Fully Homomorphic Encryption (FHE) - Part I

Additive Homomorphic Encryption (ElGamal & LWE)

Chuanqi Zhang

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.
- A public-key encryption scheme is additively homomorphic if there exists a public operation $\circ$ on the ciphertext space such that:

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.
- A public-key encryption scheme is additively homomorphic if there exists a public operation $\circ$ on the ciphertext space such that:

$$D_{sk}(E_{pk}(m_1) \circ E_{pk}(m_2)) = m_1 + m_2.$$

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.
- A public-key encryption scheme is additively homomorphic if there exists a public operation $\circ$ on the ciphertext space such that:

$$D_{sk}(E_{pk}(m_1) \circ E_{pk}(m_2)) = m_1 + m_2.$$

- **Why is this useful by itself?**

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.
- A public-key encryption scheme is additively homomorphic if there exists a public operation $\circ$ on the ciphertext space such that:

$$D_{sk}(E_{pk}(m_1) \circ E_{pk}(m_2)) = m_1 + m_2.$$

- **Why is this useful by itself?**
  - *E-Voting:* Encrypt votes as 0 or 1. Anyone can sum the encrypted votes to get the total, without seeing individual ballots.

Additive Homomorphic Encryption

Before diving into fully homomorphic encryption, we first study schemes that support a single type of operation on ciphertexts. In this video, we focus on addition.

- Let $E_{pk}(\cdot)$ be an encryption algorithm corresponding to the public key pk, and let $D_{sk}(\cdot)$ be an decryption algorithm corresponding to the secret key sk.
- A public-key encryption scheme is additively homomorphic if there exists a public operation $\circ$ on the ciphertext space such that:

$$D_{sk}(E_{pk}(m_1) \circ E_{pk}(m_2)) = m_1 + m_2.$$

- **Why is this useful by itself?**
  - *E-Voting:* Encrypt votes as 0 or 1. Anyone can sum the encrypted votes to get the total, without seeing individual ballots.
  - *Financial Aggregation:* Banks can calculate the total encrypted balances across multiple branches.

Exponential ElGamal: Setup

- Let G be a cyclic group of prime order q with generator g.

Exponential ElGamal: Setup

- Let G be a cyclic group of prime order q with generator g.
- **Key Generation:**

Exponential ElGamal: Setup

- Let G be a cyclic group of prime order q with generator g.
- **Key Generation:**
  - Choose a random secret key sk: $x \xleftarrow{\$} \mathbb{Z}_q$.

Exponential ElGamal: Setup

- Let G be a cyclic group of prime order q with generator g.
- **Key Generation:**
  - Choose a random secret key sk: $x \xleftarrow{\$} \mathbb{Z}_q$.
  - Compute the public key pk: $h = g^x$.

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

$$c_1 = g^r, \quad c_2 = g^m \cdot h^r.$$

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

$$c_1 = g^r, \quad c_2 = g^m \cdot h^r.$$

Decryption $D_{sk}(c_1, c_2)$

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

$$c_1 = g^r, \quad c_2 = g^m \cdot h^r.$$

Decryption $D_{sk}(c_1, c_2)$

1. Use the secret key x to compute the shared mask: $c_1^x = (g^r)^x = (g^x)^r = h^r$.

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

$$c_1 = g^r, \quad c_2 = g^m \cdot h^r.$$

Decryption $D_{sk}(c_1, c_2)$

1. Use the secret key x to compute the shared mask: $c_1^x = (g^r)^x = (g^x)^r = h^r$.
2. Divide c_2 by the mask to isolate g^m:

$$\frac{c_2}{c_1^x} = \frac{g^m \cdot h^r}{h^r} = g^m.$$

Exponential ElGamal: Encryption & Decryption

Encryption $E_{pk}(m)$

To encrypt a message m, choose random $r \xleftarrow{\$} \mathbb{Z}_q$. The ciphertext is a pair (c_1, c_2):

$$c_1 = g^r, \quad c_2 = g^m \cdot h^r.$$

Decryption $D_{sk}(c_1, c_2)$

1. Use the secret key x to compute the shared mask: $c_1^x = (g^r)^x = (g^x)^r = h^r$.
2. Divide c_2 by the mask to isolate g^m:

$$\frac{c_2}{c_1^x} = \frac{g^m \cdot h^r}{h^r} = g^m.$$

3. **The Catch:** m must be chosen from a very small space (e.g., $m \in \{0, 1\}$) so we can brute-force the exponent.

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

$$E(m_1) \otimes E(m_2) = (c_1 \cdot c_1',\ c_2 \cdot c_2')$$

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

$$\begin{aligned} E(m_1) \otimes E(m_2) &= (c_1 \cdot c_1',\ c_2 \cdot c_2') \\ &= (g^{r_1} \cdot g^{r_2},\ g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2}) \end{aligned}$$

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

$$\begin{aligned} E(m_1) \otimes E(m_2) &= (c_1 \cdot c_1',\ c_2 \cdot c_2') \\ &= (g^{r_1} \cdot g^{r_2},\ g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2}) \\ &= (g^{r_1 + r_2},\ g^{m_1 + m_2} h^{r_1 + r_2}) \end{aligned}$$

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

$$\begin{aligned} E(m_1) \otimes E(m_2) &= (c_1 \cdot c_1',\ c_2 \cdot c_2') \\ &= (g^{r_1} \cdot g^{r_2},\ g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2}) \\ &= (g^{r_1 + r_2},\ g^{m_1 + m_2} h^{r_1 + r_2}) \end{aligned}$$

This perfectly matches the structure of a valid encryption of $(m_1 + m_2)$ using combined randomness $(r_1 + r_2)$,

Why is Exponential ElGamal Additively Homomorphic?

Let's prove the additive homomorphism. Suppose we have two ciphertexts:

- $E(m_1) = (c_1, c_2) = (g^{r_1}, g^{m_1} h^{r_1})$
- $E(m_2) = (c_1', c_2') = (g^{r_2}, g^{m_2} h^{r_2})$

The homomorphic operation $\otimes$ is simply **component-wise multiplication**:

$$\begin{aligned} E(m_1) \otimes E(m_2) &= (c_1 \cdot c_1',\ c_2 \cdot c_2') \\ &= (g^{r_1} \cdot g^{r_2},\ g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2}) \\ &= (g^{r_1 + r_2},\ g^{m_1 + m_2} h^{r_1 + r_2}) \end{aligned}$$

This perfectly matches the structure of a valid encryption of $(m_1 + m_2)$ using combined randomness $(r_1 + r_2)$, which means decrypting $E(m_1) \otimes E(m_2)$ gives $(m_1 + m_2)$ correctly.

The Recap of Learning With Errors (LWE)

While ElGamal is beautiful, it cannot be extended to Fully Homomorphic Encryption, and it is vulnerable to quantum computers. Modern FHE is built on **Lattices**, specifically the **LWE problem**.

The Recap of Learning With Errors (LWE)

While ElGamal is beautiful, it cannot be extended to Fully Homomorphic Encryption, and it is vulnerable to quantum computers. Modern FHE is built on **Lattices**, specifically the **LWE problem**.

The LWE Problem (Simplified)

Given a random matrix $\mathbf{A}$ and a vector $\mathbf{b}$, find the secret vector $\mathbf{s}$ such that:

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$$

Where $\mathbf{e}$ is a vector of **small random noise/errors**.

The Recap of Learning With Errors (LWE)

While ElGamal is beautiful, it cannot be extended to Fully Homomorphic Encryption, and it is vulnerable to quantum computers. Modern FHE is built on **Lattices**, specifically the **LWE problem**.

The LWE Problem (Simplified)

Given a random matrix $\mathbf{A}$ and a vector $\mathbf{b}$, find the secret vector $\mathbf{s}$ such that:

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$$

Where $\mathbf{e}$ is a vector of **small random noise/errors**.

The Recap of Learning With Errors (LWE)

While ElGamal is beautiful, it cannot be extended to Fully Homomorphic Encryption, and it is vulnerable to quantum computers. Modern FHE is built on **Lattices**, specifically the **LWE problem**.

The LWE Problem (Simplified)

Given a random matrix $\mathbf{A}$ and a vector $\mathbf{b}$, find the secret vector $\mathbf{s}$ such that:

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$$

Where $\mathbf{e}$ is a vector of **small random noise/errors**.

- Without $\mathbf{e}$, this is just standard linear algebra (which can be solved instantly by Gaussian elimination).

The Recap of Learning With Errors (LWE)

While ElGamal is beautiful, it cannot be extended to Fully Homomorphic Encryption, and it is vulnerable to quantum computers. Modern FHE is built on **Lattices**, specifically the **LWE problem**.

The LWE Problem (Simplified)

Given a random matrix $\mathbf{A}$ and a vector $\mathbf{b}$, find the secret vector $\mathbf{s}$ such that:

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$$

Where $\mathbf{e}$ is a vector of **small random noise/errors**.

- Without $\mathbf{e}$, this is just standard linear algebra (which can be solved instantly by Gaussian elimination).
- With $\mathbf{e}$, this problem is believed to be post-quantum secure.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key sk:

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*:
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: **s**
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: $\mathbf{s}$
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Public Key *pk*:

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: $\mathbf{s}$
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Public Key *pk*:
  - Generate a random matrix $A \xleftarrow{\$} \mathbb{Z}_q^{k \times n}$.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: $\mathbf{s}$
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Public Key *pk*:
  - Generate a random matrix $A \xleftarrow{\$} \mathbb{Z}_q^{k \times n}$.
  - Sample a small error vector $\mathbf{e} \xleftarrow{\$} \mathbb{Z}_q^k$.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: $\mathbf{s}$
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Public Key *pk*:
  - Generate a random matrix $A \xleftarrow{\$} \mathbb{Z}_q^{k \times n}$.
  - Sample a small error vector $\mathbf{e} \xleftarrow{\$} \mathbb{Z}_q^k$.
  - Compute $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$.

Regev's LWE: Key Generation

Now we recall the Regev's public-key encryption scheme based on the LWE problem.

- Secret Key *sk*: $\mathbf{s}$
  - Sample a uniformly random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Public Key *pk*: $(\mathbf{A}, \mathbf{b})$
  - Generate a random matrix $A \xleftarrow{\$} \mathbb{Z}_q^{k \times n}$.
  - Sample a small error vector $\mathbf{e} \xleftarrow{\$} \mathbb{Z}_q^k$.
  - Compute $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$.

Regev's LWE: Encryption

To encrypt a single bit $m \in \{0, 1\}$ using the public key $(\mathbf{A}, \mathbf{b})$:

Regev's LWE: Encryption

To encrypt a single bit $m \in \{0, 1\}$ using the public key $(\mathbf{A}, \mathbf{b})$:

- **Step 1 (Randomizing):** Choose a random small vector $\mathbf{r} \in \mathbb{Z}_q^k$. This $\mathbf{r}$ acts as the ephemeral randomness for the specific encryption every time.

Regev's LWE: Encryption

To encrypt a single bit $m \in \{0, 1\}$ using the public key $(\mathbf{A}, \mathbf{b})$:

- **Step 1 (Randomizing):** Choose a random small vector $\mathbf{r} \in \mathbb{Z}_q^k$. This $\mathbf{r}$ acts as the ephemeral randomness for the specific encryption every time.
- **Step 2 (Masking):** Compute the two parts of the ciphertext $c = (\mathbf{u}, v)$:

Regev's LWE: Encryption

To encrypt a single bit $m \in \{0, 1\}$ using the public key $(\mathbf{A}, \mathbf{b})$:

- **Step 1 (Randomizing):** Choose a random small vector $\mathbf{r} \in \mathbb{Z}_q^k$. This $\mathbf{r}$ acts as the ephemeral randomness for the specific encryption every time.
- **Step 2 (Masking):** Compute the two parts of the ciphertext $c = (\mathbf{u}, v)$:

$$\mathbf{u} = \mathbf{A}^\top \mathbf{r} \pmod{q}$$
$$v = \mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m \pmod{q}$$

Regev's LWE: Encryption

To encrypt a single bit $m \in \{0, 1\}$ using the public key $(\mathbf{A}, \mathbf{b})$:

- **Step 1 (Randomizing):** Choose a random small vector $\mathbf{r} \in \mathbb{Z}_q^k$. This $\mathbf{r}$ acts as the ephemeral randomness for the specific encryption every time.
- **Step 2 (Masking):** Compute the two parts of the ciphertext $c = (\mathbf{u}, v)$:

$$\mathbf{u} = \mathbf{A}^\top \mathbf{r} \pmod{q}$$
$$v = \mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m \pmod{q}$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$v - \mathbf{u}^\top \mathbf{s} = (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s}$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$\begin{aligned} v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\ &= (\mathbf{As} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{As} \quad \text{(Substitute } \mathbf{b}\text{)} \end{aligned}$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$\begin{aligned} v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\ &= (\mathbf{A}\mathbf{s} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \quad \text{(Substitute } \mathbf{b}\text{)} \\ &= (\mathbf{s}^\top \mathbf{A}^\top + \mathbf{e}^\top)\mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \end{aligned}$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$
\begin{aligned}
v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\
&= (\mathbf{A}\mathbf{s} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \quad \text{(Substitute } \mathbf{b}\text{)} \\
&= (\mathbf{s}^\top \mathbf{A}^\top + \mathbf{e}^\top)\mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{s}^\top \mathbf{A}^\top \mathbf{r} + \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s}
\end{aligned}
$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$
\begin{aligned}
v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\
&= (\mathbf{A}\mathbf{s} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \quad \text{(Substitute } \mathbf{b}\text{)} \\
&= (\mathbf{s}^\top \mathbf{A}^\top + \mathbf{e}^\top)\mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{s}^\top \mathbf{A}^\top \mathbf{r} + \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m
\end{aligned}
$$

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$
\begin{aligned}
v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\
&= (\mathbf{A}\mathbf{s} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \quad \text{(Substitute } \mathbf{b}\text{)} \\
&= (\mathbf{s}^\top \mathbf{A}^\top + \mathbf{e}^\top)\mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{s}^\top \mathbf{A}^\top \mathbf{r} + \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m
\end{aligned}
$$

- Let $e' = \mathbf{e}^\top \mathbf{r}$. Since both $\mathbf{e}$ (small noise) and $\mathbf{r}$ (small randomness) are small, their dot product e' is also small (i.e., bounded).

Regev's LWE: Decryption

To decrypt $c = (\mathbf{u}, v)$, the user uses the secret key $\mathbf{s}$ to compute $v - \mathbf{u}^\top \mathbf{s}$. Let's expand this:

$$
\begin{aligned}
v - \mathbf{u}^\top \mathbf{s} &= (\mathbf{b}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m) - (\mathbf{A}^\top \mathbf{r})^\top \mathbf{s} \\
&= (\mathbf{A}\mathbf{s} + \mathbf{e})^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \quad \text{(Substitute } \mathbf{b}\text{)} \\
&= (\mathbf{s}^\top \mathbf{A}^\top + \mathbf{e}^\top)\mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{s}^\top \mathbf{A}^\top \mathbf{r} + \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m - \mathbf{r}^\top \mathbf{A}\mathbf{s} \\
&= \mathbf{e}^\top \mathbf{r} + \lceil \frac{q}{2} \rfloor m
\end{aligned}
$$

- Let $e' = \mathbf{e}^\top \mathbf{r}$. Since both $\mathbf{e}$ (small noise) and $\mathbf{r}$ (small randomness) are small, their dot product e' is also small (i.e., bounded).
- The result is $e' + \lceil \frac{q}{2} \rfloor m$. We simply round to 0 or $q/2$ to recover m. The noise e' is discarded!

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2.

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} = (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s})$$

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$\begin{aligned}(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} &= (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s}) \\ &= (e_1' + \lceil \frac{q}{2} \rfloor m_1) + (e_2' + \lceil \frac{q}{2} \rfloor m_2)\end{aligned}$$

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$\begin{aligned}(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} &= (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s}) \\ &= (e_1' + \lceil \frac{q}{2} \rfloor m_1) + (e_2' + \lceil \frac{q}{2} \rfloor m_2) \\ &= (e_1' + e_2') + \lceil \frac{q}{2} \rfloor (m_1 + m_2)\end{aligned}$$

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$\begin{aligned}(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} &= (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s}) \\ &= (e_1' + \lceil \frac{q}{2} \rfloor m_1) + (e_2' + \lceil \frac{q}{2} \rfloor m_2) \\ &= (e_1' + e_2') + \lceil \frac{q}{2} \rfloor (m_1 + m_2)\end{aligned}$$

The Crucial Insight: It works! BUT the noise grows.

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$\begin{aligned}(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} &= (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s}) \\ &= (e_1' + \lceil \frac{q}{2} \rfloor m_1) + (e_2' + \lceil \frac{q}{2} \rfloor m_2) \\ &= (e_1' + e_2') + \lceil \frac{q}{2} \rfloor (m_1 + m_2)\end{aligned}$$

The Crucial Insight: It works! BUT the noise grows. The new error is $(e_1' + e_2')$. If we add too many times, the accumulated noise will bleed into the message bits, causing a decryption failure.

Why is Regev's LWE Additively Homomorphic?

Let $c_1 = (\mathbf{u}_1, v_1)$ encrypt m_1, and $c_2 = (\mathbf{u}_2, v_2)$ encrypt m_2. Homomorphic addition is just **vector addition** $\oplus$ modulo q:

$$c_1 \oplus c_2 = (\mathbf{u}_1 + \mathbf{u}_2,\ v_1 + v_2)$$

Let's try to decrypt $c_1 \oplus c_2$ to get $m_1 + m_2$:

$$\begin{aligned}(v_1 + v_2) - (\mathbf{u}_1 + \mathbf{u}_2)^\top \mathbf{s} &= (v_1 - \mathbf{u}_1^\top \mathbf{s}) + (v_2 - \mathbf{u}_2^\top \mathbf{s}) \\ &= (e_1' + \lceil \frac{q}{2} \rfloor m_1) + (e_2' + \lceil \frac{q}{2} \rfloor m_2) \\ &= (e_1' + e_2') + \lceil \frac{q}{2} \rfloor (m_1 + m_2)\end{aligned}$$

The Crucial Insight: It works! BUT the noise grows. The new error is $(e_1' + e_2')$. If we add too many times, the accumulated noise will bleed into the message bits, causing a decryption failure. Such noise growth is the central challenge of FHE.

How Secure is Homomorphic Encryption (HE)?

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise.

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)
   - *The Paradox:* The whole magic of HE is allowing computations on locked data. But this means the data is "malleable".

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)
   - *The Paradox:* The whole magic of HE is allowing computations on locked data. But this means the data is "malleable".
   - Hackers cannot read your data, but they can blindly modify it!

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)
   - *The Paradox:* The whole magic of HE is allowing computations on locked data. But this means the data is "malleable".
   - Hackers cannot read your data, but they can blindly modify it!
   - *Example:* A hacker intercepts an encrypted \$1000. Without decryption, they can blindly minus an encrypted \$900 to it. The receiver will decrypt to see only \$100!

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)
   - *The Paradox:* The whole magic of HE is allowing computations on locked data. But this means the data is "malleable".
   - Hackers cannot read your data, but they can blindly modify it!
   - *Example:* A hacker intercepts an encrypted \$1000. Without decryption, they can blindly minus an encrypted \$900 to it. The receiver will decrypt to see only \$100! Active attackers can forge outcomes. ✗

How Secure is Homomorphic Encryption (HE)?

1. **The Good: Complete Privacy** (IND-CPA Security)
   - Hackers cannot read your data.
   - Even if a hacker intercepts a message, it is guaranteed to look like random noise. Passive eavesdroppers learn nothing. ✓
2. **The Bad: Anyone Can Alter It** (NO IND-CCA Security)
   - *The Paradox:* The whole magic of HE is allowing computations on locked data. But this means the data is "malleable".
   - Hackers cannot read your data, but they can blindly modify it!
   - *Example:* A hacker intercepts an encrypted \$1000. Without decryption, they can blindly minus an encrypted \$900 to it. The receiver will decrypt to see only \$100! Active attackers can forge outcomes. ✗

Conclusion

To achieve both privacy and integrity in the real world, homomorphic ciphertexts must be wrapped in Zero-Knowledge Proofs (ZKPs) or Message Authentication Codes (MACs).

Summary

1. What is Additive Homomorphic Encryption (AHE)?
2. Exponential ElGamal: Group-Based AHE
3. Regev's LWE: Lattice-Based AHE