends up with a $\mathcal{O}(p\sqrt{c_i \cdot m})$ factor (easily over 30 bits). We note that the main source of contribution for this asymptotic error-size reduction is that the HLT technique

TABLE IV
PERFORMANCE COMPARISON ON CONVOLUTION BENCHMARKS

Layer	Input Dimension	Filter Dimension	Parameter Set	Ciphertext Size (MBytes)	Latency (ms) (ASIC)	Energy (μJ) (ASIC)
CR	$(3 \times 32 \times 32)$	$(192 \times 5 \times 5)$	$\text{BNN}_{\max}$	1.58	52.9	5.33
BCR	$(192 \times 32 \times 32)$	$(160 \times 1 \times 1)$	BNN_{mid}	2.49	113	26.9
BCR	$(160 \times 32 \times 32)$	$(96 \times 1 \times 1)$	BNN_{mid}	1.81	57.0	13.5
BCR	$(96 \times 16 \times 16)$	$(192 \times 5 \times 5)$	BNN_{mid}	0.585	423	13.5
BCR	$(192 \times 16 \times 16)$	$(192 \times 1 \times 1)$	BNN_{mid}	0.679	34.0	13.5
BCR	$(192 \times 16 \times 16)$	$(192 \times 1 \times 1)$	BNN_{mid}	0.679	34.0	13.5
BCR	$(192 \times 8 \times 8)$	$(192 \times 3 \times 3)$	$\text{BNN}_{\min}$	0.195	76.8	13.5
BCR	$(192 \times 8 \times 8)$	$(192 \times 1 \times 1)$	$\text{BNN}_{\min}$	0.169	8.59	13.5
BCR	$(192 \times 8 \times 8)$	$(10 \times 1 \times 1)$	$\text{BNN}_{\max}$	0.102	0.489	13.5
			Technique	Ciphertext Size	Latency (ms)	Energy (μJ)
Total	-	-	APAS-ASIC	8.30 MB	$\approx$ 800	194
	-	-	APAS-FPGA	8.30 MB	$\approx$ 927	3.17×10^5
	-	-	[1], [38]-ASIC	23.3 MB	$\approx$ 6,493	5,078
	-	-	[1], [38]-FPGA	23.3 MB	$\approx$ 3,784	26.1×10^5
	-	-	[1], [38]-CPU	23.3 MB	$\approx$ 2,799	2.8×10^8

does not need NTT, as NTT operates over the entire lattice dimension m and plaintext modulus p.

D. Synthesis Results and Comparisons

Based on the parameter sets and Step 5) in the APAS framework, the approximate hardware are synthesized, and the statistics are summarized in Table III. For APAS, since we use a fixed hardware architecture for the entire NN inference, the hardware cost is estimated for $\text{BNN}_{\max}$ and will be used across all layer evaluations. Moreover, the hardware performance are provided on both the ASIC and the FPGA platforms. It is observed from Table III that, while FPGA enables per-network reconfiguration of multiplier architectures, it does induce a significant power overhead when compared to an ASIC implementation. We compare our instantiated BNN architecture with general-purpose 55-bit and 60-bit modular multipliers, as these architectures can be adopted in NTT-based homomorphic convolutions in DArL [38] and Gazelle [1]. We observe 2× latency, 6×–9× area and 2.3×–3.2× power reductions in the ASIC implementation.

Using Table III, we compare the ciphertext size, latency and energy efficiency of APAS to existing works [1], [38]. The comparisons are performed using all of the convolution layers in Table IV. The latency for APAS is estimated by

$$(c_i \cdot m + \frac{c_i \cdot c_o}{c_n} \cdot n_w \cdot m)) \cdot d, \tag{46}$$

where $c_n = \lfloor n/(n_u + n_w - 1) \rfloor$, i.e., the number of convolution results that fit into a single ciphertext. For instance, for $\text{BNN}_{\max}$, we have a circuit delay of $d = 3.5\,\text{ns}$, lattice dimension of $m = 1,049$, filter size of $n_w = 25$, number of input channels $c_i = 3$, and number of output channels $c_o = 192$. The per-layer latency in this case becomes 52.9 ms. For comparison, we estimated the performance of Gazelle [1] and DArL [38] using the ASIC multiplier in Table III with Algorithm 2 as

$$\begin{aligned}&\big((n \lg n) + (n \lg n \cdot \lg q) \cdot (c_n n_w - 1)\\&+ (n \lg n + n) \cdot c_o\big) \cdot \frac{c_i}{c_n} \cdot d_{\text{NTT}}\end{aligned} \tag{47}$$

It is obvious that the NTT-based approach involves much more operations, as we need a large number of NTT runs for the rotations (Line 1 in Eq. (47)) as well as the filter applications (Line 2 in Eq. (47)). Finally, the CPU performance are extrapolated from the benchmarks reported in existing works [1], [38].

As for the the ciphertext size, the following estimation is used for our technique

$$2m \cdot \big((c_i \cdot \lceil \lg \mathsf{Approx}(q) \rceil) + (c_o \cdot \lceil \lg \mathsf{Approx}(q) \rceil)\big). \tag{48}$$

Since existing works do not instantiate approximate hardware, their ciphertext size is simply calculated as $2 \cdot m((c_i \cdot \lceil \lg q \rceil) + (c_o \cdot \lceil \lg q \rceil))$.

Compared to the best known works [1], [38], we achieve nearly 3× ciphertext reduction, due to the small error growth and approximate hardware. Furthermore, we see up to 8× inference time reduction over the ASIC implementation and 3.4× over the CPU implementation of Gazelle. Moreover, we reduce the total energy consumption of evaluating all of the linear layers in Table I by 26× and 8.2× on ASIC and FPGA platforms, respectively. Lastly, we point out that our software-hardware co-design approach is successful in keeping the runtime of the ASIC-based design competitive against the FPGA and CPU ones, while reducing the total energy consumption by $10^3\times$–$10^5\times$. Therefore, we consider the proposed HLT technique and APAS as essential building blocks in establishing truly practical NN-based secure inference infrastructures.

VI. CONCLUSION

In this work, we proposed a novel way of performing homomorphic linear transformation over RLWE-based PAHE schemes that are widely used in machine-learning-as-a-service applications. To maximize the efficiency of the HLT operators, we also designed a hardware-software co-design framework that automatically instantiates network-specific parameters and hardware platforms. In the experiments, it was shown that the latency in a real-world neural network architecture can be reduced by as much as 8× on ASIC and 3.4× on CPU. Furthermore, we observe up to 3× ciphertext and all-round energy reductions across platforms.

References

[1] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A low latency framework for secure neural network inference," *arXiv preprint arXiv:1801.05507*, 2018.

[2] M. Keller, E. Orsini, and P. Scholl, "MASCOT: faster malicious arithmetic secure computation with oblivious transfer," in *Proc. 2016 Conference on Computer and Communications Security*. ACM, 2016, pp. 830–842.

[3] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1575–1590.

[4] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annual Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 160–164.

[5] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, "XONN: Xnor-based oblivious deep neural network inference." *IACR Cryptology ePrint Archive*, vol. 2019, p. 171, 2019.

[6] A. Brutzkus, O. Elisha, and R. Gilad-Bachrach, "Low latency privacy preserving inference," *arXiv preprint arXiv:1812.10659*, 2018.

[7] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *International Conference on Selected Areas in Cryptography*. Springer, 2013, pp. 68–85.

[8] S. S. Roy *et al.*, "Compact ring-LWE cryptoprocessor," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 371–391.

[9] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede, "Efficient ring-LWE encryption on 8-bit AVR processors," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 663–682.

[10] Y. Doröz, E. Öztürk, and B. Sunar, "Accelerating fully homomorphic encryption in hardware," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1509–1521, 2015.

[11] S. S. Roy *et al.*, "Hardware assisted fully homomorphic function evaluation and encrypted search," *IEEE Transactions on Computers*, vol. 66, no. 9, pp. 1562–1572, 2017.

[12] S. S. Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, "FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data," in *2019 IEEE International symposium on high performance computer architecture (HPCA)*. IEEE, 2019, pp. 387–398.

[13] S. Bian, M. Hiromoto, and T. Sato, "Filianore: Better multiplier architectures for lwe-based post-quantum key exchange," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[14] V. Migliore *et al.*, "A high-speed accelerator for homomorphic encryption using the karatsuba algorithm," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 138, 2017.

[15] W. Liu, S. Fan, A. Khalid, C. Rafferty, and M. O'Neill, "Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2459–2463, 2019.

[16] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2505–2522.

[17] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 394–399, 2020.

[18] "Homomorphic linear transformation," https://github.com/sbian3/homomorphicmatrix, April 2021, Kyoto University.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: Imagenet classification using binary convolutional neural networks," in *ECCV*. Springer, 2016, pp. 525–542.

[20] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[21] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 3–18.

[22] "Microsoft SEAL (release 3.6)," https://github.com/Microsoft/SEAL, June 2019, microsoft Research, Redmond, WA.

[23] Y. Polyakov, K. Rohloff, and G. W. Ryan, "PALISADE lattice cryptography library." https://git.njit.edu/palisade/PALISADE, 2018.

[24] S. Bian, W. Jiang, Q. Lu, Y. Shi, and T. Sato, "NASS: Optimizing secure inference via neural architecture search," *arXiv preprint arXiv:2001.11854*, 2020.

[25] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[26] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," *arXiv preprint arXiv:1811.09953*, 2018.

[27] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MinioNN transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 619–631.

[28] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[29] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 38th IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[30] E. Öztürk, Y. Doröz, E. Savaş, and B. Sunar, "A custom accelerator for homomorphic encryption applications," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 3–16, 2016.

[31] J. Cathebras, "Hardware acceleration for homomorphic encryption," Ph.D. dissertation, Université Paris-Saclay, 2018.

[32] F. Turan, S. S. Roy, and I. Verbauwhede, "Heaws: An accelerator for homomorphic encryption on the amazon aws FPGA," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1185–1196, 2020.

[33] B. Hunt, "A matrix theory proof of the discrete convolution theorem," *IEEE Transactions on Audio and Electroacoustics*, vol. 19, no. 4, pp. 285–288, 1971.

[34] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The fast fourier transform and its applications," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 27–34, 1969.

[35] C. Peikert, "How (not) to instantiate ring-LWE," in *International Conference on Security and Cryptography for Networks*. Springer, 2016, pp. 411–430.

[36] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.

[37] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*. Springer, 2010, pp. 420–443.

[38] S. Bian, M. Hiromoto, and T. Sato, "DArL: Dynamic parameter adjustment for LWE-based secure inference," in *2019 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2019, pp. 1739–1744.

[39] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[40] S. Bian, M. Hiromoto, and T. Sato, "DWE: Decrypting learning with errors with errors," in *Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[41] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Cryptographers' Track at the RSA Conference*. Springer, 2011, pp. 319–339.

[42] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope." in *USENIX Security Symposium*, 2016, pp. 327–343.

[43] Q. Lou, B. Song, and L. Jiang, "AutoPrivacy: Automated layer-wise parameter selection for secure neural network inference," *arXiv preprint arXiv:2006.04219*, 2020.

[44] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 311–323.

[45] *Design Compiler I-2013.06*, Synopsys, Inc., 2013.

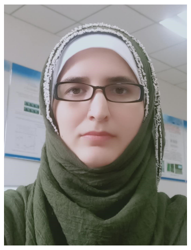
[46] A. Paszke and others, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[47] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of ring-lwe for any ring and modulus," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 461–473.

[48] J. Blömer and S. Naewe, "Sampling methods for shortest vectors, closest vectors and successive minima," *Theoretical Computer Science*, vol. 410, no. 18, pp. 1648–1665, 2009.

[49] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

Song Bian Song Bian received B.S. from University of Wisconsin-Madison in 2014. He received M.S. and Ph.D. from Kyoto University in 2017 and 2019, respectively. He is currently an assistant professor at Kyoto University. His main areas of interest include applied cryptography, secure multi-party computation and domain-specific hardware accelerators. He was a research fellow of the Japan Society for the Promotion of Science from 2017 to 2019.



Dur-e-Shahwar Kundi is currently a Post-doc Fellow at College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China under the scheme of China Postdoctoral Science Foundation 2019. She received her M.Sc. and Ph.D. degree in Electrical Engineering from National University of Sciences and Technology (NUST), Karachi, Pakistan, in 2010 and 2016 respectively. Her research interests include hardware security, cryptographic engineering and reconfigurable computing.

Kazuma Hirozawa received the B.E. degree in Electrical and Electronic Engineering from Kyoto University in 2020, and is now pursuing a master degree at the Graduate School of Informatics, Kyoto University. His current research interests include applied cryptography and hardware accelerators.

Weiqiang Liu (M'12-SM'15) received the B.Sc. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China and the Ph.D. degree in Electronic Engineering from the Queen's University Belfast (QUB), Belfast, UK, in 2006 and 2012, respectively. In Dec. 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently a Professor and the Vice Dean of the college. He has been awarded the prestigious Excellent Young Scholar Award by National Natural Science Foundation of China in 2020. His research interests include approximate computing, hardware security and VLSI design for digital signal processing and cryptography.

Takashi Sato received B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan. He was with Hitachi, Ltd., Tokyo, Japan, from 1991 to 2003; with Renesas Technology Corp., Tokyo, Japan, from 2003 to 2006; and with the Tokyo Institute of Technology, Yokohama, Japan. In 2009, he joined the Graduate School of Informatics, Kyoto University, Kyoto, Japan, where he is currently a professor. He was a visiting industrial fellow at the University of California, Berkeley, from 1998 to 1999. His research interests include CAD for nanometer-scale LSI design, fabrication-aware design methodology, and performance optimization for variation tolerance.

Appendix

A. Proof for Proposition 1

Proposition 1. *The cryptosystem specified in Section III-B is as secure as that specified in Section II-B against semi-honest adversaries.*

Proof. Without loss of generality, we assume that the client performs the encryption function over some input $\mathbf{u} \in \mathbb{Z}_p^n$, and the server carries out the homomorphic computation $\mathsf{HLT}_{\mathfrak{F}}$ over the ciphertext $[\mathbf{u}]$. From the key generation and encryption procedures for both the original and the modified cryptosystems, we know that the client holds the private key $\mathbf{s}$, and outputs the ciphertext $[\mathbf{u}] = (\mathbf{c}_0, \mathbf{c}_1)$. Here, we safely hold it true that the original BFV cryptosystem [20] achieves indistinguishability under chosen-plaintext attack (i.e., secure against IND-CPA adversary). In addition, we assume that the adversary against the client in the secure inference protocol (the server) is semi-honest, i.e., the server follows the inference protocol honestly, but wishes to learn as much information as possible from the client.

Recall that the IND-CPA security of the BFV cryptosystem [20] relies on the hardness of the decision ring learning with errors (R-LWE) problem, formulated as follows. Note that some notations are changed to match those used in this manuscript.

Definition 1. *(Definition 1 in [20]) For security parameter λ, let $\Phi_m(x)$ be the m-th cyclotomic polynomial with degree $n = \phi(m)$, and let $R = \mathbb{Z}[x]/\Phi_m(x)$. Let $q = q(\lambda) \geq 2$ be an integer. For a random element $\mathbf{s} \in R_q$ and a distribution $\chi = \chi(\lambda)$ over R, denote with $A_{\mathbf{s},\chi}^{(q)}$ the distribution obtained by choosing a uniformly random element $\mathbf{a} \leftarrow R_q$ and noise term $\mathbf{e} \leftarrow \chi$ and outputting $(\mathbf{a}, \mathbf{a} * \mathbf{s} + \mathbf{e} \bmod \Phi_m(x), q)$. The Decision-R-LWE$_{d,q,\chi}$ problem is to distinguish between the distribution $A_{\mathbf{s},\chi}^{(q)}$ and the uniform distribution $U(R_q^2)$.*

We also make use the following corollary from [47].

Corollary 2. *(Corollary 6.3 in [47]) Let K be any number field of degree n and $R = \mathcal{O}_K = \mathbb{Z}[\xi_m]$ be its ring of integers. Let $\alpha \in (0,1)$ be some real number, and $q \geq 2$ be some integer where $\alpha q \geq 2 \cdot \omega(1)$. There is a polynomial-time quantum reduction from K-SIVP$_\gamma$ to (average-case, decision) R-LWE$_{q,\upsilon_\alpha}$ for any*

$$\gamma \leq \{\max(\omega(\sqrt{n \log n}/\alpha), \sqrt{2}n\} \quad \text{(A1)}$$

Here, K-SIVP is the shortest independent vector problem restricted to the ideal lattices in the number field K, and solving K-SIVP is assumed to be $\mathcal{NP}$-hard [47], [48]. Combined with Definition 1, Corollary 2 from [47] basically states that the decision R-LWE problem is hard regardless of the choices of the number field K and modulus q.

At this stage, we only need to prove that the view of any adversary on the modified cryptosystem is indistinguishable from the original BFV cryptosystem as described in Section II-B. Since we use two types of cyclotomic polynomials to instantiate the modified cryptosystem, we divide the proof into two parts: first, we use a simple contradiction to prove Proposition 1 in the special case where m is a power of two. Then, we provide a simulator-based general proof for any value of m with Corollary 2.

Power-of-Two Case: Observe that when m is a power of two, the encryption functions in Eq. (1) and Eq. (7) are equivalent by the convolution theorem. Specifically, we have that, for the same $\mathbf{a}$, $\mathbf{s}$, $\mathbf{u}$ and $\mathbf{e}$,

$$\mathbf{c}_0 = -\mathbf{a}, \tag{A2}$$

$$\begin{aligned}\mathbf{c}_1 &= \mathsf{NegINTT}(\mathsf{NegNTT}(\mathbf{a}) \circ \mathsf{NegNTT}(\mathbf{s})) + \mathbf{e} + \mathbf{u} \cdot q/p \\ &= (\mathbf{a} * \mathbf{s} + \mathbf{e} + \mathbf{u} \cdot q/p) \bmod \Phi_m(x).\end{aligned} \tag{A3}$$

Now, suppose that the modified cryptosystem is not secure. In other words, there exists an efficient (i.e., polynomial-time) distinguisher $\mathcal{D}$ that can distinguish samples from the modified cryptosystem in the form

$$[\mathbf{u}]' = (\mathbf{c}'_0, \mathbf{c}'_1) = (-\mathbf{a}', \mathbf{a}' * \mathbf{s}' + \mathbf{e}' + \mathbf{u} \cdot q/p \bmod \Phi_m(x)) \tag{A4}$$

encrypted using Eq. (7) from uniform random polynomials $U(R_q^2)$ in R_q. With Eq. (A2), we know that with the exactly same $\mathbf{a}'$, $\mathbf{s}'$, $\mathbf{u}$ and $\mathbf{e}'$, Eq. (1) also outputs the exactly same ciphertext $(\mathbf{c}'_0, \mathbf{c}'_1)$, i.e.,

$$\begin{aligned}&[\mathbf{u}]' = (\mathbf{c}'_0, \mathbf{c}'_1) \\ &= (-\mathbf{a}', \mathsf{NegINTT}(\mathsf{NegNTT}(\mathbf{a}') \circ \mathsf{NegNTT}(\mathbf{s}')) + \mathbf{e}' + \mathbf{u} \cdot q/p)\end{aligned} \tag{A5}$$

which is a valid BFV ciphertext. As a result, if the modified cryptosystem is not secure, we obtain a distinguisher $\mathcal{D}$ that efficiently distinguishes valid BFV ciphertexts from uniform random polynomials, which contradicts the security of the BFV cryptosystem.

While the above proof works for m a power of two, we cannot directly reduce the security of the modified cryptosystem to the original BFV cryptosystem for arbitrary values of m (including the prime-order case), as for different m, the instantiated cryptosystems have different cyclotomic fields and moduli. In what follows, we provide a general security proof for the modified cryptosystem with any lattice dimension m and ciphertext modulus q.

General Case: Here, we construct a simple simulator $\mathcal{S}$ that simulates the view of the semi-honest server on the modified cryptosystem. The simulator is constructed as follows.

1) $\mathcal{S}$ chooses a random tape.
2) $\mathcal{S}$ generates and outputs an encryption of zero $[\mathbf{0}]$ using the key generation and encryption steps specified by the modified cryptosystem with the random tape.

Essentially, $\mathcal{S}$ simulates the behavior of the client in the secure inference protocol π without knowing any input. Here, the real view of the semi-honest server on the client is

$$\mathsf{view}^{\pi}(n, p, q, \chi, \mathbf{u}) = (\mathbf{c}_0, \mathbf{c}_1) = [\mathbf{u}]. \tag{A6}$$

Meanwhile, the output of the simulator is

$$\mathcal{S}(n, p, q, \chi) = (\mathbf{c}'_0, \mathbf{c}'_1) = [\mathbf{0}]. \tag{A7}$$

From Definition 1 and Corollary 2, we know that the decision-R-LWE problem is hard for any number field with any modulus (while the reduction is quantum only, classical reductions also exist that guarantee the hardness of decision R-LWE for all cyclotomic fields with any modulus through modulus switching [47], [49]). In other words, there exists no efficient distinguisher that distinguishes R-LWE samples computed using Eq. (7) from uniform random polynomials $U(R_q^2)$, i.e.,

$$[\mathbf{u}] \overset{c}{\equiv} U(R_q^2), \tag{A8}$$

for any input $\mathbf{u} \in \mathbb{Z}_q^n$. Here, $\overset{c}{\equiv}$ means that the left and right hand sides are computationally indistinguishable. Since $[\mathbf{0}]$ is a valid encryption of a vector of zeroes, we know that

$$[\mathbf{u}] \overset{c}{\equiv} U(R_q^2) \overset{c}{\equiv} [\mathbf{0}] \tag{A9}$$

for any input $\mathbf{u}$. In other words, a semi-honest server cannot distinguish a valid encryption procedure from a simulator assuming the hardness of the decision R-LWE problem. Consequently, we can reduce the semantic security of protocol π with the modified cryptosystem directly to the hardness of the decision R-LWE problem, i.e., any efficient semi-honest adversary who breaks the security of π can efficiently solve the decision R-LWE problem (and therefore the K-SIVP problem). $\square$