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

Instructor: Dakshita Khurana
Scribe: Sarah Christensen, Siheng Pan,
Nerla Jean-Louis
Date: October 11, 2019

LECTURE
12

Fully Homomorphic Encryption II

In the prior lecture, we were first introduced to the concept of fully homomorphic encryption (FHE). In particular, we covered the motivation for the problem, its formal definition, and how FHE schemes can leverage learning with errors (LWE) in their construction. In this lecture, we continued our discussion of FHE, diving more deeply into the details of *how* to construct a leveled-FHE using LWE and *why* this construction is correct.

The key technical difficulty focused on in this lecture is ensuring that the leveled-FHE construction is multiplicatively homomorphic, which requires significantly more care than its additive analog. Indeed, the intuition behind the construction only becomes apparent in hindsight, after understanding how it behaves under multiplication. The lecture culminated with a discussion of how leveled-FHE, which allows only a bounded number of homomorphic operations, can be combined with an additional security assumption to produce FHE with an unbounded number of homomorphic operations [1]. This result, called bootstrapping, is one of the most exciting breakthroughs in recent cryptography.

12.1 A leveled-FHE scheme

We begin by describing a leveled fully homomorphic public-key encryption scheme that utilizes LWE (for a review on LWE, see Lecture 10). Recall that any encryption scheme is defined by a tuple of probabilistic polynomial-time algorithms (*KeyGen*, *Encrypt*, *Decrypt*).

KeyGen(prime number q, 1^n):

- Construct an $n \times m$ matrix $\mathbf{A}$ by drawing its entries uniformly at random from the space of integers modulo q (i.e., $\mathbf{A} \longleftarrow \mathbb{Z}_q^{n \times m}$).
- Construct an n-dimensional vector $\mathbf{s}$ by also drawing its entries uniformly at random from the space of integers modulo q (i.e., $\mathbf{s} \longleftarrow \mathbb{Z}_q^n$).
- Construct an m-dimensional vector $\mathbf{e}$ by drawing its entries from the χ distribution (i.e., $\mathbf{e} \longleftarrow \chi^m$). With high probability, $||\mathbf{e}|| < q/(mc)$ for some sufficiently large constant c by the properties of the χ distribution.

– Define an m-dimensional vector $\mathbf{b} = \mathbf{sA} + \mathbf{e}^T$. Notice that these entries are simply the sums of the LWE system of equations.

– Define the secret key to be the following $(n+1)$-dimensional vector: $\mathbf{t} = \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}$

– Define the public key to be the following $(n+1) \times m$ dimensional matrix: $\mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \mathbf{b} \end{bmatrix}$

– Output the public key $\mathbf{B}$.

Encrypt(public key $\mathbf{B}$, message $\mu \in \{0,1\}$):

– Construct an $m \times m$ matrix $\mathbf{R}$ by drawing its entries uniformly at random from $\{0,1\}$. (i.e., $\mathbf{R} \longleftarrow \{0,1\}^{m \times m}$).

– Define a linear operator G using matrix multiplication. In particular, G can be represented with an $(n+1) \times \Big((n+1)(\lfloor \log q \rfloor + 1)\Big)$ matrix as follows:

$$\mathbf{G} := \begin{bmatrix} 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} & 0 & \dots & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & \dots & 2^{\lfloor \log q \rfloor} & \dots & 0 \\ \vdots & \vdots & & \vdots & 0 & \dots & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & 2^{\lfloor \log q \rfloor} \end{bmatrix}$$

In other words, $\mathbf{G}$ is a matrix with the sequence $[1, 2, ..., 2^{\lfloor \log q \rfloor}]$ repeated in each row; in the ith row, this pattern begins in column $j = i \cdot (\lfloor \log q \rfloor + 1)$. We state the entries of $\mathbf{G}$ here for completeness, but the intuition behind it will only make sense in hindsight when proving this scheme is multiplicatively homomorphic.

– Define the ciphertext to be the following:

$$\mathbf{C} = \mathbf{BR} + \mu \mathsf{G}.$$

Note that $\mathbf{BR}$ has dimension $(n+1) \times m$. For this matrix multiplication to make sense, we need to set $m = \Big((n+1)(\lfloor \log q \rfloor + 1)\Big)$.

– Output the ciphertext $\mathbf{C}$.

Decrypt(secret key $\mathbf{t}$, ciphertext $\mathbf{C}$):

– Compute the matrix product of the secret key with the ciphertext.

$$\begin{aligned} \mathbf{t}^T\mathbf{C} &= \mathbf{t}^T(\mathbf{BR} + \mu\mathsf{G}) \\ &= \mathbf{t}^T\mathbf{BR} + \mu\mathbf{t}^T\mathbf{G} \\ &= \begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{sA}+\mathbf{e}^T \end{bmatrix} \mathbf{R} + \mu\mathbf{t}^T\mathbf{G} \\ &= \mathbf{e}^T\mathbf{R} + \mu\mathbf{t}^T\mathbf{G} \\ &= \text{Low norm error term} + \mu\mathbf{t}^T\mathbf{G} \qquad (12.1) \\ &= \begin{cases} \text{Low norm error term} & \mu = 0 \\ \text{Low norm error term} + \mathbf{t}^T\mathbf{G} & \mu = 1 \end{cases} \end{aligned}$$

– The $1 \times m$ vector $\mathbf{t}^T\mathbf{G} = [-s_1 \cdot 1, -s_1 \cdot 2, \ldots, -s_1 \cdot 2^{\lfloor \log q \rfloor}, -s_2 \cdot 1, \ldots, 1 \cdot 2^{\lfloor \log q \rfloor}]$ is then sufficiently big enough to create separation and allow us to reliably distinguish between the case where $\mu = 0$ and $\mu = 1$. To see this, recall that $||\mathbf{e}^T\mathbf{R}|| < m||\mathbf{e}^T|| < q/c$ with high probability by construction. Moreover, $||\mathbf{t}^T\mathbf{G}|| > 2q/c$ with high probability since the secret key will likely contain at least one "big" coefficient close to q. Note that this matrix norm is serving a similar function as $q/2$ did in previous constructions (i.e., creating sufficient separation).

12.2 Verifying Additive Homomorphism

In order for this scheme to be fully homomorphic, it is sufficient to show we can homomorphically perform the operations of addition and subtraction. Note that we cannot make two ciphertexts insecure by adding or multiplying them (recall our proof of security in the multi-message setting) so we focus on correctness exclusively. We begin with addition, which we already showed in the prior lecture can be achieved with a much simpler encryption scheme. Here, we use a similar approach to show the above scheme is additively homomorphic.

Assume that we are given two ciphertexts $\mathbf{C}$ and $\mathbf{C}'$ encrypting messages μ_1 and μ_2, respectively. We wish to show that we can compute the ciphertext encoding $\mu_1 + \mu_2 \mod q$.

Evaluate(+, ciphertext $\mathbf{C}$, ciphertext $\mathbf{C}'$):

– Return $\mathbf{C}_{Add} = \mathbf{C} + \mathbf{C}'$.

What remains to be shown is that *Decrypt*(secret key $\mathbf{t}$, ciphertext $\mathbf{C}_{Add}$) is correct. To do this, we chase through the calculations of our decryption protocol.

$$\begin{aligned}
\mathbf{t}^T\mathbf{C}_{Add} &= \mathbf{t}^T(\mathbf{C} + \mathbf{C}') \\
&= \mathbf{t}^T((\mathbf{B}\mathbf{R}_1 + \mu_1\mathsf{G}) + (\mathbf{B}\mathbf{R}_2 + \mu_2\mathsf{G})) \\
&= \mathbf{t}^T\mathbf{B}(\mathbf{R}_1 + \mathbf{R}_2) + (\mu_1 + \mu_2)\mathbf{t}^T\mathbf{G} \\
&= \mathbf{e}^T(\mathbf{R}_1 + \mathbf{R}_2) + (\mu_1 + \mu_2)\mathbf{t}^T\mathbf{G}
\end{aligned}$$

Observe that we have recovered the same functional form as that above in Equation 12.1. However, the error has likely grown since in the first term $\mathbf{e}^T$ is now multiplied with $(\mathbf{R}_1 + \mathbf{R}_2)$, which has entries in $\{0, 2\}$. The analysis will still go through since the norm is still bounded: $||\mathbf{e}^T(\mathbf{R}_1 + \mathbf{R}_2)|| < 2m||\mathbf{e}^T|| < 2q/c$. It is important though that our choice of c be large enough to tolerate this increase by a factor of 2.

This raises an important point. Under a leveled-FHE scheme such as this one, error will accumulate as more operations are performed. For ℓ additions, the error is proportional to $(\mathbf{R}_1 + ... + \mathbf{R}_\ell) \in \{0, \ell\}^{m \times m}$. This implies that the upper bound on the norm of $\mathbf{e}^T$ fixes an upper bound on the number of computations that can be performed before the decryption algorithm becomes unreliable. This is a major limitation since the number of computations may not be known in advance, reducing flexibility.

12.3 Verifying Multiplicative Homomorphism

We now wish to show that our scheme is multiplicatively homomorphic. Again, assume that we are given two ciphertexts $\mathbf{C}$ and $\mathbf{C}'$ encrypting messages μ_1 and μ_2, respectively. This

time we wish to show that we can compute the ciphertext encoding $\mu_1\mu_2 \mod q$. The most natural thing to try for an evaluation algorithm is the following (ignoring dimension issues):

Evaluate$_{Naive}(\times$, ciphertext $\mathbf{C}$, ciphertext $\mathbf{C}')$:

– Return $\mathbf{C}_{Mult} = \mathbf{C}\mathbf{C}'$.

Unfortunately, when we simplify $\mathbf{C}_{Mult}$ in our decryption step, we run into some difficulties.

$$\begin{aligned}\mathbf{t}^T\mathbf{C}_{Mult} &= \mathbf{t}^T\mathbf{C}\mathbf{C}' \\ &= \mathbf{t}^T(\mathbf{B}\mathbf{R}_1 + \mu_1\mathsf{G})(\mathbf{B}\mathbf{R}_2 + \mu_2\mathsf{G}) \\ &= \mathbf{t}^T\mathbf{B}\mathbf{R}_1\mathbf{B}\mathbf{R}_2 + \mathbf{t}^T\mathbf{B}\mathbf{R}_1\mu_2\mathsf{G} + \mathbf{t}^T\mathbf{B}\mathbf{R}_2\mu_1\mathsf{G} + \mathbf{t}^T\mu_1\mu_2\mathsf{G}^2 \\ &= \mathbf{e}^T\mathbf{R}_1\mathbf{B}\mathbf{R}_2 + \mathbf{e}^T\mathbf{R}_1\mu_2\mathsf{G} + \mathbf{e}^T\mathbf{R}_2\mu_1\mathsf{G} + \mu_1\mu_2\mathbf{t}^T\mathsf{G}^2\end{aligned}$$

In addition to the obvious dimension problems, we obtain the term $\mu_1\mu_2\mathbf{t}^T\mathsf{G}^2$, which does not fit the functional form of Equation 12.1. Also, only one of the $\mathbf{B}$s was reduced to $\mathbf{e}^T$ in the first term. This leaves an error term multiplied by the remaining $\mathbf{B}$, which will likely be too large since entries in $\mathbf{B}$ are relatively large.

We modify this evaluation scheme by introducing the operator G^{-1}, which functions as the inverse to G. The intuition behind doing this is that it will cancel an extra G from the final term as well as shrink the entries in the extra $\mathbf{B}$ of the first term to control error.

Evaluate$(\times$, ciphertext $\mathbf{C}$, ciphertext $\mathbf{C}')$:

– We first define the operator G^{-1}, which takes as input a $\mathbb{Z}_q^{(n+1)\times m}$ matrix and outputs a $\{0,1\}^{((n+1)(\lfloor\log q\rfloor+1))\times m}$ matrix. In particular, G^{-1} expands an input matrix into its component-wise binary decomposition. For instance, given an input matrix

$$\mathbf{X} := \begin{bmatrix} x_{0,0} & x_{0,1} & \dots & x_{0,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,0} & x_{n,1} & \dots & x_{n,m-1} \end{bmatrix}$$

We define G^{-1} on that matrix as follows:

$$\mathsf{G}^{-1}(\mathbf{X}) := \begin{bmatrix} Bin(x_{0,0})_0 & Bin(x_{0,1})_0 & \dots & Bin(x_{0,m-1})_0 \\ Bin(x_{0,0})_1 & Bin(x_{0,1})_1 & \dots & Bin(x_{0,m-1})_1 \\ \vdots & \vdots & \dots & \vdots \\ Bin(x_{0,0})_{(\lfloor\log q\rfloor)} & Bin(x_{0,1})_{(\lfloor\log q\rfloor)} & \dots & Bin(x_{0,m-1})_{(\lfloor\log q\rfloor)} \\ \vdots & \vdots & \ddots & \vdots \\ Bin(x_{n,0})_{(\lfloor\log q\rfloor)} & Bin(x_{n,1})_{(\lfloor\log q\rfloor)} & \dots & Bin(x_{n,m-1})_{(\lfloor\log q\rfloor)} \end{bmatrix}$$

where $Bin(x_{0,0})_0$ is the first bit in the binary decomposition of $x_{0,0} \in \mathbb{Z}_q$ into $((n+1)(\lfloor\log q\rfloor+1))$ bits. Note that the number of columns does not change. Each element in the input matrix gets expanded into $(\lfloor\log q\rfloor+1)$ rows, so the number of rows in the output matrix increases by a factor of $(\lfloor\log q\rfloor+1)$.

– Return the $(n+1)\times m$ dimensional matrix $\mathbf{C}_{Mult} = \mathbf{C}\mathsf{G}^{-1}(\mathbf{C}')$. Recall that we fixed $m = (n+1)(\lfloor\log q\rfloor+1)$ when defining our encryption scheme so the matrix multiplication here is well-defined.

$$\mathsf{G}^{-1}(\mathbf{G}) = \begin{bmatrix} \mathbf{I} & 0 & \dots & & & \dots & 0 \\ & 0 & \dots & & & & \vdots \\ 0 \quad 0 & & & & & & \\ \vdots \quad \vdots & & \mathbf{I} & 0 & \dots & & \\ & & & 0 & \dots & & \\ & & 0 \quad 0 & & & & \\ & & \vdots \quad \vdots & & \ddots & & \vdots \\ & & & & & & 0 \\ \vdots & & & & & & \mathbf{I} \\ 0 & \dots & & & \dots & 0 & \end{bmatrix}$$

FIGURE 12.1: Each identity sub-matrix $\mathbf{I}$ corresponds to the expansion of the sequence $[1, 2, ..., 2^{\lfloor \log q \rfloor}]$ for one row in $\mathbf{G}$. Each $\mathbf{I}$ is of dimension $(\lfloor \log q \rfloor + 1) \times (\lfloor \log q \rfloor + 1)$.

REMARK 12.1. G and G^{-1} are both operators. G performs binary recomposition, and G^{-1} performs binary decomposition. While G is a linear operator (whose operations can therefore be encoded in a matrix), G^{-1} is *not* a linear operator.

REMARK 12.2. G^{-1} takes as input a small matrix with large entries and creates a large matrix with small entries. How convenient!

FACT 12.3. *For any matrix* $\mathbf{X} \in \mathbb{Z}_q^{(n+1)\times m}$, $\mathsf{G}(\mathsf{G}^{-1}(\mathbf{X})) = \mathbf{X}$. *Similarly,* $\mathsf{G}^{-1}(\mathbf{G})$ *is equal to the identity matrix* $\mathbf{I}_{m\times m}$.

To convince yourself of the second half of this fact, recall that each row in the matrix representation of $\mathbf{G}$ has the sequence $[1, 2, ..., 2^{\lfloor \log q \rfloor}]$. When expanded into a component-wise binary decomposition, this sequence is just the identity matrix $\mathbf{I}_{(\lfloor \log q \rfloor+1)\times(\lfloor \log q \rfloor+1)}$. Next, observe that the spacing of the sequences in $\mathbf{G}$ is such that these identity sub-matrices line up along the diagonal of the output matrix, forming a full identity matrix as claimed (see Figure 12.3).

Now, we finally show that with this modified evaluation function, our decryption scheme returns the correct result.

$$\begin{aligned} \mathbf{C}_{Mult} &= \mathbf{C}\mathsf{G}^{-1}(\mathbf{C}') \\ &= (\mathbf{B}\mathbf{R}_1 + \mu_1\mathsf{G})\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2 + \mu_2\mathsf{G}) \\ &= (\mathbf{B}\mathbf{R}_1 + \mu_1\mathsf{G})(\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathsf{G}^{-1}(\mathsf{G})\mu_2) \\ &= \mathbf{B}\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{B}\mathbf{R}_1\mu_2 + \mu_1\mathbf{B}\mathbf{R}_2 + \mu_1\mu_2\mathsf{G} \end{aligned}$$

Multiplying by the secret key, we have

$$\begin{aligned} \mathbf{t}^T\mathbf{C}_{Mult} &= \mathbf{t}^T\Big(\mathbf{B}\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{B}\mathbf{R}_1\mu_2 + \mu_1\mathbf{B}\mathbf{R}_2 + \mu_1\mu_2\mathsf{G}\Big) \\ &= \mathbf{t}^T\mathbf{B}\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{t}^T\mathbf{B}\mathbf{R}_1\mu_2 + \mathbf{t}^T\mathbf{B}\mathbf{R}_2\mu_1 + \mathbf{t}^T\mu_1\mu_2\mathsf{G} \\ &= \mathbf{e}^T\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{e}^T\Big(\mathbf{R}_1\mu_2 + \mathbf{R}_2\mu_1\Big) + \mu_1\mu_2\mathbf{t}^T\mathsf{G} \end{aligned}$$

This is very close to the invariant functional form that we are looking for. Note that the final term is exactly in the same form as Equation 12.1. The middle term is bounded above

by $\mathbf{e}^T\Big(\mathbf{R}_1 + \mathbf{R}_2\Big)$, which we already analyzed as being sufficiently small in the additive homomorphism section. What remains is showing $\mathbf{e}^T\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2)$ is sufficiently small.

And finally we understand all the fuss about G and G^{-1}. We have defined G^{-1} precisely so that it can shrink the norm of $\mathbf{B}$! The Frobenious norm of $\mathbf{BR}$ is $O(q\sqrt{nm})$ since there are $O(nm)$ entries in $\mathbf{BR}$ with value of $O(q)$. By comparison, the Frobenious norm of $\mathsf{G}^{-1}(\mathbf{BR})$ is $O(\sqrt{mm}) = O(m) = O(n\log q)$. This is because there are m^2 entries in this binary matrix. We are operating in a regime where $q = O(2^n)$, so this is a significant reduction. Putting this all together to get an upper bound on error we have:

$$\begin{aligned}
||\mathbf{e}^T\Big(\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{R}_1\mu_2 + \mathbf{R}_2\mu_1\Big)||_F &\leq ||\mathbf{e}^T\Big(\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{R}_1 + \mathbf{R}_2\Big)||_F \\
&\leq ||\mathbf{e}^T||_F \cdot ||\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2) + \mathbf{R}_1 + \mathbf{R}_2||_F \\
&\leq ||\mathbf{e}^T||_F \cdot \Big(||\mathbf{R}_1\mathsf{G}^{-1}(\mathbf{B}\mathbf{R}_2)||_F + ||\mathbf{R}_1||_F + ||\mathbf{R}_2||_F\Big) \\
&\leq \frac{q}{mc}(m + 2m) \\
&= \frac{3q}{c}
\end{aligned}$$

Again, we see that the error is bounded and that for sufficiently large c, we can decrypt the message. In fact, the norm of the error grows by roughly m, which limits how many times you can multiply before experiencing decryption error. Since q, n, m are all fixed ahead of time, there is an apriori bounded number of operations. In the next section, we see how to fix the problem with needing to bound the number of operations in advance.

Open Problem 12.4. Construct a FHE without using a variation of LWE.

12.4 Gentry's Bootstrapping

To achieve a FHE with unbounded homomorphism, we introduce the infamous bootstrapping technique first proposed by Gentry [1]. The key to performing an unbounded number of operations is to be able to reduce the accumulation of noise. Given an ciphertext with lots of accumulated noise, we want to obtain a new ciphertext that still encrypts the same message, but with much less noise. The idea for achieving this is as follows: Take a ciphertext with high noise homomorphically decrypt it with an encrypted secret key.

We begin with a high noise ciphertext encrypting some message μ as well as a ciphertext encrypting the secret key (i.e., $\mathbf{C}_{Noise} = Enc(\mu), \mathbf{C}_{Secret} = Enc(\mathbf{t})$). We then define d as the fixed number of operations the decryption algorithm Dec requires on $\mathbf{C}_{Noise}$ and $\mathbf{C}_{Secret}$. We then obtain a new ciphertext $\mathbf{C}_{Denoise}$ as follows:

$$\begin{aligned}
\mathbf{C}_{Denoise} &= Eval(Dec, \mathbf{C}_{Noise}, \mathbf{C}_{Secret}) \\
&= Enc_{pk2}(Dec(\mathbf{C}_{Noise})) \\
&= Enc_{pk}(\mu)
\end{aligned}$$

Notice that the noise $\mathbf{C}_{Denoise}$ is independent of the noise in $\mathbf{C}_{Noise}$, and instead $\mathbf{C}_{Denoise}$ always has noise proportional to d. If we set q to be sufficiently large enough based on an upper bound for d, we can reduce the error of the ciphertext.

Notice that this construction requires us to publish an encryption of the secret key, which not allowed under the standard definition of security we have been using. Recall that under

CPA-security, we have that for all messages μ_0, μ_1 $Enc_{pk}(\mu_0) \approx Enc_{pk}(\mu_1)$. Under something called *Circular-Security*, we have that all messages μ_0, μ_1 $Enc_{p_k}(\mu_0), Enc_{p_k}(sk) \approx Enc_{p_k}(\mu_1), Enc_{p_k}(sk)$. Note that it is not known that CPA-security implies Circular-Security, and that Circular-Security is a stronger assumption. What we have shown is that if we had leveled-FHE with Circular-Security, then we can build a true FHE with unbounded homomorphism.

REMARK 12.5. FHE with bounded homomorphism is often referred to as leveled-FHE. FHE with unbounded homomorphism is often just referred to as FHE.

OPEN PROBLEM 12.6. Construct a FHE with unbounded homomorphism without relying on the additional circular security assumption.

Acknowledgement

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

References

[1] C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.