A Somewhat Informal Introduction to FHE

David Wu
August, 2014
Basic Definitions
Homomorphic Encryption

Homomorphic encryption scheme: encryption scheme that allows computation on ciphertexts

Comprises of three functions:

Must satisfy usual notion of semantic security
Homomorphic Encryption

Homomorphic encryption scheme: encryption scheme that allows computation on ciphertexts

Comprises of three functions:

\[ c_1 = \text{Enc}_{pk}(m_1) \]
\[ c_2 = \text{Enc}_{pk}(m_2) \]

\[ \text{Eval}_f (c_1, c_2, e_k) \]

\[ \text{Dec}_{sk} \left( \text{Eval}_f (e_k, c_1, c_2) \right) = f(m_1, m_2) \]
Fully Homomorphic Encryption (FHE)

Many homomorphic encryption schemes:
- ElGamal: $f(m_0, m_1) = m_0 m_1$
- Paillier: $f(m_0, m_1) = m_0 + m_1$
- Goldwasser-Micali: $f(m_0, m_1) = m_0 \oplus m_1$

Fully homomorphic encryption: homomorphic with respect to two operations: addition and multiplication
- Can evaluate Boolean and arithmetic circuits
- [BGN05]: one multiplication, many additions
- [Gen09]: first FHE construction from lattices
Fully Homomorphic Encryption

\[ c_1 = \text{Enc}_{pk}(m_1) \]
\[ c_2 = \text{Enc}_{pk}(m_2) \]
\[ c_3 = \text{Eval}(c_1, c_2) \]

\[ C(f): \text{circuit for some function } f \]

Correctness: \( \text{Dec}_{sk}(\text{Eval}_f(ek, c_1, c_2)) = f(m_1, m_2) \)

Circuit Privacy: \( \text{Enc}_{pk}(C(m_1, m_2)) \approx \text{Eval}_f(ek, c_1, c_2) \)

Compactness: Decryption circuit has size at most \( \text{poly}(\lambda) \)
Lattices and LWE
Lattices

All known FHE constructions based on lattice problems

Lattices are discrete additive subgroups

Equivalent definition: the set of integer combination of basis vectors

discrete subgroup: no other lattice point contained in ball of radius $\epsilon > 0$ around each lattice point
Hard Lattice Problems

Finding a short vector in a lattice (SVP)

“Good” basis: easy

“Bad” basis: not so easy

Exact SVP is NP-hard. Approximation algorithms try to find a “good” basis using lattice-reduction techniques
Learning with Errors (LWE) [Reg05]

LWE Assumption: distributions 1 and 2 are computationally indistinguishable
Learning with Errors (LWE)

A gold mine of applications!

- PKC: [Reg05], [KTX07], [Pei09]
- FHE: [BV11], [BGV12], [Bra12], [GSW13]
- IBE: [GPV08], [CHKP10], [ABB10]
- ABE: [GVW13], [BCG+14]
- FE: [AFV11]
- ... and many more!
Public Key Encryption from LWE [Reg05]

$t \leftarrow \mathbb{Z}_q^n$

$B \leftarrow \mathbb{Z}_q^{m \times n}$

$e \leftarrow \chi^m$

1

$-t$

secret key $s$

public key $A$

secret key is LWE secret, public key consists of LWE samples
Regev Encryption

\[ r \leftarrow \{0,1\}^m \]

\[ \text{public key} \]

\[ m \in \{0,1\} \]

Random subset sum of rows in public key, with message embedded in leading component.

\[ r^T \times \begin{bmatrix} Bt + e \\ \end{bmatrix} + \begin{bmatrix} m \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ 0^n \end{bmatrix} \]
Regev Decryption

\[ r^T (Bt + e) + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \]

\[
\begin{bmatrix}
\vdots \\
r^T B \\
\vdots
\end{bmatrix}
\times
\begin{bmatrix}
1 \\
-t
\end{bmatrix}
= r^T Bt + r^T e + m \cdot \left\lfloor \frac{q}{2} \right\rfloor - r^T Bt
\]

\[
= m \cdot \left\lfloor \frac{q}{2} \right\rfloor + r^T e
\]

Multiplying by \( \frac{2}{q} \) recovers the message if \( r^T e \) is small.
PKC from LWE: Regev Encryption [Reg05]

- **Private key:** choose $t \leftarrow \mathbb{Z}_q^n$ and set $s \leftarrow (1, -t)$

- **Public key:** Choose $B \leftarrow \mathbb{Z}_q^{m \times n}$, $e \leftarrow \chi^m$ and compute

  $$A \leftarrow (Bs + e, B) \in \mathbb{Z}_q^{m \times (n+1)}$$

- **Encrypt:** Choose random 0/1 vector $r \leftarrow \{0,1\}^m$ and compute

  $$r^T A + \left( m \cdot \left\lfloor \frac{q}{2} \right\rfloor, 0^n \right) \in \mathbb{Z}_q^{n+1}$$

- **Decrypt:** To decrypt ciphertext $c$, compute $\left[ \frac{2}{q} \langle c, s \rangle \right]$
PKC from LWE: Regev Encryption [Reg05]

**Correctness:** if error sufficiently small \( (< \frac{q}{4}) \), then rounding yields the underlying message.

**Security:** random subset sum of \((a_i, b_i)\) is statistically close to uniform (argument based on leftover hash lemma). Security follows by LWE assumption.
PKC from LWE: Regev Encryption [Reg05]

**Key intuition:** hide message by adding some noise; everything works if noise is sufficiently small.

Basic observation underlying many FHE constructions
SWHE Construction from LWE
From SWHE to FHE

• Somewhat homomorphic encryption: encryption scheme that supports a *limited* number of operations

• All known constructions based on lattices:
  • Hide messages by adding noise
  • Homomorphic operations increase noise

• Gentry’s blueprint [Gen09]: bootstrapping SWHE to FHE
  • Homomorphically evaluate the decryption circuit
  • Provides a way to “refresh” a ciphertext
A Simple SWHE Scheme [GSW13]

• Ciphertext are matrices
• Secret key is a vector \( v \in \mathbb{Z}_q^n \)
• A ciphertext \( C \) encrypts a message \( m \) if the following holds:
  \[
  Cv = mv + e
  \]
  where \( e \) is a small error term
• **Intuition:** the message is an *approximate* eigenvalue of the ciphertext
The GSW Scheme

- A ciphertext $C$ encrypts a message $m$ if the following holds:
  $Cv = mv + e$
  where $e$ is a small error term
- Can decrypt if $v$ has a "big" coefficient $v_i$ by rounding:
  $\left\lfloor \frac{\langle C_i, v \rangle}{v_i} \right\rfloor = \left\lfloor \frac{mv_i + e}{v_i} \right\rfloor$
  where $C_i$ denotes the $i^{th}$ row of $C$
The GSW Scheme

- Homomorphic operations very natural – suppose $C_1$ encrypts $m_1$ and $C_2$ encrypts $m_2$

- Homomorphic addition: $C_1 + C_2$ (almost) encrypts $m_1 + m_2$:
  \[(C_1 + C_2)v = (m_1 + m_2)v + e_1 + e_2\]

- Homomorphic multiplication: $C_1 C_2$ (almost) encrypts $m_1 m_2$:
  \[C_1 C_2 v = (m_1 m_2)v + m_2 e_1 + C_1 e_2\]

- Everything works if noise is small enough
Constraining Noise Growth

• Recall Regev decryption:

\[ m \leftarrow \left\lfloor \frac{2}{q} \langle c, s \rangle \right\rfloor \]

• Key operation is inner product

• Want transformation that preserves inner product while reducing “size” (norm) of vectors
Bit Decomposition

• Let $\ell = \lceil \log_2 q \rceil + 1$ and suppose $z \in \mathbb{Z}_q^n$

• $\text{BitDecomp}(z) = (z_{1,0}, \ldots, z_{1,\ell-1}, \ldots, z_{n,0}, \ldots, z_{n,\ell-1})$ where $z_{i,j}$ is the $j^{th}$ bit of the binary decomposition of $z_i$

• $\text{BitDecomp}^{-1}(z') = (\sum_{j=1}^{\ell} 2^j z'_{1,j}, \ldots, \sum_{j=1}^{\ell} 2^j z'_{n,j})$

• $\text{PowersOfTwo}(z) = (z_1, 2z_1, \ldots, 2^{\ell-1}z_1, \ldots, z_n, 2z_n, \ldots, 2^{\ell-1} z_n)$
Bit Decomposition

- $\text{BitDecomp}(z) = (z_{1,0}, \ldots, z_{1,\ell-1}, \ldots, z_{n,0}, \ldots, z_{n,\ell-1})$
- $\text{PowersOfTwo}(z) = (z_1, 2z_1, \ldots, 2^{\ell-1}z_1, \ldots, z_n, 2z_n, \ldots, 2^{\ell-1}z_n)$

$\langle \text{BitDecomp}(x), \text{PowersOfTwo}(y) \rangle = \langle x, y \rangle$
Flattening a Vector

- \( \text{Flatten}(z) = \text{BitDecomp}(\text{BitDecomp}^{-1}(z)) \)

- \( \text{Flatten}(z) \) is a 0/1 vector even though \( z \) need not be a 0/1 vector

\[
\langle x, \text{PowersOfTwo}(y) \rangle = \sum_{i=1}^{n} \sum_{j=0}^{\ell-1} x_{i,j} \cdot 2^j y_i
\]

Preserves inner product with \( \text{PowersOfTwo}(\cdot) \)

\[
= \sum_{i=1}^{n} y_i \sum_{j=0}^{\ell-1} 2^j x_{i,j}
\]

\[
= \langle \text{BitDecomp}^{-1}(x), y \rangle
\]

\[
= \langle \text{Flatten}(x), \text{PowersOfTwo}(y) \rangle
\]
GSW Key Generation

PowersOfTwo

\[
\begin{pmatrix}
1 \\
-t
\end{pmatrix}
\]

\[t \leftarrow \mathbb{Z}_q^n\]

secret key

PowersOfTwo(s)

Regev-like, but where we apply PowersOfTwo to the secret

\[
B \leftarrow \mathbb{Z}_{q}^{m \times n}
\]

\[
e \leftarrow \mathbb{X}^m
\]

public key \( A \)

Note: \( As = Bt + e - Bt = e \)
GSW Encryption

• Recall Regev decryption:

\[ m \leftarrow \left\lceil \frac{2}{q} \langle c, s \rangle \right\rceil \]

• So far, replaced \( s \) with \( \text{PowersOfTwo}(s) \), so to preserve inner product, we apply \( \text{BitDecomp} \) to the ciphertext \( c \).
GSW Encryption

\[ R \leftarrow \{0,1\}^{N \times m} \]

Constrains norm of ciphertext, but preserves inner product \( \langle c, \text{PowersOfTwo}(s) \rangle \)

\[ Bt + e \]

BitDecomp

\[ m \cdot I_N \]

public key

\[ m \in \{0,1\} \]
Approximate Eigenvalues

- Secret key is
  \[ \nu \leftarrow \text{PowersOfTwo}(s) \]
- Encryption of a message \( m \in \{0,1\} \) given by
  \[ C \leftarrow \text{Flatten}(m \cdot I_N + \text{BitDecomp}(R \cdot A)) \]
- Observe:
  \[ Cv = mv + RAs = mv + \boxed{Re} \]
  Small since \( R \) is 0/1 matrix
Revisiting Homomorphic Operations

• Homomorphic operations very natural – suppose $C_1$ encrypts $m_1$ and $C_2$ encrypts $m_2$

• Homomorphic addition: $C_1 + C_2$ encrypts $m_1 + m_2$:

\[(C_1 + C_2)v = (m_1 + m_2)v + e_1 + e_2\]

• If $e_1$ and $e_2$ are small, then is $e_1 + e_2$ is small
Revisiting Homomorphic Operations

• Homomorphic operations very natural – suppose $C_1$ encrypts $m_1$ and $C_2$ encrypts $m_2$

• Homomorphic multiplication: $C_1 C_2$ (almost) encrypts $m_1 m_2$:

\[ C_1 C_2 v = (m_1 m_2) v + m_2 e_1 + C_1 e_2 \]

• Noise increases based on

  • $|m_2|$: OK since $m_2 \in \{0,1\}$
  • $\|C_1\|$: OK since $C_1$ is 0/1 matrix
Revisiting Homomorphic Operations

- But homomorphic operations might produce matrix that is not 0/1
- Can use the Flatten operation again!

- Homomorphic addition: Flatten($C_1 + C_2$)
- Homomorphic multiplication: Flatten($C_1 C_2$)

- Ciphertext always consist of 0/1 matrices
Brief Note on Security [High-Level]

• Public key components are simply LWE samples

• Ciphertext components are very similar to Regev encryptions (omitting a few small details, but a very similar proof carries through), and hardness derives from LWE
Questions?