Fully Homomorphic Encryption

One of the biggest surprises in the last 20 years was **fully homorphic encryption** which allows us to evaluate arbitrary functions on encrypted data

We work with a message space $\mathbb{Z}_2$ which is just single bits

- We allow boolean circuits to be evaluated on this, which are just a series of additions and multiplications
- These are equivalent to XOR and AND gates

> **FHE Scheme**
>
> Given a function class $\mathcal{C}$, an **FHE scheme** consists of four PPT algorithms:
>
> - `KeyGen(n)` produces `(sk, ek)`
>   - `sk` is a decryption key for the user
>   - `ek` is an evaluation key that is revealed to anyone that will be performing computations on ciphertexts encrypted by `sk`
> - `Enc(sk, u)` encrypts `u` with `sk` to produce `ct`
> - `Dec(sk, ct)` decrypts `ct` with `sk` to produce `ct`
> - `Eval(ek, F, ct_1, ct_2, ..., ct_l)` produces `ct`
>   - `F` belongs to $\mathcal{C}$ and maps `l` bits to a single bit, which is the output
>   - Note that the ciphertexts do not have to be single bits, so the output of this does not have to be a single bit

An FHE scheme has to satisfy three properties:

- Correctness: evaluations of F must be correct within a negligible distance from 1
- CPA-security: the `Enc` function must be CPA-secure
- Compactness: the bit lengths of both the ciphertexts `ct_i` and the output of `Eval -> ct` must depend only on n
  - Cannot depend on ℓ or $|F|$
  - This way, the server can't just concatenate F alongside all of the ciphertexts and return that as the `ct` for the client to have to evaluate

There are different types of homomorphic schemes based on the function class $\mathcal{C}$:

- For circuits with only addition gates, we have **linearly homomorphic schemes**, which we saw we can construct from LWE
- For circuits with only multiplication gates, we have **multiplicative homomorphic schemes**
- For circuits with both gates but bounded depth, we have **leveled homomorphic schemes**
  - The reason why these are leveled is because they might have accumultating error growth
- Arbitrary circuits
  - We can actually "boost" leveled homorphic schemes to solve these with some assumptions

Leveled Homomorphic Schemes

Gentry, Sahai, and Water’s construction of levelled FHE:

- At a high level, this scheme will depend on the fact that eigenvectors of a matrix are preserved across addition and multiplication
  - That is, if we have C_1 and C_2 with the shared eigenvector v with eigenvalues λ_1 and λ_2, then:
    - $C_1 + C_2$ has eigenvector v with $\lambda_1 + \lambda_2$
    - $C_1 \cdot C_2$ has eigenvector v with $\lambda_1 \cdot \lambda_2$
- Idea is to:
  - Make the secret key an eigenvector v
  - Our ciphertexts are matrices with eigenvector v and eigenvalue equal to the message being encrypted
  - We can then decrypt by multiplying by v and looking at the eigenvalue

- However, this doesn't work directly because in practice we can find eigenvectors very quickly
  - We combine this with LWE by making sure the following equation holds:

$$C \cdot s = s \cdot \mu + e$$

We first describe the `Enc` and `Dec` steps

- Here, we will actually not use the evaluation key `ek`
  - We will need it to boost this scheme to arbitrary depths
- We assume LWE with parameters (m, n, q, χ) with sufficiently large m
- We let $\ell = (n+1)\log q$

- `KeyGen`:
  - Sample a random vector $s' \in \mathbb{Z}_q^n$
  - Output $s = \begin{pmatrix} -s \\ 1 \end{pmatrix}$
- `Enc`:
  - Sample a random matrix $A \leftarrow \mathbb{Z}_q^{\ell \times n}$ and an error $e \in \chi^\ell$
  - Build the matrix B by concatenating $As' + e$ to A
  - Let $G \in \mathbb{Z}_q^{\ell \times n+1}$ be an error correcting matrix we will define later
  - Output $C = B + \mu \cdot G$
- `Dec`:
  - Compute $v = C \cdot s$
  - Output 0 if the magnitude of each entry in v is small ($< q/4$) and 1 otherwise

Before diving into how we support homomorphic operations, we first discuss correctness and security

- Since A is randomly chosen and by LWE $As' + e$ appears random, we have that B looks random
  - Therefore, C also looks random
- Expanding $C \cdot s$ out, we get that it is equivalent to $\mu \cdot Gs + e$
  - This is known as the **decryption invariant** and we want this to hold after every homomorphic operation
  - If $\mu = 0$, this will just be an error term and all terms will be near 0
  - If $\mu = 1$, then we expect that at least one of the terms $G \cdot s$ will have a large norm close to $q/2$ since s is uniformly sampled

Homomorphic Operations

To add two ciphertexts, we can simply add the matrices:

- We get that the errors terms add together, so the error accumulates
  - It at most doubles in magnitude
  - It preserves the decryption invariant

Multiplication is more difficult:

- The error correcting matrix G has to be carefully selected
- We define a function h that has two key properties:
  - $h(C) \cdot G = C$
  - Given C as input, $h(C)$ is $\log q$ times wider, and has only entries of magnitude 0 or 1
- Multiplications can then be computed as:

$$h(C_1) \cdot C_2$$

- To see why, we can expand as:

$$\begin{aligned} (h(C_1) \cdot C_2) \cdot s &= h(C_1) \cdot (\mu_2 Gs + e_2) \\ &= \mu_2 h(C_1) \cdot Gs + h(C_1) \cdot e_2 \\ &= \mu_1 \mu_2 \cdot C_1 s + h(C_1) \cdot e_2 \\ &= \mu_1 \mu_2 \cdot Gs + (\mu_2 e_1 + h(C_1) \cdot e_2) \end{aligned}$$

- Since each entry of $h(C_1)$ is small (0 or 1), the right hand side is a new error term that is relatively small
- More specifically, this error term is at most $(n+1)\log q + 1$ times bigger

What do we choose for G?

- We choose h to be the binary decomposition function
- Each entry in C is turned into binary with $\log_2 q$ new columns taking its place
- We can construct G as a semi-diagonal matrix that just reconstructs this

With this, we have that if our initial error can sit in the range $[-B, B]$, then as long as $q >> (n \log q)^d 2B$, then we can support a boolean function of depth d

- Equivalently, for large enough q, we can support $d \approx n^{0.99}$