6.892 Computing on Encrypted Data	September 30, 2013
Lecture 4	
Lecturer: Vinod Vaikuntanathan	*Scribe: Cheng Chen*

1 Introduction

In the previous lecture we constructed a LWE-based secret-key encryption scheme that is somewhat homomorphic. More precisely, the scheme supports evaluation of $\epsilon \log n$ depth circuits for every $\epsilon < 1$ after using the Dimension Reduction technique that we also introduced in the previous lecture.

Building upon this starting point, we will introduce two more techniques in this lecture. The first technique is called *Modulus Reduction*[2] and can improve the scheme to support evaluation of $O(n^\epsilon)$ depth circuits for every $\epsilon < 1$, which results in a *leveled FHE* scheme, i.e. a FHE scheme that can evaluate L depth circuit for every L. The second technique called *Bootstrapping*[4] is a general transformation from a sufficiently strong HE scheme to a FHE scheme that can evaluate arbitrary circuits of polynomial size.

2 Starting point: Quadratic HE with Dimension Reduction

In the following, we provide a brief review of the LWE-based secret-key encryption scheme equipped with Dimension Reduction technique. The choose of parameter will be discussed later. q is some odd prime. Denote $\lceil \frac{q}{2} \rfloor = \frac{q+1}{2}$ and $\log q = \lceil \log_2 q \rceil$. All the operations throughout the note are in ring $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots, 0, \ldots, \frac{q-1}{2}\}$ unless specified explicitly. We will also use a non standard notation $[n] = \{0, 1, \ldots, n-1\}$ for convenience.

- $Gen(1^\lambda, 1^L) \to (sk, evk)$:
  - For $l \in [L+1]$, choose $\vec{t_l} \leftarrow \mathbb{Z}_q^n$, let $\vec{s_l} = (-\vec{t_l}, 1)$.
  - For $l \in [L]$, $i, j \in [n+1]$, $\tau \in [\log q]$, let $\overrightarrow{\psi_{l,i,j,\tau}} = (\overrightarrow{a_{l,i,j,\tau}}, \langle \overrightarrow{a_{l,i,j,\tau}}, \overrightarrow{t_{l+1}} \rangle + e_{l,i,j,\tau} + 2^\tau s_{l,i} s_{l,j})$ where $\overrightarrow{a_{l,i,j,\tau}} \leftarrow \mathbb{Z}_q^n$ and $e_{l,i,j,\tau} \leftarrow \chi$ are chosen independently.
  - Output $sk = (\vec{s_0}, \ldots, \vec{s_L})$ and $evk = (\{\overrightarrow{\psi_{l,i,j,\tau}}\}_{l,i,j,\tau}, \vec{c^*})$ where $(\vec{c^*}, 0) \leftarrow Enc(sk, 1)$ will be used later to raise level.
- $Enc(sk, \mu \in \{0,1\}) \to (\vec{c}, l)$:
  - Choose $\vec{a} \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi$. Let $\vec{c} = (\vec{a}, \langle \vec{a}, \vec{t_0} \rangle + e + \mu \lceil \frac{q}{2} \rfloor)$. Output $(\vec{c}, 0)$.
- $Dec(sk, (\vec{c}, l)) \to \mu$:
  - Compute $\mu' = \langle \vec{c}, \vec{s_l} \rangle$. Output 0 if $-\frac{q}{4} \le \mu' \le \frac{q}{4}$ and 1 otherwise.
- $Add(evk, (\vec{c_1}, l_1), (\vec{c_2}, l_2)) \to (\overrightarrow{c_{add}}, l_{add})$:
  - If $l_1 \neq l_2$, raise the lower one by $Mult$ with the help of $\vec{c^*}$ until we have $l = l_1 = l_2$.
  - Output $\overrightarrow{c_{add}} = \vec{c_1} + \vec{c_2}$ and $l_{add} = l$.
- $Mult(evk, (\vec{c_1}, l_1), (\vec{c_2}, l_2)) \to (\overrightarrow{c_{mult}}, l_{mult})$:
  - If $l_1 \neq l_2$, raise the lower one by $Mult$ with the help of $\vec{c^*}$ until we have $l = l_1 = l_2$.
  - For $i, j \in [n+1]$, $\tau \in [\log q]$, let $c_{i,j,\tau}$ be the τth bit[1] of $2c_{1,i}c_{2,j}$.

[1] Least Significant Bit (LSB) is 0th bit.

– Output $\overrightarrow{c_{mult}} = \sum_{i,j,\tau} c_{i,j,\tau} \overrightarrow{\psi_{l,i,j,\tau}}$ and $l_{mult} = l + 1$.

We briefly review the correctness and the maximum levels that can be correctly evaluated. To do this, we inductively argue that a ciphertext $\vec{c}$ of message $\mu \in \{0,1\}$ at level l satisfies $\langle \vec{c}, \vec{s_l} \rangle = e + \mu \lceil \frac{q}{2} \rfloor$ for some $e \ll q$. Clearly, the ciphertext at level 0 outputted by Enc satisfies this condition and Dec can decrypt ciphertext of this form perfectly.

- Addition correctness:
  $\langle \overrightarrow{c_{add}}, \overrightarrow{s_{l_{add}}} \rangle = \langle \vec{c_1} + \vec{c_2}, \vec{s_l} \rangle = \langle \vec{c_1}, \vec{s_l} \rangle + \langle \vec{c_2}, \vec{s_l} \rangle = (e_1 + e_2) + (\mu_1 + \mu_2 \bmod 2) \lceil \frac{q}{2} \rfloor$ since $2 \lceil \frac{q}{2} \rfloor = 1$.
- Multiplication correctness:
  $\begin{aligned} \langle \overrightarrow{c_{mult}}, \overrightarrow{s_{l_{mult}}} \rangle &= \langle \sum_{i,j,\tau} c_{i,j,\tau} \overrightarrow{\psi_{l,i,j,\tau}}, \overrightarrow{s_{l+1}} \rangle = \sum_{i,j,\tau} c_{i,j,\tau} \langle \overrightarrow{\psi_{l,i,j,\tau}}, \overrightarrow{s_{l+1}} \rangle = \sum_{i,j,\tau} c_{i,j,\tau} (e_{l,i,j,\tau} + 2^\tau s_{l,i} s_{l,j}) \\ &= e_{dr} + \sum_{i,j} 2 c_{1,i} c_{2,j} s_{l,i} s_{l,j} = e_{dr} + \langle 2\vec{c_1} \otimes \vec{c_2}, \vec{s_l} \otimes \vec{s_l} \rangle = e_{dr} + 2 \langle \vec{c_1}, \vec{s_l} \rangle \cdot \langle \vec{c_2}, \vec{s_l} \rangle \\ &= e_{dr} + 2(e_1 + \mu_1 \lceil \tfrac{q}{2} \rfloor)(e_2 + \mu_2 \lceil \tfrac{q}{2} \rfloor) = e_{dr} + 2(e_1 e_2 + \mu_1 e_2 + \mu_2 e_1) + \mu_1 \mu_2 \lceil \tfrac{q}{2} \rfloor. \end{aligned}$
  where $e_{dr} = \sum_{i,j,\tau} c_{i,j,\tau} e_{l,i,j,\tau}$ is the error introduced by Dimension Reduction.

Denote the errors $e_{add} = e_1 + e_2$ and $e_{mult} = e_{dr} + 2(e_1 e_2 + \mu_1 e_2 + \mu_2 e_1)$. Assume e_1 and e_2 (and also χ) are B-bounded, i.e. $Pr[|e_1| > B \text{ or } |e_2| > B] < negl(\lambda)$. Then we have $|e_{add}| \leq 2B$ and $|e_{mult}| \leq ((n+1)^2 \log q + 4)B + 2B^2$. Note that in our parameter setting, n and $\log q$ is polynomial in λ while B is usually subexponential to λ, therefore $|e_{add}| \approx poly(\lambda)B$ and $|e_{mult}| \approx O(B^2)$ and multiplication limits the maximum level that can be evaluated.[2]

When χ is e_{init}-bounded, the error at level L is approximately bounded by $e_{init}^{2^L}$. We need $e_{init}^{2^L} < q/4$ for correct decryption. For security, the best known algorithm for LWE runs in time approximately $2^{n/\log(q/e_{init})}$. Therefore we choose e_{init} to be polynomial in $n = \lambda$ and $q = 2^{n^\epsilon}$ for every $\epsilon < 1$ and $L \approx \log\log q \approx \epsilon \log n$.

3 Modulus Reduction

In this section, we will show a technique called Modulus Reduction[1] that can improve the construction above to support evaluation of $O(n^\epsilon)$ depth circuit for every $\epsilon < 1$. Before we start, note that most of the operations in this section is in $\mathbb{R}$ and $[\cdot]_q$ means modulo into $\mathbb{Z}_q$. The key observation is that to decrypt correctly, we only care about the inner product $\langle \vec{c}, \vec{s} \rangle$ instead of $\vec{c}$, while the increase of noise only depends on the magnitude of the original noise. More precisely, [1] proves that if we have a ciphertext $\vec{c}$ in $\mathbb{Z}_q$ for some secret key $\vec{s}$, we can construct a new ciphertext $\vec{c'} = \lceil \frac{q'}{q} \vec{c} \rfloor$ in $\mathbb{Z}_{q'}$ for the same secret key, where the multiplication is work in $\mathbb{R}$ in each coordinate. The correctness is ensured by $[\langle \vec{c}, \vec{s} \rangle]_q = \frac{q}{q'} [\langle \vec{c'}, \vec{s} \rangle]_{q'} + small$. You might wonder why this is useful. And the magical part can be best described in the following table excerpted from [5].

noise/modulus	without modulus reduction	using modulus reduction
fresh ciphertext	B/B^{10}	B/B^{10}
level 1, $deg = 2$	B^2/B^{10}	$B^2/B^{10} = B/B^9$
level 2, $deg = 4$	B^4/B^{10}	$B^2/B^9 = B/B^8$
level 3, $deg = 8$	B^8/B^{10}	$B^2/B^8 = B/B^7$
level 4, $deg = 16$	decryption error! B^{16}/B^{10}	$B^2/B^7 = B/B^6$

Suppose we choose χ to be B-bounded and the module q to be approximately B^{10}. In our original scheme without using modulus reduction, the noise squares after we increase one level, i.e. do one multiplication. Therefore it can only support $\lfloor \log 10 \rfloor$ levels at most and have decryption error after that. If we use this modulus reduction technique, after the first multiplication, the noise squares to B^2 alike. But using the

[2] We can write the circuit as a polynomial in the inputs, and usually the coefficients and the number of monomials are polynomial in the number of inputs, and therefore also polynomial in λ.

claim we discussed above, we can divide both noise and modular by B, resulting in a ciphertext with noise B in ring Z_{B^9}.[3] In this way, we can support 10 levels in total.

There were some concerns about security raised in the class. But none of them are necessary. First, all the operations are done on the ciphertext and we are not going to publish any additional parameters. Thus the original reduction to LWE still holds. Second, this noise control technique is not likely to be an attack to LWE since the hardness of LWE depends on the dimension and the ratio between modulus and noise. When using this technique, we change neither of them.

The technique talked in the class is a refined and more direct way to control noise. We will first show the idea roughly and then present the full scheme. By induction, we already have $\langle \vec{c_1}, \vec{s_l} \rangle = z_1 q + e_1 + \mu_1 \lceil \frac{q}{2} \rfloor$ and $\langle \vec{c_2}, \vec{s_l} \rangle = z_2 q + e_2 + \mu_2 \lceil \frac{q}{2} \rfloor$ for some integer z_1 and z_2 in $[-(n+1)q, (n+1)q]$. Therefore, we have $\langle \frac{2}{q}\vec{c_1} \otimes \vec{c_2}, \vec{s_l} \otimes \vec{s_l} \rangle = z_{mr} q + e_{mr} + \mu_1 \mu_2 \lceil \frac{q}{2} \rfloor$ where $z_{mr} = 2z_1 z_2 + z_1 \mu_2 + z_2 \mu_1$ and $e_{mr} = 2z_1 e_2 + 2z_2 e_1 + z_1 \mu_2 + z_2 \mu_1 + e_1 \mu_2 + e_2 \mu_1 + \mu_1 \mu_2 / 2 + \frac{2e_1 e_2 + e_1 \mu_2 + e_2 \mu_1 + \mu_1 \mu_2 / 2}{q}$. The result has the same form as a correct ciphertext, but we need to argue that the error term e_{mr} is much smaller than q. Clearly e_{mr} is dominated by $2z_1 e_2 + 2z_2 e_1$, which can be as large as $[-4(n+1)qB, 4(n+1)qB]$. We can improve this by using the result we discussed in the last lecture that the secret key of LWE can be instead sampled from error distribution χ and get a bound $[-4(n+1)B^2, 4(n+1)B^2]$, but this alone actually doesn't help us since it is just our original error rate. To solve this, we will apply the bitdecomp trick we have seen in the last lecture on the secret key s_l. The full scheme is exactly the same as the one we showed in Section 2 except the Gen and $Mult$ parts, which we will show below. Also note that all the operations below are in ring $\mathbb{Z}_q$ except the ones to compute $d'_{i,j,\tau}$.

- $Gen(1^\lambda, 1^L) \to (sk, evk)$:
  - For $l \in [L+1]$, choose $\vec{t_l} \leftarrow \mathbb{Z}_q^n$, let $\vec{s_l} = (-\vec{t_l}, 1)$.
  - For $l \in [L]$, $i, j, \in [n+1]$, $\tau \in [\log q]$, let $s_{l,i,j,\tau}$ be the τth bit of $s_{l,i} s_{l,j}$.
  - For $l \in [L]$, $i, j \in [n+1]$, $\tau, \upsilon \in [\log q]$, let $\overrightarrow{\psi_{l,i,j,\tau,\upsilon}} = (\overrightarrow{a_{l,i,j,\tau,\upsilon}}, \langle \overrightarrow{a_{l,i,j,\tau,\upsilon}}, \overrightarrow{t_{l+1}} \rangle + e_{l,i,j,\tau,\upsilon} + 2^\upsilon s_{l,i,j,\tau})$ where $\overrightarrow{a_{l,i,j,\tau,\upsilon}} \leftarrow \mathbb{Z}_q^n$ and $e_{l,i,j,\tau,\upsilon} \leftarrow \chi$ are chosen independently.
  - Output $sk = (\vec{s_0}, \ldots, \vec{s_L})$ and $evk = (\{\overrightarrow{\psi_{l,i,j,\tau}}\}_{l,i,j,\tau}, \vec{c^*})$ where $(\vec{c^*}, 0) \leftarrow Enc(sk, 1)$ will be used to raise level.
- $Mult(evk, (\vec{c_1}, l_1), (\vec{c_2}, l_2)) \to (\overrightarrow{c_{mult}}, l_{mult})$:
  - If $l_1 \neq l_2$, raise the lower one by $Mult$ with the help of $\vec{c^*}$ until we have $l = l_1 = l_2$.
  - For $i, j \in [n+1]$, $\tau \in [\log q]$, let $d'_{i,j,\tau} = 2^\tau \frac{2}{q} c_{1,i} c_{2,j}$.
  - For $i, j \in [n+1]$, $\tau \in [\log q]$, let $d_{i,j,\tau} = \lceil d'_{i,j,\tau} \rfloor$, i.e. round to nearest integer.
  - For $i, j \in [n+1]$, $\tau, \upsilon \in [\log q]$, let $c_{i,j,\tau,\upsilon}$ be the υth bit of $d_{i,j,\tau}$.
  - Output $\overrightarrow{c_{mult}} = \sum_{i,j,\tau,\upsilon} c_{i,j,\tau,\upsilon} \overrightarrow{\psi_{l,i,j,\tau,\upsilon}}$ and $l_{mult} = l + 1$.

The correctness of multiplication follows that

$$\begin{aligned} \langle \overrightarrow{c_{mult}}, \overrightarrow{s_{l_{mult}}} \rangle &= \langle \textstyle\sum_{i,j,\tau,\upsilon} c_{i,j,\tau,\upsilon} \overrightarrow{\psi_{l,i,j,\tau,\upsilon}}, \overrightarrow{s_{l+1}} \rangle = \sum_{i,j,\tau,\upsilon} c_{i,j,\tau,\upsilon} \langle \overrightarrow{\psi_{l,i,j,\tau,\upsilon}}, \overrightarrow{s_{l+1}} \rangle = \sum_{i,j,\tau,\upsilon} c_{i,j,\tau,\upsilon} (e_{l,i,j,\tau,\upsilon} + 2^\upsilon s_{l,i,j,\tau}) \\ &= e_{dr} + \textstyle\sum_{i,j,\tau} d_{i,j,\tau} s_{l,i,j,\tau} = e_{dr} + e_{round} + \sum_{i,j,\tau} d'_{i,j,\tau} s_{l,i,j,\tau} = e_{dr} + e_{round} + \sum_{i,j} \frac{2}{q} c_{1,i} c_{2,j} s_{l,i} s_{l,j} \\ &= e_{dr} + e_{round} + \langle \tfrac{2}{q}\vec{c_1} \otimes \vec{c_2}, \vec{s_l} \otimes \vec{s_l} \rangle = e_{dr} + e_{round} + e_{mr} + \mu_1 \mu_2 \lceil \tfrac{q}{2} \rfloor \end{aligned}$$

where $e_{dr} = \sum_{i,j,\tau,\upsilon} c_{i,j,\tau,\upsilon} e_{l,i,j,\tau,\upsilon}$ is the error introduced by Dimension Reduction, $e_{round} = \sum_{i,j,\tau} (\lceil d'_{i,j,\tau} \rfloor - d'_{i,j,\tau}) s_{l,i,j,\tau}$ is the error introduced by rounding, and e_{mr}, which we have specified before, is the error introduced by Modulus Reduction. We can simply get $|e_{dr}| \leq (n+1)^2 \log^2 qB$ and $|e_{round}| \leq \frac{1}{2}(n+1)^2 \log qB$ using the aforementioned property of LWE. The key point is that after we bitdecomp s_l above, z_1 and z_2 are in $[-(n+1)^2 \log q, (n+1)^2 \log q]$[4], and therefore $|e_{mr}| \leq (4(n+1)^2 \log q + 2)B + 2(n+1)^2 \log q + \frac{1}{2} + o(1)$.

[3] Choose $q = B^{10}$ and $q' = B^9$

[4] z_1 and z_2 are even in $[-(n+1)^2 \log B, (n+1)^2 \log B]$ using the aforementioned property of LWE.

Following the same argument as in the end of Section 2, when χ is e_{init}-bounded, the error at level L is approximately bounded by $poly(n)^L e_{init}$. We need $poly(n)^L e_{init} < q/4$ for correct decryption. For security, the best known algorithm for LWE runs in time approximately $2^{n/\log(q/e_{init})}$. Therefore we choose e_{init} to be polynomial in $n = \lambda$ and $q = 2^{n^\epsilon}$ for every $\epsilon < 1$ and $L \approx \log q \approx n^\epsilon$.

If we only require correctness to be held with high probability, we can get better noise bound. The dominating term e_{dr} is the sum of independent discrete Gaussian random variables. We can use standard way as tail inequality of Gaussian distribution (or Chernoff bound) to get a better maximum level by a constant factor.

So far, we prove the following theorem.

Theorem 1. *For every $L > 0$, there exists $\epsilon < 1$ and a L-leveled FHE scheme under $LWE_{n,q,B}$ assumption, where $q/B \leq 2^{n^\epsilon}$, and the size of the parameters are $poly(L, n^{1/\epsilon})$.*

4 Bootstrapping

In this section, we will describe Gentry's clever Bootstrapping framework[4] that transforms a "sufficiently strong" homomorphic encryption scheme into a fully homomorphic encryption scheme.

Definition 2. *($\mathcal{C}$-Homomorphism) An encryption scheme $\mathcal{E}$ is $\mathcal{C}$-homomorphic for some circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ if for any λ, any $C \in \mathcal{C}_\lambda$ with t inputs, any message $m_1, \ldots, m_t \in \{0,1\}$, and any fresh ciphertext $c_1 \leftarrow \mathcal{E}.Enc(sk, m_1), \ldots, c_t \leftarrow \mathcal{E}.Enc(sk, m_t)$, it holds that $\mathcal{E}.Dec(sk, \mathcal{E}.Eval(evk, C, c_1, \ldots, c_t)) = C(m_1, \ldots, m_t)$.*

It's possible to define a weaker notion that only requires the above equation to hold with high probability.

Definition 3. *(Augmented Decryption Circuit) The augmented decryption circuit $AD_\mathcal{E}$ for some encryption scheme $\mathcal{E}$ consists of a $NAND$-gate connecting two copies of $D_\mathcal{E}$, the decryption circuit[5] of $\mathcal{E}$ that takes a secret key and ciphertext as input and outputs a message bit.*

Theorem 4. *(Boostrapping[4]) Given an encryption scheme $\mathcal{E}$ that is $\{AD_\mathcal{E}\}$-homomorphic, there is a black-box construction of a fully homomorphic encryption scheme $\mathcal{E}'$ with additional assumption that $\mathcal{E}$ is weak circular secure. Alternatively, there is a black-box construction of a L-leveled homomorphic encryption scheme with the same assumption for any $L = poly(\lambda)$. Also, we call such scheme $\mathcal{E}$ bootstrappable.*

Note that our current scheme can only support L-level homomorphic evaluation for $L = n^\epsilon$ for any $\epsilon < 1$ and usually $n = O(\lambda)$. We will give the black-box construction below and prove its correctness. For those curious about the security proof, please refer to Section 4.2 of Gentry's PhD thesis[3].

- $\mathcal{E}'.Gen(1^\lambda) \to (sk, evk)$:
  - Let $(sk, evk) \leftarrow \mathcal{E}.Gen(1^L, 1^\lambda)$ where L is the depth of $AD_\mathcal{E}$.
  - For $i \in [|sk|]$, compute $\widehat{sk_i} \leftarrow \mathcal{E}.Enc(sk, sk_i)$ where sk_i is the ith bit of sk.
  - Output sk as the secret key and $(evk, \{\widehat{sk_i}]\}_i)$ as the evaluation key.
- $\mathcal{E}'.Enc(sk, \mu \in \{0,1\}) \to (c, l)$:
  - Output $(c, 0) \leftarrow \mathcal{E}.Enc(sk, \mu)$.
- $\mathcal{E}'.Dec(sk, (c, l)) \to \mu$:
  - Output $\mu \leftarrow \mathcal{E}.Dec(sk, (c, l))$.
- $\mathcal{E}'.Eval((evk, \{\widehat{sk_i}\}_i), C, (c_1, l_1), \ldots, (c_t, l_t)) \to (c, l)$:

[5] We are assuming that all the ciphertexts are padded into the same length

- Transform C into a circuit with only NAND gate and possibly constant input 1[6]. Label all the non-input gates by $\{0, 1, \ldots, m\}$ from lowest level to output level.
- For $k \in [m+1]$ do the following.
  - Let (c_0, l_0) and (c_1, l_1) be the ciphertexts corresponding to the two inputs of gate k that we have computed.
  - For $j \in [|(c_0, l_0)|]$, compute $\widehat{c_{0,j}} \leftarrow \mathcal{E}.Enc(sk, c_{0,j})$ where $c_{0,j}$ is the jth bit of (c_0, l_0). Do the same calculation for (c_1, l_1) and get $\widehat{c_{1,j}}$.
  - Compute $(c, l) \leftarrow \mathcal{E}.Eval(evk, A_{\mathcal{E}}, \{\widehat{sk_i}\}_i, \{\widehat{c_{0,j}}\}_j, \{\widehat{sk_i}\}_i, \{\widehat{c_{1,j}}\}_j)$ corresponding to gate k.
- Output (c, l) corresponding to the output gate m.

We prove the correctness of $\mathcal{E}'.Eval$ by inductively prove that the ciphertext (c, l) corresponding to gate k satisfies that $\mathcal{E}.Dec(sk, (c, l)) = m$ for m corresponding to the real value at gate k when evaluating circuit C, from lowest level to output level. Clearly, it holds for input level. Now for gate k with input (c_0, l_0) and (c_1, l_1) such that $\mathcal{E}.Dec(sk, (c_0, l_0)) = m_0$ and $\mathcal{E}.Dec(sk, (c_1, l_1)) = m_1$, we have $\mathcal{E}.Dec(sk, (c, l)) = \mathcal{E}.Dec(sk, \mathcal{E}.Eval(evk, AD_{\mathcal{E}}, \{\widehat{sk_i}\}_i, \{\widehat{c_{0,j}}\}_j, \{\widehat{sk_i}\}_i, \{\widehat{c_{1,j}}\}_j)) = AD_{\mathcal{E}}(sk, c_0, sk, c_1) = NAND(m_0, m_1)$ as desired since $\mathcal{E}$ is $AD_{\mathcal{E}}$-homomorphic when L is chosen to be the depth of $AD_{\mathcal{E}}$.

The second part of the theorem can be proved by constructing $\mathcal{E}'$ using a chain of encrypted secret keys of $\mathcal{E}$ instead of a self-cycle. Also, when assuming weak circular security, we no longer need a chain of "encrypted" secret key for Dimension Reduction.

5 [BGV12] is bootstrappable

In this section, we will shown that our current scheme is bootstrappable, i.e. can evaluate its own augmented decryption circuit. It suffices to prove that the decryption circuit of our current scheme is much shallower than n^{ϵ}, the maximum depth the scheme can support. More precisely, we will show that the decryption circuit is of depth $O(\log n)$ with constant factor bigger than 1. This illustrates why Dimension Reduction alone is not enough.

The decryption algorithm for our scheme is basically $Round_{q/2}(\langle c, s \rangle \bmod q)$. The multiplication of $c_i s_i$ of each coordinate can be implemented in $O(\log \log q) = O(\log n)$ depth circuit using FFT. The addition of $n+1$ integers with size at most $O(\log n)$ can be reduced to the addition of 2 integers with size slightly increased but still $O(\log n)$ in $O(\log n)$ depth circuit using a technique called three-for-two trick. The rounding actually only depends on the first constant number of bits and can be implemented in constant depth circuit. However, we never actually compute the modulo q operation. So we need to rounding and compare for each possible slot. There are at most $2(n+1)B+1 = poly(n)$ slots, the conjunction of them can be computed in $O(\log n)$ depth circuit. Above all, the decryption circuit can be implemented in $O(\log n)$ circuit and therefore our scheme is bootstrappable.

Theorem 5. *There is a FHE scheme under LWE and weak circular secure assumption. Also, there is a L-leveled FHE scheme for every $L = poly(\lambda)$ under LWE assumption.*

References

[1] Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan: *(Leveled) fully homomorphic encryption without bootstrapping*. ITCS 2012: 309-325

[2] Zvika Brakerski, Vinod Vaikuntanathan: *Efficient fully homomorphic encryption from (standard) LWE*. FOCS 2011: 97-106

[3] Craig Gentry: *A fully homorphic encryption scheme*. (PhD thesis) Stanford University 2009

[6] We can assume without loss of generality that there is always an encryption of 1 in *evk*.

[4] Craig Gentry: *Fully homomorphic encryption using ideal lattices.* STOC 2009: 169-178

[5] Shai Halevi, Reut Levi: *FHE from LWE, without bootstrapping [BV11, Gen11].* Lecture note of Homomorphic Encryption and Lattices 2011