Fully Homomorphic Encryption

Lecture 21
Learning With Errors

LWE (decision version): \((A, As + e) \approx (A, r)\), where \(A\) random matrix in \(A \in \mathbb{Z}_q^{m \times n}\), \(s\) uniform, \(e\) has “small” entries from a Gaussian distribution, and \(r\) uniform.

Average-case solution for LWE ⇒ Worst-case solution for GapSVP (for appropriate choice of parameters)
Learning With Errors

A pseudorandom matrix $M \in \mathbb{Z}_q^{m \times n'}$ and $z \in \mathbb{Z}_q^{n'}$ s.t. entries of $Mz$ are all small.
Gentry-Sahai-Waters

- Want to allow homomorphic operations on the ciphertext
- Rough plan: Ciphertext is a matrix. Addition and multiplication of messages by addition and multiplication of ciphertexts
- Recall from LWE: pseudorandom $M \in \mathbb{Z}_q^{m \times n}$ and random $z \in \mathbb{Z}_q^n$ s.t. $z^T M^T$ has small entries

**Public key $M \in \mathbb{Z}_q^{m \times n}$ and private key $z$**

- $\text{Enc}(\mu) = M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m \times km}$ and $G \in \mathbb{Z}_q^{n \times km}$ the matrix
  - to reverse bit-decomposition operation $B : \mathbb{Z}_q^{n \times d} \rightarrow \mathbb{Z}_q^{km \times d}$

- $\text{Dec}_z(C) : z^T C = \delta^T + \mu z^T G$ where $\delta^T = e^T R$
Gentry-Sahai-Waters

- Supports messages $\mu \in \{0,1\}$ and NAND operations up to an a priori bounded depth of NANDs.
- Public key $M \in \mathbb{Z}_q^{m \times n}$ and private key $z$ s.t. $z^T M$ has small entries.
- $\text{Enc}(\mu) = M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m \times km}$ (and $G \in \mathbb{Z}_q^{n \times km}$ the matrix to reverse bit-decomposition).
- $\text{Dec}_z(C) : z^T C = \delta^T + \mu z^T G$ where $\delta^T = e^T R$.
- $\text{NAND}(C_1, C_2) : G - C_1 \cdot B(C_2)$ (G is a (non-random) encryption of 1). 
  - $z^T C_1 \cdot B(C_2) = z^T C_1 \cdot B(C_2) = (\delta^T_1 + \mu_1 z^T G) \cdot B(C_2)$
    - $= \delta^T_1 B(C_2) + \mu_1 z^T C_2 = \delta^T + \mu_1 \mu_2 z^T G$
  - where $\delta^T = \delta^T_1 B(C_2) + \mu_1 \delta^T_2$ has small entries.
- In general, error gets multiplied by $km$. Allows depth $\approx \log_{km} q$.

Only “left depth” counts, since $\delta \leq k \cdot m \cdot \delta_1 + \delta_2$
Removing the need for an a priori bound

Main idea: Can “refresh” the ciphertext to reduce noise

Refresh: homomorphically decrypt the given ciphertext under a fresh layer of encryption

cf. Degree reduction via share-switching: Homomorphically reconstruct under a fresh layer of sharing

But here, we have a secret-key (and there is only one party who knows the ciphertext fully)

Ciphertext is known, but secret-key should be kept encrypted

Consider decryption of a given ciphertext as a function applied to the secret-key: $D_C(sk) := \text{Dec}(C,sk)$
Given a ciphertext \( C \) and hence the decryption function \( D_c \) s.t. 
\[
D_c(sk) := \text{Dec}(C, sk)
\]

Also given: an encryption of \( sk \) (beware: circularity!)

Goal: a fresh ciphertext with message \( D_c(sk) \)

- Fresh encryption of \( sk \), provided along with the public key
- Homomorphic evaluation in the ciphertext space
- Refreshed: Doesn’t depend on how unfresh \( C \) was, but only on the depth of \( D_c \)
If depth of $D_C$ s.t. $D_C(sk) := \text{Dec}(C,sk)$ is strictly less than the depth allowed by the homomorphic encryption scheme, a ciphertext $C$ can be strictly refreshed.

Then can carry out at least one more operation on such ciphertexts (before refreshing again).

Refreshed: Doesn’t depend on how unfresh $C$ was, but only on the depth of $D_C$.

Fresh encryption of $sk$, provided along with the public key.

Homomorphic evaluation in the ciphertext space.
Circularity: Encrypting the secret-key of a scheme under the scheme itself

- Can break security in general!
- LWE does not by itself imply security
- Stronger assumption: “Circular Security of LWE”

- Fresh encryption of $sk$, provided along with the public key
- Homomorphic evaluation in the ciphertext space
- Refreshed: Doesn’t depend on how unfresh $C$ was, but only on the depth of $D_c$
Bootstrapping GSW

- Supports log(k) depth computation with poly(k) complexity
- Need low depth decryption (as a function of secret-key)

\[ \text{Dec}_Z(C) : z^{TC} = \delta^T + \mu z^{TG} \text{ where } \delta^T = e^T R \]

And then check if the result is close to \( 0^T \) or \( z^{TG} \)

How?

Multiplying by \( B(w) \) where last coordinate of \( w \) is \( \lfloor q/2 \rfloor \) and other coordinates 0

\[ z^{TC} B(w) = \delta^T B(w) + \mu z^T w = \varepsilon + \mu \lfloor q/2 \rfloor \]

Has most significant bit = \( \mu \) (since error \( |\varepsilon| \ll q/4 \))

\[ \text{Dec}_Z(C) : \text{MSB}( z^{TC} B(w) ). \text{ All operations mod } q. \]

If \( q \) were small (poly(k)) this would be small depth (log(k))

Problem: \( q \) is super-polynomial in security parameter \( k \)

Idea: Can change modulus for decryption!
Modulus Switching for GSW

\[ \text{Dec}_z(C) : \text{MSB}( \mathbf{z}^T Y \mod q), \text{ where } Y = C \mathbf{B}(\mathbf{w}) \]

\[ \mathbf{z}^T Y = \varepsilon_0 + \mu \left( \frac{q}{2} \right) + aq \text{ (for some } a \in \mathbb{Z}) \]

To switch to a smaller modulus \( p < q \):

Consider \( Y' = \left\lceil \left( \frac{p}{q} \right) Y \right\rceil \). Let \( \Delta = Y' - (p/q)Y \).

\[ \mathbf{z}^T Y' = \left( \frac{p}{q} \right) \mathbf{z}^T Y + \mathbf{z}^T \Delta \]

\[ = \varepsilon_1 + \mu \left( \frac{p}{2} \right) + ap \text{ where } \varepsilon_1 = (p/q)\varepsilon_0 + \mathbf{z}^T \Delta \]

Need \( \mathbf{z}^T \Delta \) to be small. But \( \mathbf{z}^T = \begin{bmatrix} -s^T & 1 \end{bmatrix} \) for \( s \) uniform in \( \mathbb{Z}_q^n \).

Fix: LWE with small \( s \) is as good as with uniform \( s \) [Exercise]

Final bootstrapping:

Given \( C \), let \( Y' = \left\lceil \left( \frac{p}{q} \right) C \mathbf{B}(\mathbf{w}) \right\rceil \) where \( p \) small (poly(k)). Define function \( D_{Y'} \) which does decryption mod p. Homomorphically evaluate \( D_{Y'} \) on encryption of \( \mathbf{z} \mod p \) (encryption is mod q).
FHE in Practice

Several implementations in recent years

Prominent ones based on schemes of Fan-Vercauteren (FV) and Brakerski-Gentry-Vaikuntanathan (BGV) with various subsequent optimisations

- BGV implementations: HELib (IBM), $\Lambda \circ \lambda$
- FV implementations: SEAL (Microsoft), FV-NFLlib (CryptoExperts), HomomorphicEncryption R Package ...

Both based on “Ring LWE”

Moderately fast

E.g., HELib can apply AES (encipher/decipher) to about 200 plaintext blocks using an encrypted key in about 20 minutes (a bit faster without bootstrapping, if no need to further compute on the ciphertext)