Homomorphic Encryption

Mihir Malani, Isha Rathi, Nikita Tipule, Samiksha Shinde

College of Engineering , Pune

Department of Computer Engineering

1 Abstract

This document is the analysis and the research about an advanced topic in the field of cryptography and network security. It primarily focuses on homomorphic encryption and its various type. Four kinds of single homomorphic encryption algorithm were summarized for the advantages of homomorphic encryption technology in the cloud environment. It analyzed the security characteristic of four kinds of encryption algorithm.

2 Keywords

partially homomorphic encryption, Somewhat homomorphic encryption, leveled fully homomorphic encryption, fully homomorphic encryption.

3 Introduction

With the widespread application of cloud computing, more and more sensitive information and private data are stored in the cloud by users. Cloud storage security is one of the important security issues in cloud computing. In order to protect the privacy of user data, cloud data should be stored in the form of ciphertext. But encryption adds computational cost. It is hoped that the confidentiality of the data will be guaranteed at the lowest possible cost. Since cloud service providers are unreliable third parties, how to keep data confidential to cloud service providers and allow cloud service providers to complete various operations on data, it is difficult for traditional encryption methods to solve the above problems. Homomorphic encryption technology supports the management of ciphertext data under privacy protection.

There are two flavors of homomorphic encryption: partially and fully. Partially Homomorphic Encryption (PHE) is where only a single operation can be performed on cipher text

4 Motivation

In today's scenario, computing addresses an extra amount of risk as essential services be normally deploy to a third party, which makes it challenging to maintain the security outlines like - data security, privacy, confidentiality, integrity, authentication etc. Most of the users favor to store their data inside cloud environment in an unoriginal form to decrease the security concerns. However, to execute any operation on the data, which is residing at server, cloud needs to first decrypt the data. This operation might create the confidentiality and privacy issues of data stored in the cloud. Homomorphic Encryption (HE) is a kind of encryption mechanism that give ability to users for computations to be prosecuted on cipher text itself, thus producing an unoriginal/encrypted result when decrypted it shows similarity on the result of operations prosecuted on the plain text. [18] Homomorphic Encryption solves the problems of confidentiality and privacy of the stored data inside the cloud.

5 Homomorphic encryption

Due to privacy leakage of sensitive data, the conventional encryption systems are not completely secure from an intermediary service like cloud servers. The homomorphic encryption is a special kind of encryption mechanism that can resolve the security and privacy issues. Unlike the public key encryption, which has three security procedures, i.e., key generation, encryption and decryption; there are four procedures in HE schemes, including the evaluation algorithm as shown in Fig. The HE allows the third-party service providers to perform certain type of operations on the user's encrypted data without decrypting the encrypted data, while maintaining the privacy of the users' encrypted data. In homomorphic encryption, if the user wants to query some information on the cloud server, he first encrypts the data and stores the encrypted data in the cloud. Then, after sometime, the user sends query information to the cloud server. The cloud server runs a prediction algorithm on encrypted data using HE without knowing the contents of the encrypted data. Then, the cloud returns the encrypted prediction back to the user and the user decrypts the received encrypted data using the user's secret key, while preserving the privacy of his data as show in Fig. In HE, mathematical operation on the plaintext during encryption is equivalent to another operation performed on the cipher text. Let us consider a simple homomorphic operation on the plaintext with the corresponding cipher text operation.

5.1 Partial Homomorphic Encryption

A defined operation can be performed infinite times on the ciphertext. These encryption schemes are relatively easy to design.

Partially homomorphic encryption (PHE) allows only select mathematical functions to be performed on encrypted values. This means that only one operation, either addition or multiplication, can be performed an unlimited number of times on the ciphertext. PHE with multiplicative operations is the foundation for RSA encryption, which is commonly used in establishing secure connections through SSL/TLS.
In this section, we provide two partially homomorphic encryption schemes over finite fields and give security analysis. These encryption schemes are symmetric.

5.1.1 A multiplicative homomorphic encryption scheme

Let $F^*_q = F_q \setminus \{0\}$ and $Z^*_{q-1} = \{k \in Z_{q-1} \mid \gcd(k, q-1) = 1\}$, where q is a power of a prime.

For a positive integer n, let η be a primitive element of F_{q^n}, then

$$\beta = \eta^{(q -1)/(q-1)}$$

is a primitive element of F_q . For integers a and b such that a | b, we use a/b to denote division of a by b.
For a ring R, if $a \in R$ is invertible, then we use a^{-1} to denote the inverse of a.

- **Key-Generation**

Choose a positive integer d such that
$d \mid (q^n - 1)/(q - 1)$ and $\gcd(d, q - 1) = 1$,
and choose $l \in Z^*_{q-1}$. The tuple (d, l) is the secret key.

- **Encryption**

Let $\alpha = \eta^{(q^n -1)/d}$, which is a primitive d-th root of unity over F_q .
To encrypt a plaintext $m \in F^*_q$, one randomly chooses $r \in \{0, 1, . . . , d - 1\}$ and computes the ciphertext as

$$c = \gamma^{\log \beta\, m} \alpha^{r},$$

where $\gamma = \eta^{l(q^n -1)/d(q-1)}$
,
the discrete logarithm $\log_\beta m = a$ if $\beta^a = m$.

- **Decryption**

For $c \in F^*_{q^n}$, one computes

$$m' = c \text{ ^}(d \cdot l^{-1},)$$

where l^{-1} is the inverse of l in Z^*_{q-1}.

5.1.2 An additive homomorphic encryption scheme

The Paillier Cryptosystem was invented by Pascal Paillier in 1999. It is a Partial Homomorphic Encryption (PHE) scheme and Additively Homomorphic. The nature of the algorithm allows for homomorphic addition operations to produce the current answer once decrypted.

Paillier cryptosystem key generation algorithm

1: Select two large prime numbers p and q where
$$gcd\ (pq, (p - 1)\ (q - 1)) = 1$$

2: Calculate $n = pq$

3: Calculate $\lambda = lcm\ p - 1, q - 1)$

4: Select g as a random integer where $g \in Zn2^*$

5: Define $L(x) = x{-}1n$

6: Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse

7: $u = (L(g^\lambda \bmod n^2))^{-1} \bmod n$

8: Public Key = (n,g)
9: Private Key = (λ,u)

Paillier cryptosystem encryption algorithm

<u>Encrypt a message M where M $\in\mathbb{Z}_n$</u>

1: Select r as a random integer where $r \in Zn2^*$
2: Calculate $c = g^m \times r^n \bmod n^2$

Paillier cryptosystem decryption algorithm

<u>Decrypt a message c where c $\in Z_{n2}^*$</u>

1: Calculate $m = L(c^\lambda \bmod n^2) \times u \bmod n$

5.2 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is an emerging cryptographic technique that allows developers to perform computations on encrypted data. This represents a paradigm shift in how data processing and data privacy relate to each other.

Previously, if an application had to perform some computation on data that was encrypted, this application would necessarily need to decrypt the data first, perform the desired computations on the clear data, and then re-encrypt the data. FHE, on the other hand, simply removes the need for this decryption-encryption steps by the application, all at once.

As a result, FHE can have an enormous impact to our society. It can change the way computations are performed by preserving end-to-end privacy. For example, users would be able to offload expensive computations to cloud providers in a way that cloud providers will not have access to the users' data at all.

5.2.1 FHE Builds on Public-Key Encryption

FHE provides all the functions supported by asymmetric PKE (public key encryption). As it is used today, PKE is based on finding discrete logarithms or factoring large integers and has five properties:

Key generation: (sk, pk) $\leftarrow$ K(λ)
where key generation function K with argument random seed number λ produces a key pair consisting of a secret key sk and a public key pk.

Encryption: c$\leftarrow$E (pk, m)
where encryption function E with arguments pk and plaintext message m produces encrypted message ciphertext c.

Decryption: m $\leftarrow$ D (sk, c)
where decryption function D with arguments sk and c produce m.

Correctness: m = D (sk, E (pk, m))
for all key pairs, messages, and encryption randomness.

Semantic security:

$\forall m \in \{0, 1\}$ — for all single-bit messages m, member of the set 0 and 1, E(pk, 0) and E(pk, 1) must be computationally indistinguishable and must be probabilistic (e.g., there should be many encrypted messages c per plaintext message m).

For use in HE, two more properties must be added:

Evaluate:
Along with the K, E, and D functions, V for evaluate is added.

Correctness:
D (sk, V (pk, f, c_1, ... c_n)) = f (m_1, ..., m_n)
where decryption function D with arguments sk and evaluation function V with arguments pk; function f where f $\in \mathcal{F}$ (a set of or family of efficiently computable functions that have the desired homomorphic properties); and ciphertexts c_1, ..., c_n are equal to function f applied to arguments m_1, ..., m_n. For multiplication this would be
D (sk, HE-MULTIPLY (pk, MULTIPLY, E (pk, m_1), E (pk, m_2))) = MULTIPLY (m_1, m_2).

To encrypt a bit *b*

HE encrypts a plaintext bit into a polynomial.

1. Pick a large, odd number p to be the secret key.
2. For each encryption, pick a random, large multiple of p, say q_ip.
3. Then for each encryption, sum bit b and q_ip with a Noise expression defined by the doubling of a random small number r_i to $2r_i$.

This produces ciphertext $c = q_ip + 2r_i + b$ where $q_ip + 2r_i$ is the public key.

To decrypt ciphertext *c*

b = c modulo p modulo 2 removes the Noise.
= $q_ip + 2r_i + b$ modulo p modulo 2

HE addition — XORing two encrypted bits

$c_1 = q_1p + 2r_1 + b_1$
$c_2 = q_2p + 2r_2 + b_2$
$c_1 + c_2 = p\ (q_1 + q_2) + 2(r_1 + r_2) + (b_1 + b_2)$

HE multiplication — ANDing two encrypted bits

$c_1 = q_1p + 2r_1 + b_1$
$c_2 = q_2p + 2r_2 + b_2$
$c_1c_2 = p\ (q_1q_2 + q_1b_2 + q_2b_1) + r_1(2pq_2 + b_2) + r_2(2pq_1 + b_1) + r_1r_2 + b_1b_2$

Noise growth

- Addition: $2(r_1 + r_2)$ Noise = 2 x <initial noise>
- Multiplication: $r_1(2pq_2 + b_2) + r_2(2pq_1 + b_1) + r_1r_2$ Noise = <initial noise>^{2}

Conclusion

- Privacy of data is essential than ever before in this internet world. Outsourcing applications are in high demand due to the cloud computing revolutions.

- Although data are encrypted, the difficulty with encrypted data is that it must be decrypted before being used.

- As a result of decrypting, it becomes vulnerable to all the things that were trying to secure it. Homomorphic encryption, which processes data while maintaining privacy and security, maybe the greatest solution for this problem in the future world.

References

[1] https://www.techtarget.com/searchsecurity/definition/homomorphicencryption#:~:text=Homomorphic%20encryption%20is%20the%20conversion,data%20without%20compromising%20the%20encryption

[2] https://www.sciencedirect.com/topics/computer-science/fully-homomorphic-encryption

[3] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms[C] Foundations of Secure Computation. New York:
Academic Press, 1978: 169-179.

[4] Hill L S. Cryptography in an algebraic alphabet [J]. The American Mathematical Monthly, 1929, 36(6): 306-312.

[5] Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public key cryptosystems [J]. Communications of ACM, l978, 21(6): 120-126.

[6] Goldwasser S, Micali S. Probabilistic encryption [J]. Journal of Computer and System Sciences, 1984, 28(2): 270-299.

[7] Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms [J]. IEEE Transactions of Information Theory, 1985, 31(4): 469-472.

[8] Benaloh J, Tuinstra D. Receipt-free secret- ballot elections [C] Proc. of the 26th Annual ACM Symposium on the Theory of Computing, New York, ACM, 1994: 544-553.

[9] Paillier P. Public-key cryptosystems based on composite degree residuosity classes [J]. Proc Eurocrypt, 1999, 547 (1): 223-238.

[10] Boneh D, Goh E J, Kobbi N. Evaluating 2-DNF formulas o

[11] https://www.researchgate.net/publication/335665846_A_systematic_review_on_the_status_and_progress_of_homomorphic_encryption_technologies

[12] https://www.sciencedirect.com/topics/computer-science/fully-homomorphic-encryption

[13] https://link.springer.com/book/10.1007/978-3-030-87629-6