Realizing Homomorphic Secure Protocols through Cross-Layer Design Techniques

Song BIAN

To
my father, my mother, Rel, and Pino

Abstract

Recent technological advance sees a rising need for secure computing protocols. With cloud computing, data outsourcing and machine learning paradigms widely adopted, large data exchanges between untrusted parties put unprecedented demand on efficient protocols that allows for blind computation over encrypted inputs.

Examples for the need of multi-party secure protocols are abundant. For instance, electronic voting system requires the identities of individual voters to be remained private, while keeping the voting results verifiable by the election observers. Meanwhile, searchable encryption schemes try to ensure that a legal user can search through some database without leaking the search records to the server. More recently, with the development of the machine learning as a service (MLaaS) computing model, both user input privacy and server model privacy are required. Privacy concerns have become one of the main obstacles that prevent the widespread use of artificial intelligent algorithms as basic infrastructures.

This dissertation is devoted to use cross-layer techniques to realize the above mentioned secure protocols. In particular, emphasis is placed on lattice-based homomorphic encryption (HE) is placed, as their simple algebraic structures and flexible cryptographic constructions provide new solutions to practical secure computing schemes. We refer to security schemes based on HE schemes as homomorphic secure protocols (HSPs).

In the first part of this dissertation, better hardware primitives are investigated to improve the performance of learning-with-error (LWE) based cryptosystems, including HSPs. New architectures for the basic arithmetic circuits (i.e., the multiplier units) of cryptosystems based on LWE are proposed. Different from existing approaches that rely on built-in general-purpose multipliers, the proposed modular multiplier architectures target on minimalistic design. As a result, it is revealed that many of the software-level optimization techniques, such as modulus selection and lazy reduction, are not as useful on the proposed platforms. A focus is also put on the error characteristic of approximate multipliers for LWE decryption. It is discovered that the multiplier

architecture can be further simplified if such numerical approximation is adopted. Compared to existing approaches, an application-specific approximate multiplier can provide up to 4–12x area and 4–21x energy reductions.

The second part of this dissertation concentrates on secure naïve Bayesian filter (SNBF) based on additive HE. The main objective of SNBF is a system-level design optimization for secure email filters, such that blind filtering on encrypted emails can be practically feasible. In the algorithmic layer, weight quantization and parallel embedding techniques are suggested. In the hardware layer, multiplier circuits with large integer inputs are proposed, such that the blind filtering process, which in our case consists of more than 700,000 multiplications between 2048-bit integers, can be computed in a power-efficient manner. Results show that on both CPU and dedicated platforms, the proposed techniques are effective in reducing the overall system cost. An average email in a real-world email dataset can be classified within 0.5 s.

Based on the hardware platforms and algorithmic techniques proposed above, secure inference for MLaaS are investigated in the third part of the dissertation. In designing a better protocol, the HE-based convolution and fully-connect layers are targeted. For convolution layers, an oblivious frequency-domain convolution protocol is proposed. The key idea is that by combining homomorphic encryption with additive secret sharing, the computations involved in secure convolution can be simplified to normal discrete Fourier transformation and a lightweight homomorphic (coefficient-wise) multiplication. For both convolution and fully-connect layers, weight quantization and error characterization techniques are applied to minimize the parameter expansions. Numerical experiments show that the proposed techniques can reduce the communication bandwidth by 2–3x in convolution and fully connected layers, in addition to greatly improving the scalability of neural-network-based secure inference.

With a focus on lattice cryptography, this dissertation addresses efficiency concerns over HSPs. As increasing numbers of devices are integrated into a single network, it is expected that more cryptographic, algorithmic and architectural primitives will be combined to achieve low-cost trusted computing environments. As demonstrated in this dissertation, instead of optimizing each independent layer separately, cross-layer design techniques that are aware of the entire system are critical to the overall performance of the interconnected network, and are therefore essential to a practically useful secure protocol.

Acknowledgments

First and foremost, I would like to express my sincere appreciation to my supervisor, Prof. Takashi Sato for his critical insights in, continual support for, and valuable advices on all aspects of my life and my research works. His endeavor and dedication are extraordinary, and it is because of him that this dissertation comes into existence.

Second, this dissertation receives immense helps from Prof. Masayuki Hiromoto. Numerous discussions with him results in better idea from the start to more refined experiments in the end. Working with him was a joyful experience, and I am looking forward to collaborating with Prof. Hiromoto again in the future.

Next, I want to appreciate Prof. Hidetoshi Onodera, and Prof. Yasuo Okabe for their precious reviews and comments in examining this dissertation. This dissertation is polished into a much better shape owing to their efforts.

The previous and current lab members at the Processor Architecture and Systems Synthesis Laboratory have also been a great source of support for me and my life in Japan. Prof. Michihiro Shintani and Ms. Shuko Nishiyama are the primary figures in helping with my personal and academic lives. I am almost used to consult with them before taking actions.

Accompanied by Mr. Shumpei Morita, Mr. Hidenori Gyoten, Mr. Yuki Tanaka, Mr. Kazuki Oishi, and numerous Animes, my Otaku dreams are fulfilled in a manner that was unimaginable; a life that I assumed to only be possible somewhere further than the universe. I thank Akagi, Rel, Pino, Shirase Kobuchizawa, Mari Tamaki, Hinata Miyake, Yuzuki Shiraishi, Violet Evergarden, 02, Mai Sakurajima, $\cdots$ (not a finite set). You powered my dream, and you powered my life.

Additionally, this dissertation was partially supported by JSPS KAKENHI Grant No. 17H01713, 17J06952, 18H03214, and Grant-in-aid for JSPS Fellow (DC1). I also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

Finally, it is hard to put my gratitude toward my parents, Hai Bian and Xinying Yao, into sentences, as words do not bear weights. I hope that I lived

up to your expectations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Homomorphic Secure Protocols

The design and implementation of secure protocols involving multiple parties for the computation of a joint function consist a new area of research in the age of cloud computing. Due to the multidisciplinary nature of the topic, contributions from various fields of expertise are drawn. In designing any protocol of this kind, one needs to comprehend the entire system, from the high-level application layer, to the intermediate layer of cryptographic building blocks, and finally to the low-level hardware layer, such that the complex design trade-offs are well-captured.

The primary motivation for multi-party secure protocols is the increasing demand from the software as a service (SaaS) model, especially machine learning as a service (MLaaS). Traditionally, data are encrypted over the communication channel but remain plaintext to all the involved parties. However, in such computing paradigm, the service user and provider often regard each other as untrusted parties, and are hesitated to exchange private data and proprietary models in cleartext.

While multi-party secure protocols can provide perfect solution to the above dilemma, the main obstacle for adopting the protocols in practice is the efficiency (or, inefficiency) problem. Most general constructions that permit oblivious function evaluations are only theoretically viable, such as function encryption [1] or general garbled circuit [2]. Meanwhile, secure protocols equipped with homomorphic encryption (HE) schemes, referred to as homomorphic secure protocols (HSPs), are gaining increasing popularities. While the homomorphic property of certain public-key encryption (PKE) schemes was considered as a security risk in some applications [3], it was soon realized that HE can be highly efficient with linear function evaluations such as

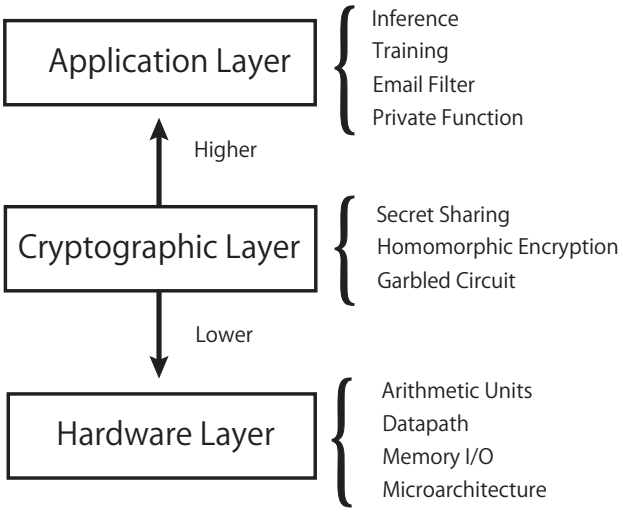

Figure 1.1: Three abstraction layers that characterize the design architectures of HSPs focused on in this dissertation.

summing and averaging, which are natural arithmetic operations in applications such as electronic voting systems [4]. As HE schemes allow for blind evaluations directly on ciphertexts by untrusted parties, HSPs are generally much more efficient in terms of the computational and communicational overhead when compared to their traditional counterparts.

The critical question that still arises, however, is exactly how "efficient" can such protocols be, where the efficiency requirement is on both the running time and total energy consumption. For example, a recently proposed fully homomorphic encryption (FHE) based secure database system requires seven days to retrieve a single row from a 10-record database [5], where the majority of the computational time is spent on the expensive bootstrapping operation (bootstrapping can be thought of as a re-encryption step that allows for additional homomorphic evaluations). It is obvious that such protocols are neither time nor energy efficient. In contrast, some of the protocols, such as key-exchange and secure inference schemes, have computational burdens on edge devices with constrained resources. In such cases, the overall energy efficiency is of particular concern. To date, most HSPs proposed and their related implementations still reside in the research domain, but several attempts were made for standardization [6,7] and commercialization [8,9]. The key observation of this dissertation is that, with a number of abstraction layers and several sets of different cryptographic tools in hand, building an HSP turns into a cross-layer design optimization problem. As illustrated in Fig. 1.1, this dissertation defines

three abstraction layers in designing an HSP, and puts an emphasis on their relationships. First, in the application layer, a precise security model and attack surface need to be defined according to the particular application. In addition, security requirements are generally application-dependent, and in HSPs, a change in the security standard can result in a completely different set of cryptographic primitives being used. Second, the trade-off between such primitives is also complex, as some perform better with smaller-size datasets, and others are asymptotically more efficient. Finally, the hardware platform upon which the protocols are executed also plays an important role, for different cryptographic primitives depend on distinct hardness assumptions (e.g., factoring versus lattice). With the assistance of hardware accelerators, as shown in this dissertation, it is possible to apply techniques that improve the time and the energy efficiency across all design layers.

1.2 Lattice Cryptography

The development of lattice cryptography has drawn major attentions in the cryptographic community over the past decade. At first, as noted in [10], the major benefit of cryptographic constructions that rely on lattice problems is the worst-case to average case reductions provided in [11]. Subsequent development of the popular learning with errors (LWE) problem introduced by Regev [12] makes lattice problems a foundation for a plethora of cryptographic primitives as well as constructions. For example, LWE enables fully homomorphic encryption (FHE) [13–17] whose construction was previously unknown. Traditional cryptographic constructions are also suggested, such as efficient public key encryption (PKE) [12,18], key-exchange [6,7,19–21], and digital signature schemes [13,22]. In addition, variants of the LWE problem are also proposed, which include Ring-LWE (RLWE) [23], Polynomial-LWE [24], Module-LWE [25], and Order-LWE [26]. The standard LWE hardness assumption also enjoys the strong security guarantee against quantum algorithms [12,27].

Besides being the only known hardness assumption permits FHE, the general additive structure of LWE-based cryptosystems allows for single-instruction multi-data (SIMD) operations in the homomorphic domain, as proposed by Smart and Vercaueren [28]. Furthermore, slot permutation based on the homomorphic automorphism technique proves to be extremely efficient for handling homomorphic matrix operations [29]. Consequently, recent HSPs often adopt LWE-based HE for efficient protocol execution, over the traditional factoring-based schemes. However, for its short history, lattice cryptography deserves much more optimizations, especially in the hardware layer, and lattice-based cryptosystems exhibit properties that were not known to be viable.

Therefore, while factoring-based cryptosystems are also adopted for practical reasons as in Chapter 5, a larger portion of this dissertation devotes to the optimization of lattice-based cryptographic constructions and HSPs.

1.3 Objectives and Methods

Here, a brief summary of the main objectives and the techniques used in this dissertation is provided.

1.3.1 Specialized Hardware Primitives for Lattice Cryptography

The first topic discussed in this dissertation is designing better hardware primitives for lattice cryptography. In order to further study and enhance the efficiency of (R)LWE-based schemes, dedicated multiplier architectures are proposed in this dissertation. First of all, in Chapter 3, (R)LWE-based post-quantum (PQ) key exchange protocols are examined. As current art lacks specialized multiplication units for such protocols, multiplier architectures are developed for both LWE and RLWE. By closely inspecting the security and error behaviors, new parameter sets are proposed for LWE to minimize client-side computations. Upon the proposed architectures, the practical efficiencies of general LWE versus RLWE are evaluated with respect to the instantiated parameters. The results indicate that a hardware-software combined effort can substantially reduce the client-side computation burdens for LWE-based PQ key exchange, to the extent that the general LWE construction is almost as efficient as its RLWE counterpart.

Next, in Chapter 4, a connection is made between LWE, a concept in the cryptographic layer, and approximate computing (AC), a field of study in the hardware layer. Different from existing hardware accelerators, the important observation is that LWE-based cryptosystems are inherently approximate, and that the decryption for LWE is probabilistic by design. Therefore, as long as the parameters permit, the probabilistic nature of LWE allows approximate hardware to be deployed virtually with no additional overhead. By carefully bounding the error sizes resulted from numerical approximation, a set of approximate multipliers is instantiated for a public-key encryption scheme and a homomorphic encryption scheme. Through the experiment, essential ciphertext size reduction and decryption speed increase are observed without affecting the empirical decryption failure probability.

Through studying hardware primitives for lattice cryptography in detail, the observation is that, while individual hardware component is relatively

simple in nature, the way different components interact with each other in a particular system can be highly complex and application-dependent. Hence, the techniques used in this part of this dissertation, e.g., hardware-aware parameter estimation methods, serves as an important entry point to the subsequent development of techniques in the application and primitive layers in HSPs.

1.3.2 Homomorphic Bayesian Filter

Although it is theoretically viable for (lattice-based) FHE schemes to evaluate circuits of polynomial depth [15, 16, 30], no known FHE, at the moment, is practically feasible. Consequently, a solution to the privacy concerns over data outsourcing is much needed. As a perfect example, outsourcing personal or organizational email data to remote server presents to be challenging in terms of balancing privacy and efficiency. On one hand, since the server possesses a large amount training email samples, and thus a good email filter, the server should be able to filter ham/spam emails based on publicly-available word lists. On the other hand, in a traditional setup, this requires the server to have access to the plaintext email data, which violates privacy requirements in many situations [31].

The second part of this dissertation, Chapter 5, seeks a practical solution to the blind email filtering problem with the assistance of HE. The email filter used in this case is a Naïve Bayesian filter (NBF), which is one of the simplest forms of machine learning. Due to the simplicity of the (plaintext) protocol, the proposed secure Naïve Bayesian filter (SNBF) scheme is realized with only the Paillier scheme, an additive homomorphic encryption (AHE) scheme based on the decision composite residuosity problem. Weight quantization and parallel filtering techniques are devised that significantly reduced the computational overhead of SNBF. Using a real-world email dataset, it is discovered that even with a relatively small filter size of 10,000 words, a modern CPU still needs around 15 seconds to filter an average-length email. With an application-specific hardware multiplier, the per-email computation time can be reduced to 0.5 s.

In designing SNBF, the critical step is to simplify the original NBF, such that the arithmetic operations within can be efficiently implemented by HE and accelerated by the underlying hardware platform. HE-based protocols share many similarities with the situation where expensive algorithms are executed on resource-constrained hardware devices. Hence, hardware-oriented design techniques are found useful.

1.3.3 Homomorphic Inference Engine

Based on the insights gained in the process of studying lattice-orient hardware architectures and secure email filter, it is realized that the design of secure inference engines, which include the filtering process of NBF, represents a novel field of research that demands design perspectives across the application, primitive and hardware layers.

As mentioned, the design and implementation of multi-party secure inference in the machine learning as a service (MLaaS) model attract increasing attentions, especially that are based on artificial neural networks (ANN) [32–37]. In Chapter 6, a secure inference protocol is proposed. Here, a specific ANN-based secure inference scheme, called Gazelle [35], is targeted. Based on the techniques developed in Chapter 4, the error behaviors of the homomorphic layers in Gazelle are characterized both theoretically and empirically. Moreover, it is demonstrated that frequency-domain convolution can be applied in a homomorphic manner via a novel application of the convolution theorem. The quantization technique from Chapter 5 is also integrated into the proposed protocol to provide a trade-off between efficiency and prediction accuracy. Finally, the proposed protocol obviously benefits from the hardware multipliers proposed in Chapters 3 and 4.

1.4 Organization

The basic layout of this dissertation can be summarized as follows. First, in Chapter 2, notations used throughout and basics on various cryptosystems are reviewed. Preliminary knowledge on related works are also presented.

Next, Chapters 3 and 4 discuss better hardware primitives for LWE cryptography. For PQ key exchange schemes, application-specific hardware multipliers are proposed for both the LWE and RLWE cases. A specific parameter set is also instantiated for the LWE-based cryptosystem, and the two cryptosystems are compared. While RLWE is still more efficient overall, given the security benefit, LWE remains as a competitive choice. To further advance the practicality of LWE-based cryptosystems, approximate computing methods are used to reduce computational cost and communicational bandwidth.

Chapter 5 then describes the protocol and implementation of SNBF. The AHE-based protocol is first outlined, and the parallel filtering technique is then sketched. With the proposed specialized hardware accelerator, it is shown that an email can be securely filtered in less than a second.

Last but not least, the secure inference engine scheme, named ENSEI, is illustrated in Chapter 6. Details on the frequency-domain secure protocol is

provided, along with rigorous error analyses for parameter instantiations across network layers. Significant performance improvements are observed with only negligible overheads added to the system.

Finally, this dissertation is summarized in Chapter 7, along with future prospects of cross-layer design methodologies and open questions examined.

Chapter 2

Backgrounds and Related Works

2.1 Ciphers, Secure Protocols and Homomorphic Encryption Schemes

A cipher is a set of algorithms that perform the encryption and decryption operations on some plaintext message. When the plaintext string m is encrypted as a ciphertext string c, the security guarantee is that, without a legitimate decryption procedure, m cannot (without a significant amount of effort) be obtained from c. On the other hand, the correctness requirement states that the decryption procedure can be easily carried out by any authorized party.

The above description can be abstractly illustrated by Fig. 2.1, where Enc and Dec denote the encryption and decryption procedures, respectively. From a mathematical standpoint, Enc and Dec serve merely as maps that operate between the plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$, i.e., $\mathsf{Enc} : \mathcal{P} \to \mathcal{C}$ and $\mathsf{Dec} : \mathcal{C} \to \mathcal{P}$. When these maps happen to satisfy some special properties, they form the basis of secure protocols, such as symmetric-key encryptions and public-key encryptions. It is natural, therefore, for the subsequent ciphers to advance in two directions: better security, and more properties. Unfortunately, the inevitable side-effect that comes with all the benefits is performance degradation, which is the main issue addressed by the cross-layer design techniques proposed in the rest of this dissertation.

In terms of security, Section 2.4 summarizes recent developments in lattice cryptography. The importance of these ciphers lies in the fact that they remain secure against general-purpose quantum computers, which are known to be much more efficient than classical computers, at least over a certain set of problems that guarantee the security of existing ciphers (e.g., the renowned Shor's algorithm [38]).

In studying the additional properties of existing ciphers, it is realized that

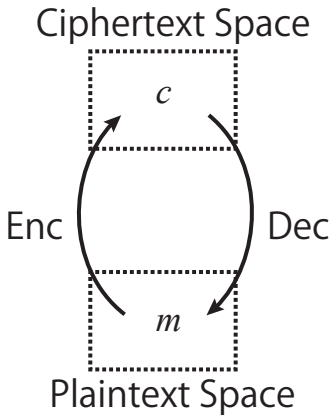

Figure 2.1: The abstract encryption and decryption functions.

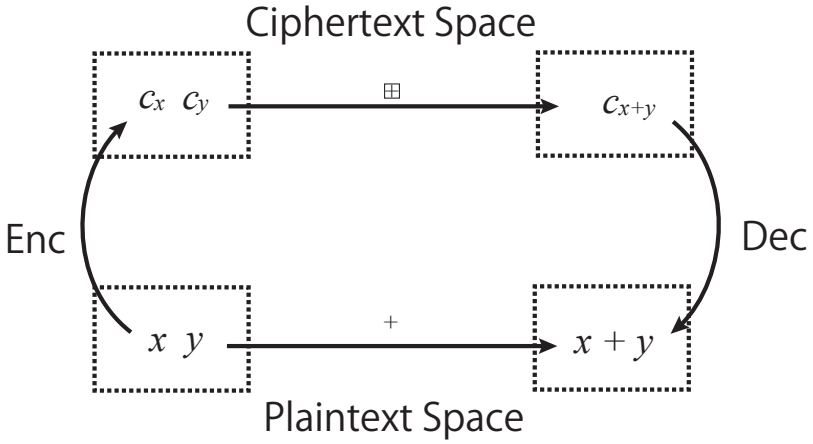

Figure 2.2: A homomorphic encryption scheme that preserves the operation $+$.

Enc and Dec can also be homomorphisms. Figure 2.2 explains what it means for an encryption function to also be a homomorphism. Informally, it means that for some operation $+$ in the plaintext space $\mathcal{P}$, there exists a corresponding $\boxplus$ in the ciphertext space $\mathcal{C}$, such that the following property is satisfied

$$\mathsf{Enc}(x + y) = \mathsf{Enc}(x) \boxplus \mathsf{Enc}(y). \tag{2.1}$$

Note that $+$ and $\boxplus$ are abstract operations, and do not strictly need to be additions. Detailed discussions on available homomorphic operations and cryptographic constructions that satisfy the corresponding homomorphic properties are presented in Sections 2.5 and 2.6.

2.2 A Brief Summary of Homomorphic Encryption Schemes and HSPs

Traditionally, partially homomorphic encryption (PHE) schemes are developed under the integer factorization assumption. These include additive and multiplicative homomorphic encryption schemes (AHE/MHE) [39, 40], but not fully HE schemes. As mentioned, with the fast development of lattice cryptography, a number of FHE schemes are proposed [13–17].

The exact definitions and detailed algebraic constructions of various HE schemes are presented in Section 2.5 and 2.6. Here, the practicality of HE schemes and their relationships to HSPs are discussed. As said, some FHE schemes require the bootstrapping operation to allow for the evaluation of polynomial-depth circuits [13, 41]. The bootstrapping procedures are not particularly "fast" in a practical sense, i.e., it takes 0.1 seconds for a recently proposed FHE scheme to bootstrap the single-bit output of a NAND gate. Other studies that avoid the bootstrapping procedures are known as leveled FHE schemes [15,16]. In particular, the ring-variant [17] of the GSW scheme [16] can multiply two ciphertexts in less than 4 ms on a GPU, with a ciphertext size of 487.5 KB per word. Unfortunately, as shown in Chapter 5, even for simple constructions such as a secure email filter, such schemes are still too costly in time and bandwidth.

In contrast to FHE, PHE is much more efficient. A homomorphic addition in the AHE scheme, Paillier [40] can be performed in microseconds on a conventional CPU, and lattice-based PHE schemes also have similar performances (e.g., HElib [42] and PALISADE [43] can both be used as efficient AHE libraries). This performance gap is the main motivation for the development of HSPs that rely only on PHE, instead of FHE. For example, in Gazelle [35], AHE is used for the evaluation of linear layers, and the resulting protocol is faster than the leveled FHE-based SecureML scheme by orders of magnitude. Although PHE-based HSPs are (obviously) still much slower than unencrypted protocols executed in a trusted environment, the performances of these HSPs do reside in practical domains, and can certainly become practical with additional optimization efforts.

Unfortunately, it is observed that previously proposed techniques and implementations are generally layer-specific. In other words, existing works focus on improving the efficiencies of HSPs from a certain aspect, that falls into one of the layers defined in Section 1.1. While these optimization efforts are important, they are insufficient, and sometimes not entirely useful in a cross-layer sense. A typical example is the optimization of modular multiplication on general-purpose CPUs. While a number of literatures propose

various lazy-reduction techniques [42–44] and special prime moduli [17] for efficient modular arithmetic, it is likely that HSPs are too costly to run on general-purpose computing devices. Meanwhile, on dedicated hardware platform, such techniques contribute marginally to the overall efficiency of HSPs, as the underlying hardware components can be optimized to provide single-cycle performance capabilities for modular arithmetic. Similar examples can be found between the application and cryptographic layers. For example, the HSP proposed in [45] optimizes cryptographic protocols for machine learning applications. In their work, Bos et al. use special representations to embed 52-bit floating point numbers into the Paillier ciphertext for implementing a secure Naïve Bayesian filter (NBF). In reality, what is discovered in Chapter 5 is that practical applications of NBFs generally do not need representations with such high precisions. As a result, significant speed increase can be obtained by working in a low-precision environment. The critical observation, and thereby main motivation, of this dissertation is that, in order to avoid unnecessary optimization efforts and protocol slowdown, a cross-layer perspective is essential.

2.3 General Notations

Throughout the dissertation, the standard notations of $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ are used to denote the set of integers, real numbers, and complex numbers, respectively. $\mathcal{R}$ is reserved for rings, and $\mathbb{F}$ is some field.

For vectors and matrices, $\mathbf{a} \in \mathbb{Z}_q^n$ is used to refer to a vector $\mathbf{a}$ with a dimension of n and elements drawn from $\mathbb{Z}_q$, the residual class of some integer modulus q. Vectors that have been transferred to the frequency domain through the discrete Fourier transform (DFT) or number-theoretic transform (NTT) are noted with a hat, e.g., $\hat{\mathbf{a}} = \mathsf{NTT}(\mathbf{a})$, and the Hadamard product of such vectors are in the form $\hat{\mathbf{a}} \circ \hat{\mathbf{b}}$. Matrices are written in capital letters without bold (A), and $\lg x$ is the shorthand for $\log_2 x$. The set notation $\{x\}$ refers to some set containing a number of elements, and the i-th element is denoted as x_i.

For lattice cryptography, this dissertation focuses exclusively on cryptosystems based on the generic [12] and ring [23] learning with errors (LWE) problems. A generic LWE instance is parameterized by (n, q, χ_σ), where n is the lattice dimension, q a modulus, and χ some distribution. Meanwhile, RLWE has similar parametrizations to LWE, where the parameters are also (n, q, χ_σ). For the sake of simplicity of presentation, the same notations are used for LWE and RLWE, while the underlying mathematical meanings differ. Here, n denotes the degree of some irreducible polynomial $f(x)$, which is functionally equivalent as the number n in generic LWE (specifies a lattice dimension). q and χ also serve similar purposes as their generic LWE counterparts.

Finally, Enc and Dec are used to abstractly identify the encryption and decryption functions, respectively.

2.4 Lattice-Based Cryptosystems

2.4.1 Errors and Error Distributions

Here, the notations in [46, 47] are adopted. χ_σ is the discrete Gaussian distribution with a standard deviation of σ, instead of the s parameter used in some of the literatures [12, 18] (note $\sigma = \frac{s}{\sqrt{2\pi}}$). χ_σ^n means n independent samples each drawn from χ_σ.

The security of every LWE cryptosystem depends on a certain level of errors being artificially embedded into the LWE instances of matrix-vector (or vector-vector) products during encryption. In other words, for two n-dimensional vectors $\mathbf{a}, \mathbf{s} \in \mathbb{Z}_q^n$ whose elements come from the residue classes of integer modulo q,

$$b = \langle \mathbf{a}, \mathbf{s} \rangle + e \tag{2.2}$$

is considered indistinguishable from uniformly random when e comes from a *continuous* Gaussian distribution, due to the original proofs [12,27]. Subsequent works approximate a continuous Gaussian with discrete Gaussian, denoted as χ_σ in some finite fields, where the "closeness" of discrete Gaussian to a continuous one is measured by statistical distance [18,46] or, more recently, the Rényi divergence [6, 48]. Theoretically, the relative error rate $\alpha_{\text{rel}} = \sqrt{2\pi}\sigma/q$ needs to satisfy the condition $q \cdot \alpha_{\text{rel}} \geq 2\sqrt{n}$ for the reduction to follow. However, for concrete parameter instantiations, it is often the case that only best plausible attacks are considered [6, 7, 18], where η is considerably smaller than the theoretical requirement. In addition, while an extremely high-quality discrete Gaussian is important for the security of LWE in certain cases (e.g., signatures [7, 22]) , some special assumptions can loosen this requirement. For instance, the standard deviation of the error distribution can be extremely small as in [6], and the distribution can be non-Gaussian as in [7], given a close enough statistical distance.

In this work, two error bounds provided in Lindner and Peikert (LP) [18] are extensively used.

Lemma 1. (Lemma 2.1 in [18]) *Let $c \geq 1$ and $C = c \cdot \exp\left(\frac{1-c^2}{2}\right) < 1$. Then for any real $\sigma > 0$ and any integer $n \geq 1$, it holds that*

$$\Pr[\|\chi_\sigma^n\| \geq c \cdot \sigma\sqrt{n}] \leq C^n. \tag{2.3}$$

Lemma 2. (Lemma 2.2 in [18]) *For any real* $\sigma > 0$ *and* $T > 0$*, and any* $\mathbf{x} \in \mathbb{R}^n$*, it holds that*

$$\Pr[\|\langle \mathbf{x}, \chi_\sigma^n \rangle\| \geq T\sigma\|\mathbf{x}\|] < 2 \cdot \exp\left(-\frac{T^2}{2}\right). \tag{2.4}$$

Here, $\exp(x) = e^x$, and $\|\mathbf{x}\|$ is the Euclidean norm of $\mathbf{x}$.

2.4.2 The Lindner-Peikert (LP) Cryptosystem

First of all, the basics on the LP cryptosystem are sketched [18], which serve as introductory materials to LWE cryptography. The correctness condition of LP is also analyzed in detail. A single-bit version of LP is presented without loss of generality, since an ℓ-bit version is merely ℓ instances of the single bit instance. All additions and multiplications are in $\mathbb{Z}_q$. Here, the subscript A is used to denote the client party (Alice), and B for the server (Bob).

- LP.$\mathsf{Setup}(\lambda)$: Upon input the security parameter λ, output parameters (n, χ_σ, q). Here, n is the lattice dimension, σ is the standard deviation for the discrete Gaussian distribution χ, and q is some modulus.

- LP.$\mathsf{KeyGen}(n, \chi_\sigma, q)$: Sample a uniform matrix $A \leftarrow \mathbb{Z}_q^{n \times n}$, and two vectors $\mathbf{s}_B, \mathbf{e}_B \leftarrow \chi_\sigma^{n \times 1}$. Let $\mathbf{c}_B = \mathbf{e}_B - A \cdot \mathbf{s}_B$. Output the public key $A, \mathbf{c}_B$, and secret key $\mathbf{s}_B$.

- LP.$\mathsf{Enc}(A, \mathbf{c}_A, m \in \{0,1\})$: For plaintext message $m \in \mathbb{Z}_2$, draw three error vectors $\mathbf{s}_A, \mathbf{e}_A \leftarrow \chi_\sigma^{n \times 1}, e_m \in \chi_\sigma$. Output two ciphertexts $\mathbf{c}_{A1}, c_{A2}$ where

$$\mathbf{c}_{A1} = \mathbf{s}_A^t A + \mathbf{e}_A^t \in \mathbb{Z}_q^{1 \times n} \tag{2.5}$$
$$c_{A2} = \mathbf{s}_A^t \mathbf{c}_B + e_m + m \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q. \tag{2.6}$$

- LP.$\mathsf{Dec}(\mathbf{c}_{A1}, c_{A2}, \mathbf{s}_B)$: Compute

$$m = \lfloor (\mathbf{c}_{A1}\mathbf{s}_B + c_{A2}) / \lfloor 2/q \rfloor \rceil. \tag{2.7}$$

Here, $\mathbf{s}_A^t$ is the transpose of $\mathbf{s}_A$. $\lfloor \cdot \rfloor$ and $\lfloor \cdot \rceil$ depict the flooring and rounding functions, respectively.

It is easily observed that in the decryption function, the inner product $\langle \mathbf{c}_{A1}, \mathbf{s}_B \rangle$ is the most computationally intensive step, and the possibility of computing such product approximately is explored later in this work.

Bounding the Parameters for LP

The decryption function $\mathbf{c}_{A1}\mathbf{s}_B + c_{A2}$ in Eq. (2.7) is internally computed as

$$\begin{aligned}&(\mathbf{s}_A^t A + \mathbf{e}_A^t)\mathbf{s}_B + \mathbf{s}_A^t \mathbf{e}_B - \mathbf{s}_A^t A \mathbf{s}_B + e_m + m \cdot \lfloor q/2 \rfloor \\ &= \mathbf{e}_A^t \mathbf{s}_B + \mathbf{s}_A^t \mathbf{e}_B + e_m + m \cdot \lfloor q/2 \rfloor. \end{aligned} \tag{2.8}$$

The condition for correct decryption is that the absolute value of the result $|\mathbf{e}_A^t \mathbf{s}_B + \mathbf{s}_A^t \mathbf{e}_B + e_m|$ (shorthand as $|e|$) is less than $\lfloor q/4 \rfloor$. When the error is within the range, it holds that $-\lfloor q/4 \rfloor < m \cdot \lfloor q/2 \rfloor + e < \lfloor q/4 \rfloor$ if $m = 0$, and otherwise if $m = 1$. Since both $\mathbf{e}_A^t \mathbf{s}_B$ and $\mathbf{s}_A^t \mathbf{e}_B$ are n-dimensional vector sums, e_m being a single Gaussian entry is considered small and often ignored. In addition, because the elements of vectors involved in the products are drawn independently from the Gaussian distribution, the error term can be rewritten as $|e| = |\langle \mathbf{e}, \mathbf{s} \rangle|$ where $\mathbf{e}, \mathbf{s} \leftarrow \chi_\sigma^{2n}$.

It is then trivial to use Lemma 2 to bound the size of the vector products $\langle \mathbf{e}, \mathbf{s} \rangle$. Let $T = \frac{\lfloor q/4 \rfloor}{\sigma \|\mathbf{e}\|}$, it follows that

$$\Pr[|\langle \mathbf{e}, \mathbf{s} \rangle| > \lfloor q/4 \rfloor] < 2 \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{\lfloor q/4 \rfloor}{\sigma \|\mathbf{e}\|} \right)^2 \right), \tag{2.9}$$

where $\|\mathbf{e}\|$ is bounded by Lemma 1. In [18], $\exp\left(-\frac{\lfloor q/4 \rfloor}{2\sigma\|\mathbf{e}\|}\right)$ is called the per-symbol error probability, also defined as the failure probability in some works [6]. Currently, the setting of this error probability is somewhat arbitrary. For example, in [18], the error probability is set to be 0.01, which means that there is a 2% chance that a bit encrypted under LP cannot be correctly decrypted. Hence, LP suggests to use error correction codes to improve the reliability of the scheme. Other works set a much lower bound on the probability, e.g., $\approx 2^{-40}$ in [6, 7], where no error correction codes are employed.

2.4.3 LWE-based Key Exchange: The Frodo Scheme

For generic LWE, q is fixed to be a power of two, and σ a custom-defined Gaussian-like probability density functions specified in Table 1 of [6].

The complete key exchange protocol proposed by [6] for generic LWE is outlined in Fig. 2.3. Here, a brief summary is presented on the protocol. First, Alice generates a seed $seed_A$ for the generation of the public key A. The generation algorithm Gen is implemented using the AES128 algorithm in [6], where $U(\{0,1\}^s)$ is a function that uniformly samples an s-bit ($s = 128$) integer. Alice then samples a secret vector and an error vector $S, E \in \mathbb{Z}_q^{n \times \overline{n}}$ where each element in the matrices is drawn from the distribution χ. After computing

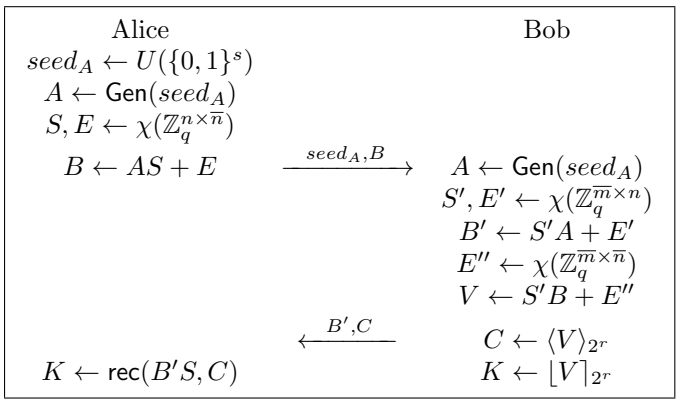

Figure 2.3: Key exchange from generic LWE proposed by [6].

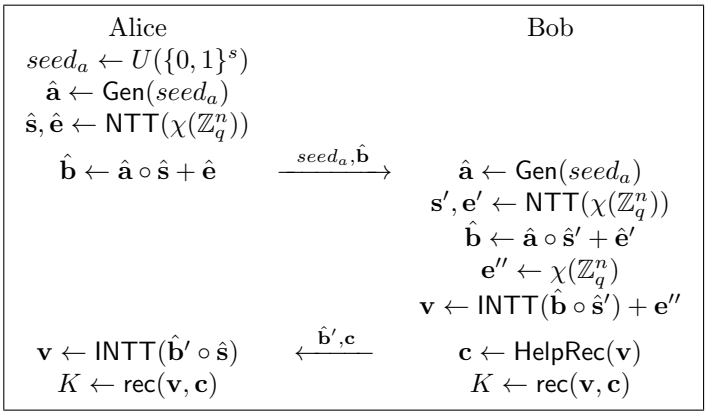

Figure 2.4: Key exchange from RLWE proposed by [7].

the result $B = AS + E$, B is sent to Bob. The security of the secret key S is guaranteed by the decisional LWE problem. Bob essentially repeats the same process, with an additional step of generating a reconciliation matrix $V = S'B + E''$, where $V \in \mathbb{Z}_q^{\overline{m} \times \overline{n}}$ contains secrets from both parties. V is then used to derive a shared key K for Alice and Bob through the reconciliation algorithms $\langle V \rangle_{2^r}$, rec and $\lfloor V \rceil_{2^r}$.

The important note here is that, the only heavy computation in this type of generic LWE construction is the step involving multiplication by the matrix A. Since A is an n-by-n matrix, multiplication by A requires at least n^2 multiplications, and n is a relatively large integer. $\overline{m}$ and $\overline{n}$ are relatively small integers, where the equality $r\overline{m}\,\overline{n} = \ell$ is required for deriving ℓ-bit key K ($n = 752, \ell = 256, r = 4$, and $\overline{m} = \overline{n} = 8$ for the recommended parameters set in [6]).

Thus, while Alice needs to perform two multiplications AS and $B'S$, AS would require $n^2\overline{n}$, while $B'S$ only requires $n\overline{m}$ multiplications. Furthermore, the aforementioned key-reconciliation algorithms are simple operations (multiply or modulo by a power of 2) on a very small matrix of dimension $\overline{m} \times \overline{n}$, and the performance impact of this step on the whole procedure can be safely ignored.

2.4.4 RLWE-based Key Exchange: The NewHope Scheme

The RLWE-based key exchange protocol is outlined in Fig. 2.4. The communication protocol is slightly modified for the ease of comparison with LWE, and it can be observed that the two protocols work almost identically. One distinct difference is that RLWE works on n-dimensional vectors that needs the special NTT and INTT treatment. Here, NTT is the number theoretic transform operator, and INTT is the inverse operator of NTT, i.e., it applies the inverse number theoretic transform. The concrete parameter instantiation and thereby detailed performance comparisons, are delayed to Section 3.2. Nevertheless, compared to LWE, RLWE has much smaller public key size ($\hat{\mathbf{a}} \in \mathbb{Z}_q^n$ compared to $A \in \mathbb{Z}_q^{n \times n}$). This simplifies all subsequent computations, making RLWE asymptotically faster.

2.5 Lattice-Based Homomorphic Cryptosystems

2.5.1 Abstract Homomorphic Properties

An HE scheme is a cryptographic scheme that satisfies a certain set of properties, resulting in the fact that the encryption (Enc) and decryption (Dec) functions of the scheme become homomorphisms. In addition to the detailed discussions on the specific algebraic constructions of HEs presented in the respective sections, the list of currently-known homomorphic properties are (abstractly) summarized here.

1. Homomorphic addition ($\boxplus$): for $x, y \in \mathbb{Z}$, $\mathsf{Dec}(\mathsf{Enc}(x) \boxplus \mathsf{Enc}(y)) = x + y$.
2. Homomorphic constant multiplication ($\boxdot$): for $x, y \in \mathbb{Z}$, $\mathsf{Dec}(\mathsf{Enc}(x) \boxdot y) = x \cdot y$.
3. Homomorphic multiplication ($\boxdot$): for $x, y \in \mathbb{Z}$, $\mathsf{Dec}(\mathsf{Enc}(x) \boxdot \mathsf{Enc}(y)) = x \cdot y$.

4. Packed homomorphic addition ($\boxplus$): for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathsf{Dec}(\mathsf{Enc}(\mathbf{x}) \boxplus \mathsf{Enc}(\mathbf{y})) = \mathbf{x} + \mathbf{y}$, where $+$ is matrix addition.

5. Packed homomorphic scalar multiplication (EMult): for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathsf{Dec}(\mathsf{EMult}(\mathsf{Enc}(\mathbf{x}), \mathbf{y})) = \mathbf{x} \circ \mathbf{y}$, where $\circ$ is the Hadamard product.

6. Packed homomorphic multiplication (EMult): for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathsf{Dec}(\mathsf{EMult}(\mathsf{Enc}(\mathbf{x}), \mathsf{Enc}(\mathbf{y}))) = \mathbf{x} \circ \mathbf{y}$.

7. Homomorphic rotation (rot): for $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{x} = [x_0, x_1, \cdots, x_{n-1}]$, $\mathsf{rot}([\mathbf{x}], k) = [x_k, x_{k+1}, \cdots, x_{n-1}, x_0, \cdots, x_{k-1}]$ for any $k \in \{0, \cdots, n-1\}$.

8. Homomorphic permutation (Perm): for $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{x} = [x_0, x_1, \cdots, x_{n-1}]$ and π be some permutation, $\mathsf{Perm}([\mathbf{x}], \pi) = [x_{\pi(0)}, x_{\pi(1)}, \cdots, x_{\pi(n-1)}]$.

In this work, an HE scheme that satisfies properties 1 and 2 is referred to as an additive homomorphic encryption (AHE) scheme, or a PHE scheme. If property 3 also holds for an AHE scheme, it is called a fully homomorphic encryption (FHE) scheme. AHE with properties 4, 5, 7, and 8 is known as packed AHE (PAHE). The HE scheme that bares all the above properties is identified as a packed FHE (PFHE) scheme.

In what follows, the Gentry-Sahai-Waters FHE scheme, the Brakerski-Fan-Vercauteren PFHE scheme, and the Paillier AHE scheme are described in detail.

2.5.2 The Gentry-Sahai-Waters (GSW) Cryptosystem

The GSW scheme is the third-generation FHE that offers polynomial-depth circuit evaluation without bootstrapping or relinearization [16], with its ring-variant provided in [17, 49]. Here, the slightly modified version of GSW in [49] is reviewed, along with its homomorphic properties. Similar to [49], G^{-1} and G are used for the respect $\mathsf{BitDecomp}$ and $\mathsf{BitDecom}^{-1}$ in the original work [16].

- GSW.$\mathsf{Setup}(\lambda, L)$: Upon input the security parameter λ and the homomorphic evaluation depth L, output parameters (n, χ_σ, q). Here, n is the lattice dimension, σ is the standard deviation for the discrete Gaussian distribution χ, and q is some modulus. Let $\ell = \lceil \lg q \rceil$ (here, $\lg q = \log_2 q$) and $N = n \cdot \ell$.

- GSW.$\mathsf{KeyGen}(n, \chi_\sigma, q)$: Sample a uniform matrix $\overline{A} \leftarrow \mathbb{Z}_q^{N \times (n-1)}$, and two vectors $\bar{\mathbf{s}}_B \leftarrow \chi_\sigma^{(n-1)\times 1}$ and $\mathbf{e}_B \leftarrow \chi_\sigma^{N \times 1}$. Let $\mathbf{s}_B = \begin{pmatrix} \bar{\mathbf{s}}_B \\ 1 \end{pmatrix} \in \mathbb{Z}_q^{n \times 1}$, and $\mathbf{c}_B = \mathbf{e}_B - \overline{A} \cdot \bar{\mathbf{s}}_B \in \mathbb{Z}_q^{N \times 1}$. Output the public key $A = (\overline{A}\ \mathbf{c}_B) \in \mathbb{Z}_q^{N \times n}$, and secret key $\mathbf{s}_B$.

- GSW.$\mathsf{Enc}(A, m \in \{0,1\})$: For a single-bit plaintext message m, compute $C = A + m \cdot G$ and output C.
- GSW.$\mathsf{Dec}(\mathbf{s}_B, C)$: Use Pen to extract the penultimate row of C as $\mathbf{c} = \mathsf{Pen}(C) \in \mathbb{Z}_q^{1\times n}$, and compute

$$m = \lfloor (\mathbf{c} \cdot \mathbf{s}_B)/2^{\ell-3} \rceil. \tag{2.10}$$

To be consistent with the notations used in this dissertation, $G \in \mathbb{Z}_q^{N\times n}$ is a transposed version of the one used in [49], and the bit decomposition function is defined as $G^{-1} : \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_q^{n\times N}$, i.e., it decomposes a number into a row vector of ℓ entries.

The two "gadgets" G and G^{-1} are important for controlling error growth in homomorphic evaluation. G^{-1} can be roughly thought as the column-wise bit decomposition that expands each $\lceil \lg q \rceil$-bit entry in some matrix $A \in \mathbb{Z}_q^{m\times n}$ into ℓ binary bits. G reverses the bit decomposition by multiplying the expanded column vector with a row vector $\mathbf{g} = [2^0, 2^1, \cdots, 2^{\ell-1}]$ to restore the original integers in A. These operations are purely to control the error growth in homomorphic evaluation and do not change the result of the dot products. The readers are referred to the original works for more details on the respect gadgets [16, 49].

Correctness for GSW

Note that the decryption extracts the penultimate row of C, denoted as $\mathsf{Pen}(C)$, due to the assumption that q is a prime. In this case, the penultimate row vector is as follows.

$$\mathbf{c} = [\overline{\mathbf{a}}_{N-2} \quad e_{N-2} - \overline{\mathbf{a}}_{N-2} \cdot \overline{\mathbf{s}}_B + 2^{\ell-2} \cdot m], \tag{2.11}$$

where $\overline{\mathbf{a}}_{N-2}$ is the $(N-2)$-th (penultimate) row of $\overline{A}$. Multiplying $\mathbf{c}$ with $\mathbf{s}_B = \begin{pmatrix} \overline{\mathbf{s}}_B \\ 1 \end{pmatrix}$ gives

$$\overline{\mathbf{a}}_{N-2} \cdot \overline{\mathbf{s}}_B + e_{N-2} - \overline{\mathbf{a}}_{N-2}\overline{\mathbf{s}} + 2^{\ell-2} \cdot m. \tag{2.12}$$

When $|e_{N-2}| < q/8$, $2^{\ell-2} \cdot m + e_{N-2}$ is between 0 and $q/8$ when $m = 0$, and $2^{\ell-3}$ to $2^{\ell-2}$ when $m = 1$. Diving the result by $2^{\ell-3} > q/8$ can correctly recover m. Note that if q is a power of 2, the last row is extracted, and the result is divided by $2^{\ell-2} = q/4$ to obtain the correct decryption. In this case, the maximum error tolerance goes to $|e_{N-2}| < q/4$.

Even though controlling errors is an essential step in developing efficient FHE schemes, existing theoretical studies are less interested in the exact failure

probability of decryption for such schemes [16, 49] (even in the case where concrete parameters are assigned, as in [17], no specific failure probability is reported). This is partly due to the fact that FHE requires extremely large LWE parameters, and the difference in parameters between a target decryption failure of 2^{-10} and 2^{-40} can be negligible.

Homomorphic Evaluation

For a freshly encrypted ciphertext, the large parameter setting generally guarantees perfect decryption (since discrete Gaussian has cut-off tail probability). The analysis becomes more difficult when homomorphic evaluation is taken into account. In GSW, the homomorphic addition and multiplication operations between two ciphertexts C_0 and C_1 encrypting messages m_0 and m_1 are defined as follows

$$m_0 + m_1 = \text{GSW.}\mathsf{Dec}(\mathbf{s}_B, C_0 + C_1) \tag{2.13}$$

$$m_0 \cdot m_1 = \text{GSW.}\mathsf{Dec}(\mathbf{s}_B, G^{-1}(C_1) \cdot C_0), \tag{2.14}$$

where $+$ and $\cdot$ are ordinary addition and multiplication defined over integers and matrices. The error growth for addition is trivial, where the error terms in Eq. (2.12) are added up. For homomorphic multiplication, the decryption error is given by

$$\mathbf{e}_{\text{mult}} = G^{-1}(C_1)\mathbf{e}_0 + m_0\mathbf{e}_1 \tag{2.15}$$

in [49], where $\mathbf{e}_0$ and $\mathbf{e}_1$ are the respect error vectors added to C_0 and C_1 during encryption. Using the asymmetric error growth property, a chain of left-associative multiplications

$$C = (((C_{d-2} \cdot G^{-1}(C_{d-1})) \cdots) \cdot G^{-1}(C_1)) \cdot C_0 \tag{2.16}$$

has quasi-linear error growth given by $\sum_{i=0}^{d-1}(G^{-1}(C_i) \cdot \mathbf{e}_i)$. This multiplication technique is referred to as chained multiplication.

For the above analysis, if it is assumed that all C_i are freshly encrypted ciphertexts with independent Gaussian distribution, the final error accumulated in d multiplications is $O(\sqrt{d})$, as the summations are between Gaussian distributions. However, one subtlety is that implementing G^{-1} as simple bit decomposition cannot ensure this property in all cases. For example, if the same ciphertexts are multiplied d times, the resulting error increases by $O(d)$. On the other hand, employing randomized G^{-1} as suggested in [49] can be costly in practice. In this work, a focus on homomorphic chained multiplications is placed, where the bit decomposition operation results in equivalent error analysis as a randomized G^{-1}. A more detailed explanation on error analysis for GSW is provided in Section 4.4.2.

2.5.3 The Brakerski-Fan-Vercauteren (BFV) Cryptosystem

The BFV cryptosystem is a standard ring LWE (RLWE)-based PFHE scheme from [15, 30]. The cryptosystem is implemented by Chen et al. in SEAL [50] and Polyakov et al. in PALISADE [43].

In terms of notations, similar to Gazelle [35], $[\mathbf{u}]$ refers to a ciphertext holding a plaintext vector $\mathbf{u}$, where $\mathbf{u} \in \mathbb{Z}_p^n$ for some plaintext modulus p and lattice dimension n. In BFV, owing to the Smart-Vercauteren packing technique [28], a ciphertext $[\mathbf{u}] \in \mathcal{R}_q^2$ is a set of two polynomials in some quotient ring $\mathcal{R}_q$ for a ciphertext modulus q. Here, a short overview on the basic properties and error behaviors of the private-key version of BFV is provided.

- BFV.$\mathsf{Setup}(\lambda, L)$: Upon input the security parameter λ and the homomorphic evaluation depth L, output parameters (n, χ_σ, q).
- BFV.$\mathsf{KeyGen}(n, \chi_\sigma, q)$: Sample and output the secret key polynomial $s \leftarrow \chi_\sigma^n$.
- BFV.$\mathsf{Enc}(s, m \in \mathcal{R}_q)$: For a plaintext polynomial m, sample an uniform random polynomial $a \leftarrow \mathcal{R}_q$ and an error polynomial $e_0 \leftarrow \chi_\sigma^n$. Compute and output

$$\begin{aligned} c_0 &= -a, \\ c_1 &= a \cdot s + \frac{q}{p} m + e_0. \end{aligned} \tag{2.17}$$

- BFV.$\mathsf{Dec}(s, c_0, c_1)$: Compute

$$\lfloor \frac{p}{q}(c_0 \cdot s + c_1) \rceil \tag{2.18}$$

The details on the correctness of BFV are left out as it is similar to LP and GSW. Note that $s, e_0 \in \chi_\sigma^n$ means that s, e_0 are polynomials whose coefficients are drawn from χ_σ (more precisely, $\mathbf{s}, \mathbf{e}_0 \leftarrow \chi_\sigma^n$, assuming $\mathbf{s}, \mathbf{e}_0$ are vectors containing the n coefficients in s and e_0).

The scheme remains correct when $e_0 \ll q/p$, and as long as $k \cdot e_0 \ll q/p$ for some constant $k \in \mathcal{R}$, scaling the ciphertext by k does not affect the correct decryption of the ciphertext $(k \cdot c_0, k \cdot c_1)$. This applies similarly to homomorphic addition, where $[\mathbf{u}] + [\mathbf{v}]$ decrypts correctly under the same key s when the underlying error polynomials satisfy $e_{0,u} + e_{0,v} \ll q/p$. Since the homomorphic multiplication operation is not used in this dissertation, the descriptions are omitted.

Rotating the Plaintext Slots in BFV

Since BFV is a packed HE, plaintext rotation and permutation operations are an important part of the cryptosystem. A (left) homomorphic rotation $\mathsf{rot}([\mathbf{u}], k)$ on a ciphertext $[\mathbf{u}]$ permutes the underlying plaintext vector $[u_0, u_1, \cdots, u_{n-1}]$ to $[u_k, u_{k+1}, \cdots, u_{n-1}, u_0, \cdots, u_{k-1}]$. The computations on ciphertext are a simple index swapping followed by a key-switching procedure. While the index swapping adds no additional errors to the ciphertext, some additive error components from switching the ciphertext are introduced, such that it is decryptable under the original secret key s. It is noted that this additive error, denoted as η_{rot} in [35], is independent from the original errors contained in the ciphertext $[\mathbf{u}]$.

More concretely, as demonstrated in [15], if a ciphertext $[\mathbf{u}]$ is rotated to $\mathsf{rot}([\mathbf{u}], k)$ for some integer k, switching $\mathsf{rot}([\mathbf{u}], k)$ back to some ciphertext $[\mathbf{v}] = \mathsf{SwitchKey}(\mathsf{rot}([\mathbf{u}], k), \mathcal{K})$ using the auxiliary key $\mathcal{K}$ results in the following decrypted equality

$$\mathsf{Dec}([\mathbf{v}]) = 2 \cdot (\mathsf{Decomp}_b([\mathbf{u}]) \cdot \mathbf{e}_{\mathcal{K}}) + \mathsf{Dec}([\mathbf{u}]), \tag{2.19}$$

where $\mathsf{Decomp}_b([\mathbf{u}]) \in \mathcal{R}^{\lceil \log_b(q) \rceil}$ is the component-wise decomposition function under base b (i.e., $\mathsf{Decomp}_b(x) = x_0 + x_1 b + x_2 b^2 + \cdots x_{\lceil \log_b(x) \rceil} b^{\lceil \log_b(x) \rceil}$). It is obvious that if b is taken to be 2, the term $2 \cdot \langle \mathsf{Decomp}_2([\mathbf{u}]), \mathbf{e}_{\mathcal{K}} \rangle$ is an inner product between a vector of 0/1 polynomial and a vector of freshly error polynomial ($\mathbf{e}_{\mathcal{K}} \in \mathcal{R}_q^{\lceil \log_2 q \rceil}$ are the errors in $\mathcal{K}$, instead of that in $[\mathbf{u}]$). More details on the exact computations involved in rot can be found in [29, 51, 52].

2.6 Factoring-Based Homomorphic Cryptosystems

2.6.1 The Paillier Cryptosystem

In deciding the concrete cipher to be utilized in SNBF, a number of cryptosystems are examined, and the finalized choice is on the PHE described by Pascal Paillier [40]. While the Paillier cryptosystem is a slightly older cryptographic construct based on the decision composite residuosity (DCR) assumption, its algebraic structure permits a (practically) infinite number of homomorphic additions (as long as the ciphertext can hold the result). This is different from the recent LWE-based constructions where the internal error size increases as the levels of homomorphic evaluation increase, which results in impractical parameter size. However, it is noted that the algorithmic construct of SNBF, as later described in Section 5.3, works on any PHE with additive homomorphism.

Paillier.Enc depicts the encryption function under Paillier, and Paillier.Dec for the decryption function. The basics of the Paillier cryptosystem is sketched as follows, where lcm denotes the least common multiple.

- Paillier.$\mathsf{KeyGen}(\lambda)$: Choose two large primes $p, q = \mathcal{O}(\lambda)$ and some element $g \in \mathbb{Z}^*_{\nu^2}$ with order $\nu\kappa$. Output the public key $e = (\nu, g)$ and private key $d = \mathsf{lcm}(p-1, q-1)$.
- Paillier.$\mathsf{Enc}(e, m \in \mathbb{Z}_\nu)$: Select a random integer $r \in \mathbb{Z}_\nu$, and compute

$$c = g^m \cdot r^\nu \bmod \nu^2. \tag{2.20}$$

  Output ciphertext c.

- Paillier.$\mathsf{Dec}(d, c)$: Recover m as

$$m = \frac{\mathrm{L}(c^d \bmod \nu^2)}{\mathrm{L}(g^d \bmod \nu^2)} \bmod \nu \tag{2.21}$$

  where $\mathrm{L}(u) = \frac{u-1}{\nu}$. Output plaintext m.

2.6.2 Properties of the Paillier Cryptosystem

Given the encryptions of two plaintext messages $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, the additive homomorphism in Paillier is as follows.

$$\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) \bmod \nu^2 = \mathsf{Enc}(m_1 + m_2) \bmod \nu. \tag{2.22}$$

Additive homomorphism also means that constant multiplication by plaintext variable can be computed. For two messages m_1 and m_2,

$$\mathsf{Enc}(m_1) \cdot g^{m_2} \bmod \nu^2 = \mathsf{Enc}(m_1 + m_2 \bmod \nu) \tag{2.23}$$

$$\mathsf{Enc}(m_1)^{m_2} \bmod \nu^2 = \mathsf{Enc}(m_1 \cdot m_2 \bmod \nu). \tag{2.24}$$

In this work, Eq. (2.23) is denoted as $\mathsf{Enc}(m_1) \boxplus m_2$ and Eq. (2.24) as $\mathsf{Enc}(m_1) \boxdot m_2$. The two properties are essential in constructing a secure keyword search scheme.

On the security note, the Paillier cryptosystem meets the indistinguishable under chosen plaintext attack (IND-CPA) security notion under the DCR assumption. Under a semi-honest server assumption, which generally holds true for established email servers, the IND-CPA property of Paillier provides enough security for SNBF.

2.6.3 Comparison to Lattice-based PHE

Recently, a stream of works has been conducted to improve the practicality of lattice-based homomorphic encryption adopted in secure learning problems, notably [53–55]. In addition to the usual homomorphic operations, lattice-based PHE offers single-instruction multi-data (SIMD) homomorphic operations being carried out. Specifically, the coefficient-wise addition between two ciphertexts $[\mathbf{u}], [\mathbf{v}]$ encrypting vectors $\mathbf{u}$ and $\mathbf{v}$ can be computed with only a single ciphertext addition $[\mathbf{u}] \boxplus [\mathbf{v}]$, which shares a similar idea with the proposed batching technique described in Section 5.3.3.

For applications such as secure email filtering in Chapter 5, Lattice-based schemes can be better-performing than factoring-based techniques when the length of the email is short. Roughly speaking, the same plaintext can be put into different ciphertext slots, and use a per-coordinate SIMD constant multiplication to achieve a single-instruction weight embedding. However, as discussed in Section 5.5, the average length of an email in a real-world dataset is around 200 words. Since each bit of the hashed version of each word (around 61-bit per word) needs to be embedded in a per-slot fashion, there exists no empty slots to replicate the same plaintext vectors. As a result, in the average case, the slot utilization in both Paillier (1024 to 2048 bits per ciphertext) and lattice-based methods (lattice dimension $n = 1024$ to $n = 2048$) are virtually the same. Therefore, while a lattice-based approach can be adopted if it is assumed that all incoming emails are very short, in general, lattice-based methods do not provide significant performance benefit from its packing ability.

On the other hand, lattice-based HE schemes have a major drawback in its error accumulation behavior, and an increase in the number of plaintext slots (and thus lattice dimension) makes the error growth much worse. It is known that every homomorphic operation induces some level of errors into the ciphertext of a lattice-based HE scheme, so the number of total homomorphic operations are limited by the amount of error margin provided by the ciphertext modulus q. For applications such as comparing a word with a large list (e.g., 10000 words in SNBF) of words, the amount of error margin required becomes the performance bottleneck, essentially for all lattice-based HE schemes. As a result, a 2048-bit ciphertext is used to encode a 1024-bit vector to perform SIMD matching using the Paillier scheme (a ciphertext expansion factor of 2); whereas, the smallest parameter set in a lattice-based ciphertext, for example in [55], require 1742 17-bit integers (an expansion factor of roughly 29, nearly 15x less efficient than the Paillier scheme). Moreover, such parameter instantiation is not likely to be able to evaluate the proposed filter, since weight embedding operations (homomorphic constant multiplication) significantly amplifies the ciphertext error. Hence, in order to maximize the practicality of SNBF, Paillier

is chosen as the foundation of a general-purpose email classification scheme.

Chapter 3

Hardware Accelerators for Lattice-based Key Exchange

3.1 Introduction

As the National Institute of Standards and Technology (NIST) initiated the discussions on the standardization of quantum-resistant public-key cipher suite [56], serious research efforts were devoted in evaluating and improving the performance of cryptographic constructions based on the (ring) learning with errors (RLWE/LWE) problem [6, 7, 20, 21]. Along with the well-established security reductions [12, 23], recent advances show that the performance of (R)LWE-based key exchange can be as efficient as traditional schemes such as RSA or elliptic curve cryptography [6, 7], making (R)LWE-based algorithms an attractive candidate for the age of post-quantum security.

Due to the emerging nature of the field [57], most existing hardware approaches on lattice cryptography focus on highly reconfigurable platforms such as embedded processors [58] or field programmable gate array (FPGA) devices [59, 60]. Such platforms are generally integrated with high-speed digital signal processing (DSP) hardware multipliers, and these multipliers are employed in almost all existing works. Hence, the architectural design for the computational unit in (R)LWE cryptography remains somewhat untouched.

Within the realm of LWE-based cryptography, the ring variant RLWE is generally considered much more efficient than generic LWE (also known as standard LWE). By working with the ideal lattices instantiated by elements of certain polynomial rings, RLWE reduces the n-by-n matrix that is required for generic LWE to yield a full-rank lattice down to an n-by-1 vector [23]. Furthermore, RLWE allows for efficient per-coordinate plaintext packing, which leads to efficient key exchange with only one set of polynomial coefficients [7].

The cost that comes with this performance improvement, however, is the security concern. At this moment, it is not known whether problems in ideal lattices are significantly easier to solve when compared to generic lattices. Unfortunately, these special lattices do present to be easier, where certain algorithms obtain a constant-factor time reduction by working exclusively with ideal lattices [61].

With all its theoretical advances, concrete parameter instantiations for LWE present to be problematic, especially in the case of RLWE. The general practice in instantiating LWE parameters is to ensure security against the best known attacks, rather than in the theoretically secure domain [6, 7, 18, 26]. In addition, the best known attacks are designed against general lattices, which do not contain further algebraic structure. This further complexes the security analysis for parameters instantiated for RLWE. RLWE operates in ideal lattices that are generated by fractional ideals in the ring of integers $\mathcal{O}_K$ of some number field K. By adopting this additional algebraic property, RLWE improves both the computational and storage efficiency by an order of $O(n)$. Nevertheless, as noted in [6, 26, 47], since ideal lattices are only a subset of general lattices, it is not known if RLWE is as secure as general LWE, especially in the suboptimal parameter regime. Unfortunately, RLWE does present to be slightly easier than general LWE, where certain algorithms obtain a constant factor reduction in runtime [61].

Although instantiating parameters for (R)LWE is still not trivial, a number of analyses have been proposed [6, 7, 18], and thereby the CPU [58] and FPGA implementations follow [47, 59, 60, 62, 63]. Notably, while almost all existing RLWE implementations [59, 60, 62] follow the parameter analysis in [18], the Lindner and Peikert analysis is for their proposed general LWE cryptosystem, instead of RLWE. As noted in [57], most existing works are straightforward hardware implementations based on CPU or FPGA, where designated digital signal processor (DSP) multiplier units are used. While DSP-based architectures are flexible, the resources on such platforms are generally not fully utilized (e.g., 18-bit multiplier for 14-bit multiplication in [47]). For resource constrained and resource demanding applications (e.g., embedded devices, or HSP processors), application-specific solutions are much needed. Moreover, since existing works either only focus on application- and cryptographic-layer algorithms or hardware-layer architectures, it is observed that some of the proposed techniques are not particularly useful when the entire system is considered. For example, in [44], the K-RED technique is proposed, where lazy reduction and modulus switching techniques are used to speed up the modular reduction operation in RLWE cryptography. Nevertheless, K-RED does not give significant speedup when well-pipelined hardware units are available, as the low-level platform has already provided (asymptotic) single-cycle modular

reduction capability. Therefore, in this chapter, and this dissertation as a whole, hardware designs are always proposed in cooperate with algorithmic designs, such that the overall system obtains better efficiency.

In this chapter, Filianore, an application-specific hardware accelerator for Frodo [6], the-state-of-the-art LWE key exchange scheme, and R-Filianore for NewHope [7], the latest RLWE-based key exchange scheme, are proposed. In particular, a cross-layer point of view is taken towards the hardware designs. Instead of simply adopting DSP units, algebraic properties in the cryptographic layers are exploited, and hardware-friendly algorithms are selected to simplify the underlying low-level architectures. Meanwhile, arithmetic units are carefully pipelined, such that complex algorithms, such as NTT, obtains single-cycle performance asymptotically. Through the experiment in Section 3.4, it is illustrated that compared to existing FPGA-based implementations, application-specific integrated circuit (ASIC) multiplier with optimized parameter selection gives us better throughputs by nearly 40x compared to the most recent art [63]. For Frodo-I, the slightly modified version of the recommended parameters suggested in [6], reduce the average energy consumption of the client-side LWE key exchange by roughly 6 times. For the proposed aggressive parameters, the energy consumption is further reduced by 3x (a total of 18x) to as low as 34.97 nJ. This performance is even better than R-Filianore, which averages around 35.04 nJ per key exchange. While generic LWE is still not quite as efficient as RLWE when memory bandwidths are concerned, compared to previous studies, the energy gap between generic LWE and RLWE is significantly reduced. The contributions of this chapter are summarized as follows.

- **Better Hardware Primitives for (R)LWE Key Exchange**: As [57] points out, previous design explorations for (R)LWE have been exclusively on embedded processors or FPGA. In this work, careful design explorations are provided for the computational units used in both generic and ring LWE. As later shown, even restricted to key exchange schemes, the design space can still be highly application-dependent.
- **Generic versus Ring LWE**: For its practical efficiency, RLWE is generally preferred over generic LWE for almost all cryptographic applications based on the LWE problem. In this chapter, an attempt is made to reduce the performance gap by adopting ASIC multiplier architectures that utilize the intrinsic algebraic simplicity and flexibility of generic LWE for post-quantum key exchange schemes. Through the experiment, it is demonstrated that generic LWE can be of practical use, and is more secure with the aforementioned worst-case hardness reductions.

Table 3.1: Parameter Instantiations

	q	n	dist.	σ	r	$\overline{n}$	$\overline{m}$
Frodo-Rec	2^{15}	752	χ_1	1.75	4	8	8
Frodo-I	2^{15}	752	χ_1	1.75	4	1	64
Frodo-II	2048	570	χ_2	1	1	1	256
NewHope	12289	1024	ψ_{16}	2.83	-	-	-

Table 3.2: Probability Mass Function for χ_1 and χ_2

	0	±1	±2	±3	±4	±5	±6
χ_1 (Frodo-I)	19304	14701	6490	1659	245	20	1
χ_2 (Frodo-II)	1570	990	248	24	1		

- **Instantiation of Asymmetric LWE Key Exchange**: As [6] suggests, LWE-based key exchange protocol is flexible, in the sense that the amount of client- and server-side computational powers can be highly unbalanced (asymmetric). However, no parameter instantiations or evaluations are available in the original work [6] and the recent implementation [63]. As an effort to lift the client-side computational burden, different sets of asymmetric parameters are proposed and compared, and it is shown that under such construction, generic LWE can be as energy-efficient as RLWE.

The rest of this chapter will be organized as follows. First, the proposed parameter instantiations for the respective schemes are presented in Section 3.2. Second, the hardware architecture is described in Section 3.3 and the synthesized results are compared in Section 3.4. Finally, the chapter is summarized in Section 3.5.

3.2 Proposed Parameter Instantiation

In this section, Frodo [6] is first instantiated under different parameter sets, and the parameter instantiation is then provided for NewHope [7] with discussions on its efficiency.

3.2.1 Parameter Instantiation for LWE

For the LWE-based key exchange protocol in Fig. 2.3, three sets of parameter instantiations are used, Frodo-Rec, Frodo-I, and Frodo-II. Here, a set of detailed analyses of the security and failure probability of each parameter set is presented.

Table 3.3: Security and Correctness Results for the Instantiated Parameter Sets

	F-I	F-II	F-Rec	NewHope
P.Q. Security [bit]	143	137	143	> 256
Correctness [lg]	-36.5	-43	-36.5	-61

In general, LWE cryptography is parameterized entirely by the parameters (n, q, σ). The key trade-off is between security and correctness, where larger n and σ give better security, but result in more failed key reconciliations. In contrast, larger q reduces the security level (in $\mathcal{O}(q^{1/n})$), but exponentially increases the success probability of key reconciliation. In addition, n is the main parameter that affects the computational efficiency of LWE. In this work, the concrete security analysis is based on the original work [6], and the bit security is calculated from $2^{0.265b}$ with 0.265 the best known constant for the post-quantum version of the BKZ algorithm [64]. Then, the least viable b which allows BKZ to yield a successful attack (details can be found on page 11–12 in [6]) is determined. For correctness, as suggested in Claim 3.2 in [6], as long as the (absolute) distance $|e|$ between each entry of $B'S$ and C is less than $q/2^{r+2}$, we have $\mathsf{rec}(B'S, C) = \lfloor V \rceil_{2^r}$, and the two parties derive the same key K. The continuous Gaussian is used to bound the tail probability of the discrete Gaussian, i.e.,

$$\Pr[|e| > q/(2^{r+2})] = 2 \cdot \Phi\left(\frac{q/(2^{r+2})}{\sigma_c}\right), \tag{3.1}$$

where Φ is the cumulative distribution function of a standard normal. The combined standard deviation σ_c describes the tail behavior of the Gaussian errors presented in the product $S'B$ and $B'S$, where two Gaussian variables of variance σ^2 are multiplied and added together. Combining with the E'' added for Bob, σ_c can be calculated as $\sigma_c^2 = 2n\sigma^4 + \sigma^2$.

The overall parameter instantiations, error distributions, and security and failure probability estimations are summarized in Table 3.1, 3.2, and 3.3, respectively. As noted, P.Q. security denotes post-quantum security. Frodo-Rec is the recommended parameter set in [6] with CPU implementation in mind, and Frodo-I is the unbalanced version of Frodo-Rec. In Frodo-I, $\overline{n}$ is set to 1, and the computation is essentially only $A\mathbf{s}$. This product translates precisely into 565,504 15-bit integer multiplications. Compared to Frodo-Rec, Frodo-I reduces the amount of computations by 8x for Alice, but causes an 8x increase for Bob.

Frodo-II is presented as the more aggressive parameter instance. It essentially puts as much computational burden to Bob as possible, and let Alice do the minimum amount of work. By decreasing the size of q, fewer bits

of keys are derived per entry in the matrix V, which requires (either Alice or) Bob to produce even more secret vectors to compensate. The benefit of having a smaller q is significant: smaller q means smaller n, which leads to smaller cumulative error causing less decryption failures, and eventually, even smaller q. For Frodo-II, Alice only needs to compute 324,900 11-bit multiplications, with provable 128-bit post-quantum security and improved failure probability. Note that since the exchanged key is only 128-bit post quantum, Frodo-II is as secure as Frodo-I and Frodo-Rec. Combining this parameter set with the specially designed hardware, it is shown that for Alice, the core matrix-vector multiplication for LWE and RLWE can be computed equally efficient.

Note that, for Alice who only needs to compute the matrix-vector product against one secret vector, a certain number of rows in A are actually multiplied by zero. As Table 3.2 indicates, according to the probability density function that generates these secret elements, nearly an average of 29.5% of such elements will be zeroes for χ_1 (D_4 in [6]) used in Frodo-I, and 38.3% for χ_2 (D_2 in [6]) in Frodo-II. By avoiding the actual need for multiplying and adding these zeroes, the number of multiplications can further be reduced by 29.5% to 38.3%, depending on the parameter instantiations. This approach has the drawback that a side-channel adversary may be able to obtain the exact number of zeroes in the secret vector. However, as long as the adversary does not know *which* entry is zero, the lattice dimension is retained, and the security of the scheme will not be tampered. In fact, some post-quantum LWE constructs set the number of zeroes in the secret vector to be a fixed and public value [65].

Lastly, note that Alice may not always be the client. If the client is a full-fledged desktop computer, and the server is an energy-efficient data center concurrently handling millions of connections, the roles can be reversed to ease the server-side computation, with potential benefit such as improved server response time.

3.2.2 Parameter Set for RLWE

The parameters provided in Table 3.1 for RLWE are directly taken from NewHope. In this section, some of the problems related to this parameter setting is discussed.

The first problem is that n has to be a power of 2. This limitation means that it becomes hard to balance between security and efficiency. As given, parameters in Table 3.1 provide a post-quantum security of 256 bits. However, since the scheme shares a 256-bit key, which only provides 128-bit post-quantum security for the subsequent communications, this level of security is clearly an overkill. Second, arithmetics for RLWE are also more difficult to implement on specialized hardware. After applying NTT, the numerical range for elements in

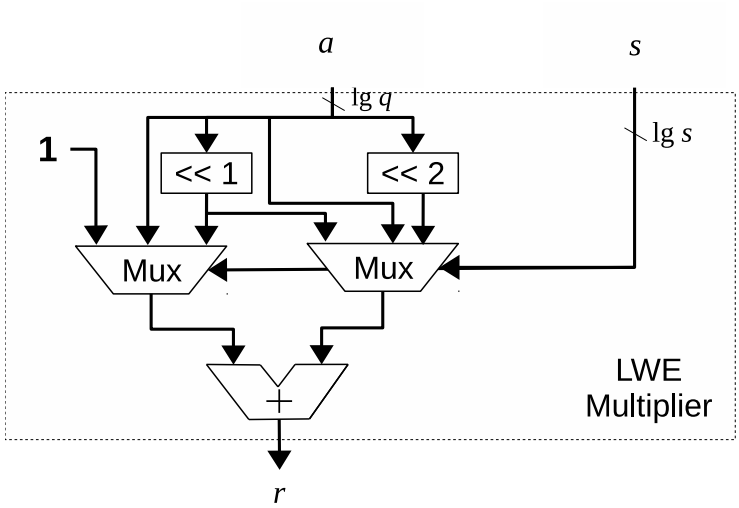

Figure 3.1: Proposed hardware multiplier for generic LWE.

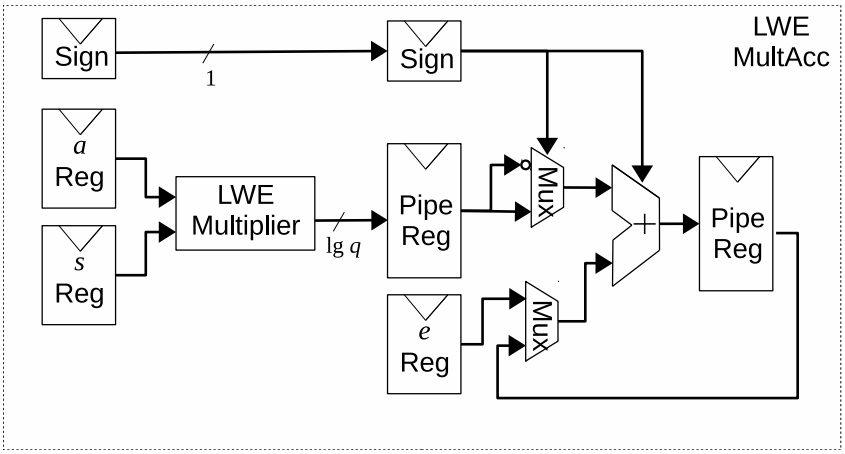

Figure 3.2: Proposed multiply-accumulate unit for generic LWE.

s and **e**, which were originally sampled from Gaussian distribution with small variance, become uniform across the 14-bit range. Combined with the need to perform Montgomery reduction (for q is not a power of 2), RLWE inevitably requires a much more complexed design.

3.3 Proposed Hardware Architecture

3.3.1 Filianore

The proposed architecture for the computing unit used in generic LWE-based key exchange is shown in Fig. 3.1. The insight is that all multiplications in

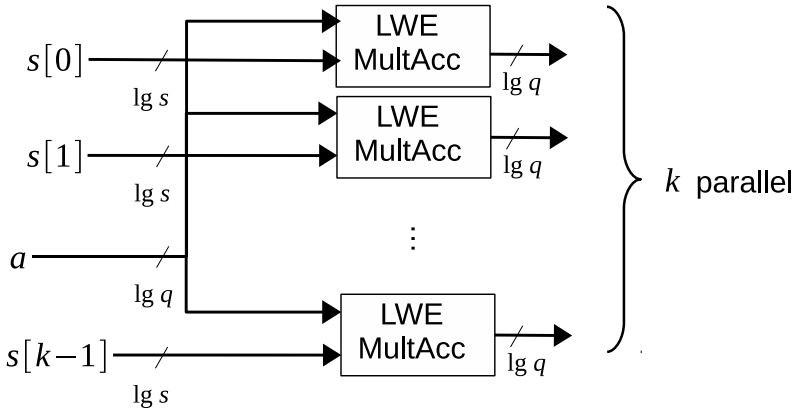

Figure 3.3: Parallelized multiply-accumulate units for generic LWE.

generic LWE are between some arbitrary $\lg q$-bit number (with q a power of 2), and a "small" number drawn from a Gaussian-like distribution. The range of this number is precisely defined in Table 3.2. In both χ_1 and χ_2, this "small" number does not exceed 6. Hence, the idea here is simple: multiplications by such small numbers can be done through shifts and additions. In addition, all multiplications by a number less than 7 can be done through exactly one addition. Thus, the "multiplier" unit in generic LWE can be implemented using a single adder with two multiplexers. Although the hardware complexity can be further reduced by the assumption that only χ_2 will be used (the maximum input from s is 4 in this case), this optimization loses compatibility of different parameter sets. Since having a slightly smaller multiplexer gives us marginal performance benefit, this optimization is not applied on the proposed design.

The complete system for computing AS and $B'S$ in Fig. 3.2 illustrates the simplicity of Filianore. An extra adder is used to perform the accumulation of matrix products, and it is also possible to use the system as is to perform plain integer additions such that $AS + E$ can both be computed on Filianore (by setting a multiplication by 1). The performance discussions are delayed into the next section where the design is synthesized, but this succinct design is clearly small in area, low in power, and highly parallelizable.

To parallelize the design, there are two approaches available. First, in Fig. 3.3, the a input is shared among all computational units, whereas the s is different for each. The benefit of this approach is that it minimizes memory bandwidth while retaining paralleled efficiency. Since the secret integer is only 4-bit wide ($\lg s = 3$ with an additional sign bit), even if the degree of parallelization is $k = 16$, the input bus will still be $64 + \lg q$ bits. The drawback and the reason that Alice cannot use this method is that it relies on the fact that the secret matrix S has dimension $n \times k$ where $k > 1$ strictly. For Alice

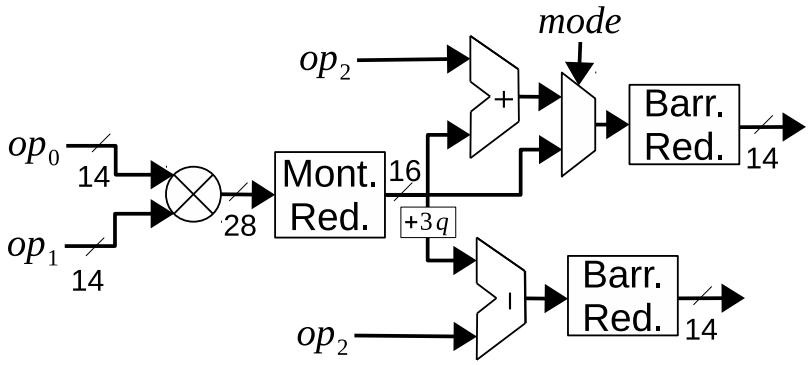

Figure 3.4: Proposed hardware accelerator for RLWE.

Table 3.4: Operational Modes for R-Filianore

	op_0	op_1	op_2
Mode i)	s	3186	0
Mode ii)	$a[j]$	ω	$a[j+t]$
Mode iii)	$\hat{a}$	$\hat{b}$	$\hat{e}$

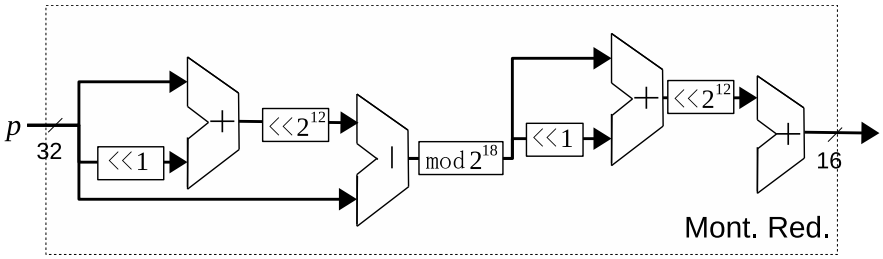

Figure 3.5: Hardware Montgomery reduction unit for a fixed $q = 12289$ and $R = 2^{18}$.

where the secret is a *vector*, i.e., $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$, there is no room for parallelization using different secret vectors. In such a case, the second approach can still be used, where the architecture in Fig. 3.2 is simply stamped k times for k-degree parallelization. This approach requires much larger memory bandwidth when k gets larger (320-bit when $k = 16$), and can be deployed if latency is extremely important. However, as [6] explains, for a typical HTTPS connection based on TLS, a single unit of Filianore fulfills the computational demand for Alice.

3.3.2 R-Filianore

Since all existing RLWE implementations avoid designing the core computational unit, to enable fair comparison, R-Filianore is also proposed, an

Algorithm 1 Cooley-Tukey butterfly from [60].

Require: $a[j], a[j+t], \omega$
1: $U \leftarrow a[j]$, $V \leftarrow a[j+t] \cdot \omega$
2: $a[j] \leftarrow U + V$
3: $a[j+t] \leftarrow U - V$

Algorithm 2 Montgomery reduction for $R = 2^{18}$ from [7].

Require: p
1: $u = (p \cdot 12287) \bmod 2^{18} \cdot 12289$
2: $p = (p + u)/2^{18}$

Algorithm 3 Short Barrett reduction from [7].

Require: p
1: $u = (p \cdot 5)/2^{16}$
2: $p = p - u$

accelerator for RLWE-based key exchange. The proposed architecture shares similarity with [60], but without a convenient DSP unit, the design of modular multiplication unit needs to be reconsidered.

The hardware accelerator proposed in this section is sketched in Fig. 3.4. The optimization techniques suggested in [7] are adopted, where all operations are carried out on unsigned integers in the Montgomery form. All operands that are read from or written to the memory are of 14-bit length. This is achieved through two consecutive reductions: the Montgomery reduction brings 32-bit integers down to 14 bits, and the short Barrett reduction is a light-weight unit that brings any 16-bit number down to 14 bits. This structure of a two-level reduction is distinct compared to the CPU implementation in [7], where additional reductions mean additional CPU cycles. This is also the reason why, instead of Gentleman-Sande (GS) butterfly, CT butterfly from [60] is utilized. The CPU-based implementation can have 32-bit additions and multiplications for free, but this is not the case for specialized hardware. Observe that in GS butterfly used by [7], the maximum input to the multiplier can be 69634, which is beyond 16 bits. Whereas, in R-Filianore, all input and output operands are strictly 14-bit.

This cryptographic processing unit is designed to operate under three types of arithmetic modes: i) converting the secrets into Montgomery form, ii) the Cooley-Tukey (CT) butterfly outlined in Alg. 1, and iii) the coordinate-wise multiplication and addition involved in computing $\hat{\mathbf{a}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$. Previous works have only focused on the design exploration for mode ii), for they either adopts a DSP

Table 3.5: Results of Hardware Implementations

	Delay [ns]	Area [NAND2]	Power [μW]
Filianore-I	2.8	1007	60.8
Filianore-II	2.5	639	48.1
Filianore-Rec	3.0	5822	385
R-Filianore	5.5	8229	323

that internally performs modular reduction [60], or works on general-purpose processors where i) and iii) are trivial.

Table 3.4 indicates the assignments for inputs op_0, op_1, op_2 under three different operational modes. For mode i), the 14-bit input operand s is multiplied by $3186 = (2^{18})^2 \bmod 12289$. This operation is to bring s into Montgomery domain by multiplying s with R^2, where $R = 2^{18}$, and Montgomery-reduce to sR^2 through the Montgomery reduction unit illustrated in Fig. 3.5. In mode ii), computations for Alg. 1 can be performed in a single pipelined cycle. These two steps essentially perform one NTT operation required in Fig. 2.4. Lastly, coordinate-wise multiplications and additions are computed in mode iii). By these three modes, all heavy computations in Fig. 2.4 can carried out on R-Filianore.

Finally, the efficiency of the modular reduction units is discussed. As Fig. 3.5 shows, these reductions are optimized to work on a specific q, namely $q = 3 \cdot 2^{12} + 1 = 12289$. Thus, multiplications are computed as consecutive additions, e.g., $p \cdot 12289 = (p + 2p) \cdot 2^{12} + p$. For the ease of presentation, the detailed architecture for short Barrett reduction which shares a similar structure to Fig. 3.5, is omitted. Since these units are optimized for a fixed modulus, this efficiency gain comes at the cost of flexibility.

3.4 Hardware Implementation and Comparison

The architectures shown in Fig. 3.3 and Fig. 3.4 are synthesized on a commercial library of a 65 nm low-power process node using logic-synthesis tool [66], and the power is analyzed by [67]. The synthesized results are summarized in Table 3.5. Here, F-I, II, and Rec are shorthands for instantiating the parameter sets Frodo-I/II/Rec on Filianore, and R-F means R-Filianore. The delay in Table 3.5 refers to circuit delay, which is different from the key-exchange latency later discussed.

To compare the actual performance, the energy consumption of (R-)Filianore over one set of key exchange is calculated. The latency for Frodo based on

Table 3.6: Latency, Energy Consumption and Memory Bandwidth Comparison for Alice

	F-I	F-II	F-Rec	R-F	[63]
L [ms] $(g = 1)$	1.718	1.178	1.715	0.1084	40.01
L [ms] $(g = g_{\max})$	1.211	0.7271	-	-	-
E [nJ] $(g = 1)$	104.5	56.68	660.1	35.04	-
E [nJ] $(g = g_{\max})$	73.64	34.97	-	-	-
Read [KiB]	1457	781.1	3358	89.23	-
Write [KiB]	1.530	1.081	1.425	61.87	-

Filianore is calculated as

$$L = g \cdot (n^2\overline{n} + n + n\overline{m}\,\overline{n}) \cdot 2.8. \tag{3.2}$$

The first term $n^2\overline{n} + n$ is the cost of $AS + E$, and $n\overline{m}\,\overline{n}$ for $B'S$. Note that in F-I and II, $\overline{n}$ is 1, and $S = \mathbf{s} \in \mathbb{Z}_q^n$. A constant g is defined to express the average number of zeroes in Alice's secret vector. $g_{\max}$ is taken to be 0.705 for Frodo-I and $g_{\max} = 0.617$ for Frodo-II. Computational latency for NewHope on R-Filianore is

$$L = \left(2n + 2 \cdot \frac{n}{2} \lg n + 2n + \frac{n}{2} \lg n\right) \cdot 5.5. \tag{3.3}$$

Here, the first $2n$ represents costs for transforming errors $\mathbf{s}, \mathbf{e}$ into Montgomery form. Two NTT operations need to be performed to map $\mathbf{s}, \mathbf{e}$ into $\hat{\mathbf{s}}, \hat{\mathbf{e}}$. Note that the transformation of public key $\hat{\mathbf{a}}$ is not needed, for a uniformly distributed vector is still uniformly distributed in the frequency domain in Montgomery form. The third $2n$ refers to computing $\hat{\mathbf{a}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$ and $\hat{\mathbf{b}'} \circ \hat{\mathbf{s}}$, and the final term is the cost of computing INTT.

The latency, memory consumption and energy consumption for (R-)Filianore are summarized in Table 3.6, and the results are compared with the most relevant work in [63]. The energy consumption is calculated using the equation $E = L \cdot P$, where L is the latency and P is the power in Table 3.5. Although LWE-based constructions are generally slower in delay, due to the simplicity of the proposed architecture, the energy efficiency is comparable between LWE and RLWE for F-I and II, and the area is 8 to 12x less. For the minimal parameter set F-II, it is observed that a generic LWE instance can be even slightly more energy-efficient than RLWE. On the other hand, in either case, under design-specific optimizations, the proposed designs are much more efficient than CPU-based approaches (10^3 to 10^4 energy reduction). Finally, while generic LWE consumes extensive memory-read bandwidth, since it only writes every n cycles, there are considerably less writes compared to read ($\mathcal{O}(n)$ for write). On

the other hand, since the NTT writes two outputs per butterfly, more writes are generated ($\mathcal{O}(n \log n)$).

The proposed multiplier achieves much better throughput than the existing FPGA-based design in [63], due to custom architecture and aggressive parameter instantiation. It is also noted that focusing on the multiplier architecture, Filianore is much more energy-efficient than existing studies on generic LWE. For example, the design in [47] is 28x slower in latency while having a similar power consumption compared to [59], on a per-bit encryption scale. Conversely, F-I and II have similar energy consumptions compared to R-Filianore. This performance gain comes from the faster clock and less power consumption of the proposed architectures. Therefore, although generic LWE is still less efficient overall when memory bandwidths are considered, the performance gap can be reduced by adopting Filianore-like multiplier architectures.

Lastly, it is noted that as explained in [6], the latency for key exchange has diminishing impact on the connection speed for a real-world workload. Since the proposed hardware implementation is almost as fast as the CPU implementation in [6], it is likely that a single-unit Filianore is practical enough to perform LWE-based key exchange for real-world HTTPS connections. Furthermore, instead of a standalone co-processor, the designed unit can easily be integrated into an embedded processor as an extension to its instruction set architecture, where existing data buses are generally adequate for the small amount of data per cycle required (15 to 20 bits).

3.5 Summary

In this chapter, (R-)Filianore, a set of multiplier designs for the (R)LWE-based key exchange algorithms are proposed. By adopting an ASIC approach, compared to existing architectures, the energy efficiency of the proposed approaches are greatly improved. In addition, by carefully exploiting the algebraic structure of generic LWE, it is demonstrated that under the asymmetric setting, client-side computations for LWE be as energy-efficienct as RLWE, with an extra area reduction of 8-12x. Furthermore, optimizing LWE-based cryptosystems using tight parameter estimation along with novel hardware designs becomes one of the main design perspectives for the subsequent chapters. Finally, it is concluded that given its flexible nature and solid security reduction, generic LWE remains competitive against RLWE for the post-quantum age.

Chapter 4

Approximate Architectures for Lattice Cryptography

4.1 Introduction

As mentioned in Chapter 3, existing hardware instantiations for generic and ring LWE, including (R-)Filianore, are generally only exploring how hardware architectures can be optimized to fulfill the computational demands of a set of established algorithms in the application and cryptographic layers, i.e., the hardware components serve solely as a tool to implement the algorithms.

This chapter explores how upper-layer algorithms can be modified to adopt to specialized lower-layer arithmetic units. Specifically, the applicability of approximate computing (AC) in LWE-based cryptosystems is explored. The key observation is that any (R)LWE-based cryptosystem is approximate by nature. For such ciphers, profiling and controlling error growth is essential in ensuring the correctness of the scheme, especially in instantiating concrete parameters. Theoretically, a cipher is considered correct as long as the decryption failure probability exhibits an exponentially decaying tail with respect to some parameters [18]. However, for a concrete parameter instantiation, a much more rigorous approach is required. In this chapter, existing LWE-based cryptosystems are modified such that numerical approximation can be adopted. As observed in [68], since LWE cryptography is erroneous by nature, the consequences of the proposed modifications are examined from both theoretical and empirical perspectives. As demonstrations, the well-known LP cryptosystem in [18] and the (R)LWE FHE scheme GSW by [16, 49] and [17] are instantiated with concrete parameters, and show that such cryptosystems almost always permit parameter approximation to some extent. Therefore, the proposed techniques are important optimizations for practical implementation of LWE cryptography,

especially in terms of the design and resource utilization of ASIC hardware.

In the experiment, the ASIC implementations of the exact and approximate versions of the underlying hardware are designed and compared. It is shown that in the LP case (resp., GSW case), it is possible to simultaneously achieve 1.75x (resp., 2.9x) speed increase, 4.17x (resp., 12.89x) area reduction, 2.29x (resp., 7.4x) power reduction and an average of 27.1% (resp., 62.5%) ciphertext size reduction in the decryption hardware without tempering the security and correctness requirement. The main contribution of this chapter is summarized as follows.

- **Approximate LWE Decryption**: A cryptosystem transformation is proposed such that existing LWE schemes can run on a set of approximate algorithms, instead of exact ones. As later shown in Section 4.5, additional noises can be infused into LP and GSW without breaking its security and correctness requirement. Hence, the practical aspect of the proposed technique is that the speed, area, power, and ciphertext-size gain come at (almost) zero cost. In addition, the proposed technique is general, since all LWE-based cryptosystems perform the decryption operation by computing products between two matrices.
- **Theoretical Bound for Correct Decryption**: While a similar analysis is performed in [68], their theoretical analysis is incomplete, in that no asymptotic failure probability bound was provided. In the case where extremely low failure probability is required (e.g., 2^{-40}), it is practically impossible to use Monte-Carlo (MC) simulations to determine the failure probability. In this chapter, the error profile of the state-of-art DRUM multiplier proposed in [69] is rigorously studied, and develop a theoretical bound for error growth under such approximate hardware. Fitted MC is also used to verify the higher range of the theoretical bound.
- **Approximating Homomorphic Encryptions**: The complex error growth in GSW is generally not well-studied [16, 17], where a simple $\mathcal{B}$ bound is used to instantiate the parameter set. In this chapter, it is shown that using the established error analysis, a concrete bound that is much smaller than the absolute worst-case assumption can be established. Combining with the approximate decryption technique, as illustrated in Section 4.5, the proposed approximation is especially useful for the homomorphic evaluations where, in general, have large unused error margins.
- **ASIC Multiplier for Standard LWE**: To fully benefit from AC, the DRUM multiplier is optimized to implement the computational unit of generic LWE. As pointed out in [57], ASIC implementations for LWE-

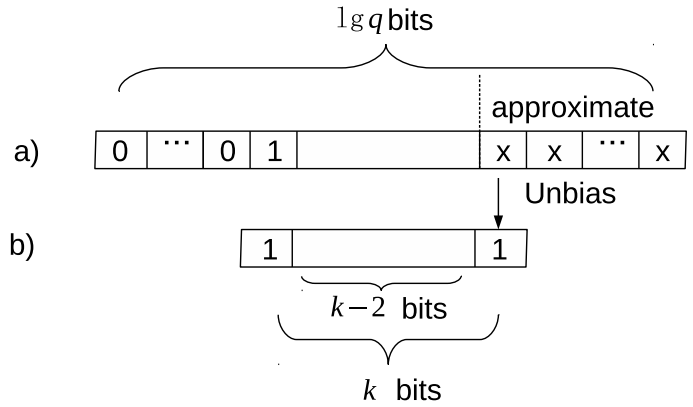

Figure 4.1: The DRUM [69] scheme approximates each operand in a) as b).

based cryptosystems are scarce, especially for generic LWE. Compared to the only existing work by Howe et al. [47], where an 18-bit DSP multiplier on an FPGA is used, the specialized multiplier further achieves 1.325x speed improvement, 2x area reduction and 1.84x power reduction in the decryption process of LP.

The rest of this chapter is organized as follows. First, the DRUM multiplier is portrayed in Section 4.2. Second, in Section 4.3, the error profile for generic LWE is analyzed, and investigate the effect of numerical approximation on the error growth. Third, existing ciphers are transformed into approximate ones and provide correctness analyses on their parameters in Sections 4.3 and 4.4, respectively. Forth, the approximate hardware architecture for both PKE and FHE with their respective implementation results are presented in Section 4.6. Finally, this chapter is concluded in Section 4.7.

4.2 Preliminaries: The DRUM Technique

In Chapter 4, a specific approximation scheme suggested in [69] called DRUM is used. The basic technique is outlined in Fig. 4.1. First, in a), a $\lceil \lg q \rceil$-bit integer (i.e., $\log_2 q$) is scanned by the algorithm to find the first occurrence of bit 1, called the leading one. The following $k-2$ bits are cut from the original integer, with the least significant bit (LSB) being fixed to 1. The remaining bits are thrown out, and the following product are taken between two approximated k-bit integers. Thus, the algorithm effectively compresses each $\lceil \lg q \rceil$-bit operand involved in the multiplication into a k-bit one, where the LSB is deterministically 1 to unbias the approximation error. Here, the

Algorithm 4 Encoding transformation function $\mathcal{E}$

Require: $x \in \mathbb{Z}_q, k \in \mathbb{Z} \cap [1, \lg q]$
1: $p = \mathsf{LOD}(x)$
2: $\tilde{x} = (x - 2^p) \gg (p + 1 - k)$
3: ↵ $p \in \mathbb{Z}_{\lceil \lg q \rceil}, \tilde{x} \in \mathbb{Z}_{2^{k-2}}$

Algorithm 5 Decoding transformation function $\mathcal{D}$

Require: $p \in \mathbb{Z}_{\lceil \lg q \rceil}, \tilde{x} \in \mathbb{Z}_{2^{k-2}}$
1: $y = (((\tilde{x} \ll 1) + 1) \ll (p + 1 - k)) + 2^p$
2: ↵ $y \in \mathbb{Z}_q$

relative approximation factor α is defined as $\alpha = \lceil \lg q \rceil - k$. A more detailed error analysis will be provided in Section 4.3.2.

After the approximation, only $k - 2$ bits of data need to be transferred, with additionally the position of the leading one, to the decryption side such that an approximated product can be carried out between the ciphertext and the secrete key. In Section 4.5.4, a leading-one encoding technique based on Huffman coding is developed, such that the decryption algorithm only needs an average of k bits to perform a successful decryption.

4.3 General Error Analysis

Before delving into the actual application of AC on LWE cryptography, this section first sets on a detailed analysis on the general decryption error characterization that lays the foundation for all subsequent analyses.

4.3.1 Formalizing DRUM

The error analysis is set off by formalizing the DRUM technique into a pair of encoding and decoding operations $\mathcal{E}$ and $\mathcal{D}$. First, as outlined in Alg. 4, the encoding algorithm $\mathcal{E}$ takes as input a $\lceil \lg q \rceil$-bit number x and the approximation factor k. $\mathcal{E}$ proceeds by locating the position p of the leading one in x ($\mathsf{LOD}(x)$), and assign the subsequent $(k-2)$-bit portion of x to $\tilde{x}$. The resulting position $p \in \mathbb{Z} \cap [0, \lceil \lg q \rceil - 1]$ and approximated $\tilde{x}$ are the output of the algorithm, which is used in the decoding algorithm. For decoding algorithm $\mathcal{D}$, $\tilde{x}$ is first shifted such that the least significant 1 for unbiasing can be added back. The approximated version of x, denoted as y, is then restored by adding 2^p, which is the leading one in x, to the shifted sum.

The algorithms described above are equivalent to the DRUM scheme described in Section 4.2, so the hardware implementation for the algorithms can readily be found in the original work [69]. A definition of $(\mathcal{E}, \mathcal{D})$ on vectors and matrices is also required, where the algorithm works component-wise. For vectors, we have $\boldsymbol{\pi}, \tilde{\mathbf{x}} = \mathcal{E}(\mathbf{x}, k)$, and $\mathbf{y} = \mathcal{D}(\boldsymbol{\pi}, \tilde{\mathbf{x}})$, where each $\tilde{x}_i \in \tilde{\mathbf{x}}$ is an approximation of $x_i \in \mathbf{x}$. The leading one positions of each x_i also form the vector $\boldsymbol{\pi} = [p_0, \cdots, p_{n-1}] \in \mathbb{Z}_{\lceil \lg q \rceil}^{n \times 1}$. For matrices, similarly, we have $\Pi, \tilde{X} = \mathcal{E}(X, k)$ and $Y = \mathcal{D}(\Pi, \tilde{X})$, where $X, \tilde{X}$, and Π all have the same dimensions.

4.3.2 Formulating the Approximation Error from DRUM

It can be observed that both decryption functions in LP and GSW consist of a simple inner product between two vectors. This generally applies to LWE-based cryptosystems. Hence, the error characteristics of integer-valued inner products in a finite field, i.e., $\langle \mathbf{a}, \mathbf{s} \rangle$ where $\mathbf{a}, \mathbf{s} \in \mathbb{Z}_q^n$, are studied, in the presence of arithmetic approximations.

Existing approximate computing techniques generally do not specifically target on working with uniformly random integers in a finite field [70]. As it turns out, if trivially applied, the error rate can easily exceed 100% in such case, rendering the approximation scheme useless. For example, consider the product $2^{15} \cdot (2^{15} + 1) \bmod 2^{16}$. Without LSB approximation, the product evaluates to 2^{15}; whereas, by ignoring the single LSB from $2^{15} + 1$, a product of 0 is obtained. The problem originates from the fact that, while relative to 2^{30}, an error of 2^{15} is "small" (less than 0.0001%), for integers modulo 2^{16}, the size of the error is the same as the correct product (100%). The development of profitable approximation techniques for multiplication on finite fields remain as an interesting open field of research.

Fortunately for most applications of LWE, the secret vector can be sampled from a Gaussian distribution with small variance (generally from the same distribution as the errors [18], or even ternary/binary distributions [65]) without violating the security requirement [18]. Hence, since approximating the secret vector has negligible return, the error behavior of the product $\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle$ is focused on, where $\tilde{\mathbf{a}} = \mathcal{D}(\mathcal{E}(\mathbf{a}, k))$, and $\mathbf{s}$ not approximated. An entry-wise analysis shows that the DRUM approximation can be thought as inducing an (correlated) additive error to each $a_i \in \mathbf{a}$, where $\tilde{a}_i = a_i + e_i$, where e_i is i-th

error due to approximation. Consequently, we have the equality

$$\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle = \sum_{i=0}^{n-1} (\tilde{a}_i s_i) = \sum_{i=0}^{n-1} (a_i s_i) + \sum_{i=0}^{n-1} (e_i s_i) \tag{4.1}$$

$$= \langle \mathbf{a}, \mathbf{s} \rangle + \langle \mathbf{e}_\alpha, \mathbf{s} \rangle \tag{4.2}$$

where the first term $\langle \mathbf{a}, \mathbf{s} \rangle$ is the exact inner product, and the second term $\langle \mathbf{e}_\alpha, \mathbf{s} \rangle$ is the inner product between the error due to DRUM approximation, $\mathcal{D}(\mathcal{E}(\cdot))$, and the secret vector $\mathbf{s}$. $\mathbf{e}_\alpha \leftarrow \phi^n$ is used to denote the fact that each entry $e_i \in \mathbf{e}_\alpha$ can be thought as drawn from some probability distribution ϕ parameterized by the approximation factor α, i.e., $e_i \sim \phi_\alpha$. This will be a helpful notion in the next section.

Without loss of generality, two assumptions are made on the product of $\langle \mathbf{a}, \mathbf{s} \rangle$: i) the product contains a simple sum of a message component and an error term (both integers $\bmod q$), and ii) the error term is the result of an inner product between two independent random vectors, one of which being the secret vector $\mathbf{s}$ sampled from some Gaussian distribution. For LWE cryptography, the above assumptions generally hold true, where the second assumption is proved by Lindner and Peikert in [18]. Using the assumptions, the analysis of DRUM on LWE cryptography can be simplified into the evaluation of one equation

$$\Pr[\|\langle \boldsymbol{\varepsilon}, \mathbf{s} \rangle\| \geq T\sigma\|\boldsymbol{\varepsilon}\|] < 2 \cdot \exp\left(-\frac{T^2}{2}\right), \tag{4.3}$$

where $\boldsymbol{\varepsilon}$ is the combined error term including the original error from LWE ($\mathbf{e}$ in Section 2.4.2) and the additive error from DRUM ($\mathbf{e}_\alpha$). Taking $T = \frac{\lfloor q/4 \rfloor}{\sigma\|\boldsymbol{\varepsilon}\|}$, we get precisely Eq. (2.9) (except for $\|\boldsymbol{\varepsilon}\|$), and all the LWE-based correctness analysis follows. By the above assumptions, the following equalities hold

$$\langle \boldsymbol{\varepsilon}, \mathbf{s} \rangle = \langle \mathbf{e}, \mathbf{s} \rangle + \langle \mathbf{e}_\alpha, \mathbf{s} \rangle = \langle \mathbf{e} + \mathbf{e}_\alpha, \mathbf{s} \rangle. \tag{4.4}$$

Note that if $\mathbf{e}$ and $\mathbf{e}_\alpha$ have different dimensions, as in the case of LP, one of the vectors needed to be padded with zeroes (the padding does not change the result of the inner products). As it turns out, the L_2 norm of each term in $\mathbf{e} + \mathbf{e}_\alpha$ can be computed separately, and the final bound is given by Pythagorean additivity as

$$\|\boldsymbol{\varepsilon}\| = \sqrt{\|\mathbf{e}\|^2 + \|\mathbf{e}_\alpha\|^2}. \tag{4.5}$$

Note that this theoretical derivation is not specific to DRUM (although it is required that the error is linearly separable from the product), and applies to

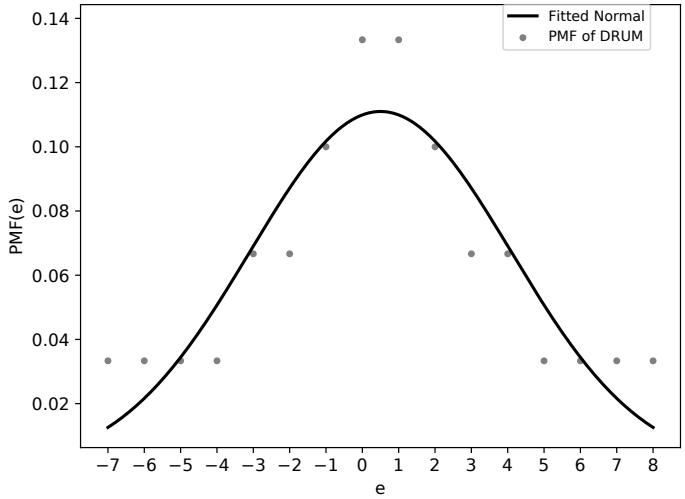

Figure 4.2: The probability mass function of DRUM approximation when $\alpha = 3$ and a normal distribution with the same mean and variance.

general approximation techniques. Therefore, the conclusion here is that, as long as the Euclidean norm of the approximation error $\|\mathbf{e}_\alpha\|$ is subexponential in n, a polynomial q guarantees asymptotic correctness of decryption. In the next section, the error analysis is completed by providing an upper bound on the approximation error $\boldsymbol{\varepsilon}$ in the case of DRUM.

4.3.3 Bounding the L_2 Norm of Approximation Error

Since a small q is essential to the efficiency of LWE-based cryptography, developing an asymptotically better and tighter error bound for DRUM is important. Figure 4.2 illustrates an example of the probability mass function (PMF) of the approximation error resulting from DRUM with an approximation factor $\alpha = \lceil \lg q \rceil - k = 3$. The benefit of DRUM is that the PMF is precisely defined, and its moments along with the expected values can be easily computed. Hence, the general results from studying norms of random matrices (also originates in the field of machine learning) are used to bound the L_2 norm of $\boldsymbol{\varepsilon} \in \phi^n$. For example, the Chernoff-Cramer inequality is stated as follows.

Theorem 1. (Adopted from Theorem D.1 in [7]) *Let ϕ be a distribution over $\mathbb{R}$ and let $X_0, \cdots, X_{n-1}$ be independent and identically distributed variables drawn from ϕ, with mean μ. Then, for any t such that $M_\phi(t) < \infty$, it holds that*

$$\Pr\left[\sum_{i=0}^{n-1} X_i \geq n\mu + \beta\right] \leq \exp(-\beta t + n \ln(M_\phi(t))), \tag{4.6}$$

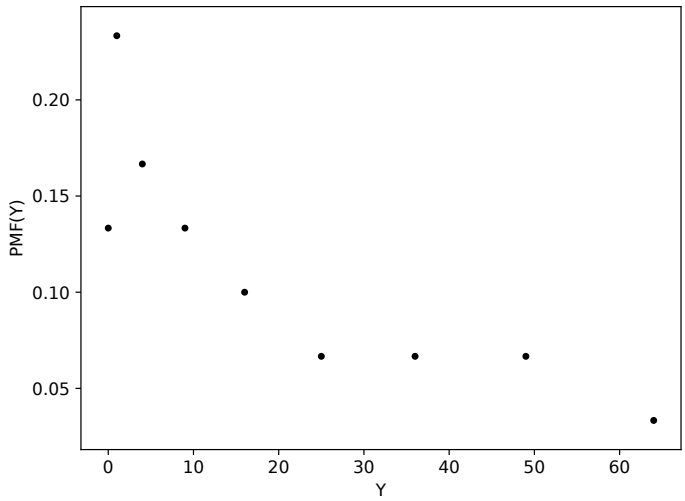

Figure 4.3: The probability mass function of Y for $\alpha = 3$.

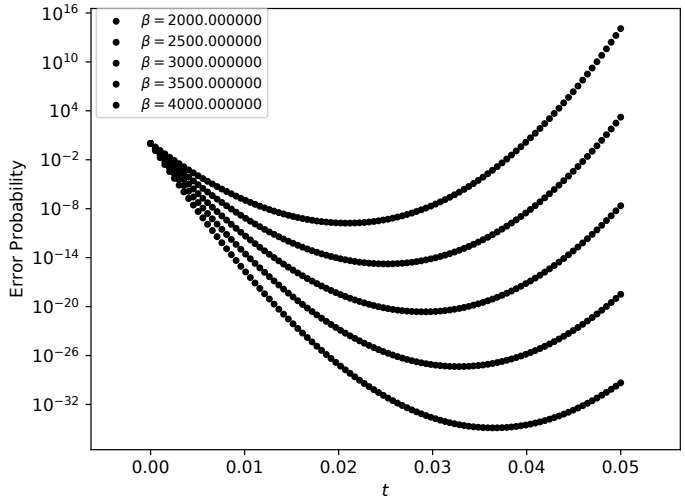

Figure 4.4: Convex bounds on the probability of the summed Y distribution evaluated at $\alpha = 3$ for different β with respect to t.

where $M_\phi(t)$ is the moment generating function defined as

$$M_\phi(t) := \mathbb{E}[\exp(t(\phi - \mathbb{E}[\phi]))] \tag{4.7}$$

For any bounded distribution as in the case of DRUM (and discrete Gaussian), the t-th moments for all $t \in \mathbb{R}$ are finite. Hence, the distribution has an exponentially decaying tail bound.

Since the L_2 (Euclidean) norm of the vector $\boldsymbol{\varepsilon} \leftarrow \phi_\alpha^n$ is required, a random variable $Y = X^2$ is built, where $X \sim \phi_\alpha$. The distribution of Y can be easily

enumerated from the distribution of X, where each Y is the squared value of X carrying the same probability weight $\phi(X)$. Here, an example of the evaluation results for the case where $\alpha = 3$ is given. The resulting distribution is characterized in Fig. 4.3. Compared to Fig. 4.2, Fig. 4.3 contains fewer points due to the fact that squaring is not an injective map. In addition, the PMF is not monotonic in the sense that the probability that $Y = 1$ is larger than $Y = 0$. This is because while $\Pr[X = 0] = \Pr[X = 1]$, $\Pr[Y = 1] = \Pr[X = -1] + \Pr[X = 1]$, making the probability of $Y = 1$ larger than that of $Y = 0$.

By computing the moment generating function of Y using Eq. (4.7) and applying the Chernoff-Cramer bound in Eq. (4.6), the choices on β can be optimized by incrementally changing the value of t and calculate the failure probability using Theorem 1. In the example where $\alpha = 3$ with $n = 256$ as in [18], the optimization process of β is sketched in Fig. 4.4. The smallest β such that there exists some t at which the failure probability is below a certain threshold (e.g., 2^{-40} is used in [18]) is desired. The calculated $n\mu + \beta \leq 5610$, and since we have $\sum_{i=0}^{n-1} X_i^2 = \|\mathbf{e}_\alpha\|_2^2 \leq 5610$, the resulting bound is $\|\mathbf{e}_\alpha\| \leq 75$. This example is actually the case for approximating the public key cryptosystem by LP [18], further analyses on the exact decryption failure probability in the presence of DRUM approximation are provided in Section 4.5.

4.4 Cryptosystem Transformation

In this section, a high-level description on the cryptosystem transformation on LP and GSW is provided.

4.4.1 Approximating LP

In general, the only modification needed is on the encryption (LP.Enc) and decryption (LP.Dec) functions, which is transformed as follows.

- $\widetilde{\text{LP}}.\mathsf{Enc}(A, \mathbf{c}_A, m, k)$: Using the same secret and errors $\mathbf{s}_A, \mathbf{e}_A, e_m$, output the ciphertext

$$\boldsymbol{\pi}, \mathbf{c}_{A1} = \mathcal{E}(\mathbf{s}_A^t A + \mathbf{e}_A^t, k) \tag{4.8}$$

$$c_{A2} = \mathbf{s}_A^t c_B + e_m + m \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q. \tag{4.9}$$

- $\widetilde{\text{LP}}.\mathsf{Dec}(\boldsymbol{\pi}, \mathbf{c}_{A1}, c_{A2}, \mathbf{s}_B)$:

$$m = \lfloor (\mathcal{D}(\mathbf{c}_{A1}, \boldsymbol{\pi}) \cdot \mathbf{s}_B + c_{A2}) / \lfloor 2/q \rfloor \rceil. \tag{4.10}$$

Correctness

For correctness, it needs to be proven that Eq. (4.10) can recover m as Eq. (2.7) does. From Section 2.4.2, it is shown that the decryption $\mathbf{c}_{A1}\mathbf{s}_B + c_{A2}$ results in Eq. (2.8). Adding the DRUM error term, we get

$$\mathbf{e}_A^t\mathbf{s}_B + \mathbf{e}_\alpha{}^t\mathbf{s}_B + \mathbf{s}_A^t\mathbf{e}_B + e_m + m \cdot \lfloor q/2 \rfloor. \tag{4.11}$$

In Section 2.4.2, the original errors are concatenated into a single vector $\mathbf{e}$. By padding $\mathbf{e}_\alpha$ into a $2n$ vector filled with zeroes in the form of $\mathbf{e}_\alpha^* = \begin{pmatrix} \mathbf{e}_\alpha \\ \mathbf{0} \end{pmatrix}$ where $\mathbf{0} \in \mathbb{Z}_q^n$, a single inner product is enough to express the error term as

$$\mathcal{D}(\mathbf{c}_{A1}, \boldsymbol{\pi}) \cdot \mathbf{s}_B + c_{A2} = \langle \mathbf{e} + \mathbf{e}_\alpha^*, \mathbf{s} \rangle + m\lfloor q/2 \rfloor, \tag{4.12}$$

and the analysis in Eq. (4.3) follows. The correctness of the scheme follows by instantiating an appropriate parameter set.

Security

The security of $\widetilde{\text{LP}}$ follows immediately from LP. As suggested, $\mathcal{E}$ and $\mathcal{D}$ compress the ciphertext $\mathbf{c}_1$ with loss. Thus, if an adversary is able to recover the secret or plaintext from $(\mathbf{c}_f, c_2)$, the adversary will be able to do the same thing for $(\mathbf{c}_1, c_2)$, with less effort. Thus, $\widetilde{\text{LP}}$ has at least the same, and presumably stronger security compared to LP (due to additional error).

4.4.2 Approximating GSW

For GSW, the encryption and decryption functions are modified as

- $\widetilde{\text{GSW}}.\mathsf{Enc}(A, m \in \{0,1\}, k)$: Notice that $A = (\overline{A} \ \mathbf{c}_B)$, output the ciphertext C, Π

$$C = A + m \cdot G \tag{4.13}$$

$$\Pi, \tilde{C} = \mathcal{E}(C, k) \tag{4.14}$$

$$\mathsf{Pen}(\tilde{C})_{n-1} = \mathsf{Pen}(C)_{n-1}. \tag{4.15}$$

- $\widetilde{\text{GSW}}.\mathsf{Dec}(\Pi, \tilde{C}, \mathbf{s}_B)$: Extract the penultimate row of $\tilde{C}$, $\tilde{\mathbf{c}} = \mathsf{Pen}(\tilde{C}) \in \mathbb{Z}_q^{1\times n}$, and compute

$$m = \lfloor (\tilde{\mathbf{c}} \cdot \mathbf{s}_B)/2^{\ell-3} \rceil. \tag{4.16}$$

Note that in Eq. (4.15), an exact version of $\mathsf{Pen}(C)$ is used instead of the approximate one. This is similar to Eq. (4.9) in $\widetilde{\mathrm{LP}}$, where an exact version of the ciphertext gives simpler error analysis and better error bound.

Since the security proof is trivial as in the case of LP, a formal discussion is omitted.

Correctness for Approximate Decryption

The error growth for homomorphic evaluations involving homomorphic multiplications is highly non-trivial, and is a part of the reason why existing works have been employing the simple $\mathcal{B}$-bounded approach as in [17]. For example, in [17], each ciphertext C is said to be $\mathcal{B}$-bounded if all the internal error terms $e \leq \mathcal{B}$. While Khedr et al. claimed that the error growth is $O(\sqrt{N})$ "in practice," they did not provide a theoretical analysis on the exact error growth for their homomorphic protocols. The parameters are instantiated with a trial-and-error approach without specific bounds being suggested (Section 6 in [17]). Unlike the LP scheme where simple error correcting codes are sufficient in the case of decryption failure [18], the result of an arbitrary set of homomorphic evaluations on some inputs are consequently arbitrary, and the asymptotic correctness of such scheme needs to be ensured (more concretely, e.g., a decryption failure probability of 2^{-40}).

As an attempt to further formalize the analysis for decryption failure probability in HE schemes, a general analysis on the error growth of a homomorphic multiplication between two arbitrary ciphertext C_0 and C_1 is first provided. It is assumed that C_0 and C_1 originally hold errors bounded by $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively. It is pointed out that the error resulted in the product $C_0 \cdot G^{-1}(C_1)$ can be expressed by Eq. (2.15). Since the second term $m_0\mathbf{e}_1$ can be trivially bounded by $\mathcal{B}_1$ (or goes to 0 when $m_0 = 0$), the main focus here is the errors generated in $G^{-1}(C_1) \cdot \mathbf{e}_0$. When $\mathbf{e}_0$ comes from a discrete Gaussian, the product bound follows directly from Lemma 2, where $\sigma = \sigma_{\mathbf{e}_0}$. However, when C_1 is the result ciphertext of some arbitrary homomorphic encryption, it becomes slightly harder to apply Lemma 2. The observation here is that, since $G^{-1}(C_1)$ is a 0-1 matrix, and by the leftover hash lemma [12], the distribution of $G^{-1}(C_1)$ is known to be extremely close (except for probability 2^{-n}) to uniform random. Therefore, the entries of $G^{-1}(C_1)$ can be viewed as a binomial distribution with $\nu = \kappa = 1$. For $p = 1/2$, the following inequality can be obtained.

$$M_{\text{binomial}}(t) = 1/2 + 1/2e^t \leq e^{t^2}, \forall t \in \mathbb{R}, \tag{4.17}$$

where e^{t^2} is the moment generating function of a centered Gaussian distribution with standard deviation $\sigma = \sqrt{1/2}$. Thus, Lemma 2 is applied in the reversed

Table 4.1: Parameter Instantiation for $\widetilde{\text{LP}}$ and $\widetilde{\text{GSW}}$

	q	n	σ	$\lg q$	α
$\widetilde{\text{LP}}$	4096	256	3.33	12	3
$\widetilde{\text{GSW}}$	0x7FFE0001	1024	3.98	31	20

order as

$$\Pr[\|\langle \mathbf{e}_0, G^{-1}(\mathbf{c}_1)\rangle\| \geq T\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\|] < 2 \cdot \exp\left(-\frac{T^2}{2}\right), \tag{4.18}$$

where $\mathbf{c}_1 = \mathsf{Pen}(C_1)$, and $\sigma_{\mathbf{c}_1} = \sqrt{1/2}$. Since $\mathbf{e}_0$ is $\mathcal{B}_0$-bounded, $\|\mathbf{e}_0\| = \mathcal{B}_0$ by definition, and the product bound can be easily calculated.

Consequently, after decryption, the total error becomes

$$\mathbf{e}_\alpha \cdot \mathbf{s}_B + G^{-1}(\mathbf{c}_1) \cdot \mathbf{e}_0 + m_0 \cdot e_1. \tag{4.19}$$

Although it is not entirely trivial to sum up the two error terms, by the triangular inequality, it holds that $\|\mathbf{e}_\alpha \cdot \mathbf{s}_B + G^{-1}(\mathbf{c}_1 \cdot \mathbf{e}_0)\| \leq \|\langle \mathbf{e}_0, G^{-1}(\mathbf{c}_1)\rangle\| + \|\langle \mathbf{e}_\alpha, \mathbf{s}_B\rangle\|$, and that $\|\langle \mathbf{e}_0, G^{-1}(\mathbf{c}_1)\rangle\| + \|\langle \mathbf{e}_\alpha, \mathbf{s}_B\rangle\| \leq T_s(\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\| + \sigma\|\mathbf{e}_\alpha\|)$ by overwhelming probability for some T_s. Setting $T_s = \frac{\lfloor q/8 \rfloor}{\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\| + \sigma\|\mathbf{e}_\alpha\|}$, and $\|e_H\| = \|\mathbf{e}_\alpha \cdot \mathbf{s}_B + G^{-1}(\mathbf{c}_1 \cdot \mathbf{e}_0)\|$, we get

$$\Pr[\|e_H\| \geq \lfloor q/8 \rfloor] < 2 \cdot \exp\left(\frac{-1}{2}\left(\frac{\lfloor q/8 \rfloor}{\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\| + \sigma\|\mathbf{e}_\alpha\|}\right)^2\right), \tag{4.20}$$

albeit not entirely tight. However, as demonstrated in Section 4.5.3, since the average error growth is much less than $q/8$, the above bound allows for a concrete analysis of asymptotic correctness under large approximation factors.

4.5 Parameter Instantiation

In this section, $\widetilde{\text{LP}}$ and $\widetilde{\text{GSW}}$ are instantiated using concrete parameters suggested in [18] and [17], respectively. Concrete DRUM approximation factor α is also instantiated for each of the schemes.

The parameters used in this work is summarized in Table 4.1. As discussed in [47, 71], q can be a power of 2 in generic LWE, which simplifies both the notation and the hardware circuitry. In $\widetilde{\text{LP}}$, the same n and σ in [18, 47] are used to guarantee a security level roughly equivalent to AES-128. Similarly, the parameters for $\widetilde{\text{GSW}}$ has a security level of 80-bit as discussed in [17]. For the DRUM multiplier in $\widetilde{\text{LP}}$ (resp., $\widetilde{\text{GSW}}$), a 9-bit (resp., 12-bit) multiplier is used instead of a full 12-bit (resp., 32-bit) one, and both theoretical and empirical methods are utilized to study the error growth under such approximations.

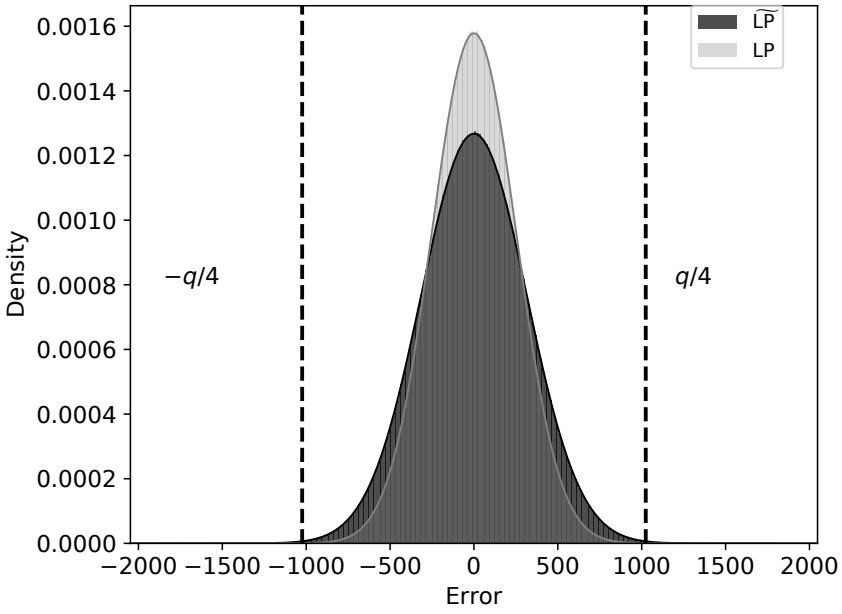

Figure 4.5: The decryption error distributions of LP and $\widetilde{\text{LP}}$ from 10M simulations.

4.5.1 Theoretical Bound for Errors in $\widetilde{\text{LP}}$

As described in Section 4.3.3, using $\alpha = 3$, an L_2 norm bound on the approximation error is acquired as $\|\mathbf{e}_\alpha\| \leq 75$. From Lemma 1, the L_2 bound on the original error vector can be calculated as $\|\mathbf{e}\| \leq c \cdot \sigma\sqrt{2n}$ where $n = 256$ and $c = 1.25$. The resulting bound is $\|\mathbf{e}\| \leq 94.2$. By Pythagorean additivity, the combined bound is 120.5. Taking $q/4 = 1024$ into Eq. (4.3), we have

$$\Pr[\|\langle \boldsymbol{\varepsilon}, \mathbf{s} \rangle\| \geq 1024] < 2 \cdot \exp\left(\frac{(1024/(120.5 \cdot 3.33))^2}{-2}\right), \tag{4.21}$$

and a concrete probability of 7.64% is obtained. At first glance, the parameter instantiation does not work, and a larger q is required such that the failure probability is safely below 2%. However, since the LP failure probability is relatively large, a simulated approach can be adopted to find a tighter bound on the failure probability. As shown in the next section, an $\alpha = 3$ assures the correctness of LP with a failure probability of 0.127%.

4.5.2 Empirical Bound for Errors in $\widetilde{\text{LP}}$

As suggested, the LP bound on failure probability is relatively large, where a simulated approach can guarantee reasonable confidence interval.

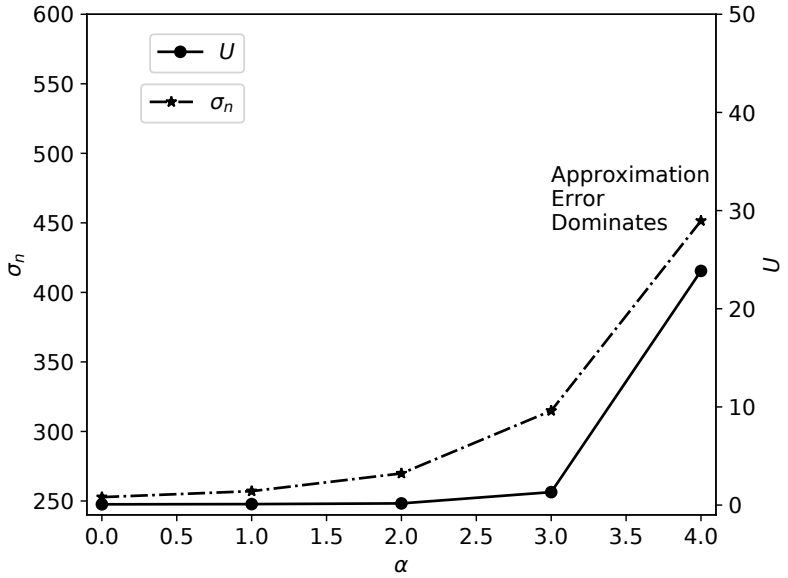

Figure 4.6: The upper bound U for different approximation parameter $\lg q - k$.

In this section, Monte-Carlo simulation is used to study the decryption failure probability of 10 million randomly produced LP as well as $\widetilde{\text{LP}}$ ciphertexts. Figure 4.5 demonstrates the simulated size of decryption errors for 10M LP samples as defined in Eq. (2.8), and 10M $\widetilde{\text{LP}}$ samples in Eq. (4.3). In Fig. 4.5, the original LP distribution is in light gray, and $\widetilde{\text{LP}}$ is in black. Each distribution is fitted with a normal distribution on top. σ_n denotes the standard deviation calculated for the fitted normal.

By examining the simulated results, it is found that for LP, 597 samples have decryption error that is greater than $|q/4|$ in 10M simulations, giving a decryption failure probability of 0.00597% with $\sigma_n = 252$ for the fitted normal. Meanwhile, for $\widetilde{\text{LP}}$, 12429 decryption failures occurred, translating to a 0.12429% of failure probability and a $\sigma_n = 314$. The failure probability can also be obtained by from σ_n of the fitted normal. For $\sigma_n = 252$, $1024 = 4.06\sigma_n$, and we have $\Phi(4.06) = 0.006\%$. On the other hand, for a $\sigma_n = 314$, 1024 is roughly $3.261\sigma_n$, giving a failure probability of 0.2%. The normal fittings agree with the Monte-Carlo simulation, and seems to serve as an upper bound. In both cases, the decryption failure probability is much less than what Eq. (4.21) derives.

Confidence Interval for Monte-Carlo Simulation

Since Monte-Carlo simulation is stochastic by design, the confidence internal of the above simulation process is inspected. One of the approaches (e.g., [72]) is

Table 4.2: Results of Experiment on Tightness for Various Bounds

	$\|\mathbf{e}\|$	$\|\mathbf{e}_\alpha\|$	LP Bound	$\widetilde{\text{LP}}$ Bound
Theoretical	94.2	74.9	0.971%	7.64%
Empirical	92.6	73.4	0.0767%	0.127%

to use a standard technique to derive the confidence interval from a series of small-sized simulations. In this work, the 10M simulations are divided into 10K simulations of some random variable Z, where Z is the number of decryption failures (i.e., $|\langle \mathbf{e}, \mathbf{s} \rangle| > q/4$) per 1000 decryption trials.

As described in [72], for a simulated mean of $\overline{y}$, some parameter $z_c > 0$, simulated standard deviation S_y, and total number of simulations N, the upper bound U on the mean is derived as

$$U = \overline{y} + z_c \frac{S_y}{\sqrt{N}}. \tag{4.22}$$

The $\overline{y}$ and S_y terms can be calculated from each set of 1000 simulations, and z_c is a parameter describing the confidence level in the interval. z_c is 7.2, which gives a confidence level of $1 - 2 \times 10^{-13}$. In other words, $\Pr[Y > U] < 2 \times 10^{-13}$ (roughly around 2^{-40}).

The result of U for different α is shown in Fig. 4.6. $\alpha = 0$ depicts the case of original LP decryption error. The observation here is that, when the approximate error $\mathbf{e}_\alpha$ is much smaller than the original discrete Gaussian $\mathbf{e}$, the total error increases gradually due to Pythagorean additivity of the errors, as illustrated in Eq. (4.5). Therefore, at first, the original LP error ($\sigma_n = 252$ for $\lg q - k = 0$) dominates the total error, until $\alpha = 3$. At this stage, σ_n has only increased to 314. Unfortunately, when $\alpha = 4$, the approximation error becomes dominant term, and a significant increase in $\sigma_n = 451$ and $U = 23.85$ is observed per 1000 decryption operations.

Based on the empirical results, it seems that the theoretical bound is not entirely tight. The tightness of each theoretical bound in Lemma 1, Lemma 2, and Theorem 1 is investigated. The comparison result is summarized in Table 4.2, where the empirical bounds on $\|\mathbf{e}\|$ and $\|\mathbf{e}_\alpha\|$ is based on the $7.2\sigma_n$ value calculated from 10M Monte-Carlo simulations. From Table 4.2, what is learned is that Lemma 2 is quite loose, since both Lemma 1 and Theorem 1 agrees with Monte-Carlo simulation extremely well.

Finally, the parameter instantiations conclude with $\alpha = 3$ (failure probability $\approx 0.127\%$) satisfies the per symbol error probability in LP [18] with overwhelming confidence, and these parameters are used for implementing the approximate hardware.

4.5.3 Bounding $\widetilde{\text{GSW}}$

Using a similar technique, parameters can be instantiated for $\widetilde{\text{GSW}}$. The subtle difficulty here is that without some assumptions on the underlying homomorphic evaluations, we end up with a worst-case analysis without fruitful approximations.

In this section, an analysis on the failure probability of GSW and $\widetilde{\text{GSW}}$ under the parameters instantiated by [17] is provided. Khedr et al. targets a multiplicative depth $d = 10$ for the proposed parameters, with $q =$ 0x7FFE0001 a Solinas prime, and $n = 1024$, as summarized in Table 4.1. Since a depth-10 circuit can evaluate 1024 multiplications, the worst-case and average-case error behaviors are characterized in the particular application of consecutive multiplications of 1024 ciphertexts.

First, the concrete error growth in the worst-case analysis is conducted, where Khedr et al. claims that the error growth is $O(\mathcal{B}2^d N)$. By examining Eq. (2.15), it is seen that if all N entries in $\mathbf{e}_0$ are sampled at the maximum bound $\mathcal{B}$, the resulting error is $\frac{1}{2}N\mathcal{B}$ in one multiplication evaluation, where the $\frac{1}{2}$ is due to the fact that the GSW ciphertexts are uniform random integers modulo q. Then, it is needed to ensure that all 2^d homomorphic multiplications are between ciphertexts that contain such (extremely unlikely) error distributions. The resulting error evaluates to

$$\mathcal{B} \cdot 2^{d-1} \cdot N = 16252928\mathcal{B}. \tag{4.23}$$

Since we have $\lfloor q/8 \rfloor = 268419072$, a maximum $\mathcal{B} \approx 16.5$ is calculated, which corresponds to an $\approx 4.18\sigma$. First, it is concluded that the parameter set is reasonable. Since the discrete Gaussian samplers have cut-off probability tail generally around 4 to 5 σ (e.g., [6]), having a $\mathcal{B}$ bound at 4.18 will ensure the correctness of the scheme in the absolute worst-case scenario. However, it is also noticed that this parameter set is clearly an overkill for 1024 multiplications, for the probability that a Gaussian sampler consecutively gives N outputs of values at $\geq 4\sigma$ can be practically ignored.

For a typical chain of 1024 homomorphic multiplications, on the other hand, the errors are much small in size. Using Lemma 1, the bound of the L_2 norm of $\mathbf{e}_0$ is calculated as

$$\Pr[\|\mathbf{e}_0\| \geq 3.94 \cdot \sqrt{1024} \cdot c] \leq c \cdot \exp\left(\frac{1-c^2}{2}\right), \tag{4.24}$$

and taking $c \approx 1.03$, we have $\Pr[\|\mathbf{e}_0\| \geq 722.9] \leq 2^{-40}$. With a L_2 bound on $\mathbf{e}_0$, Lemma 2 can be applied to bound the size of the product between $C^{-1}(\mathbf{c}_1)$ and

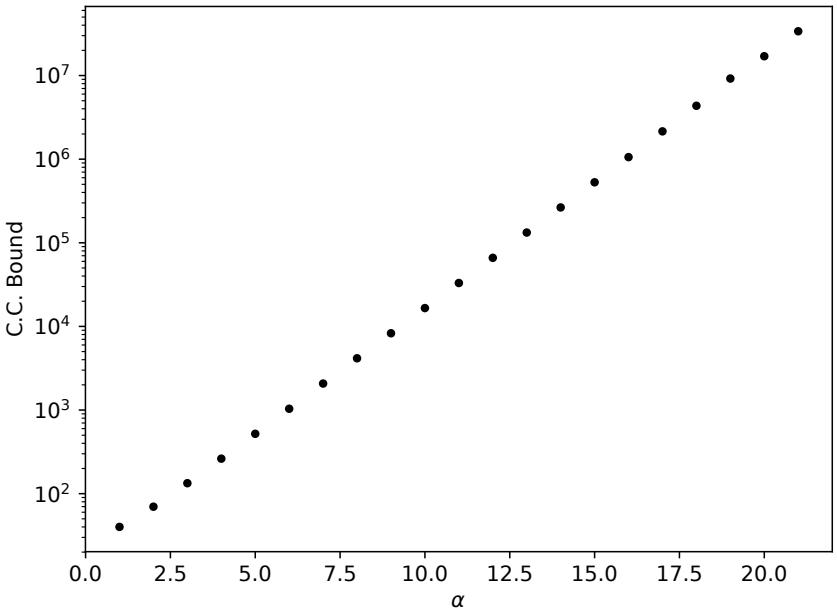

Figure 4.7: The calculated Chernoff-Cramer bound with respect to the approximation factor α in $\widetilde{\text{GSW}}$.

$\mathbf{e}_0$ using Eq. (4.18) as

$$\Pr\left[\|\langle \mathbf{e}_0, G^{-1}(\mathbf{c}_1)\rangle\| \geq T \cdot \sqrt{1/2} \cdot 722.9\right] < 2 \cdot \exp\left(-\frac{T^2}{2}\right). \tag{4.25}$$

To ensure a failure probability of 2^{-40}, the right-hand side of Eq. (4.25) gives a $T \approx 7.55$, and it is known that $\|\langle \mathbf{e}_0, G^{-1}(\mathbf{c}_1)\rangle\| \leq 3859.4$ almost certainly. For 2^d such multiplications, the final error growth will be less than $2^d \cdot 3859.4 \leq 3951939$ if the ciphertexts are correlated (e.g., in the case of calculating C^{1024}), and $\sqrt{2^d} \cdot 3859.4 \leq 123499$ in the independent case by the Pythagorean additivity as explained in [49]. Compared to $\lfloor q/8 \rfloor$, it is clear that the error margin is, by in large, wasted under typical homomorphic functions evaluated on some freshly encrypted inputs.

With the original error in hand, the Chernoff-Cramer bound can be applied on $\mathbf{e}_\alpha$ and $\mathbf{s}_B$ using the instantiated $\lceil \lg q \rceil$ and α. The relationship between α and the obtained bound, illustrated in Fig. 4.7, presents to be exponential. This is a natural result since the approximation error increases by 2x for each α step. With this broad range of error choices, the approximation errors can be limited when the original errors due to homomorphic encryption are large, and a large α can be chosen when the original errors are small. Using Eq. (4.20), the bound on the sum of errors can be calculated by setting the decryption failure

probability to a particular value, e.g., 2^{-40}

$$\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\| + \sigma\|\mathbf{e}_\alpha\| = \sqrt{\frac{(\lfloor q/8 \rfloor)^2}{\log(2^{-40}/2)\cdot(-2)}}, \tag{4.26}$$

and we get $\sigma_{\mathbf{c}_1}\|\mathbf{e}_0\| + \sigma\|\mathbf{e}_\alpha\| \leq 54025843$. When $\alpha = 19$, $\sigma\|\mathbf{e}_\alpha\| = 3626289$, the margin for the homomorphic evaluation errors is large. Both typical multiplications of 1024 ciphertexts mentioned earlier fit into the bound, and the conclusion here is that $\alpha = 19$ works for many, if not most, applications involving homomorphic evaluations. Even in the absolute worst-case, if a slightly smaller cut-off tail is used for the discrete Gaussian sample (for example, at $\sigma \approx 4$ as in D_1, D_2 and D_3 in [6]), the error margin can be as large as 12272926 for the approximation error, and an $\alpha = 14$ works in such case. In other words, given the large parameters used in a homomorphic encryption scheme and the exponential approximation error growth, it is always viable to find a certain level of profitable α that significantly affect the communication cost (as in ciphertext reduction) and computational cost.

4.5.4 Ciphertext Size Reduction

As discussed, if $\mathcal{E}$ and $\mathcal{D}$ are used as compression algorithms with loss, the compressed ciphertexts should contain less information, and thus be smaller in size, than the pristine version. In what follows, the resulting ciphertext size reductions for $\widetilde{\text{LP}}$ and $\widetilde{\text{GSW}}$ are analyzed under the instantiated parameter sets.

Ciphertext Reduction for $\widetilde{\text{LP}}$

The data needed to be handed to the decryption side contain each entry of $\mathbf{c}_{A1}$, a $k-2$ bit integer from DRUM approximation. p also needs to be sent, the position vector, for the dynamic range of DRUM. While trivially transmitting p can diminish the size reduction from $\lg q$ to $k-2$, p can be expressed through its Huffman coding representation. The Huffman coding can achieve optimal compression rate because the probability mass function of each p can be analytically defined, which is a direct result from the uniformity requirement of entries in $\mathbf{c}_{A1}$. For $\lg q = 12$, we know that $\Pr[p = \lg q - 1] = 1/2$, $\Pr[p = \lg q - 2] = 1/4$, and so on. After simple calculation, an average of $n \cdot (k - 2 + \rho)$ bits are transmitted to the decryption side for $\mathbf{c}_A$, where

$$\rho = 1 \times 1/2 + 2 \times 1/4 + 3 \times 1/8 + 3 \times 1/8 = 1.75, \tag{4.27}$$

reducing the size of a ciphertext by roughly $1 - ((7 + 1.75)/12) = 27.1\%$. Note that c_{A2} needs to be transmitted as is, but it is only one integer in $\mathbb{Z}_q$. The vast majority ($n = 256$) of integers are compressed by 27.1%.

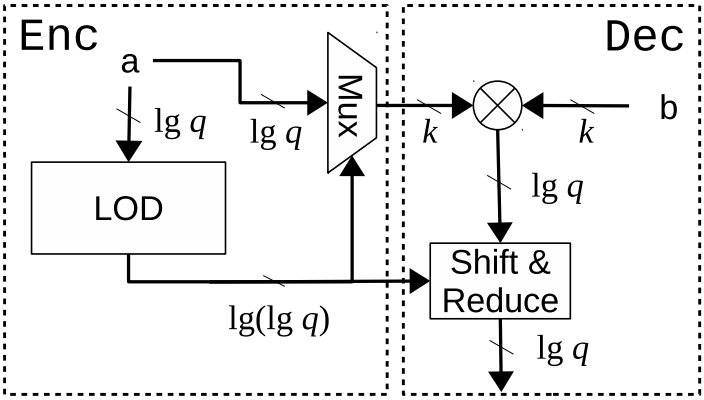

Figure 4.8: The proposed approximate hardware multiplier used in small-secret LWE decryption.

Ciphertext Reduction for $\widetilde{\text{GSW}}$

The analysis for $\widetilde{\text{GSW}}$ shares a similar approach with $\widetilde{\text{LP}}$, only with different parameters. For typical-case parameters ($\alpha = 19$), $k - 2 = 10$, and the final ciphertext is $10 + \rho$-bit, where $\rho = \sum_{i=1}^{19}(i \times (1/2^i)) + 19 \times (1/2^{19})$. Since $\rho < 2$, the ciphertext reduction in this case is $1 - ((10 + 2)/32) = 62.5\%$. Even in the worst case with $\alpha = 14$, we have $k - 2 = 15$, giving a ciphertext reduction of $1 - (17/32) = 46.9\%$.

4.6 Hardware Implementation

4.6.1 Hardware Architecture

To decrypt a standard LWE-based ciphertext, it is generally needed to compute an inner product between two n-dimensional vectors, where each multiplication involves two modulo q integers with q a power of 2. Therefore, the main component in such design is the multiplier, and there is minimal control logics due to the trivial dataflow. In this work, on the performance characteristics of three types of multiplier architectures are experimented: i) a plain full q-bit multiplier, ii) an approximated DRUM k-bit multiplier, and iii) an LWE-specific architecture accommodating the fact that LWE is small-secret.

Fig. 4.8 illustrates the architecture of the proposed approximate multiplier used in small-secret LWE. Half of the LOD logics for the secret operand can be squashed, and simplify the internal multiplier architecture as the output is only $\lg q$-bit instead of $2k$. Using the fact that q is a power of 2, simple bit-select operations function as modulo reductions in both the multiplier and the barrel shifter.

Table 4.3: Results of Hardware Implementations

	Delay (ns)	Area (NAND)	Power (μW)
Full 12-bit	7.0	2912	33.3
DRUM9_12	5.3	1413	26.5
This work	4.0	698	14.4
Full 31-bit	16.8	23869	162.6
DRUM12_31	7.0	4092	42.7
This work	5.7	1857	22.0

4.6.2 Experiment Setup and Results

The design in Fig. 4.8 is synthesized on a commercial 65 nm low-power process node using the standard logic-synthesis tool [66], and the power statistics are analyzed by [67].

The synthesized results for $\widetilde{\text{LP}}$ and $\widetilde{\text{GSW}}$ are summarized in Table 4.3. In both cases, all-round delay, area and power reductions are obtained. The actual cost for such reductions comes in the form of increased decryption failure probability. However, since the parameters are selected to meet the decryption failure probability bound, the cryptographic schemes remain correct. By further optimizing the hardware architecture for LWE, a total of 1.75x speed increase, 4.17x area reduction and 2.29x power reduction is achievable in the case of $\widetilde{\text{LP}}$. In the case of $\widetilde{\text{GSW}}$, a roughly 21.7x energy reduction is demonstrated when compared to a full 31-bit multiplier, along with the significant 12.89x area reduction.

4.7 Summary

This chapter discusses how approximate decryption can be used in LWE cryptography. A theoretical approach is developed to examine the decryption failure probability under complex error behavior, and novel empirical methods are used to verify the theoretical derivations. Using the instantiated parameters, the complexity of decryption logics as well as ciphertext size are reduced significantly without violating the correctness of the underlying ciphers.

Chapter 5

Homomorphic Bayesian Filter

5.1 Introduction

In the age of cloud computing, outsourcing at least some portion of private data to remote servers is an unavoidable option, even for organizations or institutions [73, 74]. However, in general, the outsourced data are limited to insensitive parts, such that the cloud efficiency is gained without compromising privacy [73,75]. The manual classification of sensitive versus insensitive data can be troublesome, for example, in the case of outsourcing emails. On one hand, since a public email server usually possesses a large amount of training email samples, high-quality email filter is available as part of the hosting service. On the other hand, the filtering process requires the server to inspect the plaintext content of user emails, violating privacy constraint for institutional or personal policies [31].

To resolve the efficiency-privacy dilemma, public-key encryption with keyword search (PEKS) [76] or partially or fully homomorphic encryption (PHE or FHE) [77, 78] are suggested as possible choice for constructing a secure email filter. Figure 5.1 outlines the general construction of a secure email filter. A server, here named Charlie, is presented the task to classify an email without inspecting the plaintext message from one client, Bob, to another, Alice, as spam or not. Without concerns for security, such filter can easily be constructed using a naïve Bayesian filter (NBF) [79].

A secure email filter can thus be implemented by a secure NBF, where the server blindly computes the NBF function using advanced cryptographic constructions. For example, PEKS-based techniques proposed for secure filtering are known to have practical performance [76]. However, since PEKS gains its efficiency at the cost of revealing the search results to the potentially malicious server, it violates the security requirement of an SNBF with public

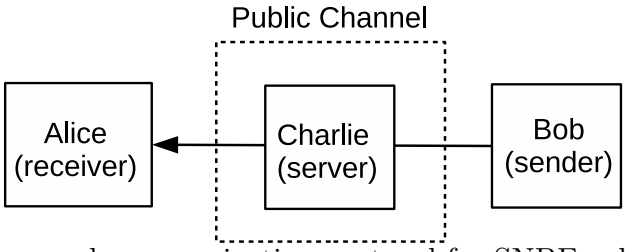

Figure 5.1: The general communication protocol for SNBF, where the server is considered as a part of the public channel.

filter [74, 80]

From the security perspective, FHE-based schemes such as [17, 45, 77, 81] are provably secure, where the underlying FHE scheme allows for the blind computation of arbitrary functions over a set of data. Unfortunately, FHE does not satisfy the performance level required in practical applications. For instance, two of the best performing FHE schemes in [17] and [82] compute a homomorphic multiplication in 300 ms and 17–22 ms, respectively, on modern CPUs. For a reasonable email filter as later shown in Section 5.5, this means that to classify a single word, both schemes need more than three hours of computation.

As an effort to design a secure NBF with provable security *and* practical performance, a custom filter based on a partially homomorphic encryption (PHE) scheme with designated hardware platform is proposed. Listed below are the key contributions of this chapter.

- **Cryptographic-Layer Protocol Design**: A novel secure filtering scheme is developed using an additive-homomorphic scheme. In this chapter, the Paillier scheme [40] is used as a demonstrative implementation. However, SNBF works with arbitrary PHE schemes, such as quantum-secure LWE-based constructs (e.g., [13]). Specific optimization techniques are developed for SNBF, e.g., the batch filtering technique where the wide space of Paillier ciphertext is used to batch up to 100 words per word filter. On the designed hardware platform, an average-length email can be classified in 0.5 s, well in the practical domain.

- **Application-Layer Weight Embedding**: The important observation in this chapter is how naïve Bayesian weights can be trivially embedded (as an integer) into the ciphertexts of any PHE scheme, instead of adopting the fixed-point representation in previous works [17, 45]. This weight-embedding technique is combined with an efficient homomorphic word matching function to achieve a practical performance for the

proposed homomorphic email filter. Through accuracy test on a realistic email dataset, it is confirmed that the integer-embedding technique makes negligible impact on classification recall, and the proposed SNBF scheme is a new type of machine learning technique optimized for secure applications.

- **Hardware-Layer Support for PHE**: The importance of hardware accelerator platforms in the application of homomorphic encryption is advocated. While the proposed CPU implementation is already better than existing works, the scalability and energy-efficiency of such implementation is constrained. In instantiating SNBF on specialized hardware, the energy consumption of filtering a set of ciphertext (which may contain one or more words) is reduced by 10^5 in magnitude compared to software implementation on CPU.

The rest of this chapter is outlined as follows. First, in Section 5.2, basics on the plaintext version of NBF is explained. Second, the algorithmic- and hardware-level specifications of SNBF are provided in Section 5.3 and 5.4, respectively. The accuracy and practical performance of SNBF is demonstrated in Section 5.5, and finally, this chapter is concluded in Section 5.6.

5.2 Preliminaries: Naïve Bayesian Filter

The naïve Bayesian filter is based on the simple application of Bayes' rule. First, in the training phase, the probability $\Pr(s|w)$ for each word w is computed as

$$\Pr(s|w) = \frac{\Pr(w|s)\Pr(s)}{\Pr(w)}. \tag{5.1}$$

Here, $\Pr(w|s)$ is the per-email probability of word w conditioned on the email being a spam, $\Pr(s)$ is the *a priori* email spam probability (it is assumed that $\Pr(s) = \Pr(h) = 0.5$ in the dataset. This value can be adjusted and made public by the server), and $\Pr(w)$ is the probability that the word occurs in all emails. The trained probability $\Pr(s|w)$ is that the email is a spam given that word w appears. In an SNBF scheme, the training phase is performed on a set of emails independent from the classifying ones, since the server is assumed to have access to its own set of emails. These emails are public to the server, so they are distinct from the private (to the user) emails that need to be classified. Thus, the trained filter is also assumed to be public to the server.

To apply the trained filter, let the incoming emails contain N words w_i in the set $W = \{w_0, \ldots, w_{N-1}\}$. Let p_{w_i} be the learned conditional probability

$\Pr(s|w_i)$, the final probability that the email is a spam can be computed as

$$\Pr(s_{\text{MAP}}) = \frac{\prod_{i=0}^{N-1} p_{w_i}}{\prod_{i=0}^{N-1} p_{w_i} + \prod_{i=0}^{N-1}(1 - p_{w_i})}, \tag{5.2}$$

where the numerator is the combined spam probabilities conditioned on each word simultaneously being in the email (as independent random events), and the denominator is the sum of the binary probabilities whether the email is a spam or a ham given each word in W. The computed $\Pr(s_{\text{MAP}})$ is then the maximum *a posteriori* probability for the email being a spam. As the consecutive multiplications result small fractional numbers, $\Pr(s_{\text{MAP}})$ is generally calculated by taking the logarithm of both sides of Eq. (5.2) and rearranging the terms as

$$\eta = \sum_{i=0}^{N-1} (\log p_{w_i} - \log(1 - p_{w_i})), \text{ and} \tag{5.3}$$

$$\Pr(s_{\text{MAP}}) = \frac{1}{1 + e^{-\eta}} \quad . \tag{5.4}$$

Equation (5.3) is the target function SNBF that needs to be implemented in a homomorphic manner. The subsequent notations are simplified using $\rho_i = \log p_{w_i} - \log(1 - p_{w_i})$, and the user should be able to compute $\eta = \sum_{i=0}^{N-1} \rho_i$ after the homomorphic classifications of the email.

5.3 Secure Email Filtering: the Algorithm

In this section, the notations used throughout this section is first listed in Section 5.3.1. Then, the plain SNBF structure including the communication model in Section 5.3.2 is presented. The optimization techniques for SNBF are discussed in Section 5.3.3.

5.3.1 Notations

Here, w is used for input word, and $\mathbf{w}$ is the binary-decomposed vector consisting of bits in w. The bit-width of w, i.e., length of the vector $\mathbf{w}$, is denoted as ℓ ($\ell = |\mathbf{w}|$). The pre-trained filter V is a collection of word-probability pair (v, ρ_v). Each v_i denotes a word where $|v_i| = \ell$, and $\rho_i = \log p_{v_i} - \log(1 - p_{v_i})$ is the Bayesian weight associated with v_i. The binary decomposition of w is $\mathbf{w} = \{w_{\ell-1}, \ldots, w_0\}$, where w_i means the i-th bit of w. Similarly, $\mathbf{v}_i = \{b_{\ell-1}, \ldots, b_0\}$ is used to specify the binary decomposition of v_i. Thus, all words in $\{v\}$ and w are assumed to have the same bit-width ℓ, which can be achieved by simple

Algorithm 6 The SNBF.Filter function.

Require: $\mathbf{c} \in \mathbb{Z}^{\ell}_{\nu^2}, \{(v, \rho_v)\}$
1: **for each** $v_i \in \{(v, \rho_v)\}$ **do**
2: $\quad c_{\text{match}_\text{i}} = 0$
3: $\quad$ **for each** $b_j \in \mathbf{v}_i$, $c_j \in \mathbf{c}$ and $\overline{c}_j \in \overline{\mathbf{c}}$ **do**
4: $\quad\quad c_{\text{xnor}} = ((c_j \boxdot b_j) \boxplus (\overline{c}_j \boxdot \overline{b}_j))$
5: $\quad\quad c_{\text{match}_\text{i}} = c_{\text{match}_\text{i}} \boxplus c_{\text{xnor}}$
6: $\quad r_i = c_{\text{match}_\text{i}} \boxdot \rho_i$
7: Let $\mathbf{r} = [r_{N-1}, \ldots, r_1, r_0]$
8: **return** $\mathbf{r}$

Algorithm 7 The SNBF.Decrypt function.

Require: $\mathbf{r} \in \mathbb{Z}^{N}_{\nu^2}$
1: $\rho = 0$
2: **for each** $r_i \in \mathbf{r}$ **do**
3: $\quad m_i = \text{Paillier.Dec}(r_i)$
4: $\quad$ **if** $\ell \mid m_i$ **then**
5: $\quad\quad \rho = m_i/\ell$
6: **return** ρ

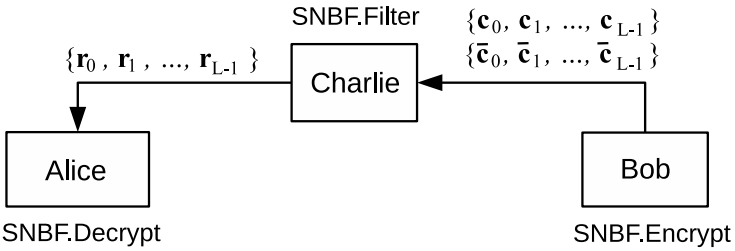

Figure 5.2: General communication protocol for SNBF.

padding or hashing. The total number of word-probability pairs in V is N, i.e., $N = |V|$. Additionally, $\lg x$ is the shorthand for $\log_2 x$.

5.3.2 Communication Protocol and Algorithmic Construction

The basic setup of the SNBF is outlined in Fig. 5.2. Different from conventional public-key cryptosystems, three parties are involved in the algorithm: Bob who sends the email, Charlie who filters the email, and Alice who receives the email.

In Fig. 5.2, Bob starts the protocol by using SNBF.Encrypt to encrypt his email into two vectors $\{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{L-1}\}$ and $\{\overline{\mathbf{c}}_0, \overline{\mathbf{c}}_1, \ldots, \overline{\mathbf{c}}_{L-1}\}$ using the public-key from Alice. Here, $(\mathbf{c}_i,\ \overline{\mathbf{c}}_i)$ is a tuple representing the encryptions of the i-th word in the mail, and the email contains a total of L words. The two sets of ciphertexts are sent to Charlie, where the classification SNBF.Filter is executed. Each encrypted word $(\mathbf{c}_i, \overline{\mathbf{c}}_i)$ is classified according to the trained filter list V, and the filtered results, $\{\mathbf{r}_i\}$, are transmitted to Alice. Alice applies SNBF.Decrypt to decrypt each $\mathbf{r}_i$ to obtain ρ_{w_i}. Having obtained the plaintext Bayesian weights, Eq. (5.3) is used to calculate $\Pr(s_{\text{MAP}})$. This section proceeds by giving detailed descriptions on each algorithm involved in the protocol. It is noted that all algorithms act on a single word, and if multiple words are to be processed, the algorithms need to be applied repetitively.

SNBF.Encrypt(w): takes as input a word w, and outputs the corresponding ciphertext $(\mathbf{c}_w, \overline{\mathbf{c}}_w)$. SNBF.Encrypt breaks w into a binary vector $\mathbf{w} = \{w_{\ell-1}, \ldots, w_0\}$, where $w_i \in \mathbb{Z}_2$ and its negation $\overline{w}_i$ are encrypted under Paillier.Enc. The output will be two vectors $\mathbf{c}_w = \{c_{\ell-1}, \ldots, c_0\}$ where, as mentioned, c_i = Paillier.Enc(w_i), and $\overline{\mathbf{c}}_w = \{\overline{c}_{\ell-1}, \ldots, \overline{c}_0\}$ where $\overline{c}_i$ = Paillier.Enc($\overline{w}_i$).

SNBF.Filter($\mathbf{c}_w, \overline{\mathbf{c}}_w, V$): takes as input the ciphertext $(\mathbf{c}_w, \overline{\mathbf{c}}_w)$, and the trained filter V. The outputs are the probability-weighted result $\mathbf{r}_w$. The detailed algorithm is outlined in Alg. 6. The overall idea of the algorithm is to perform a bit-wise homomorphic comparison between the inputs $\mathbf{c}_w$ and $\overline{\mathbf{c}}_w$, and words in the trained filter, i.e., $\{v\}$ in V. The filter function has a two-level nested loop structure. The outer loop performs a homomorphic comparison between the input ciphertext $(\mathbf{c}_w, \overline{\mathbf{c}}_w)$ and each filter word $v_i \in \{v\} \in V$. Inside each iteration, the inner loop from Lines 3 to 6 compares $(\mathbf{c}_w, \overline{\mathbf{c}}_w)$ and $\mathbf{v}_i$, the bit decomposed v_i, on a per-bit scale. The comparison result is embedded with the corresponding Bayesian probability weight ρ_i, delivering the i-th comparison result r_i. Each comparison result from one iteration of the outer loop gives one r_i, and all r_is' are collected into one vector $\mathbf{r}$, which is the output of the filter function. One obvious problem is that, any ρ_i is not likely to be an integer. However, the homomorphic operations permitted by PHE, in general, are on integers. The existing approach takes a simple approach, where real-valued weights are converted into fixed-point representations, and embedded into the matching results as large integers [17, 45]. By thoroughly examining the sensitivity of classification accuracy to weight approximation, the use of the integer rounding function is proposed to convert ρ_i into the nearest integer with almost no loss in classification accuracy (average difference of less than 1%). This property of NBF is demonstrated through experiments on real-world email dataset in Section 5.5.3.

SNBF.Decrypt: takes as input a ciphertext vector $\mathbf{r}$, outputs the Bayesian weight ρ if the input word w matches some word v in the filter, or 0 otherwise. The details of SNBF.Decrypt is sketched in Alg. 7. Algorithm 7 contains a loop that takes each result in $r_i \in \mathbf{r}$ and decrypts it using Paillier.Dec. The decrypted result m_i is first tested for word matching by dividing the bit length $\ell = |\mathbf{w}|$ into m_i. Later, it is shown that the division of ℓ from the decrypted m_i can simultaneously test word matching and recover the embedded integer probability weight ρ .

Correctness

To illustrate that SNBF properly functions as an email filter, a proof that the set of (probabilistic polynomial) algorithms (SNBF.Encrypt, SNBF.Filter, SNBF.Decrypt) works as claimed is needed. First, observe that the correctness proof for SNBF.Encrypt is simple: it follows trivially from the correctness of the Paillier scheme. Thus, it only remains to prove that after SNBF.Filter, SNBF.Decrypt can recover the Bayesian weights of matching words.

The following claim is used to construct a proof for the proposed scheme.

Claim 1. *Let ℓ be a prime, and $m_i = \rho_i \cdot k$ for some integer $k \in \{0, \cdots, \ell\}$. Assuming $\ell \nmid \rho_i$, if $\ell \mid m_i$, then $\rho_i = m_i/\ell$.*

Since the claim follows trivially from Euclid's Lemma, a formal proof is left out. Nonetheless, note that two constraints exist on ℓ, which requires careful parameter settings. In Section 5.5.3, it is shown that, in a typical parameter set, these assumptions are relatively easy to fulfill. The correctness for SNBF can then be guaranteed by the following Lemma.

Lemma 3. $\mathsf{SNBF.Decrypt}(\mathsf{SNBF.Filter}(w)) = \rho_w$ *if* $w \in V$, 0 *otherwise.*

Proof. First, note that Eq. (2.24) allows for direct computation of a single-bit XNOR gate using pure additive homomorphism. Line 4 in Alg. 6 is the exact computation to obtain c_{xnor} by evaluating

$$(c_j \boxdot b_j) \boxplus (\overline{c}_j \boxdot \overline{b}_j). \tag{5.5}$$

Since negation requires homomorphic multiplication, the user also needs to provide the negated bit $\overline{c}$ to compute homomorphic XNOR. The truth table of Eq. (5.5) is sketched in Table 5.1. After the XNOR comparison, the bit-wise result is summed up in Line 5. Therefore, if all bits result in matches (XNOR gives 1), the numbers (homomorphically) add up to ℓ. Meanwhile, if at least one bit is a mismatch, the encrypted sum will result in one of the encryptions in $\{\mathsf{Enc}(0), \mathsf{Enc}(1), \cdots, \mathsf{Enc}(\ell - 1)\}$. Finally on Line 7 of Alg. 6, the integer

Table 5.1: Truth Table for XNOR

c_j	$\overline{c}_j$	b_j	$\overline{b}_j$	$((c_j \boxdot b_j) \boxplus (\overline{c}_j \boxdot \overline{b}_j))$
$\mathsf{Enc}(0)$	$\mathsf{Enc}(1)$	0	1	$\mathsf{Enc}(1)$
$\mathsf{Enc}(0)$	$\mathsf{Enc}(1)$	1	0	$\mathsf{Enc}(0)$
$\mathsf{Enc}(1)$	$\mathsf{Enc}(0)$	0	1	$\mathsf{Enc}(0)$
$\mathsf{Enc}(1)$	$\mathsf{Enc}(0)$	1	0	$\mathsf{Enc}(1)$

Bayesian weight associated with v_i, ρ_i, is embedded into the result. Here, it is observed that if w and v_i are a match, $r_i = \mathsf{Enc}(\rho_i \cdot \ell)$. If there was a mismatch, $r_i = \mathsf{Enc}(\rho_i \cdot k)$ for some $k \in \{0, \cdots, \ell - 1\}$.

To prove that the above procedure correctly filters a word w, the decrypted result needs to be ρ_i if $w = v_i$, and 0 otherwise. In either case, when ℓ is divided into the decrypted result, it holds that $m_i = \mathsf{Dec}(r_i)$. Line 3 in Alg. 7 performs this division test on the decryption of r_i. By Claim 1, if r_i contains an encrypted match, $m_i = \rho_i \cdot \ell$, and we get ρ_i by dividing ℓ. If the division test fails, the output defaults to 0. □

Lastly, it is emphasized that if ℓ is not a prime, Claim 1 is not necessarily true, and the proof breaks apart.

Securing the Email Contents

In SNBF, the main security goal is to protect the content of the email, i.e., w from Bob to Alice. Hence, the encrypted words need to be protected against a potentially corrupted Charlie. Under such construction, SNBF.Encrypt and SNBF.Decrypt are secure, since they involve no trust on the server.

On the other hand, the security of SNBF.Filter follows directly from that of the Paillier scheme. More specifically, since all Charlie sees are Paillier ciphertexts, Charlie cannot distinguish any incoming ciphertext from uniform random. As a result, Charlie will also not be able to distinguish any of the intermediate ciphertexts from uniform random, as he only observes random ciphertexts being added and scaled, which also remain random.

5.3.3 The Batch Filtering Technique

Most emails contain more than one word. However, the proposed algorithm works on a per-word scale, which is clearly inefficient. The main insight here is that, the ciphertext space for the Paillier scheme is large (up to 1024-bit), but only a handful of bits are actually used by r_i. For example, since $r_i = \mathsf{Enc}(\ell \cdot \rho_i)$, under the parameter instantiation, $\ell \cdot \rho_i$ is at most a 10-bit integer. As a result,

large ciphertext space is wasted. Fortunately, a batch filtering approach can be used to pack more than one search bit into a ciphertext, while maintaining the correctness requirement of SNBF.

Consider the following simple example that illustrates the concept of the batching technique. In this example, a toy parameter set of $\ell = 2$ is used, and there are two input words $\mathbf{x}$ and $\mathbf{y}$ where $\mathbf{x} = \{x_1, x_0\}$ and $\mathbf{y} = \{y_1, y_0\}$. For $\mathsf{SNBF.Encrypt}$, instead of a bit-wise encryption, both words are encrypted for each bit position, i.e.,

$$c_0 = \mathsf{Enc}((x_0 \ll k) + y_0), \text{ where,} \tag{5.6}$$

$$k = \lceil \lg((\ell + 1) \cdot \max(\{\rho\})) \rceil. \tag{5.7}$$

Since $\ell = 2$, let us assume $\max(\{\rho\}) = 16$, which means $k = \lceil \lg((\ell + 1) \cdot \max(\{\rho\})) \rceil = 6$. The content of c_0, i.e., $\mathsf{Dec}(c_0)$, then becomes

$$\mathsf{Dec}(c_0) = 0\ 0\ 0\ 0\ 0\ x_0\ 0\ 0\ 0\ 0\ 0\ y_0$$
$$\mathsf{Dec}(\overline{c}_0) = 0\ 0\ 0\ 0\ 0\ \overline{x_0}\ 0\ 0\ 0\ 0\ 0\ \overline{y_0}$$

Now consider a filter with only one word $\mathbf{v} = \{b_1, b_0\}$, where $b_1 = 1$ and $b_0 = 0$. Without modifying Alg. 6, line 4 processes as

$$c_{\text{xnor0}} = ((c_0 \boxdot 0) \boxplus (\overline{c}_0 \boxdot 1)). \tag{5.8}$$

Under Paillier, $c_0 \boxdot 0 = c_0^0 = 1$, and $1 \boxplus (\overline{c}_0 \boxdot 1) = 1 \cdot \overline{c}_0^1 = \overline{c}_0$. Hence, for zeroth bit, $c_{\text{xnor0}} = \overline{c}_0$. Similarly, for first bit, $c_{\text{xnor1}} = c_1$. The match bit can thus be computed as $c_{\text{match}} = c_{\text{xnor1}} \boxplus c_{\text{xnor0}}$. In plaintext space, calculating c_{match} translates to the following procedure

$$\begin{array}{r} 0\ 0\ 0\ 0\ 0\ \overline{x_0}\ 0\ 0\ 0\ 0\ 0\ \overline{y_0} \\ +\ 0\ 0\ 0\ 0\ 0\ x_1\ 0\ 0\ 0\ 0\ 0\ y_1 \end{array}$$

Instantiating $\mathbf{x}$ and $\mathbf{y}$ as $\{x_1 = 1, x_0 = 0\}$, and $\{y_1 = 0, y_0 = 0\}$, respectively, we get

$$\begin{array}{r} 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ \mathbf{1} \\ +\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ \mathbf{0} \\ =\ 0\ 0\ 0\ 0\ \mathbf{1}\ \mathbf{0}\ 0\ 0\ 0\ 0\ 0\ \mathbf{1} \end{array}$$

where $\max(\{\rho\}) = 16$. In order to satisfy the constraint $\ell \nmid \rho$ in Claim 1, ρ_v is set to be 15. Note that when $\ell > \rho$, as in the real parameter instantiation, any $\rho \leq 16$ works. Following Line 6 in Alg. 6, the Bayesian weight is embedded into the matching result as

$$\mathbf{1\ 1\ 1\ 1\ 1\ 0}\ 0\ \mathbf{0\ 1\ 1\ 1\ 1} \tag{5.9}$$

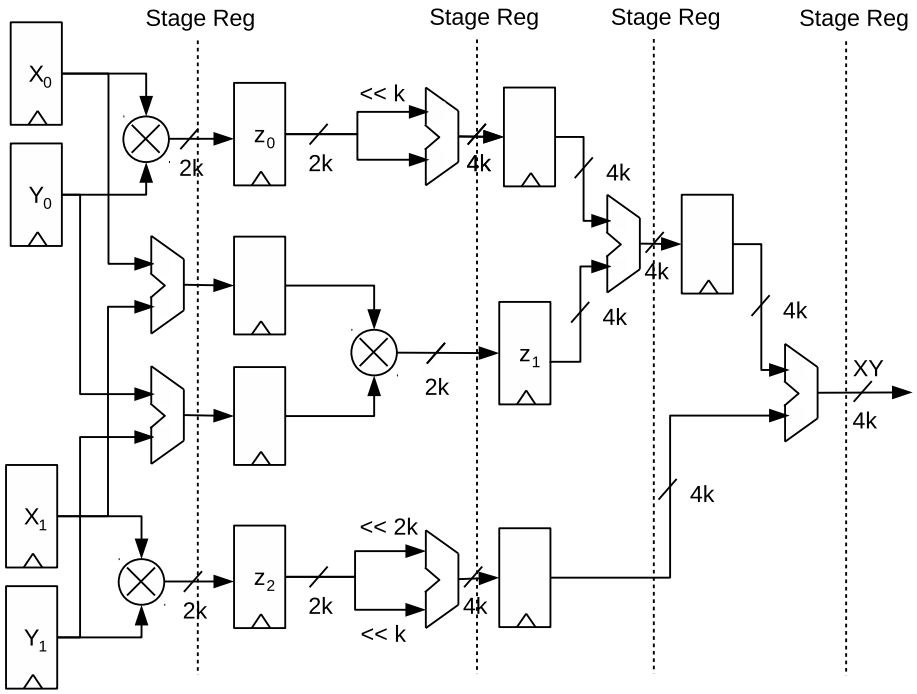

Figure 5.3: The architecture of a single recursive layer of the proposed RKM multiplier.

Equation (5.9) is basically the decrypted c_{match}, and the result is examined slot-by-slot. The first plaintext slot represents the match result for $\mathbf{y}$ and $\mathbf{v}$, and the output is $01111 = 15$. Because $2 \nmid 15$, $\mathbf{y}$ does not match $\mathbf{v}$. Next, for the second slot where $\mathbf{x}$ and $\mathbf{v}$ are compared, the result is $11110 = 30$ and $2 \mid 30 = 15$. This is a match, and the corresponding Bayesian weight for $\mathbf{v}$ is $\rho_v = 15$.

The batching technique is general, as long as the ciphertext can hold the shifted value without overflow modulo ν. In addition, it is noted that the only modifications are on SNBF.Encrypt and SNBF.Decrypt, where simple shifts can be applied to pack multiple words in an email into one set of ciphertexts. Since SNBF.Filter is not modified, the view of the server on the ciphertext remains unchanged. Hence, essentially, the server will not be able to distinguish between a single-word and a multi-word filter. As later shown in Section 5.5, under the proposed parameters, more than 100 words can be packed into one set of ciphertext vectors $\mathbf{c}$ and $\overline{\mathbf{c}}$, greatly improving the practicality of SNBF.

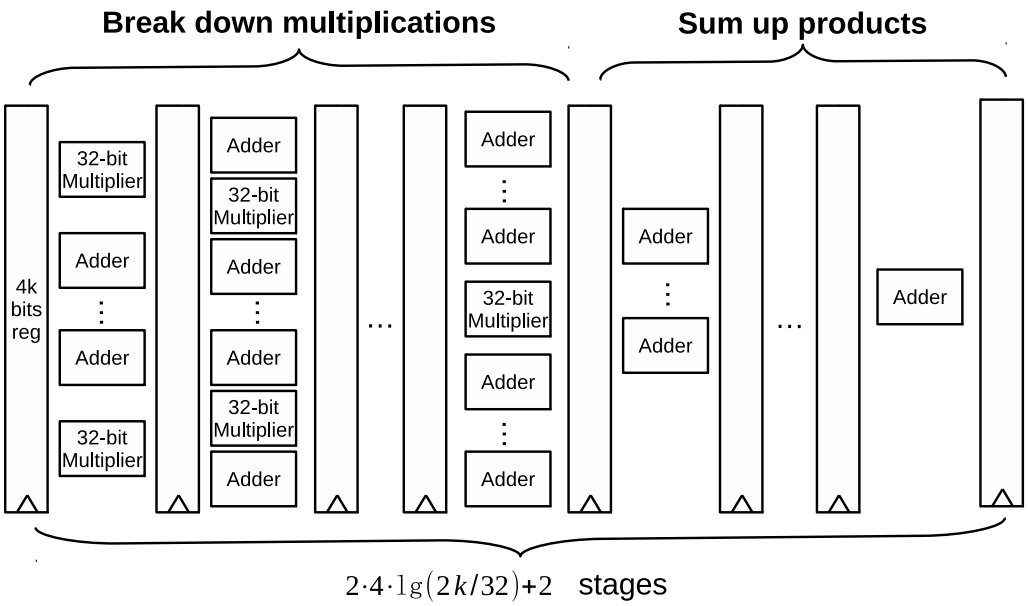

Figure 5.4: The general architecture of the proposed RKM multiplier with the recursive layers unfolded.

5.4 Hardware Architecture

Accelerating cryptographic primitives through hardware is a popular field of study. In particular, large integer multipliers are known to be used to accelerate encryption and decryption process of factoring-based cryptosystems like RSA [83–85] and ECC [86]. Nevertheless, many of the existing architectures mainly targets on prototyping on FPGA for embedded use [84, 86]. While high-speed architectures do exist [83, 85], they generally only rely on Montgomery algorithm, and have poor extendibility for even larger bit-width.

Different from existing cryptographic needs [83–85], the proposed SNBF has a distinct requirement for the underlying cryptographic hardware: the amount of computation required by SNBF is much larger than existing applications. Abstractly, the filter requires $\ell \cdot \lceil \lg(p_i) \rceil$ multiplications per filter word, and there are N words. Any practical instantiation of the SNBF results in hundreds of thousands of large (e.g., 2048-bit) integer multiplications mod the Paillier modulus ν^2. Thus, for the design of a hardware multiplier targeting on accelerating SNBF, flexible solutions are desired. In addition to the low-power requirement, the hardware platform also needs to be fast enough to make SNBF a practical solution for secure email filtering.

In this work, the recursive Karatsuba-Montgomery (RKM) multiplier based on the work in [84] is adotped. The major benefit of this approach is its

flexibility and design simplicity. Since previous cryptographic applications does not require such large modulus, Chow et al. implemented an unoptimized version of the RKM algorithm, and only tested their design up to 512-bit multiplications. To maximize the practicality of SNBF, a full-custom design for the RKM multiplier is devised.

The basic structure of RKM follows the idea of the Karatsuba multiplication algorithm. Two $2k$-bit integer inputs X and Y are broken up into the respective upper and lower part $X = 2^k X_1 + X_0$ and $Y = 2^k Y_1 + Y_0$. The algorithm proceeds by computing three intermediate sums

$$z_2 = X_1 Y_1, z_0 = X_0 Y_0, \text{ and} \tag{5.10}$$

$$z_1 = z_2 + z_0 - (X_1 - X_0)(Y_1 - Y_0). \tag{5.11}$$

The product can then be computed as $XY = 2^{2k} z_2 + 2^k z_1 + z_0$. By simply rearranging the terms, the algorithm can be further optimized as

$$XY = (2^{2k} + 2^k) z_2 + 2^k (X_0 - X_1)(Y_1 - Y_0) + (2^k + 1) z_0. \tag{5.12}$$

The hardware architecture of the adopted RKM multiplier is outlined in Figs. 5.3 and 5.4. First, in Fig. 5.3, a single recursive layer is depicted, where four pipeline stages are required. The amount of recursions can be adjusted according to design trade-offs, and in this work, the recursion is unfolded down to 32-bit, as shown in Fig. 5.4. To count the total number of pipeline stages, it is observed that both the first and second stages will have $\lg(2k/32)$ levels of recursions, where each recursion expands one multiplier into four levels of pipelined multipliers. Thus, the total number of pipeline stages is $2 \cdot 4 \cdot \lg(2k/32) + 2$ in the proposed design. The third and fourth stages do not expand, so they only add two levels to the total number of stages. The asymptotic performance is unlikely to be impacted by the deep pipeline structure, for that SNBF executes a large number of modular multiplications per word filtering (more than 70 K in the proposed parameter instantiation). Therefore, even with a large number of pipeline stages, asymptotic single cycle performance is still achievable.

On the other hand, Montgomery multiplication can be easily implemented on top of Karatsuba. Let $R = 2^\kappa$, $\gcd(R, \nu^2) = 1$ where ν^2 is the Paillier modulus, and $R > M$. Pre-compute R^{-1} and M^{-1} where $RR^{-1} = \nu^2 M^{-1}$. For each incoming pair of ciphertext A and B where the goal is to compute $A \cdot B \bmod \nu^2$, $T = AR \cdot BR$ is first computed, and then U is set to be $(T + (T \cdot M^{-1} \bmod R) \cdot M)/R$. If $U \geq M$, the algorithm returns $U - M$, and U otherwise. The result of this multiplication will be $ABR \bmod M$, which allows us to multiply another integer using the same algorithm without the need to convert the input by multiplying R.

5.5 Experiment

5.5.1 Dataset

In the experiment, the Enron email dataset parsed by [79] is used. The Enron dataset contains six sets of real-world emails, where each set belongs to one particular employee in the Enron investigation. The emails are pre-classified into hams or spams.

One subtle difference between SNBF and a regular NBF is that self-learning, i.e., optimizing the filter weights using the encrypted mails, is impossible. Thus, different from the study in [79], the email filter is trained using emails in Enron1 to Enron5 (total of 27,716 emails), and test the accuracy with Enron6 (total of 6,000 emails). The best filter size $|V| = N$ is examined, since larger N means higher computational cost, especially in the case of SNBF. The accuracy is measured through ham/spam recall. For a set of testing emails, if the ham recall is low, more ham emails are being misclassified as spams. Similarly, if spam recall is low, spams are misclassified as hams. In the context of email classification, users generally do not tolerate low ham recall, and occasional misclassification of spams as hams (low spam recall) are more bearable.

5.5.2 Cryptographic Instantiations

For the cryptographic parameters, the Paillier scheme is instantiated with a 1024-bit key size, which translates to 80-bit security. 1024-bit key requires the Paillier modulus ν^2 to be 2048-bit, and the RKM multiplier is instantiated with a $k = 1024$. Consequently, the multiplier contains 50 pipeline stages. All words in the emails and filters are hashed into 61-bit bit strings, making $\ell = 61$, which is a prime. This number directly affect the number of distinct words that can exist (2^ℓ), so 61 is chosen to ensure a large word space.

The software version of SNBF is implemented in C++ and tested its performance on an Intel Xeon E5-2630 2.3 GHz processor using a single core with 32 GB of memory. The 2048-bit RKM multiplier is synthesized on a commercial 65 nm low-power process node using a logic-synthesis tool [66].

5.5.3 Filter Accuracy

This section starts by experimenting on the impact of filter size on the ham/spam recalls. The results are shown in Figs. 5.5 and 5.6. The first conclusion here is that, even training on emails sets from different employees, there is still a diminishing effect of filter size N on classification recalls. Since N

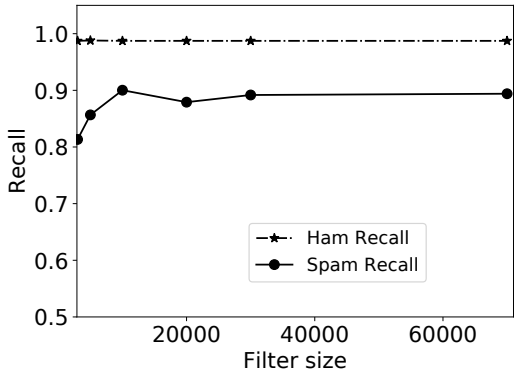

Figure 5.5: Ham/spam recall for different filter size N with floating point ρ.

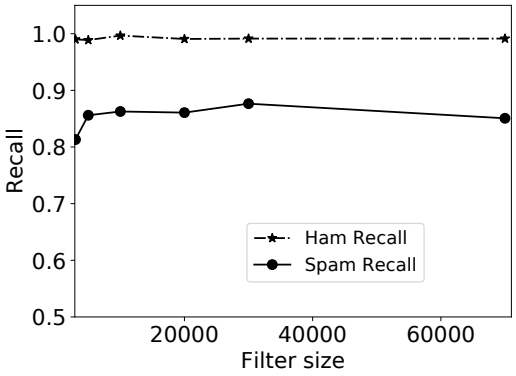

Figure 5.6: Ham/spam recall for different filter size N with integer ρ.

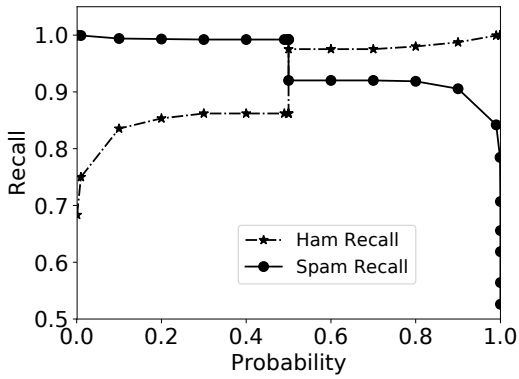

Figure 5.7: Ham/spam recall for different threshold probabilities with $N = 10000$.

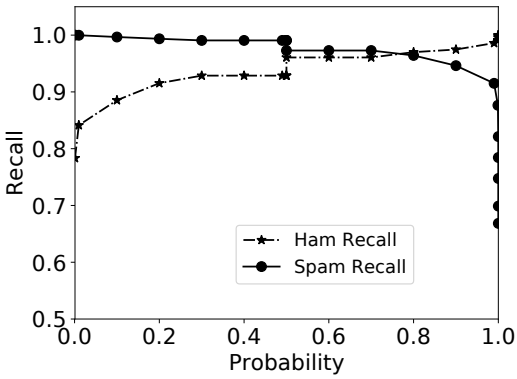

Figure 5.8: Ham/spam recall for different threshold probabilities with $N = 30000$.

is linearly related to the computational cost, $N = 10000$ is chosen for subsequent comparisons.

The second key observation here is that changing from integer to float weights does not lead to significant recall improvement. The maximum difference occurs at $N = 10000$, where the spam recall decreases by 3.8%. The average recall difference is around 1%, and as discussed in [79], for email classification, the user can hardly notice such level of recall difference. In return, the performance is significantly improved. For example, Bost et al. converts a real number into a 52-bit fixed point representation [45]. The embedding of such a large number into a PHE scheme, in the worst case, needs to compute $\sum_{i=0}^{51}(\mathsf{Enc}(\ell) \boxdot 2^i)$, which translates to more than 1000 homomorphic additions

Table 5.2: Synthesis Result for RKM Multiplier

Bit Width	Power	Area (NAND2)	Delay
2048-bit	49.5 mW	18557560	69 ns

Table 5.3: Detailed Performance Data for SNBF

Setup	Encryption	Filtering	Decryption
113 ms	2382 ms	4981 ms	141 ms

over the ciphertext containing match result. In the proposed case, the dataset is examined, and it is discovered that the maximum trained Bayesian exponent is only 17 (i.e., $\max(\{\rho\}) = 17$ using the notation in Section 5.3.3). Therefore, all exponents are restricted to be less than the integer 16. In general, large Bayesian exponent bias the result towards one direction (very ham or very spam), and should be avoided after all. Thus, by using integer weights whose size is less than 16, at most 7 homomorphic additions will suffice ($\mathsf{Enc}(\ell)^{3+2+1+0}$ for $\mathsf{Enc}(\ell)\boxdot 15$) for weight embedding. More importantly, the embedded result is also a small number ($\lg 61 \cdot 16 < 10$ bits), where existing study requires at least 52 bits [45]. Hence, the practical efficiency of SNBF is orders of magnitude better than existing works.

Finally, it is noted that larger filter sizes do give more stable responses. In Figs. 5.7 and 5.8, the recalls are tested with respect to the threshold probability p_{th}, where only emails with $\Pr(s_{\text{MAP}}) > p_{\text{th}}$ are classified as spam. Both figures exhibit a discontinuity at $p_{\text{th}} = 0.5$, indicating that a number of emails have uncertain spam probabilities near 0.5, where a small change in p_{th} affects the ham/spam recalls significantly. In comparison, when $N = 30000$, the jump is much smaller in size, and the overall curve is smoother. Therefore, while for this particular dataset, $N = 10000$ gives better numerical recalls, for arbitrary dataset, the filter size may need to be adjusted accordingly.

5.5.4 Performance Comparison

Table 5.2 summarizes the statistics of the synthesized RKM multiplier. The stage delay is 69 ns at 49.5 mW with area at around 18.5 M gates. The stage delay measured by the longest path between two stage registers, which is basically a 32-bit multiplier. For a complete Montgomery reduction, three 2048-bit multiplications are needed, with a throughput of 4.8 M 2048-bit multiplications per second.

For SNBF, a total of 734,147 modular multiplications per word using a 10,000-word filter is required. A large amount of multiplications is devoted to per-line ciphertext comparison, where each line requires $\ell - 1$ modular

Table 5.4: Summary of the Performance Comparison SNBF and Other Existing Works

	[17]	[82]	CPU	RKM
Ctxt size	487.5 KB	567 KB	512 B	512 B
Time/Mult	3.48 ms	17 ms	9–29 μs	207 ns
Time/Word	2.55 Ks	12.4 Ks	4.98 s	0.15 s
Power	165 W	130 W	95 W	49.5 mW
Energy/Word	420 KJ	1.61 MJ	473 J	7.42 mJ

multiplications. While traditional NBF filter can easily handle this amount of computation, it proves to be the main performance bottleneck for FHE/PHE-based schemes. The runtime breakdown for different operations of SNBF is outlined in Table 5.3, where it is observed that SNBF.Encrypt (Bob) and SNBF.Filter (Charlie) take the most amount of time, and SNBF.Decrypt (Alice) is a lightweight computation (which is a general property for factoring-based cryptosystems). Note that the runtime is recorded for one set of ℓ-bit Paillier ciphertexts, which is capable of evaluating up to 100 words using the batching technique. As summarized in Table 5.4, using the proposed RKM multiplier, a single ciphertext pair can be filtered in 0.15 second, 33x faster than CPU implementation. Compared to existing FHE-based schemes, the proposed scheme is 10^4–10^5x faster, and obtain 10^8–10^9x energy reduction when combined with the power reduction of RKM multiplier. Furthermore, by adopting the plaintext batching technique, more than 100 words can be concurrently evaluated with one ciphertext pair. In the Enron6 dataset, an average-length email contains 287 words, which can be classified within 0.5 s using SNBF running on the RKM accelerator. Therefore, the proposed SNBF can be safely considered as a practical secure email filtering scheme.

5.5.5 Comparison to Trivial Approach

It is observed that when the filter is public to Alice, Alice can download the entire filter to her local workspace and filters the mail on her own, which is referred to as the "trivial" approach for SNBF. For the proposed instantiated parameter set where $\ell = 61$ and $N = 10000$, this is arguably true, since the whole filter is only roughly 77 KB in size. However, the SNBF can actually be thought of as a searchable encryption (SE) scheme, which also receives wide research interests in recent years [87–92]. The basic idea of SE is that the user does not want to keep the large database, the filter per se in SNBF, on her local workspace.

The performance difference between PHE SNBF and a trivial SNBF becomes

apparent when ℓ gets larger. For example, if images also appear on the list, a large $\ell = 65536 = 8\,\text{KB}$ may be required. For $N = 10000$, this makes the filter size to be 80 MB, which is a fair amount of memory for an embedded user to consume. In addition, the user needs to compute $65536 \cdot 10000$ bit comparisons per word filtering, which can be resource-taking for long emails.

In comparison, most of the works and storage burdens for such parameter instantiation can be lifted using SNBF. Since $\lg \ell = 16$, if $\max(\{\rho\}) = 17$ as discovered in Section 5.5.3, the maximum comparison result takes $16 + \lg 17 < 21$ bits to store. Therefore, SNBF can be used as is with slightly decreased batching capability, where $1024/21 > 48$ words can be batched per ciphertext. On the user's perspective, for an email of length less than 48 words, the storage requirement is only N 2048-bit Paillier ciphertext, which translates to a total of 256 KB, instead of the 80 MB filter. From the computational perspective, Alice does the same amount of work whether ℓ equals to or 61 or 65536, while in a trivial approach, the computational burden of Alice increases significantly. In summary, by outsourcing the filter database to Charlie and letting Charlie do the heavy works, both the storage and computational burden of Alice decreases from $O(N\ell)$ to $O(N \log_2 \ell)$, due to lines 4 and 5 in Alg. 6.

5.6 Summary

A secure naïve Bayesian filter with provable security and practical performance is proposed in this chapter. Additive-homomorphic bit comparison and integer weights are employed, which allows the proposed SNBF to be instantiated using efficient PHE schemes. In addition to the cryptographic-layer protocol design and application-level optimization, this chapter also proposes an accelerator in the hardware-layer for the modular multiplications that are extensively used in performing homomorphic operations. By implementing SNBF on both software and hardware, the proposed scheme is orders of magnitude better than existing works. The experimental results indicate that SNBF is able to classify an average-length email in 0.5 s, making SNBF an attractive candidate for secure email filtering.

Chapter 6

Lattice-based Homomorphic Inference

6.1 Introduction

In a secure inference setting, Bob as a client wants to perform inference on some pre-trained inference engines (e.g., deep neural networks), but he does not want to reveal his inputs to the engine. On the other hand, it is also financially unwise for Alice, the proprietary owner of the engine, to transfer the pre-trained knowledge base into the public domain. Due to the multidisciplinary nature of the topic, secure inference based on neural networks involves contributions from various fields of research. Initial design explorations focused on the feasibility of the secure execution of the inference engine, which generally carries impractical performance overheads [32, 34]. Recent advances in cryptographic primitives [29] and adversary models [33, 35] have brought input-hiding secure inference into practical domain, where realistic image datasets can be classified within seconds [35]. It is also observed that hardware-friendly network architectures (e.g., binarized neural networks (BNN) [93, 94]) can be adopted in a secure setting to reduce the computational and communicational overheads. Furthermore, design optimizations on the fundamental operations (e.g., secure matrix multiplication [95], secure convolution [35]) involved in secure inference can also greatly improve its practical efficiency.

In this chapter, DArL-ENSEI, a secure inference framework based on the Gazelle protocol proposed in [35], is proposed. The framework consists of two main components, namely, ENSEI and DArL. ENSEI is an oblivious frequency-domain convolution technique, and DArL provides dynamic parameter adjustment for ENSEI. Specifically, for ENSEI, the main objective is to reduce the number of multiply-accumulate (MAC) operations in the homomorphic domain,

as MAC requires not only homomorphic multiplications, but also expensive homomorphic permutations (automorphisms) for the accumulation process. By applying NTT in an oblivious manner, homomorphic convolution can be simplified to efficient element-wise homomorphic multiplication, completely eliminating the necessity of homomorphic MACs. Meanwhile, DArL targets on exploiting the unused correctness margin to improve the computational and communicational efficiency of HE without security loss. To achieve these goals, two approaches are used: i) application-specific theoretical derivations for error bounds, and ii) a fast empirical method for simulating rare failure events. As a proof-of-concept, the BFV cryptosystem [30, 51] adopted in Gazelle [35] is used as a demonstration. Here, BFV serves as a fundamental design element in implementing secure neural network inference. First, theoretical bounds are applied to examine the error behavior of BFV in ENSEI, and minimize the unused error margin by dynamically switching parameter sets during the inference stage. Second, the sigma-scaled sampling technique [96] is adopted to obtain the exact decryption failure probability, which is important in fine-tuning the parameters by techniques such as approximate computing [68].

In the experiment, DArL-ENSEI is compared with Gazelle on well-known datasets such as MNIST [97] and CIFAR [98]. Using ENSEI, 4–10x online and 31x offline time reductions, as well as 2x bandwidth reduction for the convolutional layers are observed. For FC layers, by instantiating tighter parameter sets with a smaller ciphertext modulus q, the ciphertext size can be reduced by as much as 67%. More importantly, by switching to hardware-friendly network architectures such as BNN, a further 2x–3x ciphertext size reduction and computation speedup are simultaneously attainable in DArL-ENSEI. As a result, DArL-ENSEI can execute an instance of secure inference of the (relatively) complex network architecture in [33] within 0.378 s on the CIFAR dataset, greatly enhancing the practicality of secure inference based on convolutional neural networks (CNN). The contributions are summarized as follows.

- **Frequency-Domain Secure Inference**: While the original idea of FDC [99, 100] is widely adopted in CNN libraries, its design in the secure domain remains unexplored. In this chapter, it is shown that a secure version of FDCNN also profoundly impacts the inference efficiency, and is crucial to the scalability of CNN-based secure inference schemes. With the assistance of a packed additive homomorphic encryption (PAHE) scheme, the online inference time can be reduced by 4–10x, along with at most a 31x setup time reduction. Moreover, ENSEI is applicable to all types of (interactive) protocols that require the computation of convolutions in the encrypted domain.

- **Dynamic Parameter Adjustment**: Existing HE applications generally assume one set of parameters being used throughout the whole evaluation process. In this chapter, by realizing that different neural network architecture can give rise to significant performance gap, rigorous analyses on the error behaviors in the linear kernel used in Gazelle [35] and ENSEI is conducted, with a set of candidate parameters proposed to exploit the unused error margin.

- **Fast Simulation of Decryption Failures**: Previous works on LWE generally focused on developing theoretical bounds on the size of the decryption errors, except for [68] where a Monte-Carlo approach was employed. While empirical simulation provides a tighter bound as shown in [68], it is clearly not practical to simulate a failure probability of 2^{-40} using brute-force Monte-Carlo. In this chapter, the sigma-scaled sampling [96] is used to further optimize the runtime parameters of DArL.

- **A New Precision-Accuracy Trade-off**: Existing works on trading a small portion of prediction accuracy for more efficient network implementation generally target resource-constrained devices that operate in embedded environment [93, 94, 101]. This chapter demonstrates that these techniques also yield trade-offs that inspire better network designs in the secure domain. In particular, three sets of RLWE parameters are instantiated for CNN with different precisions to demonstrate their influence on the practical performance of CNN-based secure inference.

The rest of this chapter is organized as follows. First of all, preliminary knowledge on Gazelle and the concrete BFV instantiation are delivered in Section 6.2. Second, the design of ENSEI is laid out in Section 6.3. Third, ENSEI is integrated into the Gazelle protocol, and the parameter requirements are discussed in Section 6.4. Forth, theoretical error analyses on the general linear kernel is performed in Section 6.5. Fifth, in Section 6.6, the dynamic parameter adjustment procedure is described in detail, and the application of fast empirical methods for even tighter bound analysis. Sixth, the proposed parameter sets and experiment results are presented in Section 6.7. Last but not least, this chapter is summarized in Section 6.8.

6.2 Gazelle: Secure Neural Network Inference

Gazelle [35] represents one of the most recent advances in the line of prior arts [33,34] on designing secure inference. The general protocol and architecture of Gazelle is outlined in Fig. 6.1, where Bob wants to classify some input (e.g.,

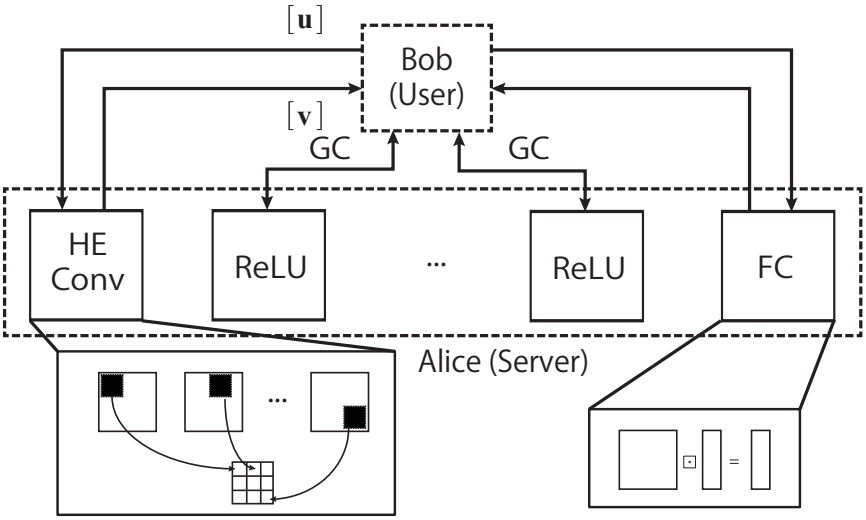

Figure 6.1: An example of the architecture in Gazelle with two Conv (convolutional) layers, two non-linear layers and one FC (fully-connected) layer. The FC layer, much like the Conv layers, is internally a homomorphic matrix-vector product.

image), and Alice holds the weights. The threat model is that both Alice and Bob are semi-honest, in the sense that both parties follow the described protocol, but want to learn as much information as possible from the other party. Here, the error behaviors of the linear Conv (convolutional) and FC (fully-connected) layers are summarized, as these layers are implemented by the PAHE-based matrix-vector multiplication techniques.

Homomorphic Linear Kernels

In both Conv and FC, the main arithmetic procedure is computing a set of inner products between some plaintext matrix (or vector) and a ciphertext vector, as shown in Fig. 6.1. The basic approach (called naïve method in [35]) for computing an inner product is as follows. For some row vector $\mathbf{w}$ in the weight matrix W and ciphertext $[\mathbf{u}]$, the coefficient-wise homomorphic product is computed as $[\mathbf{v}] = [\mathbf{u}] \boxdot \mathbf{w}$, where each $v_i \in \mathbf{v}$ satisfies $v_i = u_i \cdot w_i$ for $u_i \in \mathbf{u}, w_i \in \mathbf{w}$. Then, to obtain the sum $\sum_i^{n-1} v_i$, i) a rotated version of $[\mathbf{v}]$ by a step size of $k = n/2$ is first created, and ii) a coefficient-wise homomorphic addition between $[\mathbf{v}]$ and $\mathsf{rot}([\mathbf{v}], k)$ is computed. Repeating i) and ii) $\log_2(n)$ times, each time decreasing the value of k by half (i.e., $k_i = n/2^i$ for $i \in [1, \log_2(n)] \cap \mathbb{Z}$), the desired sum is obtained in the first plaintext slot.

The shortcoming of the basic technique is that, for a weight matrix of dimension $n_o \times n$, computing $W \cdot [\mathbf{u}]$ results in n_o many ciphertexts,

Table 6.1: Example of Parameter Instantiation in Gazelle

n	q	$\lg q$	p	$\lg p$	σ	b
2048	$2^{60} - 2^{12} \cdot 63549 + 1$	60	307201	18	4	10

each containing only one result of the inner product, which blows up the communication bandwidth. The proposed approach in Gazelle, called the hybrid method in [35], is to align the weight matrix with the rotating input ciphertext (instead of the rotating product ciphertext as in the basic approach). In summary, the hybrid approach computes $W \boxdot [\mathbf{u}]$ as follows.

$$[\mathbf{v}] = \sum_{i=0}^{n_o-1} \mathbf{w}_i \boxdot \mathsf{rot}([\mathbf{u}], i) \tag{6.1}$$

$$= \mathbf{w}_0 \boxdot [\mathbf{u}] + \cdots + \mathbf{w}_{n_o-1} \boxdot \mathsf{rot}([\mathbf{u}], n_o - 1), \tag{6.2}$$

$$[\mathbf{r}] = \sum_{i=1}^{\lg(n/n_o)} \mathsf{rot}\left([\mathbf{v}], \frac{n}{2^i}\right), \tag{6.3}$$

where $[\mathbf{r}]$ holds the result vector $\mathbf{r} = W\mathbf{v} \in \mathbb{Z}_p^{n_o}$, $\mathbf{w}_i$'s are the diagonally aligned columns of W with dimension $\mathbf{w}_i \in \mathbb{Z}_p^n$, and $\lg(\cdot)$ denotes $\log_2(\cdot)$. In the hybrid approach shown in Eq. (6.2), $[\mathbf{u}]$ is first rotated n_o times, each time multiplying it with the aligned vectors $\mathbf{w}_i \in \{\mathbf{w}_0, \cdots, \mathbf{w}_{n_o}\}$. Each multiplication generates an intermediate ciphertext that holds only *one* entry in $\mathbf{v}_i$ with respect to $\mathbf{w}_i$. Summing these ciphertexts gives us a single ciphertext that contains n/n_o partial sums of the corresponding inner products, and then the basic approach is used to rotate this packed result to obtain the final sum, as in Eq. (6.3).

Parameter Instantiation for BFV in Gazelle

The parameters instantiated by Gazelle are summarized in Table 6.1, where $\lg x$ denotes $\log_2(x)$. Unfortunately, Gazelle did not provide any correctness discussion on the instantiated 60-bit modulus q. However, a more rigorous correctness analysis can potentially allow us to use smaller q while fixing other parameters, which in turn gives us stronger security, better modular reduction efficiency and smaller ciphertexts.

6.3 Oblivious Convolution in the Frequency Domain

If homomorphic convolution is expensive, as observed in Section 6.2, a natural question to ask is that if this operation can be avoided in the first place. In

what follows, it is shown that by applying NTT, a homomorphic convolution operation can be replaced by a simple SIMD scalar homomorphic multiplication. It is noted that ENSEI is general in a multi-party secure computing (MPC) setting, where the cost of computations are asymmetric across different parties.

6.3.1 Finite Fields versus the Complex Field

The use of finite-field NTT in replacing complex-valued discrete Fourier transform (DFT) for faster and simpler convolution in digital circuits is a well-known technique [102]. The convolution theorem tells us that for two discrete sequences $\mathbf{w}$ and $\mathbf{u}$, there exists a general transformation in the form

$$F(x) = \sum_{i=0}^{n-1} x(i) \cdot \omega^{ik}, \tag{6.4}$$

where the following property holds

$$F(\mathbf{w} * \mathbf{u}) = F(\mathbf{w}) \circ F(\mathbf{u}), \tag{6.5}$$

Here, $\circ$ denotes element-wise multiplication, and ω is the n-th root of unity in some field $\mathbb{F}$ (i.e., $\omega^n = 1$ over $\mathbb{F}$). As shown in [102], using the complex number field as $\mathbb{F}$ is a viable choice, but not the only choice. If finite fields, e.g., $\mathbb{F}$, are chosen, we obtain NTT, and Eq. (6.5) still holds. However, since modern machines can easily evaluates 64-bit to 128-bit floating point arithmetic, NTT-based convolution is not particularly attractive in terms of its performance, due to the additional reductions modulo some large field prime.

The main observation for adopting NTT in this work is that, in a cryptographic setting, finite fields are much more natural to use than the complex number field. Most cryptographic primitives that build on established hardness assumptions live in finite fields, where arithmetic operations do not handle floating numbers (and complex numbers) particularly well. This is especially true for homomorphic encryption schemes whose securities are based on hard lattice problems. Therefore, NTT-based convolution is considered as a natural choice for secure convolution. However, since the proposed method is applicable to any transformation F, the complex number field can also be adopted as the ground field.

6.3.2 ENSEI: The General Protocol

Figure 6.2 outlines the general protocol for an input-hiding oblivious convolution in the frequency domain. This protocol assumes the existence of a linear randomization technique. Here, it is assumed that Bob holds some two-dimensional

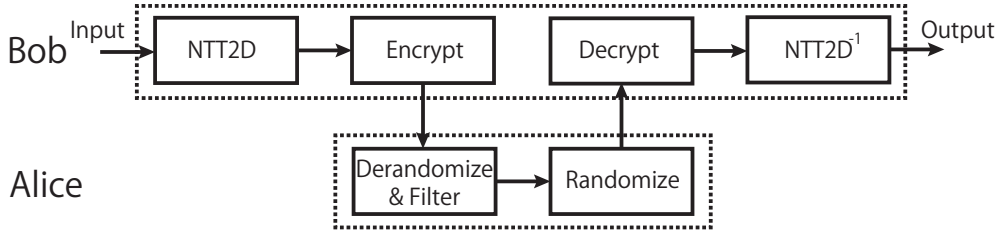

Figure 6.2: The general protocol of a round of NTT-based oblivious FDC.

(randomized) plaintext input image $U \in \mathbb{Z}_p^{n\times n}$, and this input is randomized using additive secret sharing (ASS), i.e., $U = (X + R) \bmod p$ for some (real) input X and random matrix $R \in \mathbb{Z}_p^{n\times n}$. Bob wants to compute a convolution between U and the filter $W \in \mathbb{Z}^{n\times n}$ held by Alice (if the real weight matrix is smaller, then it is assumed to be zero-padded). The protocol goes as follows.

1. **Bob**: Bob first pads the input according to the convolution type (e.g., same or valid), and computes a two-dimensional NTT on U as $\hat{U} = \mathsf{NTT2D}(U)$. Note that since DFT is a linear operation, we have that $\mathsf{NTT2D}(X + R \bmod p) = (\mathsf{NTT2D}(X) + \mathsf{NTT2D}(R)) \bmod p$.

2. **Bob**: The transformed $\hat{U}$ is encrypted using an abstract encryption function Enc. There is only one requirement for Enc, which is that it permits finite-field addition operation, i.e., $\mathsf{Dec}(\mathsf{Enc}(x) + \mathsf{Enc}(y)) \bmod p = (x + y) \bmod p$ for $x, y \in \mathcal{P}$ for some plaintext space $\mathcal{P}$. This can be fulfilled by most AHE schemes and interactive protocols such as garbled circuit (GC). Since the filtering step becomes a trivial element-wise multiplication, Bob flattens the frequency-domain matrix $\hat{U}$ to $\hat{\mathbf{u}}$ and encrypts it using Enc. The encrypted ciphertext $[\hat{\mathbf{u}}] = \mathsf{Enc}(\hat{\mathbf{u}})$ is transferred to Alice through public channels.

3. **Alice**: Upon receiving $[\hat{\mathbf{u}}]$, Alice first de-randomize the input by computing $[\hat{\mathbf{x}}] = [\hat{\mathbf{u}}] \boxminus \hat{\mathbf{r}}$, where $\hat{\mathbf{r}}$ is a flatten version of $\hat{R} = \mathsf{NTT2D}(R)$, and $\boxminus$ is a homomorphic (or oblivious) subtraction operator.

4. **Alice**: Alice performs similar NTT-flatten operation as $\hat{\mathbf{w}} = \mathsf{flatten}(\mathsf{NTT2D}(W))$, and carries out an element-wise multiplication as $[\hat{\mathbf{x}} \circ \hat{\mathbf{w}}] = \mathsf{EMult}([\hat{\mathbf{x}}], \hat{\mathbf{w}})$. In order to prevent weight leakages, Alice re-randomizes the filtered result using a new randomization vector $[\mathbf{r}_1]$ as $[\hat{\mathbf{v}}] = [\hat{\mathbf{x}} \circ \hat{\mathbf{w}} - \mathbf{r}_1]$. $[\hat{\mathbf{v}}]$ is sent back to Bob.

5. **Bob**: Bob decrypts $[\hat{\mathbf{v}}]$ as $\hat{\mathbf{v}}$ and apply INTT to obtain $\mathbf{v} = (\mathbf{x} * \mathbf{w} - \mathsf{INTT}(\mathbf{r}_1)) \bmod p$.

Upon receiving $\mathbf{v}$, with the assistance of Alice, the two parties can perform further computations based on the convolution results of $\mathbf{x}$ and $\mathbf{v}$, without Alice knowing $\mathbf{x}$ and Bob knowing $\mathbf{w}$.

Correctness

Claim 2. *The protocol in Section 6.3.2 correctly evaluates the convolution between X and W.*

The correctness proof is quite simple, and a formal presentation is not included. The most important properties used are the two-dimensional versions of

$$\begin{aligned}&((\mathsf{NTT}((\mathbf{x}+\mathbf{r}) \bmod p) - \mathsf{NTT}(\mathbf{r})) \bmod p \circ \mathsf{NTT}(\mathbf{w})) \bmod p \\ &= \mathsf{NTT}(\mathbf{x} * \mathbf{w}),\end{aligned} \tag{6.6}$$

and

$$\mathsf{INTT}((\mathsf{NTT}(\mathbf{x}) \circ \mathsf{NTT}(\mathbf{w})) + \mathbf{r}_1) \bmod p) = \mathbf{x} * \mathbf{w} + \mathsf{INTT}(\mathbf{r}_1). \tag{6.7}$$

Assuming that the underlying AHE scheme can carry the computation in Eq. (6.6) correctly, based on Eq. (6.5) and the correctness of ASS, Claim 3 can be easily proved.

Security

Claim 3. *The protocol in Section 6.3.2 remains secure if either Alice or Bob is compromised by a semi-honest adversary, but not both.*

It is pointed out that security-wise, the proposed protocol is basically identical to the linear kernel in Gazelle [35], so a formal proof is also left out. Briefly speaking, given Enc and Dec, Alice cannot temper the inputs of Bob ($\mathbf{x}$), and with ASS, Bob gains no knowledge of Alice's inputs ($\mathbf{w}$).

6.4 Integration and Parameter Instantiation

First, it is noticed that by simply replacing the "HE Conv" block in Fig. 6.1 by ENSEI in Fig. 6.2, frequency-domain secure inference is achieved. However, the integration and parameter selection processes need careful examination. In this section, an integration technique that reduces the number of total NTTs required is devised, and the precision requirement of this approach is discussed.

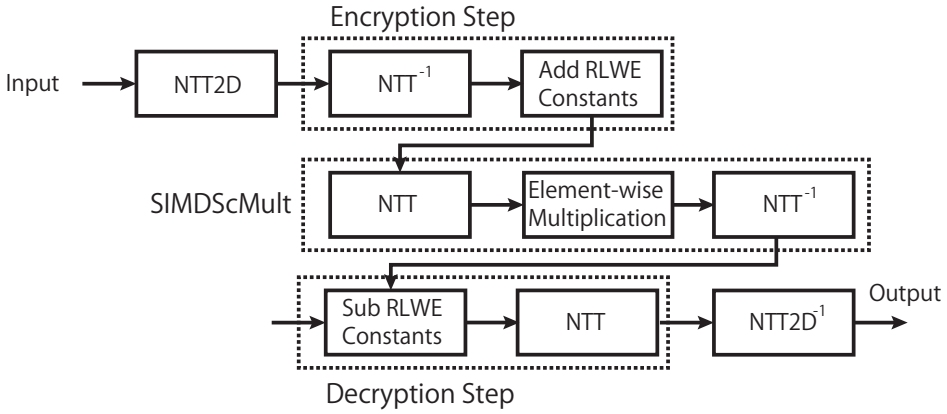

Figure 6.3: The overview for a sequence of Enc-SIMDScMult-Dec procedures based on ENSEI for general AHE schemes.

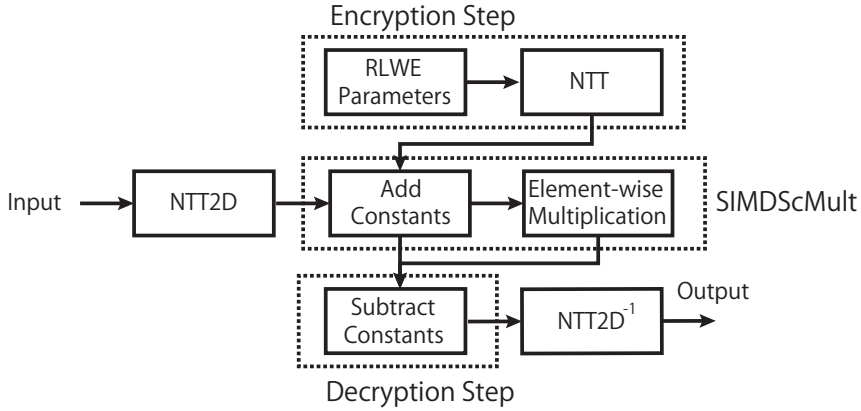

Figure 6.4: Modified Enc-SIMDScMult-Dec for Gazelle-like networks to reduce the extra NTTs for plaintext packing.

6.4.1 Reducing the Number of NTT

The plaintext packing technique [28] used for embedding a vector of plaintext integers $\mathbf{u} \in \mathbb{Z}_p^n$ into a single ciphertext pair relies on the idea that a large-degree polynomials (with proper modulus) can be decomposed into a set of independent polynomials that are of smaller degrees. The exact operation for lattice-based PAHE is sketched in Fig. 6.3. During encryption, a vector of plaintext is transformed into a polynomial via the INTT operation and embedded into the ciphertext. Later in the evaluation stage, SIMDScMult re-applies NTT on the ciphertext (and thus simultaneously on the plaintext) followed by an element-wise multiplication. While Fig. 6.3 is a straightforward application if

the general ENSEI protocol in Fig. 6.2 is adopted, it clearly involves redundant NTTs.

The key observation here is that, the internal operation of a SIMDScMult is merely conducting frequency-domain multiplication on polynomials, as it is known that a multiplication between two polynomials in some particular quotient rings equates to a convolution of their coefficients. Therefore, the NTT-transformed frequency-domain image can be directly embedded into the NTT transformed ciphertext (i.e., $\hat{a} \cdot \hat{s} + \mathsf{NTT2D}(U)$), as depicted in Fig. 6.4, and execute SIMDScMult without the NTT in the front and INTT in the back. By performing the entire Enc-SIMDScMult-Dec process in the frequency domain, as much as two NTT and one INTT per convolution can be eliminated. The number of NTTs required is even less than the time-domain convolution because homomorphic rotations employed in Gazelle (rot) force excess applications of NTTs for the key-switching procedure. The only restriction here is that the plaintext modulus p_m needs to be larger than the ENSEI modulus p_E. Further elaborations on the exact precision settings that satisfiy the requirement $p_m \geq p_E$ is discussed in Section 6.4.2.

6.4.2 The NTT Moduli and RLWE Parameters

It is known that FFT, in general, suffers when the underlying floating point operations does not have enough precision [102, 103]. By adopting NTT, the operands are all exact integers, and this problem can be safely disregarded. However, if the input operands are of floating-point, the precision problem persists in nature, and is reflected in the chosen modulus p_E. As mentioned in Section 6.4.1, p_E is closely related to p_m, the PAHE plaintext modulus. Specifically, for the inputs to be correctly encrypted, it needs to hold that $p_m \geq p_E$. In other words, if more precisions are needed, p_E becomes larger, and all operations become less efficient.

As it turns out, p_E is determined by two factors: i) the maximum value in the matrix operands, and ii) the length of the convolving sequence. For two sequences $\mathbf{u} \in \mathbb{Z}^{n_u}$ and $\mathbf{w} \in \mathbb{Z}^{n_f}$, the lower bound on p_E can be written as

$$p_E \geq \max(\mathbf{u}) \cdot \max(\mathbf{w}) \cdot n_f, \tag{6.8}$$

and p_E needs to be a prime where $p_E^n = 1 \bmod \max(n_u, n_f)$ [104]. Here, we have $n_f = f_h \cdot f_w$. In a typical CNN setting, compared to the RLWE lattice dimension n, n_f is generally small (e.g., $n_f = 9$ for the 3×3 filters used in Section 6.7). In addition, in hardware-friendly network architectures, such as BinaryConnect [93] or BinaryNet [94], $\max(\mathbf{u})$ is generally less than 10-bit, and $\max(\mathbf{w})$ is even smaller. Moreover, the lower bound on p_E can hardly be

reached practically, as it represents the extreme case of all white inputs with meaningless filter weights.

When p_E is large, since p_m only needs to be as large as p_E, the NTT merge technique in Section 6.4.1 can be adopted. When all the terms in Eq. (6.8) is small, however, a much smaller p_E can be used, but the underlying RLWE-based PAHE scheme cannot benefit from a p_E that is too small. For security reasons, n needs to be a relative large power of 2 (e.g., 1024 to 2048), and p_m can only be as small as the smallest prime that splits over the field x^n+1 (e.g., for $n = 2048$, $p_m \geq 12289$). Fortunately in this case, multiple filter channels or even multiple images can be batched into one RLWE ciphertext for efficient plaintext slot utilization.

6.5 Modeling the Matrix Multiplication Error

In both Conv and FC layers, the basic arithmetics are generally matrix-vector multiplications of different input and output dimensions. Therefore, it is important to rigorously study the error behavior of this operation. It is also noted that since every linear layer is followed by a non-linear protocol in Gazelle, the errors do not propagate through layers, i.e., the parameters only need to be large enough to endure homomorphic evaluations within a single linear layer.

6.5.1 Formulating the Error

Since a matrix-vector multiplication is merely multiple vector-vector inner products (with a subtle difference in how the plaintext elements are aligned), the error behavior in the hybrid matrix-vector multiplication in the FC layer is first focused on. Here, the multiplication involves a weight matrix that is usually a rectangular matrix $W \in \mathbb{Z}^{n_o \times n_i}$ with the important property that $n_o \ll n_i$, and a ciphertext $[\mathbf{u}]$ as inputs. Without loss of generality, it is assumed that $n_i = n$ for the rest of this work.

Homomorphic operations always incur an additive or multiplicative error scaling to the ciphertext in Eq. (2.17). However, the important observation here is that the original error polynomial e_0 has its coefficients sampled from χ, and we have powerful tail bounds available on the product distribution where one operand involved comes from χ, as in Lemma 2. According to [35], assuming a fresh $[\mathbf{u}]$ is η_0-bounded, the error growth from this operation is

$$\begin{aligned} &((\eta_0 + \eta_{\text{rot}}) \cdot \eta_{\text{mult}} \cdot n_o + \eta_{\text{rot}}) \cdot \lg(n/n_o) \\ &= k_0\eta_0\eta_{\text{mult}} + k_0\eta_{\text{rot}}\eta_{\text{mult}} + k_1\eta_{\text{rot}}, \end{aligned} \tag{6.9}$$

where $k_0 = n_o \lg(n/n_o)$, $k_1 = \lg(n/n_o)$. In Eq. (6.9), it is assumed that a rot induces an additive η_{rot} factor, and multiplying the weight scales the error by η_{mult}. There exists no formal discussion on the correctness proof in Gazelle [35], and a straightforward calculation of Eq. (6.9) is clearly over-pessimistic in terms of the error bounds. In what follows, theoretical bounds for each error term in Eq. (6.9) are developed.

6.5.2 The Error $\eta_0 \cdot \eta_{\text{mult}}$

Recall that a freshly encrypted ciphertext $[\mathbf{u}]$ in BFV is a linear sum of $a \cdot s$, $\frac{q}{p}m$ encoding $\mathbf{u}$, and e_0 bounded by η_0. Therefore, a bound on $\eta_0 \cdot \eta_{\text{mult}}$ is essentially a bound on the polynomial product $e_0 \cdot w$. Here e_0 is a simple polynomial whose coefficients come from the distribution χ, but the construction of $w \in \mathcal{R}_p$ makes its coefficients obeying no explicit distributions. In this chapter, a combined approach is taken. First, it is assumed that the coefficients of w follow a uniformly random distribution, and asymptotically, the central limit theorem (CLT) tells us that the underlying distribution actually does not matter. Furthermore, in Section 6.6, the theoretical analyses are reaffirmed through empirical examinations.

Assuming $w \leftarrow U_b$ is uniformly random over the integers in $[-b/2, b/2) \cap \mathbb{Z}$, the L_2-norm of the coefficients in $e_{p_1} = e_0 \cdot w$ needs to be bounded. Similar to [7], it is observed that the coefficients in e_p is in the form

$$(e_{p_1})_i = \sum_{i=0}^{n-1} \pm \left((e_0)_i \cdot (w)_{(i-j) \bmod n}\right), \tag{6.10}$$

where $(x)_i$ is the i-th coefficient of x in its coefficient representation. In other words, each new coefficient $(e_p)_i$ is the sum of products of coefficients from e_0 and w. Since Eq. (6.10) can also be written as an inner product between two vectors $\mathbf{e}_0 = [(e_0)_0, \cdots, (e_0)_{n-1}]$ and $\mathbf{w} = [(w)_0, \cdots, (w)_{n-1}]$, Lemma 2 can be applied to obtain a bound on the L_2-norm as

$$\Pr[||\langle \mathbf{e}_0, \mathbf{w} \rangle|| > T\sigma||\mathbf{w}||] < 2 \cdot \exp(-T^2/2). \tag{6.11}$$

In this work, an asymptotic bound of 2^{-40} ($2 \cdot \exp(-T^2/2) \leq 2^{-40}$) is used. This bound translates to a T value of roughly 7.54, and this T is used for all the subsequent analyses. Since Eq. (6.11) also requires a bound on the L_2-norm of the vector $\mathbf{w}$, the Chernoff-Cramer inequality can be applied in a similar manner to Section 4.3.3 to set up a series of independent and identically distributed random variables $X_0 = (w)_0^2, \cdots, X_{n-1} = (w)_{n-1}^2$, and Eq. (4.6) holds for this random variable X. Last but not least, it is pointed out that

Eq. (6.10) only bound the size of one coefficient in the polynomial e_{p_2}. For the L_2 norm on the size of the whole polynomial, a simple summation in the form $||e_{p_2}|| = \sqrt{\sum_{i=0}^{n-1} ||(e_{p_2})_i^2||}$ can be applied, which is basically $\sqrt{n} \cdot ||(e_{p_2})_i||$, since all coefficients in e_{p_2} are independent random variables by design.

6.5.3 The Error $\eta_{\text{rot}} \cdot \eta_{\text{mult}}$

The error distribution of $\eta_{\text{rot}} \cdot \eta_{\text{mult}}$ is slightly harder to analyze, due to the fact that it is a product of three terms,

$$\mathsf{Decomp}_2([\mathbf{u}]) \cdot \mathbf{e}_{\mathcal{K}} \cdot w, \tag{6.12}$$

where $\mathsf{Decomp}_2([\mathbf{u}])$ and $\mathbf{e}_{\mathcal{K}}$ (defined in Eq. (2.19)) are vectors of dimension $\lceil \lg q \rceil$ over the ring $\mathcal{R}_q$, and the vector product is multiplied by w as indicated in Eq. (6.1).

First, notice the following equality: $(\mathsf{Decomp}_2([\mathbf{u}]) \cdot \mathbf{e}_{\mathcal{K}}) \cdot w = \mathsf{Decomp}_2([\mathbf{u}]) \cdot (\mathbf{e}_{\mathcal{K}} \cdot w)$. Second, observe that the bound on the L_2-norm of the coefficients in $\mathbf{e}_{\mathcal{K},w} = \mathbf{e}_{\mathcal{K}} \cdot w$ follows immediately from the previous analysis, where each $e_{\mathcal{K},i} \in \mathbf{e}_{\mathcal{K}}$ follows exactly the same distribution as e_0 in Eq. (6.10) with the same σ. Therefore, the task left is to use the L_2-norm bound on $||\mathbf{e}_{\mathcal{K},w}|| = ||\mathbf{e}_{\mathcal{K}} \cdot w||$ to derive a bound on the L_2-norm of the product $\mathsf{Decomp}_2([\mathbf{u}]) \cdot \mathbf{e}_{\mathcal{K},w}$. Note that this multiplication is an inner product between vectors of dimension $\lceil \lg q \rceil$, so there is a final scaling factor $\sqrt{\lceil \lg q \rceil}$ from the L_2 (Euclidean) summation of $\lceil \lg q \rceil$ product polynomials.

Focusing on a single polynomial multiplication, the second step is to think of $\mathsf{Decomp}_2([\mathbf{u}]) \in \mathcal{R}_q^{\lceil \lg q \rceil}$ as a vector of uniformly random binary polynomials. In other words, the coefficients in such polynomials can be seen as drawn from a Bernoulli distribution with a success probability of exactly $1/2$. The source of the uniformity comes from the fact that, $\mathsf{Decomp}_2([\mathbf{u}])$ is a decomposition of the ciphertext $[\mathbf{u}]$. In the perspective of an eavesdropper, the ciphertext is (and has to be) indistinguishable from uniformly random. In addition, it is observed in [7] that a Bernoulli is $1/2$-subgaussian, which allows us to apply Lemma 2 again. Let $e_{p_2,i} = e_{\mathcal{K},w,i} \cdot \mathsf{Decomp}_{2,i}([\mathbf{u}])$ be the i-th product polynomial, we have

$$\Pr[||(e_{p_2,i})_j|| > (T/\sqrt{2})||e_{\mathcal{K},w,i}||] < 2 \cdot \exp(-T^2/2), \tag{6.13}$$

and $||(e_{p_2})_j|| = \sqrt{\lceil \lg q \rceil}||(e_{p_2,i})_j||$ as explained.

Algorithm 8 Per-Layer Adjustment of Parameters

Require: Security parameter λ, neural network architecture $\mathcal{A}$

1: **for each** linear layer $l \in \mathcal{A}$ **do**
2: Fix n_o according to l
3: Determine the plaintext modulus p such that it can fit the maximum value during the evaluation of l.
4: **for each** $q_j \in \mathbf{q}, n_j \in \mathbf{n}$ **do**
5: Estimate the size of evaluation errors, $||e||$, using Eq. (6.15).
6: **if** $\lg ||e|| < \lg q - \lg p - 1$ **then**
7: Add (p, q_j, n_j) valid parameter list $Params$.
8: Output $q, n = \min_{q,n}(Params)$

6.5.4 The Error η_{rot}

Finally, for the last term in Eq. (6.9), Lemma 2 can be applied on the i-th product polynomial $e_{p_3,i} = \mathsf{Decomp}_{2,i}([\mathbf{u}]) \cdot e_{\mathcal{K},i}$ as

$$\Pr[||(e_{p_3,i})_j|| > (T/\sqrt{2})||e_{\mathcal{K},i}||] < 2 \cdot \exp(-T^2/2), \tag{6.14}$$

where $||e_{\mathcal{K},i}||$ is bounded by Lemma 1, and the scaling factor $\lceil \lg q \rceil$ due to the inner product remains the same. Finally, to calculate Eq. (6.9), one can simply compute

$$k_0(||e_{p_1}|| + ||e_{p_2}||) + k_1||e_{p_3}||. \tag{6.15}$$

6.6 Dynamic Parameter Estimation

6.6.1 The Overall Procedure

The proposed dynamic parameter estimation technique can be summarized by the procedures outlined in Alg. 8. Note that this is mainly an offline procedure that involves only the server (here, offline means that the optimization does not depend on user input). First, by looking at the network architecture, on n_o and p are chosen accordingly. Note that while p can depend on the user inputs, the server already knows what type of workloads (i.e., an image represented by 8-bit integers) are expected, and can set p accordingly. Next, the algorithm enters a loop where it estimates the error growths from secure inference under the instantiation of the parameters (p, q_j, n_j, n_o). Here, $q_j \in \mathbf{q}$ and $n_j \in \mathbf{n}$ are pre-computed candidates of the ciphertext modulus q and lattice dimension n in the lists $\mathbf{q}$ and $\mathbf{n}$ (in practice, while q can vary, n is generally either 1024 or

2048). The q's and n's can be generated according to the parameter selection procedures described in [35]. Subsequently, if the calculated error ensures a correct decryption up to some probabilities derived in the corresponding bounds, the parameter set is accepted. Finally, the smallest q and n are selected as the parameters for concrete instantiation.

As mentioned, the size of the parameters depends critically on p and the failure probability estimation. The concrete parameter instantiations and analyses are presented in Section 6.7, and this section first discusses how to apply fast Monte-Carlo methods to further improve the correctness analysis, when the weight matrices in $\mathcal{A}$ are known.

6.6.2 Failure Probability via Sigma-Scaled Sampling

As suggested in [68], theoretical bounds on the decryption failure probability of LWE-based cryptosystems tend to be loose. However, simulating such probability can be practically challenging, as simulating a failure probability of 2^{-40} requires roughly 2^{80} brute-force Monte-Carlo runs. In Chapter 4, a straightforward Monte-Carlo technique is used to simulate the failure bound by generating 10 million decryption samples. Nonetheless, this method only works when such probability is large, e.g., $\geq 0.01\%$ in their example. Different from a public-key encryption scheme, it is hard to adopt error correction code into a homomorphic encryption scheme where the evaluation function can be arbitrary.

Fortunately, it is observed that simulating the LWE decryption failure probability shares many similarities with simulating circuit failure probability in the field of design automation. In particular, the sigma-scaled sampling (SSS) method [96] is known to be efficient at handling high-dimensional Gaussian random variables. In short, the objective is to calculate some rare failure probability $P_f \leq 2^{-40}$, which is the probability of the decryption error $||e||$ being greater than some threshold η_t. If the homomorphic evaluation is abstracted as some function f on the initial error vector $\mathbf{e}$ as $f(\mathbf{e})$, P_f can be calculated as

$$P_f = \int_{-\infty}^{+\infty} I(\mathbf{e}) f(\mathbf{e}) d\mathbf{e}, \tag{6.16}$$

where $I(\mathbf{e}) = 1$ if and only if $||e|| \geq \eta_t$, and $I(\mathbf{e}) = 0$ otherwise. Apparently, P_f is hard to simulate directly, and the SSS relies on the simple idea of sampling from a different density function g, where g is exactly like f but scales the sigma of $\mathbf{e}$ by some constant s. Then, the failure probability of P_g is estimated instead of P_f. Since P_g gives a much larger probability ($g(\mathbf{e})$ is much more likely to fail compared to $f(\mathbf{e})$), brute-force Monte-Carlo can be used to obtain an accurate

version of P_g. Converting P_g back to P_f involves multiple runs of P_g using different scaling factors and model fittings. A more detailed presentation can be found in [96].

However, when the dimension N is large, transforming P_g back to P_f is non-trivial (the importance sampling method fails, as noted by [96]). SSS solves this problem by generating a set of scaled probabilities $\{P_{g,i}|i = 0, \cdots, M\}$ with different scaling parameters $\{s_i|i = 0, \cdots, M\}$ to obtain an accurate estimation of the original probability P_f. However, one subtle obstacle persists in adopting SSS in simulating decryption failures. SSS requires f to be a joint Gaussian distribution. In DArL, error growths in products of the form $e \cdot w$ are estimated, where the resulting distribution is not entirely Gaussian (making the case even worse, w may not come from a perfectly uniform distribution). Therefore, SSS is only applied in the case where the weight matrix W is known, i.e., the server finds that a certain filter is used extensively, and want to further optimize the PAHE parameters to reduce computational cost. In such case, the product $e \cdot w$ can be thought as a high-dimensional vector of sigma-scaled Gaussian variables by the entries in w, and perform efficient failure event simulations using SSS.

6.7 Numerical Experiments and Parameter Instantiations

In order to quantitatively assess the impact of DArL-ENSEI-based secure inference, DArL-ENSEI is implemented using the SEAL library [50] in C++. DArL is used to instantiate parameter sets for two sets of plaintext weight filters of different dimensions: 1×2048 and 16×1024, taken from Gazelle. Rigorous accuracy tests are also performed to estimate the smallest NTT modulus p_E for the CIFAR-10 dataset [98] consisting of 50,000 training and 10,000 test images that are of dimension $32 \times 32 \times 3$.

For the sake of fair comparison, in evaluating CIFAR-10, the same network architecture as in [33, 35] are used. The network consists of 7 layers of Conv, ReLU, two layers of average pooling, and a final layer of FC. The accuracy results are obtained using the Tensorflow library [105], and the runtime of homomorphic convolution is recorded on an Intel Xeon E5-2630 2.3 GHz processor.

6.7.1 Concrete Analysis on the Theoretical Bounds

An example derivation of error bounds is demonstrated using parameters provided in Gazelle outlined in Table 6.1. First, note that by having an 18-bit plaintext modulus and a $b = 2^{10}$, two copies of the same ciphertext $([2^{10}\mathbf{u}], [\mathbf{u}])$

Table 6.2: Proposed Selection of Candidate Parameter Sets

p	$\lceil \lg p \rceil$	q	$\lceil \lg q \rceil$	b	Security
12289	14	137438822401	37	p	>214-bit
12289	14	36028797018910721	55	p	>157-bit
65537	16	2199023251457	41	p	>189-bit
65537	16	144115188075835393	57	p	>128-bit
307201	18	$2^{60} - 2^{12} \cdot 63549 + 1$	60	2^{10}	>121-bit

needs to be transferred to conduct a decomposed multiplication, which doubles the amount of communications and computations needed (when p is 22-bit as suggested in Gazelle, it triples resources needed). Before delving into the analysis, it is noted that when $n_o = 1$, no rotations are needed, and the error margin increases significantly as a result of this fact.

First, in the case where $n_o = 8$ (two rows are packed into one ciphertext, since $n_i = 1024$ and $n = 2048$), Eq. (4.6) is used to calculate the norm $||\mathbf{w}|| \leq 55143$, where $w_i \in \mathbf{w}$ is drawn from a uniform distribution in the range $[-512, 512) \cap \mathbb{Z}$ (since $b = 2^{10}$). Then, Eq. (6.10) gives us a bound on e_0 as $7.54 \cdot 4 \cdot 55143 < 1663113$. Applying the same procedure on $||e_{\mathcal{K},w,i}||$ infers that $||e_{\mathcal{K},w,i}|| \leq 1663113 \cdot \sqrt{2048} < 75263903$, and hence $||e_{p_2}|| = \sqrt{60} \cdot 7.54 \cdot 1/\sqrt{2} \cdot 75263903 < 3108269803$. Meanwhile, taking $c = 1.12$ in Lemma. 1 gives us a 2^{-40} bound on the norm of $e_{\mathcal{K},i}$ as $||e_{\mathcal{K},i}|| < 203$, and $||e_{p_3}|| = \sqrt{60} \cdot 7.54/\sqrt{2} \cdot 203 < 8383$. Finally, summing all norms, the total error accumulated in one evaluation of the ciphertext $[\mathbf{u}]$ is less than $3108269803k_0 + 8383k_1$, which is roughly 42 bits.

On the other hand, if $n_o = 1$, the total error contains e_0 only, and this error is bounded by 1663113, which is only around 20-bit. Combined with an 18-bit (or even 22-bit) plaintext modulus, q only needs to be at most 42-bit. Compared to the 60-bit q required when $n_o = 8$, by dynamically adjusting the per-layer parameters, the ciphertext size in a 1×2048 layer can be reduced by as much as 67%. Moreover, if application-specific hardware is adopted, the computational efficiency can also be improved by customizing to a 38-bit modulus, instead of a fixed 64-bit machine word.

Lastly, note that the proposed technique reveals no more information than Gazelle already does. Namely, Bob as the client already knew the filter dimension and the integer b, and these are all the server needs to know to adjust the per-layer parameters accordingly.

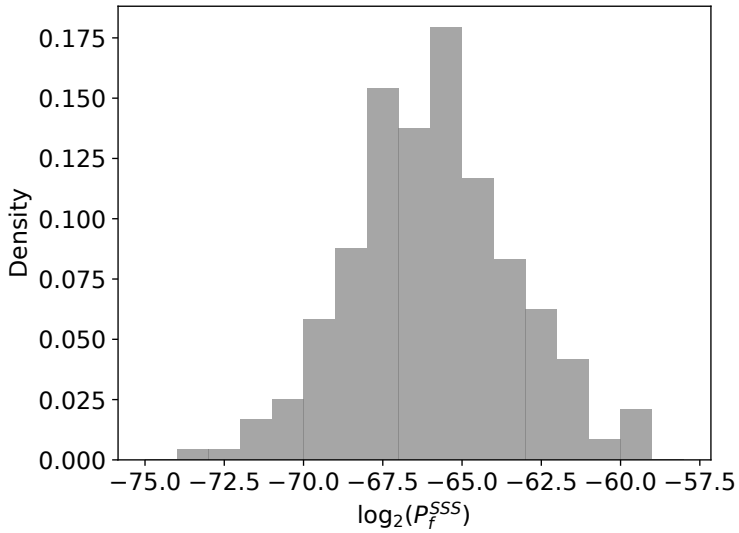

Figure 6.5: The distribution of 240 runs of P_f simulation using SSS.

6.7.2 A Different Plaintext Space

Using the previously demonstrated error bounds, the steps in Alg. 8 are followed to generate a set of plausible parameters with respect to different plaintext modulus up to the 18-bit p suggested in [35], and the results are summarized in Table 6.2 with the security levels estimated using the framework provided by [106].

Obviously, the size of the plaintext space has a strong impact on the parameter sets. First of all, for both 14-bit and 16-bit plaintext spaces, the weight matrix decomposition that is used to prevent noise overflow is unnecessary . This immediately gives us at least 2x (compared to 18-bit p) and even 3x (22-bit p) ciphertext size reduction. Moreover, due to the fact that, essentially, there are less ciphertexts to rotate and add, all of the homomorphic computations are speed up by 2x–3x as well. By fixing $n = 2048$ to allow for efficient packing, a smaller q drastically increases the security of the HE scheme.

6.7.3 Monte-Carlo on BNN-based Secure Inference

The SSS technique is applied on a 10×1024-dimensional packed binary weights pre-trained using the MNIST [97] dataset with a plaintext modulus of $p = 12289$. η_t is set to be slightly less than 40-bit, and step σ in the error distribution χ from $3{\cdot}\sigma$ to $5{\cdot}\sigma$ with a step size of 0.1. 50 K simulations are run in each σ step, which totals to 1 M simulation runs per calculation of P_f. On an Intel Xeon E5-2630 processor with 32 GB of memory, one P_f run roughly takes 2425 seconds.

Table 6.3: Comparison of Communicational and Computational Efficiency Between Different Parameter Instantiations

Method	$\lceil \lg p \rceil$	$\lceil \lg q \rceil$	Ciphertext Size	Fail. Prob.
DArL Empirical	14	54	$\approx 29.2\,\mathrm{KB}$	2^{-60}
DArL Theoretical	14	55	$\approx 29.2\,\mathrm{KB}$	2^{-40}
Gazelle	18	60	$\approx 65.5\,\mathrm{KB}$	2^{-40}
Gazelle	22	60	$\approx 98.4\,\mathrm{KB}$	$> 2^{-40}$

Hence, the computational cost of fast Monte-Carlo is still quite heavy, and it will be up to the server if further optimizations are needed for the particular filter.

Figure 6.5 shows the repeated calculation of 240 P_f estimations using SSS. As noted in [96], the derived P_f tends to follow a normal distribution on the logarithmic scale, and the calculated upper bound is $P_f^{\mathrm{Up}} < 2^{-60.89}$ on the 95% confidence interval. The best-case performance difference between DArL and Gazelle is summarized in Table 6.3. In addition, since the Monte-Carlo simulation indicates that the instantiated parameters are not entirely tight, the efficiency of homomorphic evaluations can be further improved by approximate computing techniques, as suggested in Chapter 4.

6.7.4 The Prediction Accuracy of ENSEI

Given the parameter sets, the classification accuracy of the corresponding instantiated ENSEI is examined. Figure 6.6 illustrates how the prediction accuracy of CNN improves as the bit width increases during secure inference. Here, the inputs are fixed to 10-bit integers, and the filter weights vary in bit widths. In the binary case, binarized weights are used in both the training and inference phase. Observe that while precision does have a strong impact on the overall prediction accuracy, the performance is highly adjustable. In particular, although not shown in Fig. 6.6, the binary-weight can reach a final prediction accuracy of 0.78, which is as accurate as the medium case. This is only around 0.03 (3%) less than the originally reported accuracy in [33]. Consequently, it is observed that for high accuracy instantiation with 6-bit weights, the accuracy is on an equivalent level to the full-bit precision case, and RLWE parameters are instantiated accordingly to examine the impact of different bit widths.

From the accuracy results, what is learned is that instead of setting all plaintext modulus as the same integer like Gazelle, secure inference parameters should be determined in a layer-by-layer manner. For example, depending on the exact cryptographic parameterization, the bit precisions can be decreased in the upper convolutional layers where the input lengths are long, and return

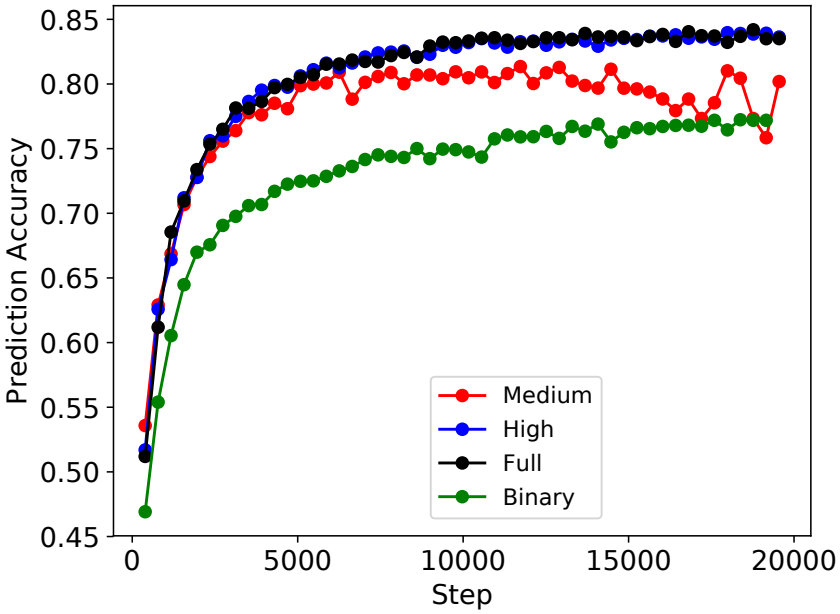

Figure 6.6: CIFAR-10 prediction accuracy with respect to time-domain bit width.

Table 6.4: Proposed Candidate Parameter Sets

Precision	p	$\lceil \lg p \rceil$	q	$\lceil \lg q \rceil$
Binary	12289	14	35184372060161	45
Medium	65537	16	9007199254614017	53
High	307201	18	$2^{60} - 2^{12} \cdot 63549 + 1$	60

to higher precision convolutions in later short layers. Unfortunately, due to the complexity of the analysis and large design space, the efficient automation of the such process remains as an interesting open field of research.

6.7.5 Efficiency Comparison

In this chapter, the results from Section 6.7.4 and Eq. (6.8) are used to choose three sets of RLWE parameters for adopting the fine-grained layer-to-layer (and network-to-network) precision adjustment. The minimal $p_m \geq p_E$ is determined for the p_E required, and then the lattice dimension n and ciphertext modulus q are adjusted accordingly to ensure a 128-bit security with overwhelming decryption success probability (the discrete Gaussian paremeter σ is set to 4).

The instantiated parameters are shown in Table 6.4. Here, the Binary parameter set is for binarized, Medium is for 4-bit, and High is for 6-bit weights.

Table 6.5: Convolution Benchmarks w.r.t Precision Levels

	Input Dim.	Filter Dim.	Precision	t_{setup}	t_{online}	Bandwidth
ENSEI	$(28 \times 28 \times 1)$	$(5 \times 5 \times 5)$	Binary	22.7 ms	2.31 ms	46.0 KB
ENSEI	$(28 \times 28 \times 1)$	$(5 \times 5 \times 5)$	Medium	22.7 ms	2.31 ms	51.2 KB
ENSEI	$(28 \times 28 \times 1)$	$(5 \times 5 \times 5)$	High	26.7 ms	2.60 ms	61.4 KB
Gazelle	$(28 \times 28 \times 1)$	$(5 \times 5 \times 5)$	-	11.4 ms	9.20 ms	130 KB
ENSEI	$(32 \times 32 \times 32)$	$(3 \times 3 \times 32)$	Binary	24.5 ms	18.5 ms	184 KB
ENSEI	$(32 \times 32 \times 32)$	$(3 \times 3 \times 32)$	Medium	24.5 ms	18.5 ms	204 KB
ENSEI	$(32 \times 32 \times 32)$	$(3 \times 3 \times 32)$	High	26.6 ms	20.8 ms	246 KB
Gazelle	$(32 \times 32 \times 32)$	$(3 \times 3 \times 32)$	-	704 ms	195 ms	-

First, it is noted that since ENSEI only requires SIMDScMult, the weight matrices and Galois keys decomposition can also be avoided in section 6.7.2 (w_{pt} and w_{relin} in [35]), and that the ciphertext modulus naturally fits into a 64-bit machine word. Second, for ENSEI involves no ciphertext rotation at all, the entire process of generating and transferring Galois keys is eliminated in the Conv layers (note that this process may still be necessary in the FC layers).

Using the instantiated parameters, the performance comparison of ENSEI and Gazelle on a set of convolution benchmarks are summarized in Table 6.5. Here, t_{setup} is the amount of time consumed by procedures that do not involve user inputs (e.g., initializing SEAL, reading in filters), and t_{online} refers to the time for input-dependent steps. In Table 6.5, it is observed that by using ENSEI, the online convolution time is reduced by 4x–10x across all precisions. Since the implementation in this work uses a different library from Gazelle, for smaller convolutions, the resulting setup time is not as fast. However, it is noticed that the setup time of ENSEI is (almost) invariant from the image, filter sizes, and color channels. Therefore, on larger benchmarks, a 31x reduction in setup and nearly 10x reduction in online time are observed. The reductions are primarily due to the fact that the sophisticated packing techniques used in Gazelle is not optimized for larger input and output channels. In contrast, the setup time for ENSEI remains almost constant, and owing to the efficient FDC technique, the online time of ENSEI scales much slower than Gazelle. Moreover, similar trends in the communication bandwidth occur, where the elimination of Galois keys and window sizing, along with the fine-tuned precision control, brings us more than 2x reduction in the communication bandwidth across all convolution sizes.

By adopting the less-accurate but more efficient activation functions such as the square activation function used in [33, 35], an execution of secure inference using a relatively deep neural network on the CIFAR-10 dataset can be completed within 0.378 s, making ENSEI-based CNN one of the most practical MLaaS schemes to date.

6.8 Summary

In this work, DArL-ENSEI is proposed to dynamically optimize the parameter instantiations for frequency-domain oblivious convolution that accelerates CNN-based secure inference. In DArL, a theoretical approach to systematically characterize the error growth in different network settings is established, and developed a Monte-Carlo-based sampling method to tighten the bound on decryption failures. In ENSEI, by using NTT, it is demonstrated that oblivious convolution can be built on any encryption scheme that is additively homomorphic. Then, DArL-ENSEI is integrated to one of the most recent work on secure inference, and observed 4x–10x reduction in online inference time, and as much as 31x reduction in setup time. It is demonstrated that PAHE-based secure inference can be simple and practical for real-world datasets.

Chapter 7

Conclusion

In conclusion, this dissertation brings novel design concepts into the cross-layer design exploration of advanced secure protocols for the age of cloud computing. In what follows, three contributions shall be briefly summarized in Section 7.1. Discussions and future prospects are also provided in Section 7.2 and 7.3.

7.1 Summary

First, focusing on better hardware primitives for LWE cryptography, multiplier architectures are proposed. By exploiting the algebraic structures of generic LWE instances, modulo reduction circuits are squashed, and approximate multipliers are adopted. For RLWE, with application-specific pipeline architectures, it is discovered that many proposed optimization techniques, such as Montgomery-like reduction and lazy reduction, are not necessary, resulting in a simplistic algorithmic design. Energy efficiencies for LWE-based cryptosystems are improved by 4–21x, with an extra 27%–46% decryption bandwidth reductions.

Second, design techniques are examined for the proposed secure email filtering scheme. The observation is that, even with the most efficient PHE scheme and simplest filtering scheme, homomorphic email filtering is still a demanding task. Therefore, in addition to the proposed quantization and parallel filtering methods, a 2048-bit hardware multiplier is also utilized to reduce the latency and power consumption of the system as much as possible. Experiment results indicate that such secure filter can only be considered as practical on a dedicated hardware platform, with a filtering throughput of 0.5 s per email, and a power consumption of 50 mW for the multiplier unit.

Lastly, neural-network-based secure inference schemes are examined. As such schemes are generally dependent on (either lattice or factoring) homo-

morphic encryption schemes, previously proposed techniques can be easily integrated. Furthermore, a frequency-domain secure convolution protocol is proposed to minimize the amount of homomorphic operations. Along with rigorous parameter analysis and dynamic instantiation techniques, the online and off-line computation time of secure inference can be reduced by up to 31x and 4–10x, respectively. The communication bandwidth is also reduced by 2–3x across all the linear layers of convolutional neural networks.

7.2 Discussions

As observed in the above summary, this dissertation touches on a large number of drastically different secure needs, cryptographic constructions and hardware architectures. However, the design concept remains consistent. For each application, cryptographic tools are selected with proper parameter instantiations, and new hardware platforms are architected to ensure efficient protocol execution. In this whole design process, optimizations are conducted without strict layer scopes, and techniques from different layers are applied in an integrated manner.

At this point, it is obvious that HSPs, or any secure protocol in general, carry heavy computational and communicational burdens. Therefore, developing a practical HSP shares many similarities to the design of early electronic computing systems. At the time, the concept of abstraction layers was not well-developed, and cross-layer techniques are naturally adopted by computer architects. The goal of this dissertation is to show that, the same design concept should also be applied in the case of HSPs, and by adopting this cross-layer concept, significant speed and efficiency gains are attainable. Moreover, as outlined in Section 7.3, the cross-layer point of view opens up new fields of research that greatly reduce the cost of adopting HSPs in practice. It is envisioned that, HSPs, much like modern-day smartphones, will become an essential commodity in the future, and work as means to protect personal privacy and safety in the age of cloud computing.

7.3 Future Prospects

In closing this dissertation, open questions and potential research directions are summarized.

- **HSP Compiler**: As the number of cryptographic primitives and target applications grows, fine-tuning the entire system becomes increasingly hard. Meanwhile, due to the high computational and communicational

cost of general multi-party secure protocols, designs without cross-layer optimizations can turn out to have prohibitive overheads. Consequently, for higher design layers, compilers need to be implemented such that enough abstraction are provided for the efficient design of application-specific protocols. Until very recently, high-level compilers for HE started to emerge [8, 107]. However, existing works generally only focus on compiling programs using HE-based instructions (e.g., homomorphic addition and multiplication), instead of a mixed design of a heterogeneous HSP employing different types of cryptographic primitives. As observed in Gazelle and ENSEI, the mixing of protocols can provide significant performance improvement over a single-protocol scheme, and a higher-level compiler shall be aware of such complex design trade-offs.

- **Better Hardware Primitives for Modular Reduction**: Although factoring-based PKE schemes have already put forth a strong demand for modular arithmetic, optimizations for modular arithmetic are generally conducted in the software layer. For example, Barrett reduction, Montgomery reduction and lookup-table-based approaches are the classic examples. However, PKE schemes are generally only executed once to exchange a private key and are not as resource-hungry as HE schemes. Moreover, factoring-based schemes generally target on single large integer, instead of a large number of small integers as in lattice-based systems. At the moment, it is not known if specialized hardware primitives exist for modular arithmetic. However, given the flexibility and post-quantum security of (R)LWE-based schemes, especially in designing efficient HSPs, it is expected that hardware-accelerated modular arithmetic is one of the key components in fulfilling practical HSPs.

- **Process-In-Memory Architectures for (R)LWE Cryptography**: The fundamental assumption of LWE cryptosystems is that certain problems in lattices are hard to solve. As mentioned in Chapter 3, in constructing a lattice-like structure, both the random matrix approach in LWE or the random polynomial approach in RLWE require a relatively large number of integers (10^3–10^5), which result in large public key sizes. In addition to the fact that (R)LWE computational units are relatively simple, the result is that moving public keys from memory to processor and back to the memory become the main performance bottleneck of lattice cryptology, for both traditional and advanced cryptographic constructions. Hence, in-memory computing can be a promising hardware solution, as data movements can be minimized on such computing platforms.

Bibliography

[1] S. Garg, P. Mohassel, and C. Papamanthou, "TWORAM: efficient Oblivious RAM in two rounds with applications to searchable encryption," in *Annual Cryptology Conference*. Springer, 2016, pp. 563–592.

[2] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.

[3] D. E. Denning, "Digital signatures with RSA and other public-key cryptosystems," *Communications of the ACM*, vol. 27, no. 4, pp. 388–392, 1984.

[4] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2000, pp. 539–556.

[5] Y. Gahi, M. Guennoun, and K. El-Khatib, "A secure database system using homomorphic encryption schemes," *arXiv preprint arXiv:1512.03498*, 2015.

[6] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1006–1018.

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

[8] Galois, https://galois.com/project/ramparts/, 2018, accessed: 2019-02-05.

[9] Duality Technologies, https://duality.cloud/, 2018, accessed: 2019-02-05.

[10] C. Peikert *et al.*, "A decade of lattice cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.

[11] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 99–108.

[12] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[13] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 197–206.

[14] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

[16] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[17] A. Khedr, G. Gulak, and V. Vaikuntanathan, "Shield: scalable homomorphic implementation of encrypted data-classifiers," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2848–2858, 2016.

[18] 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.

[19] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem." *IACR Cryptology EPrint Archive*, vol. 2012, p. 688, 2012.

[20] C. Peikert, "Lattice cryptography for the internet," in *International Workshop on Post-Quantum Cryptography*. Springer, 2014, pp. 197–219.

[21] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 553–570.

[22] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 738–755.

[23] 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.

[24] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 617–635.

[25] 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.

[26] Z. Brakerski and R. Perlman, "Order-LWE and the hardness of ring-LWE with entropic secrets," IACR Cryptology ePrint Archive, 2018: 494, Tech. Rep., 2018.

[27] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 2009, pp. 333–342.

[28] 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.

[29] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in helib," Cryptology ePrint Archive, Report 2018/244, Tech. Rep., 2018.

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

[31] C. P. Ferrette, "To outsource or not to outsource: IAs and emails," *Compliance Reporter*, 2007.

[32] 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.

[33] 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.

[34] 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.

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

[36] E. Makri, D. Rotaru, N. P. Smart, and F. Vercauteren, "Epic: efficient private image classification (or: learning from the masters)," in *Cryptographers' Track at the RSA Conference*. Springer, 2019, pp. 473–492.

[37] 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.

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

[39] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[40] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Intl. Conf. Theory and Applications of Cryptographic Techniques*, 1999, pp. 223–238.

[41] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 617–640.

[42] S. Halevi and V. Shoup, "Algorithms in helib," in *Annual Cryptology Conference*. Springer, 2014, pp. 554–571.

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

[44] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *International Conference on Cryptology and Network Security*. Springer, 2016, pp. 124–139.

[45] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, 2015.

[46] N. C. Dwarakanath and S. D. Galbraith, "Sampling from discrete gaussians for lattice-based cryptography on a constrained device," *Applicable Algebra in Engineering, Communication and Computing*, vol. 25, no. 3, pp. 159–180, 2014.

[47] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 162.

[48] S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld, "Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance," *Journal of Cryptology*, vol. 31, no. 2, pp. 610–640, 2018.

[49] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," in *International Cryptology Conference*. Springer, 2014, pp. 297–314.

[50] 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.

[51] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 868–886.

[52] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 465–482.

[53] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Secure pattern matching using somewhat homomorphic encryption," in *Proceedings of the 2013 ACM workshop on Cloud computing security workshop*. ACM, 2013, pp. 65–76.

[54] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *Designs, codes and cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

[55] W. Lu, S. Kawasaki, and J. Sakuma, "Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1163, 2016.

[56] NIST, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, 2018, accessed: 2018-04-20.

[57] T. Oder, T. Güneysu, F. Valencia, A. Khalid, M. O'Neill, and F. Regazzoni, "Lattice-based cryptography: From reconfigurable hardware to ASIC," in *Integrated Circuits (ISIC), 2016 International Symposium on*. IEEE, 2016, pp. 1–4.

[58] 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.

[59] 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.

[60] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 371–391.

[61] M. Schneider, "Sieving for shortest vectors in ideal lattices," in *International Conference on Cryptology in Africa*. Springer, 2013, pp. 375–391.

[62] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 512–529.

[63] J. Howe, T. Oder, M. Krausz, and T. Güneysu, "Standard lattice-based key encapsulation on embedded devices," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 372–393, 2018.

[64] Y. Chen and P. Q. Nguyen, "Bkz 2.0: Better lattice security estimates," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 1–20.

[65] J. H. Cheon, D. Kim, J. Lee, and Y. S. Song, "Lizard: Cut off the tail!//practical post-quantum public-key encryption from LWE and LWR." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1126, 2016.

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

[67] *PrimeTime PX H-2013.06*, Synopsys, Inc.

[68] S. Bian, M. Hiromoto, and T. Sato, "DWE: Decrypting learning with errors with errors," in *Design Automation Conference (DAC), 2018 55th ACM/EDAC/IEEE*. IEEE, 2018, pp. 1–6.

[69] S. Hashemi, R. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2015, pp. 418–425.

[70] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[71] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 575–584.

[72] C. Z. Mooney, *Monte carlo simulation*. Sage Publications, 1997, vol. 116.

[73] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *ACM Cloud Computing Security Workshop*, 2009, pp. 85–90.

[74] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *NDSS*, vol. 20, 2012, p. 12.

[75] Fujitsu Research Institute, "Personal data in the cloud: A global survey of consumer attitudes," http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu_personal-data-in-the-cloud.pdf, 2010.

[76] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 506–522.

[77] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *International Conference on Applied Cryptography and Network Security*, 2013, pp. 102–118.

[78] M. Barbosa and P. Farshim, "Delegatable homomorphic encryption with applications to secure outsourcing of computation." in *CT-RSA*, vol. 12, 2012, pp. 296–312.

[79] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive Bayes–which naive Bayes?" in *CEAS*, vol. 17, 2006, pp. 28–69.

[80] V. Bindschaedler, M. Naveed, X. Pan, X. Wang, and Y. Huang, "Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward," in *ACM SIGSAC Conf. Computer and Communications Security*, 2015, pp. 837–849.

[81] S. Bian, M. Hiromoto, and T. Sato, "Scam: Secured content addressable memory based on homomorphic encryption," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 984–989.

[82] Y. Doröz and B. Sunar, "Flattening NTRU for evaluation key free homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2016, p. 315, 2016.

[83] J. Großschädl, "High-speed rsa hardware based on barret ' s modular reduction method," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 191–203.

[84] G. C. Chow, K. Eguro, W. Luk, and P. Leong, "A Karatsuba-based Montgomery multiplier," in *Intl. Conf. Field Programmable Logic and Applications*, 2010, pp. 434–437.

[85] S.-R. Kuang, J.-P. Wang, K.-C. Chang, and H.-W. Hsu, "Energy-efficient high-throughput montgomery modular multipliers for RSA cryptosystems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 11, pp. 1999–2009, 2013.

[86] J. Portilla, A. Otero, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter, and P. Langendörfer, "Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors," *International Journal of Distributed Sensor Networks*, vol. 6, no. 1, p. 740823, 2010.

[87] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 353–373.

[88] D. Cash and S. Tessaro, "The locality of searchable symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 351–368.

[89] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 639–654.

[90] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 668–679.

[91] G. Asharov, M. Naor, G. Segev, and I. Shahaf, "Searchable symmetric encryption: Optimal locality in linear space via two-dimensional balanced allocations," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016, pp. 1101–1114.

[92] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 2092–2100.

[93] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[94] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[95] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of*

the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018, pp. 1209–1222.

[96] S. Sun, X. Li, H. Liu, K. Luo, and B. Gu, "Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1096–1109, 2015.

[97] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[98] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[99] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[100] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv preprint arXiv:1312.5851*, 2013.

[101] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[102] R. C. Agarwal and C. S. Burrus, "Number theoretic transforms to implement fast digital convolution," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 550–560, 1975.

[103] H. Nussbaumer, "Fast polynomial transform algorithms for digital convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 2, pp. 205–215, 1980.

[104] T. Toivonen and J. Heikkila, "Video filtering with fermat number theoretic transforms using residue number system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 1, pp. 92–101, 2006.

[105] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[106] 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.

[107] E. Crockett, C. Peikert, and C. Sharp, "Alchemy: A language and compiler for homomorphic encryption made easy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1020–1037.

List of Publications

Journals

1. **Song Bian**, Masayuki Hiromoto, Takashi Sato, Hardware-Accelerated Secured Naive Bayesian Filter Based on Partially Homomorphic Encryption, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E102-A, No.2, pp.430-439, Febuary 2019.

2. Yuki Tanaka, **Song Bian**, Masayuki Hiromoto, and Takashi Sato, Coin Flipping PUF: A Novel PUF with Improved Resistance Against Machine Learning Attacks IEEE Transactions Circuits and Systems II: Express Briefs, Vol.65, No.5, pp.602-606, May 2018.

3. Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, Utilization of Path-Clustering in Efficient Stress-Control Gate Replacement for NBTI Mitigation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E100-A, No.7, pp.1464-1472, July 2017.

4. **Song Bian**, Shumpei Morita, Michihiro Shintani, Hiromitsu Awano, Masayuki Hiromoto, and Takashi Sato, Identification and Application of Invariant Critical Paths Under NBTI Degradation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E100-A, No.12, pp.2797-2806, Dec. 2017.

5. **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, and Takashi Sato, Fast Estimation of NBTI-Induced Delay Degradation Based on Signal Probability, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E99-A, No.7, pp.1400-1409, July 2016.

6. Kai-wen Hsu, He Ren, Robert J. Agasie, **Song Bian**, Yoshio Nishi, and Leon J. Shohet, Effects of Neutron Irradiation of Ultra-thin HfO_2 Films. Applied Physics Letters, Vol.104, No.3, 032910, 2014.

Peer-reviewed conference

1. **Song Bian**, Masayuki Hiromoto, Takashi Sato, Filianore: Better Multiplier Architectures for LWE-based Post-Quantum Key Exchange, Proc. of ACM/IEEE Design Automation Conference (DAC), June 2019 (to appear).

2. **Song Bian**, Masayuki Hiromoto, Takashi Sato, DArL: Dynamic Parameter Adjustment for LWE-based Secure Inference, in Proc. of Design, Automation and Test in Europe (DATE) (Florence, Italy), pp.1718-1723, March 2019.

3. **Song Bian**, Masayuki Hiromoto, and Takashi Sato, Towards Practical Homomorphic Email Filtering: A Hardware-Accelerated Secure Naive Bayesian Filter, in Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC) (Tokyo Odaiba Waterfront, Japan), pp.621-626, January 2019.

4. **Song Bian**, Masayuki Hiromoto, Takashi Sato, DWE: Decrypting Learning with Errors with Errors, in Proc. of ACM/IEEE Design Automation Conference (DAC) (San Francisco, CA), 10.3, June 2018.

5. Yuki Tanaka, **Song Bian**, Masayuki Hiromoto, Takashi Sato, Coin Flipping PUF: a New PUF with Improved Resistance Against Machine Learning Attacks, in Proc. of IEEE International Symposium on Circuits and Systems (ISCAS) (Florence, Italy), pp.1-6, May 2018.

6. Zuitoku Shin, Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, A Study on NBTI-induced Delay Degradation Considering Stress Frequency Dependence, in Proc. of International Symposium on Quality Electronic Design (ISQED) (Santa Clara, CA), pp.251-256, March 2018.

7. Zuitoku Shin, Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, Comparative Study of Delay Degradation Caused by NBTI Considering Stress Frequency Dependence, in Proc. of the 21st Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2018) (Kunibiki Messe, Matsue, Japan), pp.194-199, March 2018.

8. Yuki Tanaka, **Song Bian**, Masayuki Hiromoto, Takashi Sato, A PUF Based on the Instantaneous Response of Ring Oscillator Determined by the Convergence Time of Bistable Ring, in Proc. of the 21st Workshop

on Synthesis And System Integration of Mixed Information technologies (SASIMI2018) (Kunibiki Messe, Matsue, Japan), pp.30-34, March 2018.

9. Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, Efficient Exploration of Worst Case Workload and Timing Degradation under NBTI, in Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC) (Jeju Island, Korea), pp.631-636, January 2018.

10. **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, LSTA: Learning-Based Static Timing Analysis for High-Dimensional Correlated On-Chip Variations, in Proc. of ACM/IEEE Design Automation Conference (DAC) (Austin, TX), 73.3, June 2017.

11. **Song Bian**, Masayuki Hiromoto, Takashi Sato, SCAM: Secured Content Addressable Memory Based on Homomorphic Encryption, in Proc. of Design, Automation and Test in Europe (DATE) (Lausanne, Switzerland), pp.984-989, March 2017.

12. Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, Comparative Study of Path Selection and Objective Function in Replacing NBTI Mitigation Logic, Proc. of International Symposium on Quality Electronic Design (ISQED), March 2017.

13. **Song Bian**, Michihiro Shintani, Zheng Wang, Masayuki Hiromoto, Anupam Chattopadhyay, Takashi Sato, Runtime NBTI Mitigation for Processor Lifespan Extension via Selective Node Control, in Proc. of IEEE Asian Test Symposium (ATS) (Hiroshima, Japan), pp.234-239, November 2016.

14. Shumpei Morita, **Song Bian**, Michihiro Shintani, Masayuki Hiromoto, Takashi Sato, Path Grouping Approach for Efficient Candidate Selection of Replacing NBTI Mitigation Logic, in Proc. of the 20th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2016) (Kyoto Research Park, Kyoto, Japan), pp.242-247, October 2016.

15. **Song Bian**, Michihiro Shintani, Shumpei Morita, Hiromitsu Awano, Masayuki Hiromoto, Takashi Sato, Workload-Aware Worst Path Analysis of Processor-Scale NBTI Degradation, in Proc. of Great Lakes Symposium on VLSI (GLSVLSI) (Boston, MA), pp.203-208, May 2016.

16. **Song Bian**, Michihiro Shintani, Zheng Wang, Masayuki Hiromoto, Anupam Chattopadhyay, Takashi Sato, Mitigation of NBTI-induced Timing

Degradation in Processor, in Proc. of ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU) (Santa Rosa, CA), pp.50-55, March 2016.

17. **Song Bian**, Michihiro Shintani, Shumpei Morita, Masayuki Hiromoto, Takashi Sato, Nonlinear Delay-Table Approach for Full-Chip NBTI Degradation Prediction, in Proc. of International Symposium on Quality Electronic Design (ISQED) (Santa Clara, CA), pp.307-312, March 2016.

Awards

1. Dean's Honor List at the University of Wisconsin-Madison, 2012-2014
2. Edgar H. and Laverne R. Krainer Memorial Scholarship, September 2013
3. Graduated with highest distinction at the University of Wisconsin-Madison, May 2014
4. ACM/IEEE GLSVLSI 2016 Student Travel Award, May 2016
5. IPSJ DA Symposium 2015 Excellent Student Presentation Award, September 2016
6. IPSJ DA Symposium 2016 Excellent Student Presentation Award, August 2017
7. IPSJ DA Symposium 2016 Best Paper Award, August 2017
8. IPSJ Computer Science Research Award for Young Scientists, August 2017
9. IEEE Kansai Section Student Paper Award, February 2018
10. IEICE VLD Excellent Student Author Award for ASP-DAC 2019, March 2019