CS6115: Structure versus Hardness in Cryptography Aug 11, 2021

Lecture 3 : Fully Homomorphic Encryption

Lecturer: *Shweta Agrawal* Scribe: *Simran Kumari*

1 Introduction

In this lecture we will look into the concept of Fully Homomorphic Encryption (FHE). Before defining it formally, let us consider the following motivating example where a client, say Alice, wants to compute some function f on her data D and the computation $f(D)$ requires lots of computational resource that she lacks. An ideal thought could be to let the cloud do the computation on her behalf, which would require Alice to send D to the server, but then she also doesn't want the cloud to learn about her data.

A trivial solution that one could think of is to encrypt the data and then send it to the cloud. Note that if we use a public-key encryption scheme for this task, we can be successful in hiding our data initially but the cloud won't be able to do any meaningful computation of the data without the secret key used for encrypting the data and thus we fail to reach our goal of computation that manages to hides the data. So is there a way you can let the cloud evaluate the function on your data and still manage to hide your data from the cloud?

Client **Cloud**

$$\xrightarrow{\hspace{5em}\mathsf{Enc}(D)\hspace{5em}}$$

$$\xrightarrow{\hspace{6em}f\hspace{6em}}$$

$$\xleftarrow{\hspace{5em}\mathsf{Enc}(f(D))\hspace{5em}}$$

YES! Homomorphic Encryption is the key to this question. Fully Homomorphic Encryption allows you to do computation on encrypted data for any function. So now, you can encrypt your data, send it to the cloud, get back the function output on the encrypted data and then use your secret key to reveal the function value on your data.

In section 2 we will look at some preliminary definitions that will help us in defining and constructing a fully homomorphic encryption scheme.

2 Preliminaries

Definition 2.1 (Rings)**.** A ring is a set $\mathbb{R}$ equipped with two binary operations $+$ (addition) and $\cdot$ (multiplication) satisfying the following :

1. $\mathbb{R}$ is an abelian group under addition.

2. $\mathbb{R}$ is associative under multiplication and 1 is the multiplicative identity of $\mathbb{R}$.

3. Multiplication is distributive with respect to addition in $\mathbb{R}$

In this class we will either use the ring of integers $\mathbb{R} = \mathbb{Z}$ or polynomial rings $\mathbb{Z}[x]/(x^d + 1)$ where d is a power of 2. For $r \in \mathbb{R}$, $||r||$ refers to the Euclidean norm of r.

Definition 2.2 (Dot Product). The dot product of two n-dimensional vectors $u, v \in \mathbb{R}^n$ is defined as

$$\langle u, v\rangle = \sum_{i=1}^{n} u[i]v[i]$$

where the notation $u[i]$ refers to the i-th coefficient of u.

Definition 2.3 (Negligible Function). A function f is negligible if $\forall$ polynomial $p(\cdot)$, $\exists N$ such that $\forall n > N$

$$f(n) < \frac{1}{p(n)}$$

Definition 2.4 (Learning With Errors (LWE)). For security parameter λ, 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:

1. Sample $(\vec{a_i}, b_i)$ from $\mathbb{Z}_q^{n+1}$
2. Sample $\vec{s} \leftarrow \mathbb{Z}_q^n, \vec{a_i} \leftarrow \mathbb{Z}_q^n$
   Compute $b_i = \langle \vec{a_i}, \vec{s}\rangle + e_i$ where $e_i \leftarrow \chi$

The LWE$_{n,q,\chi}$ assumption is that the LWE$_{n,q,\chi}$ problem is infeasible.

Definition 2.5 (B-bounded distribution). A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers, is called B-bounded if the

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

Theorem 1 (Hardness of LWE). For any integer dimension n, prime integer $q = q(n)$, and $B = B(n) \geq 2n$, there is an efficiently samplable B-bounded distribution χ such that if there exists an efficient (possibly quantum) algorithm that solves LWE$_{n,q,\chi}$, then there is an efficient quantum algorithm for solving $\tilde{O}(qn^{1.5}/B)$- approximate worst-case SIVP and gapSVP.

Definition 2.6 (Leftover Hash Lemma (LHL)). Given $A \leftarrow \mathbb{Z}_q^{N\times(n+1)}$, $N \geq 2(n+1)\log q$ and $r \leftarrow \{0,1\}^N$ then $\{A, A^T r\} \approx \{A, \text{uniform}\}$.

3 Homomorphic Encryption

A homomorphic encryption scheme $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is defined by the following ppt algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: Outputs (pk, evk, sk), the public key pk for encrypting, the evaluation key evk for evaluating and the secret key sk for decrypting.
- $\mathsf{Enc}(pk, m)$: Takes as input the public key pk and a message m and outputs the ciphertext c.
- $\mathsf{Eval}(evk, f, c_1, ..., c_\ell)$: Takes as input the evaluation key evk, the function f and a tuple of ciphertexts and outputs c_f, the evaluation ciphertext.
- $\mathsf{Dec}(sk, c)$: It uses the decryption key sk to decrypt a ciphertext c to recover the plaintext m'

Correctness : We say that the above homomorphic scheme is correct if the decryption of the homomorphic evaluation is correct with overwhelming probability. More formally,

$$\Pr[\mathsf{Dec}_{sk}(\mathsf{Eval}_{evk}(f, c_1, ..., c_\ell)) \neq f(m_1, \ldots, m_\ell)] = negl(\lambda)$$

where $c_i = \mathsf{Enc}_{pk}(m_i)$ and $(pk, evk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

Compactness : We say that the above homomorphic scheme is compact if there is a polynomial p, such that for any key-triplet $(pk, evk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, the function f, and the ciphertexts c_i, the size of the output $\mathsf{Eval}_{evk}(f, c_1, ..., c_\ell)$ is not more than $p(\lambda)$ bits, independent of the size of the function or the number of inputs to the function. .

Levelled Homomorphic Scheme : A homomorphic encryption scheme $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is called "levelled" if the KeyGen algorithm takes L as an additional input where L is the maximum depth of the circuits that can be evaluated by the scheme.

In the next section we will look at the building blocks of the fully homomorphic encryption scheme. We will see a public-key scheme used in [Bra12] that naturally is somewhat homomorphic with respect to addition. Then we will use the techniques from [BGV] to make this scheme fully homomorphic.

4 Regev's Ecryption Scheme

Let $q = q(n)$ be an integer function, $\chi = \chi(n)$ be a B-bounded distribution ensemble over $\mathbb{Z}$ and $N \triangleq (n+1) \cdot (\log q + O(1))$. The scheme Regev is defined as follows:

- $\mathsf{Regev.SecretKeyGen}(1^n)$: Sample $\vec{s} \leftarrow \mathbb{Z}_q^n$. Outputs $sk = \vec{s}$.
- $\mathsf{Regev.PublicKeyGen}(sk)$: Let $N = (n+1)\log q$.
  - Sample $A \leftarrow \mathbb{Z}_q^{N \times n}$, $\vec{e} \leftarrow \chi^N$.
  - Compute $b = [A\vec{s} + \vec{e}]$, $P = [b|| - A] \in \mathbb{Z}_q^{N \times (n+1)}$.
  - Output $pk = P$
- $\mathsf{Regev.Enc}_{pk}(m)$ for $m \in \{0, 1\}$:
  - Sample $\vec{r} \leftarrow \{0, 1\}^N$
  - Compute $\vec{c} = \left[P^T\vec{r} + \left\lfloor \frac{q}{2} \right\rfloor \vec{m}\right]_q \in \mathbb{Z}_q^{(n+1)}$ where $\vec{m} = (m, \ldots, 0)$
- $\mathsf{Regev.Dec}_{sk}(\vec{c})$:
  - Compute $[\langle \vec{c}, (1, \vec{s}) \rangle]_q \triangleq d$
  - Output $\left[\left\lfloor 2 \cdot \frac{d}{q} \right\rceil\right]_2$

4.1 Correctness

We shall prove that the algorithm Regev.Dec will decrypt correctly.

$$\begin{aligned}
\text{We have } \vec{c} &= (P^T\vec{r} + \left\lfloor \frac{q}{2} \right\rfloor \vec{m}) \mod q \\
\text{So, } \langle \vec{c}, (1, \vec{s}) \rangle &= \langle P^T\vec{r} + \left\lfloor \frac{q}{2} \right\rfloor \vec{m}, (1, \vec{s}) \rangle \mod q \\
&= \langle P^T\vec{r}, (1, \vec{s}) \rangle + \left\lfloor \frac{q}{2} \right\rfloor \langle (m, \ldots, 0), (1, \vec{s}) \rangle \mod q \\
&= \vec{r}^T P \cdot (1, \vec{s}) + \left\lfloor \frac{q}{2} \right\rfloor m \mod q \\
&= \vec{r}^T ([b|| - A]) \cdot (1, \vec{s}) + \left\lfloor \frac{q}{2} \right\rfloor m \mod q \\
&= \vec{r}^T (b - A\vec{s}) + \left\lfloor \frac{q}{2} \right\rfloor m \mod q \\
&= \langle \vec{r}, \vec{e} \rangle + \left\lfloor \frac{q}{2} \right\rfloor m \mod q
\end{aligned}$$

Here $\vec{r} \leftarrow \{0, 1\}^N$ in Regev.Enc and $e \leftarrow \chi^N$ where χ is B-bounded. So we have,

$$|\langle \vec{r}, e \rangle| = \left| \sum_{i=1}^{N} r_i e_i \right| \leq \sum_{i=1}^{N} r_i |e_i| \leq \sum_{i=1}^{N} B = N \cdot B$$

Let $e = \langle \vec{r}, e \rangle$. So we have $\langle \vec{c}, (1, \vec{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor m + e$.

Lemma 1. Let $\vec{s} \in \mathbb{Z}^n$ be some vector, and let $\vec{c} \in \mathbb{Z}_q^{(n+1)}$ be such that $\langle \vec{c}, (1, \vec{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor m + e$ with $m \in \{0, 1\}$ and $|e| < \left\lfloor \frac{q}{2} \right\rfloor /2$. Then $\text{Regev.Dec}_{sk}(c) = m$

Proof. We have $\langle \vec{c}, (1, \vec{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor m + e$, so

$$\left[\left\lfloor 2 \cdot \frac{d}{q} \right\rceil \right]_2 = \left[\left\lfloor 2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \right\rceil \right]_2$$

For $m = 0$, $\left\lfloor \frac{q}{2} \right\rfloor m + e \in (-(q-1)/4, q/4)$. So the output of the decryption algorithm,

$$\left\lfloor 2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \right\rceil \in \left(-\frac{1}{2} + \frac{1}{2q}, \frac{1}{2} \right)$$

which is closer to 0.

When $m = 1$, $\left\lfloor \frac{q}{2} \right\rfloor m + e \in ((q-1)/4, 3q/4)$. So the output of the decryption algorithm,

$$\left\lfloor 2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \right\rceil \in \left(\frac{1}{2} - \frac{1}{2q}, \frac{3}{2} \right)$$

which is closer to 1.

4.2 Security

Let us first look at the security game for any public-key encryption scheme

$$\begin{array}{ccc} \textbf{Challenger} & & \textbf{Adversary}(\mathcal{A}) \\ \text{Generates } (pk, sk) & \xrightarrow{\quad\quad\quad pk \quad\quad\quad} & \\ & \xleftarrow{\quad\quad\quad m_0, m_1 \quad\quad\quad} & \\ \text{Sample } b \leftarrow \{0,1\} & \xrightarrow{\quad\quad Enc(pk, m_b) \quad\quad} & \\ & \xleftarrow{\quad\quad\quad b' \quad\quad\quad} & \text{Guess } b' \end{array}$$

The adversary $\mathcal{A}$ wins the game if $b = b'$. The scheme is said to be semantically secure if Pr[$\mathcal{A}$ wins] $\leq$ $1/2 + negl(\lambda)$.
Now consider any adversary $\mathcal{A}$ for the above Regev's scheme. The adversary can see the public key $P = [b|| - A]$ and the ciphertext $\vec{c} = \left[P^T\vec{r} + \left\lfloor \frac{q}{2} \right\rfloor \vec{m}\right]_q$. We can argue the security of the Regev's scheme in two steps using LWE assumption and leftover hash lemma as follows:

- First we use the LWE assumption to replace $b = [A\vec{s} + \vec{e}]$ in the public key P with a uniformly random vector $b' \leftarrow \mathbb{Z}_q^{N\times(n+1)}$.
- Now since A is chosen uniformly at random from $\mathbb{Z}_q^{N\times n}$ in $\mathsf{Regev.PublicKeyGen}$ and b' is uniformly random, so our public key $P = [b'|| - A]$ is uniformly random. Also we have $N \geq 2(n+1)\log q$ and $\vec{r} \leftarrow \{0,1\}^N$, so by LHL, $P^T\vec{r}$ is statistically indistinguishable from some $P' \leftarrow \mathbb{Z}_q^{(n+1)}$. So the encryption algorithm computing $\vec{c} = \left[P' + \left\lfloor \frac{q}{2} \right\rfloor \vec{m}\right]_q$ acts as a one time pad for our message bit. So our message is going to be information theoretically hidden.

So any adversary $\mathcal{A}$ for the above scheme cannot guess the bit b' correctly with probability better than $1/2 + negl(n)$.

References

[BGV] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.