CS6115: Structure versus Hardness in Cryptography	Aug 19, 2021
Lecture 6 : Modulus Reduction and Bootstrapping	
Lecturer: *Shweta Agrawal*	Scribe: *Anuja Modi*

1 Introduction

In the previous lecture, we constructed somewhat homomorphic symmetric key encryption scheme (SWHE) based on LWE. This somewhat homomorphic symmetric key encryption scheme uses the crucial idea of *Dimension Reduction* to reduce the size of quadratically growing cipher text to linear. Basically, dimension reduction technique allowed to support $\epsilon \log n$ depth instead of $\log \log n$ depth.

In this lecture, we will first see Homomorphic Encryption with dimension reduction. Then we will study about two important techniques. First is *Modulus Reduction* and second is *Bootstrapping*. Modulus Reduction technique helps us to control the noise growth and thereby allow to support $O(n^\epsilon)$ depth. Bootstrapping transforms the Homomorphic Encryption scheme to Fully Homomorphic Encryption scheme that can support arbitrary polynomial depth circuits.

2 Homomorphic Encryption with dimension reduction

. In this section, we present the algorithms of Homomorphic Encryption with dimension reduction [BGV12].

Notations. Let $\lceil \frac{q}{2} \rfloor = \frac{(q+1)}{2}$. $\log q = \lceil \log q \rceil$. Ring, $\mathbb{Z}_q = \left\{ -\frac{q-1}{2}, \ldots, 0, \ldots, \frac{q-1}{2} \right\}$. $[n] = \{0, 1, \ldots, n-1\}$.

- $(sk, evk) \leftarrow Gen(1^\lambda, 1^L)$
  1. For $l \in [L+1], choose\, \overrightarrow{t_l} \leftarrow \mathbb{Z}_q^n$ and $\overrightarrow{s_l} = (-\overrightarrow{t_l}, 1)$.
  2. For $l \in [L], i, j \in [n+1], \tau, v \in [\log q]$,
  $$\overrightarrow{\psi_{l,i,j,\tau,v}} = (\overrightarrow{a_{l,i,j,\tau,v}}, \langle \overrightarrow{a_{l,i,j,\tau,v}}, \overrightarrow{t_{l+1}} \rangle + e_{l,i,j,\tau,v} + 2^v s_{l,i,j,\tau,v})$$
  where $\overrightarrow{a_{l,i,j,\tau,v}} \leftarrow \mathbb{Z}_q^n$, $e_{l,i,j,\tau,v} \leftarrow \chi$.
  3. Let $(\overrightarrow{c^*}, 0) \leftarrow Enc(sk, 1)$, $sk = (\overrightarrow{s_0}, \ldots, \overrightarrow{s_L})$ and $evk = (\{\overrightarrow{\psi_{l,i,j,\tau,v}}\}_{l,i,j,\tau,v}, \overrightarrow{c^*})$. Output (sk, evk).
  (Note: $(\overrightarrow{c^*}, 0) \leftarrow Enc(sk, 1)$ will be used later to raise level.)
- $(\overrightarrow{c}, l) \leftarrow Enc(sk, m \in \{0, 1\})$
  1. Choose $\overrightarrow{a} \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi$. Let $\overrightarrow{c} = \left(\overrightarrow{a}, \langle \overrightarrow{a}, \overrightarrow{t_0} \rangle + e + m\frac{(q+1)}{2} \right)$. Output $(\overrightarrow{c}, 0)$.
- $m \leftarrow Dec(sk, (\overrightarrow{c}, l))$
  1. Compute $m' = \langle \overrightarrow{c}, \overrightarrow{s_l} \rangle$. Output 0 if $\frac{-q}{4} \leq m' \leq \frac{q}{4}$, otherwise output 1.

- $(\overrightarrow{c_{add}}, l_{add}) \leftarrow Add(evk, (\overrightarrow{c_1}, l_1), (\overrightarrow{c_2}, l_2))$

  1. If $l_1 \neq l_2$ then with help of $Mult$, raise lower level with the help of $\overrightarrow{c^*}$ until $l = l_1 = l_2$.
  2. Output $\overrightarrow{c_{add}} = \overrightarrow{c_1} + \overrightarrow{c_2}$ and $l_{add} = l$.

- $(\overrightarrow{c}_{mult}, l_{mult}) \leftarrow Mult(evk, (\overrightarrow{c}_1, l_1), (\overrightarrow{c_2}, l_2))$

  1. If $l_1 \neq l_2$ then with help of $Mult$, raise lower level with the help of $\overrightarrow{c^*}$ until $l = l_1 = l_2$.
  2. For $i, j \in [n+1], \tau \in [\log q]$, let $c_{i,j,\tau}$ be the τth bit of $2c_{1,i}c_{2,j}$.
  3. $\overrightarrow{c_{mult}} = \sum_{i,j,\tau} c_{i,j,\tau}\overrightarrow{\psi_{l,i,j,\tau}}$ and $l_{mult} = l + 1$. Output $(\overrightarrow{c}_{mult}, l_{mult})$.

Correctness of addition. Let us see that the addition algorithm above is correct.

$$\begin{aligned}
\langle \overrightarrow{c_{add}}, \overrightarrow{s_{l_{add}}} \rangle &= \langle \overrightarrow{c_1} + \overrightarrow{c_2}, \overrightarrow{s_l} \rangle \\
&= \langle \overrightarrow{c_1}, \overrightarrow{s_l} \rangle + \langle \overrightarrow{c_2}, \overrightarrow{s_l} \rangle \\
&= e_1 + e_2 + (m_1 + m_2 mod 2)\frac{(q+1)}{2} \\
&= e_{add} + (m_1 + m_2 mod 2)\frac{(q+1)}{2}
\end{aligned} \tag{1}$$

Correctness of multiplication. Let us see that the multiplication algorithm above is correct.

$$\begin{aligned}
\langle \overrightarrow{c_{mult}}, \overrightarrow{s_{l_{mult}}} \rangle &= \langle \sum_{i,j,\tau} c_{i,j,\tau}\overrightarrow{\psi_{l,i,j,\tau}}, \overrightarrow{s_{l+1}} \rangle \\
&= \sum_{i,j,\tau} c_{i,j,\tau} \langle \overrightarrow{\psi_{i,j,\tau}}, \overrightarrow{s_{l+1}} \rangle \\
&= \sum_{i,j,\tau} c_{i,j,\tau} \left(e_{l,i,j,\tau} + 2^{\tau} s_{l,i} s_{l,j}\right) \\
&= e_{dr} + \sum_{i,j} 2c_{1,i}c_{2,j}s_{l,i}s_{l,j} \\
&= e_{dr} + \langle 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{s_l} \otimes \overrightarrow{s_l} \rangle \\
&= e_{dr} + 2(e_1e_2 + m_1e_2 + m_2e_1) + m_1m_2\frac{(q+1)}{2} \\
&= e_{mult} + m_1m_2\frac{(q+1)}{2}
\end{aligned} \tag{2}$$

where e_{add}, e_{mult} are the error corresponding to addition and multiplication respectively. It can be seen that above errors satisfies the bounds. Note that e_{mult} is addition of error due to dimension reduction, e_{dr} and error resulted from decryption $2(e_1e_2 + m_1e_2 + m_2e_1)$.

Let χ be a e_{init}-bounded distribution. Therefore, at level L, the error is bounded by $e_{init}^{2^L}$. Hence, for correctness, we need that $e_{init}^{2^L} < \frac{q}{4}$ and for security best LWE algorithm running time is $2^{n/(log(q/e_{init}))}$. Hence e_{init} is chosen to be in polynomial $n = \lambda$ and $q = 2^{n^{\epsilon}}$ and hence, $L \approx \log\log q \approx \epsilon \log n$.

Hence, using this dimension reduction technique we can support $\epsilon \log n$ depth circuit.

3 Modulus Reduction

In this section, we first see important technique called "Modulus Reduction" which support the evaluation of circuit with depth $O(n^{\epsilon})$ where $\epsilon \in (0,1)$.

Let us first see the following claim 1.

Claim 1 ([ACPS09]). LWE with secret $\overrightarrow{e} \leftarrow \chi^n$ is as hard as LWE with secret $\overrightarrow{e} \leftarrow \mathbb{Z}^n$.

Proof. When LWE secret is drawn from error distribution i.e. $\overrightarrow{e} \leftarrow \chi^n$, it is called Hermite Normal Form of LWE (hLWE).

To prove this claim, we show that if we can solve LWE, then we can solve hLWE and vice versa.

- **Case 1**: To prove. Given an oracle that solves LWE, we can find a solution to an instance of hLWE.

  Proof. Let $A, b = A^T s + e$ be an instance of hLWE such that $s \leftarrow \mathcal{X}^n$, $e \leftarrow \mathcal{X}^m$ and $A \in Z_q^{n\times m}$.

  Sample $s' \leftarrow Z_q^n$ and $e' \leftarrow \mathcal{X}^m$.

  Let $b' = A^T s' + e'$

  After subtracting the two equations,

  $b - b' = A^T(s - s') + e - e'$

  OR $b'' = A^T s'' + e''$

  This (A^T, b'') is an instance of LWE. Call LWE oracle to solve for s'', e''. Since we know s', e', we can recover s, e.

  Hence, given an oracle that solves LWE, we can find a solution to an instance of hLWE. □

- **Case 2**: To prove. Given an oracle that solves hLWE, we can find a solution to an instance of LWE.

  Proof. Let $A, b = A^T t + e$ be an instance of LWE, i.e. $t \leftarrow Z_q^n$, $e \leftarrow \mathcal{X}^m$ and $A \in Z_q^{n\times m}$.

  Express $A^T = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ where $A_1 \in Z_q^{nxn}$ and $e = \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$.

  Let $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} t + \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$.

  Note that a random square matrix is invertible with high probability.

  $\implies b_1 = A_1 t + e_1$

  Solving it for t we get $t = A_1^{-1}(b_1 - e_1)$

  Putting this t in $b_2 = A_2 t + e_2$

  $\implies b_2 = A_2 A_1^{-1} b_1 - A_2 A_1^{-1} e_1 + e_2$

  $\implies b_2 - A_2 A_1^{-1} b_1 = -A_2 A_1^{-1} e_1 + e_2$

  $\implies b_3 = B e_1 + e_2$

  Call hLWE oracle to get e_1 and e_2. We know b_1, e_1 and A_1 and hence we can get t.

  Hence, given an oracle that solves hLWE, we can find a solution to an instance of LWE. □

□

This claim helps us to choose the secret from a low norm distribution without affecting the hardness or security of the underlying LWE.

Let χ be a B-bounded distribution and let $q \approx B^{10}$. In the original scheme without using modulus reduction, the error gets squared after first level when we multiply two ciphertexts. Hence, only $\lfloor \log 10 \rfloor$ levels can be supported and after that we get decryption error. When we use modulus reduction technique, both noise and modulus can be divided with B and thus resulting ciphertext would be in integer ring $\mathbb{Z}_{B^9}$ after level 1 as mentioned in Table 1. We can follow the same procedure in subsequent levels and can support 10 levels.

Table 1: Comparison of noise growth without and with modulus reduction

Noise/Modulus	Without modulus reduction	With modulus reduction
Take fresh ciphertext	B/B^{10}	B/B^{10}
Level 1	B/B^{10}	$B^2/B^{10} \rightarrow B/B^9$
Level 2	B^2/B^{10}	$B^2/B^9 \rightarrow B/B^8$
Level 3	B^4/B^{10}	$B^2/B^8 \rightarrow B/B^7$
Level 4	B^8/B^{10} (Decryption error!)	$B^2/B^7 \rightarrow B/B^6$

Let us roughly present the idea of the modulus reduction technique of scale invariant version [Bra12].

By induction,

$$\langle \overrightarrow{c_1}, \overrightarrow{s_l} \rangle = z_1 q + e_1 + m_1 \frac{(q+1)}{2} \tag{3}$$

and

$$\langle \overrightarrow{c_2}, \overrightarrow{s_l} \rangle = z_2 q + e_2 + m_2 \frac{(q+1)}{2} \tag{4}$$

where $z_1, z_2 \in [-(n+1)q, (n+1)q]$.

After tensoring and inner product of equation (3) and (4),

$$\left\langle \frac{2}{q} \overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{s_l} \otimes \overrightarrow{s_l} \right\rangle = z_{mr} q + e_{mr} + m_1 m_2 \frac{(q+1)}{2} \tag{5}$$

where $z_{mr} = 2z_1z_2 + z_1m_2 + z_2m_1$, $e_{mr} = 2z_1e_2 + 2z_2e_1 + z_1m_2 + z_2m_1 + e_1m_2 + e_2m_1 + m_1m_2/2 + (2e_1e_2 + e_1m_2 + e_2m_1 + m_1m_2/2)/q$.

Note that this equation (5) is in the ciphertext form. However, we need to prove that e_{mr} is much less than q. The term $2z_1e_2 + 2z_2e_1$, is the dominating term in e_{mr} and it is in the interval $[-4(n+1)qB, 4(n+1)qB]$. Now if we use Claim 1, then the interval above becomes $[-4(n+1)B^2, 4(n+1)B^2]$. At this moment we can apply the Bitdecomp technique to the secret key.

The construction of the homomorphic scheme consists of algorithms same as the algorithms in Section 2. With modulus reduction in frame, only Gen and $Mult$ algorithm changes. Below are the modified Gen and $Mult$ algorithm. All the operations given below are in $\mathbb{Z}_q$ ring except the ones to compute $d'_{i,j,\tau}$.

- $(sk, evk) \leftarrow Gen(1^\lambda, 1^L)$
  1. For $l \in [L+1], choose\, \overrightarrow{t_l} \leftarrow \mathbb{Z}_q^n$ and $\overrightarrow{s_l} = (-\overrightarrow{t_l}, 1)$.

2. For $l \in [L], i,j \in [n+1], \tau \in [\log q]$, $s_{l,i,j,\tau}$ be the τth bit of $s_{l,i}s_{l,j}$.
3. For $l \in [L], i,j \in [n+1], \tau, v \in [\log q]$,

$$\overrightarrow{\psi_{l,i,j,\tau,v}} = (\overrightarrow{a_{l,i,j,\tau,v}}, \langle \overrightarrow{a_{l,i,j,\tau,v}}, \overrightarrow{t_{l+1}} \rangle + e_{l,i,j,\tau,v} + 2^v s_{l,i,j,\tau,v})$$

   where $\overrightarrow{a_{l,i,j,\tau,v}} \leftarrow \mathbb{Z}_q^n$, $e_{l,i,j,\tau,v} \leftarrow \chi$.
4. Let $(\overrightarrow{c^*}, 0) \leftarrow Enc(sk, 1)$, $sk = (\overrightarrow{s_0}, \ldots, \overrightarrow{s_L})$ and $evk = (\{\overrightarrow{\psi_{l,i,j,\tau,v}}\}_{l,i,j,\tau,v}, \overrightarrow{c^*})$. Output (sk, evk).

- $(\overrightarrow{c}_{mult}, l_{mult}) \leftarrow Mult(evk, (\overrightarrow{c}_1, l_1), (\overrightarrow{c_2}, l_2))$
  1. If $l_1 \neq l_2$ then with help of $Mult$, raise lower level with the help of $\overrightarrow{c^*}$ until $l = l_1 = l_2$.
  2. For $i,j \in [n+1], \tau \in [logq]$, let $d'_{i,j,\tau} = 2^\tau \frac{2}{q} c_{1,i}c_{2,j}$ and $d_{i,j,\tau} = \lceil d'_{i,j,\tau} \rfloor$.
  3. For $i,j \in [n+1], \tau, v \in [logq]$, $c_{i,j,\tau,v}$ be vth bit of $d_{i,j,\tau}$.
  4. $\overrightarrow{c_{mult}} = \sum_{i,j,\tau,v} c_{i,j,\tau,v}\overrightarrow{\psi_{l,i,j,\tau,v}}$ and $l_{mult} = l+1$. Output $(\overrightarrow{c}_{mult}, l_{mult})$.

Correctness of multiplication. Let us see that the multiplication algorithm above is correct.

$$\begin{aligned}
\langle \overrightarrow{c_{mult}}, \overrightarrow{s_{l_{mult}}} \rangle &= \left\langle \sum_{i,j,\tau,v} c_{i,j,\tau,v} \overrightarrow{\psi_{i,j,\tau,v}}, \overrightarrow{s_{l+1}} \right\rangle \\
&= \sum_{i,j,\tau,v} c_{i,j,\tau,v} \left\langle \overrightarrow{\psi_{i,j,\tau,v}}, \overrightarrow{s_{l+1}} \right\rangle \\
&= \sum_{i,j,\tau,v} c_{i,j,\tau,v} \left(e_{l,i,j,\tau,v} + 2^v s_{l,i,j,\tau} \right) \\
&= e_{dr} + \sum_{i,j,\tau} d_{i,j,\tau} s_{l,i,j,\tau} \\
&= e_{dr} + e_{round} + \sum_{i,j,\tau} d'_{i,j,\tau} s_{l,i,j,\tau} \\
&= e_{dr} + e_{round} + \sum_{i,j} \frac{2}{q} c_{1,i}c_{2,j} s_{l,i} s_{l,j} \\
&= e_{dr} + e_{round} + \left\langle \frac{2}{q} \overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{s_l} \otimes \overrightarrow{s_l} \right\rangle \\
&= e_{dr} + e_{round} + e_{mr} + m_1 m_2 \frac{(q+1)}{2}
\end{aligned} \tag{6}$$

where $e_{dr}, e_{round}, e_{mr}$ are the error corresponding to dimension reduction, rounding and modulus reduction respectively. It can be seen that above errors satisfies the bounds.

Let χ be a e_{init}-bounded distribution. Therefore, at level L, the error is bounded by $poly(n)^L e_{init}$. Hence, for correctness, we need that $poly(n)^L e_{init} < \frac{q}{4}$ and for security best LWE algorithm running time is $2^{n/(log(q/e_{init}))}$. Hence e_{init} is chosen to be in polynomial $n = \lambda$ and $q = 2^{n^\epsilon}$ and hence, $L \approx logq \approx n^\epsilon$.

Hence, using this modulus reduction technique we can support $O(n^\epsilon)$ depth circuit.

4 Bootstrapping

In this section we study second important topic of Bootstrapping which was first introduced by Gentry [Gen09]. This technique can support homomorphic evaluation of arbitrary polynomial depth circuit. Let us see how the scheme of [BGV12] supports bootstrapping.

Definition 4.1 (Circular Security)**.** Encryption of secret key of BGV under BGV Encryption scheme itself is secure.

The decryption circuit of BGV supports $\log n$ depth i.e. it is in NC_1 class. BGV Encryption scheme can support n^ϵ depth circuit. Hence, BGV construction is powerful enough to support homomorphic evaluation of its own decryption circuit.

4.1 Supporting arbitrary polynomial depth

Let us now see how we can support arbitrary polynomial depth. Assume the circular security of BGV and use the fact that *BGV is powerful enough to support homomorphic evaluation of its own circuit*.

Define the circuit C as

$$C_{CT}(y) = BGV.Dec(CT, y)$$

Note: This circuit C is well defined with y as input and CT hardwired in it.

By the correctness of Evaluation algorithm, we have for any circuit F,

$$F(BGV.Enc(x)) = BGV.Enc(F(x)) \tag{7}$$

Let F be our circuit C_{CT}, then

$$F(x) = C_{CT}(x) = BGV.Dec(CT, x) \tag{8}$$

Given $BGV.Enc(sk)$ as input, we have

$$F(BGV.Enc(sk)) = BGV.Enc(BGV.Dec(CT, sk)) = BGV.Enc(m)$$

The first equality is because of equation (7) and second equality is because of correctness of decryption algorithm.

We recovered $BGV.Enc(m)$ from CT and $BGV.Enc(sk)$ where CT is itself encryption of m.

Why are we taking encryption of m, doing some fancy steps and again outputting encryption of m? The reason is: CT is encryption of m with large noise. Homomorphic evaluation of decryption circuit removes this large noise and adds small new noise.

We know we can evaluate a circuit of depth n^ϵ using the BGV construction. We get some noise. This construction cannot go beyond n^ϵ depth. Therefore, we run the BGV Encryption scheme, get some noise, use the homomorphic decryption procedure which removes old large noise and give us the encryption of m with new small noise. Again we run BGV Encryption scheme up to depth n^ϵ, get noise, remove it using bootstrapping and again get encryption of m with new small noise and repeat it. Thereby, using this bootstrapping process we can support polynomial depth circuit.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 595–618, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.