University of Illinois, Urbana Champaign
CS 598DK Special Topics in Cryptography

Instructor: Dakshita Khurana
Scribe: Sanket Kanjalkar, Yunqi Li
Date: October 09, 2019

LECTURE
11

Fully Homomorphic Encryption I

11.1 Construction of Fully Homomorphic Encryption:

As described previousy, FHE is an ecryption scheme(Gen, Enc, Dec) with an additional algorithm called Eval. In particular, we want to construct such a Eval namely for two operations, Addition and Multiplication. Constructing such a FHE scheme which must work for **all** functions f might seem like daunting task, but can use the following fact to ease our task.

FACT 11.1. *It turns out that all functions can be expressed by arithematic circuits consisting of only addition and multiplication gates. Therefore, we only to implement our FHE operations for addition and Multiplication. We can recursively compute every gate in the Arthematic circuit homomorphically to get the output of the function.*

DEFINITION 11.2. An arithmetic circuit over a field Z_q is a directed acyclic graph whose vertices are called gates. Gates of incoming degree 0 are inputs to the circuit. All other gates are labelled $+$ or x.

We usually consider Arithematic circuits of fan-in 2 circuits, in which case all of the $+$ and gates have in-degree 2.

REMARK 11.3. Even though Fully Homomorphic encryption scheme is our actual goal, in practice we also consider a simplification leveled fully homomorphic encryption scheme. Leveled FHE does not allow us to compute aribritary functions f but only functions with apriory known depth d. Informally, when we already know what is the most complex(in terms of depth of the arithematic circuit) and use that in the construction of our FHE

Let us try to the simplest possible way to build a FHE for the addition operation. For simplicity, let us consider that we want to single bit numbers and output a single bit number. This is equivalent to implementing the XOR operation.

11.2 FHE: Addition operation

Consider a encryption of message μ_1 under the public key $(s^T A + e^T, A)$. We call $s^T A + e^T$ as b. One intuitive naive way to implementing addition might be addition of ciphertexts

$$c_1 = (Ar_1, br_1 + \mu_1 \left\lfloor \frac{q}{2} \right\rfloor)$$

$$c_1 = (Ar_2, br_2 + \mu_2 \left\lfloor \frac{q}{2} \right\rfloor)$$

$$c_{add} = c_1 + c_2 = (A(r_1 + r_2), b(r_1 + r_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2} \right\rfloor)$$

It is possible to extend this to multi-bit XOR outputs by simply repeating the circuit multiple times. However, it would only help in computing XOR for two k bit numbers. Let us try to decrypt the ciphertext c_{add} and check what it decrypts to:

So applying the decryption algorithm we get $\mu_1 + \mu_2$ given the total error is small $(|e_1 + e_2| \leq q/4)$. The important observation to note here is to perform addition on two ciphertexts we need to assume hardness of LWE for stronger security parameters. Therefore, if we want to perform l addition operations, we would have to keep our $max(e_i) \leq \left\lfloor \frac{q}{2} \right\rfloor / l$.

Corollary 11.4. *To compute addition of k bit numbers μ_1 and μ_2 we must change our encryption scheme to the where the factor which is multiplied to the plaintext should be $\frac{q}{2^(k+1)}$.*

$$c_1 = (Ar_1, br_1 + \mu_1 \left\lfloor \frac{q}{2^{k+1}} \right\rfloor)$$

$$c_1 = (Ar_2, br_2 + \mu_2 \left\lfloor \frac{q}{2^{k+1}} \right\rfloor)$$

$$c_{add} = c_1 + c_2 = (A(r_1 + r_2), b(r_1 + r_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2^{k+1}} \right\rfloor)$$

On the basis of the similar argument described above, the error of the equations for 1 addition be constrained by $e_i \leq \frac{q}{2^{k+1}}$.

Remark 11.5. It is also possible to implement a similar addition for the private key ecryption part scheme using LWE. That is, adding two ciphertexts c_1 and c_2 corresponding to μ_1 and μ_2 would also result in encryption of message $\mu_1 + \mu_2$.

Remark 11.6. A natural question which arises from the above discussion is about the similarility XOR operation and addition operations.

11.3 Towards FHE multiplication

11.4 Mathematical environments

For your convenience, the scribe note style file comes with the following mathematical environments predefined: theorem, lemma, corollary, proposition, fact, claim, definition, example, assumption, remark, conjecture, open problem, problem. The environments are illustrated below. Please limit yourself to these environments.

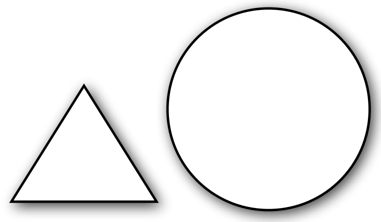

FIGURE 11.1: A triangle and a circle.

THEOREM 11.7. *Statement here*

LEMMA 11.8. *Statement here*

COROLLARY 11.9. *Statement here*

PROPOSITION 11.10. *Statement here*

FACT 11.11. *Statement here*

CLAIM 11.12. *Statement here*

DEFINITION 11.13. Statement here

EXAMPLE 11.14. Statement here

ASSUMPTION 11.15. Statement here

REMARK 11.16. Statement here

CONJECTURE 11.17. Statement here

OPEN PROBLEM 11.18. Statement here

PROBLEM 11.19. Statement here

Note that LaTeX automatically numbers these environments within the lecture number (11 in this case). The same applies to the numbering of pages (this page being page 11-3), figures (Figure **??** above), and equations:

$$a = a_1 + a_2 + \cdots + a_n. \tag{11.1}$$

For proofs, use the provided `proof` environment, illustrated below.

Proof. Proof goes here. $\square$

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.