CS6115: Structure versus Hardness in Cryptography Aug 18, 2021

Lecture 5 : Dimension reduction & Handling noise growth

Lecturer: *Shweta Agrawal* Scribe: *Akhil Vanukuri*

1 Notations

All the vectors are represented as column matrices. Note : $\overrightarrow{w}$ indicates a vector but we drop the arrow above to represent it as a column matrix when we talk about them in the context of matrix operations.

Let,

$\overrightarrow{w} = (w_1, w_2, ...w_p) \in Z^p$,

$\overrightarrow{v} = (v_1, v_2, ...v_r) \in Z^r$

then, $\overrightarrow{w} \otimes \overrightarrow{z} \triangleq wz^T$ (matrix multiplication)

Observe that, $\overrightarrow{w} \otimes \overrightarrow{z}$ according to the above definition is of order $p \times r$ matrix, but throughout this lecture and usually in crypto, we convert a $p \times r$ order matrix into a vector of pr dimensions (i.e, a column matrix of order $pr \times 1$ by concatenating one row after the other).

For example,

Let, $\overrightarrow{w} = (1, 2) \in Z^2$, $\overrightarrow{v} = (1, 2, 3) \in Z^3$, By the defintion stated above, $\overrightarrow{w} \otimes \overrightarrow{z} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix}$

but we re-write $\overrightarrow{w} \otimes \overrightarrow{z}$ as a vector in Z^6 i.e, $\overrightarrow{w} \otimes \overrightarrow{z} = (1, 2, 3, 2, 4, 6)$

if $\overrightarrow{w}$ and $\overrightarrow{v}$ are vectors of same dimensions then $< \overrightarrow{w}, \overrightarrow{v} > \ \triangleq$ dot product of $\overrightarrow{w}$ and $\overrightarrow{v}$.

Z_q means Z/qZ.

$[n] = \{1,\ 2, ..,\ n\}$ where $n \in Z$.

$\triangleq$ means defined as

$\ni$ means such that

if $\overrightarrow{w} \in Z^p$ then for any $i_1, .., i_k \in Z$

Let, $total = \prod_{j=1}^{j=k}(i_j) \leq p$, w_{total} is the $total^{th}$ element of the vector $\overrightarrow{w}$.

Note: $w \leftarrow X$ means w is chosen randomly (uniformly) from X, where X is a distribution (or) a ring.

2 Preliminaries

In this scheme, originally[3], we use nearest integer of $\lfloor \frac{q}{2} \rceil$, but instead of that for easy calculations, we replace it with $(\frac{q+1}{2})$ (This doesn't affect our scheme).

Let, SKE be the symmetric key encryption scheme mentioned in the Lecture 4 [1]

where,

n be the security parameter[2].

χ be a B-bounded error distribution[2].

q be the odd prime.

$B << q$ [5]

Note : All the operations in the below scheme and this paper here after are operations in the ring Z_q (i.e, mod q is implicitly assumed to be applied to the final result of every calculation).

Note : In this paper when we say matrix multiplication where elements of the matrix $\in Z_q$ then after the usual matrix multiplication, (mod q) is applied on each individual element of the matrix.

SKE = {keygen, enc, dec} (please refer to the detailed scheme mentioned in the previous lectures [1, 2] along with the definitions for terms that are not defined in this paper).

Let,

$\overrightarrow{s} \in Z_q^n$ be the ouput of SKE.keygen(1^n),

$\overrightarrow{t} \triangleq (-\overrightarrow{s}, 1) \in Z_q^{(n+1)}$,

$\overrightarrow{c_1}$, $\overrightarrow{c_2}$ be two cipher-texts produced by running two instances of SKE.enc with same secret key $\overrightarrow{t}$ on the messages $\overrightarrow{m_1}$, $\overrightarrow{m_2}$ respectively $\ni \overrightarrow{m_1}, \overrightarrow{m_2} \in \{0,1\}$, $\overrightarrow{c_1}, \overrightarrow{c_2} \in Z_q^{n+1}$, $\overrightarrow{t} \in Z_q^{n+1}$.

$\therefore \forall\ i \in \{1,2\},\ \overrightarrow{c_i} = (\overrightarrow{a_i}, b_i) \ni b_i \triangleq < \overrightarrow{a_i}, \overrightarrow{s} > + e_i + m_i(\frac{q+1}{2})$ where, $e_i \leftarrow \chi$

Now, we will recall how we proceeded with the multiplicative scheme. We already showed in [1] that,

$$< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t} \otimes \overrightarrow{t} > \ = \ 2e_1e_2 + (m_1e_2 + m_2e_1) + m_1m_2(\frac{q+1}{2}) \tag{1}$$

For now, let us say

$$< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t} \otimes \overrightarrow{t} > \ \approx \ m_1 m_2 (\frac{q+1}{2}) \tag{2}$$

We will deal with the consequences of this assumption a bit later. Observe that after one multiplication, the cipher-text changes from $(n+1)$ dimensions to $(n+1)^2$ dimensions in ($2\overrightarrow{c_1} \otimes \overrightarrow{c_2}$).

$\therefore$ If we take the multiplicative operation 'i' times then the dimension of cipher-text increases to $(n+1)^{2^i} \implies$ maximum depth that the multiplicative circuit can support is less than an order of $log(log(n))$.

Reason:

Substitute $i = log(log(n))$ then $2^{n^i} = n^{log(n)} \implies$ the output size is of the order super-polynomial in n. But all the parameters must remain polynomial in n. Therefore, no.of levels of multiplication allowed and thereby depth of the circuit allowed is strictly less than order of log(log(n))

Therefore, we try a method called "*dimension reduction*".

3 Dimension Reduction

3.1 Setting up the idea

Let,

$\overrightarrow{c_{mult}} \triangleq \ 2\overrightarrow{c_1} \otimes \overrightarrow{c_2} \ \in \ Z_q^{(n+1)^2}$

$\overrightarrow{sk'} \triangleq \overrightarrow{t_1} \otimes \overrightarrow{t_1} \ni \overrightarrow{t_1} = \overrightarrow{t} \ \in \ Z_q^{(n+1)^2}$

In this method, we construct a matrix $B(order\ is\ (n+1) \times (n+1)^2) \ni B^T t_2 \approx \overrightarrow{t_1} \otimes \overrightarrow{t_1}$ where $\overrightarrow{t_2} = (-\overrightarrow{s_2}, 1)$ and $\overrightarrow{s_2} \leftarrow Z_q^n$

If $B^T t_2 \approx \overrightarrow{t_1} \otimes \overrightarrow{t_1}$, then notice that,

$< Bc_{mult}, t_2 > \ = \ (Bc_{mult})^T t_2 \ = \ (c_{mult})^T (B^T t_2) \ \approx \ (c_{mult})^T (\overrightarrow{t_1} \otimes \overrightarrow{t_1}) \ \approx \ m_1 m_2 (\frac{q+1}{2})$

Notice that, $Bc_{mult} \in Z_q^{(n+1)}$(i.e, of order $((n+1) \times 1)$ as a matrix)

Therefore, if we construct a matrix $B \ni B^T t_2 \approx \overrightarrow{t_1} \otimes \overrightarrow{t_1}$ then we can use $\overrightarrow{t_2}$ as the new key and Bc_{mult} as the cipher-text which satisfies our correctness constraint while reducing the dimension of the cipher text.

3.2 Constructing the Hint matrix B

Let, $\overrightarrow{a_{ij}}^* \leftarrow Z_q^n$ (sampled afresh), $\overrightarrow{t_2}$ be as defined in (3.1).

$$\forall i, j \in [n+1], \psi_{ij} \triangleq \widetilde{Enc}(t_{1i}, t_{1j}) \tag{3}$$

where, ψ_{ij} are columns of the hint matrix $B \ni \psi_{ij} \in Z_q^{(n+1)}$.

$$\widetilde{Enc}(t_{1i}, t_{1j}) \triangleq (\overrightarrow{a_{ij}^*}, b_{ij}^*) \ , \ where, b_{ij}^* =< \overrightarrow{a_{ij}}, \overrightarrow{s_2} > + e_{ij} + t_{1i}t_{ij} \tag{4}$$

Why did we put tilda (˜) on the top of Enc in the equation (4)?

Because, we can view the process mentioned in equation (4) as an encryption on (t_{1i}, t_{1j}) but in the traditional definition of encryption, we are able to decrypt input from the output, but in the above case we donot have such a requirement. Moreover, t_{1i}, t_{1j} can have large norms and might not be recoverable in some cases ($\because \overrightarrow{t_1} = (-\overrightarrow{s}, 1)$ and $\overrightarrow{s}$ is sampled uniformly from Z_q^n)

Let, $[e_{temp}]$ be an error vector $\in Z_q^{(n+1)^2}$ where it's elements are e_{ij} $\forall i, j \in [n+1]$

Notice that,

$\forall\, i, j \in [n+1]\ < \psi_{ij}, \overrightarrow{t_2} >= - < \overrightarrow{a_{ij}^*}, \overrightarrow{s_2} > + b_{ij}^* = e_{ij} + t_{1i}t_{1j} \approx t_{1i}t_{1j} \implies B^T t_2 = \overrightarrow{t_1} \otimes \overrightarrow{t_1} + [e_{temp}]$

Let,

$\overrightarrow{c'_{mult}} \triangleq Bc_{mult}$, by the above calculations we can see that $\overrightarrow{c'_{mult}} \in Z_q^{n+1}$ ($\because$ we were able to reduce the dimension of the cipher text significantly) and $< \overrightarrow{c'_{mult}}, \overrightarrow{t_2} >$ decrypts correctly to approx. $m_1 m_2(\frac{q+1}{2})$

4 Handling the ignored noise

In the section (3), we didnot account for the noise produced at various stages.

There are two sources of noise :

Source 1 (the noise caused due to dimension reduction) :

we assumed $B^T t_2 \approx \overrightarrow{t_1} \otimes \overrightarrow{t_1}$ but it is $B^T t_2 = \overrightarrow{t_1} \otimes \overrightarrow{t_1} + [e_{temp}]$.

Source 2 (the noise caused due to $< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} >$) :

From equation (1) , $< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} > \ = \ 2e_1e_2 + (m_1e_2 + m_2e_1) + m_1m_2(\frac{q+1}{2})$

but we assumed it to be approx. $m_1 m_2(\frac{q+1}{2})$

In this section we will deal with the un-accounted noise.

4.1 Dealing with the noise caused due to dimension reduction

$[e_{temp}]$ is the noise vector caused due to the dimension reduction that was previously un-accounted. This noise causes a problem because when taking a dot product between $((\overrightarrow{t_1} \otimes \overrightarrow{t_1}) + [e_{temp}])$ and $(2\overrightarrow{c_1} \otimes \overrightarrow{c_2})$, roughly speaking we get $m_1 m_2(\frac{q+1}{2})+$ large noise.

It produces a large noise because $(2\overrightarrow{c_1} \otimes \overrightarrow{c_2})$ is not a low norm vector and, because the noise is large the clear distinction between intervals used in recovering $m_1 m_2$ values is lost $\implies$ we lose decryption correctness.

More Formally,

Let,

$e_{mult} \triangleq 2e_1e_2 + (m_1e_2 + m_2e_1)$ and $e_{dr} \triangleq \sum_{i,j\in[n+1]} c_{mult(ij)}e_{ij}$

$< \overrightarrow{c'_{mult}}, \overrightarrow{t_2} > \; = \; < Bc_{mult}, \overrightarrow{t_2} >$

$= \; (Bc_{mult})^T t_2$

$= \; (c_{mult})^T (B^T t_2)$

$= \; (c_{mult})^T ((\overrightarrow{t_1} \otimes \overrightarrow{t_1}) + [e_{temp}])$

$= \; (c_{mult})^T (\overrightarrow{t_1} \otimes \overrightarrow{t_1}) + (c_{mult})^T [e_{temp}]$

$= \; < c_{mult}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} > + \sum_{i,j\in[n+1]} c_{mult(ij)}e_{ij}$

$= \; 2e_1e_2 + (m_1e_2 + m_2e_1) + m_1m_2(\frac{q+1}{2}) + e_{dr}$

$= \; m_1m_2(\frac{q+1}{2}) + e_{mult} + e_{dr}$

Calculating the bounds on e_{dr} (this is the error caused due to dimension reduction)

$e_{dr} = \sum_{i,j\in[n+1]} c_{mult(ij)}e_{ij}$

$\leq \sum_{i,j\in[n+1]} c_{mult(ij)}B$ ($\because e_{ij}$ is sampled from a B-bounded distribution χ)

$\leq \sum_{i,j\in[n+1]} qB$ ($\because c_{mult} \in Z_q^{n+1}$)

$\leq (n+1)^2 qB$

Thus, e_{dr} can be made greater than $q/4$ easily and the final result may not decode correctly for even low depth circuits. So, let's try and see if we can reduce this bound any further.

4.1.1 Binary representation trick to bound e_{dr}

Suppose we take binary representation of $(2\overrightarrow{c_1} \otimes \overrightarrow{c_2})$. This results in blowing up the vector size by $\lfloor log(q) \rfloor$.

Here, we assume $q \in [2^{n^\epsilon}, 2 \cdot 2^{n^\epsilon})$ which is sub-exponential in n. as stated in [5]. While this might look contradictory, the $\epsilon \in (0,1)$ therefore, $\lfloor (log(q) \rfloor \approx n^\epsilon$ and if we take ϵ appropriately, then we can see that dimension reduction still manages to create a lot of impact.

For the sake of intuition, let's take an example where $n = 10^{10}$ and $\epsilon = 1/10$, Let, $q = 2^{n^\epsilon}$, then $n^2 = 10^{20}$ whereas $nlog(q) = n^{1+\epsilon} = 10^{11}$ which is still better than blowing up cipher-text vector size quadratically (which was the case without dimension reduction).

Now, the question is how will we modify $(\overrightarrow{t_1} \otimes \overrightarrow{t_1})$, so that $< \overrightarrow{t_1} \otimes \overrightarrow{t_1}, 2\overrightarrow{c_1} \otimes \overrightarrow{c_2} > \ \approx \ m_1 m_2 (\frac{q+1}{2})$

Let us define,

$BitDec(x) \triangleq (u_0, ..., u_{\lfloor (log(q)) \rfloor}) \ni x = \sum_{j=0}^{\lfloor log(q) \rfloor} \ u_j 2^j$

$PowersOfTwo(x) \triangleq (x, \ 2x, \ 4x, .., \ 2^{\lfloor log(q) \rfloor} x)$

Claim : if $p, r \in Z_q$, then $< BitDec(a), \ PowersOfTwo(b) > \ = ab \in Z_q$

Proof:

Let,

$BitDec(a) \triangleq (a_0, ..., a_{\lfloor (log(q)) \rfloor})$

$PowersOfTwo(b) \triangleq (b, \ 2b, \ 4b, .., \ 2^{\lfloor log(q) \rfloor} b)$

$< BitDec(a), \ PowersOfTwo(b) > \ = \sum_{j=0}^{\lfloor log(q) \rfloor} \ a_j 2^j b \ = \ b(\sum_{j=0}^{\lfloor log(q) \rfloor} \ a_j 2^j) \ = \ ab \in Z_q$

We can extend the definition of $BitDec$ and $PowersOfTwo$ to vectors as follows:

Let, $\overrightarrow{p} = (p_1, p_2, .., p_k)$, $\overrightarrow{r} = (r_1, r_2, .., r_k) \in Z_q^k$, then

$BitDec(\overrightarrow{p}) \ = \ (\ BitDec(p_1), .., BitDec(p_k) \ ) \in \{0,1\}^{k \lfloor (logq) \rfloor}$

$PowersOfTwo(\overrightarrow{r}) = (\ PowersOfTwo(r_1), .., PowersOfTwo(r_k) \ )$

Therefore, $< BitDec(\overrightarrow{p}), PowersOfTwo(\overrightarrow{r}) > \ = \ < \overrightarrow{p}, \overrightarrow{r} >$

By above definitions and claims, we can say that

$$< BitDec(2\overrightarrow{c_1} \otimes \overrightarrow{c_2}), PowersOfTwo(\overrightarrow{t_1} \otimes \overrightarrow{t_1}) > \ = \ < 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} > \tag{5}$$

Redefining ψ_{ij} by introducing a new index τ to accommodate the new representation.

Let,

$\tau \in [\lfloor (log(q) \rfloor], \forall\ i, j \in [n+1]$

$$\psi_{ij\tau} = (\overrightarrow{a^*_{ij\tau}}, < \overrightarrow{a^*_{ij\tau}}, \overrightarrow{s_2} > + e_{ij\tau} + 2^\tau t_{1i} t_{1j}) \tag{6}$$

where, $\psi_{ij\tau}$ are the columns of matrix B_τ and B_τ is a matrix of order $(n+1) \times (n+1)^2 \lfloor log(q) \rfloor$

Note : B_τ is the modified hint matrix to accommodate the binary representation trick.

Let, $[e_{temp\tau}]$ be the error vector $\in Z^{\lfloor (log\ q) \rfloor (n+1)^2}$ where it's elements are $e_{ij\tau}\ \forall i, j \in [n+1]\ ,\ \tau \in [\lfloor log\ q \rfloor]$

Notice that,

$\forall\ i, j \in [n+1],\ \tau \in [\lfloor (log(q) \rfloor] < \psi_{ij\tau}, \overrightarrow{t_2} > \ = \ e_{ij\tau} + 2^\tau t_{1i} t_{1j}$

$\implies \forall\ i, j \in [n+1],\ \tau \in [\lfloor (log(q) \rfloor] (\psi_{ij\tau})^T t_2 \ = \ e_{ij\tau} + 2^\tau t_{1i} t_{1j}$

$\implies B^T_\tau t_2 = PowerOfTwo(\overrightarrow{t_1} \otimes \overrightarrow{t_1}) + [e_{temp\tau}]$

Notice,

$< B_\tau BitDec(c_{mult}), \overrightarrow{t_2} > \ = \ (B_\tau BitDec(c_{mult}))^T t_2$

$= \ (BitDec(c_{mult}))^T (B_\tau)^T t_2$

$= \ (BitDec(c_{mult}))^T (PowersOfTwo(\overrightarrow{t_1} \otimes \overrightarrow{t_1}) + [e_{temp\tau}])$

$= \ < BitDec(\overrightarrow{c_{mult}}), PowersOfTwo(\overrightarrow{t_1} \otimes \overrightarrow{t_1}) > + < BitDec(\overrightarrow{c_{mult}}), [e_{temp\tau}] >$

$= \ (m_1 m_2(\frac{q+1}{2})) + < BitDec(\overrightarrow{c_{mult}}), [e_{temp\tau}] >$

Let,

$e^{'}_{dr} =< \ BitDec(c_{mult}), [e_{temp\tau}] \ >$ where, $e^{'}_{dr}$ is the error generated due to dimension reduction after using the binary representation trick.

Notice that,

$e^{'}_{dr} \leq (n+1)^2 \lfloor log(q) \rfloor |B|$ (derived similarly to the bound derived on e_{dr} but this time $(BitDec(c_{mult}))_{ij\tau} \in \{0, 1\}$).

So, we managed to get a tighter bound on the error produced due to source 1 (that is dimension reduction).

4.2 Finding bounds on error due to $< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} >$

We are yet to find bound on the second source of noise.

$< 2\overrightarrow{c_1} \otimes \overrightarrow{c_2}, \overrightarrow{t_1} \otimes \overrightarrow{t_1} > \ = \ 2e_1e_2 + (m_1e_2 + m_2e_1) + m_1m_2(\frac{q+1}{2})$

but we assumed it to be approx. $m_1m_2(\frac{q+1}{2})$

w.k.t, $|e_1|, |e_2| \leq B \implies |e_1e_2| \leq B^2$

Observe that, after the first multiplication that results in the error e_{mult}, the subsequent d multiplications result in the error equivalent to $(e_{dr} + e_{mult})^d \leq ((n+1)^2\lfloor log(q) \rfloor B + 2B^2 + 4B)^d \leq (((n+1)^2\lfloor log(q) \rfloor + 4)B + 2B^2)^d$

Note that in our parameter setting, we follow closely the parameters set in [5] to get the bounds. In [5], section 1.1, they discuss the symmetric key based FHE, then in section section 4.1, they make a note of the parameters while describing a public FHE which is very similar to our symmetric key FHE. Therefore, we use the same parameters and manage to get the bound claimed in section 1.1 which is the one that we discussed in this paper.

They take, $q \in [2^{n^\epsilon}, 2 \cdot 2^{n^\epsilon})$ which is sub-exponential in n. While they take n to be a polynomial in security parameter, we assume for ease of calculation that n is the security parameter itself. This does not affect our calculations as we are only trying to get asymptotic bounds.

Therefore, $log(q) \approx n^\epsilon$ is polynomial in n.

For a suitable ϵ, $B \geq q/2^{n^\epsilon} \implies (n+1)^2\lfloor log(q) \rfloor B \geq \frac{(n+1)^2\lfloor log(q) \rfloor q}{2^{n^\epsilon}}$. If we set n appropriately, this can be neglected in comparison to B^2.

Therefore, after d multiplications, $e_{mult} \approx B^{2^d}$

Because, the error due to multiplication is squaring $(e_{dr} + e_{mult})$ in every stage as opposed to the error due to dimension reduction which is added at every stage, therefore the error due to multiplication is much more significant and must be considered to find the depth supported by the circuit.

These are the given conditions for our scheme to work as intended:

For correctness :

$$B^{2^d} \leq q/4 \ni \ d \, is \, the \, depth \, of \, the \, circuit \tag{7}$$

For security :

$$q/B < 2^{n^\epsilon} \tag{8}$$

We need the condition in Eq(7), because as we know that error squares after every multiplication, after 'd' multiplications, the error will be of the order B^{2^d}, and as we have already stated the need for error to be less than q/4 for correctness, the above condition must hold.

The condition in Eq(8), is an empirical observation.

Using Eq(7) and Eq(8), for a large enough 'd' and 'n' and a constant B (For ease of calculation), $q/4B \approx B^{2^d}$ and $q/4B \approx 2^{n^\epsilon} \implies 2^d log(B) \approx n^\epsilon \implies d \approx \epsilon log(n)$

Though the above calculation is hand-wavy and the depth of the circuit isn't exactly of the order log(n), the idea is that we could bring the the depth supported by the circuit from something less than log(log(n)) to something closer to log(n) which is a very good improvement.

References

[1] URL: https://www.cse.iitm.ac.in/~shwetaag/6115/Lec4.pdf.

[2] URL: https://www.cse.iitm.ac.in/~shwetaag/6115/Lec3.pdf.

[3] URL: https://people.csail.mit.edu/vinodv/6892-Fall2013/lecture03.pdf.

[4] URL: https://people.csail.mit.edu/vinodv/6892-Fall2013/lecture04.pdf.

[5] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. 2011, pp. 97–106. DOI: 10.1109/FOCS.2011.12.