Report:

Recent Results in Lattice-Based Cryptography

Behzad Abdolmaleki

Advisors: Michal Zajac and Prastudy Fauzi

University of Tartu

Abstract

In this report we describe a fully homomorphic encryption (FHE) scheme based on the learning with errors (LWE) problem. Firstly we introduce some definitions related to lattices and some operators which will be needed to build an efficient fully homomorphic encryption system based on LWE. The main focus in this report is to explain the new technique to building FHE schemes that is called the approximate eigenvector method. Also we show that how this approach can make the Gentry et. al.’s scheme as an efficient FHE scheme.

1 Introduction

Recently lattice-based cryptography is an interesting topic for research, because it is conjectured that they are secure against quantum attacks. Also they have been used quite successfully in constructing secure cryptographic protocols that achieve functionalities such as fully homomorphic encryption (FHE). Fully homomorphic encryption allows to perform computations over encrypted data without decrypting them. This concept has long been regarded as an open problem until the breakthrough paper of Gentry in 2009 [Gen09] which shows the feasibility of computing any function on encrypted data. Since then, many constructions have appeared involving new mathematical and algorithmic concepts and improving efficiency. The main problem in homomorphic encryption is that the messages that are encrypted with a noise will grow at each homomorphic evaluation of an elementary operation, so if this noise become larger than the message then we might get some problems to extract the message from the ciphertext (decryption phase). So to overcoming this problem several fully homomorphic encryption schemes proposed [Gen09], [BV11], [GSW13], [CGGI16]. In this report we explain the GSW scheme which is fully homomorphic encryption schemes and its advantage and disadvantage. Then to overcome GSW’s disadvantage we explain the Chillotti et al.’s approach [CGGI16].
2 Lattices

**Definition 1 (Lattices).** Given \( n \) linearly independent vectors \( b_1, b_2, ..., b_n \in \mathbb{R}^n \), the lattice \( L \) is defined as :

\[
L = \{a_1 b_1 + ... + a_n b_n \mid a_i \in \mathbb{Z}\},
\]

where the set of \((b_1, b_2, ..., b_n)\) is called a basis of the lattice. We will use a notational shorthand when dealing with bases and denoting them by a matrix \( B \) whose columns are the basis vectors \( b_1, b_2, ..., b_n \).

2.1 Learning With Errors

The learning with errors (LWE) problem was introduced by Regev [Reg09].

**Definition 2 (LWE).** For security parameter \( \lambda \), let \( n = n(\lambda) \) be an integer dimension, let \( q = q(\lambda) \geq 2 \) be an integer, and let \( \chi = \chi(\lambda) \) be a distribution over \( \mathbb{Z} \). The \( LWE_{n,q,\chi} \) problem is to distinguish the following two distributions: In the first distribution, one samples \((a_i, b_i)\) uniformly from \( \mathbb{Z}_q^{n+1} \). In the second distribution, one first draws \( s \leftarrow \mathbb{Z}_q^n \) uniformly and then samples \((a_i, b_i) \in \mathbb{Z}_q^{n+1}\) by sampling \( a_i \in \mathbb{Z}_q^n \) uniformly, \( e_i \in \chi \), and setting \( b_i = \langle a_i, s \rangle + e_i \). The \( LWE_{n,q,\chi} \) assumption is that the \( LWE_{n,q,\chi} \) problem is infeasible.

Sometimes it is convenient to view the vectors \( b_i \) and \( a_i \) as the rows of a matrix \( A \), and to redefine \( s \) as \((1, s)\). Then, either \( A \) is uniform, or there is a vector \( s \) whose first coefficient is 1 such that \( As = e \), where the coefficients of \( e \) come from the distribution \( \chi \).

2.2 GapSVP

**GapSVP:** For lattice dimension parameter \( n \) and number \( d \), GapSVP is the problem of distinguishing whether a \( n \)-dimensional lattice has a vector shorter \( (\lambda_1) \) than \( d \) or no vector shorter than \( \gamma(n)d \) [Reg09].

- Yes if \( \lambda_1 < d \)
- No if \( \lambda_1 \geq \gamma(n)d \).

2.3 B-bounded distributions

[GSW13] defines a distribution ensemble \( \{\chi^n\}_{n \in \mathbb{N}} \), supported over the integers, is called B-bounded if:

\[
Pr_{e \sim \chi_n}[||e|| > B] = \text{negl}(n).
\]

**Theorem 1:** Let \( q = q(n) \in \mathbb{N} \) be either a prime power or a product of small (size \( \text{poly}(n) \)) distinct primes, and let \( B \geq \omega(\log n) \cdot \sqrt{n} \). Then there exists an efficient sampleable B-bounded distribution \( \chi \) such that if there is an efficient algorithm that solves the average-case LWE problem for parameters \( n, q, \chi \), then:

1. There is an efficient quantum algorithm that solves \( \text{GapSVPO}(nq/B) \) on any \( n \)-dimensional lattice.
2. If \( q \geq O(2n/2) \), then there is an efficient classical algorithm for \( \text{GapSVP}_{O(nq/B)} \) on any \( n \)-dimensional lattice.

In both cases, if one also considers distinguishers with sub-polynomial advantage, then we require \( B \geq O(n) \) and the resulting approximation factor is slightly larger than \( O(n^{1.5}q/B) \) [GSW13].

3. Flattening

Flattening is an operation that can be performed on a vector or matrix (ciphertext) such that it makes the coefficients of a vector or matrix small, without affecting its product with \( \text{Powersof2}(b) \), and without knowing \( b \). Let \( a, b \) be vectors of some dimension \( k \) over \( \mathbb{Z}_q \). Let \( \ell = \lceil \log_2 q \rceil + 1 \) and \( N = k \cdot \ell \). The \( \text{BitDecomp}(a) \) can be defined as follows,

\[
\text{BitDecomp}(a) = (a_{1,0}, \cdots, a_{1,\ell-1}, a_{2,0}, \cdots, a_{2,\ell-1}, \cdots, a_{k,0}, \cdots, a_{k,\ell-1},)
\]

which \( \text{BitDecomp}(a) \) will be the \( N \)-dimensional vector, where \( a_{i,j} \) is the \( j \)-th bit in \( a_i \) binary representation, bits ordered least significant to most significant. Now we define \( a' = (a_{1,0}, \cdots, a_{1,\ell-1}, a_{2,0}, \cdots, a_{2,\ell-1}, \cdots, a_{k,0}, \cdots, a_{k,\ell-1},) \), then \( \text{BitDecomp}^{-1}(a') \) is as follows,

\[
\text{BitDecomp}^{-1}(a') = (\sum_{j=0}^{\ell-1} a_{1,j} \cdot 2^j, \sum_{j=0}^{\ell-1} a_{2,j} \cdot 2^j, \cdots, \sum_{j=0}^{\ell-1} a_{k,j} \cdot 2^j)
\]

actually it is the inverse of \( \text{BitDecomp} \), but well-defined even when the input is not a 0/1 vector.

Finally for \( N \)-dimensional \( a' \) the \( \text{Flatten}(a') \) can be defined as follows,

\[
\text{Flatten}(a') = \text{BitDecomp}(\text{BitDecomp}^{-1}(a'))
\]

which it will be a \( N \)-dimensional vector with 0/1 coefficients.

In general, for a matrix \( A \), the \( \text{BitDecomp}(A) \), \( \text{BitDecomp}^1 \), or \( \text{Flatten}(A) \) will be the matrix formed by applying the operation to each row of matrix \( A \) separately. Now let \( \text{Powersof2}(b) = (b_1, 2 \cdot b_1, \cdots, 2^{\ell-1} \cdot b_1, \cdots, b_k, 2 \cdot b_k, \cdots, 2^{\ell-1} \cdot b_k) \), a \( N \)-dimensional vector. Then we can see that for any vectors \( a, b \) of some dimension \( k \) over \( \mathbb{Z}_q \) and any \( N \)-dimensional \( a' \) where \( N = k \cdot \ell \), the following equations hold:

1. \( \langle \text{BitDecomp}(a), \text{Powersof2}(b) \rangle = \langle a, b \rangle. \)
2. \( \langle a', \text{Powersof2}(b) \rangle = \langle \text{BitDecomp}^{-1}(a'), b \rangle = \langle \text{Flatten}(a'), \text{Powersof2}(b) \rangle. \)

Here We will give a more detailed proof than in the [GSW13] for both equations 1 and 2 as follows,

\text{Proof for equation 1}: By the definition of the \( \text{BitDecomp}(a) \) and \( \text{Powersof2}(b) \), we can write:
\[ \langle \text{BitDecomp}(a), \text{Powersof2}(b) \rangle = \sum_{i=1}^{i=k} \sum_{j=0}^{j=\ell-1} a_{i,j} \cdot (2^j b_i) = \sum_{i=1}^{i=k} \sum_{j=0}^{j=\ell-1} (a_{i,j}2^j) \cdot b_i = \sum_{i=1}^{i=k} b_i \cdot (\sum_{j=0}^{j=\ell-1} a_{i,j}2^j) = \langle a, b \rangle. \]

It is also proved by another approach in [Fau12].

### 3.1 Eigenvector Method

This method is defined as follows,

\[ C \cdot v = \mu \cdot v \mod q \]

where \( C \) can be presented as a ciphertext and \( v \) is eigenvector (secret key) and \( \mu \) is eigenvalue (message). But this encryption scheme has some problems, for example one can encrypt message \( \mu = 0 \) then he will be able to obtain the secret key \( v \) by search in the null space of \( C \) as follows,

\[ C \cdot v = 0 \mod q. \]

In order to solve this problem, Gentry, Sahai and waters proposed a new encryption scheme that is called approximate eigenvector method [GSW13]. The approximate eigenvector method can be written as follows,

\[ C \cdot v = \mu \cdot v + e \mod q \]

where \( e \) is noise (with small coefficients).

### 4 GSW-FHE(Leveled) Scheme based on LWE

In this section firstly we explain the GSW encryption based on LWE [GSW13] and then we will show that how it implies fully homomorphic encryption. The GSW scheme has three parts \texttt{keyGen}, \texttt{Enc(.)}, and \texttt{Dec(.)}. Which the keyGen contains \texttt{Setup}, \texttt{SecretKeyGen} and \texttt{PublicKeyGen}, as follows,

1. **keyGen**: It has three parts as follows,
   
   (a) \texttt{Setup}(1^\lambda, 1^L): Choose a modulus \( q = \kappa(\lambda, L) \) bits, lattice dimension parameter \( n = n(\lambda, L) \), and error distribution \( \chi = \chi(\lambda, L) \) appropriately for LWE that achieves at least \( 2^\kappa \) security against known attacks. Also, choose parameter \( m = m(\lambda, L) = O(n \log q) \). Let \( \text{params} = (n, q, \chi, m) \). Suppose \( \ell = \lceil \log q \rceil + 1 \) and \( N = (n + 1) \cdot \ell. \)
(b) **SecretKeyGen**\((\text{params})\): Sample \(t \leftarrow Z_q^n\). Output \(sk = s \leftarrow (1, t_1, \ldots, t_n) \in Z_q^{n+1}\). Let \(v = \text{Powersof}2(s)\).

(c) **PublicKeyGen**\((\text{params}, sk)\): Generate a matrix \(B \leftarrow Z_q^{m \times n}\) uniformly and a vector \(e \leftarrow \chi^m\). Set \(b = B \cdot t + e\). Set \(A\) to be the \((n + 1)\)-column matrix consisting of \(b\) followed by the \(n\) columns of \(B\). Set the public key \(pk = A\) where we can see that \(A \cdot s = e\).

2. **Enc**\((\text{params}, pk, \mu)\): To encrypt a message \(\mu \in Z_q\), sample a uniform matrix \(R \in \{0, 1\}^{N \times m}\) and output the ciphertext \(C\) given below.

\[
C = \text{Flatten}(\mu \cdot I_N + \text{BitDecomp}(R \cdot A)) \in Z_q^{N \times N}.
\]

3. **Dec**\((\text{params}, sk, C)\): For decryption we need just one row of the matrix \(C\), so let \(C_i\) be the \(i\)-th row of the matrix \(C\). We can see that the first \(\ell\) coefficients of \(v\) are 1, 2, \ldots, \(2^{\ell-1}\). Among these coefficients, let \(v_i = 2^i\) be in \((q/4, q/2]\). Then for their decryption phase is as follows,

\[
\langle C_i, v \rangle = \langle \text{Flatten}(\mu \cdot I_N)_i + \text{BitDecomp}((R \cdot A)_i), v \rangle
\]

where \(v = \text{Powersof}2(s)\) and then by equation \(\langle a', \text{Powersof}2(b) \rangle = \langle \text{Flatten}(a'), \text{Powersof}2(b) \rangle\), Eq. (2) will be:

\[
\langle C_i, v \rangle = \langle (\mu \cdot I_N)_i + \text{BitDecomp}((R \cdot A)_i), v \rangle = \langle (\mu \cdot I_N)_i, v \rangle + \langle \text{BitDecomp}((R \cdot A)_i), v \rangle,
\]

also it is shown that \(\langle \text{BitDecomp}((R \cdot A)_i), \text{Powersof}2(s) \rangle = \langle (R \cdot A)_i, s \rangle\), so Eq. (3) can be written as follows,

\[
\langle C_i, v \rangle = \langle (R \cdot A)_i, s \rangle = \mu \cdot v_i + (R \cdot A)_i \cdot s,
\]

using the fact \((R \cdot A)_i = R_i \cdot A\), then we have,

\[
\mu \cdot v_i + (R \cdot A)_i \cdot s = \mu \cdot v_i + R_i \cdot A \cdot s,
\]

also by the property \(A \cdot s = e\), finally it will be,

\[
\mu \cdot v_i + R_i \cdot e.
\]

Now, for extracting message \(\mu\), Eq. (4) will divide by \(2^i\), as follows,

\[
\mu + \frac{R_i \cdot e}{2^i}.
\]
Where the error $R_i \cdot e$ has magnitude at most $||e||$ (because $R$ is a binary matrix), and to decrypt the message correctly norm of the error $e$ should be $||e|| < \frac{1}{2}$. So for new noise $\frac{R_i \cdot e}{2^i}$ the following equation should hold,

$$||\frac{R_i \cdot e}{2^i}|| < \frac{1}{2} \iff ||R_i \cdot e|| < 2^{i-1}.$$ 

In the other hand, $2^i \in (q/4, q/2]$, so $2^{i-1} \in \left(\frac{q}{8}, \frac{q}{4}\right]$. 

4.1 Security

We can see that $BitDecomp^{-1}(C) = \mu \cdot G + R \cdot A$, where $G = BitDecomp^{-1}(I_N)$ is (the transpose of) the primitive matrix used by Micciancio and Peikert [MP12] in their construction of lattice trapdoors, and the rows of $R \cdot A$ are simply Regev [Reg09] encryptions of 0 for dimension $n$. Assuming $BitDecomp^{-1}(C)$ hides message $\mu$, $C$ does as well, since $C$ can be derived by applying $BitDecomp$. So, the security of GSW’s basic encryption scheme follows directly from the following lemma,

**Lemma 1**: Let $params = (n, q, \chi, m)$ be such that the $LWE_{n,q,\chi}$ assumption holds. Then, for $m = O(n \log q)$ and $A, R$ as generated above, the joint distribution $(A, R \cdot A)$ is computationally indistinguishable from uniform over $Z_{q}^{m(n+1)} \times Z_{q}^{N(n+1)}$. (It is used to prove the security of Regev’s encryption scheme [Reg09]).

4.2 Homomorphic Property

To show homomorphic property of GSW scheme [GSW13], firstly we introduce additional operations such as $MultConst$, $Add$, $Mult$ as follows,

1. $MultConst(C, \alpha)$: To multiply a ciphertext $C \in Z_{q}^{N \times N}$ by known constant $\alpha \in Z_q$, set $M_\alpha \leftarrow Flatten(\alpha \cdot I_N)$ and output $Flatten(M_\alpha \cdot C)$. It can be seen that:

   $MultConst(C, \alpha) \cdot v = M_\alpha \cdot C \cdot v = M_\alpha \cdot (\mu \cdot v + e) = \mu \cdot (M_\alpha \cdot v) + M_\alpha \cdot e = \alpha \cdot \mu \cdot v + M_\alpha \cdot e.$

   Result, we can see the error $e$ multiples $M_\alpha$ and because $M_\alpha \leftarrow Flatten(\alpha \cdot I_N)$ so the error increases by a factor of at most $N$, regardless of what element $\alpha \in Z_q$ is used for multiplication.

2. $Add(C_1, C_2)$: To add ciphertexts $C_1, C_2 \in Z_{q}^{N \times N}$, output $Flatten(C_1 + C_2)$. The correctness of this operation is immediate. Note that the addition of messages is over the full base ring $Z_q$. 

6
3. \textit{Mult}(C_1, C_2): To multiply ciphertexts \(C_1, C_2 \in \mathbb{Z}_{q}^{N\times N}\), output \textit{Flatten}(C_1 \cdot C_2). So we can see that:

\[
\text{Mult}(C_1, C_2) \cdot v = C_1 \cdot C_2 \cdot v = C_1 \cdot (\mu_2 \cdot v + e_2) + \mu_2(\mu_1v + e_1) + C_1 \cdot e_2
\]

\[= \mu_1 \cdot \mu_2 \cdot v + \mu_2 \cdot e_1 + C_1 \cdot e_2.\]

As we can see, here the new noise is \(\mu_2 \cdot e_1 + C_1 \cdot e_2\) which it depends on the old errors \((e_1)\), the ciphertext \(C_1\), and the message \(\mu_2\). We can see that:

(a) The dependence on the \(e_1\) seems be unavoidable which it is normal for \textit{LWE}-based homomorphic encryption schemes.

(b) Also about the dependency of \(C_1\), since all components of \(C_1\) are restricted to \(\{0, 1\}\) (by flattening operation), we can see that \(C_1\) contributes at most a factor \(N\) blow up of error.

(c) About the error which growth based on the message \(\mu_2\), however, it still presents a concern, which in general, we must address this concern by using homomorphic operations (such as NAND operations) in a way that restricts the message space to small messages.

Notice that the multiplication operator also is over the full base field \(\mathbb{Z}_q\).

4.3 Advantage and Disadvantage of the GSW scheme

The advantage of the GSW scheme is that they proposed a new technique for building FHE schemes that called the \textit{approximate eigenvector} method which for the most part, homomorphic addition and multiplication are just matrix addition and multiplication. This makes their scheme both asymptotically faster.

About its disadvantage, we saw that in the decryption phase it just needed to one row of the matrix ciphertext \(C \in \mathbb{Z}_q^{N\times N}\) but for make it FHE, the ciphertext \(C\) should be a matrix with size \(N \times N\), so multiplying two ciphertexts has much complexity.

5 CGGI-FHE Scheme

In this section we just explain some intuitions of the Chillotti et al.’s scheme [CGGI16]. Let denote CGGI as Chillotti et al.’s scheme. Firstly, we describe some preliminaries as follows,
5.1 Torus

Set of the real numbers modulo 1 is defined as a torus as follows,

\[ T_N[X] = \frac{\mathbb{R}[X]}{X^N + 1} \mod 1. \]

5.2 TLWE samples

Let \( k \geq 1 \) be an integer, \( N \) a power of 2, and \( \alpha \geq 0 \) be a noise parameter. A TLWE secret key \( s \in \mathbb{B}_N[X]^k \) is a vector with binary coefficients. The message space of TLWE samples is \( T_N[X] \). A fresh TLWE sample of a message \( \mu \in T_N[X] \) with noise parameter under the key \( s \) is an element \( (a,b) \in T_N[X]^k \times T_N[X] \) [CGGI16].

5.3 Intuition

The GSW scheme use internal product in the multiplication [GSW13]. In internal product the two ciphertexts are TGSW samples (GSW sample from torus), where are chosen from \( T_N[X]^{(k+1)\ell \times (k+1)} \). It outputs a TGSW samples.

But the CGGI scheme use the external product [CGGI16]. In the external product the first ciphertext is a TGSW sample, where is chosen from \( T_N[X]^{(k+1)\ell \times (k+1)} \) and the second one is a TLWE sample where is chosen from \( T_N[X]^k \). The external product outputs a TLWE sample which is a vector in torus with size \( k \). So the CGGI scheme is much faster than The GSW scheme to evaluate (speeds up of a factor at least \((k+1)\ell \)).

6 Conclusions

In this report, we first recall some lattices definitions and its hard problems. Then we explained the GSW encryption and detailed main intuition and efficiency that how we can get fully homomorphic encryption base on learning with error (LWE). After that we introduced the homomorphic property of GSW scheme. Finally we explain some intuitions of the CGGI scheme.

References


[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the


