

Fully Homomorphic Encryption I

Informally, a cryptosystem that supports arbitrary computation on ciphertexts is known as fully homomorphic encryption (FHE)[1]. Throughout the course, we have studied encryption schemes for the purpose of information transfer from other party to the other. However, if we were to compute on data using our traditional encryption schemes, it would require that data must be decrypted before it can be analyzed or manipulated. It would be great if we can outsource the computation while keeping the data encrypted. That is, it should be possible to compute a known function on encrypted data without having access to the secret key.

Such a FHE scheme can be used for privacy-preserving outsourced storage and computation. In today's lecture, we will see what is fully homomorphic encryption (FHE) scheme and how to build a FHE scheme achieving homomorphism. We will also look at intuitive ways to construct FHE for addition and multiplication.

FACT 11.1. *LWE is the only way we know to do an fully homomorphic encryption as of Oct 2019.*

11.1 Recap

We provide a quick primer on the notation used in this section.

- All values which are bolded like \mathbf{s}, \mathbf{a} are vectors where bolded \mathbf{A} is a matrix.
- All values which in simple like q, e are scalars. All scalars apart from field modulus q are in field Z_q .
- The values \mathbf{a}, \mathbf{A} are public value seen by all parties, \mathbf{s} is secret value and e, \mathbf{e} are errors

Before stepping into how to build LWE-based FHE schemes, let's briefly recap how to build private and public encryption scheme with LWE problem. Decisional LWE is the problem where given a matrix $A_{n,m}$, and a corresponding LWE vector result b_m , determining whether or not these are results vector of some instance of an LWE problem OR whether the vector b_m was simply drawn from sampling values uniformly randomly from Z_q^m .

DEFINITION 11.2. **Decisional** $LWEn, m, q, \mathcal{X}$: For all non-uniform probabilistic polynomial time adversary \mathcal{A}

$$\left| \Pr_{\substack{\mathbf{s} \leftarrow \mathbb{Z}_q^{n \times 1} \\ \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{e} \leftarrow \mathcal{X}^m}} [\mathcal{A}(\mathbf{A}, \mathbf{s}^T \mathbf{A} + \mathbf{e}^T) = 1] - \Pr_{\substack{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{b} \leftarrow \mathbb{Z}_q^m}} [\mathcal{A}(\mathbf{A}, \mathbf{b}) = 1] \right| = \text{negl}(n)$$

where q is a prime within $O(2^n)$, $m = O(n \log q)$ and norm $\|\mathbf{e}\| = \omega(\log n)$. Where $\omega(f(n))$ means that e should be greater than $f(n)$ asymptotically.

Next we revise the secret key encryption (SKE) built with LWE which has $m = 1$ and secret key \mathbf{s} . We refer to previous lecture for definition of SKE and only give an instantiation of one using LWE.

$$KeyGen(1^n) : \mathbf{s} \leftarrow \mathbb{Z}_q^n$$

$$Enc_{\substack{\mathbf{s} \leftarrow \mathbb{Z}_q^{n \times 1} \\ \mathbf{a} \leftarrow \mathbb{Z}_q^{n \times 1} \\ \mathbf{e} \leftarrow \mathcal{X}}}(\mathbf{s}, \mu \in \{0, 1\}) : (\mathbf{a}, (b = \mathbf{s}^T \mathbf{a} + e + \mu \lfloor \frac{q}{2} \rfloor)) \mod q$$

$$Dec(\mathbf{s}, \mathbf{a}, b) : b - \langle \mathbf{s}^T, \mathbf{a} \rangle = (e + \mu \lfloor \frac{q}{2} \rfloor) \mod q = \begin{cases} 0 & \text{if } b - \mathbf{s}^T \mathbf{a} \mod q \in [\frac{-q}{4}, \frac{q}{4}] \\ 1 & \text{if } b - \mathbf{s}^T \mathbf{a} \mod q \in [\frac{q}{4}, \frac{3q}{4}] \end{cases}$$

LWE can also be used to build public key encryption(PKE) with public key $pk = (\mathbf{A}, \mathbf{b}^T = \mathbf{s}^T \mathbf{A} + \mathbf{e}^T)$ and secret key sk . Just as before, interested reader can look at the definition of PKE in and here we only an instantiation of the scheme using LWE.

$$KeyGen(1^n) : (sk = \mathbf{s}, pk = (\mathbf{A}, \mathbf{b}^T = \mathbf{s}^T \mathbf{A} + \mathbf{e}^T))$$

$$\substack{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{a} \leftarrow \mathbb{Z}_q^{n \times 1} \\ \mathbf{e} \leftarrow \mathcal{X}^m}$$

$$Enc_{\mathbf{r} \leftarrow \{0,1\}^m}(pk = (\mathbf{A}, \mathbf{b}^T), \mu \in \{0, 1\}) : (\mathbf{c}_1 = \mathbf{A} \mathbf{r}, c_2 = (\mathbf{b}^T \mathbf{r} + \mu \lfloor \frac{q}{2} \rfloor)) \mod q$$

$$Dec(sk = \mathbf{s}, (\mathbf{c}_1, c_2)) : c_2 - \mathbf{s}^T \mathbf{c}_1 = \mathbf{e}^T \mathbf{r} + \mu \lfloor \frac{q}{2} \rfloor \mod q = \begin{cases} 0 & \text{if } c_2 - \mathbf{s}^T \mathbf{c}_1 \mod q \in [\frac{-q}{4}, \frac{q}{4}] \\ 1 & \text{if } c_2 - \mathbf{s}^T \mathbf{c}_1 \mod q \in [\frac{q}{4}, \frac{3q}{4}] \end{cases}$$

REMARK 11.3. Generally, we want the noise \mathbf{e} introduced to the equations are sampled from a distribution with zero mean and low standard variation. For the correctness of the encryption scheme, we will require that the noise distribution makes noise bound $\|\mathbf{e}\| \leq q/4$ with high probability. As shown in above decryption scheme, if the norm of noises is bounded by $q/4$, the boundary between the case $\mu = 1$ and $\mu = 0$ is very clear and the probability that decryption algorithm giving the wrong plaintext is very slow. In practice, \mathcal{X} is usually a discrete Gaussian distribution over \mathbb{Z}_q . With Gaussian distribution that has mean $\mu = 0$ and standard variation σ to be very small, according to its probability density function:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

we can get that the error bound go beyond $q/4$ is negligible when $\sigma \ll q/4$.

REMARK 11.4. We also recall that parameters e, \mathbf{e} in both of the schemes above are neither in the public key nor in the secret key! That is neither the Encryption algorithms or the Decryption algorithms use it. But, it is also important for the security of the above schemes that errors are not known to other parties even though it is not a part of secret key.

11.2 Fully Homomorphic Encryption (FHE)

Let us consider the scenario shown in 11.1. A client has a secret value x . The client wants the server do some computation on x without revealing what x is. Firstly, a ciphertext $ct = \text{Enc}(x)$ is sent to the server along with the desired function f . Then the server could compute a new ciphertext $ct^* = \text{Enc}(f(x))$ by evaluating x on another function g which is publicly computable from f . After receiving ct^* from the server, the client can use its secret key sk to get the desired result of $f(x)$. Note that in the following example, the server learns the function f so it is aware of what computation is taking place, but cannot figure the values underneath.

Fully Homomorphic Encryption refers to an extension of Encryption scheme with additional keywords **Fully** and **Homomorphic**. **Homomorphic** in FHE refers to homomorphism in mathematics: the encryption and decryption functions can be thought of as homomorphisms between plaintext and ciphertext domains. **Fully** refers to the fact that we can evaluate any function. There is another variant called **Levelled** Homomorphic encryptions where we restrict ourselves to evaluating only functions with certain complexity(depth).

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 arbitrary functions f but only functions with a priori known depth d . Informally, when we already know what is the most complex(in terms of depth of the arithmetic circuit) and use that in the construction of our FHE scheme.

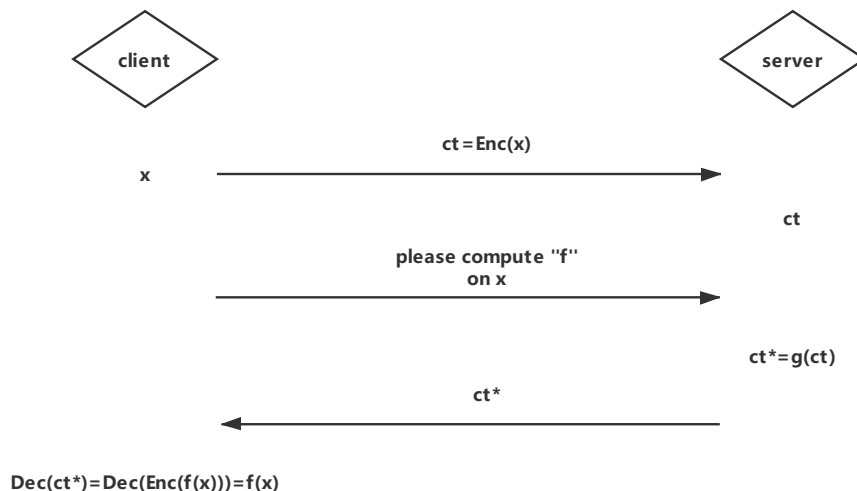


FIGURE 11.1: Outsourced Computation

A homomorphic encryption can be used for privacy-preserving outsourced storage and computation. It allows operations and analysis on encrypted data without revealing the original one, which removes the privacy barriers in several real-life applications.

DEFINITION 11.5. Let \mathcal{C} be a class of circuits where for each $f \in \mathcal{C}$, $f : \{0, 1\}^n \rightarrow \{0, 1\}$. An encryption scheme $(KeyGen, Enc, Dec, Eval)$ is **\mathcal{C} -homomorphic** if $\forall f \in \mathcal{C}$, all ciphertexts ct_1, \dots, ct_n , $Eval(f, ct_1, \dots, ct_n) = ct^*$ such that if $\forall i, \exists m_i, r_i$ s.t. $ct_i = Enc(m_i; r_i)$, then $Dec_{sk}(ct^*) = f(m_1, \dots, m_n)$ and the scheme is IND-CPA secure.

At a high level, given ciphertexts ct_1, \dots, ct_n that encrypt m_1, \dots, m_n , FHE should allow anyone to output a ciphertext ct^* that encrypts $f(m_1, \dots, m_n)$ for any desired function f by evaluating another function g which is publicly computable from f . Thus, the key holder could use the secret key sk to decrypt ct^* and get the result of $f(m_1, \dots, m_n)$.

Note that each function $f : \{0, 1\}^n \rightarrow \{0, 1\}^k$ can be split into f_1, \dots, f_k where $\forall i, f_i : \{0, 1\}^n \rightarrow \{0, 1\}$ and also we can generalize the definition by regulating the input length of circuits in \mathcal{C} from n to $poly(n)$.

REMARK 11.6. In the Leveled homomorphic setting, it is possible to hide the function f being evaluated under a Universal function evaluator. That is if we know the bound of the number of gates and depth of the circuits being evaluated, we can use the function f as a argument to another universal function evaluator.

$$Eval(U, f, args) = f(args)$$

11.3 Construction of Fully Homomorphic Encryption:

As described previously, FHE is an encryption scheme $(KeyGen, Enc, Dec)$ 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 a daunting task, but can use the following fact to ease our task.

FACT 11.7. *It turns out that all functions can be expressed by arithmetic circuits consisting of only addition and multiplication gates. Therefore, we only implement our FHE operations for addition and Multiplication. We can recursively compute every gate in the arithmetic circuit homomorphically to get the output of the function.*

DEFINITION 11.8. An arithmetic circuit over a field \mathbb{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 arithmetic circuits with fan-in 2, in which case all of the $+$ and x gates have in-degree 2.

Let us start with 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.

11.4 FHE: Addition Operation

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

$$\begin{aligned} c_1 &= (\mathbf{A}\mathbf{r}_1, \mathbf{b}\mathbf{r}_1 + \mu_1 \left\lfloor \frac{q}{2} \right\rfloor) \\ c_2 &= (\mathbf{A}\mathbf{r}_2, \mathbf{b}\mathbf{r}_2 + \mu_2 \left\lfloor \frac{q}{2} \right\rfloor) \\ c_{add} &= c_1 + c_2 = (\mathbf{A}(\mathbf{r}_1 + \mathbf{r}_2), \mathbf{b}(\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2} \right\rfloor) \end{aligned}$$

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:

$$\begin{aligned} Dec(sk, c_{add}) &= \mathbf{b}(\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2} \right\rfloor - \mathbf{s}^T \mathbf{A}(\mathbf{r}_1 + \mathbf{r}_2) \\ &= (\mathbf{s}^T \mathbf{A} + \mathbf{e}^T)(\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2} \right\rfloor - \mathbf{s}^T \mathbf{A}(\mathbf{r}_1 + \mathbf{r}_2) \\ &= \mathbf{e}^T(\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2} \right\rfloor \end{aligned}$$

So applying the decryption algorithm we get $\mu_1 + \mu_2$ given the total error is small $|e_1 + e_2| \leq q/4$ where $e_i = \|\mathbf{e}^T \mathbf{r}_i\|$. 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 $\|\mathbf{e}^T \mathbf{r}_i\| \leq \left\lfloor \frac{q}{2} \right\rfloor / l$. The image shows the decryption algorithm works. With a very high probability an decryption of an Encryption of 0 would be in the blue color zone and the decryption of 1 will be in the orange zone.

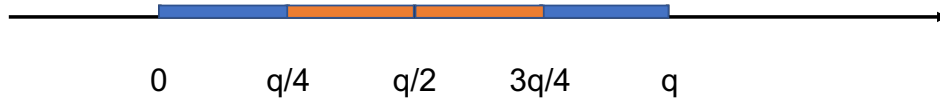


FIGURE 11.2: Decryption Error

COROLLARY 11.9. *To compute addition (instead of XOR) of k bit numbers μ_1 and μ_2 , we must modify the encryption scheme by changing the factor which is multiplied with the plaintext from $\frac{q}{2}$ to $\frac{q}{2^{k+1}}$.*

$$\begin{aligned} c_1 &= (\mathbf{A}\mathbf{r}_1, \mathbf{b}\mathbf{r}_1 + \mu_1 \left\lfloor \frac{q}{2^{k+1}} \right\rfloor) \\ c_2 &= (\mathbf{A}\mathbf{r}_2, \mathbf{b}\mathbf{r}_2 + \mu_2 \left\lfloor \frac{q}{2^{k+1}} \right\rfloor) \\ c_{add} &= c_1 + c_2 = (\mathbf{A}(\mathbf{r}_1 + \mathbf{r}_2), \mathbf{b}(\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2) \left\lfloor \frac{q}{2^{k+1}} \right\rfloor) \end{aligned}$$

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 11.10. It is also possible to implement a similar addition for the private key encryption 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 $\mu_1 + \mu_2$.

11.5 Towards FHE multiplication:

Let us try to apply to

THEOREM 11.11. *Statement here*

LEMMA 11.12. *Statement here*

COROLLARY 11.13. *Statement here*

PROPOSITION 11.14. *Statement here*

FACT 11.15. *Statement here*

CLAIM 11.16. *Statement here*

DEFINITION 11.17. Statement here

EXAMPLE 11.18. Statement here

ASSUMPTION 11.19. Statement here

REMARK 11.20. Statement here

CONJECTURE 11.21. Statement here

OPEN PROBLEM 11.22. Statement here

PROBLEM 11.23. Statement here

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

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

Proof. Proof goes here. □

Acknowledgement

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

References

- [1] Wikipedia. *Fully Homomorphic Encryption*. Cambridge University Press, 2nd edition, 2006.