DWE: Decrypting Learning with Errors with Errors

Song Bian
Kyoto University
paper@easter.kuee.kyoto-u.ac.jp

Masayuki Hiromoto
Kyoto University
paper@easter.kuee.kyoto-u.ac.jp

Takashi Sato
Kyoto University
paper@easter.kuee.kyoto-u.ac.jp

ABSTRACT

The Learning with Errors (LWE) problem is a novel foundation of a variety of cryptographic applications, including quantumly-secure public-key encryption, digital signature, and fully homomorphic encryption. In this work, we propose an approximate decryption technique for LWE-based cryptosystems. Based on the fact that the decryption process for such systems is inherently approximate, we apply hardware-based approximate computing techniques. Rigorous experiments have shown that the proposed technique simultaneously achieved 1.3x (resp., 2.5x) speed increase, 2.06x (resp., 7.89x) area reduction, 20.5% (resp., 4x) of power reduction, and an average of 27.1% (resp., 65.6%) ciphertext size reduction for public-key encryption scheme (resp., a state-of-the-art fully homomorphic encryption scheme).

ACM Reference Format:
Song Bian, Masayuki Hiromoto, and Takashi Sato. 2018. DWE: Decrypting Learning with Errors with Errors . In *DAC '18: DAC '18: The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3195970.3196032

1 INTRODUCTION

The hardness of the Learning with Errors (LWE) problem was first proved by Regev [19], and has since become the foundation for a plethora of cryptographic primitives as well as constructions. Examples include several generations of fully homomorphic encryption (FHE) schemes [3, 5, 9, 10, 14], public key encryption (PKE) schemes [15, 19], key-exchange protocols [2, 6], digital signature schemes [9, 17], and many more.

While theoretical advances for LWE flourish, its concrete parameter instantiations, and thereby hardware instantiations, are somewhat limited. Most existing hardware instantiations [11, 16, 20] are designed for ring LWE (RLWE). RLWE operates in a subset of standard lattices called ideal lattices [18], which allows for asymptotically efficient matrix multiplication through the number theoretic transformation (NTT). Nevertheless, as recent literatures point out [2, 13], it is not known if working in the ideal lattices is identical in a standard lattices, and it *may* introduce additional security assumptions. As a result, Howe et al. recently proposed a hardware instantiation on FPGA [13] implementing the Lindner and Peikert (LP) cryptosystem [15]. This is the only hardware implementation we can find that is of a standard-lattice-based cryptosystem, and the implementation result shows that cryptosystems in standard lattices can be as efficient as those in ideal lattices [13]. Unfortunately, the above mentioned hardware instantiations, in both LWE and RLWE, are generally only on the early stage of design explorations, where the hardware serves solely as a tool to realize a set of established algorithms.

Different from existing approaches, we attempt to make a connection between LWE, a concept in cryptology, and approximate computing (AC), a field of study in hardware design. The important observation here is that LWE-based cryptosystems are inherently approximate, and that the decryption for LWE is probabilistic by design. If such is the case, as long as the parameters permit, the probabilistic nature of LWE allows approximate hardware to be deployed virtually with no additional overhead.

In this work, we propose decrypting with errors (DWE, also known as Yumeko), a methodology that converts existing "exact" (as said, the scheme is approximate by nature) LWE/RLWE schemes into approximate ones where hardware-generated noise can be infused into the schemes without breaking the correctness requirements. We focus on two well-known LWE-based cryptosystems: i) the PKE scheme by LP [15], and ii) the third-generation FHE by Gentry et al. [10], whose parameter instantiation is provided in [14]. We utilize the state-of-art DRUM multiplier [12] to approximate the multiplication involved in the decryption logic. Through theoretical verification and empirical experiments, we show that by introducing AC to LWE decryption, we can simultaneously achieve 1.3x (resp., 2.5x) speed increase, 2.06x (resp., 7.89x) area reduction, 20.5% (resp., 4x) power reduction and an average of 27.1% (resp., 65.6%) ciphertext size reduction for the public-key encryption scheme by LP [15] (resp., the state-of-the-art fully homomorphic encryption scheme by [10, 14]), at very small to no cost (except for slightly increased decryption error probability, which still satisfies the original error bound). The main results and contributions of this work are summarized as follows.

- **AC for LWE/RLWE**: Theoretically, we provide a general "transformation" that alters existing LWE/RLWE schemes such that approxiamte hardware can be utilized in the decryption function. As later shown in Section 4, we can add additional noise to LP without violating its security and correctness requirement. This translates to the result that the speed, area, power, and ciphertext-size gain from our technique comes at almost zero cost. Moreover, since all LWE/RLWE-based cryptosystems require probabilistic decryption functions where inner products between two vectors (or polynomials, as in RLWE) are taken, our technique applies to almost all such systems. To the best of our knowledge, we are the first to adopt AC in LWE-based cryptosystems.

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196032

- **Theoretical/Empirical Derivation for Provable Correctness**: One of the notable contributions of our theoretical framework is its provable correctness with tight bounds. In order to characterize the hardware-generated errors during approximate decryption, we provide both theoretical and empirical derivations that verify the correctness of the transformed cryptosystem under specific failure probability requirement.
- **Application-Specific Integrated Circuit (ASIC) Multiplier for Standard LWE**: As a separate contribution, we optimized the DRUM-based ASIC multiplier for standard LWE applications. We found very little existing works in the field of standard lattice except for Howe et al. [13], and the hardware multiplier used in [13] is a standard 18-bit multiplier on an FPGA, which is obviously suboptimal for 12-bit multiplication with small secrets. By designing an ASIC multiplier for LWE decryption, we show that a further 1.325x speed improvement, 2x area reduction and 1.84x power reduction can be achieved.

The rest of this paper will be organized as follows. First, in Section 2, we provide preliminaries on existing cryptographic constructs, and the DRUM multiplier. Second, we show how to convert such existing construct to an approximate one in Section 3, and discuss the concrete parameter instantiation in Section 4. Third, we provide the hardware architecture and results implementing the proposed algorithms in Section 5. Finally, we conclude our paper in Section 6.

2 PRELIMINARIES

2.1 Error Distributions

All LWE cryptosystems depend on some error distributions that satisfies specific security requirements. Due to the original proof of hardness by Regev, a discrete Gaussian distribution is generally chosen, denoted as χ_σ, where σ is the standard deviation. In this work, we follow the notation in [8, 13], where σ is the standard deviation, instead of the s parameter used in the original work in [15] (note $\sigma = \frac{s}{\sqrt{2\pi}}$). We also use χ_σ^n to refer to n independent samples each drawn from χ_σ. We will focus on the following two tail bounds provided in the original work.

Lemma 2.1. (Lemma 2.1 in [15]) *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 2.2. (Lemma 2.2 in [15]) *For any real $\sigma > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have*

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

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

While a perfect discrete Gaussian is important for the security of LWE instances, some special assumptions can loosen this requirement. For instance, the standard deviation of the error distribution can be extremely small as in [2] or even non-Gaussian as in [1].

2.2 The LP Cryptosystem

Here we give a review on the LP cryptosystem [15] in detail. Since the decryption function in [10] is almost identical to the one in LP, it is left out in this work.

We provide a simplified version of LP with only one bit encryption, instead of ℓ bits as in [13, 15]. All additions and multiplications operate in the residual class of q, i.e., $\mathbb{Z}_q$.

LP.Setup(λ): Given the security parameter λ, output a set of parameters (n, σ, q), where n is the lattice dimension, σ is the standard deviation for the discrete Gaussian distribution χ, and q is some modulus.

LP.KeyGen(n, σ, q): Sample a uniform matrix $A \leftarrow \mathbb{Z}_q^{n\times n}$, and two secret vectors $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \chi_\sigma^{n\times 1}$. Let $\mathbf{p} = \mathbf{r}_1 - A \cdot \mathbf{r}_2$. Output the public key $A, \mathbf{p}$, and secret key $\mathbf{r}_2$.

LP.Enc($A, \mathbf{p}, m \in \{0, 1\}$): For a 1-bit plaintext message m, draw three errors $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi_\sigma^{n\times 1}, e_3 \in \chi_\sigma$. Output two ciphertext $\mathbf{c}_1, c_2$ where

$$\mathbf{c}_1 = \mathbf{e}_1^t A + \mathbf{e}_2^t \in \mathbb{Z}_q^{1\times n} \tag{3}$$

$$c_2 = \mathbf{e}_1^t \mathbf{p} + e_3 + m \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q. \tag{4}$$

LP.Dec($\mathbf{c}_1, c_2, \mathbf{r}_2$): Compute

$$m = \lfloor (\mathbf{c}_1\mathbf{r}_2 + c_2)/\lfloor 2/q \rfloor \rceil. \tag{5}$$

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

It is easily observed that in the decryption function, the inner product $\langle \mathbf{c}_1, \mathbf{r}_2 \rangle$ is the most computationally intensive step, and is thus the target of our work.

2.2.1 Correctness of LP. Observe that $\mathbf{c}_1\mathbf{r}_2 + c_2$ in Eq. (5) translates to

$$\begin{aligned}&(\mathbf{e}_1^t A + \mathbf{e}_2^t)\mathbf{r}_2 + \mathbf{e}_1^t\mathbf{r}_1 - \mathbf{e}_1^t A\mathbf{r}_2 + e_3 + m \cdot \lfloor q/2 \rfloor \\ &= \mathbf{e}_2^t\mathbf{r}_2 + \mathbf{e}_1^t\mathbf{r}_1 + e_3 + m \cdot \lfloor q/2 \rfloor.\end{aligned} \tag{6}$$

It is noticed that so long as $|\mathbf{e}_2^t\mathbf{r}_2 + \mathbf{e}_1^t\mathbf{r}_1 + e_3|$ (denoted as $|e|$) is less than $\lfloor q/4 \rfloor$, $-\lfloor q/4 \rfloor < m \cdot \lfloor q/2 \rfloor + e < \lfloor q/4 \rfloor$ if $m = 0$, and otherwise if $m = 1$. Since e_3 is small compared to the products, and the products are obtained from independently drawn Gaussian samples, we can write $|e| = |\langle \mathbf{e}, \mathbf{r} \rangle|$ where $\mathbf{e}, \mathbf{r} \leftarrow \chi_\sigma^{2n}$.

Lemma 2.2 can be used to bound $\langle \mathbf{e}, \mathbf{r} \rangle$. Let $T = \frac{\lfloor q/4 \rfloor}{\sigma\|\mathbf{e}\|}$, we can see that

$$\Pr[|\langle \mathbf{e}, \mathbf{r} \rangle| > \lfloor q/4 \rfloor] < 2 \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{\lfloor q/4 \rfloor}{\sigma\|\mathbf{e}\|}\right)^2\right), \tag{7}$$

where $\|\mathbf{e}\|$ is bounded by Lemma 2.1. In [15], $\exp\left(-\frac{\lfloor q/4 \rfloor}{2\sigma\|\mathbf{e}\|}\right)$ is defined as the per-symbol error probability, and is set to be 0.01. This means that for every bit-decryption in the scheme, there is a 2% chance that such decryption will fail. The probability is low enough that simple error correction codes can be added to assure successful plaintext recovery, as suggested in the original work [15].

It is noted that this idea of approximate decryption is generally found in LWE-based cryptosystems, where a simple dot product between the ciphertext and the secret key is used to recover the plaintext message with some degree of failing probability. Hence, the above mentioned error bound and per-symbol failure probability become the main motivation for this work. As later discussed

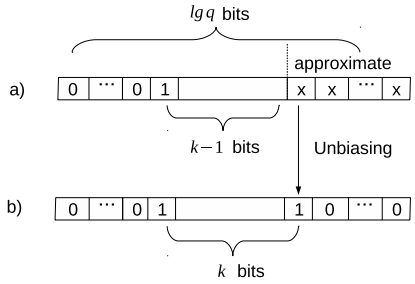

Figure 1: The approximate process for one operand used in DRUM [12] where a) is before approximation, and b) is after approximation.

Algorithm 1 Encoding transformation function f_{E}

Require: $x \in \mathbb{Z}_q$
1: $p = \mathrm{LOD}(x)$
2: $x_a = (x - 2^{p+1}) \gg (p - k)$
3: **return** $p \in \mathbb{Z}_{\lceil \lg q \rceil}, x_a \in \mathbb{Z}_{2^{k-2}}$

in Section 3, we can introduce errors for each multiplication in the dot product, and so far as the bound in Eq. (7) holds, the scheme remains correct.

2.3 DRUM

The DRUM multiplier [12] approximates its input operand using the method shown in Fig. 1. First, in a), for a $\lg q$-bit number (i.e., $\log_2 q$) the approximation algorithm finds the position of the leading one. The algorithm then copies the following $k-1$ bits along with an unbiasing bit of 1, forming a total of k bits, into the approximated result in b). The remaining bits are ignored, and approximated deterministically using 0's.

When the incoming operand follows a perfect uniform distribution in $\mathbb{Z}_q$, which is exactly the security requirement of all LWE-based cryptosystems, DRUM gives an unbiased error that roughly follows a Gaussian distribution.

In addition to its error property, it is obvious that after the approximation, only $k-2$ bits are indeterministic. Hence, for an approximate decryption engine, we only need $k-2$ bits of data, along with the position of the leading zero, to successfully compute an approximated version of the original product. We provide an encoding method in Section 4.2 such that on average, at most k bits are needed for a successful approximate decryption.

3 CRYPTOSYSTEM TRANSFORMATION

To concretely analyze the impact of approximate decryption on LWE-based cryptosystems, we actually need to slightly change the algorithmic construct of existing cryptosystems. In this work, we take LP as an example. However, as emphasized, our technique applies generally to LWE instances.

3.1 Formalizing DRUM

Before presenting the modified cryptosystem, we need to extend the DRUM algorithm into a two-party setting. Here, we define a

Algorithm 2 Decoding transformation function f_{D}

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

pair function $(f_{\mathrm{E}}, f_{\mathrm{D}})$ that transform a $\lceil \lg q \rceil$-bit number x into an approximate one y, based on the DRUM technique. The details of the functions are outlined in Alg. 1 and 2. In Alg. 1, we use the function LOD for leading one detection. The position of the leading one p, an integer in the range $[0, \lceil \lg q \rceil - 1]$, and the truncated number $x_a \in \mathbb{Z}_{2^{k-2}}$ is the output. We truncated the input x into the absolute minimum of $k-2$ bits, where all the deterministic components are discarded (including the leading one by subtracting 2^p in f_E and adding it back in f_D). Next, in Alg. 2, an approximation of x, namely $y \in \mathbb{Z}_q$, is recovered using p and x_a.

Since all operations in Alg. 1 and 2 are merely describing the DRUM multiplier, the computations translate equivalently to the hardware components in DRUM. We also define $(f_{\mathrm{E}}, f_{\mathrm{D}})$ on vectors, where $f_{\mathrm{E}}(\mathbf{x})$ and $f_{\mathrm{D}}(\mathbf{x}, \boldsymbol{\pi})$ are component-wise application of the respective functions on each element $x_i \in \mathbf{x}$, and the corresponding leading one position vector, $\boldsymbol{\pi} = [p_0, \cdots, p_{n-1}] \in \mathbb{Z}^{n \times 1}_{\lceil \lg q \rceil}$.

3.2 Proposed Modification on LP

At this stage, the modification on LP becomes simple to explain. We use identical LP.Setup and LP.KeyGen. For Enc and Dec, we add the transformation functions to the scheme.

- ModLP.Enc$(A, \mathbf{p}, m)$: We use the same errors $\mathbf{e}_1, \mathbf{e}_2, e_3$, and output the ciphertext,

$$\boldsymbol{\pi}, \mathbf{c}_f = f_{\mathrm{E}}(\mathbf{e}_1^t A + \mathbf{e}_2^t) \tag{8}$$

$$c_2 = \mathbf{e}_1^t \mathbf{p} + e_3 + m \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q. \tag{9}$$

- ModLP.Dec$(\boldsymbol{\pi}, \mathbf{c}_f, c_2, \mathbf{r}_2)$:

$$m = \lfloor (f_{\mathrm{D}}(\mathbf{c}_f, \boldsymbol{\pi}) \cdot \mathbf{r}_2 + c_2) / \lfloor 2/q \rfloor \rceil. \tag{10}$$

3.2.1 Correctness. The first observation we make is that f_{D} does not need to be applied to both operands involved in the inner product $\langle \mathbf{c}_f, \mathbf{r}_2 \rangle$. This is due to the fact that we used a small-secret LWE instance. For the instantiated parameters later discussed in Section 4, one of the operands to the DRUM multiplier, $r_i \in \mathbf{r}_2$ is always accurate, for r_i follows a Gaussian distribution with small σ ($\sigma = 3.33$). The probability that $r_i \geq 2^k$ is negligible ($2^k = 512$ in our instantiation. This means that for $r_i \geq 2^k$, r_i needs to reach $> 100\sigma$, which is practically impossible). Thus, only one operand will be approximated by the $(f_{\mathrm{E}}, f_{\mathrm{D}})$ function pair.

To further analyze the decryption error in the modified LP (ModLP) scheme, notice that for any number $x \in \mathbb{Z}_q$, the corresponding $y = f_{\mathrm{D}}(f_{\mathrm{E}}(x)) = x + e_f$, for some error e_f induced by the DRUM approximation. In other words, we can think of the DRUM approximation as introducing some additive error to the original ciphertext, which happens to allow us to ignore some of its bits. Consequently, we can see that $f_{\mathrm{D}}(\mathbf{c}_f, \boldsymbol{\pi}) = f_{\mathrm{D}}(f_{\mathrm{E}}(\mathbf{e}_1^t A + \mathbf{e}_2^t)) = (\mathbf{e}_1^t A + \mathbf{e}_2^t) + \mathbf{e}_f^t$, where each entry of $\mathbf{e}_f \in \mathbb{Z}_q^{n \times 1}$ is added to the corresponding entry in the original ciphertext $\mathbf{e}_1^t A + \mathbf{e}_2^t$. Alternatively, this can be written as $f_{\mathrm{D}}(\mathbf{c}_f, \boldsymbol{\pi}) = \mathbf{c}_1 + \mathbf{e}_f^t$. Using the same decryption equation in Eq. (6)

Table 1: Parameter Instantiation for ModLP and ModFHE

	q	n	σ	$\lg q$	k
ModLP	4096	256	3.33	12	9
ModFHE	2^{33}	1024	3.98	33	11

while replacing $\mathbf{c}_1$ with $f_{\mathrm{D}}(\mathbf{c}_f, \boldsymbol{\pi})$, we can see that the decryption result becomes

$$(\mathbf{e}_1^t A + \mathbf{e}_2^t + \mathbf{e}_f^t)\mathbf{r}_2 + \mathbf{e}_1^t \mathbf{r}_1 - \mathbf{e}_1^t A \mathbf{r}_2 + e_3 + m \cdot \lfloor q/2 \rfloor$$
$$= (\mathbf{e}_2 + \mathbf{e}_f)^t \mathbf{r}_2 + \mathbf{e}_1^t \mathbf{r}_1 + e_3 + m \cdot \lfloor q/2 \rfloor. \quad (11)$$

The important note here is that, if we can somewhat assume that the element of $\mathbf{e}_f$ comes from a discrete Gaussian distribution (which is close, but not exactly the case), we can see that $\mathbf{e}_2 + \mathbf{e}_f$ is merely the sum of two Gaussian variables with σ and σ_f, assuming the standard deviation for the presumed Gaussian distribution of $\mathbf{e}_f$ is σ_f. We can then calculate the combined standard deviation by $\sigma_s = \sqrt{\sigma^2 + \sigma_f^2}$, and use Lemma 2.1 to bound $\mathbf{e}_2 + \mathbf{e}_f$, where

$$\|\mathbf{e}_2 + \mathbf{e}_f\| \leq c \cdot \sigma_s \sqrt{n} \quad (12)$$

by overwhelming probability (concretely, $\approx 2^{-40}$ in LP).

Ideally, when $n \longrightarrow \infty$, the above bound holds by the central limit theorem, and we can define a vector $\boldsymbol{\varepsilon} = (\mathbf{e}_2 + \mathbf{e}_f)||\mathbf{e}_1 \in \mathbb{Z}_q^{2n \times 1}$, and $\mathbf{r} = \mathbf{r}_2||\mathbf{r}_1 \in \chi_\sigma^{2n \times 1}$ (Here, $||$ means vertical vector concatenation), where the per-symbol decryption failure probability is bounded using Lemma 2.2 as

$$\Pr[|\langle \boldsymbol{\varepsilon}, \mathbf{r} \rangle| > \lfloor q/4 \rfloor] < 2 \cdot \exp\left(\frac{-1}{2} \cdot \left(\frac{\lfloor q/4 \rfloor}{\sigma \|\boldsymbol{\varepsilon}\|}\right)^2\right). \quad (13)$$

In practice, since n is relatively large in almost all LWE-based cryptosystems, Eq. (13) approximates the upper bound on the decryption failure probability quite well. A more precise bound is derived from Monte-Carlo simulation in Section 4.

3.2.2 Security. The security of ModLP follows immediately from LP. We omit a formal proof due to space limitation, but as suggested, f_{E} and f_{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, ModLP has at least the same, and presumably stronger security compared to LP (due to additional error).

3.3 Transformation of FHE

Due to the limit of space, we do not describe a detailed version of ModFHE, the transformed version of the FHE scheme by [10]. However, it is not hard to see that the same procedure can be followed where f_{E} and f_{D} are applied during encryption and decryption. We will provide parameter instantiation for ModFHE in Section 4.

4 PARAMETER INSTANTIATION

In this section, we instantiate the ModLP and ModFHE schemes using concrete parameters suggested in [15] and [14], respectively, along with the corresponding DRUM instances with parameters $(\lg q, k)$. In this section, since q is a power of 2, we drop the ceiling and flooring functions around $q/4$ and $\lg q$.

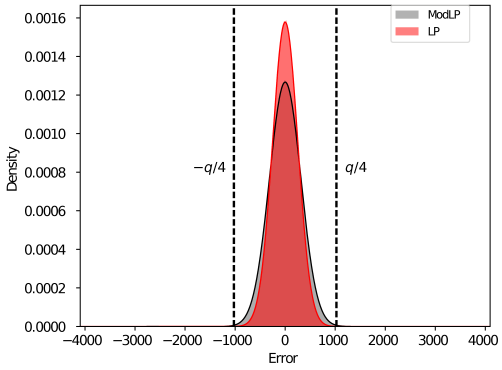

Figure 2: The decryption error distributions of LP and ModLP from 10M simulations.

The parameters used in this work is summarized in Table 1. As suggested in [4, 13], the modulus in standard lattice can be simplified to be a power of 2, which basically squashes the circuitry for modulo reductions. In ModLP, we use the same n and σ in [13, 15] that guarantee a security level roughly equivalent to AES-128. On the other hand, ModFHE has a security level of 80-bit as discussed in [14]. For the DRUM multiplier in ModLP (resp., ModFHE), we instantiate a 9-bit (resp., 11-bit) multiplier instead of a full 12-bit (resp., 32-bit) one, and study the error produced by such approximation.

4.1 Empirical Bound for Approximation Errors in ModLP

The Monte-Carlo simulation approach for the error bound is easy to perceive. We produce 10 million LP ciphertexts that are all encryptions of 0's, and studied the size of the error after decryption using the proposed ModLP and the original LP scheme. Here, decryption error for ModLP is essentially Eq. (13), and Eq. (7) for LP. The simulated error distributions for ModLP (grey area) and LP (red area) are shown in Fig. 2. A normal fitting is plotted on top of each histogram, and in both cases, the histograms follow the fitted normal almost indistinguishably well (the fitted standard deviation is denoted as σ_n here). According to the simulated results, for LP, there are 597 samples crossing $|q/4|$ in 10M simulations (0.00597%, with $\sigma_n = 252$), and 12429 samples (0.12429%, $\sigma_n = 314$) for ModLP. We can also calculate the error probability from σ_n of the fitted normal. For $\sigma_n = 252$, 1024 is 4.06σ, corresponding to a probability of $2 \cdot 0.006\%$ (2 for errors in positive and negative directions), which agrees with the Monte-Carlo simulation. Meanwhile, for a $\sigma_n = 314$, 1024 is roughly $3.261\sigma_n$, giving a probability of 0.2%.

In both cases, for 10M simulations, the decryption failure probability seems to satisfy the requirement (2% as in Section 2.2.1). Furthermore, the tail probability seems to decrease much faster than a normal fitting in both cases. Monte-Carlo simulation is stochastic by design, which means that the number cannot be taken as is. Fortunately, we can study the confidence interval for a particular set of simulations.

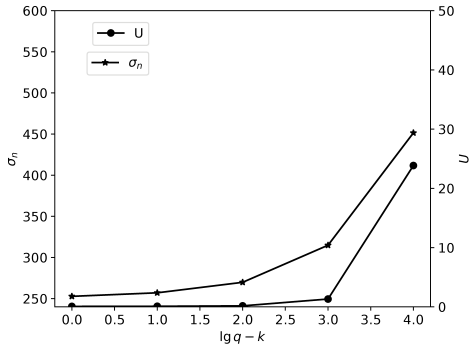

Figure 3: The upper bound U for different approximation parameter $\lg q - k$.

To concretely analyze the confidence internal, we need setup an additional random variable Y, where Y is the number of times $|e_m| > q/4$ in one set of 1000 simulations. This can also be thought of as the number of decryption failures per 1000 trials. Next, we can see that the the 10 million simulations in the previous section can be seen as 10000 independent simulations of Y. According to [7], 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 on the mean of Y can be described as (we ignore the lower bound since it is irrelevant)

$$U = \overline{y} + z_c \frac{S_y}{\sqrt{N}}. \tag{14}$$

We can obtain $\overline{y}$ and S_y from Fig. 2, and $N = 10000$ as mentioned. The z_c parameter basically describes how much confidence we hold in this interval. We set $z_c = 7$, which gives a confidence level of $1 - 2 \times 10^{-12}$. In other words, $\Pr[Y > U] < 2 \times 10^{-12}$ (roughly around 2^{-38}).

The result of U is shown in Fig. 3. The x-axis of the figure denotes $\lg q - k$, which roughly is the number of bits ignored in the DRUM multiplication (DRUM has dynamic range, so this is not always the case). When $\lg q - k = 0$, this is basically the original LP scheme where an exact multiplier is used. The important observation here is that, when the approximate error $\mathbf{e}_f$ is much smaller than the added Gaussian error $\mathbf{e}_2$, the rate for which the total error increases is slow. This fact can be explained by our theoretical insights in Section 3.2.1, where the final $\sigma_s = \sqrt{\sigma^2 + \sigma_f^2}$ is additive with a square root. Hence, the LP error ($\sigma_n = 252$ for $\lg q - k = 0$) first dominates the total error, until $\lg q - k = 3$, where σ_n has only increased to 314. Nonetheless, when $\lg q - k = 4$, the approximation error becomes dominant, and we see a significant increase in $\sigma_n = 451$ and $U = 23.85$ per 1000 decryptions.

Through the empirical study, what we found is that the LP parameter choice is rather conservative, with $U = 0.0767$ per 1000 decryptions (translates to a failure probability of 0.00767%, which is much less than 2%). Even with $\lg q - k = 4$, we are barely crossing the bar ($\approx 2.4\%$ at $z_c = 7$), and $\lg q - k = 3$ ($\approx 0.131\%$) satisfies the per symbol error probability of 2% with overwhelming confidence. Hence, we choose these parameters for the hardware implementation.

4.2 Ciphertext Size Reduction

As mentioned, since we use f_{E} to encode every element in $\mathbf{c}_1$, we only need to transform p and $c_a \in \mathbf{c}_f = f_{\mathrm{E}}(\mathbf{c}_1)$ when ciphertexts are transmitted. Note that c_2 needs to be transmitted as is, but it is only one number in $\mathbb{Z}_q$, where the majority ($n = 256$) of numbers are compressed. While we know that every c_a is a $(k-2)$-bit number, since DRUM is dynamically ranged, we also need to transmit p, which reduces the amount of ciphertext reduction. Nevertheless, since all $c \in \mathbf{c}_1$ is uniformly sampled from $\mathbb{Z}_q$, we know the exact probability mass function of each discrete p value. Hence, a simple Huffman coding can achieve optimal compression rate. 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, we can see that, on average, the total number of bits need to be transmitted for one $f_{\mathrm{E}}(c)$ is $k - 2 + \rho$, where

$$\rho = 1 \times 1/2 + 2 \times 1/4 + 3 \times 1/8 + 3 \times 1/8 = 1.75, \tag{15}$$

giving us a ciphertext reduction of roughly $1 - ((7 + 1.75)/12) = 27.1\%$.

4.3 Note on ModFHE

Using a similar technique, we can instantiate parameters for ModFHE. The only difficulty here is that, after a certain rounds of homomorphic encryption, the error distribution in the ciphertext is not as easy as that in LP. Thus, we make the absolute worst-case assumption, where all the errors is on its bound, which is $q/4$, for $q = 2^{32}$ as instantiated in [14]. Under such assumption, we need to increase the modulus q by a factor of 2, such that as long as the approximate errors are also less than $q/4$, their sum will be less than $2 \times q/4$. The ciphertext reduction in this case is $9 + \rho$, where $\rho = \sum_{i=1}^{22}(i \times (1/2^i)) + 22 \times (1/2^{22})$. Since $\rho < 2$, the ciphertext reduction in this case is $1 - ((9 + 2)/32) = 65.6\%$.

4.4 Applicability to RLWE

Up to this point, the $(f_{\mathrm{E}}, f_{\mathrm{D}})$ transformations and all of our error analysis are applicable to RLWE instances. The only problem is that with the same $(f_{\mathrm{E}}, f_{\mathrm{D}})$ that is designed from the DRUM multiplier, it is not clear how the NTT engine can be optimized to benefit from these approximations (which is an interesting open problem). Thus, while RLWE cryptosystems may not possess the hardware gain from using an approximate multiplier, it is possible to use the same analysis to reduce the ciphertext size of RLWE-based systems.

5 HARDWARE IMPLEMENTATION

5.1 Hardware Architecture

As described, the operations involved in the decryption of standard LWE with a modulus q of a power of 2 involves only an inner product between two n-dimensional vectors, and some integer additions. Hence, the main challenge is the implementation of the multiplier, with a small amount of control and adder logics. Thus, in this work, we implement and compare three types of hardware multipliers: i) a plain 12-bit multiplier as used in existing works [13], ii) an approximated 12-bit multiplier to examine the impact of AC, and iii) an LWE-specific multiplier considering modulo reduction and small secret.

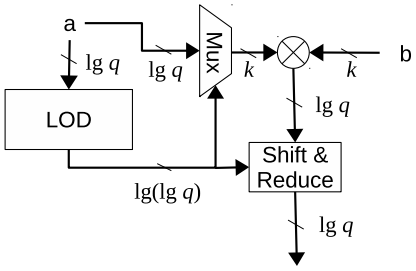

Figure 4: The proposed architecture for approximate multiplier used in small-secret LWE decryption.

Table 2: 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 32-bit	17.0	25869	160.8
DRUM11_33	6.7	3278	44.6
This work	5.5	1572	22.8

Fig. 4 is the approximate multiplier adopted for small-secret LWE. Compared to DRUM, half of the LOD detection logics are reduced, and the k-bit multiplier can be further optimized to produce only $\lg q$-bit result, instead of $2k$. Since q is a power of 2, the modulo reduction in both the multiplier and the barrel shifter is basically bit selection.

5.2 Experiment Setup and Results

The architecture shown in Fig. 4 is synthesized on a commercial 65 nm low-power process node using logic-synthesis tool [21], and the power is analyzed by [22].

The implementation result for ModLP and ModFHE is summarized in Table 2. Overall, bringing approximate computing into LWE decryption greatly reduces the amount of circuitry required for decryption. As summarized in Section 1, we see speed increase, area reduction and power reduction can be achieved all at once across all implementations. The only cost here is the decryption error probability which, as investigated in Section 4, is still under the required bound. In addition, by specializing the multiplier for LWE instances in the standard lattice, we can achieve a further 1.325x speed increase, 2x area reduction and 1.84x power reduction in the case of ModLP. The technique shines when the original scheme contains large errors, as in the case of ModFHE. We were able to use a 11-bit multiplier to calculate 33-bit multiplications, with the generated error bounded by $2^{33}/4$ by overwhelming probability.

6 CONCLUSION

In this paper, we proposed DWE, a methodology to decrypt LWE-based cryptosystems with approximate hardware. We developed a systematical approach to study the decryption failure probability when approximate hardware are used, and verified the theoretical result using empirical methods. Through the hardware implementation, we demonstrated significant speed increase, area reduction, power reduction and ciphertext reduction at almost no cost in the PKE case by LP, and a slight increase in q in the FHE case.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant No. 17H01713, 17J06952, and Grant-in-aid for JSPS Fellow (DC1). The authors also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

REFERENCES

[1] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum Key Exchange-A New Hope.. In *USENIX Security Symposium*. 327–343.
[2] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. 2016. 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, 1006–1018.
[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 13.
[4] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. 2013. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 575–584.
[5] Zvika Brakerski and Vinod Vaikuntanathan. 2014. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Computing* 43, 2 (2014), 831–871.
[6] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yong Soo Song. 2016. Lizard: Cut off the Tail!//Practical Post-Quantum Public-Key Encryption from LWE and LWR. *IACR Cryptology ePrint Archive* 2016 (2016), 1126.
[7] Morris R Driels and Young S Shin. 2004. Determining the number of iterations for Monte Carlo simulations of weapon effectiveness. (2004).
[8] Nagarjun C Dwarakanath and Steven D Galbraith. 2014. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing* 25, 3 (2014), 159–180.
[9] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 197–206.
[10] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*. Springer, 75–92.
[11] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. 2012. On the design of hardware building blocks for modern lattice-based encryption schemes. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 512–529.
[12] Soheil Hashemi, R Bahar, and Sherief Reda. 2015. Drum: A dynamic range unbiased multiplier for approximate applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 418–425.
[13] James Howe, C Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and Kevin Beeden. 2016. Lattice-based encryption over standard lattices in hardware. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 162.
[14] A. Khedr, G. Gulak, and V. Vaikuntanathan. 2016. SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers. 65, 9 (2016), 2848–2858. DOI : http://dx.doi.org/10.1109/TC.2015.2500576
[15] Richard Lindner and Chris Peikert. 2011. Better key sizes (and attacks) for LWE-based encryption. In *Cryptographers' Track at the RSA Conference*. Springer, 319–339.
[16] Zhe Liu, Hwajeong Seo, Sujoy Sinha Roy, Johann Großschädl, Howon Kim, and Ingrid Verbauwhede. 2015. Efficient Ring-LWE encryption on 8-bit AVR processors. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 663–682.
[17] Vadim Lyubashevsky. 2012. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 738–755.
[18] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1–23.
[19] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34.
[20] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. 2014. Compact ring-LWE cryptoprocessor. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 371–391.
[21] Synopsys, Inc. *Design Compiler I-2013.06*. Synopsys, Inc.
[22] Synopsys, Inc. *PrimeTime PX H-2013.06*. Synopsys, Inc.