Filianore: Better Multiplier Architectures for LWE-based Post-Quantum Key Exchange

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 (ring) learning with errors (RLWE/LWE) problem is one of the most promising candidates for constructing quantum-secure key exchange protocols. In this work, we design and implement specialized hardware multiplier units for both LWE and RLWE key exchange schemes to maximize their computational efficiency. By exploiting the algebraic structure with aggressive parameter sets, we show that the design and implementation of LWE key exchange on hardware is considerably easier and more flexible than RLWE. Using the proposed architectures, we show that client-side energy-efficiency of LWE-based key exchange can be on the same order, or even (slightly) better than RLWE-based schemes, making LWE an attractive option for designing post-quantum cryptographic suite.

ACM Reference Format:
Song Bian, Masayuki Hiromoto, and Takashi Sato. 2019. Filianore: Better Multiplier Architectures for LWE-based Post-Quantum Key Exchange . In *The 56th Annual Design Automation Conference 2019 (DAC '19), June 2–6, 2019, Las Vegas, NV, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3316781.3317850

1 INTRODUCTION

As the National Institute of Standards and Technology (NIST) initiated the discussions on the standardization of quantum-resistant public-key cipher suite [11], serious research efforts were devoted in evaluating and improving the performance of cryptographic constructions based on the (ring) learning with errors (RLWE/LWE) problem [1–3, 13]. Along with the well-established security reductions [10, 15], 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 [1, 2], making (R)LWE-based algorithms an attractive candidate for the age of post-quantum security.

The ring version of LWE, RLWE, is generally considered much more efficient than generic LWE (also known as standard LWE). RLWE reduces the n-by-n public-key matrix of generic LWE down to an n-by-1 vector [10], and also allows for efficient per-coordinate plaintext packing that leads to efficient key exchange with only one set of polynomial coefficients [1]. The cost that comes with this performance improvement, however, is the security concern, as the special lattices of RLWE do present to be easier than generic lattices.

Despite the security concern, RLWE dominates the literature for its efficiency, especially in the field of design explorations of hardware primitives [6, 9, 14, 16]. Due to the emerging nature of the field [12], most existing approaches either optimize the public-key encryption scheme based on RLWE on embedded processors [9], or prototype their hardware designs using field programmable gate array (FPGA) devices [14, 16]. In fact, the few hardware implementations for LWE-based cryptosystem available, namely [7, 8], also prototyped their designed hardware on Xilinx Artix-7 and Spartan-6 FPGAs. Since FPGA devices are integrated with high-speed digital signal processing (DSP) hardware multiplier, almost all existing works employ at least one of the DSP units, and leave the architectural design for the computational unit completely untouched.

In this work, we propose Filianore, a hardware accelerator for Frodo [2], the-state-of-the-art LWE key exchange scheme, and R-Filianore for NewHope [1], the latest RLWE-based key exchange scheme. Instead of simply adopting DSP units, we exploit the algebraic difference between the two hardness assumptions (LWE and RLWE), and instantiate distinct hardware architectures for the respective computations. Through the experiment in Section 5, we find that compared to existing FPGA-based implementations, application-specific integrated circuit (ASIC) multplier with optimized parameter selection gives us better throughputs by nearly 40x compared to the most recent art [8]. For Frodo-I, the slightly modified version of the recommended parameters suggested in [2], we were able to 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 furhter 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, we significantly reduced the energy gap between generic LWE and RLWE compared to previous studies. We summarize our contributions as follows.

- **Better Hardware Primitives for (R)LWE Key Exchange**: As [12] points out, previous design explorations for (R)LWE have been exclusively on embedded processors or FPGA. In this work, we provide careful design explorations 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

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317850

Alice		Bob
$seed_A \leftarrow U(\{0,1\}^s)$		
$A \leftarrow \mathsf{Gen}(seed_A)$		
$S, E \leftarrow \chi(\mathbb{Z}_q^{n\times\overline{n}})$		
$B \leftarrow AS + E$	$\xrightarrow{seed_A, B}$	$A \leftarrow \mathsf{Gen}(seed_A)$
		$S', E' \leftarrow \chi(\mathbb{Z}_q^{\overline{m}\times n})$
		$B' \leftarrow S'A + E'$
		$E'' \leftarrow \chi(\mathbb{Z}_q^{\overline{m}\times\overline{n}})$
		$V \leftarrow S'B + E''$
	$\xleftarrow{B', C}$	$C \leftarrow \langle V\rangle_{2^r}$
$K \leftarrow \mathsf{rec}(B'S, C)$		$K \leftarrow \lfloor V\rceil_{2^r}$

Figure 1: Key exchange from generic LWE proposed by [2].

Alice		Bob
$seed_a \leftarrow U(\{0,1\}^s)$		
$\hat{\mathbf{a}} \leftarrow \mathsf{Gen}(seed_a)$		
$\hat{\mathbf{s}}, \hat{\mathbf{e}} \leftarrow \mathsf{NTT}(\chi(\mathbb{Z}_q^n))$		
$\hat{\mathbf{b}} \leftarrow \hat{\mathbf{a}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$	$\xrightarrow{seed_a, \hat{\mathbf{b}}}$	$\hat{\mathbf{a}} \leftarrow \mathsf{Gen}(seed_a)$
		$\mathbf{s}', \mathbf{e}' \leftarrow \mathsf{NTT}(\chi(\mathbb{Z}_q^n))$
		$\hat{\mathbf{b}} \leftarrow \hat{\mathbf{a}} \circ \hat{\mathbf{s}}' + \hat{\mathbf{e}}'$
		$\mathbf{e}'' \leftarrow \chi(\mathbb{Z}_q^n)$
		$\mathbf{v} \leftarrow \mathsf{INTT}(\hat{\mathbf{b}} \circ \hat{\mathbf{s}}') + \mathbf{e}''$
$\mathbf{v} \leftarrow \mathsf{INTT}(\hat{\mathbf{b}}' \circ \hat{\mathbf{s}})$	$\xleftarrow{\hat{\mathbf{b}}', \mathbf{c}}$	$\mathbf{c} \leftarrow \mathsf{HelpRec}(\mathbf{v})$
$K \leftarrow \mathsf{rec}(\mathbf{v}, \mathbf{c})$		$K \leftarrow \mathsf{rec}(\mathbf{v}, \mathbf{c})$

Figure 2: Key exchange from RLWE proposed by [1].

all cryptographic applications based on the LWE problem. In this work, we try 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, we show that generic LWE can be of practical use, and is more secure with the aforementioned worst-case hardness reductions.

- **Instantiation of Asymmetric LWE Key Exchange**: As [2] 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 [2] and the recent implementation [8]. As an effort to lift the client-side computational burden, we propose and compare different sets of asymmetric parameters, and show that under such construction, generic LWE can be as energy-efficient as RLWE.

The rest of this paper will be organized as follows. First, in Section 2, we provide preliminaries on the (R)LWE key exchange schemes. Second, we discuss and present our parameter instantiation for the respective schemes in Section 3. Third, the hardware architecture is described in Section 4 and the synthesized results are compared in Section 5. Finally, we conclude our paper in Section 6.

2 PRELIMINARIES

2.1 Notations

Throughout the paper, we use the standard notation $\mathbf{a} \in \mathbb{Z}_q^n$ to refer to a vector $\mathbf{a}$ who has 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 number-theoretic transform (NTT) are noted with a hat, e.g., $\hat{\mathbf{a}} = \mathsf{NTT}(\mathbf{a})$, and the coordinate-wise multiplications between 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 $\log_2 x$.

2.2 LWE-based Key Exchange

A generic LWE instance is parameterized by (n, q, χ), where n is the lattice dimension, q a modulus, and χ a distribution parameterized by σ. For generic LWE, we fixed q to be a power of two, and σ a set of custom-defined Gaussian-like probability density functions specified in Table 1 of [2].

The complete key exchange protocol proposed by [2] for generic LWE is outlined in Fig. 1. Here, we give a brief summary 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 [2], 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 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 [2]). 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 we safely ignore the performance impact of this step on the whole procedure.

2.3 RLWE-based Key Exchange

RLWE has a similar construct to LWE, where the parameters are (n, q, χ). For the sake of simplicity of presentation, we use the same notations 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 purpose as their previously defined counterparts.

The RLWE-based key exchange protocol is outlined in Fig. 2. We slightly modified the communication protocol for the ease of

Table 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 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		

Table 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

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. We delay the concrete parameter instantiation, and thus detailed performance comparison into the next Section. 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.

3 PROPOSED PARAMETER INSTANTIATION

In this section, we first instantiate Frodo [2] under different parameter sets, and then provide the parameter instantiation for NewHope [1] with discussions on its efficiency.

3.1 Parameter Instantiation for LWE

For the LWE-based key exchange protocol in Fig. 1, we provide three sets of parameter instantiations, 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.

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 $O(q^{1/n})$), 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, we base our concrete security analysis on the original work [2], where 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 [4]. We then determine the least viable b which allows BKZ to yield a successful attack (details can be found on page 11–12 in [2]). For correctness, as suggested in Claim 3.2 in [2], as long as the (absolute) distance $|e|$ between each entry of $B'S$ and C is less than $q/2^{r+2}$, we have $\text{rec}(B'S, C) = \lfloor V \rceil_{2^r}$, and the two parties derive the same key K. We can use the continuous Gaussian to bound the tail probability of the discrete Gaussian, i.e.,

$$\Pr[|e| > q/(2^{r+2})] = 2 \cdot \Phi\left(\frac{q/(2^{r+2})}{\sigma_c}\right), \tag{1}$$

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, we can calculated σ_c as $\sigma_c^2 = 2n\sigma^4 + \sigma^2$.

The overall parameter instantiaions, error distributions, and security and failure probability estimations are summarized in Table 1, Table 2, and Table 3, respectively. Here, P.Q. security denotes post-quantum security. Frodo-Rec is the recommended parameter set in [2] with CPU implementation in mind, and Frodo-I is the unbalanced version of Frodo-Rec. In Frodo-I, we set $\overline{n} = 1$, and we are essentially computing $A\mathbf{s}$. This 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.

We present Frodo-II 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, we derive fewer bits of keys 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. We 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, we show 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 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 [2]) used in Frodo-I, and 38.3% for χ_2 (D_2 in [2]) in Frodo-II. By avoiding the actual need for multiplying and adding these zeroes, we can further reduce the number of multiplications by exactly 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 tempered. In fact, some post-quantum LWE constructs set the number of zeroes in the secret vector to be a fixed and public value [5].

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, we can reverse the role to ease the server-side computation, with potential benefit such as improved server response time.

3.2 Parameter Set for RLWE

The parameters provided in Table 1 for RLWE are directly taken from NewHope. We omit the formal mathematical background and only point out some of the problems related to this parameter setting.

The first problem we have 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 1 provides 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 $\mathbf{s}, \mathbf{e}$, 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.

4 PROPOSED HARDWARE ARCHITECTURE

4.1 Filianore

The proposed architecture for the computing unit used in generic LWE-based key exchange is shown in Fig. 3. Our observation is that all multiplications in generic LWE are between some arbitrary $\lg q$-bit number, and a "small" number drawn from a Gaussian-like distribution. The range of this number is precisely defined in Table 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. Furthermore, all multiplications by a number less than 7 can be done through exactly one addition. Thus, the "multiplier" unit in generic LWE can be realized by a single adder with two multiplexers. Although we can further reduce the hardware complexity by assuming 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, we did not perform this optimization on the proposed design.

The complete system for computing AS and $B'S$ in Fig. 4 illustrates the simplicity of Filianore. An extra adder is used to perform the accumulation of matrix products, and we can also 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). We will delay the performance discussion into the next section where we synthesize the design, 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. 5, 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 we set the degree of parallelization to be $k = 16$, 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 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 case, we can still use the second approach, where we just stamp the architecture in Fig. 4 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 [2] explains, for a typical HTTPS connection based on TLS, a single unit of Filianore fulfills the computational demand for Alice.

Table 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)	$\hat{a}$	$\hat{b}$	$\hat{e}$

Algorithm 1 Cooley-Tukey butterfly from [16].

```
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 [1].

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

Algorithm 3 Short Barrett reduction from [1].

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

4.2 R-Filianore

Since all previous RLWE implementations we found avoid designing the core computational unit, to enable fair comparison, we also propose R-Filianore, an accelerator for RLWE-based key exchange. Our proposed architecture shares similarity with [16], but without a convenient DSP unit, we need to rethink how modular multiplication should be realized.

The designed accelerator is sketched in Fig. 6. We adopted the optimization techniques suggested in [1], 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 [1], where additional reductions mean additional CPU cycles. This is also the reason why, instead of Gentleman-Sande (GS) butterfly, we utilized CT butterfly from [16]. The CPU-based implementation can have 32-bit additions and multiplications for free, but this is not the case for specialized hardware. We point out that in GS butterfly used by [1], the maximum input to the multiplier can be

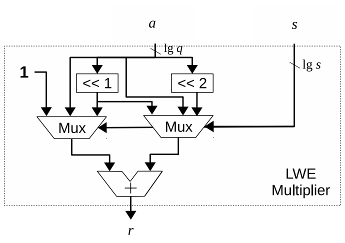

Figure 3: Proposed hardware multiplier for generic LWE.

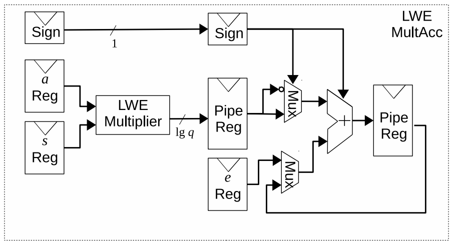

Figure 4: Proposed multiply-accumulate unit for generic LWE.

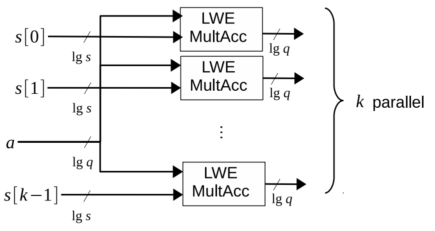

Figure 5: Parallelized multiply-accumulate units for generic LWE.

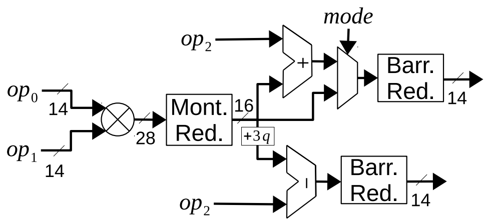

Figure 6: Proposed hardware accelerator for RLWE.

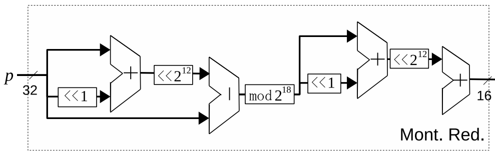

Figure 7: Hardware Montgomery reduction unit for a fixed $q = 12289$ and $R = 2^{18}$.

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 that internally performs modular reduction [16], or works on general-purpose processors where i) and iii) are trivial.

Table 4 indicates the assignments for inputs op_0, op_1, op_2 under three different operational mode. For mode i), we multiply the 14-bit input operand s 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. 7. In mode ii), computations for Alg. 1 can be carried out in a single pipelined cycle. These two steps essentially perform one NTT operation required in Fig. 2. Lastly, we compute coordinate-wise multiplications and additions in mode iii). By these three modes, all heavy computations in Fig. 2 can be performed on R-Filianore.

Finally, we make a short note on the efficiency modular reduction units. As Fig. 7 shows, we optimized these reductions to work on a specific q, namely $q = 3 \cdot 2^{12} + 1 = 12289$. Thus, multiplications are computed as consecutive additions, e.g., $p \cdot 12289 = (p + 2p) \cdot 2^{12} + p$. Due to space constraint, we omit the architecture for short Barrett reduction, which shares a similar structure to Fig. 7. Since these units are optimized for a fixed modulus, we gain efficiency at the cost of flexibility.

Table 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

Table 6: Latency, Energy Consumption and Memory Bandwidth Comparison for Alice

	F-I	F-II	F-Rec	R-F	[8]
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	-

5 HARDWARE IMPLEMENTATION AND COMPARISON

The architectures shown in Fig. 5 and Fig. 6 are synthesized on a commercial library of a 65 nm low-power process node using a logic-synthesis tool [17], and the power is analyzed by [18]. The synthesized results are summarized in Table 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 5 refers to circuit delay, which is different from the key-exchange latency later discussed.

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

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

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 $S = \mathbf{s} \in \mathbb{Z}_q^n$. We have a constant g, which is the average number of zeroes in Alice's secret vector. We take $g_{\max} = 0.705$ for Frodo-I and $g_{\max} = 0.617$ for Frodo-II.

Computational latency for NewHope on R-Filianore is

$$L = \left(2n + 2 \cdot \frac{n}{2} \lg n + 2n + \frac{n}{2} \lg n\right) \cdot 5.5. \quad (3)$$

Here, the first $2n$ represents costs for transforming errors $\mathbf{s}, \mathbf{e}$ into Montgomery form. We then need to perform two NTT operations 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 second $2n$ 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 6, our results are compared with the most relevant work in [8]. We calculate the energy consumption using the equation $E = L \cdot P$, where L is the latency and P is the power in Table 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, we see 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, we 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 ($O(n)$ for write). On the other hand, since the NTT writes two outputs per butterfly, more writes are generated ($O(n \log n)$).

The proposed multiplier achieves much better throughput than the existing FPGA-based design in [8], due to custom architecture and aggressive parameter instantiation. We also note that focusing on the multiplier architecture, Filianore is much more energy-efficient than existing studies on generic LWE. For example, the design in [7] is 28x slower in latency while having a similar power consumption compared to [14], 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, we note that as explained in [2], the latency for key exchange has diminishing impact on the connection speed for a real-world work load. Since our implementation is almost as fast as the CPU implementation in [2], 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 we require (15 to 20 bits).

6 CONCLUSION

In this paper, we proposed (R-)Filianore, a set of hardware designs for the (R)LWE-based key exchange algorithms. By carefully exploiting the algebraic structure of generic LWE, we show that under the asymmetric setting, client-side computations for LWE be as energy-efficiency as RLWE, with an extra area reduction of 8-12x. We conclude that given its flexible nature and solid security reduction, generic LWE remains competitive against RLWE for the post-quantum age.

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] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. 2015. 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, 553–570.

[4] Yuanmi Chen and Phong Q Nguyen. 2011. BKZ 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 1–20.

[5] 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.

[6] 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.

[7] 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.

[8] James Howe, Tobias Oder, Markus Krausz, and Tim Güneysu. 2018. Standard Lattice-Based Key Encapsulation on Embedded Devices. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 372–393.

[9] 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.

[10] 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.

[11] NIST. 2018. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. (2018). Accessed: 2018-04-20.

[12] Tobias Oder, Tim Güneysu, Felipe Valencia, Ayesha Khalid, Maire O'Neill, and Francesco Regazzoni. 2016. Lattice-based cryptography: From reconfigurable hardware to ASIC. In *Integrated Circuits (ISIC), 2016 International Symposium on*. IEEE, 1–4.

[13] Chris Peikert. 2014. Lattice cryptography for the internet. In *International Workshop on Post-Quantum Cryptography*. Springer, 197–219.

[14] Thomas Pöppelmann and Tim Güneysu. 2013. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *International Conference on Selected Areas in Cryptography*. Springer, 68–85.

[15] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34.

[16] 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.

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

[18] Synopsys, Inc. 2013. *PrimeTime PX H-2013.06*. Synopsys, Inc.