Lecture 19: Fully Homomorphic Encryption (FHE) Pt. 1
Plan

Recap: LWE

Fully Homomorphic Encryption

Introduction & history
Syntax & security
Building leveled FHE
Recap: Learning with Errors

\[
\text{LWE}(n,m,q,X_0)
\]

\[
\left\{ (A,A^s \oplus \hat{e}) : \begin{array}{c}
A \in \mathbb{Z}_q^{m \times n} \\
n \in \mathbb{Z}_q \\
\end{array} \right\} \cong \left\{ (A,u) : A^s \in \mathbb{Z}_q^{m \times n} \right\}
\]
Fully Homomorphic Encryption

Idea: outsource computation without revealing inputs!

\[ E, Nyt \rightarrow Enc(X) \rightarrow Cloud \]

input \( x \)
output \( f(x) \)

Note that amount of communication is independent of \( |f| \)

Examples:
- PIR: input \( i \), output \( f(i) = DB[i] \)
- private ML: training, inference
- whatever you want to outsource!
Brief history

- **1978** - Rivest, Adleman, Dertouzous introduced a notion of FHE

  ↓ no unbroken candidates for many years

- **2009** - Craig Gentry (Stanford PhD student) introduces first construction
  - from new (non-standard) assumption
  - introduces “bootstrapping” idea (see next lecture)

- **2011** - Brakerski, Vaikuntanathan

  FHE based on LWE

- **2013** - Gentry, Sahai, Waters

  “3rd gen” FHE → today’s topic

Context: Diffie&Hellman introduced PK crypto in 1976
FHE syntax

KeyGen(1^n) \rightarrow sk

Enc(sk, M) \rightarrow ct

Dec(sk, ct) \rightarrow M

Eval(F, ct_1, ..., ct_n) \rightarrow \tilde{ct}

We will look at symmetric-key FHE. There is actually a generic transformation to get PK FHE.
FHE Properties

1) Correctness:
   \( \forall F : \{0,1\}^n \rightarrow \{0,1\}^k, \quad M_1, \ldots, M_n \in \{0,1\}^k \)

   \( sk \leftarrow \text{KeyGen}(\lambda), \) then with probability \( 2^{\lambda} \):

   \[
   \text{Dec}(sk, \text{Eval}(F, \text{Enc}(sk, M_1), \ldots, \text{Enc}(sk, M_n))) = F(M_1, \ldots, M_n) + \text{usual encryption correctness}
   \]

2) Semantic security

   \( \forall M_i, \mu, \nu \in \{0,1\} \quad \text{Enc}(sk, M_i) \approx_c \text{Enc}(sk, \mu, \nu) \)

3) Compactness

   \( \forall F, sk \quad ct. \leftarrow \text{Enc}(sk, M_i) \)

   if \( \tilde{ct} \leftarrow \text{Eval}(F, ct, \ldots, ct, c) \)

   then \( |\tilde{ct}| = \text{poly}(\lambda) \leftarrow \tilde{ct} \text{ size is independent of } |F|, \lambda \)

Note: without compactness, any encryption scheme is also fully homomorphic!
Constructing FHE

Today: construct leveled FHE

\[ \text{can only evaluate low-depth circuits} \]

Reason: ciphertexts have noise that grows with each gate in circuit. Eventually, the noise overwhelms the msg.

Next time: use bootstrapping to remove restriction on circuit dep.

\[ \text{Idea: refresh ciphertexts to clear accumulated noise.} \]
Attempt 1 (insecure)

Secret key is a vector \( \vec{s} \)

\[
\text{Enc}(\vec{s}, \mu) \rightarrow \text{matrix } \mathbf{C} \text{ s.t. } \mathbf{C} \cdot \vec{s} = \mu \cdot \vec{s}
\]

\[
\text{Dec}(\vec{s}, \mathbf{C}) \rightarrow \text{compute } \mathbf{C} \cdot \vec{s} = \mu \cdot \vec{s} \text{ and find } \mu
\]

Homomorphism: if \( C_1, C_2, \ldots, C_n \) are encryptions of \( M_1, M_2, \ldots, M_n \)

Addition: \( \vec{C} = C_1 + C_2 \) \( \rightarrow \) \( (C_1 + C_2) \cdot \vec{s} = C_1 \cdot \vec{s} + C_2 \cdot \vec{s} = M_1 \cdot \vec{s} + M_2 \cdot \vec{s} = (M_1 + M_2) \cdot \vec{s} \)

Eval("+", \( C_1, C_2 \))

Multiplication: \( \vec{C} = C_1 \cdot C_2 \)

Eval("\cdot", \( C_1, C_2 \))

\[
(C_1 \cdot C_2) \cdot \vec{s} = C_1 \cdot (C_2 \cdot \vec{s}) = C_1 \cdot M_2 \cdot \vec{s} = M_2 \cdot (C_1 \cdot \vec{s}) = M_2 \cdot M_1 \cdot \vec{s} = M_1 M_2 \cdot \vec{s}
\]

Eval("\cdot", \( C_1, C_2 \))

Can eval +, \( \cdot \rightarrow \) fully homomorphic!

**Problem:** Given \( \mathbf{C} \), it’s easy to find \( \vec{s} \) using Gaussian elimination = 

idea: We can make Gaussian elimination hard by adding noise!

\[
\mathbf{C} \cdot \vec{s} = \mu \cdot \vec{s} + \vec{\varepsilon}
\]

Small noise
Attempt 2: Secret-key variant of Regev encryption

KeyGen(1^n):
\( \tilde{s} \in \mathbb{Z}_q^{n-1}, \ s \leftarrow (\tilde{s}) \in \mathbb{Z}_q^n \)

Enc(\(\tilde{s}, m\)):
\( A \in \mathbb{Z}_q^{n \times (n-1)}, \ \tilde{e} \in \mathbb{X}_q^n \)

\( output \ C = (A, A\tilde{s} + \tilde{e}) + M \cdot \text{In} \in \mathbb{Z}_q^{n \times n} \)

\[ \text{pseudorandom by LWE} \]

Dec(\(\tilde{s}, C\)):
compute \( C \cdot \tilde{s} \), output \( 0 \) if \( \| C \cdot \tilde{s} \|_\infty \) small
\( \{ 1 \) otherwise

\( C \cdot \tilde{s} = (A, A\tilde{s} + \tilde{e}) (\tilde{s}) + M \cdot \text{In} \tilde{s} \)

\( = A\tilde{s} - A\tilde{s} - \tilde{e} + M\tilde{s} = M\tilde{s} + \text{noise} \)

\( \tilde{s} \) is "approximate eigenvector" of \( C \)

with approx. eigenvalue \( M \).
Homomorphism:

Addition: \( \tilde{e} \leq C_1 + C_2 \)

\[
(C_1 + C_2)\tilde{s} = C_1\tilde{s} + C_2\tilde{s} = M_1\tilde{s} + \tilde{e}_1 + M_2\tilde{s} + \tilde{e}_2 = (M_1 + M_2)\tilde{s} + (\tilde{e}_1 + \tilde{e}_2)
\]

Noise doesn't grow too much, so we have additive homomorphism

(Would need to adjust Dec for \( M \notin \{0, 1\} \))

Multiplication: can we do \( \tilde{e} \leq C_1 \cdot C_2 \)?

\[
(C_1 \cdot C_2)\tilde{s} = C_1(C_2\tilde{s} + \tilde{e}_2) = M_2C_1\tilde{s} + C_1\tilde{e}_2
\]

\[
= M_2(M_1\tilde{s} + \tilde{e}_1) + C_1\tilde{e}_2
= M_1M_2\tilde{s} + M_2\tilde{e}_1 + C_1\tilde{e}_2
\]

Noise still reasonably small

Can be large \( \implies \) for \( M_2 \in \{0, 1\} \)

So we're still unable to multiply b/c noise grows with \( \|C\|_\infty \), which can be large.

Need a way to make ciphertext matrices have small norm.

Idea: represent number \( x \in \mathbb{Z}_q \) as a small-norm vector via binary decomposition!
For $x \in \mathbb{Z}_2$, define $\hat{x} = (x_0, x_1, \ldots, x_{\log_2 - 1})$ s.t. $x = \sum_{i=0}^{\log_2 - 1} x_i \cdot 2^i$.

In the inverse operation $\hat{x} \rightarrow x$, $\hat{x} \cdot \tilde{G} = x$.

Let $G$ be the vector that recovers $x$ from $\hat{x}$: $\hat{x} \cdot \tilde{G} = x$.

Note: we call it $\tilde{G}$ for "gadget".
can extend (^) operation to vectors

\[ \hat{x} \in \mathbb{E}_q \rightarrow \hat{x} = (x_0, x_0, \ldots, x_0, \log_q^{-1}, \ldots, x_{n_0}, x_{n_0}, \ldots, x_{n_0}) \in \mathbb{E}_{0,13}^{n \log_q} \]

\[ \hat{x} = \hat{x} \cdot G \] is a linear transformation

where \( G \) is the matrix that recovers \( \hat{x} \) from \( \hat{x} \): \( \hat{x} \cdot G = \hat{x} \)

\[ G = \begin{bmatrix}
1 \\
2 \\
\vdots \\
\log_q^{-1} \\
\log_q^{-1} \\
\log_q^{-1} \\
\vdots \\
1 \\
2 \\
\vdots \\
\log_q^{-1}
\end{bmatrix} \]
Finally, can also extend $(\cdot)$ to matrices

$$C = \begin{pmatrix} \hat{z}_1 \\ \vdots \\ \hat{z}_m \end{pmatrix} \rightarrow \hat{C} = \begin{pmatrix} \hat{c}_1 \\ \vdots \\ \hat{c}_m \end{pmatrix}$$

And $C = \hat{C} \cdot G$ is still a linear transformation

(with gadget matrix $G$ same as above)

Note: some sources refer to $G$ as $G^{-1}$ because it inverts bit decomposition

Now, let’s get back to FHE!
3rd (and final) attempt: the GSW scheme

KeyGen(1^n):  \( \hat{s} \in \mathbb{Z}_q \) \( \hat{s} - (\hat{\xi}) \in \mathbb{Z}_q^* \)

Enc(\( \hat{s} \), M): \( A \in \mathbb{Z}_q^{m \times (n-1)} \) for \( m = \log q \)

\( \hat{e} \in \mathbb{X}_B \)

\[ C = (A, A\hat{s} + \hat{e}) + \mu G \]

\( C \) Output \( \hat{C} \)

M\times M = M\times n\log q

Observe that \( C \hat{t} = \hat{C} \) has low norm since it is a \( \Theta(1) \) matrix!

Dec(\( \hat{s} \), \( \hat{C} \)): compute \( \hat{C} \cdot G \cdot \hat{s} = C \cdot \hat{s} \)

\[ = (A, A\hat{s} + \hat{e}) \begin{pmatrix} \hat{s} \\ -1 \end{pmatrix} + \mu G \cdot \hat{s} \]

\[ = \mu \cdot G \cdot \hat{s} - \hat{e} \]

if first element is small, output \( \mu = 0 \), Else, output \( \mu = 1 \).

Why first element?

\[ (\mu \cdot G \cdot \hat{s})_1 = \mu \begin{pmatrix} 1 \\ 0 \ldots \ldots 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n-1} \\ -1 \end{pmatrix} \]

\[ = \mu \cdot s_1 \text{ which is } \Theta(q) \text{ why? if } \mu = 1 \]
Let's see how this solves our multiplication problem:

Eval ($\hat{\cdot}$, $\hat{C}_1$, $\hat{C}_2$): output $\hat{C} = \hat{C}_1 \cdot \hat{C}_2$ ≤ all $m \times m$ matrices

Need to check $\text{Dec}(\hat{\cdot}, \hat{C}) ? M_1 \cdot M_2$

Proof: $\hat{C}_1 \cdot \hat{C}_2 \cdot \hat{G} \cdot \hat{S} = \hat{C}_1 \cdot (\hat{C}_2 \cdot \hat{G} \cdot \hat{S})$

$\hat{C}_2 = \hat{C}_1 \cdot (M_2 \cdot \hat{G} \cdot \hat{S} + \hat{e}_2)$

$= M_2 \cdot \hat{C}_1 \cdot \hat{G} \cdot \hat{S} + \hat{C}_1 \cdot \hat{e}_2$

$= M_2 \cdot (\hat{C}_1 \cdot \hat{G} \cdot \hat{S} + \hat{e}_2) + \hat{C}_1 \cdot \hat{e}_2$

$= M_1 \cdot M_2 \cdot \hat{G} \cdot \hat{S} + M_2 \cdot \hat{e}_1 + \hat{C}_1 \cdot \hat{e}_2$

Small if $M_2 \in \{0, 1\}$ Small Since $\|\hat{C}\|_\infty = \text{Small}$

What about addition? Could that make noise bigger?

Turns out it's sufficient to support the universal NAND gate:

\[ \text{NAND}(a, b) = \text{NOT}(\text{AND}(a, b)) \]

Using NAND for other gates:

\[ \text{NOT}(a) = \text{NAND}(a, a) \]
\[ \text{AND}(a, b) = \text{NOT}(\text{NAND}(a, b)) \]
\[ \text{OR}(a, b) = \text{NAND}(\text{NOT}(a), \text{NOT}(b)) \]
So how to build NAND?

\[ \text{Eval}(\text{NAND}, \hat{C}_1, \hat{C}_2) : I_{m \times m} - \hat{C}_1 \cdot \hat{C}_2 \]

ct has \( \mu \) added along its diagonal, so \( 1 - \mu \) is NOT mult over \( \mathbb{F}_0, 1 \) is AND

Yay! We have constructed an encryption scheme that can compute a universal gate over ciphertexts!

But this is a \textit{leveled} FHE, so we're not done yet.

Next time we'll see why this is not quite an FHE yet (noise growth) as well as a technique called \textit{Bootstrapping} that will allow us to get a full FHE scheme.