Homomorphic Encryption and Bootstrapping

Njaka Andriamandratomanana, Elie Chedemail, Adéchola Kouande, Rémi Leluc, Thi Thu Quyen Nguyen

Supervisors: Florian Méhats, Mohammed Lemou, Philippe Chartier

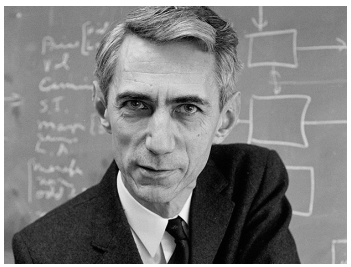
SEME Rennes, May 02 - May 06, 2022

Introduction

- Homomorphic Encryption
- Learning With Errors (LWE) encryption/decryption
- RLWE encryption
- Bootstrap

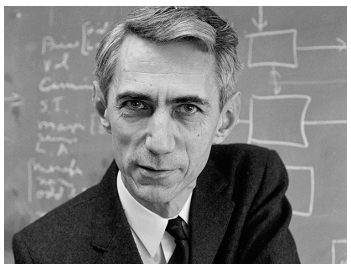
Cryptography in History

- 1900-1950: **Symmetric** Encryption (same key) (Enigma, Shannon Information theory, ...)



Cryptography in History

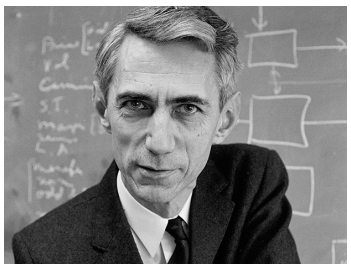
- 1900-1950: **Symmetric** Encryption (same key) (Enigma, Shannon Information theory, ...)



- 1970s: **Asymmetric** Encryption (encryption with *public key*, decryption with *private* key)

Cryptography in History

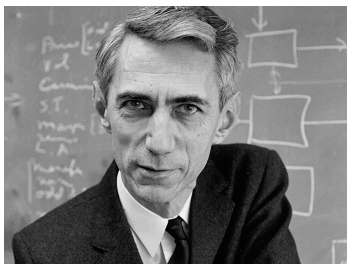
- 1900-1950: **Symmetric** Encryption (same key) (Enigma, Shannon Information theory, ...)



- 1970s: **Asymmetric** Encryption (encryption with *public key*, decryption with *private* key)
- Only thing people did with crypted data ... was decrypt it...

Cryptography in History

- 1900-1950: **Symmetric** Encryption (same key) (Enigma, Shannon Information theory, ...)



- 1970s: **Asymmetric** Encryption (encryption with *public key*, decryption with *private* key)
- Only thing people did with crypted data ... was decrypt it...
- Today (big data era), we want to perform (cloud) computing on encrypted data, *e.g.* for machine learning applications
- Need to be able to perform **sums** and **products** on encrypted data

Homomorphic Encryption

Let φ be an encryption:

$$\varphi : (\{\mathit{plaintext}\}, +, \times) \longrightarrow (\{\mathit{ciphertext}\}, \oplus, \otimes)$$

Homomorphic Encryption

Let φ be an encryption:

$$\varphi : (\{\mathit{plaintext}\}, +, \times) \longrightarrow (\{\mathit{ciphertext}\}, \oplus, \otimes)$$

- **Homomorphic encryption** (HE) preserves either addition **or** multiplication of two messages,

Homomorphic Encryption

Let φ be an encryption:

$$\varphi : (\{\mathit{plaintext}\}, +, \times) \longrightarrow (\{\mathit{ciphertext}\}, \oplus, \otimes)$$

- **Homomorphic encryption** (HE) preserves either addition **or** multiplication of two messages, ie.

$$\varphi(m_1 + m_2) = \varphi(m_1) \oplus \varphi(m_2), \quad \text{or}$$
$$\varphi(m_1 \times m_2) = \varphi(m_1) \otimes \varphi(m_2).$$

Homomorphic Encryption

Let φ be an encryption:

$$\varphi : (\{\mathit{plaintext}\}, +, \times) \longrightarrow (\{\mathit{ciphertext}\}, \oplus, \otimes)$$

- **Homomorphic encryption** (HE) preserves either addition **or** multiplication of two messages, ie.

$$\varphi(m_1 + m_2) = \varphi(m_1) \oplus \varphi(m_2), \quad \text{or}$$
$$\varphi(m_1 \times m_2) = \varphi(m_1) \otimes \varphi(m_2).$$

- **Fully homomorphic encryption** (FHE) preserves both addition **and** multiplication.

Learning With Errors (LWE) encryption/decryption

• For $n, q, t \in \mathbb{N}^*$ with $t|q$ and a message $m \in \mathbb{Z}_t \subset \mathbb{Z}_q$, the **LWE encryption** with the key $\vec{s} \leftarrow \chi_{key}(\mathbb{Z}_q^n)$ of m is defined as:

$$LWE_{q,\vec{s}}(m) := (\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + \tilde{m} + e) \in \mathbb{Z}_q^{n+1},$$

where $\vec{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi_{error}(\mathbb{Z}_q)$ and $\tilde{m} = \frac{q}{t} m$.

Learning With Errors (LWE) encryption/decryption

- For $n, q, t \in \mathbb{N}^*$ with $t|q$ and a message $m \in \mathbb{Z}_t \subset \mathbb{Z}_q$, the **LWE encryption** with the key $\vec{s} \leftarrow \chi_{key}(\mathbb{Z}_q^n)$ of m is defined as:

$$LWE_{q,\vec{s}}(m) := (\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + \tilde{m} + e) \in \mathbb{Z}_q^{n+1},$$

where $\vec{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi_{error}(\mathbb{Z}_q)$ and $\tilde{m} = \frac{q}{t} m$.

- The **decryption** of a ciphertext $(\vec{a}, b)$ of m is

$$LWE^{-1}(\vec{a}, b) := \lceil \frac{t}{q}(b - \langle \vec{a}, \vec{s} \rangle) \rfloor \in \mathbb{Z}_q.$$

Learning With Errors (LWE) encryption/decryption

- Let $Err_{LWE}((\vec{a}, b), m) = \frac{t}{q}(b - \langle \vec{a}, \vec{s} \rangle) - m = \frac{t}{q}e$. If $|\frac{t}{q}e| \in [0, 1/2]$ then $LWE^{-1}(\vec{a}, b) = m \rightarrow$ Successful decryption.

Learning With Errors (LWE) encryption/decryption

- Let $Err_{LWE}((\vec{a}, b), m) = \frac{t}{q}(b - \langle \vec{a}, \vec{s} \rangle) - m = \frac{t}{q}e$. If $|\frac{t}{q}e| \in [0, 1/2]$ then $LWE^{-1}(\vec{a}, b) = m \rightarrow$ Successful decryption.

On the other hand we have,

$$
\begin{aligned}
&LWE_{q,\vec{s}}(m_1) + LWE_{q,\vec{s}}(m_2) \\
&\qquad = (\vec{a}_1, b_1) + (\vec{a}_2, b_2) = (\vec{a}_1 + \vec{a}_2, b_1 + b_2) \\
&\qquad = (\vec{a}_1 + \vec{a}_2, \langle \vec{a}_1 + \vec{a}_2, s \rangle + (\tilde{m}_1 + \tilde{m}_2) + \underbrace{(e_1 + e_2)}_{\text{sum noise}}) \\
&\qquad = LWE_{q,\vec{s}}(m_1 + m_2).
\end{aligned}
$$

Learning With Errors (LWE) encryption/decryption

• Let $Err_{LWE}((\vec{a}, b), m) = \frac{t}{q}(b - \langle \vec{a}, \vec{s} \rangle) - m = \frac{t}{q}e$. If $|\frac{t}{q}e| \in [0, 1/2]$ then $LWE^{-1}(\vec{a}, b) = m \rightarrow$ Successful decryption.

On the other hand we have,

$$\begin{aligned} LWE_{q,\vec{s}}(m_1) + LWE_{q,\vec{s}}(m_2) & \\ &= (\vec{a}_1, b_1) + (\vec{a}_2, b_2) = (\vec{a}_1 + \vec{a}_2, b_1 + b_2) \\ &= (\vec{a}_1 + \vec{a}_2, \langle \vec{a}_1 + \vec{a}_2, s \rangle + (\tilde{m}_1 + \tilde{m}_2) + \underbrace{(e_1 + e_2)}_{\text{sum noise}}) \\ &= LWE_{q,\vec{s}}(m_1 + m_2). \end{aligned}$$

•LWE encryption is **homomorphic** at the cost of accumulated noise which can be reduced with **bootstrapping**.

RLWE encryption: work with polynomials

• For $N, Q, t \in \mathbb{N}^*$ with $t|q$ and a message $\boldsymbol{m}(X) \in \mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$ with $\mathcal{R} = \mathbb{Z}[X]/\left(X^N + 1\right)$, the **RLWE encryption** with the key $\boldsymbol{z}(X) \leftarrow \chi_{key}(\mathcal{R}_Q)$ of $\boldsymbol{m} \in \mathcal{R}_t$ is defined as:

$$RLWE_{Q,\boldsymbol{z}}(\boldsymbol{m}) := (\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{z} + \tilde{\boldsymbol{m}} + \boldsymbol{e}) \in \mathcal{R}_Q^2,$$

where $\boldsymbol{a} \leftarrow \mathcal{R}_Q$, error $\boldsymbol{e} \leftarrow \chi_{error}(\mathcal{R}_Q)$ and $\tilde{\boldsymbol{m}} = \frac{Q}{t}\boldsymbol{m}$. RLWE encryption is linearly homomorphic.

Multiplicative RLWE encryption

To **multiply** by arbitrary ring elements, we introduce the encryption scheme

$$RLWE'_{\boldsymbol{z}}(\boldsymbol{m}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m}, RLWE_{\boldsymbol{z}}(B\boldsymbol{m}), \ldots, RLWE_{\boldsymbol{z}}(B^{k-1}\boldsymbol{m})).$$

Multiplicative RLWE encryption

To **multiply** by arbitrary ring elements, we introduce the encryption scheme

$$RLWE'_{\boldsymbol{z}}(\boldsymbol{m}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m}, RLWE_{\boldsymbol{z}}(B\boldsymbol{m}), \ldots, RLWE_{\boldsymbol{z}}(B^{k-1}\boldsymbol{m})).$$

Finally, we define the encryption scheme

$$RGSW_{\boldsymbol{z}}(\boldsymbol{m}) = (RLWE'_{\boldsymbol{z}}(-\boldsymbol{z} \cdot \boldsymbol{m}), RLWE'_{\boldsymbol{z}}(\boldsymbol{m})).$$

Multiplicative RLWE encryption

To **multiply** by arbitrary ring elements, we introduce the encryption scheme

$$RLWE'_{\boldsymbol{z}}(\boldsymbol{m}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m}, RLWE_{\boldsymbol{z}}(B\boldsymbol{m}), \ldots, RLWE_{\boldsymbol{z}}(B^{k-1}\boldsymbol{m})).$$

Finally, we define the encryption scheme

$$RGSW_{\boldsymbol{z}}(\boldsymbol{m}) = (RLWE'_{\boldsymbol{z}}(-\boldsymbol{z} \cdot \boldsymbol{m}), RLWE'_{\boldsymbol{z}}(\boldsymbol{m})).$$

We have the multiplication operation

$$\diamond : RLWE \times RGSW \longrightarrow RLWE$$

Multiplicative RLWE encryption

To **multiply** by arbitrary ring elements, we introduce the encryption scheme

$$RLWE'_{\boldsymbol{z}}(\boldsymbol{m}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m}, RLWE_{\boldsymbol{z}}(B\boldsymbol{m}), \ldots, RLWE_{\boldsymbol{z}}(B^{k-1}\boldsymbol{m})).$$

Finally, we define the encryption scheme

$$RGSW_{\boldsymbol{z}}(\boldsymbol{m}) = (RLWE'_{\boldsymbol{z}}(-\boldsymbol{z} \cdot \boldsymbol{m}), RLWE'_{\boldsymbol{z}}(\boldsymbol{m})).$$

We have the multiplication operation

$$\diamond : RLWE \times RGSW \longrightarrow RLWE$$

Lemma

For two messages $\boldsymbol{m_0}, \boldsymbol{m_1} \in \mathcal{R}_Q$ with $\boldsymbol{m_1}$ small, we have:

$$RLWE_{\boldsymbol{z}}(\boldsymbol{m_0}) \diamond RGSW_{\boldsymbol{z}}(\boldsymbol{m_1}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m_0} \cdot \boldsymbol{m_1})$$

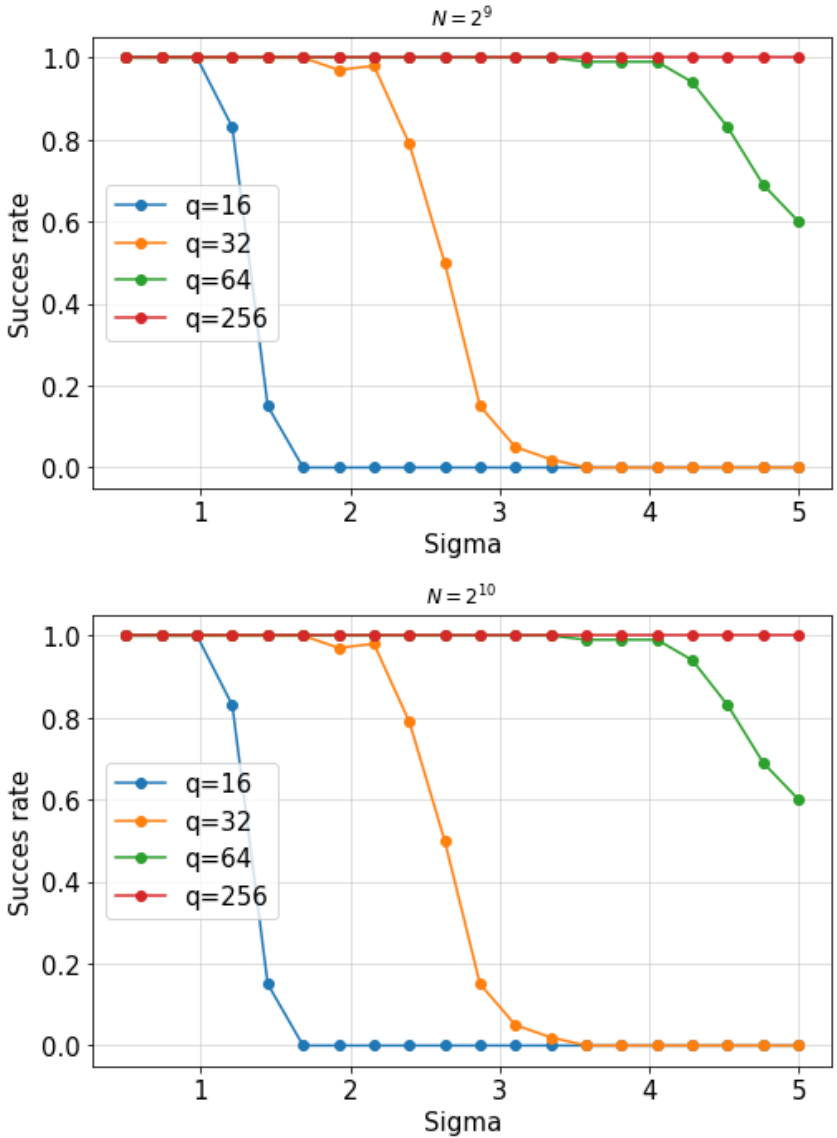
Numerical implementation

- Implement functions RLWE, RLWE^{-1}, RGSW, $\diamond$

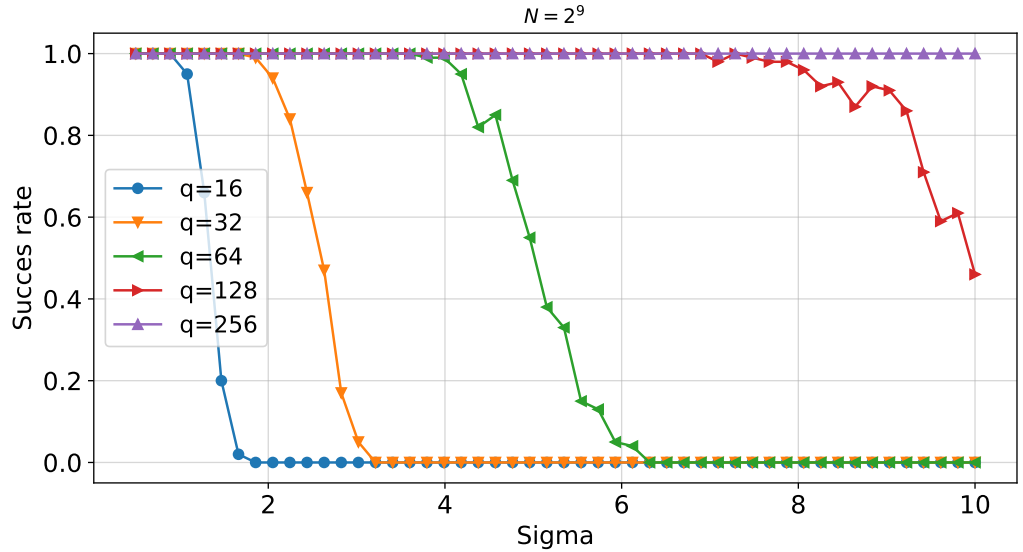
- Need to work on $R_q = \mathbb{Z}_q[X]/(X^N + 1)$, implement product and tensor product of matrices of polynomials (links with FFT).

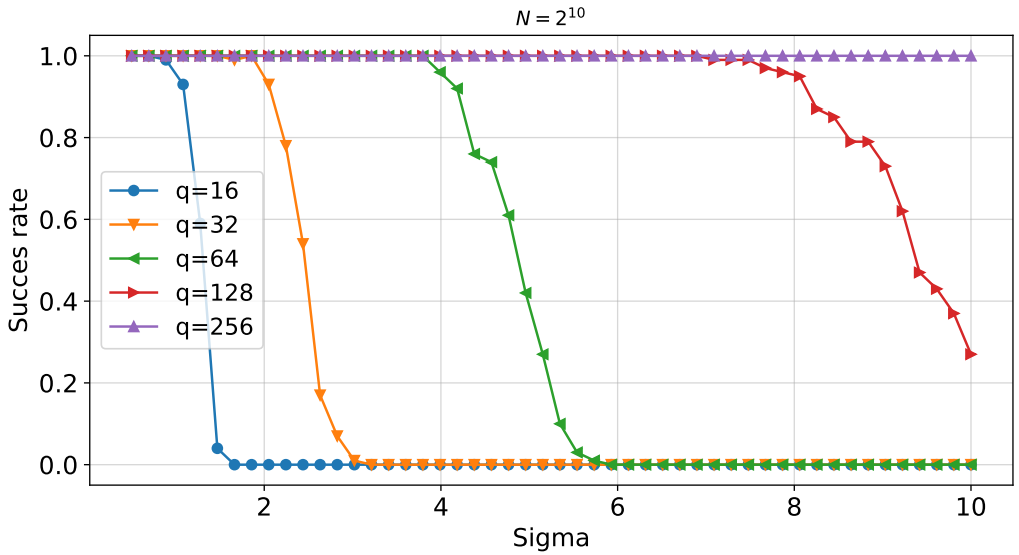
- Test code/decode functions on $N_{exp} = 100$ independent runs and test the evolution of the succes rate recovery with different parameters $N \in \{2^9; 2^{10}\}$ and $q \in \{2^4; 2^5; 2^6; 2^7; 2^8\}$ and $\sigma \in [0; 10]$, to see the effect of noise

Experiments code/decode RLWE, $N \in \{2^9, 2^{10}\}$, and $\sigma = 0.5 : 5$

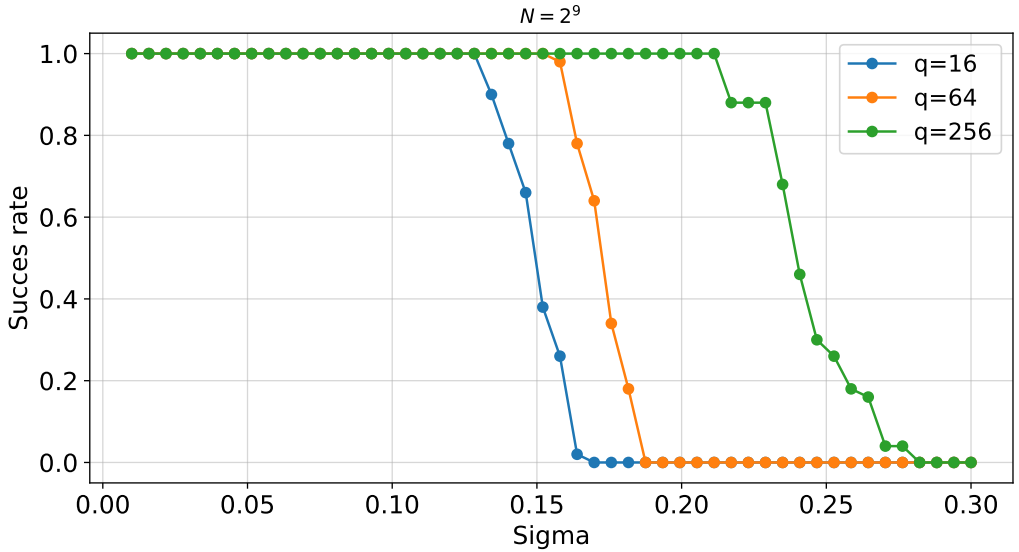


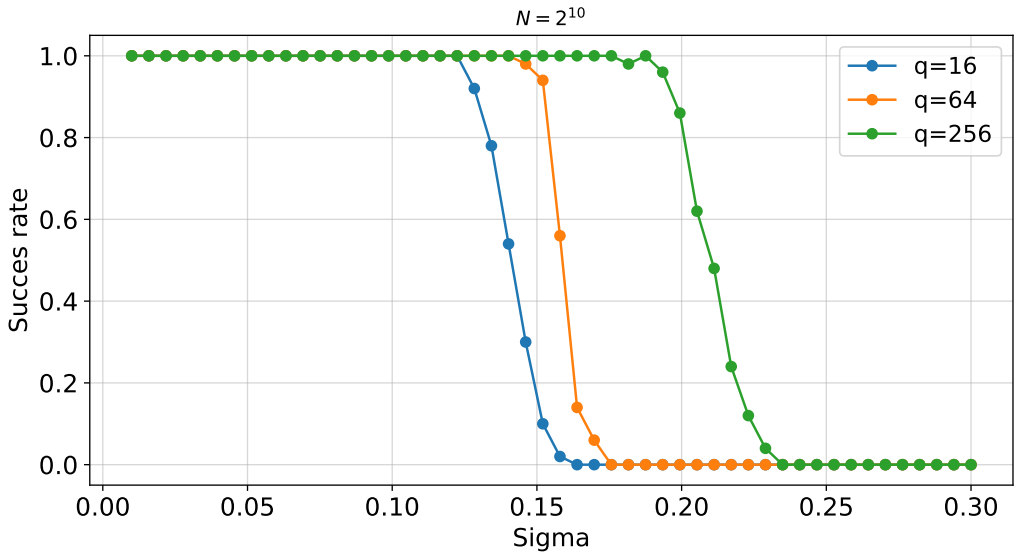
Experiments Sum RLWE, $N \in \{2^9, 2^{10}\}$ and $\sigma = 0.5 : 10$





Experiments Prod RLWE, $N \in \{2^9, 2^{10}\}$ and $\sigma = 0.01 : 0.3$





Hint of Bootstrap: Trick of working in $\mathbb{Z}_q[X]/\left(X^N+1\right)$

For a polynomial $w(X) = \sum_{i=0}^{N-1} w_i X^i$ and any $u \in \mathbb{Z}_q$, we have

$$coeff_0(X^{-u} w(X)) = coeff_u(w(X)) = w_u$$

Hint of Bootstrap: Trick of working in $\mathbb{Z}_q[X]/\left(X^N+1\right)$

For a polynomial $w(X) = \sum_{i=0}^{N-1} w_i X^i$ and any $u \in \mathbb{Z}_q$, we have

$$coeff_0(X^{-u} w(X)) = coeff_u(w(X)) = w_u$$

Recall that $b - \langle \vec{a}, \vec{s} \rangle = \tilde{m} + e$ so that with $u = b - \langle \vec{a}, \vec{s} \rangle$

$$coeff_0(X^{-(b-\langle \vec{a}, \vec{s} \rangle)} w(X)) = w_{\tilde{m}+e} = w_{\tilde{m}}$$

as soon as $w(X)$ is well-chosen with coefficients equal on subsets.

Hint of Bootstrap: Trick of working in $\mathbb{Z}_q[X]/\left(X^N+1\right)$

For a polynomial $w(X) = \sum_{i=0}^{N-1} w_i X^i$ and any $u \in \mathbb{Z}_q$, we have

$$coeff_0(X^{-u} w(X)) = coeff_u(w(X)) = w_u$$

Recall that $b - \langle \vec{a}, \vec{s} \rangle = \tilde{m} + e$ so that with $u = b - \langle \vec{a}, \vec{s} \rangle$

$$coeff_0(X^{-(b-\langle \vec{a}, \vec{s} \rangle)} w(X)) = w_{\tilde{m}+e} = w_{\tilde{m}}$$

as soon as $w(X)$ is well-chosen with coefficients equal on subsets.
GOAL: write this formula with ciphertext $\rightarrow$ bootstrap scheme
• Of course you cannot reveal the secret key s... but you can publicly reveal an encrpyption $Enc_{s'}(s)$ of it !
• Bootstrap = Use Encryption(key_s) + tricks with polynomials

Bootstrap

This is the bootstrap for noise reduction in LWE encryption.

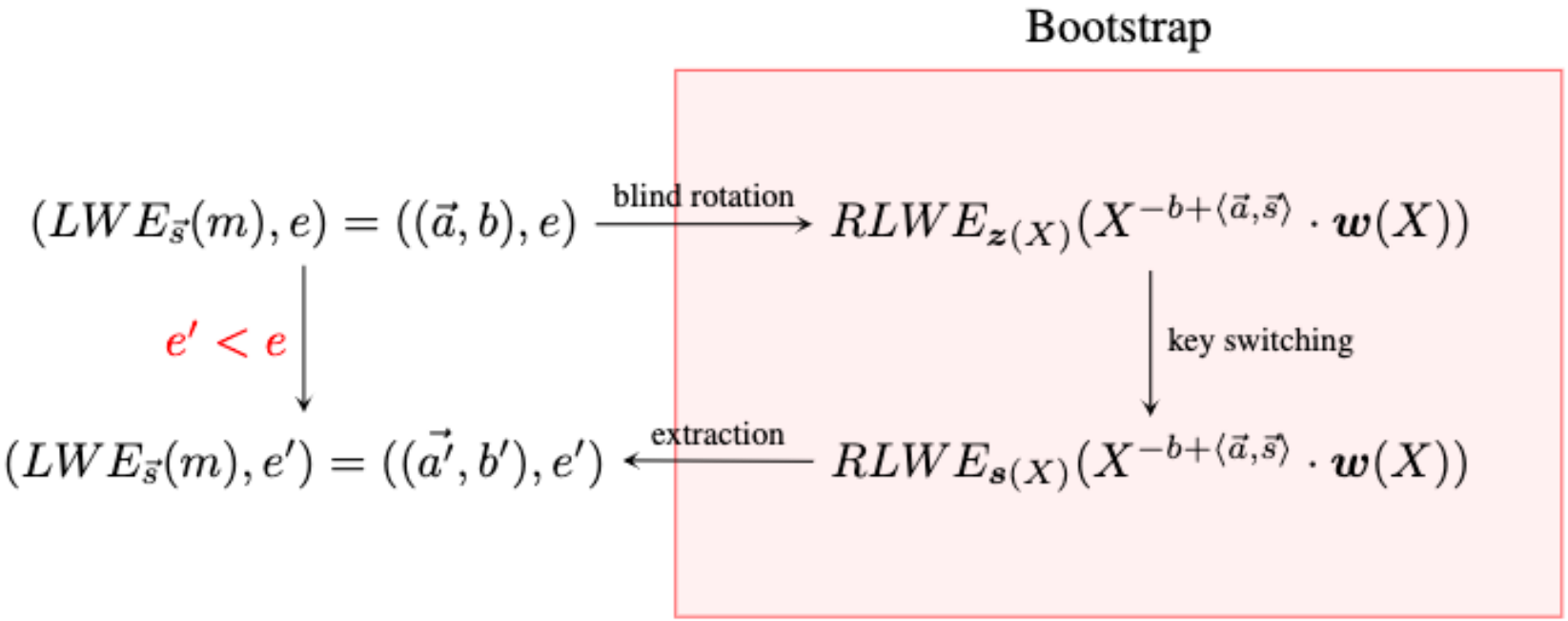


Figure: Bootstrapping steps

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$(\vec{a}, b) = LWE_s(m) \rightarrow$

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \rightarrow$$

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \rightarrow (a_s(X), b_s(X)) = RLWE_{s(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 3: Extraction**

$$(a_s(X), b_s(X)) \rightarrow$$

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \rightarrow (a_s(X), b_s(X)) = RLWE_{s(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 3: Extraction**

$$(a_s(X), b_s(X)) \rightarrow (\vec{a}, b) = LWE_s(m)$$

OUTPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error $e' < e$

Step 1: Blind rotation

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

Step 1: Blind rotation

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

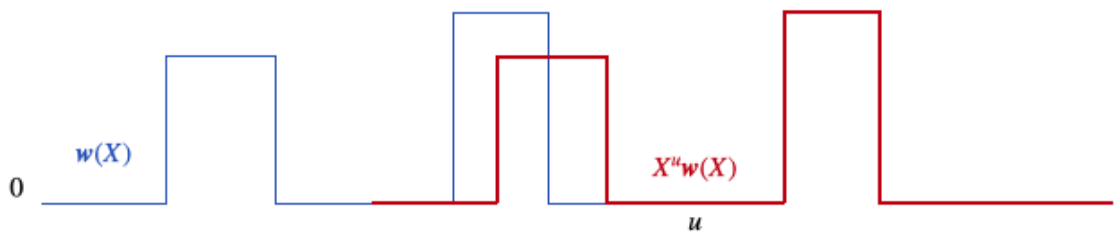


Figure: Blind Rotation $X^u \boldsymbol{w}(X)$

- $u = -b + \langle \vec{a}, \vec{s} \rangle$.

Step 1: Blind rotation

$$(\vec{a}, b) = LWE_s(m) \to (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

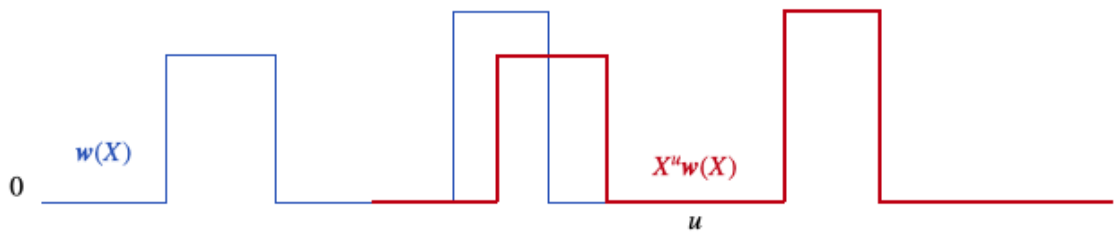


Figure: Blind Rotation $X^u \boldsymbol{w}(X)$

- $u = -b + \langle \vec{a}, \vec{s} \rangle$.
- Since $\langle \vec{a}, \vec{s} \rangle = \sum_i a_i s_i$, $X^{\langle \vec{a}, \vec{s} \rangle} = \prod_i X^{a_i s_i}$,

$RLWE_z(X^{-b+\langle \vec{a}, \vec{s} \rangle} \boldsymbol{w}) = RLWE_z(X^{-b} \boldsymbol{w})$

Step 1: Blind rotation

$$(\vec{a}, b) = LWE_s(m) \to (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

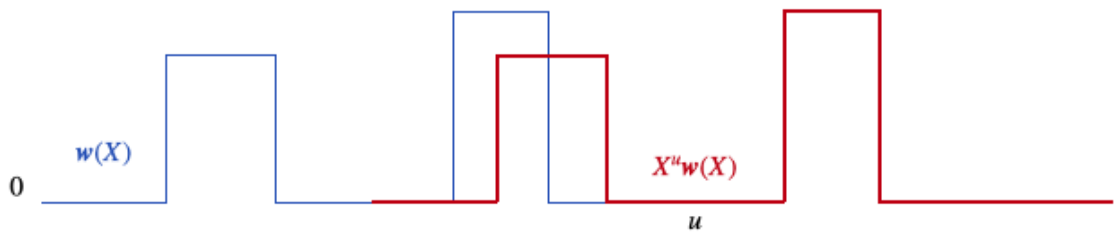


Figure: Blind Rotation $X^u \boldsymbol{w}(X)$

- $u = -b + \langle \vec{a}, \vec{s} \rangle$.
- Since $\langle \vec{a}, \vec{s} \rangle = \sum_i a_i s_i$, $X^{\langle \vec{a}, \vec{s} \rangle} = \prod_i X^{a_i s_i}$,

$$RLWE_{\boldsymbol{z}}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \boldsymbol{w}) = RLWE_{\boldsymbol{z}}(X^{-b} \boldsymbol{w}) \diamond RGSW_{\boldsymbol{z}}(X^{a_0 s_0}) \cdots \diamond RGSW_{\boldsymbol{z}}(X^{a_{n-1} s_{n-1}})$$

Step 1: Blind rotation

$$(\vec{a}, b) = LWE_s(m) \to (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

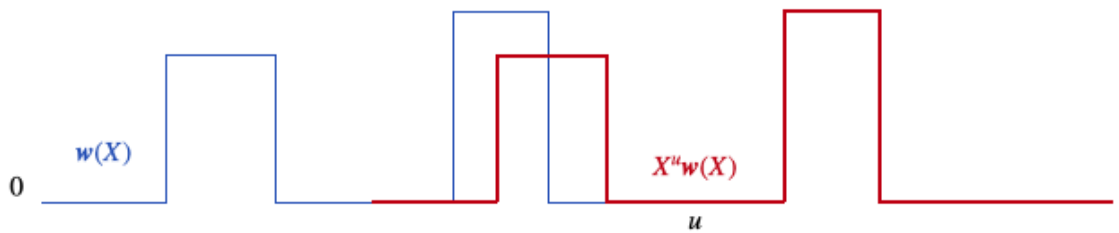


Figure: Blind Rotation $X^u \boldsymbol{w}(X)$

- $u = -b + \langle \vec{a}, \vec{s} \rangle$.
- Since $\langle \vec{a}, \vec{s} \rangle = \sum_i a_i s_i$, $X^{\langle \vec{a}, \vec{s} \rangle} = \prod_i X^{a_i s_i}$,

$$RLWE_{\boldsymbol{z}}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \boldsymbol{w}) = RLWE_{\boldsymbol{z}}(X^{-b} \boldsymbol{w}) \diamond RGSW_{\boldsymbol{z}}(X^{a_0 s_0}) \cdots \diamond RGSW_{\boldsymbol{z}}(X^{a_{n-1} s_{n-1}})$$

- The **Bootstrap Keys** are publicly available

$$BK_{i,j} = RGSW_{\boldsymbol{z}}(X^{j s_i}), \quad 0 \leq i \leq n, \quad 0 \leq j < q$$

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i,j} = RGSW_{\boldsymbol{z}}(X^{js_i}), \quad 0 \leq i \leq n, \quad 0 \leq j < q$$

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i,j} = RGSW_{\boldsymbol{z}}(X^{js_i}), \quad 0 \leq i \leq n, \quad 0 \leq j < q$$

- $X^{\langle \vec{a}, \vec{s} \rangle} = \prod_i X^{a_i s_i} = \prod_{i,j} X^{a_{i,j} B^j s_i}$ with $a_i = \sum_j a_{i,j} B^j$

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i,j} = RGSW_{\boldsymbol{z}}(X^{js_i}), \quad 0 \leq i \leq n, \quad 0 \leq j < q$$

- $X^{\langle \vec{a}, \vec{s} \rangle} = \prod_i X^{a_i s_i} = \prod_{i,j} X^{a_{i,j} B^j s_i}$ with $a_i = \sum_j a_{i,j} B^j$

$$BK_{i,j,l} = RGSW_{\boldsymbol{z}}(X^{lB^j s_i}), \quad 0 \leq i \leq n,\ 0 \leq j < \log_B(q),\ 0 \leq l < B.$$

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \rightarrow (a_s(X), b_s(X)) = RLWE_{s(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 3: Extraction**

$$(a_s(X), b_s(X)) \rightarrow (\vec{a}, b) = LWE_s(m)$$

OUTPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error $e' < e$

Step 2: Key switching

Key switching operation converts a ciphertext $RLWE_{\mathbf{z}_1}(\mathbf{m})$ of a message $\mathbf{m} \in \mathcal{R}_Q$ encrypted by a secret key $\mathbf{z}_1 \in R_Q$ to a ciphertext $RLWE_{\mathbf{z}_2}(\mathbf{m})$ encrypted by a new secret key $\mathbf{z}_2 \in \mathcal{R}_Q$.

Algorithm 1: Key Switching in RLWE

Inputs : $(\mathbf{a}, \mathbf{b}) = RLWE_{\mathbf{z}_1}(\mathbf{m}), \mathbf{z}_2$

Outputs: $(\mathbf{a}', \mathbf{b}') = RLWE_{\mathbf{z}_2}(\mathbf{m})$

1 Compute $RLWE_{\mathbf{z}_2}(B^j \cdot \mathbf{z}_1) =: (KS_j^{\mathbf{a}}, KS_j^{\mathbf{b}})$ for $j = 0, \cdots, K := \log_B Q - 1$

2 Write $\mathbf{a} = \sum_{j=0}^{K} \alpha_j B^j$ where $\alpha_j = \sum_{i=0}^{N-1} a_{i,j} X^i$ such that $|a_{i,j}| \leq B - 1$

3 Return $(\mathbf{a}', \mathbf{b}') = \left(-\sum_{j=1}^{K} \alpha_j \cdot KS_j^{\mathbf{a}}, \mathbf{b} - \sum_{j=1}^{K} \alpha_j \cdot KS_j^{\mathbf{b}} \right)$

Bootstrap Procedure

INPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error e

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \rightarrow (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \rightarrow (a_s(X), b_s(X)) = RLWE_{s(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 3: Extraction**

$$(a_s(X), b_s(X)) \rightarrow (\vec{a}, b) = LWE_s(m)$$

OUTPUT: Cipher $LWE_s(m) = (\vec{a}, b)$ with error $e' < e$

Step 3: Extraction

Let $RLWE_{\boldsymbol{s}(X)}(\boldsymbol{m}(X)) = (\boldsymbol{a}(X), \boldsymbol{b}(X)) \in \mathcal{R}_Q$. We have

$$\boldsymbol{b}(X) = \boldsymbol{a}(X) \cdot \boldsymbol{s}(X) + \tilde{\boldsymbol{m}}(X) + \boldsymbol{e}(X)$$

Step 3: Extraction

Let $RLWE_{\boldsymbol{s}(X)}(\boldsymbol{m}(X)) = (\boldsymbol{a}(X), \boldsymbol{b}(X)) \in \mathcal{R}_Q$. We have

$$\boldsymbol{b}(X) = \boldsymbol{a}(X) \cdot \boldsymbol{s}(X) + \tilde{\boldsymbol{m}}(X) + \boldsymbol{e}(X)$$

At the i^{th} coefficient we have:

$$\begin{aligned} b_i &= \sum_{k=0}^{N-1} a_{i-k} s_k + \tilde{m}_i + e_i \\ &= \langle \iota_i(\boldsymbol{a}), s \rangle + \tilde{m}_i + e_i \end{aligned}$$

where $\iota_i(\boldsymbol{a})$ is the i^{th} anti-cyclic permutation of coefficients of $\boldsymbol{a}$.

Step 3: Extraction

Let $RLWE_{\boldsymbol{s}(X)}(\boldsymbol{m}(X)) = (\boldsymbol{a}(X), \boldsymbol{b}(X)) \in \mathcal{R}_Q$. We have

$$\boldsymbol{b}(X) = \boldsymbol{a}(X) \cdot \boldsymbol{s}(X) + \tilde{\boldsymbol{m}}(X) + \boldsymbol{e}(X)$$

At the i^{th} coefficient we have:

$$\begin{aligned} b_i &= \sum_{k=0}^{N-1} a_{i-k} s_k + \tilde{m}_i + e_i \\ &= \langle \iota_i(\boldsymbol{a}), s \rangle + \tilde{m}_i + e_i \end{aligned}$$

where $\iota_i(\boldsymbol{a})$ is the i^{th} anti-cyclic permutation of coefficients of $\boldsymbol{a}$. Since $\boldsymbol{e}(X)$ is small, so is e_i, thus $(\iota_i(\boldsymbol{a}), b_i)$ is a LWE_s encryption of m.

Conclusion and Bootstrap Today

Figure: Craig Gentry (father of Bootstrap !)

- 2009: Craig Gentry Phd dissertation on Bootstrap for FHE

Conclusion and Bootstrap Today

Figure: Craig Gentry (father of Bootstrap !)

- 2009: Craig Gentry Phd dissertation on Bootstrap for FHE
- 2014: Léo Ducas and Daniele Micciancio *"Bootstrapping Homomorphic Encryption in less than a second"*
- 2017: Chilloti et al. obtained speed up from less than 1 s to **less than 0.1 s** + reduce the 1 GB bootstrapping key size to 24 MB, preserving the same security levels.
- New perspectives: parallel and distributed computing, bootstrap on GPU's (2021), faster and faster ... FHE is active field of research!