CS-E4340: Cryptography **I-II 2022/2023**

Lecture 9: Fully Homomorphic Encryption

Lecturer: Russell W.F. Lai

Abstract

This lecture aims to:

- introduce the notion of fully homomorphic encryption (FHE),
- construct FHE from public-key encryption with (almost-)bilinear decryption, and
- construct FHE from the learning with errors (LWE) assumption.

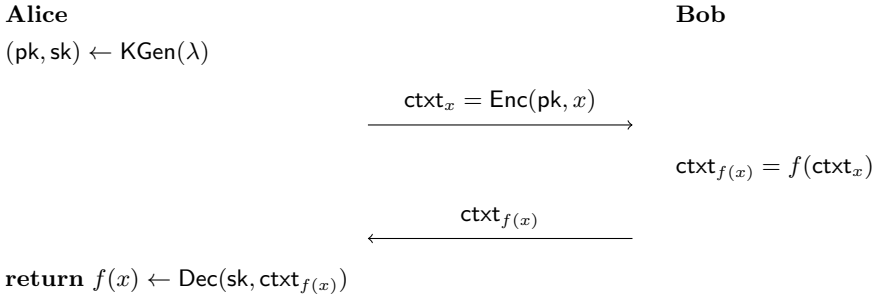
9.1 Background

A fully homomorphic encryption (FHE) is an encryption scheme where arbitrary computation can be performed on the encrypted data without knowing the decryption key. The concept of FHE was first proposed by Rivest, Adleman, Dertouzos [RAD78] in 1978 under the name "privacy homomorphism" but without a plausible candidate construction. Constructing FHE remained a major open problem until 2009, when Gentry [Gen09] proposed a candidate scheme based on ideal lattices. Perhaps more importantly, Gentry introduced the bootstrapping technique, which turns a scheme supporting a bounded number homomorphic operations into an unbounded one. Since then, numerous improvements have been made in both efficiency and security. Open-source libraries have also been made available.[1]

9.2 Why Fully Homomorphic Encryption?

There are many reasons why we may want to compute over encrypted data. Below is a classic example of communication-efficient secure two-party computation:

Suppose Alice has some secret data x and wants to compute $f(x)$ for some complicated public function f. Since doing the computation herself is too costly, she wants to delegate it to Bob. To do this securely, Alice and Bob engage in the following protocol:



Note that in the above protocol:

[1] Since this is a growing list, we refer to https://en.wikipedia.org/wiki/Homomorphic_encryption.

- Bob learns nothing about x since it is encrypted.
- Alice's work only depends on the size of x, but independent of the complexity of f.
- The communication takes only 2 rounds, which is optimal.

9.3 Definitions

A (public-key) FHE scheme is simply a public-key encryption (PKE) scheme with an additional homomorphic evaluation algorithm Eval which allows to homomorphically evaluate arbitrary functions over encrypted messages. Below, we define FHE for the message space $\{0,1\}$.

Definition 9.1 (FHE) *A fully homomorphic encryption (FHE) scheme for the message space $\{0,1\}$ is a tuple of PPT algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ with the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*: The key generation algorithm generates a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.*
- $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$*: The encryption algorithm takes a public key pk and a message $x \in \{0,1\}$ and outputs a ciphertext ctxt.*
- $x \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt})$*: The decryption algorithm takes a secret key sk and a ciphertext ctxt and outputs a message x.*
- $\mathsf{ctxt}' \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)$*: The homomorphic evaluation algorithm takes a public key pk, an n-variate Boolean function $f : \{0,1\}^n \to \{0,1\}$ for some $n \in \mathbb{N}$, and a sequence of ciphertexts $(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)$ and outputs a new ciphertext ctxt'.*

Note that, to construct the Eval algortihm of an FHE scheme, it suffices to describe the procedures for evaluating a functionally complete set of logic gates, e.g. the NAND gate. The homomorphic evaluation of arbitrary Boolean functions can be done by composing the homomorphic evaluations of NAND.

The correctness of an FHE scheme consists of two separate requirements – decryption correctness and evaluation correctness. Decryption correctness basically says that the FHE scheme is correct as a PKE scheme. Evaluation correctness says that if ctxt_i is an encryption of x_i for all i, then $\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)$ is an encryption of $f(x_1, \ldots, x_n)$.

Definition 9.2 (Correctness) *An FHE scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is correct if the following properties are satisfied for any $\lambda \in \mathbb{N}$ and any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$:*

- *Decryption correctness: For any $x \in \{0,1\}$, it holds that*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x)) = x.$$

- *Evaluation correctness[2]: For any $n \in \mathbb{N}$, n-variate Boolean function $f : \{0,1\}^n \to \{0,1\}$, any $x_1, \ldots, x_n \in \{0,1\}$ and ciphertexts $\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n$ such that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_i) = x_i$ for all i, it holds that*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)) = f(x_1, \ldots, x_n).$$

[2]The definition of evaluation correctness considered in this lecture notes is stronger than those usually found in the literature. The former requires evaluation correctness to hold for any ciphertexts in the ciphertext space, including potentially invalid ones, while the latter only requires evaluation correctness to hold for fresh ciphertexts or those resulting from prior homomorphic evaluations.

The basic security notion of FHE schemes is ciphertext indistinguishability under chosen-plaintext attacks (IND-CPA), i.e. same as that for PKE schemes. The definition of IND-CPA-security is exactly the same as that for PKE schemes. Below, we recall the definition.

Definition 9.3 (IND-CPA) *An FHE scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *has ciphertext indistinguishability under chosen-plaintext attacks (IND-CPA-secure) if for any (two-stage) PPT adversary* $\mathcal{A}$ *it holds that*

$$\left|\Pr\left[\text{IND-CPA}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\text{IND-CPA}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where the experiment $\text{IND-CPA}^b_{\Pi,\mathcal{A}}$ *for* $b \in \{0,1\}$ *is defined as follows:*

$\text{IND-CPA}^b_{\Pi,\mathcal{A}}(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$

$(x_0, x_1) \leftarrow \mathcal{A}(\mathsf{pk})$

$\mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b)$

$b' \leftarrow \mathcal{A}(\mathsf{ctxt}^*)$

return b'

Note that the IND-CPA-security of FHE only concerns about the secrecy of encrypted messages. It says nothing about the secrecy of a function f when evaluated on an FHE ciphertext. In some applications, we may want that a ciphertext ctxt', obtained by homomorphically evaluating a function f at a ciphertext ctxt encrypting x, reveals nothing more than the value $f(x)$ even with knowledge of the secret key. Such a property is known as circuit privacy (where the function f is considered to be represented by a circuit). We formalise a variant of this property below.

Definition 9.4 (Circuit Privacy) *An FHE scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *has (semi-honest, statistical) circuit privacy if there exists a PPT simulator* $\mathcal{S}$ *such that for any* $n \in \mathbb{N}$*, any* n*-variate function* f*, and any messages* $x_1, \ldots, x_n \in \{0,1\}$*, the distributions*

$$\left\{ (\mathsf{pk}, \mathsf{sk}, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n, \mathsf{ctxt}') : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ \mathsf{ctxt}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i),\ \forall i \in [n] \\ \mathsf{ctxt}' \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n) \end{array} \right\}$$

and

$$\left\{ (\mathsf{pk}, \mathsf{sk}, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n, \mathsf{ctxt}') : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ \mathsf{ctxt}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i),\ \forall i \in [n] \\ \mathsf{ctxt}' \leftarrow \mathcal{S}(\mathsf{pk}, |f|, f(x_1, \ldots, x_n), \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n) \end{array} \right\}$$

are statistically close in λ*.*

In the above, $|f|$ denotes the (circuit description) size of f. Note that the secrecy of $|f|$ is not required by the above definition.

One could think of different variants of the above definition. For example, the public and secret keys and the input ciphertexts could be adversarially generated. The simulator might only be given one-time oracle access to f instead of the output value $f(x_1, \ldots, x_n)$. The notion of closeness could be computational rather than statistical.

9.4 FHE from Homomorphic Encryption with Linear Decryption

In the invited talk "Fully Homomorphic Encryption from the Ground Up"[3] of Eurocrypt 2019, Daniele Micciancio presented an elementary construction of FHE from a PKE with (almost-)bilinear decryption. The point of the construction is not to build the most efficient or secure scheme possible, but to make it easily accessible using only simple mathematics, i.e. linear algebra. This section is a remix of Daniele's inspiring talk. We emphasise that the FHE built in this section is not entirely correct – there are issues regarding how messages are encoded in ciphertexts and containing evaluations of them in the message space – but it serves to give an intuition towards FHE constructions.

9.4.1 Warm Up: The Noise-Free Case

Consider a hypothetical public-key encryption scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for the message space $\mathbb{Z}_q$ with bilinear decryption. By bilinear decryption, we mean that $\mathsf{Dec}(\cdot,\cdot)$ is a bilinear function, i.e. for any secret keys $\mathsf{sk}, \mathsf{sk}'$, ciphertexts $\mathsf{ctxt}, \mathsf{ctxt}'$, and scalar $a \in \mathbb{Z}_q$, it holds that

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}', \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk} + \mathsf{sk}', \mathsf{ctxt})$,
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}') = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt} + \mathsf{ctxt}')$, and
- $a \cdot \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(a \cdot \mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk}, a \cdot \mathsf{ctxt})$.

The second and the third item imply that Π is linearly homomorphic. As a corollary, for any linear function L over $\mathbb{Z}_q$, secret keys sk, $(\mathsf{sk}_i)_{i=1}^k$, and ciphertexts ctxt, $(\mathsf{ctxt}_i)_{i=1}^k$, the following properties also hold:

- Key-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ctxt}), \ldots, \mathsf{Dec}(\mathsf{sk}_k, \mathsf{ctxt})) = \mathsf{Dec}(L(\mathsf{sk}_1, \ldots, \mathsf{sk}_k), \mathsf{ctxt})$
- Ciphertext-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_1), \ldots, \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_k)) = \mathsf{Dec}(\mathsf{sk}, L(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_k))$

Note that such a scheme cannot be secure even in the weakest sense, as there exists a simple key-recovery attack given only the public key pk:

- Using pk, generate encryptions $\mathsf{ctxt}_0, \mathsf{ctxt}_1, \ldots$ of a few arbitrary messages $\mathsf{msg}_0, \mathsf{msg}_1, \ldots$.
- By the bilinear decryption property, we know that the system of equations $(\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_i) = \mathsf{msg}_i)_i$ in variable sk is a system of linear equations over $\mathbb{Z}_q$.
- Solve the linear system of equations, e.g. using Gaussian elimination, to recover sk.

Despite the attack, here we focus on what we can do with the linearity of the scheme, and deal with the minor problem of the scheme being completely broken later.

Supporting Multiplication. Our goal is to construct an FHE $\Pi' = (\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ for the message space $\mathbb{Z}_q$, and our choice of functionally complete set of operations is naturally addition $+$ and multiplication $\cdot$ over $\mathbb{Z}_q$. Since the scheme Π is already linearly homomorphic, i.e. it supports addition, we only need to deal with multiplication. Towards this, we observe from the bilinear decryption property that

$$\mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = \mathsf{Dec}(\mathsf{sk}, y \cdot \mathsf{ctxt}_x) = x \cdot y.$$

The above suggests a way of computing $\mathsf{ctxt}_{x \cdot y}$ from ctxt_x and $\mathsf{ctxt}_{y \cdot \mathsf{sk}}$, the latter of which can be computed as $\mathsf{ctxt}_{y \cdot \mathsf{sk}} = y \cdot \mathsf{ctxt}_{\mathsf{sk}}$ given $\mathsf{ctxt}_{\mathsf{sk}}$:

[3] https://youtu.be/TySXpV86958

- Let $L(\cdot)$ be the linear function that computes $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$.
- Output $\mathsf{ctxt}_{L(y\cdot\mathsf{sk})} = L(\mathsf{ctxt}_{y\cdot\mathsf{sk}})$.

Since $L(y \cdot \mathsf{sk}) = \mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = x \cdot y$, by ciphertext-homomorphism we have $\mathsf{ctxt}_{L(y\cdot\mathsf{sk})} = \mathsf{ctxt}_{x\cdot y}$.

Summarising the above, from ctxt_x and $\mathsf{ctxt}_{y\cdot\mathsf{sk}}$, where the latter can be computed given y and $\mathsf{ctxt}_{\mathsf{sk}}$, we can homomorphically compute $\mathsf{ctxt}_{x\cdot y}$ without knowing sk. This allows us to construct an FHE Π' as follows:

- Key Generation: $\mathsf{pk}' = (\mathsf{pk}, \mathsf{ctxt}_{\mathsf{sk}})$, $\mathsf{sk}' = \mathsf{sk}$.
- Encryption: $\mathsf{ctxt}'_x = x \cdot \mathsf{ctxt}_{\mathsf{sk}}$.
- Decrytpion: Compute $x \cdot \mathsf{sk} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}'_x)$, then recover x from $x \cdot \mathsf{sk}$.
- Homomorphic Addition: $\mathsf{ctxt}'_x + \mathsf{ctxt}'_y (= \mathsf{ctxt}_{x\cdot\mathsf{sk}} + \mathsf{ctxt}_{y\cdot\mathsf{sk}} = \mathsf{ctxt}_{(x+y)\cdot\mathsf{sk}} = \mathsf{ctxt}'_{x+y})$.
- Homomorphic Multiplication: Let $L(\cdot) = \mathsf{Dec}(\cdot, \mathsf{ctxt}'_x)$. Output $L(\mathsf{ctxt}'_y)$.

9.4.2 Achieving Security: Adding Noise and Dealing with it

Since Π is completely broken by linear algebra, so does Π'. A natural idea to fix this issue is to add noise to secret keys/ciphertexts and rely on the LWE assumption for security. This brings two consequences:

- Noisy linear decryption: For any ciphertext ctxt_x, $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ is only approximated by a linear function $L(\cdot)$ where $L(\mathsf{sk}) = x + \mathsf{error}$. (Think of Regev's and dual-Regev PKE.)
- Error accumulation: Suppose ctxt_x and ctxt_y has (absolute) error e_x and e_y respectively, then
  - $\mathsf{ctxt}_x + \mathsf{ctxt}_y$ has error $e_x + e_y$, and
  - for scalar $a \in \mathbb{Z}_q$, $a \cdot \mathsf{ctxt}_x$ has error $a \cdot e_x$.

To apply the transformation from Π to Π', we need to tackle two challenges:

- Since $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ is non-linear, it cannot be evaluated homomorphically on $\mathsf{ctxt}_{y\cdot\mathsf{sk}}$.
- Even if we settle for noisy linear decryption $L(\cdot) \approx \mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$, the function L likely has large coefficients which blow up the error.

In the following, we will first discuss how multiplications by large scalars can be supported. This allows us to homomorphically evaluate the noisy linear decryption function $L(\cdot)$ and, using a transformation similar to that from Π to Π', to perform noisy homomorphic multiplication. Second, we will discuss the bootstrapping technique which allows to reduce the error in a ciphertext.

Supporting Large-Scalar Multiplication. To handle multiplication by a large scalar $a \in \mathbb{Z}_q$, the idea is to break a down into its binary representation $a = \sum_{i=0}^{\ell} a_i \cdot 2^i$ where $\ell = \lfloor \log q \rfloor$. Based on this idea, we construct another scheme $\Pi'' = (\mathsf{KGen}'', \mathsf{Enc}'', \mathsf{Dec}'', \mathsf{Eval}'')$ from the scheme Π', which itself is built from Π. Note that now Π and Π' are noisy. In Π'', a message x is encrypted as $\mathsf{ctxt}''_x = (\mathsf{ctxt}'_{x\cdot 2}, \ldots, \mathsf{ctxt}'_{x\cdot 2^\ell})$ where $\ell = \lfloor \log q \rfloor$. To compute $\mathsf{ctxt}'_{a\cdot x}$ given a and ctxt''_x , we can perform the following:

- Write down the binary decomposition $a = \sum_{i=0}^{\ell} a_i \cdot 2^i$.

- Output $\mathsf{ctxt}'_{a\cdot x} = \sum_{i=0}^{\ell} a_i \cdot \mathsf{ctxt}'_{x\cdot 2^i} (= \mathsf{ctxt}'_{\sum_{i=0}^{\ell} a_i \cdot x \cdot 2^i})$.

Note that the error in $\mathsf{ctxt}'_{a\cdot x}$ remains small since $a_i \in \{0,1\}$ for all i. By repeating the above for $a \cdot 2 \bmod q, \ldots, a \cdot 2^\ell \bmod q$, we obtain

$$\mathsf{ctxt}''_{a\cdot x} = (\mathsf{ctxt}'_{a\cdot x\cdot 2}, \ldots, \mathsf{ctxt}'_{a\cdot x\cdot 2^\ell}).$$

If we start with ctxt''_x with error e, (each component of) the resulting ciphertext $\mathsf{ctxt}''_{a\cdot x}$ will have error $e \cdot \log q$.

Supporting Homomorphic Multiplication. To show that Π'' supports homomorphic multiplication, we revisit the transformation from Π to Π'. Recall that the ciphertexts ctxt''_x and ctxt''_y are of the form

$$\mathsf{ctxt}''_x = (\mathsf{ctxt}'_{x\cdot 2^j})_{j=0}^{\ell} = (\mathsf{ctxt}_{x\cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell} \qquad \text{and} \qquad \mathsf{ctxt}''_y = (\mathsf{ctxt}'_{y\cdot 2^j})_{j=0}^{\ell} = (\mathsf{ctxt}_{y\cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell}$$

To obtain $\mathsf{ctxt}''_{x\cdot y}$ from ctxt''_x and ctxt''_y, we can perform the following for each (i,j):

- Let $L_{i,j}(\cdot) \approx \mathsf{Dec}(\cdot, \mathsf{ctxt}_{x\cdot \mathsf{sk}_i \cdot 2^j})$.
- Write $L_{i,j}(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) = \sum_{h=1}^{n} a_h \cdot \mathsf{sk}_h$ with binary decomposition $a_h = \sum_{k=0}^{\ell} a_{h,k} \cdot 2^k$.
- Output $\mathsf{ctxt}_{L_{i,j}(y\cdot \mathsf{sk})} = \sum_{h=1}^{n} \sum_{k=0}^{\ell} a_{h,k} \cdot \mathsf{ctxt}_{y\cdot \mathsf{sk}_h \cdot 2^k}$.

Since

$$L_{i,j}(y \cdot \mathsf{sk}) = \mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_{x\cdot \mathsf{sk}_i \cdot 2^j}) = x \cdot y \cdot \mathsf{sk}_i \cdot 2^j,$$

the above yields

$$\mathsf{ctxt}''_{x\cdot y} = (\mathsf{ctxt}_{x\cdot y\cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell}.$$

If we start with ctxt''_x and ctxt''_y with error e, the resulting ciphertext $\mathsf{ctxt}''_{x\cdot y}$ will have error roughly $e \cdot n \cdot \log q$.

Bringing Down the Noise by Bootstrapping. In the above, we constructed a scheme Π'' which supports noisy homomorphic additions and multiplications. Evaluating a depth-d circuit brings the error from e to roughly $e \cdot (n \cdot \log q)^d$ which is exponential in the depth d. In order to evaluate circuit of arbitrary depth, we need a mechanism to bring the noise level down. The bootstrapping technique of Gentry [Gen09] provides precisely such a mechanism. The idea is to homomorphically evaluate the (exact) decryption circuit on an encryption of the secret key. To be more concrete, suppose we are given the following:

- ctxt_x: An encryption of some message x whose noise level is very close to blowing up so that no more homomorphic operation can be safely performed.
- $\mathsf{ctxt}_{\mathsf{sk}}$: A fresh encryption of the secret key sk with low noise level.

We investigate the output of the following operation:

$$\mathsf{Eval}(\mathsf{pk}, \mathsf{Dec}(\cdot, \mathsf{ctxt}_x), \mathsf{ctxt}_{\mathsf{sk}})$$

In words, we consider the decryption circuit, hardwired with the ciphertext ctxt_x, as a function of the secret key sk, and homomorphically evaluate this function on a ciphertext $\mathsf{ctxt}_{\mathsf{sk}}$ of sk. Note that since the homomorphic operation is performed on $\mathsf{ctxt}_{\mathsf{sk}}$, the noise level of the resulting ciphertext depends only on the noise level of $\mathsf{ctxt}_{\mathsf{sk}}$ and the complexity of $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$, but is independent of the noise level of ctxt_x. By choosing parameters so that the maximum supported depth is considerably greater than the depth of

$\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$, the ciphertext resulting from $\mathsf{Eval}(\mathsf{pk}, \mathsf{Dec}(\cdot, \mathsf{ctxt}_x), \mathsf{ctxt}_{\mathsf{sk}})$ will have a moderate noise level. Moreover, by evaluation correctness, the homomorphic operation results in a ciphertext of

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_x) = x.$$

We therefore went from a ciphertext ctxt_x of x with a high noise level to a new ciphertext of x with a moderate noise level, which allows us to carry on with whatever homomorphic operations that we wish to perform on x.

9.5 GSW Construction

To conclude this lecture, we study below the FHE construction of Gentry, Sahai, and Waters (GSW) [GSW13] which can be explained by the above generic construction.

Gadget Matrix. To explain the GSW construction, we need to introduce a structured matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$, where $m = n \cdot (\lfloor \log q \rfloor + 1)$, known as the "gadget matrix":

$$\mathbf{G} \coloneqq \begin{pmatrix} 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} & & & & & & & & \\ & & & & 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} \end{pmatrix}$$

Observe that left-multiplying $\mathbf{G}$ to the concatenation n binary representations yields the n $\mathbb{Z}_q$ elements that they represent. Correspondingly, define the binary-decomposition operator $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$ which takes as input a vector $\mathbf{x} \in \mathbb{Z}_q^n$ and outputs a binary vector $\mathbf{u}_{\mathbf{x}} \in \{0,1\}^m$ which consists of binary representations of entries of $\mathbf{x}$ concatenated together. We emphasise that $\mathbf{G}^{-1}(\cdot)$ is not a matrix but an operator that acts on $\mathbb{Z}_q^n$ values. We can extend this operator naturally to operate on matrices $\mathbb{Z}_q^{n \times m}$, so that for a matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_m) \in \mathbb{Z}_q^{n \times m}$,

$$\mathbf{G}^{-1}(\mathbf{X}) \coloneqq (\mathbf{G}^{-1}(\mathbf{x}_1), \dots, \mathbf{G}^{-1}(\mathbf{x}_m)).$$

Observe that for any $\mathbf{X} \in \mathbb{Z}_q^{n \times m}$, it holds that

$$\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{X}) = \mathbf{X} \bmod q.$$

Scheme Description. We are now ready to describe the GSW construction. The GSW FHE scheme is for the message space $\{0,1\}$. Below, we give an informal summary of the construction.

The public key consists of m LWE samples $(\overline{\mathbf{A}}, \mathbf{b})$ packed into the form $\mathbf{A} \coloneqq \begin{pmatrix} \overline{\mathbf{A}} \\ \mathbf{b}^{\mathsf{T}} \end{pmatrix} \in \mathbb{Z}_q^{n \times m}$. The secret key is the corresponding LWE secret $\mathbf{s}$. Note that the public and secret keys satisfy $(-\mathbf{s}^{\mathsf{T}}, 1) \cdot \mathbf{A} \approx \mathbf{0}^{\mathsf{T}} \bmod q$. To encrypt, the public key $\mathbf{A}$ is rerandomised as $\mathbf{A} \cdot \mathbf{R}$ by a random square matrix $\mathbf{R}$ with small entries, and then used as a one-time-pad to mask the message $x \in \{0,1\}$, which is encoded as $x \cdot \mathbf{G}$. Decryption relies on the observation that if a ciphertext $\mathbf{C}$ is encrypting x, then $(-\mathbf{s}^{\mathsf{T}}, 1) \cdot \mathbf{A} \approx x \cdot (-\mathbf{s}^{\mathsf{T}}, 1) \cdot \mathbf{G} \bmod q$. Therefore, a decryption procedure similar to that of Regev's construction can recover the message x.

To support homomorphic evaluation of bounded-depth Boolean functions, it suffices to support the homomrophic evaluation of the NAND operation which, on the ciphertexts $\mathbf{C}_0$ and $\mathbf{C}_1$, can be computed as $\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$. To support arbitrary Boolean functions, the parameters are chosen such that the scheme is capable of evaluating its own decryption circuit, and the bootstrapping technique applies.

Formally, let $n, m, \log q, \log \beta \in \mathsf{poly}(\lambda)$ and let χ be the uniform distribution over $\mathbb{Z}_\beta$. We first describe the basic key generation, encryption, and decryption algorithms.

Key Generation $\mathsf{KGen}(1^\lambda)$:

- Sample a uniformly random matrix $\overline{\mathbf{A}} \leftarrow\!\!\$\ \mathbb{Z}_q^{(n-1)\times m}$, an LWE secret vector $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^{n-1}$, and a short noise vector $\mathbf{e} \leftarrow\!\!\$\ \chi^m$.
- Compute $\mathbf{b}^\intercal := \mathbf{s}^\intercal \cdot \overline{\mathbf{A}} + \mathbf{e}^\intercal \bmod q$.
- Set $\mathbf{A} := \begin{pmatrix} \overline{\mathbf{A}} \\ \mathbf{b}^\intercal \end{pmatrix} \in \mathbb{Z}_q^{n\times m}$.
- Output $\mathsf{pk} := \mathbf{A}$ and $\mathsf{sk} := \mathbf{s}$.

Encryption $\mathsf{Enc}(\mathsf{pk}, x \in \{0,1\})$:

- Sample short matrix $\mathbf{R} \leftarrow\!\!\$\ \chi^{m\times m}$.
- Output $\mathsf{ctxt} := \mathbf{C} := \mathbf{A} \cdot \mathbf{R} + x \cdot \mathbf{G}$.

Decryption $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt})$:

- Compute $\overline{\mathbf{x}}^\intercal := (-\mathbf{s}^\intercal, 1) \cdot \mathbf{C} \bmod q$.
- Let $\bar{x}$ be the last entry of $\overline{\mathbf{x}}$.
- If $|\bar{x}| < q/4$, output 0. Else, output 1.

Decryption Correctness. The analysis of decryption correctness is similar to that of Regev's encryption. We provide a sketch. Let $q \gg \beta$ be sufficiently large. Observe that

$$\begin{aligned}\overline{\mathbf{x}}^\intercal &:= (-\mathbf{s}^\intercal, 1) \cdot \mathbf{C} \bmod q \\ &= (-\mathbf{s}^\intercal, 1) \cdot \mathbf{A} \cdot \mathbf{R} + x \cdot (-\mathbf{s}^\intercal, 1) \cdot \mathbf{G} \bmod q \\ &= \mathbf{e}^\intercal \cdot \mathbf{R} + x \cdot (-\mathbf{s}^\intercal, 1) \cdot \mathbf{G} \bmod q.\end{aligned}$$

If $\mathbf{r}$ is the last column of $\mathbf{R}$, we have

$$\bar{x} = \mathbf{e}^\intercal \cdot \mathbf{r} + 2^{\lfloor \log q \rfloor} \cdot x \bmod q.$$

Since $\mathbf{e}$ and $\mathbf{r}$ are both short vectors, $\mathbf{e}^\intercal \cdot \mathbf{r}$ is still short. Therefore $x = 0$ if and only if $\bar{x}$ is closer to 0 than to $q/2$.

Homomorphic Evaluation and Evaluation Correctness. To understand the homomorphic NAND operation, we first describe a way to evaluate addition + and multiplication $\cdot$ over $\mathbb{Z}$ which is not always correct. Then, we use the fact that, for $x, y \in \{0,1\}$, $\mathsf{NAND}(x,y) = 1 - (x \cdot y)$, to evaluate the NAND operation homomorphically.

Addition (over $\mathbb{Z}$) $\mathsf{Eval}(\mathsf{pk}, +, \mathsf{ctxt}_0 = \mathbf{C}_0, \mathsf{ctxt}_1 = \mathbf{C}_1)$:

- Output $\mathbf{C}_0 + \mathbf{C}_1 \bmod q$.

Multiplication (over $\mathbb{Z}$) $\mathsf{Eval}(\mathsf{pk}, \cdot, \mathsf{ctxt}_0 = \mathbf{C}_0, \mathsf{ctxt}_1 = \mathbf{C}_1)$:

- Output $\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$.

NAND Operation $\mathsf{Eval}(\mathsf{pk}, \mathsf{NAND}, \mathsf{ctxt}_0 = \mathbf{C}_0, \mathsf{ctxt}_1 = \mathbf{C}_1)$:

- Output $\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$.

Let us investigate the evaluation correctness of the $+$ and $\cdot$ operations. Suppose $\mathbf{C}_0$ and $\mathbf{C}_1$ are ciphertexts of x_0 and x_1 with randomness $\mathbf{R}_0$ and $\mathbf{R}_1$ respectively, then

$$\mathbf{C}_0 + \mathbf{C}_1 = \mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G} + \mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G} = \mathbf{A} \cdot \underbrace{(\mathbf{R}_0 + \mathbf{R}_1)}_{\text{short}} + (x_0 + x_1) \cdot \mathbf{G} \bmod q.$$

For the case of $\cdot$, we have

$$\begin{aligned}
\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{C}_1 \bmod q \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) \bmod q \\
&= \mathbf{A} \cdot \underbrace{(\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1)}_{\text{short}} + x_0 \cdot x_1 \cdot \mathbf{G} \bmod q
\end{aligned}$$

where the encryption randomness in the resulting ciphertext is short because

- $\mathbf{R}_0$ and $\mathbf{R}_1$ are short encryption randomness in the input ciphertexts, and
- $\mathbf{G}^{-1}(\mathbf{C}_1)$ and x_0 are binary and hence short.

At first glance, the above suffices to prove the evaluation correctness of the $+$ and $\cdot$ operations, which would mean that we have constructed an FHE for $(\{0,1\}, +, \cdot)$! However, there is a subtle issue: For $x_0, x_1 \in \{0,1\}$, it does not necessarily hold that $x_0 + x_1 \in \{0,1\}$. In particular, we have $1 + 1 = 2$. Therefore, as we homomorphically evaluate more and more $+$ and $\cdot$ operations, the underlying message will grow so large that decryption correctness no longer holds. To fix this, we use the fact that the NAND operation can be expressed in terms of operations over $\mathbb{Z}$ by $\mathsf{NAND}(x, y) = 1 - (x \cdot y)$, and observe that

$$\begin{aligned}
\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= \mathbf{G} - (\mathbf{A} \cdot (\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1) + x_0 \cdot x_1 \cdot \mathbf{G}) \bmod q \\
&= \mathbf{A} \cdot \underbrace{(-\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) - x_0 \cdot \mathbf{R}_1)}_{\text{still short}} + (1 - x_0 \cdot x_1) \cdot \mathbf{G} \bmod q.
\end{aligned}$$

IND-CPA-Security. The analysis of IND-CPA-security is again similar to that of Regev's encryption. We provide a sketch. First, notice that the public key matrix $\mathbf{A}$ consists of the LWE samples $(\overline{\mathbf{A}}, \mathbf{b})$. By the LWE assumption (with the appropriate parameters), the distribution of public keys is computationally indistinguishable to that of uniformly random matrices. Second, by the leftover hash lemma, the distribution of $(\mathbf{A}, \mathbf{A} \cdot \mathbf{R} \bmod q)$ is statistically close to a uniform distribution. Consequently, ciphertexts are statistically close to uniformly random matrices.

Bootstrapping and Circular Security. By setting parameters appropriately, it is possible make the GSW construction support the homomorphic evaluation of its own (exact) decryption circuit, thereby allowing bootstrapping. However, note that bootstrapping requires publishing an encryption of the secret key sk as part of the public key. This makes proving IND-CPA-security from the LWE assumption tricky. Instead, what is always done in the literature is to simply assume that publishing an encryption of the secret key under itself does not harm IND-CPA-security. This is known as circular security. To date, the security of all FHE constructions rely on the bootstrapping technique and circular security assumptions.

References

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, May 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. "On Data Banks and Privacy Homomorphisms". In: *Foundations of Secure Computation, Academia Press* (1978), pp. 169–179.