Fully Homomorphic Encryption

Prof. Dr. Ir. Frederik Vercauteren

Katholieke Universiteit Leuven

29 October 2014
Fully Homomorphic Encryption

RLWE-based Somewhat Homomorphic Encryption

Bootstrapping: From Somewhat to Fully
Encryption / Decryption

- **Encryption**: function $\text{Enc}(\cdot, \cdot)$ from $\mathcal{M} \times \mathcal{K}_E$ to $\mathcal{C}$

$$c = \text{Enc}(m, k_E)$$

- input message $m$
- encryption key $k_E$
- ciphertext $c$
Encryption / Decryption

- **Encryption**: function $\text{Enc}(\cdot, \cdot)$ from $\mathcal{M} \times \mathcal{K}_E$ to $\mathcal{C}$
  
  $$c = \text{Enc}(m, k_E)$$

  - input message $m$
  - encryption key $k_E$
  - ciphertext $c$

- **Decryption**: function $\text{Dec}(\cdot, \cdot)$ from $\mathcal{C} \times \mathcal{K}_D$ to $\mathcal{M}$

  - If $k_D$ is the decryption key corresponding to $k_E$ we have
    
    $$\text{Dec}(\text{Enc}(m, k_E), k_D) = m$$

  - If $c$ is not a valid encryption, Dec should return ⊥
Homomorphic Encryption

- $\text{Enc}(\cdot, \cdot)$ is **homomorphic** for an operation $\square$ on message space $\mathcal{M}$ iff

  $$\text{Enc}(m_1 \square m_2, k_E) = \text{Enc}(m_1, k_E) \circ \text{Enc}(m_2, k_E)$$

  with $\circ$ operation on ciphertext space $\mathcal{C}$

- If $\square = +$, then $\text{Enc}$ is additively homomorphic
- If $\square = \times$, then $\text{Enc}$ is multiplicatively homomorphic
Homomorphic Encryption: Examples

- Caesar cipher is homomorphic for concatenation
Homomorphic Encryption: Examples

- Caesar cipher is homomorphic for concatenation
- Textbook RSA is multiplicatively homomorphic
- Public key: modulus $N = p \cdot q$, encryption exponent $e$
- Given two ciphertexts $c_1 = m_1^e \mod N$ and $c_2 = m_2^e \mod N$

$$c_1 \cdot c_2 = (m_1 \cdot m_2)^e \mod N$$
A fully homomorphic encryption allows evaluation of arbitrary functions on encrypted messages.
A fully homomorphic encryption allows evaluation of arbitrary functions on encrypted messages.

\[ m_1, \ldots, m_t \]

\[ \text{Enc}(\cdot, k_E) \]

\[ c_1, \ldots, c_t \]
A fully homomorphic encryption allows evaluation of arbitrary functions on encrypted messages.

\[ m_1, \ldots, m_t \xrightarrow{\text{some function } F} F(m_1, \ldots, m_t) \]

\[ \text{Enc}(\cdot, k_E) \]

\[ c_1, \ldots, c_t \]
A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages.

\[
\begin{align*}
\text{Enc}(\cdot, k_E) & \quad m_1, \ldots, m_t \quad \text{some function } \mathcal{F} \quad \mathcal{F}(m_1, \ldots, m_t) \\
\text{Evaluate}(\mathcal{F}, k_E, \cdot) & \quad c_1, \ldots, c_t \quad \text{Enc}(\mathcal{F}(m_1, \ldots, m_t), k_E)
\end{align*}
\]
A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages.

\[
\begin{align*}
m_1, \ldots, m_t & \xrightarrow{\text{some function } F} F(m_1, \ldots, m_t) \\
c_1, \ldots, c_t & \xrightarrow{\text{Evaluate}(F, k_E, \cdot)} \text{Enc}(F(m_1, \ldots, m_t), k_E) \\
\end{align*}
\]
A fully homomorphic encryption allows evaluation of arbitrary functions on encrypted messages.

\[ m_1, \ldots, m_t \xrightarrow{\text{some function } \mathcal{F}} \mathcal{F}(m_1, \ldots, m_t) \]

\[ \mathcal{E}(\cdot, k_E) \]

\[ c_1, \ldots, c_t \xrightarrow{\text{Evaluate}(\mathcal{F}, k_E, \cdot)} \mathcal{E}(\mathcal{F}(m_1, \ldots, m_t), k_E) \]

Size of new ciphertext should still be compact.
Uses of Homomorphic Encryption

- Private data and public functions:
  - Example: Health care data stored encrypted in cloud
  - Cloud computes known functions on this data, e.g. statistics on blood pressure measurements
Uses of Homomorphic Encryption

- **Private data and public functions:**
  - Example: Health care data stored encrypted in cloud
  - Cloud computes known functions on this data, e.g. statistics on blood pressure measurements
- **Private data and private functions:**
  - Example: Financial data and models are both encrypted
Uses of Homomorphic Encryption

- Private data and public functions:
  - Example: Health care data stored encrypted in cloud
  - Cloud computes known functions on this data, e.g. statistics on blood pressure measurements

- Private data and private functions:
  - Example: Financial data and models are both encrypted

- Note: in some scenario’s do not need power of fully homomorphic encryption
  - Computing average: additive scheme suffices
  - Standard deviation / logistic regression: additions and one multiplication
Uses of Homomorphic Encryption

- Use: delegate **processing** of data without revealing data

- **Fully homomorphic** encryption:
  - Any processing function is possible
  - Computationally very expensive

- **Somewhat homomorphic** encryption
  - Restricted set of functions can be evaluated
  - Cheaper computationally
Homomorphic Functionality

- A normal computer computes any function of data by **manipulating bits**
Homomorphic Functionality

- A normal computer computes any function of data by manipulating bits
- Any function on bits can be expressed by using only two operations:
  - **AND**: $b_1 \text{ AND } b_2 = 1$ iff $b_1 = b_2 = 1$
  - Note that **AND** is *multiplication modulo* 2
  - **XOR**: $b_1 \text{ XOR } b_2 = 0$ iff $b_1 = b_2$
  - Note that **XOR** is *addition modulo* 2
Homomorphic Functionality

- A normal computer computes any function of data by manipulating bits.
- Any function on bits can be expressed by using only two operations:
  - \text{AND} : \ b_1 \ AND \ b_2 = 1 \iff b_1 = b_2 = 1
  - Note that \text{AND} is multiplication modulo 2
  - \text{XOR} : \ b_1 \ XOR \ b_2 = 0 \iff b_1 = b_2
  - Note that \text{XOR} is addition modulo 2
- Sufficient to compute addition and multiplication homomorphically to evaluate any function!
  - Need to express the function as a Boolean circuit . . .
Definition of FHE

- Like before: KeyGen, Enc, Dec
- Two extra functions:
  - Add: $c_3 = \text{Add}(c_1, c_2)$ then
    \[ \text{Dec}(c_3) = \text{Dec}(c_1) + \text{Dec}(c_2) \mod 2 \]
  - Mult: $c_3 = \text{Mult}(c_1, c_2)$ then
    \[ \text{Dec}(c_3) = \text{Dec}(c_1) \cdot \text{Dec}(c_2) \mod 2 \]
Fully Homomorphic Encryption

- Concept proposed in 1978, but unsolved for 30 years
- In 2009, Gentry presented first Fully Homomorphic Encryption scheme
Fully Homomorphic Encryption

- Concept proposed in 1978, but unsolved for 30 years
- In 2009, Gentry presented first Fully Homomorphic Encryption scheme
- Gentry’s scheme uses two key ideas:
  - **Ideal lattices**
  - **Bootstrapping** a somewhat homomorphic scheme into a fully homomorphic scheme
Fully Homomorphic Encryption

- Concept proposed in 1978, but unsolved for 30 years
- In 2009, Gentry presented first Fully Homomorphic Encryption scheme
- Gentry’s scheme uses two key ideas:
  - Ideal lattices
  - Bootstrapping a somewhat homomorphic scheme into a fully homomorphic scheme
- Since then several other schemes and optimizations have appeared
  - Schemes based on Learning With Errors (LWE) and ring LWE
  - Schemes based on Approximate GCD assumption
  - Schemes based on NTRU (similar to ring LWE)
Limitations of Fully Homomorphic Encryption

- FHE can evaluate circuits efficiently in time proportional to size of circuit

▶ FHE does not handle Random Access Machines
▶ Conditional jumps: if . . . then . . . else
▶ Since the condition in the if statement is encrypted, one has to compute both branches!
▶ Indirect addressing: FHE does not do pointers
▶ Since address you want to fetch is encrypted, have to get everything . . .
Limitations of Fully Homomorphic Encryption

- FHE can evaluate circuits efficiently in time proportional to size of circuit
- FHE does **not** handle Random Access Machines
  - Conditional jumps: `if...then...else`
  - Since the condition in the `if` statement is encrypted, one has to compute both branches!
- Indirect addressing: FHE does not do pointers
- Since address you want to fetch is encrypted, have to get everything...
Learning with Errors over Rings

- Let $f(x) \in \mathbb{Z}[x]$ be monic irreducible polynomial of deg $n$
  - Popular choice is $f(x) = x^n + 1$ with $n = 2^k$
- Denote $R = \mathbb{Z}[x]/(f(x))$
- For an integer $q$, let $R_q = R/qR$
Learning with Errors over Rings

Let $f(x) \in \mathbb{Z}[x]$ be monic irreducible polynomial of deg $n$
  
  - Popular choice is $f(x) = x^n + 1$ with $n = 2^k$
  
  Denote $R = \mathbb{Z}[x]/(f(x))$

For an integer $q$, let $R_q = R/qR$

RLWE:
  
  - choose $s \in R_q$ at random
  - distinguish uniform random distribution on $R_q \times R_q$ from

\[
(a_i, a_i \cdot s + e_i)
\]

where $a_i \in R_q$ random and $e_i \in R_q$ has “small” coefficients

- “small”: when reduced to $(-q/2, q/2]$
Encryption based on RLWE

- Plaintext space is taken as $R_2$
- Let $\Delta = \lfloor q/2 \rfloor$
- Denote $[\cdot]_q$ reduction in $(-q/2, q/2]$
- $\chi$ error distribution on $R_q$
Encryption based on RLWE

- Plaintext space is taken as $R_2$
- Let $\Delta = \lfloor q/2 \rfloor$
- Denote $[\cdot]_q$ reduction in $(-q/2, q/2]$
- $\chi$ error distribution on $R_q$
- **Secret key**: sample $s \leftarrow \chi$
Encryption based on RLWE

- Plaintext space is taken as $R_2$
- Let $\Delta = \lfloor q/2 \rfloor$
- Denote $[\cdot]_q$ reduction in $(-q/2, q/2]$
- $\chi$ error distribution on $R_q$
- **Secret key**: sample $s \leftarrow \chi$
- **Public key**: sample $a \leftarrow R_q$, $e \leftarrow \chi$ and output

$$pk = ([(-a \cdot s + e)]_q, a).$$

- Can interpret $pk$ as degree 1 polynomial $pk(x)$ with

$$[pk(s)]_q = e$$
Encryption based on RLWE

- **Encrypt** message $m \in R_2$, let $\mathbf{p}_0 = \mathbf{pk}[0], \mathbf{p}_1 = \mathbf{pk}[1]$
- Sample $\mathbf{u}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$ and set

\[
\mathbf{ct} = \left( [\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m}]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q \right)
\]
Encryption based on RLWE

- **Encrypt** message $m \in R_2$, let $p_0 = pk[0]$, $p_1 = pk[1]$

- Sample $u, e_1, e_2 \leftarrow \chi$ and set

  $$ct = \left( [p_0 \cdot u + e_1 + \Delta \cdot m]_q, [p_1 \cdot u + e_2]_q \right)$$

- **Decrypt** ciphertext $ct$: set $c_0 = ct[0]$, $c_1 = ct[1]$ and compute

  $$\left\lfloor \frac{[c_0 + c_1 \cdot s]_q}{\Delta} \right\rfloor_2$$
Decryption Analysis

- Writing out definition

\[ c_0 + c_1 \cdot s = p_0 \cdot u + e_1 + \Delta \cdot m + p_1 \cdot u \cdot s + e_2 \cdot s \mod q \]

\[ = \Delta \cdot m + e \cdot u + e_1 + e_2 \cdot s \mod q \]
Decryption Analysis

- Writing out definition

\[ c_0 + c_1 \cdot s = p_0 \cdot u + e_1 + \Delta \cdot m + p_1 \cdot u \cdot s + e_2 \cdot s \mod q = \Delta \cdot m + e \cdot u + e_1 + e_2 \cdot s \mod q \]

- Error term \( e \cdot u + e_1 + e_2 \cdot s \) is small in \((-q/2, q/2)\)

- As long as error term \(< \Delta/2\) decryption works correctly
Decryption Analysis

- Writing out definition
  \[ c_0 + c_1 \cdot s = p_0 \cdot u + e_1 + \Delta \cdot m + p_1 \cdot u \cdot s + e_2 \cdot s \mod q \]
  \[ = \Delta \cdot m + e \cdot u + e_1 + e_2 \cdot s \mod q \]

- Error term \( e \cdot u + e_1 + e_2 \cdot s \) is small in \((-q/2, q/2]\)
- As long as error term \( < \Delta/2 \) decryption works correctly
- Valid ciphertext = polynomial \( ct(x) \) such that
  \[ [ct(s)]_q = \Delta \cdot m + v \]
  with \( |v| < \Delta/2 \)
Homomorphinc Operation: Addition

- Let $c_t_i$ for $i = 1, 2$ be two ciphertexts, with

$$[c_t_i(s)]_q = \Delta \cdot m_i + v_i$$

then

$$[c_t_1(s) + c_t_2(s)]_q = \Delta \cdot [m_1 + m_2]_2 + v_1 + v_2 + \epsilon,$$

where $\epsilon$ comes from reduction modulo 2 of $m_1 + m_2$

- Polynomial addition thus gives plaintext addition modulo 2

- Error grows additively in original errors
Homomorphic Operation: Multiplication

- Write the evaluation of $ct_i(x)$ in $s$ as an equality in $R$ as follows
  \[ ct_i(s) = \Delta \cdot m_i + v_i + q \cdot r_i. \]

- Multiply these expressions together to obtain:
  \[
  (ct_1 \cdot ct_2)(s) = \Delta^2 \cdot m_1 \cdot m_2 + \Delta \cdot (m_1 \cdot v_2 + m_2 \cdot v_1) \\
  + q \cdot (v_1 \cdot r_2 + v_2 \cdot r_1) + v_1 \cdot v_2 \\
  + q \cdot \Delta \cdot (m_1 \cdot r_2 + m_2 \cdot r_1) + q^2 \cdot r_1 \cdot r_2.
  \]

- Need to scale over $\Delta$ to recover encryption of product of plaintexts.
Homomorphic Operation: Multiplication

- Write $ct_1(x) \cdot ct_2(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2$, then product of ciphertexts is

$$[d_0, d_1, d_2] := \left\lfloor \frac{c_0}{\Delta} \right\rfloor, \left\lfloor \frac{c_1}{\Delta} \right\rfloor, \left\lfloor \frac{c_2}{\Delta} \right\rfloor$$
Homomorphic Operation: Multiplication

- Write $c_{t_1}(x) \cdot c_{t_2}(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2$, then product of ciphertexts is

$$[d_0, d_1, d_2] := \lceil \frac{c_0}{\Delta} \rceil, \lceil \frac{c_1}{\Delta} \rceil, \lceil \frac{c_2}{\Delta} \rceil$$

- If original errors $< E$, then we have

$$\left[ d_0 + d_1 \cdot s + d_2 \cdot s^2 \right]_q = \Delta \cdot [m_1 m_2]_2 + v,$$

with $||v|| < 2 \cdot C_R \cdot ||s|| \cdot E$

- $C_R$ a constant depending only on $R$

- Use secret with $||s|| = 1$
Homomorphic Operation: Multiplication

- Problem: ciphertext grows with each multiplication.

\[ ct_0 + c_1 \cdot s + c_2 \cdot s^2 \cdot q = c_0' + c_1' \cdot s + r \cdot q, \]

where \( |r| \) is small.
Problem: ciphertext grows with each multiplication.

Relinearisation: from degree 2 ciphertext to degree 1

Given $\mathbf{ct} = [c_0, c_1, c_2]$, we want $\mathbf{ct}' = [c'_0, c'_1]$ such that

$$\left[c_0 + c_1 \cdot s + c_2 \cdot s^2\right]_q = \left[c'_0 + c'_1 \cdot s + r\right]_q,$$

where $||r||$ is small.
Problem: ciphertext grows with each multiplication.

Relinearisation: from degree 2 ciphertext to degree 1

Given $\mathbf{ct} = [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2]$, we want $\mathbf{ct}' = [\mathbf{c}'_0, \mathbf{c}'_1]$ such that

$$\left[ \mathbf{c}_0 + \mathbf{c}_1 \cdot s + \mathbf{c}_2 \cdot s^2 \right]_q = \left[ \mathbf{c}'_0 + \mathbf{c}'_1 \cdot s + \mathbf{r} \right]_q,$$

where $\|\mathbf{r}\|$ is small.

Requires to give out encryptions of $T^i \cdot s^2$, so extra assumption
Homomorphic Encryption: Summary

- Secret key: sample $s \leftarrow \chi$
- Public key:
  - sample $a \leftarrow R_q$, $e \leftarrow \chi$ and output
    
    $$pk = ([(-a \cdot s + e])_q, a).$$
  
- Valid ciphertext = polynomial $ct(x)$ such that

  $$[ct(s)]_q = \Delta \cdot m + v$$

  with $v < \Delta/2$. 
Homomorphic Encryption: Summary

- Homomorphic addition: polynomial addition
- Homomorphic multiplication:
  - polynomial multiplication
  - scaling down by $\Delta$ and rounding
  - relinearisation from degree 2 to degree 1
- Requires $\text{rlk}$ containing masked versions of $T^i s^2$

\[
\text{rlk} = \left[ \left( -\left( a_i \cdot s + e_i \right) + T^i \cdot s^2 \right)_q, a_i \right) : i \in [0..\ell] .
\]
Homomorphic Capability

- Assume original errors are bounded by $B$
- Can evaluate all circuits of multiplicative depth $L$ where
  \[ C_R^L \cdot 2^{L-1} < q/B \]
- Multiplicative depth $L$: can evaluate terms of the form
  \[ \prod_{i=1}^{2^L} c_i \]
  where $c_i$ are “clean” ciphertexts
From Somewhat To Fully Homomorphic . . .

- Enc is **bootstrappable** if it can homomorphically evaluate its own Dec function
From Somewhat To Fully Homomorphic . . .

- **Enc** is **bootstrappable** if it can homomorphically evaluate its own **Dec** function.

\[ \text{Enc}(c, k_D) \xrightarrow{\text{Dec}(c, k_D)} m \]

\[ \text{Enc}(\cdot, k_E) \xrightarrow{\text{Evalute}(\text{Dec}, k_E, \text{Enc}(c), \text{Enc}(k_D))} \text{Enc}(m, k_E) \]

Prof. Dr. Ir. Frederik Vercauteren
Fully Homomorphic Scheme

- Homomorphically decrypting is called Recrypt
- Noise level after Recrypt is constant $r_{\text{Rec}}$
- If addition/multiplication of two Recrypts can be decrypted, the scheme is fully homomorphic
Fully Homomorphic Scheme

- Homomorphically decrypting is called Recrypt
- Noise level after Recrypt is constant $r_{\text{Rec}}$
- If addition/multiplication of two Recrypts can be decrypted, the scheme is fully homomorphic
Fully Homomorphic Scheme: Implementation

S. Halevi, V. Shoup: HElib

- Polynomials of degree 16384
- Can encrypt 1024 elements in $\mathbb{F}_{2^{16}}$
- Security level 76 bits
- Boostrapping: 320s and 3.4GB memory
Conclusions

- **Fully** homomorphic encryption is possible
  - BUT: at the moment not very practical
  - Have been major advancements in efficiency: SIMD, bootstrapping, . . .
- Many applications: fixed number of multiplications only
- Other constructions are possible
  - approx GCD, NTRU, LWE, . . .
- Can we get a scheme without noise?