Advances in Computer Science Research, volume 80
International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018)

A Dynamic Symmetric Fully Homomorphic Encryption Mechanism for Privacy Protection of Cooperative Precision Positioning Cloud Service

Dibin Shan[1,2,*], Xuehui Du[1,2], Wenjuan Wang[1,2] and Nan Wang[1]
[1]Zhengzhou Institute of Information Science and Technology, Zhengzhou, China
[2]National Digital Switching System Engineering and Technological Research Center, China
[*]Corresponding author

***Abstract*—With the increasingly serious threat of privacy protect, the collaborative precision positioning cloud service urgently needs to be carried out homomorphic encryption protection for sensitive data. But the existing asymmetrical approaches are inefficient and are not practical for applications, while the symmetric approaches has low security. In this paper, based on the MORE and Enhanced MORE symmetric homomorphic encryption algorithms, we proposes a fully homomorphic encryption mechanism called DSFHE, with dynamic key generation, symmetric encryption and random encryption. Analysis and experimental results show that the proposed mechanism reduces the cipher-text storage overhead and execution time, which also can prevent the strong attacks.**

Keywords—cooperative precise positioning; fully homomorphic encryption; symmetric encryption; cloud computing; cipher-text computation

I. INTRODUCTION

Collaborative precision positioning service cloud have a large amount of data and users, which is accessed frequently. Users' privacy data stored in cloud platform are easy to reveal, which need to be encrypted for privacy protection. But it is difficult to calculate for the data after encryption. The data on the cloud platform are facing security risks, such as external attacks to steal and leakage of internal operations staff. Traditional encryption technology can protect the confidentiality of the data processing. But data query, statistics, analysis and calculation are operated in the cloud data center, data cannot be treated under encryption. Data decryption reprocessing efficiency is quite low, and Sensitive data and privacy data are easy to leak to the attackers.

Existing cipher-text computation can perform computation on encrypted data, as encryption of computation on plaintext in theory. But asymmetric homomorphic encryption technology is faced with low calculation efficiency and serious problem with enforcing cloud platform to adjust to algorithms requirement, while the symmetric homomorphic encryption is not enough secure. Homomorphic encryption technology is difficult to get practical application in cloud computing. Therefore, it is time to put forward a kind of secure and efficient new homomorphic encryption algorithm, which can break through the bottleneck of homomorphic encryption on the efficiency, safety and practical, achieving its application in the cloud data resource security protection.

The rest of this paper is organized as follows, Section 2 describes the Related Work of homomorphic encryption algorithm. Section 3 construct a dynamic symmetric fully homomorphic encryption mechanism. Security analysis and performances of DSFHE mechanism are given in Section 4. Conclusions are drawn in Section 5.

II. RELATED WORK

In 1978, the problem of homomorphic encryption [1] (privacy) was introduced by Rivest. According to its development stage [2], it can be divided into Partial Homomorphic Encryption, class Homomorphic Encryption, and Fully Homomorphic Encryption. Partial homomorphic encryption (PHE) only supports a single type of cipher domain homomorphism (add or multiply homomorphism). For example, the RSA algorithm only supports multiplicative homomorphism, and the GM algorithm only supports additive homomorphism. The class homomorphic encryption (SHE) can support the addition and multiplication of the finite number of cipher-text fields. Fully homomorphic encryption (FHE) can achieve arbitrary cipher-text addition and multiplication. In 2009, Gentry firstly proposed fully homomorphic encryption based on the Ideal case [3]. However, the fully homomorphic encryption algorithms are confronted with problems such as complex operation process, low operation efficiency and large cost of key generation.

The homomorphic encryption algorithms with symmetric key are also proposed. The MORE [4] algorithm supports the fully homomorphic encryption calculation, but whose security is not high. It is vulnerable to the chosen plaintext attack. The multi-user homomorphic encryption scheme [5] is designed, which also faces the chosen plaintext attack threat [6]. The enhanced MORE algorithm [7] supports key dynamic generation randomly, which needs both sides to negotiate synchronous secret keys beforehand. Adoption of different symmetric key may affect the results of cipher-text computation. Homomorphic encryption algorithm based on threshold scheme [9] was proposed, whose shared keys r remains unchanged, protected by threshold function.

In conclusion, homomorphic encryptions with asymmetric key are inefficient and not practical for applications. While the confidentiality and randomness of K in the symmetric homomorphic encryption mechanism need to be improved.

III. HYBRID FULLY HOMOMORPHIC ENCRYPTION MECHANISM

A. Design Concept

We proposes a dynamic symmetric fully homomorphic encryption mechanism for cloud computing data security protection. Considering the advantage of symmetric scheme, the scheme takes homomorphic encryption with symmetrical keys on plaintext encryption in order to improve the calculation efficiency, which also improves the randomness of symmetric key and changes the simple linear relations by comprehensively utilizing of the displacement, the random number, dynamic key, which prevent chosen plain text attacks.

A Fully homomorphic encryption consists of four algorithms [2]:

(1)$Gen: U \to key$, is a randomized algorithm that takes a security parameter *U* as input, and outputs a secret *key*.

(2) $Enc: (key, P) \to C$, Enc is an Encryption algorithm that takes key and a plaintext *P* as input, and outputs a cipher text *C*.

(3) $Dec: (key, C) \to P$, Dec is a Decryption algorithm that takes key and a cipher text *C*

P as input, and outputs a plaintext.

(4) $Cal: (P, F) \to (C, F), f \in F, (p_1, p_2, \cdots p_n) \in P$, F is a set of P on operation, for $f \in F, (p_1, p_2, \cdots p_n) \in P$ .The operation on the *P* can be converted to the operation on the *C* by Cal, whose results are equivalent.

Definition 1, Homomorphic [2]: The encryption algorithm ε and operation on plaintext *P* satisfy Eq. 1, if $\forall p_1, p_2, \cdots p_n \in P$.

$$Dec: \left(key, Cal: \left((c_1, c_2, \cdots c_n), f\right)\right) = (p_1, p_2, \cdots p_n) . \quad (1)$$

B. Key Generation and Protection

In order to overcome the security problem of key in the MORE algorithm, the key sequence is generated and randomly selected by the stream cipher algorithm, and the public key algorithm is used to encrypt the protection of *K*, which is a $n \times n$ matrix.

The key stream sequence *S* is generated by stream cipher algorithm. A new sequence *S'* consists of $n^2/4$ elements (n is an even number) which are randomly selected from the *S*, as a child of a key matrix *SK*. A non-zero invertible matrix *K* is comprised of matrix *SK* and the identity matrix *I*. The inverse matrix [7] of the *K* is the K^{-1}.The *S'*, *SK*, *K* and $K^{-1}$is shown as Eq. 2, 3, 4 and 5.

Public key cryptography algorithm (RSA) are used to encrypt the key sequence S', which is deleted after encryption of plaintext and random sequence.

$$S' = \left\{s_{11}, s_{12}, \cdots s_{1j}, s_{21}, s_{22}, \cdots s_{2j}, \cdots s_{j1}, s_{j2}, \cdots s_{jj}\right\}, \left(j = \frac{n}{2}\right), \quad (2)$$

$$SK = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1j} \\ s_{21} & s_{22} & \cdots & s_{2j} \\ \cdots & \cdots & \cdots & \cdots \\ s_{j1} & s_{j2} & \cdots & s_{jj} \end{bmatrix}, (j = \frac{n}{2}), \quad (3)$$

$$K = \begin{bmatrix} SK & SK + I \\ SK - I & SK \end{bmatrix}, \quad (4)$$

$$K^{-1} = \begin{bmatrix} SK & -(SK + I) \\ -(SK - I) & SK \end{bmatrix}. \quad (5)$$

C. Encryption Operation

The permutation sequence $Q = \left\{q_1, q_2, \cdots q_L\right\}$ is generated by the stream cipher algorithm. After the permutation, the permuted plaintexts of *P* is divided into *H* blocks, where *L* is plaintext length, $H = 2L/n$. Each blocks M_i is a diagonal matrix which contains $n^2/4$ elements, whose $n/2$ elements belong to *P*. The random sequence $R = \left\{r_1, r_2, \cdots r_L\right\}$ is generated by the stream cipher algorithm. Each blocks R_i is a diagonal matrix which contains $n^2/4$ elements, whose $n/2$ elements belong to *R*. Matrix MR_i is constructed with M_i and R_i, which is n×n matrix. M_i, R_i and MR_i are shown as Eq. 6, 7, and 8.

$$M_i = \begin{bmatrix} m_{i1} & 0 & \cdots & 0 \\ 0 & m_{i2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & m_{ij} \end{bmatrix}, \left(1 \le i \le \left\lceil 2L/n \right\rceil, j = \frac{n}{2}\right). \quad (6)$$

$$R_i = \begin{bmatrix} r_{i1} & 0 & \cdots & 0 \\ 0 & r_{i2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & r_{ij} \end{bmatrix}, (1 \le i \le \left\lceil 2L/n \right\rceil, j = \frac{n}{2}). \quad (7)$$

$$MR_i = \begin{bmatrix} M_i & 0 \\ 0 & R_i \end{bmatrix}. \quad (8)$$

For each plaintext matrix, the corresponding cipher-text matrix $C_i = K^{-1} * MR_i * K$.

With symmetric encryption algorithm, *Q* and *R* are encrypted by the randomly generated keys *T*, which is encrypted with asymmetric encryption. After encryption, *Q*, *R*, and *T* can be eliminated.

D. Decryption

The replacement sequence **Q**, random number sequence *R* and key matrix *K* are decrypted by using the private key of

legitimate users. The plaintext MR_i is decrypted from the received cipher text Ci. $MR_i = K * C_i * K^{-1}$. M_i is reproduced from MR_i by permutation sequence Q and random sequence R.

E. Homomorphic Calculation

Addition and Multiplication calculation are shown as Eq. 9 and Eq. 10.

$$\begin{aligned} E(M1) + E(M2) &= K^{-1}\begin{bmatrix} M1 & 0 \\ 0 & R1 \end{bmatrix} K + K^{-1}\begin{bmatrix} M2 & 0 \\ 0 & R2 \end{bmatrix} K \\ &= K^{-1}\begin{bmatrix} M1+M2 & 0 \\ 0 & R1+R2 \end{bmatrix} K \\ &= E(M1+M2) \end{aligned} \tag{9}$$

$$\begin{aligned} E(M1) * E(M2) &= K^{-1}\begin{bmatrix} M1 & 0 \\ 0 & R1 \end{bmatrix} K * K^{-1}\begin{bmatrix} M2 & 0 \\ 0 & R2 \end{bmatrix} K \\ &= K^{-1}\begin{bmatrix} M1*M2 & 0 \\ 0 & R1*R2 \end{bmatrix} K \\ &= E(M1*M2). \end{aligned} \tag{10}$$

Obviously, the DSFHE scheme is FHE because it satisfies the both homomorphic properties.

F. Application Example

For individual users of collaborative precision positioning cloud service system, the client generate dynamic key, permutation sequence, and random sequence, who take advantage of DSFHE algorithm to encrypt plaintext data and the key. The cipher-text data are sent to the data center. The users submit the computation instructions to the data center. After the data center computes the cipher-text, the calculation results are fed back to the users. Only legitimate users can decrypt and restore the calculated result in plain text by using their private key.

IV. Security and Performance Analysis

A. Privacy and Key Security

The plaintext encryption key K is randomly selected and dynamically generated from the key stream, which ensures the randomness and dynamic of the key. The key is protected by the public key algorithm, which guarantees the confidentiality of the key. Hybrid fully homomorphic encryption mechanism can be immune to choose plaintext attack. On the one hand, the permutation of plaintext sequence and participation calculation of random grouping R_i , have changed the simple linear relationship between plaintext and cipher-text. On the other hand, the dynamic random key has increased the complexity of the relationship between plaintext and cipher-text.

B. Execution Time

DSFHE, Enhanced MORE and MORE are done in ThinkPad X230 laptop with specifications of Processor Intel Core i5 CPU, 2.5GHZ, quad-core processor, 4GB of memory. The execution time is studied for different plaintexts size, and for each plaintext size the mean execution time is measured for 200 iterations. Because each M_i encryption increased with random matrix R_i and K encryption compared with Enhanced MORE algorithm, the execution time of DSFHE is between Enhanced MORE and MORE, as shown in figure1. Compared with MORE algorithm, the increase of random number group is smaller, and the computation time cost is reduced.

C. Cipher-text Storage Overhead

In DSFHE mechanism, every plaintext data block M_i ($n/2$ valid elements) is encrypted to cipher-texts group C_i (n^2 elements). In addition to storage cipher-texts of the permutation sequence Q and the random number sequence R, the whole storage overhead of cipher-texts needs 2**m***n* bytes per *m* bytes of plaintext. The storage overhead of cipher-texts is m*n*n per m bytes plaintext in MORE algorithm. Enhanced MORE algorithm needs m * n bytes of the cipher-texts to storage for m bytes of plaintexts, as shown in figure 2. But the store overhead for random number Matrix R makes sense for improving data security of symmetric encryption.

V. Summary

In this paper, we proposed a dynamic symmetric fully homomorphic encryption mechanism. Based on the comprehensive MORE and Enhance MORE algorithm, the scheme encrypt plaintext with symmetric key, with encrypting symmetric key, permutation sequence and random number sequence by asymmetric encryption schemes. This scheme reduces the cipher storage overhead, which improved the security key and ability against the attacks. However, the algorithm proposed in this paper have not solve the problem of realizing homomorphic computation and security sharing of multi-user cipher text with different keys, which needs further study.

Acknowledgements

This work is supported by National Key R&D Plan of China (Grant No.2016YFB0501901).

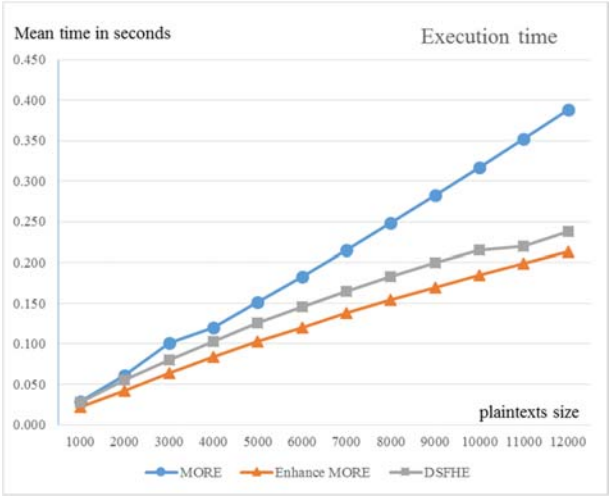

FIGURE I. EXECUTION TIME

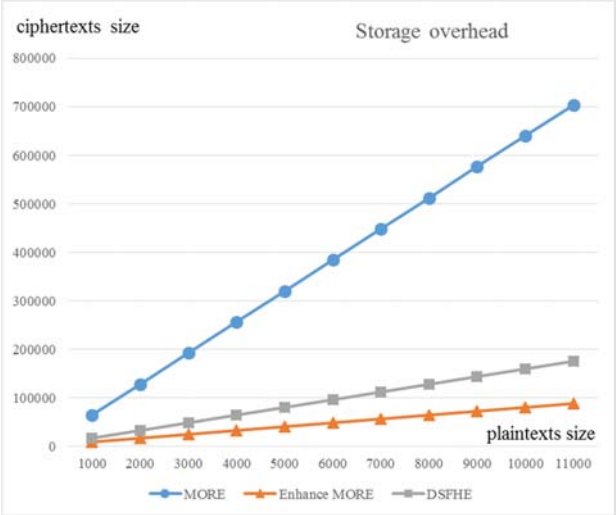

FIGURE II. STORAGE OVERHEAD

REFERENCES

[1] Rivest R, Shamir A, Adleman L, A method for obtaining digital signatures and public-key cryptosystems, Commun ACM 1978, 21(2):120-126.

[2] LI Zong-Yu, GUI Xiao-Lin, GU Ying-Jie, A Survey on Homomorphic Encryption Algorithm and Its Application in the Privacy preserving for Cloud Computing, Journal of Software, http://kns.cnki.net/kcms/detail/11.2560.TP.20171017.1337.001.html.

[3] Gentry C, A fully homomorphic encryption scheme, Phd thesis, Stanford University, 2009.

[4] Kipnis A., Hibshoosh E, Efficient methods for practical fully homomorphic symmetric-key encryption, randomization and verification, IACR Cryptology ePrint Archive 2012:1-20.

[5] Xiao L., Bastani O., Yen I.-L, An efficient homomorphic encryption protocol for multi-user systems, Citeseer, IACR Cryptology ePrint Archive 2012:193-212.

[6] Vizár D, Vaudenay S, Cryptanalysis of chosen symmetric homomorphic schemes, Studia Scientiarum Mathematicarum Hungarica 2015, 52(2):288-306.

[7] Khalil Hariss, Hassan Noura, Abed Ellatif Samhat, Fully Enhanced Homomorphic Encryption algorithm of MORE approach for real world applications. Journal of Information Security and Applications 34 (2017):233-242.

[8] Zhou Jun, Cao Zhenfu, Dong Xiaolei, etal, Security and privacy in cloud-assisted wireless wearable communications: challenges, solutions and future directions [J].IEEE Wireless Communications, 2015, 22(2):136-144.