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 $'$ be an encryption:

$$' : (\mathit{fplaintext}g; +; \quad) \quad ! \quad (\mathit{fciphertext}g; \quad ; \quad)$$

Homomorphic Encryption

Let $'$ be an encryption:

$$' : (fplaintextg; +; \quad) \quad ! \quad (fciphertextg; \quad ; \quad)$$

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

Homomorphic Encryption

Let $'$ be an encryption:

$$' : (fplaintextg; +; \quad) \quad ! \quad (fciphertextg; \quad ; \quad)$$

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

$$'(m_1 + m_2) = '(m_1) \quad '(m_2); \quad \text{or}$$
$$'(m_1 \quad m_2) = '(m_1) \quad '(m_2):$$

Homomorphic Encryption

Let $'$ be an encryption:

$$' : (\mathit{fplaintextg}; +; \quad) \quad ! \quad (\mathit{fciphertextg}; \quad ; \quad)$$

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

$$'(m_1 + m_2) = '(m_1) \quad '(m_2); \quad \text{or}$$
$$'(m_1 \quad m_2) = '(m_1) \quad '(m_2):$$

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

Learning With Errors (LWE) encryption/decryption

• For $n; q; t \ 2$ N with $t j q$ and a message $m \ 2 \ Z_t \quad Z_q$, the **LWE encryption** with the key $s \quad {}_{key}(Z_q^n)$ of m is defined as:

$$LWE_{q;s}(m) := (a; b) = (a; ha; si + \tilde{m} + e) \ 2 \ Z_q^{n+1};$$

where $a \quad Z_q^n$, error $e \quad {}_{error}(Z_q)$ and $\tilde{m} = \frac{q}{t} m$.

Learning With Errors (LWE) encryption/decryption

- For $n; q; t \; 2$ N with $t j q$ and a message $m \; 2 \; Z_t \quad Z_q$, the **LWE encryption** with the key $s \quad {}_{key}(Z_q^n)$ of m is defined as:

$$LWE_{q;s}(m) := (a; b) = (a; ha; si + \tilde{m} + e) \; 2 \; Z_q^{n+1};$$

where $a \quad Z_q^n$, error $e \quad {}_{error}(Z_q)$ and $\tilde{m} = \frac{q}{t} m$.

- The **decryption** of a ciphertext $(a; b)$ of m is

$$LWE \;\; {}^{1}(a; b) := d\frac{t}{q}(b \quad ha; si)c \; 2 \; Z_q:$$

Learning With Errors (LWE) encryption/decryption

- Let $Err_{LWE}((a;b);m) = \frac{t}{q}(b \quad h a;s i) \quad m = \frac{t}{q}e$. If $j\frac{t}{q}ej\ 2\ [0;1=2]$ then $LWE \quad ^{1}(a;b) = m \quad !$ Successful decryption.

Learning With Errors (LWE) encryption/decryption

- Let $Err_{LWE}((\vec{a}; b); m) = \frac{t}{q}(b - h\vec{a}; \vec{s}i) - m = \frac{t}{q}e$. If $j\frac{t}{q}ej\ 2\ [0; 1=2]$ then $LWE^{-1}(\vec{a}; b) = m\ !$ Successful decryption.

On the other hand we have,

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

Learning With Errors (LWE) encryption/decryption

• Let $Err_{LWE}((\vec{a}; b); m) = \frac{t}{q}(b \quad h\vec{a}; \vec{s}i) \quad m = \frac{t}{q}e$. If $j\frac{t}{q}ej\ 2\ [0; 1{=}2]$ then $LWE^{\ 1}(\vec{a}; b) = m\ \ !$ Successful decryption.

On the other hand we have,

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

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

RLWE encryption: work with polynomials

• For $N; Q; t\ 2\ \mathbb{N}$ with tjq and a message $m(X)\ 2\ R_Q = R=QR$ with $R = \mathbb{Z}[X]= X^N + 1$, the **RLWE encryption** with the key $z(X)$ $_{key}(R_Q)$ of $m\ 2\ R_t$ is defined as:

$$RLWE_{Q;z}(m) := (a; b) = (a; a\ \ z + \tilde{m} + e)\ 2\ R_Q^2;$$

where a R_Q, error e $_{error}(R_Q)$ and $\tilde{m} = \frac{Q}{t}m$. RLWE encryption is linearly homomorphic.

Multiplicative RLWE encryption

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

$$RLWE_z^{\emptyset}(m) = RLWE_z(m; RLWE_z(Bm); : : : ; RLWE_z(B^{k\ \ 1}m)):$$

Multiplicative RLWE encryption

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

$$RLWE'_z(m) = RLWE_z(m; RLWE_z(Bm); \dots; RLWE_z(B^{k-1}m)).$$

Finally, we define the encryption scheme

$$RGSW_z(m) = (RLWE'_z(-zm); RLWE'_z(m)).$$

Multiplicative RLWE encryption

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

$$\mathrm{RLWE}'_z(m) = \mathrm{RLWE}_z(m; \mathrm{RLWE}_z(Bm); \dots; \mathrm{RLWE}_z(B^{k-1}m)).$$

Finally, we define the encryption scheme

$$\mathrm{RGSW}_z(m) = (\mathrm{RLWE}'_z(-zm); \mathrm{RLWE}'_z(m)).$$

We have the multiplication operation

$$\boxdot : \mathrm{RLWE} \times \mathrm{RGSW} \to \mathrm{RLWE}$$

Multiplicative RLWE encryption

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

$$\mathrm{RLWE}'_z(m) = \mathrm{RLWE}_z(m; \mathrm{RLWE}_z(Bm); \ldots; \mathrm{RLWE}_z(B^{k-1}m)).$$

Finally, we define the encryption scheme

$$\mathrm{RGSW}_z(m) = (\mathrm{RLWE}'_z(-z \cdot m); \mathrm{RLWE}'_z(m)).$$

We have the multiplication operation

$$\boxdot : \mathrm{RLWE} \times \mathrm{RGSW} \to \mathrm{RLWE}$$

Lemma

For two messages $m_0, m_1 \in R_Q$ with m_1 small, we have:

$$\mathrm{RLWE}_z(m_0) \boxdot \mathrm{RGSW}_z(m_1) = \mathrm{RLWE}_z(m_0 \cdot m_1)$$

Numerical implementation

- Implement functions RLWE, RLWE^{-1}, RGSW,

- Need to work on $R_q = Z_q[X]/(X^N + 1)$, 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 di erent parameters $N \in \{2^9; 2^{10}\}$ and $q \in \{2^4; 2^5; 2^6; 2^7; 2^8\}$ and $\in [0; 10]$, to see the e ect of noise

Experiments code/decode RLWE$_{n,2}$ f $2^9; 2^{10}$g, and $= 0:5 : 5$

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

Experiments Prod RLWE), $n \in \{2^9, 2^{10}\}$ and $\sigma = 0.01 : 0.03$

Hint of Bootstrap: Trick of working in $Z_q[X] = X^N + 1$

For a polynomial $w(X) = \sum_{i=0}^{N-1} w_i X^i$ and anyu $2 Z_q$, we have

$$\text{coe}_0(X^u w(X)) = \text{coe}_u(w(X)) = w_u$$

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

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

$$\mathrm{coeff}_0(X^u w(X)) = \mathrm{coeff}_u(w(X)) = w_u$$

Recall that $\mathrm{bitrev}(s, i) = m + e$ so that with $u = \mathrm{bitrev}(s, i)$

$$\mathrm{coeff}_0(X^{(\mathrm{bitrev}(s,i))} w(X)) = w_{m+e} = w_m$$

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

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

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

$$\mathrm{coeff}_0(X^{-u} w(X)) = \mathrm{coeff}_u(w(X)) = w_u$$

Recall that $\mathrm{bin}(a,s) = m + e$ so that with $u = \mathrm{bin}(a,s)$

$$\mathrm{coeff}_0(X^{-(\mathrm{bin}(a,s))} w(X)) = w_{m+e} = w_m$$

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

GOAL: write this formula with ciphertexts ! $\Rightarrow$ bootstrap scheme

- Of course you cannot reveal the secret key s... but you can publicly reveal an encrpyption $\mathrm{Enc}_{s'}(s)$ 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

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

Bootstrap Procedure

INPUT : Cipher $LWE_s(m) = (a, b)$ with error e

- Step 1: Blind rotation

$(a, b) = LWE_s(m) \to (a_Z(X), b_Z(X)) = RLWE_{Z(X)}(X^{b + \langle a, s \rangle} w(X))$

- Step 2: Key switching

$(a_Z(X), b_Z(X)) \to$

Bootstrap Procedure

INPUT : Cipher $LWE_s(m) = (a, b)$ with error e

- Step 1: Blind rotation

$(a, b) = LWE_s(m) \to (a_Z(X), b_Z(X)) = RLWE_{Z(X)}(X^{b + \langle a, s \rangle} w(X))$

- Step 2: Key switching

$(a_Z(X), b_Z(X)) \to (a_S(X), b_S(X)) = RLWE_{S(X)}(X^{b + \langle a, s \rangle} w(X))$

- Step 3: Extraction

$(a_S(X), b_S(X)) \to$

Bootstrap Procedure

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

- **Step 1: Blind rotation**

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{b + h\vec{a}; \vec{s}i} \; w(X))$$

- **Step 2: Key switching**

$$(a_z(X); b_z(X)) \;\; ! \;\; (a_s(X); b_s(X)) = RLWE_{s(X)}(X^{b + h\vec{a}; \vec{s}i} \;\; w(X))$$

- **Step 3: Extraction**

$$(a_s(X); b_s(X)) \;\; ! \;\; (\vec{a}; b) = LWE_s(m)$$

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

Step 1: Blind rotation

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

Step 1: Blind rotation

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{\;\; b + h\vec{a}; \vec{s}i} \; w(X))$$

Figure: Blind Rotation $X^u w(X)$

- $u = \;\; b + h\vec{a}; \vec{s}i$.

Step 1: Blind rotation

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{b+\langle \vec{a}; \vec{s} \rangle} w(X))$$

Figure: Blind Rotation $X^u w(X)$

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

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

Step 1: Blind rotation

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{b+\langle \vec{a}; \vec{s} \rangle} w(X))$$

Figure: Blind Rotation $X^u w(X)$

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

$$RLWE_z(X^{b+\langle \vec{a}; \vec{s} \rangle} w) = RLWE_z(X^b w) \;\; RGSW_z(X^{a_0 s_0}) \;\; \cdots \;\; RGSW_z(X^{a_{n-1} s_{n-1}})$$

Step 1: Blind rotation

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{b+\langle \vec{a}; \vec{s} \rangle} w(X))$$

Figure: Blind Rotation $X^u w(X)$

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

$$RLWE_z(X^{b+\langle \vec{a}; \vec{s} \rangle} w) = RLWE_z(X^{b} w) \;\; RGSW_z(X^{a_0 s_0}) \;\; \cdots \;\; RGSW_z(X^{a_{n-1} s_{n-1}})$$

- The **Bootstrap Keys** are publicly available

$$BK_{i;j} = RGSW_z(X^{j s_i}); \quad 0 \leq i \leq n; \quad 0 \leq j < q$$

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i;j} = RGSW_z(X^{js_i}); \quad 0 \quad i \quad n; \quad 0 \quad j < q$$

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i,j} = RGSW_z(X^{js_i}); \quad 0 \quad i \quad n; \quad 0 \quad j < q$$

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

Step 1: Blind rotation with further decompostion

- The **Bootstrap Keys** are publicly available

$$BK_{i;j} = RGSW_z(X^{js_i}); \quad 0 \quad i \quad n; \quad 0 \quad j < q$$

- $X^{\hbar a;s i} = \overset{Q}{}_i X^{a_i s_i} = \overset{Q}{}_{i;j} X^{a_{i;j} B^j s_i}$ with $a_i = \overset{P}{}_j a_{i;j} B^j$

$$BK_{i;j;l} = RGSW_z(X^{lB^j s_i}); \quad 0 \quad i \quad n; \; 0 \quad j < \log_B(q); \; 0 \quad l < B:$$

Bootstrap Procedure

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

- **Step 1: Blind rotation**

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_z(X); b_z(X)) = RLWE_{z(X)}(X^{\;\; b + h\vec{a};\vec{s}i} \; w(X))$$

- **Step 2: Key switching**

$$(a_z(X); b_z(X)) \;\; ! \;\; (a_s(X); b_s(X)) = RLWE_{s(X)}(X^{\;\; b + h\vec{a};\vec{s}i} \;\; w(X))$$

- **Step 3: Extraction**

$$(a_s(X); b_s(X)) \;\; ! \;\; (\vec{a}; b) = LWE_s(m)$$

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 $\mathbf{m}$ *2* R_Q encrypted by a secret key $\mathbf{z}_1$ *2* R_Q to a ciphertext $RLWE_{\mathbf{z}_2}(\mathbf{m})$ encrypted by a new secret key $\mathbf{z}_2$ *2* R_Q.

Algorithm 1: Key Switching in RLWE

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

Outputs: $(\mathbf{a}^{\prime}; \mathbf{b}^{\prime}) = RLWE_{\mathbf{z}_2}(\mathbf{m})$

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

2 Write $\mathbf{a} = \quad \sum_{j=0}^{K} \quad_j B^j$ where $\quad_j = \quad \sum_{i=0}^{N \quad 1} a_{i;j} X^i$ such that $ja_{i;j}j \quad B \quad 1$

3 Return $(\mathbf{a}^{\prime}; \mathbf{b}^{\prime}) = \quad \sum_{j=1}^{K} \quad_j \quad KS_j^{\mathbf{a}}; \mathbf{b} \quad \sum_{j=1}^{K} \quad_j \quad KS_j^{\mathbf{b}}$

Bootstrap Procedure

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

- **Step 1: Blind rotation**

$$(\vec{a}; b) = LWE_s(m) \;\; ! \;\; (a_Z(X); b_Z(X)) = RLWE_{Z(X)}(X^{\;\; b+h\vec{a};\vec{s}i} \; w(X))$$

- **Step 2: Key switching**

$$(a_Z(X); b_Z(X)) \;\; ! \;\; (a_s(X); b_s(X)) = RLWE_{s(X)}(X^{\;\; b+h\vec{a};\vec{s}i} \;\; w(X))$$

- **Step 3: Extraction**

$$(a_s(X); b_s(X)) \;\; ! \;\; (\vec{a}; b) = LWE_s(m)$$

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

Step 3: Extraction

Let $RLWE_{s(X)}(m(X)) = (a(X); b(X))\ 2\ R_Q$. We have

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

Step 3: Extraction

Let $RLWE_{s(X)}(m(X)) = (a(X); b(X))\ 2\ R_Q$. We have

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

At the i^{th} coefficient we have:

$$\begin{aligned} b_i &= \sum_{k=0}^{N-1} a_i\ \ _k s_k + \tilde{m}_i + e_i \\ &= h\ _i(a); s\, i + \tilde{m}_i + e_i \end{aligned}$$

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

Step 3: Extraction

Let $RLWE_{s(X)}(m(X)) = (a(X); b(X))\ 2\ R_Q$. We have

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

At the i^{th} coefficient we have:

$$\begin{aligned} b_i &= \sum_{k=0}^{N-1} a_i\ \ {}_k s_k + \tilde{m}_i + e_i \\ &= h\ {}_i(a); s\, i + \tilde{m}_i + e_i \end{aligned}$$

where ${}_i(a)$ is the i^{th} anti-cyclic permutation of coefficients of a. Since $e(X)$ is small, so is e_i, thus $({}_i(a); b_i)$ 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!