ELK: 래티스 기반 암호 라이브러리의 확장[1)]

안형철* Muhamad Erza Aminanto** 최락용** 김광조*,**

*카이스트 정보보호대학원/ **전산학부

ELK: Extending Lattice-based Cryptographic Library by KAIST

Hyeongcheol An* Muhamad Erza Aminanto**

Rakyong Choi** Kwangjo Kim*,**

*Graduate School of Information Security/ **School of Computing, KAIST

요 약

래티스 기반 암호는 포스트 양자 암호(Post Quantum Cryptography) 방식 중 하나로 알려져 있다. 현재 래티스 기반 암호 라이브러리는 Microsoft사에서 만든 LatticeCrypto(Lattice Cryptography Library)가 있으나, 키 교환 프로토콜을 위하여 제작되어 일반적인 래티스 기반 암호 프리미티브에 적용하기 어렵다. 이에, 본 논문은 일반적으로 래티스 기반 암호를 구현할 수 있는 Extending Lattice-crypto library by KAIST(ELK) 라이브러리를 설계하고 실험 결과를 제시하였다.

I. 서론

현재 널리 사용하고 있는 RSA와 DH방식은 소인수분해 또는 이산대수문제의 어려움에 기반하고 있다. 그러나 이 방식들은 Shor 알고리즘[1]에 따라 양자컴퓨터가 나오게 되면 다항식 시간 내에 해독이 가능해진다. 양자컴퓨터의 공격에도 안전한 포스트 양자 암호의 필요성의 제기되고 있으며, 그 중 하나로 래티스 기반 암호 알고리즘이 알려져 있다.

현재 래티스 기반 암호 라이브러리는 Microsoft사의 LatticeCrypto(Lattice Cryptography Library)[2]가 있다. 그러나 이 라이브러리는 키 교환 프로토콜을 위하여 만들어져 일반적인 적용에는 한계가 있고, 래티스를 기반한 전자서명이나 암호화 방식에 사용하기는 어렵다.

본 논문은 키 교환뿐만 아니라 전자서명이나 암호화 과정에도 사용할 수 있는 통합적인 라이브러리인 Extending Lattice-crypto library by KAIST(ELK)에 대하여 구성 방안과 기본적인 2가지 실험을 제시하였다.

1) 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2015R1A2A2A01006812).

II. 배경지식

ELK는 오픈소스 라이브러리인 NTL (Number Theory Library)[3], GMP(GNU Multiple Precision Arithmetic Library)[4]를 이용하여 구현하였다. 위의 두 라이브러리는 동시에 사용할 수 있어 활용도가 높다. 이 장에서는 NTL과 GMP 라이브러리에 대하여 간단히 기술하였다.

2.1 NTL 라이브러리

NTL 라이브러리는 Shoup에 의해 빠른 암호학적 연산을 위해 만들어진 라이브러리이다. C와 C++로 구현되어 있어 연산속도가 빠르며 리눅스 시스템에서 설치가 가능하다. NTL 라이브러리가 지원하는 연산 알고리즘은 임의의 길이를 가진 정수 연산, 부동소수점 연산 알고리즘, 정수와 유한체 기반의 다항식 연산 알고리즘, 래티스 축약 알고리즘, 그리고 기본적인 선형대수 연산이 있다.

2.2 GMP 라이브러리

GMP 라이브러리는 정수 및 소수의 연산 라이브러리이며, 특히 암호 프리미티브의 개발 및 연구를 위해 고안되었다. 리눅스 시스템에서 설치가 가능하며 Cygwin 도구가 설치된 윈도우 시스템에서도 사용이 가능하다. 앞서 제시한 NTL 라이브러리와 달리 무제한의 길이를 가진 연산이 가능하다는 장점이 있다.

III. ELK

3.1 구현 및 검증

본 논문에서 제시하는 ELK는 C++로 구현하였고, 크게 ELK.cpp, ELK_math.cpp, 그리고 ELK_utility.cpp로 구성되어있다.

ELK.cpp는 베르누이 행렬(Bernoulli Matrix) 및 가우시안 행렬(Gaussian Matrix)을 생성하는 기능을 가지고 있으며 샘플링(Sampling) 과정에서 사용할 수 있다. 또한 LWE(Learning with Errors) 행렬과 임의의 벡터에 대하여 사영(projection)하는 기능이 있다. 또한 Closest Vector Problem(CVP)문제를 풀기 위한 Klein 알고리즘[5]을 구현하였다. ELK_math.cpp에서는 기본적인 연산인 순열 계산과 크기, 해밍무게를 계산하는 함수가 구현되었고, ELK_utility.cpp는 정수 벡터 및 행렬을 실수로 변환하는 기능을 가지고 있다.

3.2 ELK를 이용한 실험

본 실험은 리눅스 우분투 14.04 버전에서 수행하였다. 가우시안 행렬과 주어진 기저에서의 Gram-Schmidt 행렬을 생성하였다. 래티스 기반 암호에서 가우시안 행렬을 사용하여 샘플링하는 과정은 필수적으로 사용되고 있으며[6], Gram-Schmidt 행렬도 NTRUSign[7]에서 중요하게 사용한다.

3.2.1 가우시안 행렬 생성

가우시안 행렬을 생성하기 위하여 필요한 함수는 RandMat_Gen, GaussDistSample(GDS)[표 1] 그리고 GaussMat_Gen(GMG)[표 2]이다. 생성과정을 간략히 설명하면 주어진 크기의 행렬에 대하여 임의의 행렬을 만들고, GDS 함수를 이용하여 가우시안 행렬을 생성한다.

표 1. GDS 함수

GDS()
long GDS (const double c, const double s, const double k)
Input: c: 분포의 중심, s: 표준편차, k: 표준화 거리 Output: 가우시안 분포에서 샘플링한 정수 값

표 2. GMC 함수

GMat()
void GMat (mat_ZZ & result, const long n, const long m, const double σ, const double c, const double p, const ZZ q, bool row)
Input: n,m: 행렬 크기, σ: 표준편차, c: 분포의 중심 p: 표준화 거리, q: 소수 Output: 정수 가우시안 행렬

파라미터의 크기를 $n=m=5$, $\sigma=3.2$, $c=1.1$, $p=500.1$, $q=31$로 하였을 때, 실제 위의 기능들을 이용하여 실험한 결과는 [그림 1]과 같다.

```
Generating the Gaussian matrix randomly :

[[1 1 -1 1 1]
[1 1 2 0 3]
[1 1 2 2 -1]
[-1 0 1 3 -1]
[1 1 2 1 2]
]
```

그림 1. 임의의 5×5 가우시안 행렬

3.2.2 Gram-Schmidt 행렬 생성

Gram-Schmidt 행렬을 생성하기 위하여 먼저 LWE 행렬을 만든 뒤, 이 행렬을 사용하여 Gram-Schmidt 행렬을 생성한다. 이 과정을 수행하기 위하여 필요한 함수는 LWEMat[표 3]과

GSMat[표 4]이다.

표 3. LWEMat 함수

LWEMat()
void LWEMat (mat_ZZ & result, const long n, const long m, const ZZ q)
Input: n,m: 행렬 크기, q: 소수 Output: 임의의 $n \times m$ LWE 행렬

표 4. GSMat 함수

GSMat()
void GSMat (mat_RR & result, const mat_ZZ & A)
Input: A: 정수 행렬 Output: A의 Gram-Schmidt 행렬

파라미터의 크기를 $n=m=5$, $q=31$ 으로 하였을 때, 생성한 LWE 행렬과 Gram-Schmidt 행렬은 [그림 2]과 같다.

```
Generating LWE matrix :

[[28 17 10 7 4]
[17 2 4 14 7]
[12 26 11 8 17]
[2 4 24 24 3]
[1 4 16 3 11]
]
Generating Gram-Schmidt matrix for a given basis :

[[28 17 10 7 4]
[1.71082391 -7.282714055 -1.460420032 10.17770598 4.815831987]
[-10.68869606 11.25273074 2.654939793 3.466680648 14.29272619]
[-7.221312702 -0.8458082907 19.94890386 8.432580284 -10.48540098]
[1.56243976 -4.751228452 5.368289482 -5.358062398 5.211528088]
]
```

그림 2. 임의의 5×5 LWE 행렬 및 Gram-Schmidt 행렬

IV. 결론 및 향후 연구

본 논문에서는 포스트 양자 암호로 주목받고 있는 래티스 기반 암호를 구현할 수 있는 라이브러리의 소개와 두 가지의 기본 기능 실험을 수행하였다. ELK는 Microsoft사의 LatticeCrypto 라이브러리와는 달리, 키 교환 프로토콜에만 한정되어 있지 않기 때문에 다양한 래티스 기반 암호 프리미티브에 적용할 수 있다.

한편, 활발히 연구 중인 동형암호[8]에서도 샘플링과정이 필요하기 때문에 ELK를 활용하여 구현할 수 있으며, 기존의 래티스 기반 암호 알고리즘의 고속구현에도 활용할 수 있다.

[참고문헌]

[1] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in Proceedings of the Foundations of Computer Science, 35th Annual Symposium on IEEE, 1994, pp. 124-134.

[2] LatticeCrypto Library. https://www.microsoft.com/en-us/research/project/lattice-cryptography-library/

[3] NTL Library. http://www.shoup.net/ntl/

[4] GMP Library. https://gmplib.org/

[5] P. Klein, “Finding the Closest Lattice Vector when It’s Unusually Close,” in Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), 2000, pp. 937-941.

[6] O. Goldreich, S. Goldwasser and S. Halevi, “Public-Key Cryptosystems from Lattice Reduction Problems,” in Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 1997), 1997, pp. 112-131.

[7] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman and W. Whyte, “NTRUSign: Digital Signatures Using the NTRU Lattice,” in the Proceedings of Cryptographers’ Track at the RSA Conference (CT-RSA 2003), 2003, pp. 122-140.

[8] C. Gentry, “A Fully Homomorphic Encryption Scheme,” Doctoral dissertation, Stanford University, 2009.