격자 암호화를 위한 메모리 절약 방식의 정수 FFT

*1 문상국
1 목원대학교, smoon@mokwon.ac.kr

Memory-efficient Integer Fourier Transform for Lattice Cryptography

*1 Sangook Moon
1 Mokwon University, smoon@mokwon.ac.kr

요 약

완전 동형 암호화 시스템을 구현하기 위해서 가장 핵심이 되는 연산은 암호화 (encrypt), 복호화 (decrypt), 그리고 재암호화 (recrypt)라고 알려져 있다. 각각의 연산은 모두 백만 비트 이상의 정수에 대한 다항식 법 곱셈 (modulo multiplication)을 기본적인 연산으로 요구한다. 백만 비트 이상의 정수에 대한 법 곱셈은 정수 FFT 로 가능하다. 본 논문에서는 Schonhage-Strassen 알고리즘을 응용하여 메모리를 효율적으로 사용하는 정수 법 곱셈을 제안하고 이를 차세대 암호화 방식인 Ring-LWE 프로세서에 적용하여 FPGA 에 구현하여 성능을 검증하였다. 제안한 방식으로 V6LX75T 에 구현한 Ring-LWE 프로세서는 14k LUT, 1 BRAM18 을 소모하고 암호화, 복호화에 42us, 18us 의 성능을 각각 보인다.

Abstract

The three indispensable operations realizing a fully homomorphic encryption system are encrypt, decrypt, and recrypt. Each operation requires polynomial modulo multiplication over million bits of operands in common, which can be obtained by integer Fourier Transform. In this paper, we apply and modify Schonhage-Strassen algorithm to propose a memory efficient integer multiplication method and implement a Ring-LWE processor on an FPGA equipped with it. Using V6LX75T, we evaluated and validated our method which shows memory consumption of 14k LUT and 1 BRAM18. Performance analysis represents 42 us and 18 us in calculating encryption and decryption respectively.

Keywords: *Fully homomorphic encryption, Integer Fourier transform, Polynomial modulo multiplication, Memory saving, FPGA*

1. 서론

동형 암호화 (HE; homomorphic encryption)란 암호 기술의 일종으로 정보들이 암호화 된 상태에서 특정한 연산을 수행했을 경우 그 결과를 복호화 했을 때 같은 정보들이 암호화되지 않은 상태에서 연산된 결과와 같은 결과를 보이며, 암호화 된 상태에서 정보들이 난수의 성질을 가지는 암호 시스템을 말한다. 암호화 시스템이 하나의 연산에 대해 위와 같은 성질을 만족할 때 동형 암호화 시스템이라고 하며 덧셈과 곱셈에 대해 위의 성질을 만족하면서 정수론 상의 환 (ring) 특성을 보일 때 이를 완전 동형 암호화 (FHE; fully

* Corresponding Author
Received: Nov. 10, 2017, Revised: Nov. 25, 2017, Accepted: Dec. 21, 2017

homomorphic encryption)시스템이라고 한다[1-3]. 완전 동형 암호 시스템의 특징을 수식으로 기술하면 아래 식 1 과 같다 (E : 암호화, m : 정보, k : 열쇠).

$$E_k(m_1) + E_k(m_2) = E_k(m_1 + m_2), E_k(m_1) * E_k(m_2) = E_k(m_1 * m_2) \quad (1)$$

완전 동형 암호화는 향후 클라우드 시스템에 저장되는 큰 데이터, 병원 처방 기록, 세금 보고 기록, 은행 기록 등 대리인에게 노출될 수 있는 민감한 정보를 근본적으로 보호한다[4]. 본 논문에서는 완전 동형 암호화 시스템을 처리할 수 있는 Ring-LWE (Learning With Error) 프로세서를 격자 암호화 (lattice cryptography)를 적용하여 구현할 때 꼭 필요한 다항식 법 곱셈을 처리하는 데 있어 메모리를 절약하는 방법에 대한 알고리즘을 제시하고 FPGA 상에 구현하여 성능을 평가한다.

2. 정수 FFT (NTT; Number Theoretic Transform)

$$NTT_w(a):$$

$$A_i = \sum_{j=0}^{n-1} a_j w^{ij} \bmod p, i = 0,1,\cdots n-1$$

$$NTT_w^{-1}(A):$$

$$a_i = n^{-1} \sum_{j=0}^{n-1} A_j w^{-ij} \bmod p, i = 0,1,\cdots n-1$$

```
Input: Polynomial a(x) ∈ Z_q[x] of degree n − 1 and n − th primitive root w_n ∈ Z_q of unity
Output: Polynomial A(x) ∈ Z_q[x] = NTT(a)
A <- bit_Inverse(a(x))
m <- 2
while m≤N {
        s <- 0
        while s < N {
                for i to m/2-1 {
                        N <- i.n/m
                        a <- s + i
                        b <- s + i + m/2
                        c <- A[a]
                        d <- A[b]
                        A[a] <- c + ω^(Mmod n)d mod q
                        A[b] <- c - ω^(Mmod n)d mod q
                        }
                s <- s + m
                }
        m <- m*2
        }
return A
```

Figure 1. Shonhage-Strassen number theoretic algorithm

암호화, 복호화, 그리고 재암호화 계산은 완전 동형 암호화 시스템을 구현하는데 뼈대 계산을 수행한다. 그리고 각 계산에 공통적으로 필요한 연산은 백만 비트가 넘는 오퍼랜드에 대한 법 곱셈이며, 이는 두 단계로 나뉘어 진다. 하나는 FFT 를 반복적으로 수행하는 큰 정수들에 대한 곱셈 과정이고 다른 하나는 곱셈 결과물에 대한 법 축약 (modular reduction)이다. 정수 FFT 를 수행하기 위해서는 Schonhage-Strassen 알고리즘이 좋은 성능을

보인다[5]. 그림 1 은 Schonhage-Strassen 을 나타낸다. 여기서 $\mathrm{nmod}(p-1)=1$이고 n 이 2 의 제곱 수라고 가정하면 NTT 는 O(nlogn) 시간 내에서 계산된다. 유한체 Z_p 에서 소수 n 이 있어서 ω 의 n 제곱이 주어졌을 때 ($\omega^n = 1 \ mod \ p$) 벡터 $\{a_0, \cdots, a_{n-1}\}$ 에 대하여 벡터 $\{A_0, \cdots, A_{n-1}\}$ 로의 정수 FFT 와 그 역변환, 즉 $\mathrm{NTT}_w(a)$ 와 $\mathrm{NTT}_w^{-1}(A)$ 는 다음과 같이 정의할 수 있다 [6-8].

3. Ring-LWE 암호화와 복호화

본 논문에서 제안하는 메모리 절약 방식의 법 곱셈 방법을 적용하기 위해서는 타겟 Ring-LWE 프로세서 상에서 메시지에 대한 암호화, 복호화의 정의가 필요하다. 그림 2, 그림 3 은 덧셈, 곱셈에 대한 암호화와 복호화가 제대로 되는 지에 대한 증명이다.

메시지는 다항식 표현의 m 이며 이는 소수 다항식 x^n+1을 근으로 하는 유한체 $Z_2[x]$ 상에 존재한다. 암호화된 메시지는 한 쌍의 c = (c_0, c_1) 으로 표현하며, 소수 다항식 x^n+1을 근으로 하는 유한체 $Z_q[x]$ 상에 존재한다. 이 때, c_0 과 c_1 을 그림 2 에서와 같이 정의해주면, 암호화 시와 복호화 시 덧셈이 유효하다는 것을 알 수가 있다.

Message : m (polynomial) $\in R_2 = Z_2[x] / < x^n + 1 >$
Ciphertext : c (pair of polyomials) $\in R_q = Z_q[x] / < x^n + 1 >$

$c = (c_0, c_1)$ $\quad c_0 = a * s + 2 * e + m$, $\quad c_1 = -a$ — Encrypt

$$c_{add} = c + c' = (c_0 + c_0', \quad c_1 + c_1')$$
$$= (a * s + 2 * e + m + a' * s\, 2 * e' + m', -(a + a'))$$
$$= ((a + a') * s + 2 * (e + e') + \ m + m', -(a + a'))$$

$c_{add} = (c_{add_0}, c_{add_1})$ $\quad find \ \ c_{add_0} + c_{add1} * s$ — Decrypt (+)

$$(a + a') * s + 2 * (e + e') + \ m + m' \ - (a + a') * s$$
$$= 2 * (e + e') + \ m + m' \cong m + m' \ (mod \ q)$$

Figure 2. Definition of *encryption* and *decryption* for addition

또한, 암호화된 텍스트 c 를 그림 2 에서와 같이 정의해준 상태에서 암호화된 텍스트에 대한 곱셈을 그림 3 과 같이 정의해준 다음 그 결과를 복호화 하는 경우 최종 결과 값이 원래의 메시지 m 과 m'을 곱한 결과와 같다는 것을 보인다.

4. 메모리 절약 방식의 NTT

[11]에서 저자들은 기본적으로 n 번 반복 수행하는 알고리즘을 차원 개념을 도입하여 수정하여 자원을 중복 사용함으로써 메모리를 절감하고자 하였다. 본 논문에서는 이 개념을 더 깊이 적용하여 다차원 추상화를 제안한다. 실험적으로 적용한 4 차원 추상화를 도입하여 제안하는 알고리즘은 그림 4 와 같다. 시간과 자원을 트레이드-오프 대상으로 삼고, 타겟 어플리케이션의 허용 수치 안에서 실행 시간은 증가하게 되고, Ring-LWE 프로세서 상의 구현 면적은 대폭 절감할 수 있다.

Message : m (polynomial) $\in R_2 = Z_2[x] / < x^n + 1 >$
Ciphertext : c (pair of polyomials) $\in R_q = Z_q[x] / < x^n + 1 >$

$c = (c_0, c_1)$ $\quad c_0 = a * s + 2 * e + m$, $\quad c_1 = -a$ — Encrypt

$$c_{mul} = c * c' = (c_0 * c_0', \quad c_0 * c_1' + c_0' * c_1, \quad c_1 * c_1')$$

$$c_0 * c_0' = (-a * a' * s^2 + (c_0 * a' + c_0' * a) * s + 2 * (2 * e * e' + e * m' + e' * m) + m * m'$$

$$c_1 * c_1' = a * a \qquad c_0 * c_1' + c_0' * c_1 = \ldots$$

$c_{mul} = (c_{mul_0}, c_{add_1}, c_{mul_2})$ — Decrypt (*)

$find \quad c_{mul_0} + c_{mul1} * s + c_{mul_2} * s^2$

$$c_{mul_0} + c_{mul1} * s + c_{mul_2} * s^2 = (c_0 + c_1 * s) * (c_0' + c_1' * s) \cong m * m' \pmod q$$

Figure 3. Definition of *encryption* and *decryption* for multiplication

```
begin
    A ← BitInverse(a);
    for (m = 2 ; n/4 ; m = 4m) {
        ω_m ← m − th root of unity;
        ω ← √ω_m or 1
        for (j = 0 ; m/4 − 1) {
            for (k = 0 ; n/4 − 1 ; m) {
            (t_1, u_1) ← (A[k + j + m/4], A[k + j])
            (t_2, u_2) ← (A[k + m + j + m/4], A[k + m + j])
              t_1 ← ω • t_1;
              t_2 ← ω • t_2;
(A[k + j + m/4], A[k + j]) ← (u_1 − t_1, u_1 + t_1);
(A[k + m + j + m/4], A[k + m + j]) ← (u_2 − t_2, u_2 + t_2);
        mem[k + j] ← (A[k + j + m], A[k + j]);
        mem[k + j + m/4] ← (A[k + j + m/4], A[k + j + m/4]);
        mem[k + j + 2m/4] ← (A[k + j + 2m/4], A[k + j + m/4]);
        mem[k + j + 3m/4] ← (A[k + j + 3m/4], A[k + j + m/4]);
            }
            ω ← ω • ω_n;
        }
    }
    m ← n;
    k ← 0;
    ω ← √ω_m or 1
    for (j = 0 ; m/2 − 1) {
        (t_1, u_1) ← (A[j + m/2], A[j])
        t_1 ← ω • t_1;
        (A[j + m/2], A[j]) ← (u_1 − t_1, u_1 + t_1);
        mem[j] ← (A[j + m/2], A[j]);
        ω ← ω • ω_m;
    }
}
```

Figure 4. n-dimension memory-efficient NTT algorithm ($n = 4$)

5. 성능 평가 및 결론

그림 5 는 본 논문의 성능을 검증하기 위해 FPGA 로 구현한 Ring-LWE 격자 암호 프로세서이다. Ring-LWE 암호화를 수행하기 위해 NTT 가 필요하였고, 우리는 4 차원 추상화 개념을 적용한 알고리즘을 제안하여 타겟 프로세서에 구현하였다. 성능은 표 1 에 보인다. 평가 결과는 타원 곡선 암호화 (ECC), 다른 형태의 격자 암호화인 NTRU (Number Theorists R Us) 들과 비교하였을 때 메모리의 사용율이 현저하게 낮아진 것을 알 수가 있다. 그리고 [11]은 2 차원 추상화를 적용한 연구이며 본 논문의 결과 비교하여 볼 때 허용 가능한 만큼 시간을 소모하면서 메모리를 추상화한 차원에 비례한 만큼 절약한 결과를 보인다.

격자 암호화 방식을 적용하여 완전 동형 암호 시스템을 구현하기 위해 가장 핵심적인 연산은 백만 비트 이상에 대한 법 곱셈인데, 이는 곱셈 결과에 정수 FFT 를 반복적으로 수행하면서 얻을 수 있다. 본 논문에서는 메모리 절약 대비 소모 시간을 허용 시간만큼 소비하면서 효율적으로 NTT 연산을 수행하는 알고리즘을 제안하였고, 이는 추후 격자 암호 기반의 모든 연산기에 적용이 가능하다.

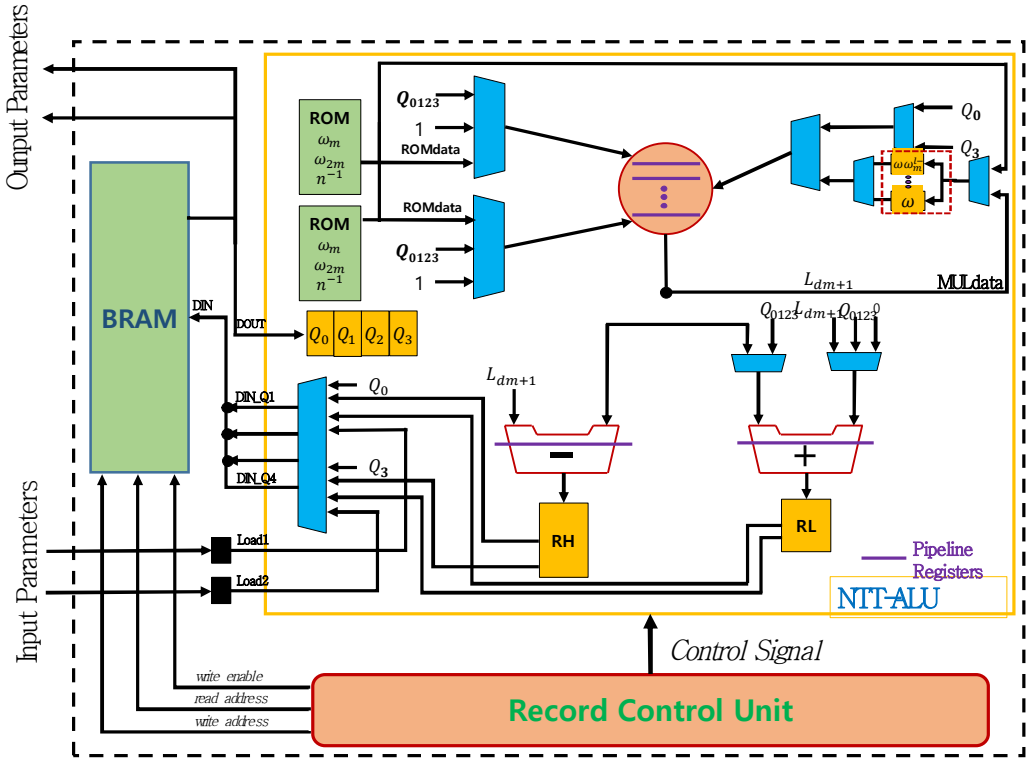

Figure 5. Target crypto-processor

Table 1. Modified NTT results

Implementation	*Parameters*	*Device*	*LUT/ BRAM18*	*Freq(Mhz)*	*Cycles/Time (us)*	
					Enc	*Dec*
ECC	Binary-233	V5LX85T	18k/0	156	1.9k/12.3	1.9k/12.3
NTRU	NTRU-251	XCV1600E	27k/0	62.3	-/1.54	-/1.4
[11]	(256,7681,11,32)	V6LX75T	13k/2	313	6.6k/20.1	2.8k/9.1
Our work	(256,7681,11,32)	V6LX75T	7k/1	306	13k/42	5k/18

6. 감사의 글

이 논문은 2017 년도 목원대학교 연구년 지원에 의하여 연구되었음.

7. 참고문헌

[1] Gentry. C, "A fully homomorphic encryption scheme," Ph.D. dissertation, ECE, SU, CA, 2009, http://crypto.stanford.edu/craig.
[2] Ray A. Perlner and David A. Cooper, "Quantum Resistant Public Key Cryptography: A Survey," in Proc. of IDtrust 2009, Gaithersburg, MD, Apr. 2009, pp. 85-93.
Nigel P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," IACR Cryptology ePrint Archive, 2011.
[3] Tsz-Wo Sze, “Schonhage-Strassen Algorithm with MapReduce for Multiplying Terabit Integers,” Proc. of the 2011 International Workshop on Symbolic-Numeric Computation, San Jose, California, 2011, Jun. 7-9.
[4] M. Ayinala, M. Brown, K.K. Parhi, “Pipelined parallel FFT architectures via folding transformation,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol.20, pp. 1068-1081, no. 6, Jun. 2012.
[5] Jung Hee Cheon, “Batch Fully Homomorphic Encryption over the Integers,” Proc. of Advances in Cryptology – EUROCRYPT 2013, Lecture Notes in Computer Science Vol. 7881, 2013, pp. 315-335
[6] Wei Wang, “Exploring the Feasibility of Fully Homomorphic Encryption,” IEEE Trans. Comput., Vol. 64, Issue 3, pp. 698-706, Mar. 2013.
[7] 1363 working group of the C/MSC committee, IEEE P1363.1 D12 Draft Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices, Mar. 2009.
[8] Arvind, Rishiyur S. Nikhil, Joel S. Emer, and Murali Vijayaraghavan, Computer Architecture: A Constructive Approach, draft version, 2012.
[9] Byoungcheon Lee, "Hierarchical ID-based Encryption with Minimized Key Escrow", Journal of Security Engineering, Vol.12, No.6 (2015), pp.545-552, http://dx.doi.org/10.14257/jse.2015.12.01
[10] Yeon Tae Kim, Hyoseung Kim, HyoJin Jo, Dong Hoon Lee, "Secure Messenger System using Attribute Based Encryption", Journal of Security Engineering, Vol.12, No.5 (2015), pp.469-486, http://dx.doi.org/10.14257/jse.2015.10.05
[11] S. S. Roy, “Compact Ring-LWE Cryptoprocessor,” Lecture Notes in Computer Science, Vol. 8731, pp. 371-391, 2014.