JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 39, 525-548 (2023)
DOI: 10.6688/JISE.202305_39(3).0005

Outsourced K-means Clustering for High-Dimensional Data Analysis Based on Homomorphic Encryption*

RAY-I CHANG[1], YEN-TING CHANG[1] AND CHIA-HUI WANG[2,+]
[1]*Department of Engineering Science and Ocean Engineering*
National Taiwan University
Taipei, 106 Taiwan
E-mail: {rayichang; r09525057}@ntu.edu.tw
[2]*Department of Computer Science and Information Engineering*
Ming Chuan University
Taoyuan City, 333 Taiwan
E-mail: wangch@mail.mcu.edu.tw

In the machine learning (ML) era, people are paying more and more attention to the economic value of data in improving the efficiency of analysis, simulation, calculation, forecasting, and decision-making. It results the rise of data markets. As ML requires high-complexity calculations, individuals and companies tend to use cloud computing with data markets. However, this platform is known to have data security issues in privacy protection. The most modern method for privacy protection in cloud computing is fully homomorphic encryption (FHE). However, the high calculation cost makes conventional FHE impractical for real-world applications. Although many researchers use CKKS FHE to resolve this problem, our experiments show that the calculation cost of some operators in CKKS FHE are still very high. In this paper, we propose new security protocols to design a new data packing method and to reduce the usage of time-consuming calculations. Then, an outsourced K-means clustering method based on these new security protocols is proposed for demonstration and evaluation. Experiments show that our method is faster than SEOKC. It has shown good performance in high-dimensional data analysis with our new data packing method.

Keywords: privacy protection, K-means clustering, cloud computing, high-dimensional data analysis, fully homomorphic encryption

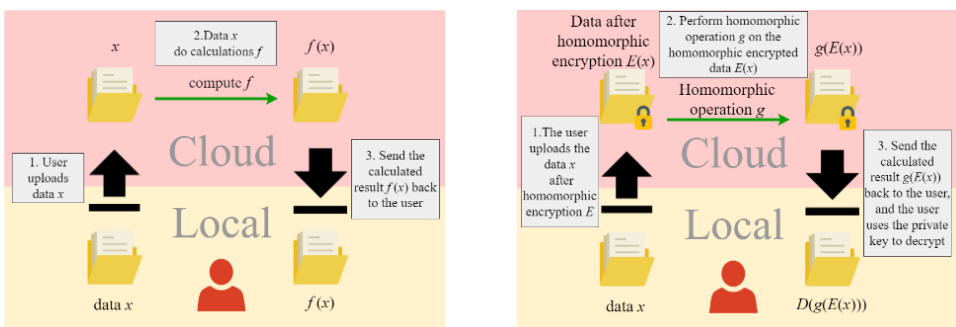
1. INTRODUCTION

In the machine learning (ML) era, people pay more and more attention to the economic value of data in improving the efficiency of analysis, simulation, forecasting, and decision-making. It results the rise of data markets, such as Statista [1], Snowflake [2], Datarade [3]. ML often requires processing with high computational complexity. ML as a Service (MLaaS) has become popular because of its cost-benefit advantages. Many cloud computing companies, *e.g.*, Alibaba [4], AWS [5], Microsoft [6], and Google [7], provide users with ML platforms to use the high-performance computing resources. However, there is a challenge of MLaaS to generate ML models on the cloud under the protection of privacy and confidentiality of user data. The security problem of private data has become an important research topic to train ML models. In data applications, privacy is defined as

Received January 8, 2022; revised February 28, 2022; accepted April 21, 2022.
Communicated by Po-Wen Chi.
+ Corresponding author.
* This research was partially funded by Ministry of Science and Technology, Taiwan, grant number 110-2410-H-002-094-MY2 and 110-2221-E-130-001.

people's right to control the use of their personal information, exclude others from viewing, using, or intruding, including the collection, processing, storage, and use of personal information [8-12]. Data privacy and confidentiality are now considered indispensable. Companies, organizations, and other institutions must securely handle users' personal information. While providing user services, they must also pay attention to confidentiality and privacy.

To protect data privacy in MLaaS, a modern method is to use Homomorphic Encryption (HE). HE is a form of encryption which allows specific types of calculations to be carried out on ciphertexts and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintexts [13]. Conventional MLaaS uploads the data x to the cloud for calculation f and then send the calculated result $f(x)$ back (as shown in Fig. 1 (a)). There may have a data privacy problem if the uploaded data is not properly deleted after the calculation. HE solves data privacy problem by uploading the homomorphically encrypted data $E(x)$ to the cloud. Perform the homomorphic operation g (related to f) on this homomorphically encrypted data $E(x)$ in the cloud, then return the result $g(E(x))$ to the user. The user uses the secret key (related to E) to decrypt the result of the homomorphic operation as $D(g(E(x)))$. This decrypted result is exactly the expected target $f(x)$ (as shown in Fig. 1 (b)). As the data uploaded to the cloud for computing are encrypted, no one can read the content of the data even if the data is not deleted afterward.



(a) (b)

Fig. 1. (a) Conventional MLaaS operation method; (b) HE MLaaS operation method.

Nowadays, there are three categories of HE algorithms; (1) Partially HE (PHE), which only allows an unlimited number of limited operations to be performed on the ciphertext; (2) Somewhat HE (SWHE) supports a limited number of calculations; (3) Fully HE (FHE) has no restrictions on the number of times and calculation methods, but the efficiency is poor [14-17]. Due to the theoretical breakthrough in these years, FHE has made great progress in efficiency [18]. The current mainstream FHE algorithms include BFV (Brakerski/ Fan-Vercauteren) [19, 20]. CKKS (Cheon-Kim-Kim-Song) [21], and TFHE (Fast Fully Homomorphic Encryption over the Torus) [22-24]. [25] evaluated the suitable application scenarios of these three HE algorithms and shown that CKKS has the best performance for the homomorphic operation of floating-point data with approximated precision. However, our experiments show that the calculation cost of some operators in CKKS FHE (such as ciphertext multiplication, ciphertext rotation, encryption, and decryption) are still very

high. In this paper, we propose new security protocols to design a new data packing method and to reduce the usage of time-consuming calculation. Then, an outsourced K-means clustering method based on these new security protocols is proposed for demonstration and evaluation.

We adopt two non-colluding servers same as SEOKC (Secure and Efficient Outsourced K-Means Clustering) [26] to design our security protocols. The data packing method of SEOKC is based on attributes. It uses some calculations that are more time-consuming in FHE. Our paper takes the shortcomings of SEOKC as a reference to design a new outsourced K-means clustering method through CKKS FHE. Our method uses a new data packing method based on record, and can reduce the usage of time-consuming calculation in CKKS FHE. Experiments show that our method is faster than SEOKC. It has shown good performance in high-dimensional data analysis with our new data packing method.

2. LITERATURE REVIEW

In 2014, [27] used FHE to propose privacy of outsourced K-means clustering. In order to compare encrypted distances in the process of clustering, the client is required to provide trapdoor information with huge online costs on the client-side. In 2017, [28] proposed K-means clustering using HE and updatable distance matrix (UDM). Secure third party stores encrypted data and associated UDMs. Then, the data owner (DO) can request the clustering of the data. In order to make the clustering run correctly, UDM needs to be updated in each iteration of the K-means algorithm. It relieves [27] from requiring the client to provide trapdoor information with participation cost. Both of the above two papers disclose private information, such as the distance between the data record and the cluster center, to the cloud server. They are proved to be unsafe [29].

In 2014, [30] used Asymmetric Scalar Product Preservation Encryption (ASPE) [31] with secure cloud computing to resist sample attacks of outsourcing data mining. In 2016, [32] proposed a method with linear transformation and random perturbation of the kernel matrix in K-means clustering to protect the privacy and perform outsourcing. The above methods also disclose some private information to the cloud server. In addition, neither of these two tasks can achieve the semantic security of encrypted databases, that is, they cannot resist chosen plaintext attack (CPA).

In 2015, [33] used the Paillier cryptosystem to propose privacy-preserving and outsourced multi-user K-means clustering. They further considered the overhead of multiple DOs, the participation of DO is not required if outsourcing the encrypted database. In addition to protecting database security, their method also uses two non-colluding cloud servers, which together form a joint cloud environment. However, this method introduces high calculation cost through the use of complex interactive protocols for encrypted data, which affects its application in large encrypted databases. In 2017, [34] proposed a privacy-preserving K-means clustering under a multi-owner setting in the distributed cloud environment. The ciphertext under different secret keys is converted to the ciphertext under the unified secret key. Their experimental results show that their solution requires less calculation cost than [33] but produces higher communication costs.

Although these two works [33, 34] achieve better privacy requirements such as the

semantic security of databases and the hiding of data access patterns, they cannot be applied to large databases due to the high cost of computing and communication. In 2017, [35] proposed privacy-preserving data clustering in cloud computing based on an FHE PPC framework, which uses MapReduce to complete distributed computing to perform data clustering on a large number of virtual machines (VM). They use a fixed-width clustering (FWC) algorithm to implement basic operations in the cloud. However, the ciphertext comparison operation proposed in this work is not safe, because the semantically safe ciphertext cannot be compared directly.

In 2017, [36] proposed CKKS FHE whose security depends on the difficulty of the Ring Learning with Errors (RLWE) problem. As shown in Fig. 2, data m is a vector on which you want to perform certain calculations. First, encode this vector into a data polynomial $p = m(X)$, and then use public key encryption to encrypt it into c. Through the calculations in f, the result of the ciphertext operation is expressed as $c' = f(c)$, and decryption with the key will produce $p' = f(p)$. Therefore, the decoding results in $m' = f(m)$.

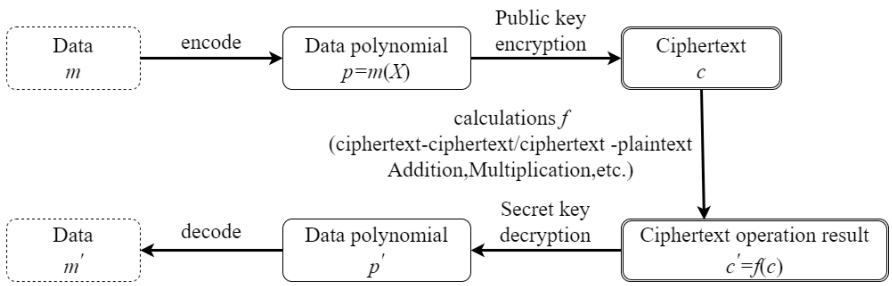


Fig. 2. CKKS flow chart [37].

The data appears in the form of a vector, not a polynomial, so the data vector $z \in \mathbb{C}^{\frac{N}{2}}$ must be encoded into a data polynomial $m(X) \in \mathcal{R}$. N is the degree of polynomial modulus, a power of 2, and $N = \frac{M}{2}$. $\Phi_M(X) = X^N + 1$ represents the Mth cyclotomic polynomial. The plaintext space is a polynomial ring $\mathcal{R} = \frac{\mathbb{Z}[X]}{\Phi_M(X)} = \frac{\mathbb{Z}[X]}{(X^N+1)}$. The ciphertext space is the remaining ring $\mathcal{R}_q = \frac{\mathcal{R}}{q^{\mathcal{R}}} = \frac{\mathbb{Z}_q[X]}{(X^N+1)}$. $\mathbb{H}$ is a subspace of $\mathbb{C}^N$ that is isomorphic to $\mathbb{C}^{\frac{N}{2}}$. Embedding standard deviation σ: $\mathcal{R} \to \sigma(\mathcal{R}) \subseteq \mathbb{H}$. Natural projection $\pi : \mathbb{H} \to \mathbb{C}^{\frac{N}{2}}$, which projects the vector from the subspace of $\mathbb{C}^N$ to $\mathbb{C}^{\frac{N}{2}}$.

Our proposed method uses the CKKS FHE. Based on the general HE of RLWE, the plaintext space polynomial quotient ring is $\frac{Z_q[X]}{X^N+1}$, and the values involved in general use can only be represented by a natural number smaller than the prime q. The plaintext space of CKKS is the complex vector space $\mathbb{C}^{\frac{N}{2}}$. Through the mapping of these two spaces, the complex vector can be expressed in polynomial terms as the input to the encryption; conversely, the result of decryption can be reduced to the complex vector. The calculations provided by CKKS FHE are as follows [36]:

KeyGeneration(N): Let $s(X) \in \mathcal{R}_q$ be the key polynomial, and the public key polynomial $p(X) = (-a(X) \cdot s(X) + e(X), a(X))$ where $a(X) \in \mathcal{R}_q$ is a uniformly randomly selected polynomial, and $e(X) \in \mathcal{R}_q$ is a small random polynomial.

Encoding(z): In order to encode the data vector $z \in \mathbb{C}^{\frac{N}{2}}$ into a data polynomial $m(X) \in \mathcal{R}$, first apply $\pi^{-1}(z) \in \mathbb{H}$ to convert the data vector z from $\mathbb{C}^{\frac{N}{2}}$ expands to $\mathbb{H}$. Multiply it by the scaling factor Δ and then randomly round to $\lfloor \Delta \cdot \pi^{-1}(z) \rceil$ to scale the vector appropriately. Since rounding may lose precision, scaling is performed to achieve a predefined precision. In order to obtain the data polynomial, use the canonical embedding σ^{-1} and get $m(X) = \sigma^{-1}(\lfloor \Delta \cdot \pi^{-1}(z) \rceil) \in \mathcal{R}$.

Decoding($m(X)$): In order to decode the data polynomial $m(X) \in \mathcal{R}$ into a data vector $z \in \mathbb{C}^{\frac{N}{2}}$, first use canonical embedding σ to obtain $z = \lfloor \Delta \cdot \pi^{-1}(z) \rceil \in \mathbb{H}$. Then divide it by the scaling factor Δ to get $\Delta^{-1} \lfloor \Delta \cdot \pi^{-1}(z) \rceil \approx \pi^{-1}(z)$. To obtain the data vector, use the π projection vector and get $\pi(\pi^{-1}(z)) = \in \mathbb{C}^{\frac{N}{2}}$.

Encryption($m(X)$, $p(X)$): In order to obtain the ciphertext polynomial $c(X)$ corresponding to the data polynomial $m(X) \in \mathcal{R}$, apply RLWE encryption and obtain $c(X) = (m(X), 0) + p(X) = (m(X) - a(X) \cdot s(X) + e(X), a(X)) \in (\mathcal{R}_q)^2 = (c_0(X), c_1(X))$.

Decryption($c(X)$, $s(X)$): In order to obtain the data polynomial corresponding to the ciphertext polynomial $c(X) \in \mathcal{R}_q$, use the secret key polynomial $s(X)$, apply RLWE to decrypt and get $m(X) \approx c_0(X) + c_1(X) \cdot s(X) = m(X) + e(X)$.

Addition($c(X)$, $c'(X)$): Two ciphertexts $c(X) = (c_0, c_1)$ and $c'(X) = (c'_0, c'_1)$ add the ciphertext $c_{add}(X) = (c_0 + c'_0, c_1 + c'_1)$.

Multiplication($c(X)$, $c'(X)$: Two ciphertexts $c(X) = (c_0, c_1)$ and $c'(X) = (c'_0, c'_1)$. Multiply to generate ciphertext $c_{mult}(X) = ((c'_0, c'_1) \cdot c_0, (c'_0, c'_1) \cdot c_1) = (c_0c'_0, c'_0c_1 + c_0c'_0, c'_0c_1) = (c_0(X), c_1(X), c_2(X))$ then the ciphertext is re-linearized and then the modulus is switched.

Relinearization($c_{mult}(X)$, $r(X)$): Re-linearization reduces the size of the ciphertext after multiplying two ciphertexts. Let $c_{mult}(X) = (c_0(X), c_1(X), c_2(X))$ be the result ciphertext after multiplying two ciphertexts. After re-linearization, the ciphertext $c_{relin}(X) = (c_0(X), c_1(X)) + \lfloor b^{-1} \cdot c_2(X) \cdot r(X) \rfloor \bmod(q)$ is obtained.

3. PROPOSED SCHEMES

In this paper, two schemes are proposed. In the first scheme, we propose LiteSEOKC to improve the conventional SEOKC method by reducing its time-consuming calculations. Besides, the original data packing method of SEOKC is based on attributes, which is not suitable for high-dimensional data. In the second scheme, a new data packing method for outsourced K-means clustering (called RP-OKC) is proposed to improve the original data packing method of SEOKC.

3.1 LiteSEOKC: Reduce Time-Consuming Calculation in SEOKC

The system architecture of LiteSEOKC is shown in Fig. 3. DO packs the data into D' by attributes, generates index values with random numbers, uses the index values to retrieve the corresponding data, encrypts them as initial clustering centers, and then sends

them to C_1 for K-means clustering. Since DO encrypts the data and outsources it to C_1 for *k*-means clustering, DO does not participate in the iterative process and receives the returned results of clustering indices only after the whole clustering process is completed by the interactions between C_1 and C_2, so C_1 and C_2 do not know the expected clustering results when they interact. The actual clustering process is calculated by C_1 and C_2 interactively. At last, C_1 sends the clustering result of clustering indices only back to DO. In our first proposed LiteSEOKC, the detail modifications to reduce the time-consuming process for SEOKC's CAM (Computation of Assignment Matrix), CNEC (Computation of New Encrypted Centers) and CNES (Computation of New Encryption Sizes) are described as follows.

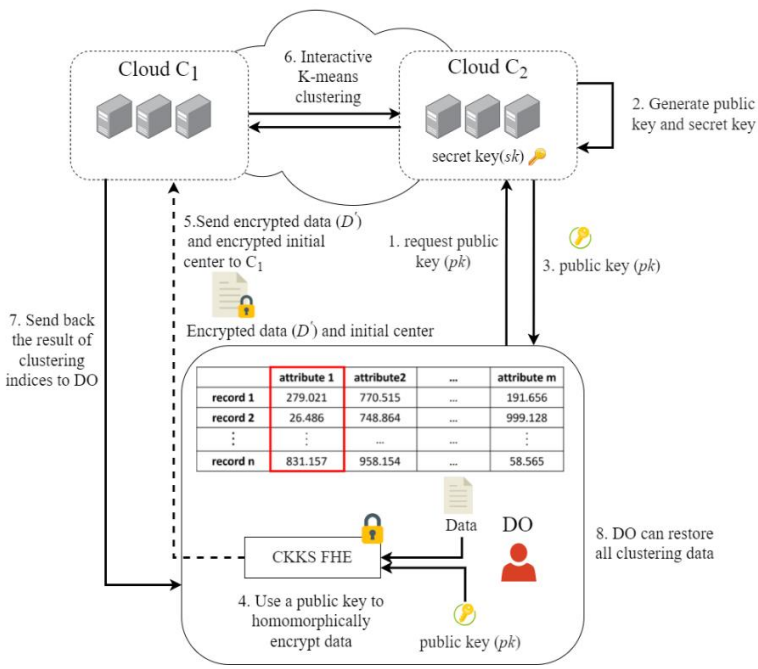


Fig. 3. LiteSEOKC system diagram.

SEOKC CAM is a security protocol executed by C_2 to find the minimum distance values to current clustering centers. While C_1 send the encrypted data to C_2, SEOKC CAM will initialize a zero matrix (*i.e.*, assignment matrix) at the beginning. The corresponding position on assignment matrix will be set to 1 when the minimum distance value and its index are found. The size of each cluster will be calculated by SEOKC CNES using this CAM assignment matrix, since the cluster sizes can be simply calculated by the sum of 1 on each column.

SEOKC CNEC is a security protocol to find the new cluster centers during the interactive K-means clustering process between non-colluding C_1 and C_2 using the cluster sizes obtained from SEOKC CNES. In CNES and CNEC of the original SEOKC, C_1 will always send C_2 with the public-key-encrypted CAM assignment matrix after randomly transposing matrix columns, C_2 will then use the secret key to decrypt the incoming data. Finally,

C_2 will send the calculated result back to C_1 after public-key encryption, C_1 will then transpose back the result of public-key-encrypted assignment matrix to original column orders.

In LiteSEOKC, we merge CNES and CNEC of SEOKC into our CNEC to find new cluster sizes and centers furnished by C_1 only without bothering C_2 for the iterative secret-key decryptions and public-key encryptions. Since C_1 doesn't have the secret key to restore the plaintext from CAM assignment matrix and known encrypted data, LiteSEOKC can still achieve the privacy protection for DO's outsourced K-means clustering data. Meanwhile, both of the chances of revealing the plaintext of clustering information to C_2 and the FHE computations of private-key decryption and public-key encryption are cost-effectively decreased.

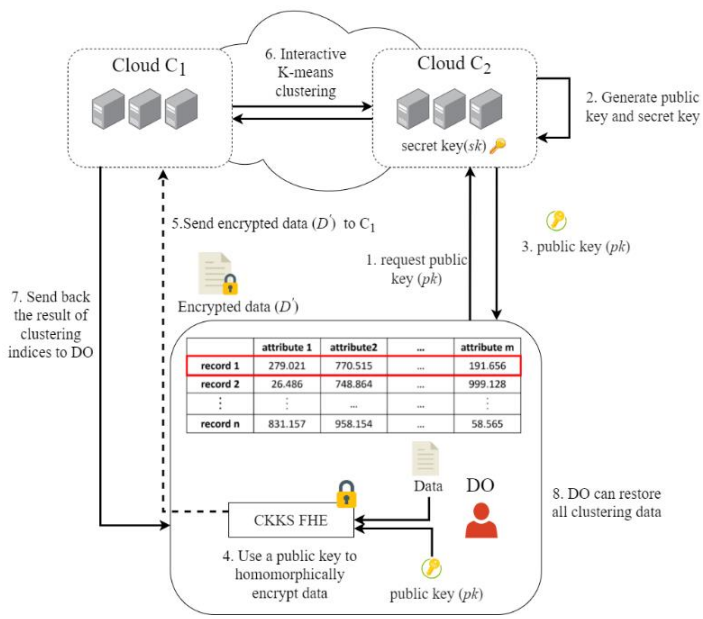


Fig. 4. RP-OKC system diagram.

3.2 RP-OKC: Record Packing Outsourced K-means Clustering

In the second proposed RP-OKC scheme of this paper, we consider the non-collusion architecture which is widely used in cloud computing for data protection [33, 34, 38-40]. This RP-OKC system architecture is shown in Fig. 4 where clouds C_1 and C_2 cannot collude with each other and are semi-honest. Assume that DO hopes to safely perform data applications (such as K-means clustering) on the cloud server. By following the basic security protocol of FHE (*i.e.* CKKS), C_2 first generates (pk, sk) as a pair of public key and secret key. The secret key sk is kept secret by C_2. DO obtains the public key from C_2. Then, DO uses the public key to homomorphically encrypt its private data according to the data record packing, and the encrypted data is outsourced to C_1 for homomorphic data application. When the termination conditions are met, the final results will return to DO for decryption.

Although CKKS FHE was shown to resolve the problem of data outsourcing privacy

security, there are some operators (such as encryption/decryption, homomorphic multiplication of ciphertext/plaintext, and rotation) still take high calculation cost. In this paper, we use an outsourced K-means clustering as an example to design new security protocols to reduce the usage of time-consuming calculation. Then, we propose a record packaging method, called RP-OKC, to improve the conventional attribute packaging method. The time sequence diagram of DO's data encryption and outsourcing is shown in Fig. 5 (a).

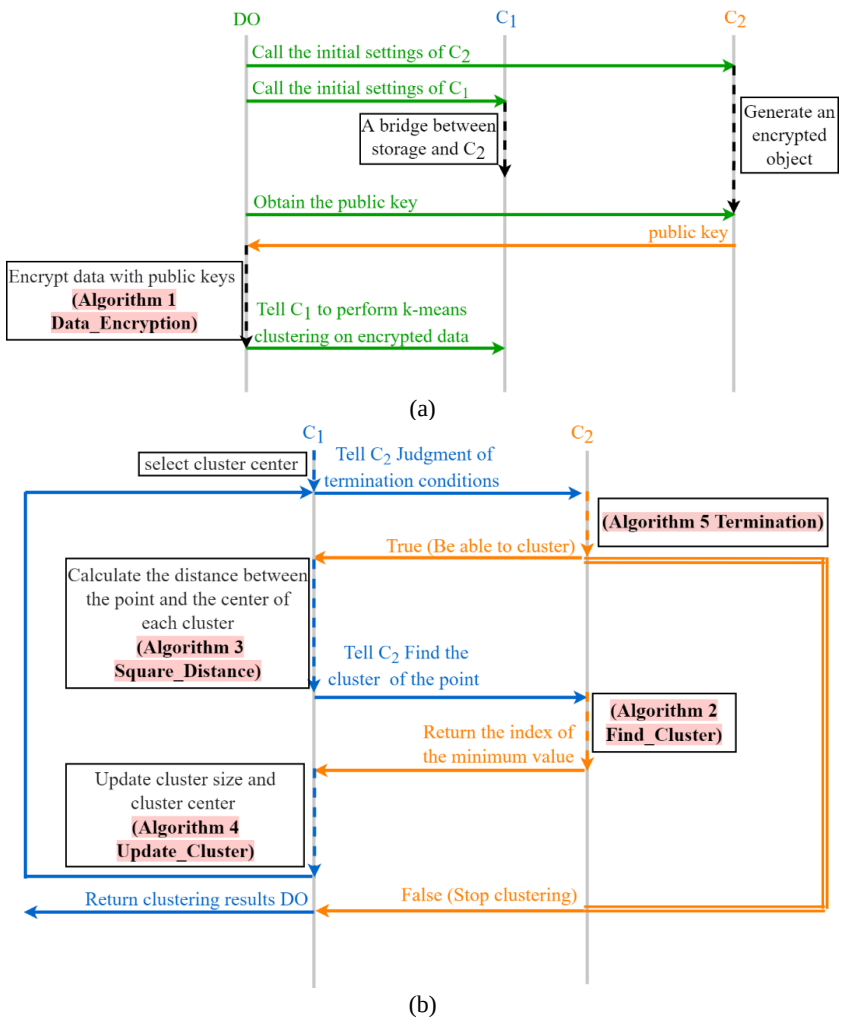


Fig. 5. (a) Timing diagram of DO encrypting the data and then outsourcing; (b) Timing diagram of K-means clustering process after receiving encrypted data.

Step 1: DO calls the initial settings of C_2 and stores the communication bridge with C_2. C_2 uses TenSEALContext [41] to encrypt the managed object and generates an object containing encryption keys and parameters. Remove the secret key from the object, and keep the public key to wait for DO to get it. The purpose is to keep the secret key secret.

Step 2: DO calls the initial settings of C_1, tells C_1 and C_2 the communication bridge, and stores the communication bridge with C_1.
Step 3: DO obtains the public key through the bridge of communication with C_2, C_2 will pass the object with the secret key removed to DO, then DO will receive the object with only the public key.
Step 4: DO uses objects with only public keys to encrypt data. After the data is encrypted, DO tells C_1 to perform K-means clustering on these encrypted data. As shown in Fig. 5 (b), C_1 selects the cluster center through random numbers. Then, it checks the termination conditions of the clustering through. C_2 decrypt the value passed in, do arithmetic to determine whether to terminate, and return True or False.
Step 4 (a): When C_1 receives True, it will do clustering to calculate the center of each cluster. C_2 will find the minimum value in the distance array passed from C_1, and return the index value of the minimum value. After C_1 receives the return of C_2, it will update the cluster. Then, it checks the termination conditions of the clustering through C_2.
Step 4 (b): When C_1 receives False, it will stop clustering and return the cluster that of the point DO.

3.2.1 Data encryption

DO first pre-processes the data and converts the data into tensor data type. The processed data is called D. DO first obtains the public key from C_2, and uses the public key to encrypt the data. The encryption method is shown in Algorithm 1. The input is processed data D. The output is the encrypted data D'. Algorithm 1 is performed by DO. z is the number of data, and m is the attribute of the data. Then, Algorithm 1 saves all the attribute values in each piece of data as an array assignment to *Pack*. It assigns *Pack* to D'_a using CKKS FHE. All D'_a is saved as an array assignment to D'. DO sends the encrypted data D' to C_1 for K-means clustering.

Algorithm 1: Encrypt_Data $(D)\rightarrow D'$

Input: database D
Output: Encrypted database D'
Performed by: DO

```
D' ← [ ]
for a = 1 to z
    Pack ← [ ]
    for s = 1 to m
        Pack.append(r_a[s])
    D'_a ← Encrypt(Pack)
    D'.append(D'_a)
```

3.2.2 Find the cluster of the point

C_2 performs the calculation of finding the cluster to which the point belongs. In Algorithm 2, C_2 first decrypts the received encrypted distance *Dis*, decrypts to plaintext array *PDis*, rounds the values in *PDis*, runs *Min*(·) to find the minimum distance *MDecDis*, finds the position *IdxDis* of *MDecDis* in *PDis*, and sends *IdxDis* back to C_1.

Algorithm 2: Find_Cluster(Dis) → $IdxDis$

Input: Encrypted distance Dis
Output: Minimum index $IdxDis$
Performed by: C_2

1: $PDis \leftarrow [\ ]$
2: **for** $j = 1$ **to** k
3: $PDis.append(Decrypt(Dis_{aj}))$
4: $PDis \leftarrow Roung(PDis, decimals = 4)$
5: $MDecDis \leftarrow \text{Min}(PDis)$
6: $IdxDis \leftarrow PDis.index(MDecDis)$

3.2.3 Calculate the distance between the point and the center of each cluster

Algorithm 3 is performed by C_1, takes the encrypted data D', the cluster center *Center*, and the belonging cluster *Cluster* as input. It outputs the updated cluster *Cluster* and the encrypted cluster distance *DisCluster*. C_1 first calculates the Euclid distance (Steps 1-11). When the length of the calculated distance array *Dis* is equal to k, it means that the data is not the cluster center. The calculated distance array *Dis* is passed to C_2, and the index value *IdxDis* of the minimum value in the distance array *Dis* is obtained through C_2. C_1 obtains the index value *IdxDis*, uses the index value to find the encrypted minimum distance *MEncDis*, stores it in the encrypted cluster distance *DisCluster*, and updates the cluster to which it belongs (Steps 12-16). If the data is the cluster center, store zero in the encrypted cluster distance *DisCluster* (Steps 17-18).

Algorithm 3: Square_Distance (D', *Center*, *Cluster*)→*Cluster*, *DisCluster*

Input: Encrypted database D', cluster centers *Center* and Belongs to the cluster *Cluster*
Output: Belongs to the cluster *Cluster* and Encrypted cluster distance *DisCluster*
Performed by: C_1

1: $DisCluster \leftarrow [\ ]$
2: **for** $a = 1$ **to** z
3: $Dis \leftarrow [\ ]$
4: **for** $j = 1$ **to** k
5: if $D'_a \neq Center_j$
6: $\beta \leftarrow [\ ]$
7: **for** $s = 1$ **to** m
8: $\beta_s \leftarrow (D'_{as} \ominus Center_j[s])^2$
9: $\beta.appned(\beta_s)$
10: $Dis_{aj} \leftarrow Sum(\beta)$
11: $Dis.appned(Dis_{aj})$
12: **if** len(Dis) = k
13: Get the minimum index *IdxDis* through C_2 /* from Algorithm 2 */
14: $MEncDis \leftarrow Dis[IdxDis]$
15: $DisCluster.append(MEncDis)$
16: $Cluster[a] = IdxDis + 1$
17: **else**
18: $DisCluster.append(0)$

3.2.4 Update cluster center

Algorithm 4 is executed by C_1, takes the encrypted data D', the *Cluster* to which it belongs, and the encrypted cluster distance *DisCluster* as input, and outputs the updated cluster center *NewCenter* and the sum of squared errors (SSE) within the new clusters *NewClusterSize*. The initial value setting (Steps 1-4), the method of updating the cluster center is to add up all the points of the cluster and divide by the number of the cluster (Steps 5-18). SSE within the new clusters *NewClusterSize* is to add the encrypted cluster distance *DisCluster* (Steps 19-21).

Algorithm 4: Update_Cluster (D', *Cluster*, *DisCluster*) $\rightarrow$ *NewCenter*, *NewCluster-Size*

Input: Encrypted database D', Belongs to the cluster *Cluster* and Encrypted cluster distance *DisCluster*
Output: Updated cluster center *NewCenter* and SSE within the new clusters *New-ClusterSize*
Performed by: C_1

1: *TmpNewCenter* ← [ ]
2: **for** $j = 1$ **to** k
3: *TmpNewCenter.update*({j: 0})
4: *Count* ← [0]×k
5: **for** $a = 1$ **to** z
6: **if** *Cluster*[a] **in** *TmpNewCenter*
7: *Temp* ← *TmpNewCenter.get*(*Cluster*[a])
8: *Count*[*Cluster*[a] − 1] = *Count*[*Cluster*[a] − 1] + 1
9: **for** $s = 1$ **to** m
10: *Temp* = *Temp* ⊕ D'_a
11: *TmpNewCenter.update*({*Cluster*[a]: *Temp*})
12: *NewCenter* ← [ ]
13: **for** $j = 1$ **to** k
14: *Temp* ← *TmpNewCenter.get*(j)
15: *TempCount* ← 1 ÷ *Count*[j − 1]
16: **for** $s = 1$ **to** m
17: *Temp* ← *Temp* ⊠ *TempCount*
18: *NewCenter.append*(*Temp*)
19: *NewClusterSize* ← 0
20: **for** $a = 1$ **to** z
21: *NewClusterSize*← *NewClusterSize*⊕*DisCluster*[a]

3.2.5 Check termination conditions

Algorithm 5 is executed by C_2, which takes SSE within the new clusters *NewCluster-Size* and SSE within the old clusters *OldClusterSize* as input, and outputs the termination condition *Flag*. If the difference between SSE within the new and old clusters is less than 1, set flag to False. Otherwise, set to True.

Algorithm 5: Termination(*NewClusterSize, OldClusterSize*)→*Flag*

Input: SSE within the new clusters *NewClusterSize* and old cluster size *OldClusterSize*
Output: Termination condition *Flg*
Performed by: C_2
1: **if** *abs*(*Decrypt*(*NewClusterSize*) − *Decrypt*(*OldClusterSize*)) ≤ 1
2: *Flag* ← *False*
3: **else**
4: *Flag* ← *True*

3.2.6 The methods applied in RP-OKC

Algorithm 6 for furnishing RP-OKC is executed by C_1, taking the encrypted data D' and the number of clusters k needed to be divided as input, and outputs the last clustering result. First randomly generate k integers *CenterIndex* and takes out the corresponding data in the data set as the cluster center *Center*. Initial value setting (Steps 2-3), update the cluster according to *Center* (Steps 4-6), set the initial value of the new cluster SSE and the old cluster SSE. According to the proposed termination condition in Algorithm 5, check whether to continue clustering. After confirming to continue clustering, use the calculation distance and encrypted cluster distance (in Algorithm 3) to obtain the updated cluster *Cluster* and the encrypted cluster from the distance *DisCluster*. Through Algorithm 4, the updated cluster center *NewCenter* and SSE within the new clusters *NewClusterSize* are obtained. The updated cluster center is set as the cluster center, and check whether to continue clustering (Algorithm 5). After receiving the stop clustering, the last cluster to belong to is passed to DO.

Algorithm 6: *k*Means(*D′*, *k*)→*Cluster*

Input: Encrypted database *D′* and Number of cluster *k*
Output: Belongs to the cluster *Cluster*
Performed by: C_1
1: Randomly generate *k* integer numbers *CenterIndex* and take out the corresponding data from the dataset as the center of the cluster *Center*
2: *Cluster* ← [None] ∗ *z*
3: *TempK* ← 0
4: **for** *j* = 1 **to** *k*
5: *Cluster*[*CenterIndex*[*j*]] = *TempK* + 1
6: *TempK* ← *TempK* + 1
7: *NewClusterSize* ← 100
8: *OldClusterSize* ← 0
9: *Flag* ← Termination(*NewClusterSize*, *OldClusterSize*)
10:**While** *Flag*
11: *OldClusterSize* ← *NewClusterSize*
12: *Cluster*, *DisCluster*←Square_Distance(*D′*, *Center*, *Cluster*)
13:*NewCenter*, *NewClusterSize*←Update_Cluster(*D′*, *Cluster*, *DisCluster*)
14: *Center*←*NewCenter*
15: *Flag*←Termination(*NewClusterSize*, *OldClusterSize*)
16: **if** *Flag* = *False*
17: Send *Cluster* to DO

Since clustering is to group all data into similar groups together, each data can only belong to one group, and each group is called a cluster. Assume DO has a dataset $D = \{r_1, r_2, \ldots, r_z\}$, with z pieces of data, and each data record $r_a (a \in [z])$ consists of m attributes denoted as $r_a[s]$ for $s \in [m]$. Apply its data to traditional K-means clustering [42], the algorithm divides the dataset D into k clusters $\{cluster_1, cluster_2, \ldots, cluster_k\}$, and it is hoped that the inter-cluster similarity of the clustering results is low and intra-cluster similarity is high. Using Euclidean distance as a similarity measure, there are four main stages: (1) Initialization; (2) Find the cluster of the point; (3) Update cluster center; and (4) Termination.

Stage (1): in order to initialize k clusters $cluster_1, cluster_2, \ldots, cluster_k$, DO selects k index values by random numbers and sets the data corresponding to the index values as cluster centers μ_1, μ_1, μ_k.

Stage (2): The K-means clustering algorithm calculates the Euclidean distances between each $record_a$ and μ_j for $a \in [z]$ and $j \in [k]$, which is given in Eq. (1) as follows

$$\| r_a - \mu_j \| = \sqrt{\sum_{s=1}^{m} (r_a[s] - \mu_j[s])^2}, \tag{1}$$

where $\mu_j[s]$ denotes the sth attribute of μ_j. Based on the Euclidean distance, K-means clustering algorithm determines the cluster center closest to r_a (such as μ_j) and assigns r_a to a new cluster $cluster'_j$, where $j \in [k]$.

Stage (3): The K-means clustering algorithm calculates the mean values of all data records in the corresponding cluster $cluster'_j$, which is the new cluster center $\mu'_1, \mu'_2, \ldots, \mu'_k$. Let $cluster'_j = \{r_1, \ldots, r_{|cluster'_j|}\}$ with a cluster size of $|cluster'_j|$, then the sth attribute of μ'_j can be expressed as Eq. (2)

$$\mu'_j = \frac{r_1[s] + r_1[s] + \ldots + r_{|cluster'_j|}[s]}{|cluster'_j|}. \tag{2}$$

Stage (4): The K-means clustering algorithm uses the within-cluster SSE, as in Eq. (3), to determine SSE within the new and old clusters determine whether to terminate the clustering process. If the difference between SSE within the new and old clusters is less than 1, the K-means clustering algorithm stops and returns the final clustering result. Otherwise, the algorithm uses the new cluster center as input to continue with the next iteration (*i.e.* Stage (2)).

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n} (r_i - \mu'_j)^2 \tag{3}$$

4. EXPERIMENTAL RESULTS AND ANALYSIS

There are four parts in this section to demonstrate the experimental performance for proposed methods. In the first part, the calculation cost of SEOKC and LiteSEOKC are compared. In the second part, analysis for various calculation costs based on record packing method proposed in this paper is through experiments, to establish a table of calculation costs and use a proportional method to evaluate computing performance. In the third part,

the accuracy of RP-OKC was tested. In the last part, the calculation cost of SEOKC, LiteSEOKC and RP-OKC are all analyzed and compared. Our experiments were performed on a Windows system with an Intel Core i7-7700HQ 2.80 GHz CPU and 24 GB of RAM. The TenSEAL library [41] was used to implement the proposed methods of this paper, the degree of the CKKS modulus polynomial was 8192, and four prime numbers with sizes of 60, 40, 40 and 60 bits were created.

4.1 Comparison of Calculation Cost Between SEOKC and LiteSEOKC

We use a synthetic database containing 1000 records with 5, 10 and 15 attributes, generated randomly in the floating point domain among [0, 1000]. These data were clustered into 3 clusters. We compare the calculation cost of SEOKC with LiteSEOKC by computing one iteration of K-means clustering. The experimental results are shown in the Fig. 6.

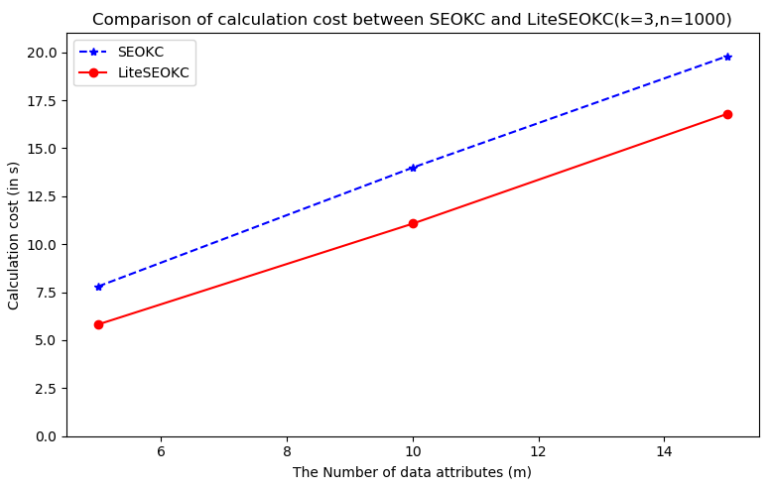


Fig. 6. Comparison of calculation cost between SEOKC and LiteSEOKC.

4.2 Analysis of Various Calculation Costs Based on Record Packing

In this paper, we analyze various calculation costs through experiments. By randomly generating 10 integer random numbers in the range of 0~1000 as attributes, 10,000 records are generated. Use these 10,000 pieces of data to analyze the calculation cost of encryption, decryption, additive/subtractive/multiplicative homomorphic on ciphertext, and additive/multiplicative homomorphic on ciphertext-plaintext, and general addition, subtraction, multiplication and division, and rotation. Each calculation method is executed 10 times and then the average value is taken. At the same time, various calculation costs of one attribute are also calculated. The calculation cost statistics results after execution are shown in Table 1, time unit for seconds, it can be seen that the calculation cost of 10 attributes and 1 attribute is almost due to parallel processing. Calculate the proportion of each calculation cost with 10,000 pieces of data and 1 attribute, set X with the time of general division as the denominator, do the average calculation cost (ACC), and round the result of the calculation to the first place of the integer. The calculated result is as follows in Table 1.

Table 1. Calculate cost statistics (The unit is second).

Calculations	10,000 records with 10 attributes	10,000 records with1 attribute
encryption	58.046	56.851
decryption	14.513	14.509
additive homomorphism on ciphertext	1.223	1.221
subtractive homomorphism on ciphertext	1.218	1.215
multiplicative homomorphism on ciphertext	42.659	42.657
additive homomorphism on ciphertext-plaintext	8.756	8.535
multiplicative homomorphism on ciphertext-plaintext	17.180	17.013
General addition	0.054	0.053
General subtraction	0.050	0.050
General multiplication	0.050	0.047
General division	0.048	0.047
Rotation	14.513	14.509

Table 2. Percentage of calculation cost.

Calculations	10,000 records with 1 attribute	Proportion (round off to first digit of integer)
encryption	56.851	1210X
decryption	14.509	309X
additive homomorphism on ciphertext	1.221	26X
subtractive homomorphism on ciphertext	1.215	26X
multiplicative homomorphism on ciphertext	42.657	908X
additive homomorphism on ciphertext-plaintext	8.535	182X
multiplicative homomorphism on ciphertext-plaintext	17.013	362X
General addition	0.053	1X
General subtraction	0.050	1X
General multiplication	0.047	1X
General division	0.047	1X
Rotation	14.509	309X

We use the calculation cost in Table 2 to get the percentage of each calculation cost in RP-OKC as shown in Table 3. The graph is drawn as shown in Fig. 7. It can be seen that the more time-consuming calculations are encryption, multiplicative homomorphism, ciphertext-plaintext multiplicative homomorphism, decryption and rotation.

The calculation cost of RP-OKC was estimated from the experimentally obtained calculation cost ratio table in Section 4.2. First, count all the calculations used in RP-OKC, as shown in the following Table 4, where z is the number of data items, k is the number of clusters, m is the number of attributes, c is a constant, and $*$ is repeated execution.

Because the experimental results show that the calculation results of 1 attribute and 10 attributes are similar, it is assumed that 10,000 data are divided into 3 clusters, and each data has 1 attribute, $z = 10000$, $k = 3$, $m = 1$, $c = 1$. According to the statistical results in Table 4.

Table 3. Percentage of each calculation cost.

Calculations	Proportion (round off to the first digit of the integer)
encryption	0.1210
decryption	0.0309
additive homomorphism on ciphertext	0.0026
subtractive homomorphism on ciphertext	0.0026
multiplicative homomorphism on ciphertext	0.0908
additive homomorphism on ciphertext-plaintext	0.0182
multiplicative homomorphism on ciphertext-plaintext	0.0362
General addition	0.0001
General subtraction	0.0001
General multiplication	0.0001
General division	0.0001
Rotation	0.0309

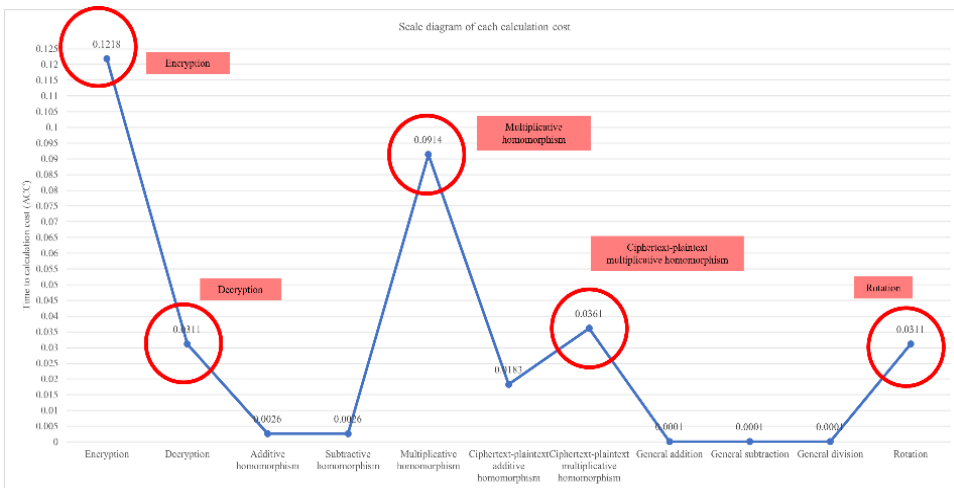


Fig. 7. Scale diagram of each calculation cost.

Table 4. Count all the calculations used in RP-OKC.

Calculations	Applied parameters	Ratio
Encryption	zm	1210.0000
Decryption	$zk + c^*$	927.0309
Additive homomorphism	$zkm + zm + km + z$	130.0078
Subtractive homomorphism	zkm	78.0000
Multiplicative homomorphism	$3zkm$	8172.0000
Ciphertext-plaintext additive homomorphism	0	0.0000
Ciphertext-plaintext multiplicative homomorphism	km	0.1086
General addition	$2z + k$	2.0003
General subtraction	c^*	0.0001
General division	k	0.0003
Rotation	0	0.0000

4.3 RP-OKC Security Analysis and Accuracy Test

Based on the security analysis of SEOKC [26], security analysis to verify the security of our proposed RP-OKC is briefly illustrated as follows:

Security of **Algorithm 1 Encrypt_Data:**

This security can be directly achieved from the applied efficient CKKS encryption. Since CKKS-encrypted data records are only submitted to C_1 without colluding with C_2, private data are protected from these two clouds in semantic security.

Security of **Algorithm 2 Find_Cluster:**

Theorem 1. The proposed protocol of "Find the cluster of the point" is secure against the semi-honest C_2 as long as the CKKS encryption is semantically secure and the non-collusion assumption between C_1 and C_2 is satisfied.

***Proof*:** To prove the security of "Find the cluster of the point protocol" under the semi-honest model, we need to demonstrate that the simulated image of **Algorithm 2 Find_Cluster** is computationally indistinguishable from its actual execution image. An execution image generally contains the exchanged messages and the results computed from these messages. According to **Algorithm 2 Find_Cluster**, we have the execution image of C_2 denoted as $\Pi_{C_2}(\text{Find\_Cluster}) = \{Dis_{aj}, IdxDis_{aj}\}$ for $a, j \in [z]$, $[k]$, where Dis_{aj} are CKKS ciphertexts. Let the simulated image of C_2 be $\Pi^{S}_{C_2}(\text{Find\_Cluster}) = \{\theta^1_{aj}, \gamma^1_{aj}\}$, where $\theta^1_{aj}(a, j \in [z], [k])$ are randomly generated from the ciphertext space of CKKS cryptosystem. Since the CKKS encryption is semantically secure, it implies that Dis_{aj} are computationally indistinguishable from θ^1_{aj}. Since we randomly initialize the cluster centers in our RP-OKC method, the real assignment values $IdxDis_{aj}$ are randomly distributed, which are computationally indistinguishable from $\gamma^1_{aj}(a, j \in [z], [k])$. However, the information leaked to C_2 by the plaintext distance PDis_j for $j \in [k]$ in Algorithm 2 Find_Cluster. Since PDis_j is the result of the calculation of two raw data records containing noise. Although C_2 can know the ratios of difference between plaintext distance PDis_j. But these ratios contain privacy information and are not sufficient for inferring data records. Therefore, the leakage of information has little impact on the security of the Find the cluster of the point protocol. Based on these analyses, $\Pi_{C_2}(\text{Find\_Cluster})$ is computationally indistinguishable from $\Pi^{S}_{C_2}(\text{Find\_Cluster})$ based on Theorem 1, which implies that C_2 cannot learn any private information during the execution of Find the cluster of the point.

Security of **Algorithm 3 Square_Distance:**

Theorem 2: The proposed **Algorithm 3** for protocol of "Calculating the distance between the point and the center of each cluster" is secure against the semi-honest C_1 as long as the CKKS encryption is semantically secure and the non-collusion assumption between C_1 and C_2 is satisfied.

***Proof*:** To prove the security of "Calculate the distance between the point and the center of each cluster" under the semi-honest model, we need to demonstrate that the simulated image of **Algorithm 3 Square_Distance** is computationally indistinguishable from its actual execution image. According to **Algorithm 3**, we have the execution image of C_1 denoted as $\Pi_{C_1}(\text{Square\_Distance}) = \{Dis_{aj}\}$ for $a, j \in [z]$, $[k]$, where Dis_{aj} are CKKS ciphertexts. Let the simulated image of C_1 be $\Pi^{S}_{C_1}(\text{Square\_Distance}) = \{\theta^2_{aj}\}$, where $\theta^2_{aj}(a, j \in [z],$

[k]) are randomly generated from the ciphertext space of CKKS cryptosystem. Since the CKKS encryption is semantically secure, it implies that Dis_{aj} are computationally indistinguishable from θ_{aj}^2. Therefore, Π_{C_1}(Square_Distance) is computationally indistinguishable from $\Pi_{C_1}^S$(Square_Distance) based on Theorem 2. $DisCluster_a$ for $a \in [z]$ are semantically secure CKKS ciphertexts. Therefore, by using the same analysis method in Theorem 2. Since *Dis* is an array of semantically safe encrypted distances, C_1 cannot compute the correspondence between data records and clusters *Cluster*. This means that C_1 cannot learn any private information during the execution of Calculate the distance between the point and the center of each cluster.

Security of **Algorithm 4 Update_Cluster:**

To prove the security of "Update cluster center" under the semi-honest model, we need to demonstrate that the simulated image of **Algorithm 4 Update_Cluster** is computationally indistinguishable from its actual execution image. According to **Algorithm 4 Update_Cluster**, we have the execution image of C_1 denoted as Π_{C_1}(Update_Cluster) = {$NewCenter_j[s]$, $NewClusterSize$} for $j, s \in [k]$, $[m]$, where $NewCenter_j[s]$, $NewClusterSize$ are CKKS ciphertexts. Therefore, by using the same analysis method in Theorem 2. This means that C_1 cannot learn any private information during the execution of Update cluster center.

Security of **Algorithm 5 Termination:**

According to **Algorithm 5** of "Check termination conditions", we have the execution image of C_2 denoted as Π_2(Termination) = {*NewClusterSize*, *OldClusterSize*}. Except that *NewClusterSize*, *OldClusterSize* are set to the initial value of plaintext at the beginning, in other cases, *NewClusterSize*, *OldClusterSize* are CKKS ciphertexts. Therefore, by using the same analysis method in Theorem 1. This means that C_2 cannot learn any private information during the execution of Check of termination conditions.

Security of **Algorithm 6 *k*Means:**

As shown in **Algorithm *k*Means,** the furnished RP-OKC method is a combination of the proposed 5 security protocols mentioned above, we have already proved the security for all these algorithms. Since the two clouds cannot collude with each other, C_1 and C_2 cannot figure out the assignments between data records and clusters. Therefore, data access patterns are hidden from both of them. Although the minimum distance between each point and k clusters is revealed to C_2 in the Find the cluster of the point protocol, the data cannot be understood from it because C_2 does not know the number of attributes of the data and the value of k clusters.

Furthermore, our experiments conducted to compare the results of one iteration of general K-means clustering with proposed RP-OKC. The approach is to use customer data on Kaggle [43]. The attributes Annual_Income_(k$) and Spending_Score are selected and all duplicates in them are packaged according to the data records, and the packaged data are encrypted and passed to C_1. C_1 receives these data and generates a fixed random number with a random seed. This random number seed is executed 1000 times from 0 to 999. If the decryption result is rounded to the first to eighth decimal place in **Algorithm 2 Find_Cluster.** For these eight test cases, the result of one iteration in these 1000 executions will be the same as the general K-means clustering.

4.4 Analysis and Comparison of Calculation Cost Between SEOKC, LiteSEOKC and RP-OKC

Encryption is the encoding of data to protect the privacy of the data so that it cannot be accessed by unauthorized parties. Unencrypted means that the data itself, without any protection, can be easily viewed and accessed. The difference between encrypted and unencrypted data is shown in Table 5 below:

Table 5. Difference between clustering encrypted/unencrypted data.

	Encrypted	Unencrypted
Definition	*Encode data*	*Original data*
Called	*Ciphertext*	*Plaintext*
Security	*Secure*	*Insecure*
Data Access	*Only authorized party can access data*	*Anyone can access data*

Experiments are conducted to compare and evaluate the performance of clustering between encrypted and unencrypted data. This experimental method is to fix the number of data records to 500 and the number of attributes as 100, 200, 500 and 1000, and to check the calculation cost of the three methods respectively. It can be seen that the calculation cost of this RP-OKC is lower than that of LiteSEOKC after the number of attributes is 200. The results are shown in Fig. 8, then the RP-OKC method is suitable for use in high-dimensional data.

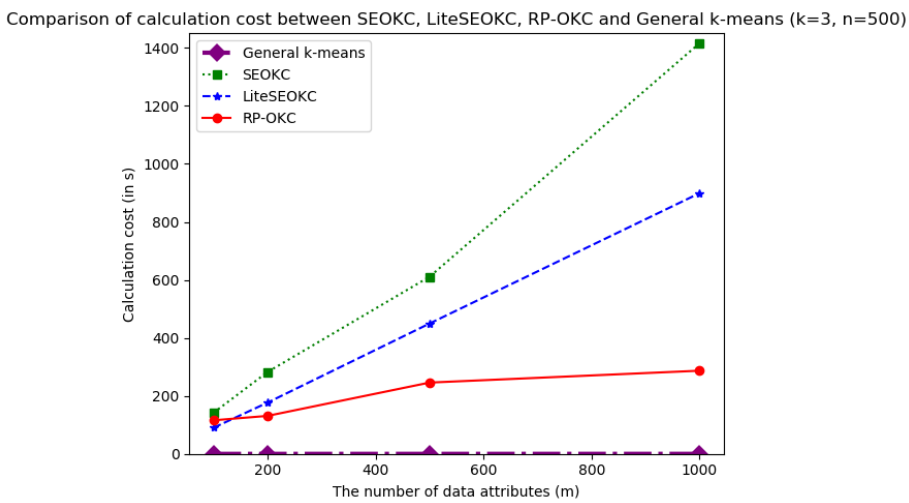


Fig. 8. Comparison of calculation cost between SEOKC, LiteSEOKC, RP-OKC and general K-means (k = 3, n = 500).

In Fig. 8, the RP-OKC method is suitable for attribute number in greater than 200. The following experimental method is to fix the data record. In this experiment, we fix the attribute number to 500, the number of recorded data is 100 to 500, with interval 100. To

observe the calculation costs of the three methods respectively, the result is shown in Fig. 9. In the method with encryption, it can be seen that the calculation cost of RP-OKC is lower than SEOKC and LiteSEOKC.

The computational cost of clustering unencrypted data is the lowest because both encrypting and computing encrypted data are more computationally expensive, but clustering encrypted data is a way to protect data security. The above two experiments show that RP-OKC is suitable for high-dimensional data among the methods of clustering encrypted data.

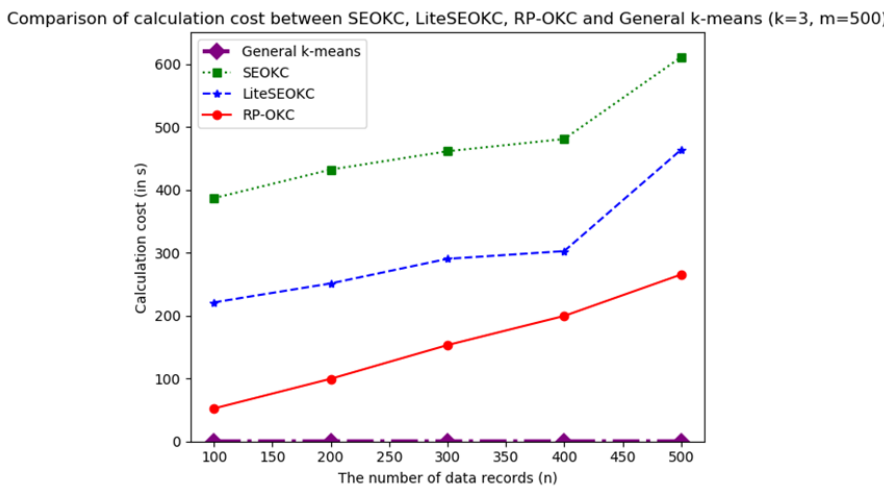


Fig. 9. Comparison of calculation cost between SEOKC, LiteSEOKC, RP-OKC and general K-means (k = 3, n = 500).

5. CONCLUSIONS AND FUTURE WORK

After finding the CKKS FHE operations of encryption, multiplicative homomorphism, ciphertext-plaintext multiplicative homomorphism, decryption and rotation operations are time-consuming calculation, we first propose LiteSEOKC to improve the SEOKC efficiency by slimming CKKS FHE operations. In order to meet more needs of diversified-attribute data for users in secure and efficient outsourced K-means clustering, we further propose a new security protocol called RP-OKC with a new diversified-attribute data-record packing method for more secure and efficient outsourced K-means clustering. There are 5 security protocols in the proposed RP-OKC scheme, including data encryption, find the cluster of the point, calculate the distance between the point and the center of each cluster, update cluster center, and check termination conditions. It also uses these 5 security protocols to achieve privacy for outsourcing encryption database in MLaaS. The calculation cost of SEOKC and LiteSEOKC for large number of data records were conducted in our experiments. The performance results show that LiteSEOKC is more efficient than original SEOKC and more suitable for applications using large database. The further performance results of computational costs of SEOKC, LiteSEOKC and RP-OKC with high-

dimensional data demonstrate that the RP-OKC method is more applicable to high-dimensional database than both LiteSEOKC and original SEOKC. In this paper, more effective privacy protection is proposed for outsourcing encryption database in MLaaS using K-means clustering as an example.

In the near future, we hope to meet the needs of more users and various applications, such as using different ML and deep learning for calculations. In addition, the HE technology built on the cloud platform is used for calculations. This results in users not knowing the accuracy of the calculation results. We hope to provide verifiable methods for evaluating the accuracy of the calculation results for their services in the future.

REFERENCES

1. "Statista," https://www.statista.com/, 2021.
2. "Snowflake," https://www.snowflake.com/data-marketplace/, 2021.
3. "Datarade," https://datarade.ai/, 2021.
4. "Alibaba cloud machine learning platform for AI," https://www.alibabacloud.com/tc/product/machine-learning, 2021.
5. "AWS machine learning," https://aws.amazon.com/tw/machine-learning/, 2021.
6. "Microsoft Azure AI," https://azure.microsoft.com/zh-tw/overview/ai-platform/, 2021.
7. "Google cloud AI," https://cloud.google.com/products/ai, 2021.
8. L. Brandeis and S. Warren, "The right to privacy," *Harvard Law Review*, Vol. 4, 1890, pp. 193-220.
9. R. H. Coase, "The problem of social cost," *Classic Papers in Natural Resource Economics*, Springer, 1960, pp. 87-137.
10. A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, Vol. 25, 1968, p. 166.
11. R. Gavison, "Privacy and the limits of law," *The Yale Law Journal*, Vol. 89, 1980, pp. 421-471.
12. E. van den Haag, "On privacy," *Privacy & Personality*, Routledge, 2017, pp. 149-168.
13. X. Yi, R. Paulet, and E. Bertino, "Homomorphic encryption," *Homomorphic Encryption and Applications*, Springer, 2014, pp. 27-46.
14. Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proceedings of Annual Cryptology Conference*, 2011, pp. 505-524.
15. Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, Vol. 6, 2014, pp. 1-36.
16. A. Acar *et al.*, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, Vol. 51, 2018, pp. 1-35.
17. J. Kim, S. Kim, and J. H. Seo, "A new scale-invariant homomorphic encryption scheme," *Information Sciences*, Vol. 422, 2018, pp. 177-187.
18. K. C. Laudon, *Dossier Society: Value Choices in the Design of National Information Systems*, Columbia University Press, 1986.
19. Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proceedings of Annual Cryptology Conference*, 2012, pp. 868-886.

20. J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptology ePrint Archive*, Vol. 2012, 2012, p. 144.
21. J. H. Cheon *et al.*, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, 2017, pp. 409-437.
22. I. Chillotti *et al.*, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Proceedings of International Conference on Theory and Application of Cryptology and Information Security*, 2016, pp. 3-33.
23. I. Chillotti *et al.*, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, 2017, pp. 377-408.
24. I. Chillotti *et al.*, "TFHE: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, Vol. 33, 2020, pp. 34-91.
25. P.-E. Clet, O. Stan, and M. Zuber, "BFV, CKKS, TFHE: Which one is the best for a secure neural network evaluation in the cloud?" in *Proceedings of International Conference on Applied Cryptography and Network Security*, 2021, pp. 279-300.
26. W. Wu *et al.*, "Secure and efficient outsourced K-means clustering using fully homomorphic encryption with ciphertext packing technique," *IEEE Transactions on Knowledge and Data Engineering*, Vol. --, 2020, pp. --.
27. D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced K-means clustering," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 123-134.
28. N. Almutairi, F. Coenen, and K. Dures, "K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction," in *Proceedings of International Conference on Big Data Analytics and Knowledge Discovery*, 2017, pp. 274-285.
29. Y. Wang, "Notes on two fully homomorphic encryption schemes without bootstrapping," *IACR Cryptology ePrint Archive*, Vol. 2015, 2015, p. 519.
30. Y. Huang, Q. Lu, and Y. Xiong, "Collaborative outsourced data mining for secure cloud computing," *Journal of Networks*, Vol. 9, 2014, pp. 2655.
31. W. K. Wong *et al.*, "Secure kNN computation on encrypted databases," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2009, pp. 139-152.
32. K.-P. Lin, "Privacy-preserving kernel K-means clustering outsourcing with random transformation," *Knowledge and Information Systems*, Vol. 49, 2016, pp. 885-908.
33. F.-Y. Rao *et al.*, "Privacy-preserving and outsourced multi-user K-means clustering," in *Proceedings of IEEE Conference on Collaboration and Internet Computing*, 2015, pp. 80-89.
34. H. Rong *et al.*, "Privacy-preserving-means clustering under multiowner setting in distributed cloud environments," *Security and Communication Networks*, Vol. 2017, 2017, , Article ID 3910126.
35. A. Alabdulatif *et al.*, "Privacy-preserving data clustering in cloud computing based on fully homomorphic encryption," in *Proceedings of the 21st Pacific Asia Conference on Information Systems*, 2017, No. 301373083.
36. S. Panda, "Principal component analysis using CKKS homomorphic scheme," in *Proceedings of the 5th International Symposium on Cyber Security Cryptography and*

Machine Learning, 2021, pp. 52-70.
37. J. Liu *et al.*, "Secure KNN classification scheme based on homomorphic encryption for cyberspace," *Security and Communication Networks*, Vol. 2021, 2021, Article ID 8759922.
38. B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "K-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, 2014, pp. 1261-1273.
39. H. Rong *et al.*, "Privacy-preserving *k*-nearest neighbor computation in multiple cloud environments," *IEEE Access*, Vol. 4, 2016, pp. 9589-9603.
40. W. Wu *et al.*, "Efficient *k*-nearest neighbor classification over semantically secure hybrid encrypted cloud database," *IEEE Access: Practical Innovations, Open Solutions*, Vol. 6, 2018, pp. 41771-41784.
41. "TenSEAL," https://github.com/OpenMined/TenSEAL.
42. S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28, 1982, pp. 129-137.
43. "Kaggle customer data," https://www.kaggle.com/shrutimechlearn/customer-data/version/1.

Ray-I Chang received his Ph.D. degree in Electrical Engineering and Computer Science from National Chiao Tung University in 1996, where he was a member of Operating Systems Laboratory. Then, he joined Computer Systems and Communications Laboratory in Institute of Information Science, Academia Sinica, to develop video-on-demand servers and digital library systems. In 2002, he joined the Department of Information Management, National Central University. Now, he is a Professor in the Department of Engineering Science and Ocean Engineering, National Taiwan University. Dr. Chang has published over 300 papers, including papers published in IEEE Transactions on Multimedia, IEEE Transactions on Broadcasting, and IEEE Transactions Neural Networks. His current research interests include multimedia networking and data mining. Dr. Chang is a member of IEEE, CERP and IICM.

Yen-Ting Chang received B.S. degree in Computer Science and Information Engineering from Ming Chuan University in 2020. Now, she is a master student at the department of Engineering Science and Ocean Engineering, National Taiwan University. Her main research interests include privacy protection and machine learning.

Chia-Hui Wang received BS degree in Computer Science from Tamkang University in 1986 and received MS degree in Computer Science from New Jersey Institute of Technology, USA, in 1991. Then, he received Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in 2002. His industry experience included four years as a software developer in high-tech industry. Now, he is an Associate Professor of Department of Computer Science and Information Engineering, Ming Chuan University. His research interests include multimedia communications, multimedia security and embedded systems. He is a member of IEEE and ACM.