Fan, S., Liu, W., Howe, J., Khalid, A., & O'Neill, M. (2019). Lightweight Hardware Implementation of R-LWE Lattice-Based Cryptography. In *2018 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2018* (pp. 403-406). [8605630] Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/APCCAS.2018.8605630

Lightweight Hardware Implementation of R-LWE Lattice-Based Cryptography

Sailong Fan*, Weiqiang Liu*, James Howe†, Ayesha Khalid‡ and Maire O'Neill‡
*College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China
†Department of Computer Science, University of Bristol, Bristol, UK
‡Centre for Secure Information Technologies (CSIT), ECIT, Queen's University Belfast, Belfast, UK
E-mail:{fansl4g, liuweiqiang}@nuaa.edu.cn, james.howe@bristol.ac.uk, {a.khalid, m.oneill}@qub.ac.uk

***Abstract*—Lattice based cryptography (LBC) is one of the most promising post-quantum cryptographic candidates. Ring-learning with errors (R-LWE) is an encryption scheme of LBC. In this paper, a lightweight hardware implementation is presented including key generation, encryption, and decryption. The R-LWE encryption scheme consists of a Gaussian sampler and polynomial multiplication. This paper uses cumulative distribution table (CDT) as the Gaussian sampler and schoolbook approach for the polynomial multiplication. The purpose of this architecture is to achieve small area consumption with high frequency. The hardware implementation results on the Xilinx Kintex-7 FPGA show that the design consumes 808 slices and the frequency can be up to 288.35MHz.**

***Index Terms*—lattice-based cryptography; ring-learning with errors; cumulative distribution table; polynomial multiplication; FPGA**

I. Introduction

In the increasingly prominent context of information security, higher security encryption algorithms are required to protect personal information and privacy. With the breakthrough of quantum computer, especially the invention of quantum algorithm represented by Shor [1] in 1994, existing public-key encryption algorithms based on large number decomposition and discrete logarithm such as RSA [2] and elliptic curve cryptography (ECC) [3] will be no longer secure. In order to resist the attacks of quantum computer, post-quantum cryptography (PQC) has been created, such as the commonly cited lattice based [4], code-based [5], hash-based [6], multivariate quadratic [7] cryptography, and the latest supersingular isogeny Diffie-Hellman (SIDH) key exchange protocol scheme [8]. Amongst the potential PQC algorithms, lattice-based cryptography (LBC) is one of the most promising candidates. LBC algorithms are based a hard problem of the short (or closest) vector problem (SVP or CVP) in a lattice. This hard problem in LBC is believed to be difficult for classical and quantum computers. Therefore, even the practical quantum computer comes true, it can still be robust enough to withstand an attack.

Ring-learning with errors (R-LWE) algorithm was proposed by Lyubashevsky et al. [4]. It operates on the ring $\mathbb{Z}_q[x]/(f)$, where f is an irreducible polynomial and p is a prime. In most cases, $f = x^n + 1$ where n is a power of 2. R-LWE algorithm uses polynomial multiplication, which leads to large key sizes and signnificant hardware resources. However, compared to RSA and ECC, it is compact.

In this paper, we aim to design a lightweight hardware implementation of R-LWE, which uses a small amount of resources but also guarantees security. For the Gaussian sampler, the cumulative distribution table (CDT) method is used due to its simplicity and lightweight feature. This method does not use any RAM and has a fast speed. Before the Gaussian sampler, we use M-sequence as the random number generator. To implement the costly polynomial multiplication, we use schoolbook multiplication instead of the number theoretic transform (NTT). Rather than using the block RAM, the proposed design is implemented with distributed RAM (i.e., LUTM), which can achieve high frequency and throughput.

The lightweight design includes implementations of R-LWE for key generation, encryption, and decryption. The parameter set used is $(n, q, \sigma) = (256, 7681, 4.51)$ which is consistent with [9] and attains the medium security levels.

The paper is organized as follows. Section II presents the background of R-LWE public-key scheme. Section III presents the proposed hardware architecture for both of the encryption and decryption of the R-LWE public key scheme. Section IV gives the hardware results and compares with previous designs. Section V concludes the paper.

II. Background

Lattice-based cryptography has already been implemented in a number of different situations, with schemes based on the hardness of the learning with errors problem (LWE) and the Ring-LWE (R-LWE) problem. Compared with LWE, R-LWE algorithm improves the public and private key size and improves efficiency. However, R-LWE algorithms also have some disadvantages, such as complex operation of polynomial multiplication. Pöppelmann and Güneysu proposed an efficient polynomial multiplication based on NTT [10], and later they provided the whole design of the R-LWE scheme with some practical optimizations [11]. In addition, they investigate a low-cost scenario with very limited resources [12], using different values of parameter sets and using the Bernoulli sampler.

The R-LWE algorithm mainly consists of two operations, that is, the Gaussian sampler and multiplication. Discrete Gaussian distribution (devoted as D_σ) is considered over the

integers with standard deviation σ and mean $\mu = 0$. In this work, the Gaussian standard deviation $\sigma = 4.51$ is used. The Gaussian probability distribution function is as follows.

$$f(x) = \frac{1}{(\sqrt{2\pi}\sigma)} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = 0.88457 exp(-0.2458x^2) \quad (1)$$

And the cumulative probability distribution equation is $F(x) = \int_{\infty}^{x} f(t)dt$. According to the probability theory, we can have the following results.
For $X \sim N(\mu, \sigma^2)$,

$$F(x) = P\{X \leq x\} = P\{(X-\mu)/\sigma \ \leq (x-\mu)/\sigma\} = \Phi((x-\mu)/\sigma)$$

then,

$$F(1) = P\{X \leq 1\} = \Phi(1/\sigma) = 0.5871$$

$$F(2) = P\{X \leq 2\} = \Phi(2/\sigma) = 0.6700$$

$$F(3) = P\{X \leq 3\} = \Phi(3/\sigma) = 0.7486$$

Next, some uniform distribution (devoted as U) random numbers are needed to compose the discrete Gaussian distribution number. The maximum number of uniform random numbers determine the precision of the Gaussian distribution.

Another complex operation is polynomial multiplication. It's the most time-consuming operation, resulting in slow computation. The schoolbook algorithm can be written as follows.

$$\boldsymbol{ab} = \left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j x^{i+j}\right] mod(x^n + 1) \quad (2)$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the two inputs to the multiplier. This calculation has a time complexity $O(n^2)$.

The public key encryption scheme of R-LWE including $Key\_gen()$, $Enc(\boldsymbol{a}, \boldsymbol{p}, m)$ and $Dec(\boldsymbol{c_1}, \boldsymbol{c_2})$ is defined as shown in Table I.

TABLE I
THE PUBLIC KEY ENCRYPTION SCHEME OF R-LWE

$Key\_gen()$	Generate $\boldsymbol{r_1}$, $\boldsymbol{r_2} \in D_\sigma$ and $\boldsymbol{a} \in U$. Let $\boldsymbol{p} = \boldsymbol{r_1} - \boldsymbol{a r_2}$. Then the public key is $\boldsymbol{p}$ and the secret key is $\boldsymbol{r_2}$.
$Enc(\boldsymbol{a}, \boldsymbol{p}, m)$	Generate $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3} \in D_\sigma$. Let $\bar{\boldsymbol{m}} = ENCODE(m)$. Then the cipher text is $\boldsymbol{c_1} = \boldsymbol{ae_1} + \boldsymbol{e_2}$, $\boldsymbol{c_2} = \boldsymbol{pe_1} + \boldsymbol{e_3} + \bar{\boldsymbol{m}}$
$Dec(\boldsymbol{c_1}, \boldsymbol{c_2})$	The plaintext is $DECODE(\boldsymbol{c} = \boldsymbol{c_1 r_2} + \boldsymbol{c_2})$

III. THE PROPOSED LIGHTWEIGHT HARDWARE ARCHITECTURE OF THE R-LWE

In this section, the proposed lightweight hardware architecture of R-LWE lattice-based encryption is presented. We use a fast and low-cost Gaussian sampler based on CDT method and a pipelined schoolbook polynomial multiplier.

A. Discrete Gaussian Sampling

For the discrete Gaussian sampler, the standard deviation $\sigma = 4.51$ ($s = \sqrt{2\pi}\sigma = 11.31$) and the mean $\mu = 0$ is used. In this design, we use a precomputed table (3140 bits) to sample the data. 40 random bits are generated to compare with the table and calculate the rank of the data in which the maximum number is 31.

The first step of the Gaussian sampler is generating the random bits. Considering the good statistical properties of M-sequence, we apply it to generate the required random numbers. Then we need to find the order of the number, comparing with those number in the precomputed data. In this process, we can use method of binary search to increase the speed of calculation. However, the number generated through this method is too small and will be insecure. A better way to solve this problem is to distribute the number to both sides of x-axis and this covers both positive and negative sides of the distribution as it is symmetrical. The details are described in Algorithm 1.

Algorithm 1 Discrete Gaussian Sampling Based on CDT

Input: seed : 40-bit unsigned, q = 7681 modulus, PGD : precomputed Gaussian data
Output: gdata : 13-bit unsigned

```
1: rnd = m_squence(seed);
2: min = 0; cur = 16; jmp = 16;
3: do
4:     cur = min + jmp;
5:     jmp = jmp ≫ 1;
6:     if(rnd ≥ PGD(cur))
7:         min = cur;
8: while(jmp == 0)
9: if( rnd[0] )
10:    gdata = cur;
11: else
12:    gdata = q - cur;
13: return gdata;
```

Fig. 1 shows the hardware structure of discrete Gaussian sampler. The primitive polynomial of M-sequence is $f(x) = x^{40} + x^5 + x^4 + x^3 + 1$. The hardware circuit brings one Gaussian data per clock. Hence a full Gaussian polynomial can be produced after 256 clocks.

B. Polynomial Multiplication

We use schoolbook multiplication for the parameter set $n = 256$ and $q = 7681$ [9]. It can be implemented with just a multiplier of 13×13 bits. The cost is much lower than NTT method. Algorithm 2 shows the process of a 13×13 bit multiplier, where addition and modular reduction are also required. It is not easy to perform the modulo reduction. If you use the divider to implement it, it will waste a lot of resources. Barrett's reduction algorithm [13] can solve this problem, but this needs a small modification. In [14], an algorithm was proposed to fit the AVR micro processor. The variant of algorithm is shown in Algorithm 2.

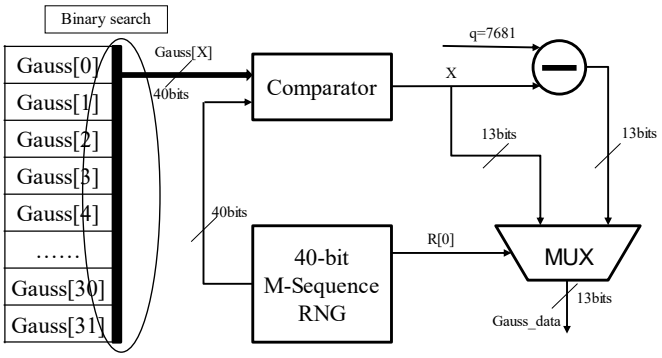

Fig. 1. Hardware structure of the discrete Gaussian sampling.

Algorithm 2 Barrett's Reduction Variant Algorithm

Input: x : 26-bit unsigned, q = 7681 modulus
Output: y : 13-bit unsigned
1: t = x[25:13] + x[25:17] + x[25:21];
2: tq = (t≪13) + t - (t≪9);
3: y = x - tq;
4: if(y ≥ q)
5: y = y - q;
6: if(y ≥ q)
7: y = y - q;
8: if(y ≥ q
9: y = y - q;
10: **return** y;

In this algorithm, x is the product of two numbers and y is number x modulo q. In Step 2, we observe that $7681 = 0x1e01 = 0x2000 - 0x0100 + 0x0001$. Therefore, we can simplify the multiplication of $t \times q$ to addition and shift operations. Then, if the obtained result is not totally reduced, additional subtractions are needed (steps 4-9 in Algorithm 2), where the maximum number of subtractions is 3.

Fig. 2 shows the hardware structure of modular reduction with multiplication. It uses two additions, five subtractions and shift operations.

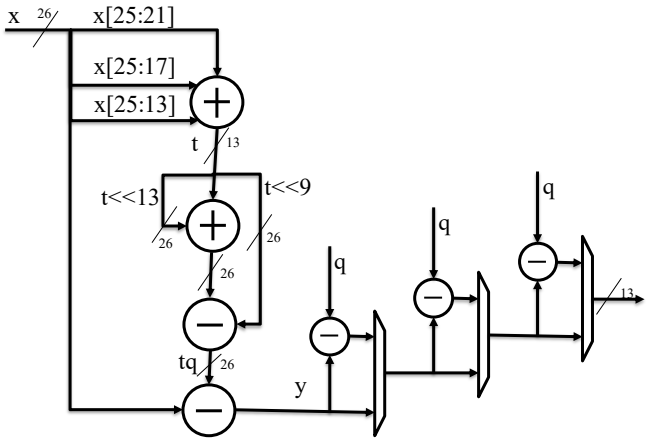

Fig. 2. Hardware structure of the modular reduction.

C. Hardware Architectures of R-LWE

The R-LWE algorithm includes three parts, key generation, encryption, and decryption. In the key generation part, two Gaussian distributed polynomials $\boldsymbol{r_1}, \boldsymbol{r_2}$ and a uniformly distributed polynomial $\boldsymbol{a}$ are produced. Polynomial $\boldsymbol{a}$ is sampled directly from M-sequence. Polynomial multiplication and subtraction is used to obtain the public key $\boldsymbol{p}$ and the private key $\boldsymbol{r_2}$. Polynomial subtraction can be performed by addition and using a multiplexer. When the addition operation is finished, there is a simple modulus operation with a comparator using a multiplexer. Fig. 3 is the hardware architecture of the key generation process.

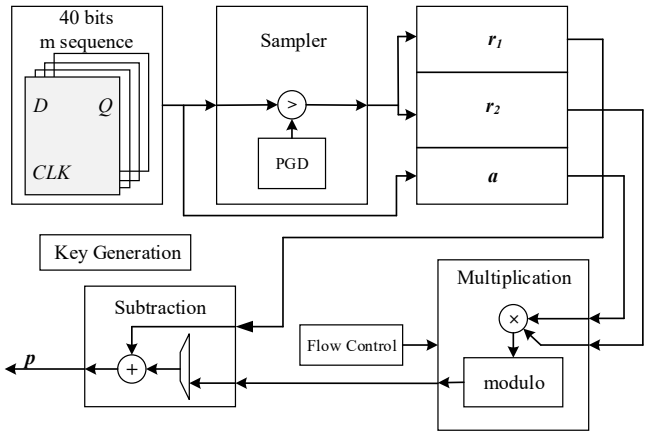

Fig. 3. Hardware architecture of the key generation process.

For encryption, Gaussian noise polynomials $\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}$ are required for the ciphertext calculation. Every addition needs another multiplexer to ensure the result is below 7681. Message m is hidden by $\boldsymbol{c_2}$. The polynomial $\boldsymbol{c_1}$ does not contain any information of the message and it is just an assistant during the decryption process. Fig. 4 shows the hardware architecture of the encryption process.

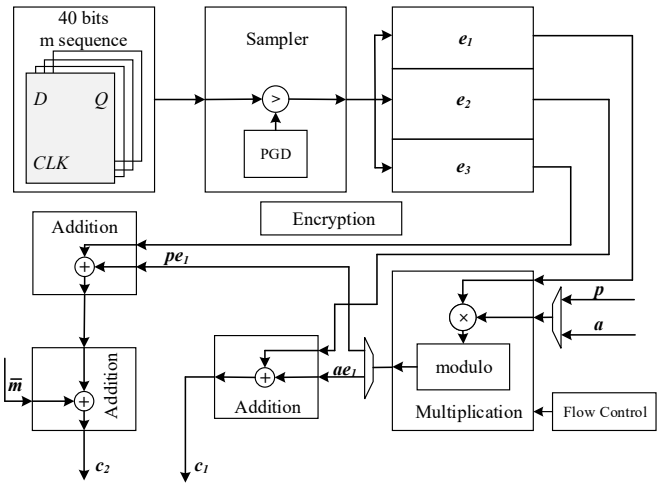

Fig. 4. Hardware architecture of the encryption process.

TABLE II
RESULT OF PERFORMANCE AND RESOURCE CONSUMPTION

K7 (256,7681,4.51)			
Resources	Key	Enc	Dec
LUT	934	1098	609
FF	356	407	318
Slices	289	337	182
LUTM	195	281	215
BRAM	0	0	0
DSP	1	1	1
Freq.	288.35MHz		
Cycles	66057	131604	65802
Op./s	4365.17	2191.04	4382.09

TABLE III
COMPARISON WITH OTHER R-LWE IMPLEMENTATIONS (K-7: KINTEX-7, S-6: SPARTAN-6, V-6: VIRTEX-6, S-IV: STRATIX-IV).

Designs	Part	LUT	FF	BRAM	DSP	Freq (MHz)	Op./s
This work (K-7)	Key	934	356	0	1	288.35	4365
	Enc	1098	407	0	1		2191
	Dec	609	318	0	1		4382
[12] (S-6)	Enc	360	290	2	1	128	934
	Dec	162	136	1	1	179	2700
[11] (V-6)	Tol	4549	3624	12	1	262	14162
[15] (S-IV)	Enc	28977	29290	234	22	235.40	107181
	Dec	6761	7616	31	20	249.44	217864

Lastly, in the decryption, there is no need to generate Gaussian values, which only need to calculate $\boldsymbol{c = c_1 r_2 + c_2}$. Fig. 5 shows the hardware architecture of the decryption process.

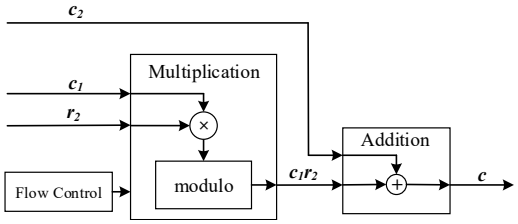

Fig. 5. Hardware architecture of the decryption process.

IV. RESULTS AND COMPARISON

We have implemented the proposed R-LWE encryption scheme on Xilinx Kintex-7 FPGA (using Vivado 2016.4) with parameter sets (256,7681,4.51). Table II presents the performance and hardware cost. The design includes three parts, which do not share any hardware, due to that the key management, encryption and decryption core may not be contained in the same device. Pipelining has been applied to increase the frequency of design with registers. In addition, we use LUT RAM (LUTM) instead of block memory, which can save the memory resources and achieve higher frequency, but costs extra LUT resources. Table III presents the comparison with other R-LWE encryption implemented in hardware.

[12] uses the different parameter sets (256,4093,3.33) and its frequency is not high but it is low-cost. [11] uses the NTT algorithm as polynomial multiplication and reuse the same multiplication module. [15] is a high throughput design and high security with $n = 512$, which targets high speed. But it costs much more DSP blocks and RAM resource. Comparing with these R-LWE designs, the proposed design does not use any RAM and has the highest frequency with rather low hardware cost.

V. CONCLUSIONS

In this work, a lightweight R-LWE hardware implementation is presented. The proposed design uses CDT as the Gaussian sampler and a schoolbook multiplier for polynomial multiplication. The design achieves the highest frequency with rather low hardware cost, compared with previous works.

ACKNOWLEDGMENT

This has been supported by NSFC (61871216) and Six Talent Peaks Project of Jiangsu Province (XYDXX-009).

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Computing*, vol. 26, pp. 1484–1509, 1997.
[2] E. Öksüzoglu and E. Savas, "Parametric, secure and compact implementation of RSA on FPGA," in *Int. Conf. Reconfigurable Computing & FPGAs*, pp. 391–396, 2008.
[3] K. Sakiyama, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p)," *Int. J. Electronics*, vol. 94, pp. 501–514, 2007.
[4] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology – EUROCRYPT*, pp. 1–23, 2010.
[5] R. Overbeck and N. Sendrier, "Code-based cryptography," in *Post-Quantum Cryptography*, pp. 95–145, Springer Berlin Heidelberg, 2009.
[6] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT*, pp. 368–397, Springer Berlin Heidelberg, 2015.
[7] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *Applied Cryptography and Network Security*, pp. 164–175, Springer Berlin Heidelberg, 2005.
[8] C. Liu, J. Ni, W. Liu, Z. Liu, and M. O'Neill, "Design and optimization of modular multiplication for SIDH," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1–5, 2018.
[9] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Topics in Cryptology – CT-RSA*, pp. 319–339, 2011.
[10] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in *Progress in Cryptology – LATINCRYPT*, pp. 139–158, Springer Berlin Heidelberg, 2012.
[11] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Selected Areas in Cryptography – SAC*, pp. 68–85, Springer Berlin Heidelberg, 2014.
[12] T. Poppelmann and T. Guneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 2796 – 2799, 2014.
[13] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *Advances in Cryptology — CRYPTO*, pp. 311–323, Springer Berlin Heidelberg, 1986.
[14] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede, "Efficient ring-LWE encryption on 8-bit AVR processors," in *Lecture Notes in Computer Science*, pp. 663–682, Springer Berlin Heidelberg, 2015.
[15] C. P. Renteria-Mejia and J. Velasco-Medina, "High-throughput ring-LWE cryptoprocessors," *IEEE Trans. VLSI Systems*, vol. 25, pp. 2332–2345, 2017.