DArL: Dynamic Parameter Adjustment for LWE-based Secure Inference

Song Bian, Masayuki Hiromoto, and Takashi Sato
Deptartment of Communications and Computer Engineering, School of Informatics, Kyoto University
Yoshida-hon-machi, Sakyo, Kyoto 606-8501, Japan
E-mail: paper@easter.kuee.kyoto-u.ac.jp

***Abstract*—Packed additive homomorphic encryption (PAHE) based secure neural network inference is attracting increasing attention in the field of applied cryptography. In this work, we seek to improve the practicality of LWE-based secure inference by dynamically changing the cryptographic parameters depending on the underlying architecture of the neural network. First, we develop and apply theoretical methods to closely examine the error behavior of secure inference, and propose parameters that can reduce as much as 67% of ciphertext size when smaller networks are used. Second, we use rare-event simulation techniques based on the sigma-scale sampling method to provide tight bounds on the size of cumulative errors drawn from (somewhat) arbitrary distributions. Finally, in the experiment, we instantiate an example PAHE scheme and show that we can further reduce the ciphertext size by 3.3x if we adopt a binarized neural network architecture, along with a computation speedup of 2x–3x.**

I. Introduction

The quantum-secure learning with errors (LWE) conjecture appears to be one of the most popular and powerful cryptographic hardness assumptions that are established in recent years. Along with its provable security reductions, LWE is known to enable efficient implementation of classic and advanced cryptographic primitives, including (somewhat) fully homomorphic encryption (FHE) [1]–[3], identity- and attribute-based encryptions [4], [5], functional encryption [6], and secure multi-party computation [7], [8].

Despite theoretical advances, the concrete security and correctness analyses are somewhat left out in prior arts. This is partially due to the fact that current LWE parameters are usually instantiated against best-known attacks, instead of in the theoretically secure domain [9]. Recent advances establish evaluation frameworks (e.g., [10]) to allow for fast and thorough analysis on the bit security of a particular set of parameters. On the other hand, the correctness analysis is still somewhat case-by-case. For example, even in a simple key exchange scheme, substituting a standard discrete Gaussian distribution by a binomial distribution requires a sophisticated correctness analysis on the target product distribution [11]. For advanced schemes such as homomorphic encryption (HE), the error analysis is further complicated by the fact that the resulting error distribution is dependent on the exact homomorphic evaluation being carried out.

In this work, we propose DArL, a dynamic parameter adjustment framework for LWE-based HE schemes. DArL targets on exploiting the unused correctness margin to improve the computational and communicational efficiency of HE without security loss. The proposed framework consists of two main techniques: i) application-specific theoretical derivations for error bounds, and ii) a fast empirical method for simulating rare failure events. As a proof-of-concept, we focus on the BFV cryptosystem [2], [3] adopted in Gazelle [12], where the HE scheme serves as a fundamental design element in implementing secure neural network inference. First, we apply theoretical bounds to examine the error behavior of BFV in Gazelle, and minimize the unused error margin by dynamically switching parameter sets during the inference stage. Second, we adopt the sigma-scaled sampling technique [13] to obtain the exact decryption failure probability, which is important in fine-tuning the parameters by techniques such as approximate computing [14]. In the experiment, we apply DArL on the parameter sets instantiated in Gazelle, and inspect its error behavior when classifying specific datasets such as MNIST [15]. By dealing with a smaller ciphertext modulus q, we observe that for some network architecture, we can reduce the ciphertext size by as much as 67%. More importantly, by switching to hardware-friendly network architectures such as binarized neural network (BNN) [16], a further 2x–3x ciphertext size reduction and computation speedup are simultaneously attainable. We highlight our main contributions as follows.

- **Dynamic Parameter Adjustment**: Existing HE applications generally assume one set of parameters being used throughout the whole evaluation process. In this work, by realizing that different neural network architecture can give rise to significant performance gap, we conduct rigorous analyses on the error behaviors in the linear kernel used in Gazelle [12], and propose a set of candidate parameters 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 [14] where a Monte-Carlo approach was employed. While empirical simulation provides a tighter bound as shown in [14], it is clearly not practical to simulate a failure probability of 2^{-40} using brute-force Monte-Carlo. In this work, we use the sigma-scaled sampling [13] to further optimize the runtime parameters of DArL.
- **Architecture Exploration for Secure Neural Networks**: As a separate contribution, we compare different types of deep neural network (DNN) architectures to explore the design trade-off of network architecture in the context of secure applications. Specifically, we compare the traditional convolutional neural network implemented in [12]

978-3-9819263-2-3/DATE19/©2019 EDAA

with BNN. In the experiment, we find that BNN requires much smaller plaintext modulus, and thereby smaller parameter sets.

The rest of this paper is organized as follows. First, in Section II, we discuss the basics of packed additive homomorphic encryption (PAHE) and Gazelle. Second, we provide theoretical error analyses in Section III. Third, we describe DArL in detail, and discuss the application of fast empirical methods for even tighter bound analysis Section IV. Fourth, we propose a set of parameters and conduct experiment on BNN in Section V. Finally, our work is summarized in Section VI.

II. Preliminaries

A. Error Distributions

In this work, we use χ_σ^n to refer to n independent samples each drawn from χ_σ. We will extensively use the following two tail bounds provided by Lindner and Peikert [9].

Lemma 1: (Lemma 2.1 in [9]) 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 $n \geq 1$, we have

$$\Pr[\|\chi_\sigma^n\| \geq c \cdot \sigma\sqrt{n}] \leq C^n. \tag{1}$$

Lemma 2: (Lemma 2.2 in [9]) For any real $\sigma > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have

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

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

B. The BFV Cryptosystem

The BFV cryptosystem is a standard ring LWE (RLWE)-based FHE scheme from [1], [3]. The cryptosystem is implemented by Chen et al. in SEAL [17]. Here, we give a short overview on the basic properties and error behaviors of BFV.

Similar to Gazelle [12], we use $[\mathbf{u}]$ to refer 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, thanks to the Smart-Vercaueren packing technique [18], 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. Due to space limitation, we leave out the details on the encryption and decryption functions in BFV. In short, for the encryption of a packed polynomial m containing the elements in $\mathbf{u}$, a BFV ciphertext is structured as a vector of two polynomials $(c_0, c_1) \in \mathcal{R}_q^2$, where

$$\begin{aligned} c_0 &= -a, \\ c_1 &= a \cdot s + \frac{q}{p}m + e_0. \end{aligned} \tag{3}$$

Here, a is a uniformly sampled polynomial, and s, e_0 are polynomials whose coefficients drawn from χ_σ (i.e., $\mathbf{s}, \mathbf{e}_0 \leftarrow \chi_\sigma^n$, assuming $\mathbf{s}, \mathbf{e}_0$ are vectors containing the n coefficients in s and e_0). The decryption follows simply by computing $\frac{p}{q}(c_0 s + c_1)$ and rounding off the errors.

The scheme remains correct when $e \ll q/p$, and we can see that as long as $k \cdot e \ll q/p$ 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 $e_u + e_v \ll q/p$.

TABLE I
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

The parameters instantiated by Gazelle are summarized in Table I, 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.

C. Rotating the Plaintext Slots in BFV

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 introduces no additional errors to the ciphertext, we get some additive error components from switching the ciphertext, such that it is decryptable under the original secret key s. It is noted that this additive error, denoted as η_{rot} in [12], is independent from the original errors contained in the ciphertext $[\mathbf{u}]$.

More concretely, as demonstrated in [1], if we rotate a ciphertext $[\mathbf{u}]$ 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

$$\mathsf{Dec}([\mathbf{v}]) = 2 \cdot (\mathsf{Decomp}_b([\mathbf{u}]) \cdot \mathbf{e}_\mathcal{K}) + \mathsf{Dec}([\mathbf{u}]), \tag{4}$$

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 we take $b = 2$, 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 [2], [19], [20].

D. Gazelle: Secure Neural Network Inference

Gazelle [12] represents one of the most recent advances in the line of prior arts [21], [22] on designing secure inference. The general protocol and architecture of Gazelle is outlined in Fig. 1, where Bob wants to classify some input (e.g., 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. Due to space limitation, we omit a formal discussion on Gazelle and refer the readers to the original works for more details [12]. Here, we focus on the error behavior of the linear Conv and FC layers, which are implemented by the PAHE-based matrix-vector multiplication techniques.

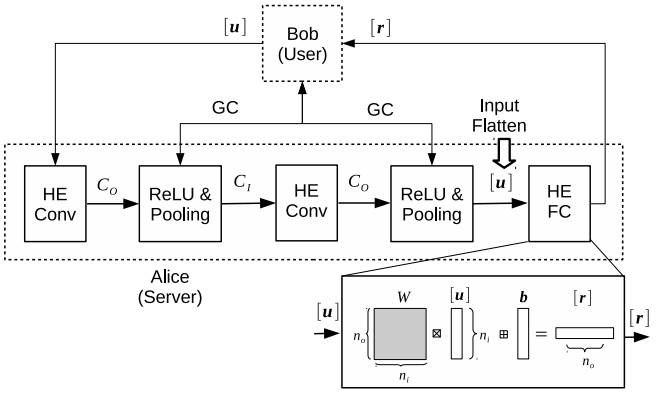

Fig. 1. An example of the architecture in Gazelle with two Conv layers, two non-linear layers and one FC layer. The FC layer, much like the Conv layers, is internally a homomorphic matrix-vector product.

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. 1. The basic approach (called naïve method in [12]) for computing an inner product is as follows. For some row vector $\mathbf{w}$ in the weight matrix W and ciphertext $[\mathbf{u}]$, we homomorphically compute the coefficient-wise product vector $[\mathbf{v}] = [\mathbf{u}] \boxdot \mathbf{w}$, 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 $\sum_i^{n-1} v_i$, we i) create a rotated version of $[\mathbf{v}]$ by a step size of $k = n/2$, and ii) compute a coefficient-wise homomorphic addition between $[\mathbf{v}]$ and $\mathsf{rot}([\mathbf{v}], k)$. 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 $i \in [1, \log_2(n)] \cap \mathbb{Z}$), we obtain the desired sum 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, 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 [12], 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.

$$[\mathbf{v}] = \sum_{i=0}^{n_o-1} \mathbf{w}_i \boxdot \mathsf{rot}([\mathbf{u}], i) \tag{5}$$

$$= \mathbf{w}_0 \boxdot [\mathbf{u}] + \cdots + \mathbf{w}_{n_o-1} \boxdot \mathsf{rot}([\mathbf{u}], n_o - 1), \tag{6}$$

$$[\mathbf{r}] = \sum_{i=1}^{\lg(n/n_o)} \mathsf{rot}\left([\mathbf{v}], \frac{n}{2^i}\right), \tag{7}$$

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), we first rotate $[\mathbf{u}]$ 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 we can then use the basic approach to rotate this packed result to obtain the final sum, as in Eq. (7).

III. Modeling the Matrix Multiplication Error

In both Conv and FC layers, we are basically dealing with matrix-vector multiplication 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.

A. 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), we first take a look at the error behavior in the hybrid matrix-vector multiplication in the FC layer. Here, we have a rectangular matrix $W \in \mathbb{Z}^{n_o \times n_i}$ with the important property that $n_o \ll n_i$, and an input ciphertext $[\mathbf{u}]$ as inputs. Without loss of generality, we assume 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. (3). 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 [12], assuming a fresh $[\mathbf{u}]$ is η_0-bounded, the error growth from this operation is

$$\begin{aligned}&((\eta_0 + \eta_{\mathrm{rot}}) \cdot \eta_{\mathrm{mult}} \cdot n_o + \eta_{\mathrm{rot}}) \cdot \lg(n/n_o) \\ &= k_0\eta_0\eta_{\mathrm{mult}} + k_0\eta_{\mathrm{rot}}\eta_{\mathrm{mult}} + k_1\eta_{\mathrm{rot}},\end{aligned} \tag{8}$$

where we set $k_0 = n_o \lg(n/n_o)$, $k_1 = \lg(n/n_o)$. In Eq. (8), we assume that a rot induces an additive η_{rot} factor, and multiplying the weight scales the error by η_{mult}. We found no formal discussion on the correctness proof in Gazelle [12], and a straightforward calculation of Eq. (8) is clearly over-pessimistic in terms of the error bounds. In what follows, we develop theoretical bounds for each error term in Eq. (8).

B. The Error $\eta_0 \cdot \eta_{\mathrm{mult}}$

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 $\eta_0 \cdot \eta_{\mathrm{mult}}$ 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 work, we take a combined approach. We first assume 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 IV, we reaffirm our theoretical analyses through empirical approaches.

Assuming $w \leftarrow U_b$ is uniformly random over the integers in $[-b/2, b/2) \cap \mathbb{Z}$, we are interested in the L_2-norm of the

coefficients in $e_{p_1} = e_0 \cdot w$. Similar to [11], we observe that the coefficients in e_p is in the form

$$(e_{p_1})_i = \sum_{i=0}^{n-1} \pm \left((e_0)_i \cdot (w)_{(i-j) \bmod n}\right), \quad (9)$$

where we use $(x)_i$ to depict 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. (9) can also be written as an inner product between two vectors $\mathbf{e}_0 = [(e_0)_0, \cdots, (e_0)_{n-1}]$ and $\mathbf{w} = [(w)_0, \cdots, (w)_{n-1}]$, we can apply Lemma 2 and obtain a bound on the L_2-norm as

$$\Pr[||\langle \mathbf{e}_0, \mathbf{w} \rangle|| > T\sigma||\mathbf{w}||] < 2 \cdot \exp(-T^2/2). \quad (10)$$

In this work, we use an asymptotic bound of 2^{-40} ($2 \cdot \exp(-T^2/2) \leq 2^{-40}$). This bound translates to a T value of roughly 7.54, and we use this T for all the subsequent analyses. Since Eq. (10) also requires a bound on the L_2-norm of the vector $\mathbf{w}$, we can use the Chernoff-Cramer inequality (e.g., applied in [11]) to set up a series of independent and identically distributed random variables $X_0 = (w)_0^2, \cdots, X_{n-1} = (w)_{n-1}^2$. The inequality states that

$$\Pr\left[\sum_{i=0}^{n-1} X_i \geq n\mu + \beta\right] \leq \exp\left(-\beta t + n \ln(M_{U_b}(t))\right), \quad (11)$$

where μ is the mean of X_i's, n is the lattice dimension, M_{U_b} is the moment generating function of U_b, and we can optimize t to find the smallest β that satisfies the above equation. Last but not least, we point out that Eq. (9) 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, we can simply perform a summation in the form $||e_{p_2}|| = \sqrt{\sum_{i=0}^{n-1} ||(e_{p_2})_i^2||}$, which is basically $\sqrt{n} \cdot ||(e_{p_2})_i||$, since all coefficients in e_{p_2} are independent random variables by design.

C. 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 we are looking at products between three terms,

$$\mathsf{Decomp}_2([\mathbf{u}]) \cdot \mathbf{e}_{\mathcal{K}} \cdot w, \quad (12)$$

where $\mathsf{Decomp}_2([\mathbf{u}])$ and $\mathbf{e}_{\mathcal{K}}$ (defined in Eq. (4)) 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. (5).

First, we 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. (9) 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 we have a final scaling factor $\sqrt{\lceil \lg q \rceil}$ from the L_2 (Euclidean) summation of $\lceil \lg q \rceil$ product polynomials.

Algorithm 1 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. (15).
6: **if** $\lg ||e|| < \lg q - \lg p - 1$ **then**
7: Add (p, q_j, n_j) valid parameter list $Params$.
8: Output $q, n = \min_{q,n}(Params)$

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, we know from [11] 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

$$\Pr[||(e_{p_2,i})_j|| > (T/\sqrt{2})||e_{\mathcal{K},w,i}||] < 2 \cdot \exp(-T^2/2), \quad (13)$$

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

Finally, for the last term in Eq. (8), we can apply Lemma 2 on the i-th product polynomial $e_{p_3,i} = \mathsf{Decomp}_{2,i}([\mathbf{u}]) \cdot e_{\mathcal{K},i}$ as

$$\Pr[||(e_{p_3,i})_j|| > (T/\sqrt{2})||e_{\mathcal{K},i}||] < 2 \cdot \exp(-T^2/2), \quad (14)$$

where $||e_{\mathcal{K},i}||$ is bounded by Lemma 1, and we have the same scaling factor $\lceil \lg q \rceil$ due to the inner product. Finally, to calculate Eq. (8), one can simply compute

$$k_0(||e_{p_1}|| + ||e_{p_2}||) + k_1||e_{p_3}||. \quad (15)$$

IV. Dynamic Parameter Estimation

A. The Overall Procedure

Our dynamic parameter estimation technique can be summarized by the procedures outlined in Alg. 1. 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, we set our choices on n_o and p. 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, we enter a loop where we estimate 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 [12]. Subsequently, if the calculated error ensures a correct decryption up to some probabilities derived in the corresponding bounds, we accept the parameter set. Finally, we select the smallest q and n as the parameters for concrete instantiation.

As mentioned, the size of the parameters depends critically on p and the failure probability estimation. We delay our concrete instantiations and analyses to Section V, and first discuss how to apply fast Monte-Carlo methods to further improve the correctness analysis, when the weight matrices in $\mathcal{A}$ are known.

B. Failure Probability via Sigma-Scaled Sampling

As suggested in [14], 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.

Fortunately, what we can see is 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 [13] is known to be efficient at handling high-dimensional Gaussian random variables. In short, we want 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 we abstract the homomorphic evaluation as some function f on the initial error vector $\mathbf{e}$ as $f(\mathbf{e})$, P_f can be calculated as

$$P_f = \int_{-\infty}^{+\infty} I(\mathbf{e}) f(\mathbf{e}) d\mathbf{e}, \tag{16}$$

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. We then estimate the failure probability of P_g instead of P_f. Since P_g gives is a much larger probability ($g(\mathbf{e})$ is much more likely to fail compared to $f(\mathbf{e})$), we can use brute-force Monte-Carlo 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 [13].

V. Numerical Experiments and Parameter Instantiations

In order to quantitatively assess the impact of DArL, we conduct numerical experiments on the matrix-vector product for two sets of plaintext weight filters of different dimensions: 1×2048 and 16×1024. Both dimensions are taken directly from Gazelle [12] with the decomposition parameter $b = 2^{10}$.

A. Concrete Analysis on the Theoretical Bounds

We demonstrate an example derivation of error bounds using parameters provided in Gazelle outlined in Table I. We first note that, by having an 18-bit plaintext modulus and a $b = 2^{10}$, we need to transmit two copies of the same ciphertext $([2^{10}\mathbf{u}], [\mathbf{u}])$ 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, we note that when $n_o = 1$, we do not need any rotation, and the error margin increases significantly as a result of this fact.

TABLE II
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

In the case where $n_o = 8$ (we pack two rows into one ciphertext, since $n_i = 1024$ and $n = 2048$) , we first use Eq. (11) 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. (9) 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, we get that 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 we know that the error is bounded by 1663113 in this case, the error is only around 20-bit. Combined with an 18-bit (or even 22-bit) plaintext modulus, we require at most a 42-bit q. Compared to the 60-bit q required when $n_o = 8$, by dynamically adjusting the per-layer parameters, we can reduce the ciphertext size in a 1×2048 layer by as much as 67%. Moreover, if application-specific hardware is adopted, we can also improve the computational efficiency by customizing to a 38-bit modulus, instead of using a fixed 64-bit machine word.

Last but not least, 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.

B. A Different Plaintext Space

Using the previously demonstrated error bounds, we follow the steps in Alg. 1 to generate a set of plausible parameters with respect to different plaintext modulus up to the 18-bit p suggested in [12], and the results are summarized in Table II with the security levels estimated using the framework provided by [10].

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, we do not need to decompose the weight matrix to prevent noise overflow. This immediately gives us at least 2x (compared to 18-bit p) and even 3x (22-bit p)

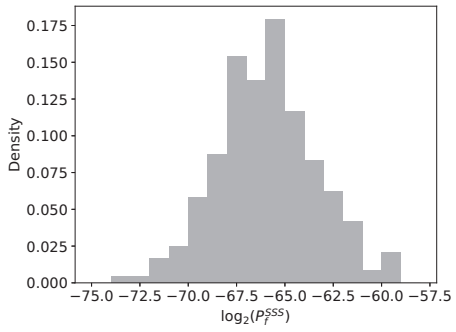

Fig. 2. The distribution of 240 runs of P_f simulation using SSS.

TABLE III
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 KB	2^{-60}
DArL Theroetical	14	55	$\approx$ 29.2 KB	2^{-40}
Gazelle	18	60	$\approx$ 65.5 KB	2^{-40}
Gazelle	22	60	$\approx$ 98.4 KB	$> 2^{-40}$

ciphertext size reduction. Moreover, due to the fact that we are essentially having less ciphertexts to rotate and add, we speed up all of the homomorphic computations 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.

C. Monte-Carlo on BNN-based Secure Inference

We applied the SSS technique on a 10×1024-dimensional packed binary weights pre-trained using the MNIST [15] dataset with a plaintext modulus of $p = 12289$. We set η_t 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. 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.

Fig. 2 shows the repeated calculation of 240 P_f estimations using SSS. As noted in [13], 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. We summarize the best-case performance difference between DArL and Gazelle in Table III. In addition, since the Monte-Carlo simulation indicates that the instantiated parameters are not entirely tight, we can further improve the efficiency of homomorphic evaluations and decryptions by approximate computing techniques, as suggested in [14].

VI. CONCLUSION

In this work, DArL is proposed to dynamically optimize the parameter instantiations used in BFV-based secure inference. In DArL, we establish theoretical approach to systematically characterize the error growth in different network settings, and developed a Monte-Carlo-based sampling method to tighten the bound on decryption failures. In the experiment, we observe a maximum of 3.3x ciphertext reduction, and 2x–3x computation speedup.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant No. 17H01713, 17J06952, 18H03214, and Grant-in-aid for JSPS Fellow (DC1).

REFERENCES

[1] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Computation Theory*, vol. 6, no. 3, p. 13, 2014.
[2] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. CRYPTO*, 2012, pp. 868–886.
[3] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.
[4] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *Proc. ASIACRYPT*, 2014, pp. 22–41.
[5] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Attribute-based encryption for circuits," *JACM*, vol. 62, no. 6, p. 45, 2015.
[6] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," *SIAM J. Computing*, vol. 45, no. 3, pp. 882–929, 2016.
[7] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. CRYPTO*, 2012, pp. 643–662.
[8] P. Mukherjee and D. Wichs, "Two round multiparty computation via multi-key FHE," in *Proc. EUROCRYPT*, 2016, pp. 735–763.
[9] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Proc. RSA Conf.*, 2011, pp. 319–339.
[10] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.
[11] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange – a new hope," in *USENIX Security Symposium*, 2016, pp. 327–343.
[12] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A low latency framework for secure neural network inference," *CoRR*, vol. abs/1801.05507, 2018.
[13] 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 TCAD*, vol. 34, no. 7, pp. 1096–1109, 2015.
[14] S. Bian, M. Hiromoto, and T. Sato, "DWE: Decrypting learning with errors with errors," in *Proc. DAC*. IEEE, 2018, pp. 1–6.
[15] "The MNIST database of handwritten digits," *http://yann.lecun.com/exdb/mnist*, 2010.
[16] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. NIPS*, 2015, pp. 3123–3131.
[17] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library – SEAL v2.1," in *Proc. Int'l Conf. Financial Cryptography and Data Security*, 2017, pp. 3–18.
[18] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Proc. Int'l Workshop on Public Key Cryptography*, 2010, pp. 420–443.
[19] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *Proc. EUROCRYPT*, 2012, pp. 465–482.
[20] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in helib," Cryptology ePrint Archive, Report 2018/244, Tech. Rep., 2018.
[21] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. Symp. Security and Privacy*, 2017, pp. 19–38.
[22] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proc. Conf. Computer and Communications Security*, 2017, pp. 619–631.