Lecture 10: Homomorphic Encryption

Cunsheng Ding

HKUST, Hong Kong

March 13, 2026

Contents

A Motivation for Homomorphic Encryption

Partially Homomorphic Encryption Schemes

Fully Homomorphic Encryption Schemes

Applications of Homomorphic Encryption

Appendix: A Fully Homomorphic Encryption Scheme

A Motivation for Homomorphic Encryption

A Motivation for Homomorphic Encryption

- The main goal of encryption is to ensure the confidentiality of data.
- In some cases it is desirable to **delegate computations to an untrusted computer** (e.g., a cloud service provider).
- In such case, only the encrypted version of the data is given to the untrusted computer to process. The computer will perform computations on this encrypted data, without knowledge of the original plaintext.
- Finally, the computer will send back the computed result, and whoever has the decryption key can decrypt the computed data and obtain the desired computational result.
- To this end, the encryption scheme must have a special structure.
- Rivest, Adleman, and Dertouzous in 1978 called such encryption schemes **homomorphic encryption schemes**.

A Protocol Illustration of Homomorphic Encryption (1)

A **general** problem of outsourcing computation

A client C wants a cloud server S to compute $f(m_1, m_2)$, but does not want the server S to know m_1 and m_2. The client C and the server S would use the following protocol and a **"special"** encryption scheme.

The protocol

- The client C chooses a key k and computes $E_k(m_1)$ and $E_k(m_2)$.
- The client C sends f, $E_k(m_1)$ and $E_k(m_2)$ to the server S, and asks S to return $E_k(f(m_1, m_2))$
- According to f, the server S performs "computational operations" on $E_k(m_1)$ and $E_k(m_2)$, and computes "$E_k(f(m_1, m_2))$", and sends it to the client.
- After receiving $E_k(f(m_1, m_2))$, the client C decrypts it and recovers $f(m_1, m_2)$.

Questions: What are the special operations on $E_k(m_1)$ and $E_k(m_2)$? How does the server S compute $E_k(f(m_1, m_2))$?

A Protocol Illustration of Homomorphic Encryption (2)

A **specific** problem of outsourcing computation

A client C wants a cloud server S to compute $f(m_1, m_2) = m_1 \oplus_p m_2$, but does not want the server S to know $m_1 \in \mathbb{Z}_p^\times$ and $m_2 \in \mathbb{Z}_p^\times$. The client C and the server S would use the following protocol and the cipher $E_k(m) = k \otimes_p m$, $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_p^\times := \{1, 2, \ldots, p-1\}$.

The protocol

- The client C chooses a key k and computes $E_k(m_1) = k \otimes_p m_1$ and $E_k(m_2) = k \otimes_p m_2$.
- The client C sends $f(x, y) = x \oplus_p y$, $E_k(m_1)$ and $E_k(m_2)$ to the server S, and asks S to return $E_k(f(m_1, m_2))$
- Since $f(x, y) = x \oplus_p y$, the server S computes $E_k(f(m_1, m_2)) := E_k(m_1) \oplus_p E_k(m_2)$, and sends it to the client.
- After receiving $E_k(f(m_1, m_2))$, the client C decrypts it and recovers $f(m_1, m_2) = m_1 \oplus_p m_2$.

Partially Homomorphic Encryption Schemes

Definition of Partially Homomorphic Encryption

Let $\mathcal{D}$ and $\mathcal{R}$ be the domain and range of the encryption function E_k of a cipher respectively. Assume that $\mathcal{D}$ and $\mathcal{R}$ are associated with two binary operations $\diamond_d$ and $\diamond_r$, respectively. The encryption scheme is said to be **homomorphic** if for any given encryption key k, the encryption function E_k has the following property:

$E_k(m_1 \diamond_d m_2)$ can be computed from $E_k(m_1) \diamond_r E_k(m_2) \ \ \forall m_1, m_2 \in \mathcal{D}$.

- It is called a **partially homomorphic encryption** (PHE) scheme, as it is homomorphic for only one pair $(\diamond_d, \diamond_r)$ of operations.
- $\mathcal{D}$ and $\mathcal{R}$ may be the same. $\diamond_d$ and $\diamond_r$ could be the same.
- A partially homomorphic encryption system may be a symmetric or asymmetric cipher!
- An example symmetric cipher is $E_k(m) = k \otimes_p m$. An example asymmetric cipher is the RSA public key cipher.

Partially Homomorphic Encryption with RSA

The RSA is **multiplicatively homomorphic**

The domain and range of the RSA encryption function $E_{k_e}(m) = m^e \bmod n$ are $\mathbb{Z}_n$, where $n = pq$ and $k_e = (e, n)$.
The operations associated to $\mathbb{Z}_n$ are $\oplus_n$ and $\otimes_n$.

Justification: For $m_1 \in \mathbb{Z}_n$ and $m_2 \in \mathbb{Z}_n$, we have

$$\begin{aligned} E_{k_e}(m_1 \otimes_n m_2) &= (m_1 \otimes_n m_2)^e \bmod n \\ &= (m_1^e \bmod n) \otimes_n (m_2^e \bmod n) \\ &= E_{k_e}(m_1) \otimes_n E_{k_e}(m_2). \end{aligned}$$

Remark: RSA is not known to be **additively homomorphic** (i.e., it is unknown how to compute $E_{k_e}(m_1 \oplus_n m_2)$ from $E_{k_e}(m_1) \oplus_n E_{k_e}(m_2)$ or $E_{k_e}(m_1) \otimes_n E_{k_e}(m_2)$).

Partially Homomorphic Encryption with ElGamal (1)

The domain and range of the encryption function

Recall that the encryption key $k_e = (p, \alpha, \beta)$, where $\beta = \alpha^u \bmod p$, and the decryption key is u. The encryption function $E_{k_e}(x, v) = (y_1, y_2)$, where

$$y_1 = \alpha^v \bmod p, \quad y_2 = x\beta^v \bmod p.$$

The domain and range of the encryption function E_{k_e} of the ElGamal cipher are $\mathbb{Z}_p^\times := \{1, 2, \ldots, p-1\}$ and $\mathbb{Z}_p^\times \times \mathbb{Z}_p^\times$.
The binary operation associated to $\mathbb{Z}_p^\times$ is $\otimes_p$ and the binary operation associated to $\mathbb{Z}_p^\times \times \mathbb{Z}_p^\times$ is

$$(x_1, y_1) \otimes_p (x_2, y_2) := (x_1 \otimes_p x_2, y_1 \otimes_p y_2).$$

Proposition: ElGamal is multiplicatively homomorphic.
A proof is given in the next slide.

Partially Homomorphic Encryption with ElGamal (2)

ElGamal is multiplicatively homomorphic.

Proof: Recall that the encryption key $k_e = (p, \alpha, \beta)$, where $\beta = \alpha^u \bmod p$, and the decryption key is u. The encryption function $E_{k_e}(x, v) = (y_1, y_2)$, where

$$y_1 = \alpha^v \bmod p, \quad y_2 = x\beta^v \bmod p.$$

By Fermat's theorem, we have

$$\begin{aligned}
&E_{k_e}(x_1, v_1) \otimes_p E_{k_e}(x_2, v_2) \\
&= (\alpha^{v_1} \bmod p, x_1\beta^{v_1} \bmod p) \otimes_p (\alpha^{v_2} \bmod p, x_2\beta^{v_2} \bmod p) \\
&= (\alpha^{v_1+v_2} \bmod p, x_1x_2\beta^{v_1+v_2} \bmod p) \\
&= (\alpha^{(p-1)s+(v_1 \oplus_{p-1} v_2)} \bmod p, (x_1 \otimes_p x_2)\beta^{(p-1)s+(v_1 \oplus_{p-1} v_2)} \bmod p) \\
&= E_{k_e}(x_1 \otimes_p x_2, v_1 \oplus_{p-1} v_2),
\end{aligned}$$

where s is some nonnegative integers.

A Symmetric Partially Homomorphic Encryption Scheme

The symmetric cipher:

- The plaintext and ciphertext spaces: $\mathcal{M} = \mathcal{C} = \{0,1\}^{128}$, which are associated with the bitwise exclusive-or operation $\oplus$.
- The key space $\mathcal{K}$ consists of all binary 128×128 invertible matrices.
- For a secret key $K \in \mathcal{K}$ and a plaintext m, the encryption is $c := E_K(m) = mK$. The decryption function is $D_K(c) = cK^{-1} = m$.

Conclusion: The cipher is partially homomorphic

Proof:

$E_K(m_1 \oplus m_2) = (m_1 \oplus m_2)K = m_1K \oplus m_2K = E_K(m_1) \oplus E_K(m_2)$.

Remark: The cipher could be secure if each key K is used for only one session of application, where K is used for encrypting a few plaintexts. For the outsourcing computation, the client does not need to share the secret key K with anyone else. Hence, this cipher used under this key policy for such an application is practical.

Fully Homomorphic Encryption Schemes

Definition of Fully Homomorphic Encryption

Let $\mathcal{D}$ and $\mathcal{R}$ be the domain and range of the encryption function E_k of a cipher respectively. Assume that $\mathcal{D}$ and $\mathcal{R}$ are associated with two pairs of binary operations $(\sqcap_d, \sqcup_d)$ and $(\sqcap_r, \sqcup_r)$, respectively. The encryption scheme is said to be **fully homomorphic** if for any given encryption key k, the encryption function E_k has the following properties:

$$E_k(m_1 \sqcap_d m_2) \text{ can be computed from } E_k(m_1) \sqcap_r E_k(m_2) \text{ and}$$
$$E_k(m_1 \sqcup_d m_2) \text{ can be computed from } E_k(m_1) \sqcup_r E_k(m_2)$$

for all $m_1, m_2 \in \mathcal{D}$.

Remark: By definition, a fully homomorphic encryption scheme is an encryption scheme that is partially homomorphic with respect to two pairs of operations.

Existence of Fully Homomorphic Encryption Scheme

Question

- Is there a fully homomorphic encryption scheme?

Remark

The question remained open for a long time!

Several Fully Homomorphic Encryption Schemes

Schemes

- Breakthrough scheme of Gentry in 2009, based on “ideal latices”.
- van Dijk, Gentry, Halevi and Vaikuntanathan’s scheme over the integers in 2010.
- RLWE schemes in 2011.
- A scheme is available in the Appendix.

Remark: You are not required to read these fully homomorphic encryption schemes!

Comment

They are not practical due to low performance!

Reference

A. Acar, H. Aksu, and A. S. Uluagac, A survey on homomorphic encryption schemes: theory and implementation, 2017, arXiv:1704.03578v2 [cs.CR].

Applications of Homomorphic Encryption

Applications of Homomorphic Encryption

- Computation outsourcing
- Data mining
- Electronic voting
- Electronic cash
- Machine leaning

- Ismail San, Nuray At, Ibrahim Yakut, Huseyin Polat, Efficient paillier cryptoprocessor for privacy-preserving data mining, Security, Volume 9, Issue 11, 25 July 2016, pp. 1535–1546.
- Cuong Ngo, Secure Voting System Using Paillier Homomorphic.

  https://pdfs.semanticscholar.org/73a6/503dc37faacc96734f551719aec3392e8dc4.pdf
- Michele Minelli. Fully Homomorphic Encryption for Machine Learning. Computer Science [cs]. PSL University, 2018. English. tel-01918263. https://hal.archives-ouvertes.fr/tel-01918263

Appendix: A Fully Homomorphic Encryption Scheme

A Fully Homomorphic Encryption Scheme (1)

The symmetric cipher:

- The plaintext and ciphertext spaces: $\mathcal{M} = \mathcal{C} = \mathrm{GF}(2^{2^n})$, which has the two binary operations $+$ and $\times$ of the finite field $\mathrm{GF}(2^{2^n})$.
- The key space $\mathcal{K} = \{0, 1, 2, \ldots, 2^n - 1\}$.
- For a secret key $k \in \mathcal{K}$ and a plaintext $m \in \mathcal{M}$, the encryption is $c := E_k(m) = m^{2^k}$. The decryption function is $D_k(c) = c^{2^{2^n - k}} = m$.

Conclusion: The cipher is fully homomorphic, as it is well known that

$$E_k(m_1 + m_2) = E_k(m_1) + E_k(m_2) \text{ and } E_k(m_1 \times m_2) = E_k(m_1) \times E_k(m_2).$$

Remark: The cipher could be secure if each key k is used for only one session of application, where k is used for encrypting a few plaintexts. For the outsourcing computation, the client does not need to share the secret key k with anyone else. In such a case, there is no need for key distribution.

A Fully Homomorphic Encryption Scheme (2)

Comments on the fully homomorphic encryption scheme

- With the fast exponentiation algorithm, the encryption and decryption require k and $2^n - k$ multiplications in the finite field $\mathrm{GF}(2^{2^n})$, respectively. If n is large, this cipher is very inefficient.
- Since the encryption function is linear with respect to both pairs of operations $(+, +)$ and $(\times, \times)$, only under the special key usage policy described in the previous slide the cipher could be secure.
- To understand the encryption scheme, one must have the basic knowledge of finite fields. There are books and online lecture notes on finite fields for downloading. For example, one may download the related lecture slides in the following link:
  - https://www.cse.ust.hk/faculty/cding/COMP2711H/slides.html
- COMP5631 students are not required to read this fully homomorphic encryption scheme. Hence, this is optional.