Implementation of Fully Homomorphic Encryption using NFLlib

Arinta Primandini Auza
Research Center for Informatics
Indonesian Institute of Sciences
Indonesia, Cibinong 16916
Email: arinta.primandini.auzarifki.sadikin@lipi.go.id

Rifki Sadikin
Research Center for Informatics
Indonesian Institute of Sciences
Indonesia, Cibinong 16916
Email: arinta.primandini.auza@lipi.go.id

***Abstract*—Fully homomorphic encryption allows us to perform the computation on encrypted data. After Gentry showed that this scheme is implementable, research on this area has developed rapidly. Several cyrptosystems based on the hardness of LWE problem have been proposed to achieve the fully homomorphic properties. In this study, our aim is to implement the Ring-based BGV scheme with NFLlib and to compare the performance with previous studies. NFLlib is an open-source C++ library dedicated to ideal lattice cryptography which optimizes the underlying operations on $R_p = Z_p[x]/x^d + 1$.**

***Keywords*—FHE, NFLlib, BGV.**

I. Introduction

The idea of homomorphic encryption was first suggested by Rivest, Adleman and Dertouzos [1]in 1978. However, before Gentry's breakthrough in 2009 several candidates were partially homomorphic such as Pailier cryptosystem [2] which is homomorphic under addition and ElGamal public key cryptosystem [3] which is homomorphic under multiplication. Gentry [4] showed that fully homomorphic encryption can be realised by modifying a somewhat homomorphic scheme to be bootstrappable and introducing squashing step to reduce the noise in the ciphertext. His cryptosystem used ideals over polynomial rings which security relies on the hardness of problems on ideal lattices. Since this idea was published, various schemes such as [5] [6] [7] [8] [9] have been proposed with some improvements from previous schemes.

Fully homomorphic encryption has been dubbed as the 'holy grail' of cryptography because it opens the door to many new capabilities in the cloud-centric, data-driven world. We can perform computation on encrypted information without having to decrypt it beforehand. Numerous practical applications have been proposed such as consumer privacy in advertising, medical applications, data mining, financial privacy, and forensic image recognition [10].

There are several libraries available in order to implement lattice based fully homomorphic encryption. The generic number theory library NTL [11] is used in a lot of implementation of homomorphic encryption. HElib [12] [13] implements BGV scheme and modifies the internal routines of NTL to achieve better performance. Another generic number theory library is FLINT [14] which implements and compares FV [15] and YASHE [16] schemes. Moreover, improvement on performances of polynomial operations is achieved by NFLlib [17] which allows more efficient ideal lattice cryptography.

In this study, we use NFLlib to implement one of the fully homomorphic encryption schemes which is called BGV scheme [7] and compare our implementation with HElib's implementation of this scheme. BGV scheme improved previous techniques by Brakerski and Vaikuntanathan [6]. This scheme introduced a noise-management technique without the use of bootstrapping procedure which is the key ingredient in previous schemes. We measure the performance of this scheme and provide the comparison between our implementation and HElib's implementation.

The paper is structured as follows : Section 2 provides basic notions of fully homomorphic encryption and RLWE. In Section 3 we describe the BGV scheme. Section 4 explains the implementation of BGV scheme on NFLlib. Section 5 discusses the performance and comparison with previous study. Finally, section 6 concludes the paper.

II. Preliminaries

A. Fully Homomorphic Encryption

An encryption scheme is *somewhat homomorphic* if given ciphertexts $c_1, c_2, \dots c_n$ that encrypt $m_1, m_2, \dots m_n$ we can compute a function f over those ciphertexts and output a new ciphertext that encrypts $f(m_1, m_2, \dots m_n)$ without noise reduction. There is no leak occurs throughout the evaluation of the function because all information are encrypted. A homomorphic encryption scheme is *compact* if there is a polynomial $p = p(\lambda)$ such that for every value of security parameter λ, the decryption algorithm can be expressed as a circuit of size at most p. Furthermore, a homomorphic encryption scheme is *fully homomorphic* if it compactly evaluate all arithmetic circuits.

B. Ring-LWE

Lyubashevsky, Peikert, and Regev [18] introduced Ring Learning With Errors (RLWE) problem which is defined as follow.

Definition. For security parameter λ, let $f(x) = x^d + 1 \in \mathbb{Z}[x]$ where $d = d(\lambda)$ is a power of 2. Let $q = 1 \mod 2d$ be a sufficiently large public prime modulus and $R_q = R/qR$. Fix a certain error distribution χ over R. The RLWE problem is to distinguish the following : **(a)** $\{(a_i, b_i)\}$ selected at random from R_q^2 and **(b)** $\{a_i, a_i \cdot s + e_i\}$ where $a_i, s \leftarrow_r R_q$ and $e_i \leftarrow_r \chi$. The RLWE assumption is that the two distributions are indistinguishable.

The RLWE problem is thought to be infeasible because there is a quantum reduction from worst-case approximate SVP on ideal lattices to the search version of RLWE.

III. BGV Scheme

In this section we will describe the RLWE based fully homomorphic encryption introduced in [7]. The scheme is set up with these parameters: Let R be a ring defined by $R = \mathbb{Z}[x]/f(x)$ where $f(x) = x^d + 1$ and d is a power of 2. Let q be a μ bits odd modulus, χ a noise distribution over R and N as additional parameter. These parameters are based on the security parameter λ which represents 2^λ security against known attacks.

A. Non Homomorphic Operations

We present the basic RLWE based encryption without homomorphic operations. These operations will be used as subroutines in the fully homomorphic encryption scheme.

```
E.Setup($1^\lambda, 1^\mu$):
    q = GenerateModulus($\mu$)
    d = $2^x$
    N = $\lceil 3\log q \rceil$
    $\chi$ = Gaussian()
    params = $(q, d, N, \chi)$

E.SecretKeyGen(params):
    s $\leftarrow_r \chi$
    sk = (1,s)

E.PublicKeyGen(params, sk):
    M $\leftarrow_r R_q^N$
    e $\leftarrow_r \chi^N$
    b $\leftarrow$ M $\cdot$ t + 2e
    pk = A = (b, -M)


E.Enc(params, pk, m):
    v $\leftarrow$ (m,0)
    t $\leftarrow_r R_2^N$
    c $\leftarrow$ v + $A^T \cdot t$

E.Dec(params, sk, c):
    m $\leftarrow$ (<c, sk> mod q) mod 2
```

B. (Leveled) Fully Homomomorphic Encryption Scheme based on RLWE

There are two key techniques used to achieve fully homomorphic encryption: key switching and modulus switching. Before we describe the techniques we will give several subroutines which will be useful for the procedures.

- BitDecomp$(\mathbf{x}, q)$: write $\mathbf{x} = \sum_{j=0}^{\lfloor \log q \rfloor} 2^j \cdot u_j$. Output $(u_1, u_2, \dots u_{\lfloor \log q \rfloor})$.
- Powersof2(x, q) : output $(x, 2.x, \dots 2 \cdot x)$.

The key switching procedure is used to reduce the dimention of the ciphertext and can transform a ciphertext c_1 that is encrypted by a secret key s_1 into a different ciphertext c_2 that encrypts the same message but decryptable by secret key s_2.

```
SwitchKeyGen($s_1, s_2$):
    N $\leftarrow n_1 \cdot \lceil \log q \rceil$
    A $\leftarrow$ E.PublicKeyGen($s_2$, N)
    $\tau_{s_1 \to s_2}$ = A + Powersof2($s_1$)

SwitchKey($\tau_{s_1 \to s_2}, c_1$):
    $c_2$= BitDecomp$(c_1)^T \cdot \tau_{s_1 \to s_2}$
```

Another procedure is called modulus switching which change the inner modulus of the decryption equation into a smaller number while preserving the correctness of the decryption under the same secret key.

Scale($\mathbf{x}$,p,q,r) : output $\mathbf{y}$ = the closest vector to $(p/q) \cdot \mathbf{x}$ where $\mathbf{y} = \mathbf{x} \mod r$.

After we have all the procedures set, we can describe the fully homomorphic encryption scheme as follows.

1) Key Generation: We use parameter L that indicates the number of levels of arithmetic circuit to evaluate the scheme.

```
FHE.Setup($1^\lambda, 1^L$) :
    $\mu = \theta(\log\lambda + \log$ L)
    for j=L to 0
        params[j] $\leftarrow$ E.Setup($1^\lambda, 1^{(j+1)\mu}$)
    for j=L-1 to 0
        d[j] = d[L]
        $\chi$[j] = $\chi$[L]

FHE.KeyGen(params):
    for j=L to 0
        s[j] $\leftarrow$ E.SecretKeyGen(params[j])
        A[j] $\leftarrow$ E.PublicKeyGen(params[j])
        s'[j] $\leftarrow$ s[j] $\otimes$ s[j]
        s"[j] $\leftarrow$ BitDecomp(s'[j],q[j])
        if (j $\neq$ L) then
            $\tau_{s"[j] \to s[j]}$ $\leftarrow$ SwitchKeyGen(s"[j],s[j])
    sk = s
    pk=(A[j], $\tau_{s"[j] \to s[j]}$
```

2) Encryption and Decryption: The encryption and decryption operation use the non homomorphic operations where the parameter l is used to identify the circuit depth.

```
FHE.Enc(params, pk, m):
    c' $\leftarrow$ E.Enc(A[L], m)
    c = (c', L)

FHE.Dec(params, sk, c): E.Dec(sk[l], c)
```

3) Addition and Multiplication: The Refresh procedure is important to reduce the noises that are produced by performing the addition and multiplication operations.

```
FHE.Add(pk,$c_1$,$c_2$):
    If $l_1 \neq l_2$ then
        j = $l_1$;
        FHE.Refresh($c_1, \tau_{s_{l_2} \to s_j}$,q[$l_2$],q[j])
    $c_3 \leftarrow c_1 + c_2 \mod$ q[j]
    $c_4$ $\leftarrow$ FHE.Refresh($c_3, \tau_{s''_j \to s_{j-1}}$,q[j],q[j-1])
```

```
FHE.Mult($pk, c_1, c_2$):
   If $l_1 \neq l_2$ then
      j = $l_1$;
      FHE.Refresh($c_1, \tau_{s_{l_2} \to s_j}$,q[$l_2$],q[j])
   $c_3 \leftarrow c_1 \cdot c_2 \mod$ q[j]
   $c_4 \leftarrow$ FHE.Refresh($c_3, \tau_{s''_j \to s_{j-1}}$,q[j],q[j-1])

FHE.Refresh(c, $\tau_{s_1,s_2}, q_1, q_2$):
   $c_1 \leftarrow$ Powersof2(c,q1)
   $c_2 \leftarrow$ Scale($c_1, q_1, q_2, 2$)
   $c_3 \leftarrow$ SwitchKey($\tau_{s_1,s_2}, c_2, q_2$)
```

REFERENCES

[1] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 169-177. Academic Press, 1978.

[2] Pascal Paillier. Composite-Residuosity Based Cryptography: An Overview. RSA Cryptobytes, pages 2026, 2002.

[3] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and David Chaum, editors, CRYPTO 1984, volume 196, pages 1018,1984.

[4] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[5] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 2443.

[6] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Manuscript, to appear in FOCS, 2011.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS, pages 309325. ACM, 2012.

[8] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Advances in Cryptology - Crypto 2012, volume 7417 of LNCS, pages 868886. Springer, 2012.

[9] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti and Garay [CG13], pages 75 92.

[10] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjsteen, Angela Jschke, Christian A. Reuter, Martin Strand: A Guide to Fully Homomorphic Encryption. IACR Cryptology ePrint Archive, 2015.

[11] Shoup, V.: Number Theory Library (Version 8.1) (2015), http://www.shoup.net/ntl

[12] Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 554571. Springer, 2014.

[13] Halevi, S., Shoup, V.: Bootstrapping for HElib. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 641670. Springer, 2015.

[14] 0. Hart, W., et al.: Fast Library for Number Theory (Version 2.5) (2015), http: //www.flintlib.org

[15] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012.

[16] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. 2013.

[17] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancr'ede Lepoint. Nfllib: Nttbased fast lattice library, 2016.

[18] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT, volume 6110 of Lecture Notes in Computer Science, pages 123, 2010.