Fully Homomorphic Encryption

Prof. Dr. Ir. Frederik Vercauteren

Katholieke Universiteit Leuven

29 October 2014

Encryption / Decryption

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

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

  - input message m
  - encryption key k_E
  - ciphertext c

Encryption / Decryption

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

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

  - input message m
  - encryption key k_E
  - ciphertext c
- **Decryption**: function $\mathrm{Dec}(\cdot,\cdot)$ from $\mathcal{C} \times \mathcal{K}_D$ to $\mathcal{M}$
- If k_D is the decryption key corresponding to k_E we have

$$\mathrm{Dec}(\mathrm{Enc}(m, k_E), k_D) = m$$

- If c is not a valid encryption, Dec should return $\perp$

Homomorphic Encryption

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

$$\mathrm{Enc}(m_1 \square m_2, k_E) = \mathrm{Enc}(m_1, k_E) \diamond \mathrm{Enc}(m_2, k_E)$$

  with $\diamond$ operation on ciphertext space $\mathcal{C}$
- If $\square = +$, then Enc is additively homomorphic
- If $\square = \times$, then 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 $c_1 = m_1^e \bmod N$ and $c_2 = m_2^e \bmod N$

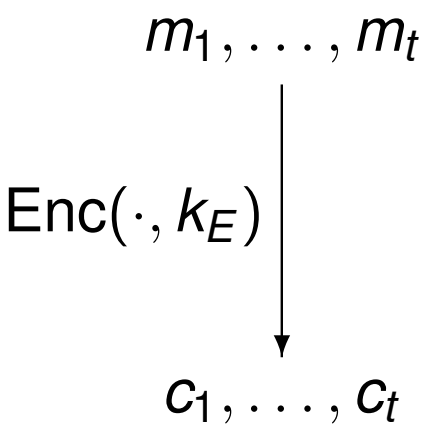
$$c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod N$$

Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages

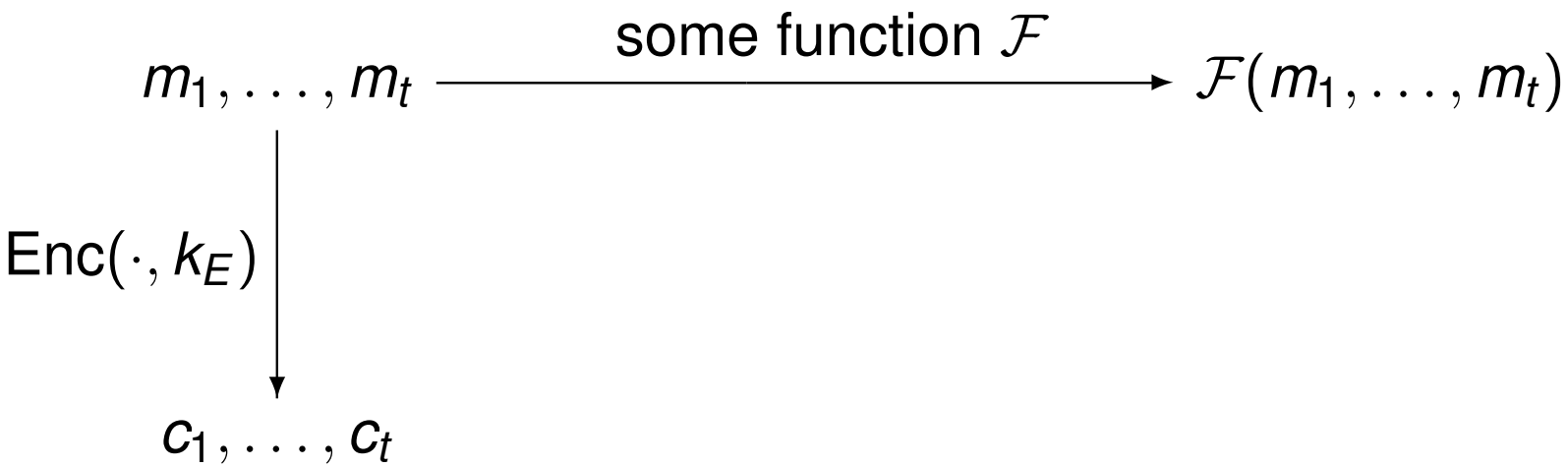
Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages



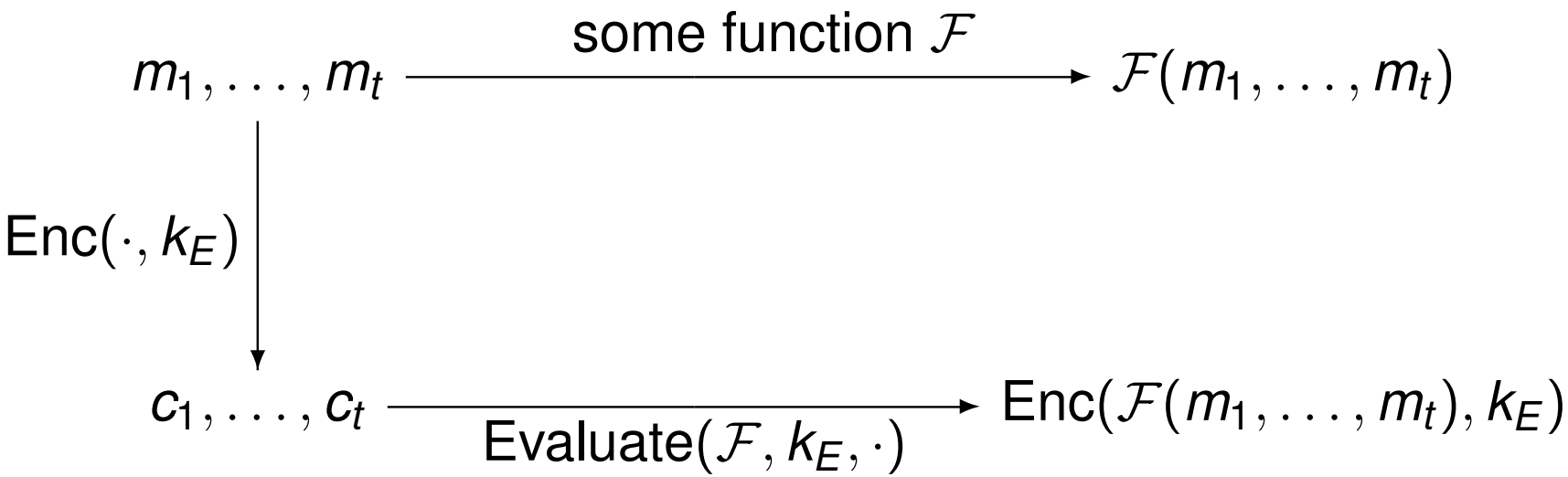
Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages



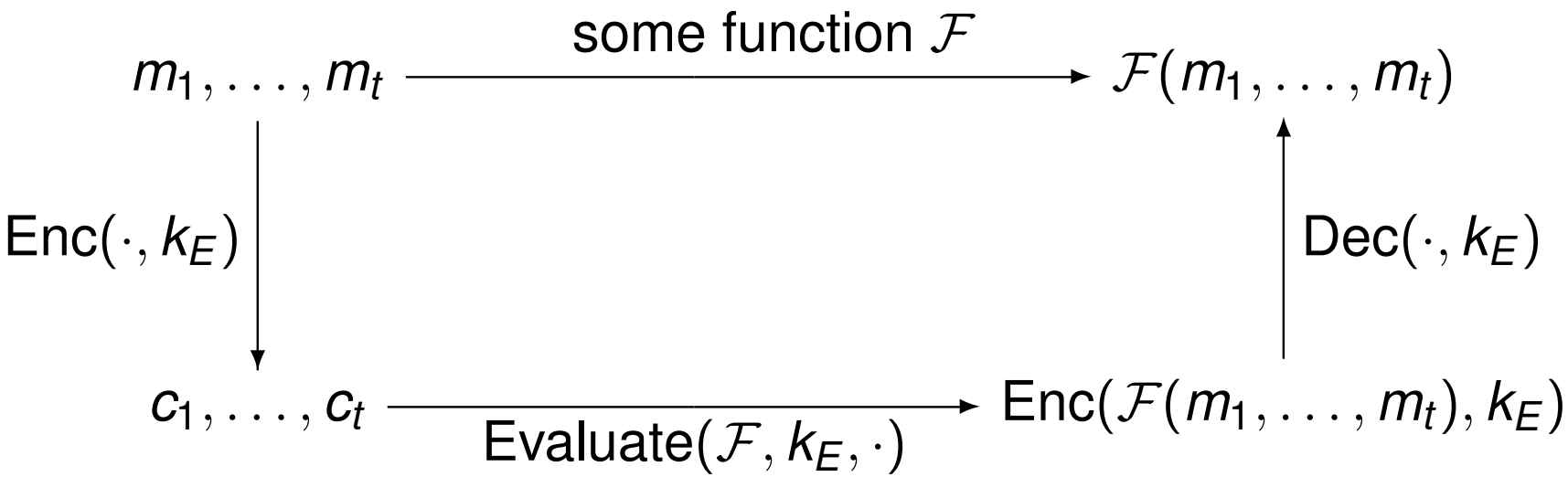
Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages



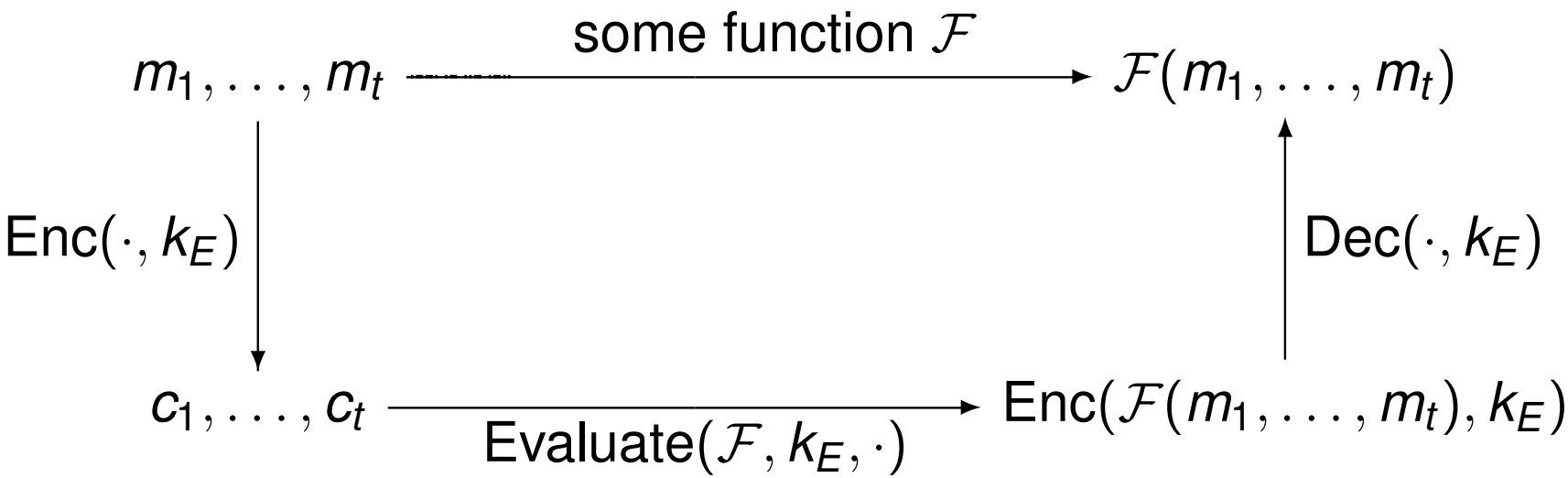
Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages



Fully Homomorphic Encryption

- A **fully homomorphic** encryption allows evaluation of arbitrary functions on encrypted messages



- 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 `AND` $b_2 = 1$ iff $b_1 = b_2 = 1$
  - Note that `AND` is **multiplication modulo** 2
  - `XOR` : b_1 `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:
  - `AND` : b_1 `AND` $b_2 = 1$ iff $b_1 = b_2 = 1$
  - Note that `AND` is **multiplication modulo** 2
  - `XOR` : b_1 `XOR` $b_2 = 0$ iff $b_1 = b_2$
  - Note that `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
    $\text{Dec}(c_3) = \text{Dec}(c_1) + \text{Dec}(c_2)$ modulo 2
  - Mult: $c_3 = \text{Mult}(c_1, c_2)$ then
    $\text{Dec}(c_3) = \text{Dec}(c_1) \cdot \text{Dec}(c_2)$ modulo 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

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 $\mathbf{s} \in R_q$ at random
  - distinguish uniform random distribution on $R_q \times R_q$ from

$$(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i)$$

    where $\mathbf{a}_i \in R_q$ random and $\mathbf{e}_i \in R_q$ 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 $(-q/2, q/2]$
- χ 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 $(-q/2, q/2]$
- χ error distribution on R_q
- **Secret key**: sample $\mathbf{s} \leftarrow \chi$

Encryption based on RLWE

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

$$\mathtt{pk} = \left(\left[(-\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \right]_q, \mathbf{a} \right).$$

- Can interpret $\mathtt{pk}$ as degree 1 polynomial $\mathtt{pk}(x)$ with

$$[\mathtt{pk}(\mathbf{s})]_q = \mathbf{e}$$

Encryption based on RLWE

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

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

Encryption based on RLWE

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

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

- **Decrypt** ciphertext $\texttt{ct}$: set $\mathbf{c}_0 = \texttt{ct}[0]$, $\mathbf{c}_1 = \texttt{ct}[1]$ and compute

$$\left[\left\lfloor \frac{[\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q}{\Delta} \right\rceil \right]_2$$

Decryption Analysis

- Writing out definition

$$\begin{aligned}\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} &= \mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{p}_1 \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \bmod q \\ &= \Delta \cdot \mathbf{m} + \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s} \bmod q\end{aligned}$$

Decryption Analysis

- Writing out definition

$$\begin{aligned}\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} &= \mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{p}_1 \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \bmod q \\ &= \Delta \cdot \mathbf{m} + \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s} \bmod q\end{aligned}$$

- Error term $\mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s}$ is small in $(-q/2, q/2]$
- As long as error term $< \Delta/2$ decryption works correctly

Decryption Analysis

- Writing out definition

$$\begin{aligned}\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} &= \mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{p}_1 \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \bmod q \\ &= \Delta \cdot \mathbf{m} + \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s} \bmod q\end{aligned}$$

- Error term $\mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s}$ is small in $(-q/2, q/2]$
- As long as error term $< \Delta/2$ decryption works correctly
- Valid ciphertext = polynomial $\mathtt{ct}(x)$ such that

$$[\mathtt{ct}(\mathbf{s})]_q = \Delta \cdot m + \mathbf{v}$$

with $|\mathbf{v}| < \Delta/2$

Homomorphic Operation: Addition

- Let $\mathtt{ct}_i$ for $i = 1, 2$ be two ciphertexts, with

$$[\mathtt{ct}_i(\mathbf{s})]_q = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i$$

then

$$[\mathtt{ct}_1(\mathbf{s}) + \mathtt{ct}_2(\mathbf{s})]_q = \Delta \cdot [\mathbf{m}_1 + \mathbf{m}_2]_2 + \mathbf{v}_1 + \mathbf{v}_2 + \epsilon,$$

where ϵ comes from reduction modulo 2 of $\mathbf{m}_1 + \mathbf{m}_2$

- Polynomial addition thus gives plaintext addition modulo 2
- Error grows additively in original errors

Homomorphic Operation: Multiplication

- Write the evaluation of $\mathtt{ct}_i(x)$ in $\mathbf{s}$ as an equality in R as follows

$$\mathtt{ct}_i(\mathbf{s}) = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i + q \cdot \mathbf{r}_i \,.$$

- Multiply these expressions together to obtain:

$$\begin{aligned}(\mathtt{ct}_1 \cdot \mathtt{ct}_2)(\mathbf{s}) = \Delta^2 \cdot \mathbf{m}_1 \cdot \mathbf{m}_2 + \Delta \cdot (\mathbf{m}_1 \cdot \mathbf{v}_2 + \mathbf{m}_2 \cdot \mathbf{v}_1) \\ + q \cdot (\mathbf{v}_1 \cdot \mathbf{r}_2 + \mathbf{v}_2 \cdot \mathbf{r}_1) + \mathbf{v}_1 \cdot \mathbf{v}_2 \\ + q \cdot \Delta \cdot (\mathbf{m}_1 \cdot \mathbf{r}_2 + \mathbf{m}_2 \cdot \mathbf{r}_1) + q^2 \cdot \mathbf{r}_1 \cdot \mathbf{r}_2 \,.\end{aligned}$$

- Need to scale over Δ to recover encryption of product of plaintexts.

Homomorphic Operation: Multiplication

- Write $\mathtt{ct}_1(x) \cdot \mathtt{ct}_2(x) = \mathbf{c}_0 + \mathbf{c}_1 \cdot x + \mathbf{c}_2 \cdot x^2$, then product of ciphertexts is

$$[\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2] := [\lfloor \mathbf{c}_0/\Delta \rceil, \lfloor \mathbf{c}_1/\Delta \rceil, \lfloor \mathbf{c}_2/\Delta \rceil]$$

Homomorphic Operation: Multiplication

- Write $\mathtt{ct}_1(x) \cdot \mathtt{ct}_2(x) = \mathbf{c}_0 + \mathbf{c}_1 \cdot x + \mathbf{c}_2 \cdot x^2$, then product of ciphertexts is

$$[\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2] := [\lfloor \mathbf{c}_0/\Delta \rceil, \lfloor \mathbf{c}_1/\Delta \rceil, \lfloor \mathbf{c}_2/\Delta \rceil]$$

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

$$\left[\mathbf{d}_0 + \mathbf{d}_1 \cdot \mathbf{s} + \mathbf{d}_2 \cdot \mathbf{s}^2\right]_q = \Delta \cdot [\mathbf{m}_1 \mathbf{m}_2]_2 + \mathbf{v},$$

  with $||\mathbf{v}|| < 2 \cdot C_R \cdot ||\mathbf{s}|| \cdot E$
- C_R a constant depeding only on R
- Use secret with $||\mathbf{s}|| = 1$

Homomorphic Operation: Multiplication

- Problem: ciphertext grows with each multiplication.

Homomorphic Operation: Multiplication

- Problem: ciphertext grows with each multiplication.
- Relinearisation: from degree 2 ciphertext to degree 1
- Given $\mathtt{ct} = [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2]$, we want $\mathtt{ct}' = [\mathbf{c}'_0, \mathbf{c}'_1]$ such that

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

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

Homomorphic Operation: Multiplication

- Problem: ciphertext grows with each multiplication.
- Relinearisation: from degree 2 ciphertext to degree 1
- Given $\mathtt{ct} = [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2]$, we want $\mathtt{ct}' = [\mathbf{c}'_0, \mathbf{c}'_1]$ such that

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

  where $||\mathbf{r}||$ is small.
- Requires to give out encryptions of $T^i \cdot \mathbf{s}^2$, so extra assumption

Homomorphic Encryption: Summary

- Secret key: sample $\mathbf{s} \leftarrow \chi$
- Public key:
  - sample $\mathbf{a} \leftarrow R_q$, $\mathbf{e} \leftarrow \chi$ and output

$$\mathtt{pk} = \left(\left[(-\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \right]_q, \mathbf{a} \right).$$

- Valid ciphertext = polynomial $\mathtt{ct}(x)$ such that

$$\left[\mathtt{ct}(\mathbf{s}) \right]_q = \Delta \cdot m + \mathbf{v}$$

with $\mathbf{v} < \Delta/2$.

Homomorphic Encryption: Summary

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

$$\texttt{rlk} = [(\left[-(\mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i) + T^i \cdot \mathbf{s}^2\right]_q, \mathbf{a}_i) : i \in [0..\ell]] .$$

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

$$\prod_{i=1}^{2^L} \mathbf{c}_i$$

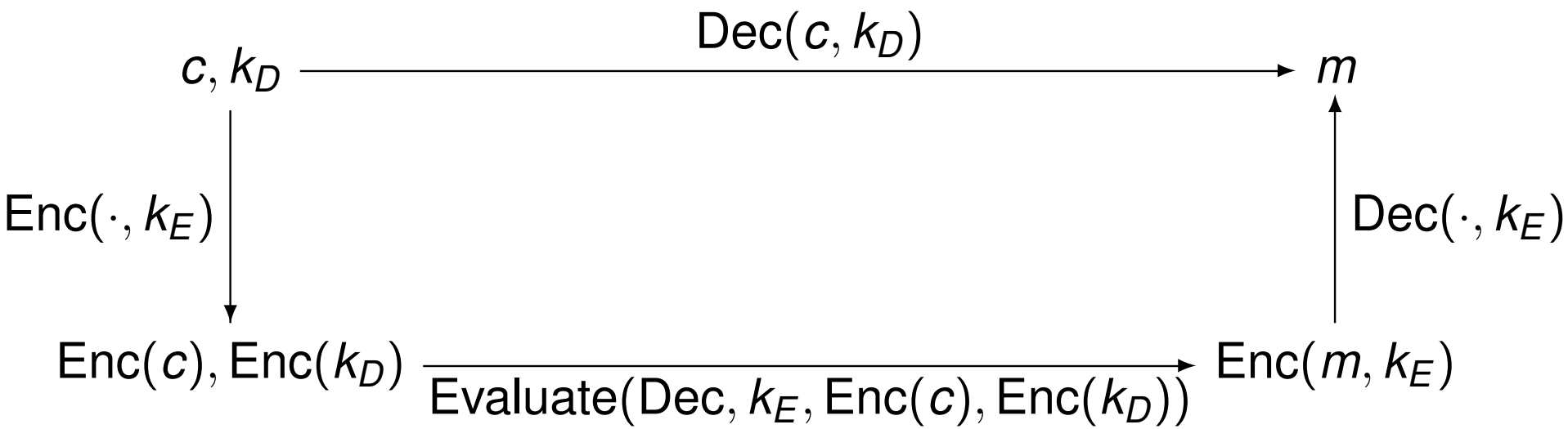
where $\mathbf{c}_i$ are "clean" ciphertexts

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

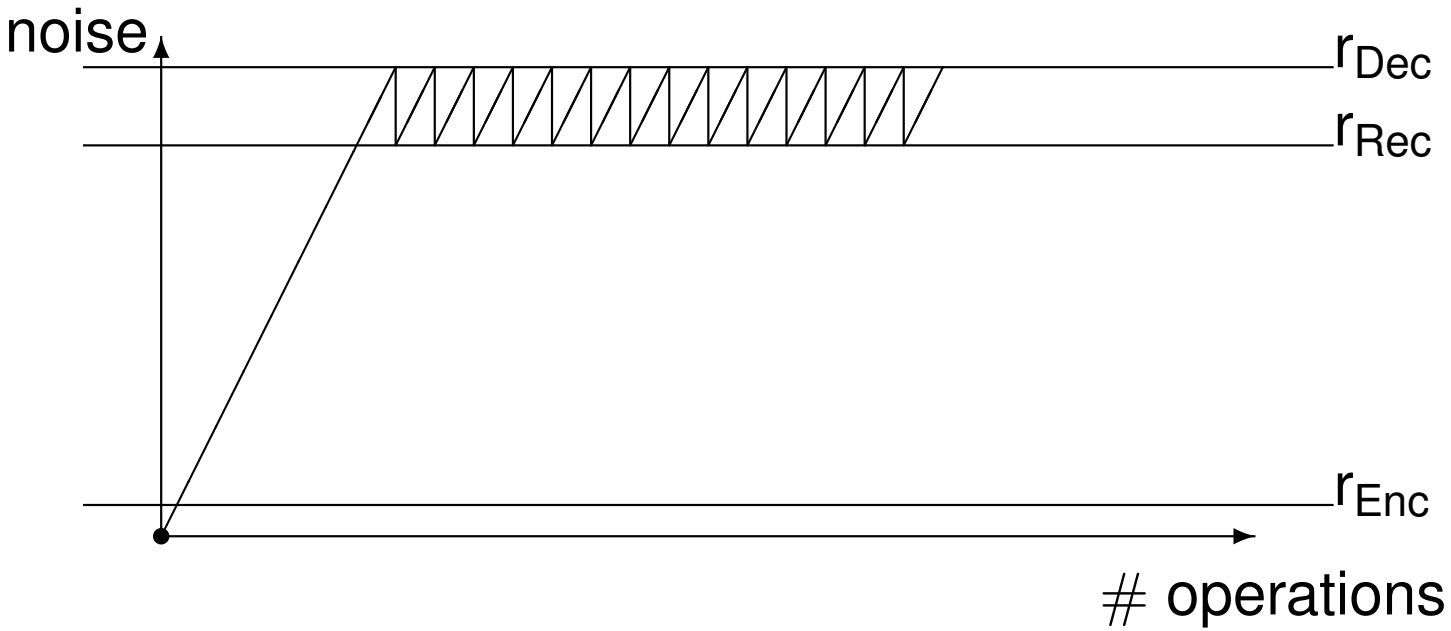


Fully Homomorphic Scheme

- Homomorphically decrypting is called Recrypt
- Noise level after Recrypt is constant r_{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_{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?