Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 3, No. 1, 2023

Homomorphic Encryption and its Application to Blockchain

Sen Hu[1], Zhengquan Zhang[1], Kevin (Xiaokang) Mo[2,3]

[1] School of Mathematical Sciences at USTC and Wu Lab of Mathematics of Chinese Academy of Sciences, Hefei, China
[2] Beijing Data Institute, Beijing, China
[3] National Innovation Research Institute (Xiamen), Inc, Xiamen, China

Abstract: The concept, method, algorithm and application of the advanced field of cryptography, homomorphic encryption, as well as its application to the field of blockchain are discussed in this paper. There are various forms of homomorphic encryption and they can be classified into three categories: partially homomorphic encryption, hierarchical homomorphic encryption and fully homomorphic encryption. We explain the concept, method and algorithm for each category with down to earth examples. The milestone example of RSA algorithm is explained which is still widely used in the internet and in blockchain as well. Applications of homomorphic encryption depends on efficiency of implementation which has been in good progress in recent years.

Keywords: Cryptography; Public key cryptography; Privacy computing; Fully homo- morphic encryption; Blockchain application.

1. Introduction

The problem of homomorphic encryption originates from issues of data confidentiality in privacy computing, such as cloud computing. After the clients provide the data, it is required that the original data shall not be accessed by suppliers of the cloud computing service which only access the encrypted data.

Homomorphic encryption refers to an unconventional encryption mode with special ap- plications, and it can be used to realize encryption operation to maintain some algebraic operations, that is to say, the homomorphic encryption of one of algebraic operations of the object is equal to the same algebraic operation of the homomorphic encryption of the object. Such kind of encryption mode enable people to operate the ciphertext without the need to know the original text, thus to conduct operations and manipulations such as search and comparison with the premise of confidentiality, without the need of decryption before such kind of operations and manipulations.

The effect of homomorphic encryption is to improve the confidentiality of privacy com- puting. That is, during the process of data encryption, the calculation can also be entrusted to others, so that people's requirement of data security can be better satisfied. Compared with general forms of encryption, homomorphic encryption can avoid other people's access of the original data. For example, in the field of cloud computing, the provider of the cloud can compute on the cloud while ensuring the security and confidentiality of data.

2. Various forms of homomorphic encryption

Homomorphic encryption's form is similar to the concept of homomorphism in traditional mathematics.

Definition 4.1.1.: We call E an homomorphic encryption, if $E^{-1}(E(m1) * E(m2)) = m1 * m2$, for any $m1, m2 \in M$.

In the formula, M is the set of all possible information, E is the encrypted operator, $*$ is a kind of algebraic operation.

Homomorphic encryption can be roughly divided into the following categories:

Partially homomorphic encryption: It refers to the homomorphic encryption whose homomorphism is only applicable to a certain kind of single algebraic operation, including multiplication, addition etc., such as RSA algorithm, etc. The advantage of partially ho- momorphic encryption lies in the low degree of complexity, which means the amount of computation can be greatly reduced in most cases, and the disadvantage of it lies in that it is only applicable to a single kind of operation.

Hierarchical homomorphic encryption: It refers to the homomorphic encryption whose homomorphism is only applicable to the algebraic operation with limited times. Its advantage lies in that it can meet the requirement of more complicated operations, and its disadvantage is that the homomorphism can no longer be guaranteed in case of multiple times of addition or multiplication operations.

Fully homomorphic encryption: Such form of encryption refers to the homomorphic encryption whose homomorphism will be remained no matter any times of algebraic opera- tions, including addition and multiplication are conducted. Its advantage lies in that there is no limitation of the complexity of the operation. However, such kind of homomorphic encryption's calculation is very complicated, which will lead to a great limitation in the application of actual projects.

Classical homomorphic encryption protocols include RSA protocol, ElGamal protocol, Paillar protocol, GM protocol and BGN protocol. In general, homomorphic encryption is composed of four parts, which are secret key generation, homomorphic encryption, homo- morphic decryption and homomorphic assignment.

Secret key generation. This process refers to the generation of secret keys, including the public key pk and the private key sk, with the parameter λ.

Homomorphic encryption. This process refers to the process of using the open public key pk to conduct the encryption for the provided original text $m \in \{0, 1\}$ to

obtain the ciphertext c.

Homomorphic decryption. This process refers to the process of using the private key secretly kept by the individual to conduct the decryption for the ciphertext to obtain the new original text.

Homomorphic assignment. This process refers to the process of calculation based on the public key pk and t circuits C, and the ciphertext c = (c1 , ...,ct) to obtain c∗ = evaluate(pk,C,c), and it is required that the equation dec(sk,c∗) = C(m1 , ...,mn) be valid.

Remark: During the whole process of secret key generation, encryption, decryption and assignment, homomorphic assignment is the most crucial step. The homomorphic assignment can be conducted for any function through assignment, which is conducted for the ciphertext. The most important is that the decryption function can be obtained in this way.

The secret key generated in the homomorphic encryption protocol can be used for not only the scheme applying the same secret key for encryption and decryption, but also the scheme applying different secret keys for encryption and decryption. Such two kinds of schemes are named as symmetric homomorphic encryption and asymmetric homomorphic encryption, and the latter is generally more popular scheme.

3. Examples of partially homomorphic encryption

The homomorphism of partial homomorphic encryption is only applicable to single algebraic operations, such as addition, multiplication, etc. Its calculation is relatively simple, but the range of application is also narrow.

Example 4.2.1 (RSA scheme): The RSA encryption scheme refers to the protocol that is applicable to the homomorphism of multiplication, and its brief structure and the form of presentation are as follows:

Key generation:

1. Two prime numbers that are large enough and not close to each other, p and q , are arbitrarily chosen and let n = pq, $\phi(n) = (p-1)(q-1)$.
2. The integer e is chosen at random, and e and $\phi(n)$ are relative prime of each other and $1 < e < \phi(n)$. d is found such that $de = 1 \bmod \phi(n)$.
3. The public key is (n,e), and the private key is (n,d).

Encryption: For the random number r ∈ Zp, calculate $c = E(r) = re \pmod n$.

Decryption: $m = D(c) = cd \pmod n$.

Here is another partial homomorphic encryption scheme.

Example 4.2.2 (Paillier scheme): Paillier scheme refers to the scheme that carries out homomorphic encryption for the part of the addition that conforms to homomorphism, and its brief structure and the form of presentation are as follows:

Key generation:

1. Two prime numbers that are large enough and not close to each other, p and q , are arbitrarily chosen. pq and $(p-1)(q-1)$ are relative prime of each other. Let n = pq and λ is equal to the least common multiple of p − 1 and q − 1.
2. A random integer g ∈ Zn2 is chosen, and there is an $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$.
3. The public key is (n,g), and the private key is (λ,μ).

Encryption: c = E(m,r) = gmrn(mod n), r2 ∈ Zn is calculated for the random number r ∈ Zp with m the encrypted text.

Decryption:

$m = D(c,r) = \frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n.$

In the above, $L(x) = \frac{x-1}{n}$.

Partially homomorphic encryption is still applied in some relatively simple systems of encryption.

4. An implementation of hierarchical homomorphic encryp- tion

Compared with partially homomorphic encryption, hierarchical homomorphic encryption could apply to more complicated operations. However, the homomorphism can't be ensured if the time of operation of addition and multiplication exceeds a limit.

A simple example of hierarchical homomorphic encryption is given here.

Example 4.3.1: b is a binary number, and the encryption scheme meet c = E(b) = b + 2x + kp.

In this formula, p is an odd number and the secret key of the encryption b, and x is a random number.

Since x is random, the parity of c is also random. However, the parity of b can be known after mod p for c, that is, decryption is conducted for c.

The homomorphism of the encryption is not hard to be verified. Assuming that:

c1 = E(b1) = b1 + 2x1 + k1p,

c2 = E(b2) = b2 + 2x2 + k2p.

Then the homomorphism of the addition is:

E(b1) + E(b2) = c1 + c2 = (b1 + b2) + 2(x1 + x2) + (k1 + k2)p = E(b1 + b2).

The homomorphism of the multiplication:

E(b1)E(b2) = c1 c2 = b1 b2 + 2(b1x2 + b2x1+ 2x1x2) + kp = E(b1 b2).

x will gradually increase during the accumulation of the time of calculations until it affects the parity of the ciphertext c (mod p). Therefore, this scheme is hierarchical homomorphic encryption rather than fully homomorphic encryption.

5. Fully homomorphic encryption

The requirement of fully homomorphic encryption should be met for the practical applica- tion of homomorphic encryption. In 1978, a protocol of homomorphic encryption [1] was put forward, which was attributed to Rivest et al. In comparison with the ordinary defini- tion of homomorphic encryption, the homomorphism of fully homomorphic encryption for the two algebraic operations of addition and multiplication should be met, and the degree of complexity of the calculation is very high.

The first fully homomorphic encryption algorithm was put forward by Gentry in 2009 [2]. After continuous research and improvement, the efficiency of the latest protocol is close to the degree accepted by people. In 2017, the academic circle and the industrial circle jointly founded the organization of fully homomorphic encryption standardization, indicating that homomorphic encryption fully entered the stage of application or even standardization [3, 4].

On May 22, 2022, the Special Interest Group on Automata and Computability Theory of ACM announced that the Go¨del Prize of 2022 was awarded to Craig Gentry, Zvika

Brakerski and Vinod Vaikuntanathan [5] to commend their revolutionary contributions to the homomorphic encryption system.

5.1. Realization of fully homomorphic encryption

The algorithm scheme of this protocol shall be attributed to the talented cryptologist, Zvika Brakerski, whose research was based on the learning with errors (LWE). The confidentiality of the algorithm is ensured by the difficulty of the worst case in any problems of short vectors, that is, the resolution of such kind of problem shall not be used as the guarantee of finishing in the time of multiple-term formulae.

The famous cryptologist Regev creatively proposed the difficulty of learning with errors.

Definition 4.5.1: Let s $\in$ Zq . Linear combination of s-coefficient random polynomials is not distinguishable with elements in Zq of uniform distribution. We use the secret key s to encrypt the text m $\in$ {0, 1}. Randomly choose a vector a $\in$ Zq, adding a noise e, we have encrypted cipertext: c = (a,b =< a,s > +2e + m) $\in$ Zq × Zq .

The key point of decryption is the elimination of two interference: < a,s > and 2e, which can be respectively eliminated by the calculation of < a,s > and the verification of parity.

Proof: The homomorphism of addition is easy to be verified.

The decryption algorithm is taken into account to verify the homomorphism of multi- plication. The ciphertext (a,b) and the linear function f(a,b) : Zq — Zq are given by

$$f(a,b)(x) = b - < a,x > (\mod q) = b - \Sigma_{i=1}^{n} a[i]x[i] \in Zq .$$

The addition of the ciphertext corresponds to the addition of the linear function: f(a+a\ ,b+b\ )(x) = f(a,b)(x) + f(a\ ,b\ )(x).

This is another linear function corresponding to the ciphertext (a + a\ ,b + b\ ). The multiplication of the two ciphertexts is:

$$f(a,b)(x)f(a\ ,b\ )(x) = (b - \Sigma_{i=1}^{n} a[i]x[i])(b\ - \Sigma_{i=1}^{n} a\ [i]x[i])$$

= h0+ Σhix[i] + Σxi,jx[i]x[j]. It can be found that the result is a quadratic polynomial about the variable x. Therefore,

a serious problem is encountered, that is, the coefficient of the quadratic polynomial has to be calculated.

To narrow the scale of the ciphertext, the technology of re-linearization is adopted. In case of the new secret key t,h0 + Σhis[i] + Σhi,js[i]s[j] is approximately equal to h0 + Σhi(bi− < ai,t >) + Σhi,j(bi,j− < ai,j,t >).

This is the linear function about the secret key t, with

bi,j =< ai,j,t > +2ei,j + s[i]s[j] ≈< ai,j,t > +s[i]s[j].

Therefore, the re-linearization technology reduces the ciphertext to n + 1, which elimi- nates the dependence on the assumption of ideal cases, and a part of homomorphic encryptions is acquired through learning with errors, which becomes the basis of realizing fully homomorphic encryption.

5.2. Application of privacy computing in the field of blockchains

The combination of technologies of privacy computing and blockchains produces the ad- vantages of the two, thus realizing data security in the whole process of computing and applications.

Data privacy receives great attention during the process of blockchain application. One of the schemes to protect privacy is to produce links after the desensitization of data. However, the desensitization of data will lead to the incompleteness of data, which can affect the integrity and processing of data. Privacy computing plays a role in this process. It makes data to be processed and verified, and keep invisible with the premise of ensuring the integrity of data.

The application of the privacy computing technology in the field of blockchains includes two extremely crucial areas, namely privacy transaction protocol and privacy computing protocol. These two forms of application mainly solve the problems in the aspect of privacy.

The privacy transaction protocol refers to the encryption of the transaction data on the blockchain, and the verification of the relationship between transaction data can be ensured at the same time, that is to say, it can conduct audit accounts in the situation of keeping accounts private. At present, multiple blockchain projects adopt the privacy transaction protocol. The privacy computing protocol realizes the protection of privacy during the whole process from the generation, storage to analysis, calculation, application, and deletion at last, guaranteeing the safety of users' data to the greatest extent. It is different from the preservation of data security through the encryption and transcoding conducted by the original blockchain.

References

[1] Adleman L. Rivest R L, Shamir A. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21, 1978.

[2] Gentry C. Fully homomorphic encryption using ideal lattices. Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 169– 178, 2009.

[3] Vercauteren F. Fan J. Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive, 2012.

[4] Kim M et al. Cheon H J, Kim A. Homomorphic Encryption for Arithmetic of Approx-imate Numbers. ADVANCES IN CRYPTOLOGY - ASIACRYPT, 2017 PT I, pages 409–437, 2017.

[5] Vaikuntanathan V. Brakerski Z, Gentry C. Fully Homomorphic Encryption without Bootstrap- ping. IACR Cryptology ePrint Archive, 2011.