Secure and Evaluable Clustering based on a Multifunctional and Privacy-Preserving Outsourcing Computation Toolkit

JIALIN LI¹, PENGHAO LU², AND XUEMIN LIN.³, (Member, IEEE)

¹School of Software Engineering, East China Normal University, Shanghai 200062, China (e-mail: katleesh@outlook.com)
²School of Software Engineering, East China Normal University, Shanghai 200062, China (e-mail: lupenghao1997@163.com)
³School of Software Engineering, East China Normal University, Shanghai 200062, China

ABSTRACT Although tremendous revolution has been made in the emerging cloud computing technologies over digital devices, privacy gradually becomes a big concern in outsourcing computation. Homomorphic encryption has been proposed to facilitate the preservation of data privacy while computational tasks being executed on ciphertext. However, many existing studies only support limited homomorphic calculation functions which barely satisfy complex computing tasks such as machine learning with massive computing resources and rich types of function. To address this problem, a novel multifunctional and privacy-preserving outsourcing computation toolkit is proposed in this paper, which supports several homomorphic computing protocols including division and power on ciphertext of integers and floating point numbers. Specifically, we first implement the homomorphic mutual conversion protocol between integer and floating point ciphertext to balance the efficiency and feasibility, considering the high-precision ciphertext operation on floating point numbers costs 100x computational overhead than that on integers. Second, we implement a homomorphic K-means algorithm based on our proposed toolkit for clustering and design the homomorphic silhouette coefficient as the evaluation index, thereby providing an informative cluster assessment for local users with limited resources. Then, we simulate the protocols of our proposed toolkit to explore the parameter sensitivity in terms of computational efficiency. Last, we report security analysis to prove the security of our toolkit without privacy leakage to unauthorized parties. Comprehensive experiments further demonstrate the efficiency and utility of our toolkit.

INDEX TERMS Privacy-Preserving, Secure Outsourcing Computation, K-means, Silhouette Coefficient

I. INTRODUCTION

CLOUD computing is a third party provider delivering on-demand services of data storage, computing resources and running functions without direct management by local users. It exceeds local servers in massive storage, unlimited computational resources and increasing availability. Therefore, individuals and companies with limited resources resort to the cloud services by outsourcing their computational tasks to the cloud. Meanwhile, sensitive information exists in countless domains, such as medical health monitoring [1], financial transaction records and location services [2]. Though powerful and flexible, the misgiving of privacy preservation on the cloud remains as a prior concern, which may be deteriorating the functionality and performance of the cloud services. For example, the personal information of over 77 million of accounts in Sony PlayStation Network outage were stolen due to an illegal and unauthorized intrusion, which damaged the reputation of the enterprise and caused huge economic loss. The breach of photographs of the celebrity from iCloud service in Apple company raised widespread condemnation from the public which forced the company to improve the data protection policy as well as technologies. As a consequence, the security issues exposed in such incidents have aroused an upsurge of research in both industry and academia.
A. MOTIVATIONS

In general, the privacy preservation approaches rely on the following assumptions: the cloud servers are honest-but-curious, and they do not act maliciously against the users and will follow the protocols as expected. However, the cloud server may not always be trusted. On the one hand, they might be vulnerable to the attackers who intend to gain access to the sensitive information from the outsourced data on the cloud. On the other hand, some intelligent cloud servers can be curious and sometimes even malicious who may deduce the information of the original data through the model and intermediate calculation results [4]. Therefore, how to design effective and suitable privacy-preserving protocols to achieve secure outsourcing computation on the cloud becomes an opening challenge.

In recent years, machine learning technology has been widely applied in natural language processing, computer vision, intrusion detection and etc. Consider a clustering task requested by a local user who uploaded the resource-demanding task to the cloud. The privacy concerns of this user are in threefold. First, the private training data has potential risk of leakage through the deduction from model parameters or predictions. Second, existing machine learning algorithms may involve complex operations such as division, power, exponential and logarithmic, which few secure outsourcing computation protocols are able to support. Previous frameworks [5], [6] can only support homomorphic addition, multiplication and comparison on ciphertext of floating point numbers (FPN) which cannot meet the requirement of clustering. Third, the secure outsourcing computation protocols on integer ciphertext are not capable of ensuring the convergence and precision of these machine learning algorithms. If running the entire task on the ciphertext of floating-point numbers is a must, it will cause huge computational overhead and time cost since the computation overhead on FPN is hundreds of times higher than that on integers. Furthermore, some of the previous outsourcing clustering algorithms [7]–[9] only implemented the clustering algorithm on ciphertext, and they did not consider how to evaluate the performance of clustering on ciphertext. In practice, the data owners are usually not aware of the number of classes that need to be clustered, which is necessary to be compared through multiple experimental evaluations.

Based on the above problems, we summarize the main contributions of this paper in the next section.

B. MAIN CONTRIBUTIONS

In this paper, we propose a multifunctional and privacy-preserving outsourcing computation toolkit and we have implemented a secure and evaluable homomorphic clustering algorithm. The main contributions of our paper are summarized as follows:

- A multifunctional and privacy-preserving outsourcing computation toolkit (MPOCT) is proposed in this paper which provides several protocols both on ciphertext of integers and floating point numbers (FPNs). Specifically, our protocols achieve homomorphic operations such as division, power, logarithm, etc.
- In MPOCT, mutual ciphertext conversion protocols between FPNs and integers are provided to balance calculation efficiency and precision. During outsourcing computations, when encountering high-precision calculation tasks, we can transform the integer ciphertext to FPN ciphertext for calculation. If tasks do not have high requirements for precision, we can use integer ciphertext for calculations to improve calculation efficiency and reduce computational overhead.
- Based on MPOCT, we implement the homomorphic algorithm of K-means. Furthermore, the homomorphic silhouette coefficient is designed on the ciphertext of clustering results, providing the informative clustering assessment to the local data owners.

The rest of this paper is organized as follows. We discuss the related works in Section II. Then we introduce the preliminaries of this paper in Section III. Our toolkit is described in Section IV and its application on clustering is introduced in Section V. We analyze the security of our toolkit in Section VI. The experimental evaluation is reported in Section VII. Section VIII concludes the paper and outlines the future work.

II. RELATED WORK

In this section, we briefly review the most relevant research on homomorphic encryption, secure outsourcing computation frameworks and previous secure outsourcing computation solutions for K-means.

Cloud computing drives the transition of local on-demanding tasks to the cloud servers for storage and calculation [10]. In recent years, privacy-preserving in cloud security has gradually become an increasing important concern [11]. The emergence of homomorphic encryption allows people to perform simple calculations on the ciphertext which is a powerful cryptographic primitive to secure outsourcing computations against an untrusted third-party provider. For example, Paillier cryptosystem [12] which is widely used in industry supports additive homomorphism, while Elgamal [13] supports multiplicative homomorphism. However, partially homomorphic cryptosystem only supports the homomorphic operation of addition or multiplication, so the application scenarios are very limited. Gentry [14] first proposed a fully homomorphic encryption (FHE) based on ideal lattice from a theoretical perspective in 2009. However, FHE cannot be applied in real world due to its large computational overhead. To address with the efficiency problem of FHE, many researchers have proposed more efficient FHE schemes like BGV [15], BFV [16], CKKS [17]. Based on the above-mentioned cryptographic schemes, many scholars have also proposed secure outsourcing computation frameworks. In 2016, Liu not only designed operations such as homomorphic multiplication and comparison through dual servers and Paillier cryptosystem, but also proposed secure outsourcing computation frameworks that support rational numbers [6].
and floating-point numbers [5]. Liu et al. [18] implemented a secure k-nearest neighbor based multi-label classification scheme based on Paillier cryptosystem, which both guarantees the security of the training data and classification model. Liu et al. [19] proposed a secure outsourcing computation framework based on FHE and key packaging technology, and this framework only required a single server to perform outsourced computing tasks. Li et al. [20] proposed a novel homomorphic encryption framework over non-abelian rings and defined the homomorphism operations in ciphertexts space which support real numbers encryption and privacy-preserving for machine learning training and classification in data ciphertexts environment. Zong et al. [21] investigated secure outsourcing computation of matrix determinant based on BGV and proposed an efficient matrix encoding technique which packs a matrix into a single ciphertext and can be easily applied as a submodule in the high-level applications. Nevertheless, above works more or less suffered from low efficiency, accuracy degradation, lack of scalability or ciphertext expansion issues.

As a common data mining algorithm, K-means [25] has abundant application scenarios in daily lives. Therefore, many scholars have studied how to perform clustering on cloud servers under the premise of protecting data privacy. Rao et al. [7] implemented the outsourcing computation of K-means based on Paillier in 2015. Then, Rong et al. [9] improved the computing efficiency by adding a trusted third party with limited computing capabilities. Almutairi et al. [8] also implemented the outsourcing computation of K-means based on homomorphic encryption. This solution has greatly improved the computing efficiency through a third party and limited interaction between user and server. However, neither of the above solutions provided evaluation indicators on clustering ciphertext. Therefore, the user can only get

![Secure and Evaluable Clustering](image)

**FIGURE 1.** A Call Graph of MPOCT. Blue: our implemented protocols. Grey: previous protocols in [5].

#### TABLE 1. Comparison between Different Solutions under Untrusted Server Scenario

| Characteristic                  | [7] | [9] | [8] | [22] | [23] | [24] | Ours |
|-------------------------------|-----|-----|-----|------|------|------|------|
| Without Trusted 3rd Party     | ✓   | ×   | ×   | ×    | ✓    | ✓    | ✓    |
| Only One-round Interaction    | ✓   | ✓   | ×   | ×    | ×    | ✓    | ✓    |
| Precision Not Reduced         | ✓   | ✓   | ✓   | ✓    | ✓    | ✓    | ✓    |
| Supporting Division           | ×   | ×   | ✓   | ✓    | ✓    | ✓    | ✓    |
| Supporting FPNs               | ×   | ×   | ×   | ✓    | ✓    | ✓    | ✓    |
| Supporting Evaluation         | ×   | ×   | ✓   | ✓    | ✓    | ✓    | ✓    |


This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
the clustering result, but cannot get the evaluation of them. Besides, the emergence of differential privacy [26] opened up new areas of secure outsourcing computation. Xia et al. [22] could efficiently complete outsourcing clustering tasks through local differential privacy technology. However, the choice of privacy budget had a great impact on the results, which means a better privacy may lead to a poorer precision. At the same time, the frequent interaction between user and server also puts a lot of pressure on the user side. Mohassel et al. [23] presented a scalable privacy-preserving clustering algorithm and design a modular approach for multi-party clustering. Although this scheme is computationally efficient, it relies on frequent collaboration and interaction between sender and receiver. Zou et al. [24] proposed a highly secure privacy-preserving outsourced k-means clustering under multiple keys in cloud computing, which applied both AES encryption and BCP encryption to provide privacy preservation against semi-honest adversaries. While the scheme guarantees strong security, it does not provide users with an evaluation metric for clustering. Table 1 summarizes the characteristics of different solutions to the aforementioned studies.

III. PRELIMINARIES

In this section, we will introduce the preliminaries of this paper. Our toolkit is mainly based on privacy-preserving outsourcing computation framework (POCF) [5], and we have an expansion and improvement on it. The following will introduce the overview of our system model, problem statement, the basic framework of POCF, and its protocols on integers and floating-point numbers.

A. SYSTEM MODEL

In our system, we mainly focus on how the cloud server can fulfill the computing tasks requested by the users while meeting the requirements of privacy protection. As shown in Figure 2, the simple system comprises a Key Generation Center(KGC), a Cloud Platform(CP), a Computation Service Provider(CSP) and Data Owners(DoS), while it can be extended to a distributed system with multiple pairs of CP and CSP. In this paper, we only use the simple model. The participants in this model are shown below.

- **KGC**: KGC is a trusted third party whose task is to distribute and manage the keys in the system.
- **DoS**: DoS own the data and they are the requesters of the computing tasks. DoS can encrypt their data by the public key(pk). Then they outsourced the encrypted data to CP for storage and requester CP to do some calculations on the outsourced encrypted data. In addition, DoS can use the private key(sk) to decrypt the encrypted result calculated by CP and CSP.
- **CP**: CP has ‘unlimited’ data storage space. CP can store and manage the outsourced data of all registered data owners. CP has the public key(pk) and the partially private key sk1. CP always stores the encrypted results of the computational tasks.
- **CSP**: CSP assists CP in completing computing tasks requested by the DoS. CSP owns the public key(pk) and the partially private key sk2. CSP can partially decrypt the ciphertext sent by CP and perform some operations on the decrypted results. Then CSP encrypts the calculated results and returns them to CP.

B. PROBLEM STATEMENT

Consider DoS have a lot of data and a clustering task, and their limited local resources make it impossible to complete the clustering task. The data therefore needs to be encrypted before being uploaded to the CP for storage. DoS can submit the input parameters required for the clustering task to CP. After CP and CSP cooperate to complete the calculation task, the clustering result and its evaluation are returned to DoS. In this process, we have the following challenges:

- Clustering algorithms and evaluation indicators involve division and comparison operations, but many homomorphic encryption schemes cannot support division homomorphism and comparison homomorphism. Therefore, more calculation protocols need to be constructed.
- Traditional K-means outsourcing computation schemes based on homomorphic encryption always normalize the original data and enlarge it. Then they encrypt the data after rounding. However, simply calculating the magnification of the plaintext is likely to exceed the limit of the plaintext space after a finite number of multiplications.
- Through previous experiments, we have learned that the calculation efficiency of floating-point number ciphertext is low and the communication overhead is high. However, in the process of clustering and evaluation, we may encounter situations where floating-point numbers must be used. If the whole process is based on floating-point ciphertext, it will take a lot of time to complete the clustering task.

C. ATTACK MODEL

In our attack model, DoS, CP and CSP are honest and curious. They will strictly implement the protocols, but they are also interested in data uploaded. We assume that CP and CSP are cooperative and not colluding. Meanwhile we assume that the communication between KGC and all parties are secure. Here, we introduce the adversary A∗ in our model. The purpose of adversary A∗ is to use the following capabilities to decrypt the ciphertext of challenger DoS:

1) Adversary A∗ may eavesdrop on all communication channels to obtain encrypted data transmitted.
2) Adversary A∗ may compromise CP, thereby guessing the plaintext corresponding to encrypted data outsourced from DO. In addition, it can perform interactive protocols with CSP to guess the plaintext corresponding to ciphertext sent from CSP.
3) Adversary $A^*$ may compromise CSP. Therefore, it can perform interactive protocols with CP to guess the plaintext corresponding to ciphertext sent from CP.

4) Adversary $A^*$ may compromise DOs to obtain their decryption capabilities, with the exception of the challenger DO. Then adversary $A^*$ may try to guess all the plaintext belonging to challenger DO through the decryption capabilities of other DOs.

However, Adversary $A^*$ cannot compromise CP and CSP at the same time. Moreover, adversary $A^*$ is restricted from compromising challenger DO. The above restrictions are common in adversary models in cryptographic protocols [27].

D. BASIC FRAMEWORK

The toolkit proposed in this paper is based on the Paillier Cryptosystem with Partial Decryption (PCPD) [5]. The basic functions and key splitting algorithm (KeyS) of Paillier have been described in PCPD, the description is omitted here. In this paper, we specify the following symbols: $[x]$ represents the ciphertext of integer $x$ and $\langle x \rangle$ represents the ciphertext of floating-point number (FPN) $x$, where $\langle x \rangle = ([s_x], [m_x], [t_x])$. $s_x$ represents the sign of $x$, $m_x$ represents $\eta$-significant digits, $t_x$ represents the exponent of base $\beta$. For the sake of convenience, we will use $\eta = 16$ and $\beta = 10$ for protocols on FPNs in this paper. Given $[x], [y]$ and public key $N$, we have the following properties:

$$[x + y] = [x] \cdot [y]; [\neg x] = [x]^{N-1}; [x - y] = [x] \cdot [y]^{N-1}.$$ 

E. PREVIOUS WORK ON INTEGERS & FLOATING-POINT NUMBERS

Before presenting our toolkit, we briefly review the previous study [5] of secure outsourcing computational protocols on integers and floating-point numbers respectively. Besides, the construction of floating point number ciphertext is also introduced. Note that our toolkit is an extension based on this previous work.

For simplicity, here we define '@' as 'performed by'. For example, '@CP' means 'performed by CP'.

1) Protocols on Integers

Protocols on integers enable the cloud platform CP to cooperate with the computation service provider CSP to implement multiplication, comparison, xor and other operations of integer plaintext on the ciphertext.

A. Revised Secure Multiplication Protocol (RSM):

It enables CP and CSP to cooperate to perform secure integer plaintext multiplication on ciphertext, i.e. [$f$] = RSM($[x], [y]$), where $f = x \cdot y$. Details are as follows:

- **Step-1 (@CP):** CP randomly selects two random numbers $r_x, r_y \in Z_N^*$ and compute:
  
  $X = [x + r_x] = [x] \cdot [r_x]$;
  
  $Y = [y + r_y] = [y] \cdot [r_y]$;
  
  $X_1 = PD_{sk_1}(X)$; $Y_1 = PD_{sk_1}(Y)$;
  
  After computation, CP will send $X, Y, X_1$ and $Y_1$ to CSP.
• **Step-2 (@CSP):** CSP partially decrypts $X_1$ and $Y_1$ transmitted from CP based on CSP’s owned partially private key $sk_2$, namely:
  
  \[ h_x = x + r_x = PD_{sk_2}(X, X_1); \]
  
  \[ h_y = y + r_y = PD_{sk_2}(Y, Y_1); \]
  
  \[ h = h_x \cdot h_y = (x + r_x) \cdot (y + r_y). \]

  Then, CSP encrypts $h$ based on public key $pk$ to get ciphertext $H = [h]$ and sends $H$ to CP.

• **Step-3 (@CP):** When CP receives $H$, CP first calculates:
  
  \[ [S_1] = [r_x \cdot r_y]^{N-1} = [-r_x \cdot r_y]; \]
  
  \[ [S_2] = [x]^{N-r_y} = [-r_y \cdot x]; \]
  
  \[ [S_3] = [y]^{N-r_x} = [-r_x \cdot y]; \]

  Last, CP can compute the result of $[x \cdot y]$:
  
  \[ [x \cdot y] = H \cdot [S_1] \cdot [S_2] \cdot [S_3] = [h - t \cdot r_y \cdot r_y - x - r_x \cdot y]. \]

**B. Secure Less Than Protocol (SLT):** It performs secure equivalent between two integer plaintext on ciphertext, i.e. $[f] = \text{SLT}([x], [y])$, if $x \geq y$, then $f = 0$; otherwise, $f = 1$. Details are as follows:

• **Step-1 (@CP):** CP computes:
  
  \[ [x_1] = ([x]^2) \cdot [1] = [2x + 1]; [y_1] = ([y]^2) = [2y]; \]

  Then, CP randomly chooses a coin $s \in 0, 1$ and generates a random number $r$, where $s \neq 0$ and $|r| \leq \frac{N}{2}$. If $s = 1$, then CP computes:
  
  \[ [l] = ([x_1] \cdot (y_1)^{N-1}) = [r \cdot (x_1 - y_1)]; \]

  If $s = 0$, then CP computes:
  
  \[ [l] = ([y_1] \cdot (x_1)^{N-1}) = [r \cdot (y_1 - x_1)]; \]

  After that, CP uses its partially private key $sk_1$ to compute $K = PD_{sk_1}([l])$, and then send it to CSP.

• **Step-2 (@CSP):** CSP decrypts $K$ with its partially private key $sk_2$ to get $l$. If $|l| \leq \frac{N}{2}$, CSP sets $u$ to 0; otherwise, set $u$ to 0. Next, CSP encrypts $u$ with the public key $pk$ to get $[u]$, and sends $[u]$ to CP.

• **Step-3 (@CP):** When CP receives $[u]$, CP will compute $[f]$ according to the value of $s$. If $s = 0$, CP will compute $[f] = [1] \cdot ([u]^{N-1}[1 - u]$; if $s = 1$, CP will refresh the value of $[u]$, i.e. $[f] = \text{CR}([u])$.

**C. Secure Exclusive OR Calculation Protocol (SXOR):** It performs secure exclusive or result of two integer plaintext on ciphertext, i.e. $[f] = \text{SXOR}([x], [y])$, where $x, y \in \{0, 1\}$. If $x = y$, $f = 0$; otherwise, $f = 1$. CP and CSP jointly calculate as follows:

\[ c_1 = \text{RSM}([1] \cdot [x]^{N-1}, [y]) = ([1 - xy]^{N-1}); \]

\[ c_2 = \text{RSM}([x], [1] \cdot [y]^{N-1}) = ([1 - y]^{N-1}); \]

\[ [f] = [x \cdot y] = [f_1] \cdot [f_2] = [x + y - 2xy]; \]

**D. Secure Equivalent Testing Protocol (SEQ):** It performs secure equivalent between two integer plaintext on ciphertext, i.e. $[f] = \text{SEQ}([x], [y])$. If $x = y$, then $f = 0$; otherwise, $f = 1$. CP and CSP jointly calculate as follows:

\[ [u_1] = \text{SLT}([x], [y]); [u_2] = \text{SLT}([y], [x]); \]

\[ [f] = [u_1 \oplus u_2] = \text{SXOR}([u_1], [u_2]). \]

**E. Secure Exponent Calculation Protocol (SEXP):** It performs secure integer plaintext exponent on ciphertext, i.e. $[f] = \text{SEXP}([x], [y])$ where $x$ is a public integer, $x > 0$, $y \geq 0$ and $f = x^y$. Details are as follows:

• **Step-1 (@CP):** CP selects a random number $r \in Z_N^*$ and computes:
  
  \[ [y_1] = [y] \cdot [r] = [y + r]; S = (x^r)^{-1} \text{ mod } N; \]

  Then, CP uses its own partially private key $sk_1$ to compute $Y = PD_{sk_1}([y_1])$, and sends $[y_1]$ and $Y$ to CSP.

• **Step-2 (@CSP):** CSP decrypts $Y$ using its partially private key $sk_2$ to obtain $y_1$. Then, CSP computes $h = x^{h_1} \text{ mod } N$ and encrypts it into $[h]$ using the public key $pk$, and sends $[h]$ to CP. Obviously, there exists $h = x^{h_1+r} \text{ mod } N$.

• **Step-3 (@CP):** When CP receives $[h]$, it will compute
  
  \[ [f] = [h]^{S} = [x^{h_1+r} \cdot (x^r)^{-1}] = [x^y]; \]

**F. Secure Inverse Calculation Protocol (SINV):** It performs secure integer plaintext inverse on ciphertext, i.e. $[f] = \text{SINV}([x])$ where $x \neq 0$ and $f = x^{-1} \text{ mod } N$. Details are as follows:

• **Step-1 (@CP):** CP selects a random number $r \in Z_N^*$ and computes $[y_1] = [y] \cdot [r] = [y + r]$. Then, CP computes $Y = PD_{sk_1}([y_1])$ using its partially private key $sk_1$ and sends $[y_1]$ and $Y$ to CSP.

• **Step-2 (@CSP):** CSP decrypts $Y$ using its partially private key $sk_2$ to obtain $y_1$. Then, CSP computes $h = y_1 \text{ mod } x$ and encrypts it as $[h]$ using the public key $pk$. Then it sends $[h]$ to CP. Obviously, there exists $h = (y + r) \text{ mod } x$.

• **Step-3 (@CP):** When CP receives $[h]$, it will compute $[y \text{ mod } x]$ by the following process:
  
  \[ [U] = [h] \cdot [r] \text{ mod } x^{N-1} = ([y + r] \text{ mod } x) \cdot [r \text{ mod } x]^{N-1} = ([y + r] \text{ mod } x) \cdot (r \text{ mod } x); \]

**Step-4 (@the collaboration of CP and CSP):** Through the above formula, we can get $-x \leq U \leq x$. But $((y + r) \text{ mod } x) - (r \text{ mod } x)$ and $y \text{ mod } x$ are not always equal. Therefore, the following two cases are discussed. If $-x \leq U < 0$, then $f = U + x$. If $0 \leq U < x$, then $f = U$. So it will only need to compare the size of $U$ and $0$:

\[ [k_0] = \text{SLT}([U], [0]); [f] = [U] \cdot [k_0]^2 = [U + x \cdot k_0] = [((y+r) \text{ mod } x) - (r \text{ mod } x)] + x \cdot [y \text{ mod } x]. \]

**G. Secure Modular Calculation Protocol (SMOD):** It performs secure integer plaintext modular on ciphertext, i.e. $[f] = \text{SMOD}([x], [y])$ where $y$ is a public integer and $y \ll N$ and $f = x \text{ mod } y$. Details are as follows:
• **Step-1 (@CP):** CP selects a random number \( r \in \mathbb{Z}_N^* \) and computes \( [x_1] = [x]^r = [r \cdot x] \). Then, CP computes \( X = PD_{sk_1}([x_1]) \) using its partially private key \( sk_1 \) and sends \([x_1]\) and \( X\) to CSP.

• **Step-2 (@CSP):** CSP decrypts \( X \) using its partially private key \( sk_2 \) to obtain \( x_1 \). Then, CSP computes \( h = (x_1)^{-1} \mod N \) and encrypts it as \([h]\) using the public key \( pk \). Then it sends \([h]\) to CP. Obviously, there exists \( h = (r \cdot x)^{-1} \mod N \).

• **Step-3 (@CP):** When CP receives \([h]\), it will compute \([x^{-1}] \mod N\) by the following process:

\[
[f] = [h]^r = [(r \cdot x)^{-1}] \mod N = [(r \cdot x)^{-1} \cdot r \mod N] = [x^{-1} \mod N];
\]

2) Construction of Floating-Point Number Ciphertext (FPN)

In computer, floating-point number is expressed as follows. Given base \( \beta \) and extreme value index \( e_{min},e_{max} \), there exists at least one triple \( (s,m,t) \), so that \( x = (-1)^s \cdot m \cdot \beta^t \).

In order to facilitate the storage of floating-point ciphertext and the construction of secure outsourcing computation protocols, it is stipulated that all floating-point numbers in the system should use the same significant digit \( \eta \), base \( \beta \), sign bit \( s \in \{0,1\} \), the significant digit \( m \) is an rbit integer, and the exponential bit \( t \) satisfies \( e_{min} - \eta + 1 \leq t \leq e_{max} - \eta + 1 \).

For example, when \( \eta = 5, \beta = 10 \) are fixed in the system, \( -31 \) can be expressed as \((-1)^1 \cdot 31000 \cdot 10^{-3}, i.e. \) begin stored as \((1,31000,\text{--3})\). \( 1.024 \) can be expressed as \((-1)^0 \cdot 10240 \cdot 10^{-4}, i.e. \) begin stored as \((0,10240,\text{--4})\). Through the above example, it can be found that this triple expression is uniquely determined when a finite floating-point number is given. When the integer triplet \( (s_x, m_x, t_x) \) representing the floating point number \( x \) is uploaded to the cloud platform, it is necessary to separately encrypt \( s_x, m_x, t_x \). On the cloud platform, the ciphertext storage form of floating point number \( x \) is \([(s_x), (m_x), (t_x)]\).

3) Protocols on Floating-Point Numbers

Based on the above two parts(i.e. protocols on integers and construction of FPN ciphertext), we now introduce the protocols on floating-point numbers \([5]\) which drives the collaboration of CP and CSP to implement addition, multiplication, comparison and other operations of floating point number plaintext on ciphertext.

• **Secure Floating Point Numbers Equivalent Protocol (FPEQ):** It performs secure equivalent between two floating point number plaintext on ciphertext, i.e. \([f] = FPEQ([x],[y])\).

• **Secure Floating Point Numbers Comparison (SFPC):** It performs secure equivalent between two floating point number plaintext on ciphertext, i.e. \([f] = SFPC([x],[y])\).

\[f \geq y \text{ if } f = 1; \text{ otherwise } f = 0.\]

• **Secure Floating Point Numbers Comparison (SFPC):** It performs secure equivalent between two floating point number plaintext on ciphertext, i.e. \([f] = SFPC([x],[y])\).

\[f \geq y \text{ if } f = 1; \text{ otherwise } f = 0.\]

• **Secure Floating Point Numbers Additions Protocol (SFPA):** It performs secure addition of two floating point number plaintext on ciphertext, i.e. \([f] = SFPA([x],[y])\).

\[f + y \text{ where } f = [x + y].\]

• **Secure Floating Point Numbers Multiplications Protocol (SFPM):** It performs secure multiplication of two floating point number plaintext on ciphertext, i.e. \([f] = SFPM([x],[y])\).

\[f \cdot y \text{ where } f = [x \cdot y].\]

IV. OUR TOOLKIT

In this section, we will introduce the protocols in our toolkit concretely. To make the relationship and independence of the protocols clearer, a call graph is presented in Fig[1]. Here, our implemented protocols are colored in blue, while the previous protocols in \([5]\) are colored in grey. Similar to the secure outsourcing computation protocols proposed in POCF \([5]\), our extended protocols can handle outliers(NaN) in the same way. For the sake of simplicity, we only describe the process that excludes these outliers. Below we will describe the details of these secure computation protocols.

A. EXTENDED PROTOCOLS ON INTEGERS

1) Secure Integer Division Protocol (SDIV)

Given two encrypted numbers \([x],[y]\), where \(0 \leq |x| < \beta_0^0\), \(0 < |y| < \beta_0^0\), the goal of SDIV protocol is to calculate the result \((f)\), s.t. \(f = [x \cdot y]^{-1}\). The idea of SDIV is as follows: we suppose \(f = [x \cdot y]^{-1}\). Given the above range of \(x\) and \(y\), we can know that \(0 \leq |x| \cdot |y|^{-1} < \beta_0^0\), i.e. \(0 \leq f < \beta_0^0\). Therefore, there exists \(f' \in [0,\beta_0^0)\) s.t. \(f' \cdot |y| \leq |x|\). Then \(f\) is the maximal \(f'\) that satisfies the above inequality.

Based on the above idea, the overall steps of SDIV protocol are shown as follows:

**Step-1:** First, CP and CSP jointly determine whether \(x, y \geq 0\),

\[L_x = \text{SLT}([\frac{x}{y}], [x]; \lfloor 1 - L_x \rfloor = \lceil 1 \cdot L_x \rceil^{-1}; \quad L_y = \text{SLT}([\frac{y}{x}], [y]; \lfloor 1 - L_y \rfloor = \lceil 1 \cdot L_y \rceil^{-1}; \quad A_x = \text{RSM}([L_x], [x]) \cdot \text{RSM}([1 - L_x], [x]^{-1}); \quad A_y = \text{RSM}([L_y], [y]) \cdot \text{RSM}([1 - L_y], [y]^{-1});

**Step-2:** Initialize \(V = [0]\).

\[\{\text{for } i = 0 \text{ to } \eta_0 - 1{\}; \quad [u_{i+1}] = [0]; \quad [U] = [0]; \quad \text{for } j = 0 \text{ to } \beta_0 - 1{\}; \quad [B_0] = [(\beta_0 - j) \cdot \beta_0^{\frac{1}{\eta_0 - 1 - i}}]; \quad [V] = [V \cdot [B_1]; \quad [M_1] = \text{RSM}([A_x], [V]); \quad [L_1] = \text{SLT}([M_1], [A_x]); \quad [u_2] = \text{SXOR}([u_{i+1}], [L_1]); \quad [U] = [U \cdot [M_2]; \quad [M_2] = \text{RSM}([B_1], [u_2]); \quad}\}

\[\text{Let } [V] = [V \cdot [U] .\]
Step-3: Let $[S_1] = \text{SXOR}([L_x],[L_y])$; $[S_2] = [1-S_1] = [1] \cdot [S_1]^{N-1}$; $[V] = [V]^{N-1}$; $[f] = \text{RSM}([S_1],[V]) \cdot \text{RSM}([S_2],[V])$.

Through the above steps, CP and CSP can jointly compute $[[x \cdot y^{-1}]] = \text{SDIV}([x],[y])$.

2) Secure Integer Ciphertext Divided by Integer Plaintext Protocol(SCDP)

Given one encrypted number $[x]$ and a public integer $y(y \neq 0)$, the goal of SCDP protocol is to calculate the result $[f]$, s.t. $f = [x \cdot y^{-1}]$. The idea of SCDP is as follows: based on the definition of $[x \cdot y^{-1}]$, we have:

$$[x \cdot y^{-1}] = \begin{cases} \frac{x-(x \mod y)}{y} & y > 0 \\ \frac{-x-(x \mod |y|)}{|y|} & y < 0 \end{cases} \quad (1)$$

And, $x - (x \mod y)$ can be divided by $y$, so that we can compute ciphertext of $y^{-1}$ through Secure Inverse Calculation Protocol(SINV), and then compute the result of $[f]$ through Revised Secure Multiplication Protocol (RSM).

The overall steps of SCDP protocol is shown as follows: $[M_1] = \text{SMOD}([y],[x])$; $[M_2] = [x] \cdot [M_1]^{N-1}$; $[I] = \text{SINV}([y])$; $[f] = \text{RSM}([M_2],[I])$; if $y < 0$ then $[f] = [f]^{N-1}$.

Then, we have $[[x \cdot y^{-1}]] = \text{SCDP}([x],[y])$.

3) Secure Integer Ciphertext Multiply Floating Point Number(SIMF)

Given one encrypted number $[x]$ and a floating point number $y$, the goal of SIMF protocol is to calculate the result $[f]$, s.t. $f = [x \cdot y]$. In addition, we need a precision parameter $n_1$, where $n_1$ is an integer. The overall process of SIMF protocol is shown as follows:

$[Y] = [[y \cdot 10^{n_1}]]$; $[R] = \text{RSM}([x],[Y])$; Let $[f] = \text{SCDP}([R],10^{n_1})$.

Then, we have $[[x \cdot y]] = \text{SIMF}([x],[y])$.

4) Secure Integer Power Protocol(SPOW)

Given one encrypted number $[x]$ and one public integer $y$, where $y \geq 0$. The goal of SPOW protocol is to calculate the result $[f]$, s.t. $f = [x^y]$.

The idea of SPOW is as follows: since $y$ is a positive integer, we can calculate the power within $O(\log_2 y)$ time complexity through Exponentiation by Squaring algorithm. Compared with cumulative multiplication, it can largely reduce the number of calls to RSM, so as to improve the computation efficiency and reduce the communication overhead.

The overall process of SPOW protocol is shown as follows:

Step-1:(@CP) Let $B = (b_n \cdots b_1)$ is the binary string of $y$, where the bit length of $y$ is $n_y = \log_2 y + 1$.

Step-2: Then, initialize $[f] = [0]$; $[X] = [x]$.

for $i = 1$ to $n_y$

if $b_i = 1$ then $[f] = \text{RSM}([f],[X])$;

if $i < n_y$ then $[X] = \text{RSM}([X],[X])$;

Through the above steps, CP and CSP can jointly compute $[x^y] = \text{SPOW}([x],[y])$.

5) Secure Integer Logarithm Protocol(SLOG)

Given one encrypted number $[x]$ and a public integer $y$, where $y \geq 2$ and $0 < x < y^6$. The goal of SLOG protocol is to calculate the result $[f]$, s.t. $f = \log_y x$. The overall process of SLOG protocol is shown as follows:

for $i = 1$ to $\delta$

$[u_2] = \text{SLT}([x],[y])$; $[v] = \text{SXOR}([u_1],[u_2])$;

$[u_1] = [u_2]$; $[f] = [f] \cdot \text{RSM}([v],[i])$;

}. Let $[f] = [f] \cdot [-1]$.

Then, we have $[[\log_y x]] = \text{SLOG}([x],[y])$.

6) Secure Integer Array Maximum(SIAM)

Given an integer ciphertext array $C = \{[x_0],[x_1],\cdots,[x_{n-1}]\}$, where $n \geq 2$. We calculate: $[x_{\max}] = [x_0]$; $[i_{\max}] = [0]$;

for $i = 1$ to $n - 1$

$[r_1] = \text{SLT}([x_{\max}],[x_i])$; $[r_2] = \text{SEQ}([r_1],[1])$;

$[1 - r_2] = [1] \cdot [r_2]^{N-1}$;

$[i_{\max}] = \text{RSM}([i_{\max}],[r_2]) \cdot \text{RSM}([i],[1 - r_2])$;

$x_{\max} = \text{RSM}([x_{\max}],[r_2]) \cdot \text{RSM}([x_i],[1 - r_2])$.

Then, we have $[i_{\max}],[x_{\max}] = \text{SIAX}(C)$, where $i_{\max} = \arg \max \{x_0,\cdots,x_{n-1}\}$, $x_{\max} = \max \{x_0,\cdots,x_{n-1}\}$.

B. EXTENDED PROTOCOLS ON FLOATING-POINT NUMBERS

1) Secure Floating Point Number Opposite(SFPO)

Given one encrypted FPN $\langle x \rangle = \{[s_x],[m_x],[t_x]\}$, the goal of SFPO protocol is to calculate the result $\langle f \rangle$, s.t. $f = -x$. The overall process of SFPO protocol can be executed only by CP:

$\langle s_f \rangle = \{[s_x]^{N-1}\}$; $[m_f] = [m_x]$; $[t_f] = [t_x]$; Let $\langle f \rangle = \{[s_f],[m_f],[t_f]\}$.

Then, we have $\langle -x \rangle = \text{SFPO}(\langle x \rangle)$.

2) Secure Floating Point Number Subtraction(SFPU)

Given two encrypted FPNs $\langle x \rangle$ and $\langle y \rangle$, the goal of SFPU protocol is to calculate the result $\langle f \rangle$, s.t. $f = x - y$. Since subtraction is the inverse of addition, the overall process of SFPU protocol is shown as follows:

- **Step-1(@CP)**: $\langle -y \rangle = \text{SFPO}(\langle y \rangle)$;

- **Step-2**: $\langle x - y \rangle = \text{SFPA}(\langle x \rangle, \langle -y \rangle)$.

Then, we have $\langle x - y \rangle = \text{SFPU}(\langle x \rangle, \langle y \rangle)$.

3) Secure Floating Point Number Reciprocal(SFPR)

Given one encrypted FPN $\langle x \rangle = \{[s_x],[m_x],[t_x]\}$, the goal of SFPR protocol is to calculate the result $\langle f \rangle$, s.t. $f = x^{-1}$. Assuming that $f = x^{-1}(x \neq 0)$, $f$ can also be converted to a
triple \{s_f, m_f, t_f\}. According to the property of reciprocal, we can easily have \(s_f = s_x\). Then the main problem is how to determine \(m_f\) and \(t_f\). Notice that \(\beta^{n-1} \leq m_x < \beta^n\), \(m_x = m_f \leq \beta^{2n-1}\). Then \(m_f\) is the maximal \(m\) that satisfies the above inequality.

Next, we will discuss this problem in two cases according to the value of \(m_f\).

**Case I:** if \(m_f = \beta^n\), then the number of significant digits is \(\eta + 1\) bit. Since \(\beta^{n-1} \leq m_f < \beta^n\), we need to decrease \(m_f\) by one bit, i.e. \(m_f = \beta^{n-1}\).

**Case II:** if \(m_f \in [\beta^{n-1}, \beta^n)\), then it is consistent with \(m_f\) in terms of value range.

Then we confirm the value of \(t_f\), we have:

\[
(m_f \cdot \beta^{t_f}) \cdot (m_x \cdot \beta^{t_x}) = (m_f \cdot m_x) \cdot (\beta^{t_f+t_x}) = (m_f \cdot m_x) \cdot \beta^{t_f+t_x+1} \approx 1
\]

Consider \(m_f \cdot m_x \approx \beta^{2n-1}\), we have \(t_f + t_x = 1 - 2\eta\), i.e. \(t_f = 1 - 2\eta - t_x\).

Based on the above analysis, the overall steps of SFPR protocol are shown as follows:

**Step-1:** CP and CSP first jointly handle the special cases. We determine whether \(x = 0\) to \(0\), if \(x\) is equal to \(0\), then the result is \(0\). This can be achieved by the following calculations:

\[
[Z] = \text{FPEQ}(\langle x \rangle, \langle 0 \rangle); \quad [1 - Z] = [1] \cdot [Z]^{N-1}
\]

**Step-2:** Then CP and CSP can jointly evaluate the significance of \(f\). Initialize \([v] = [0], [E_1] = [\beta^{2n-1}]\).

**Step-3:** Finally, CP and CSP can jointly evaluate the exponents of \(f\) by combining the two cases in the analysis. Then we calculate:

\[
[v^*] = \text{SMOD}(\beta^n, [v]); \quad [r] = \text{SEQ}(\langle v^* \rangle, \langle v \rangle); \\
[1 - r] = [1] \cdot [r]^{N-1}; [B_0] = [\beta^{3n-1}]; \\
[m_f] = \text{RSM}(\langle r \rangle, [B_0]); [m_x^*] = \text{RSM}(\langle m_f \rangle, [Z]); \\
[s_f] = [s_x]; [T_1] = \text{RSM}(\langle r \rangle, [Z]); [t_1] = [t_x]^{N-1}; [T_1]; \\
t_2 = \text{RSM}(\langle \eta_{max} \rangle, [1 - Z]); [t_f] = [t_1] \cdot [t_2]; \\
t_f^* = [t_f] \cdot [2n - 1]^{N-1} \cdot (f) = \{s_f, m_f^*, t_f^*\}
\]

Then, CP and CSP can jointly compute \((x^{-1}) = \text{SFPR}(\langle x \rangle)\).

4) Secure Floating Point Number Division(SFPD)

Given two encrypted FPNs \((x), \langle y \rangle\), the goal of SFPD protocol is to calculate the result \(\langle f \rangle\), s.t. \(f = x \cdot y^{-1}\). The overall process of SFPD protocol are shown as follows:

\[
(y^{-1}) = \text{SFPR}(\langle y \rangle); \quad \langle f \rangle = \text{SFPM}(\langle x \rangle, \langle y^{-1} \rangle)
\]

Then, we have \((x \cdot y^{-1}) = \text{SFPD}(\langle x \rangle, \langle y \rangle)\).

**C. CIPHERTEXT MUTUAL CONVERSION PROTOCOLS**

1) Secure Integer to Floating Point Number(SITF)

Given one encrypted number \([x]\) where \(\mid x \mid < 2^{10n}\), the goal of SITF protocol is to calculate the result \(\langle x \rangle\). First, we need to determine the sign of \(x\), which is easy to achieve. Then the question can be discussed in two cases:

**Case I:** If \(|x| \geq \beta^n\), then \(t_x > 0\). We should keep the first \(\eta\) significant digits of \(x\) as \(m_x\).

**Case II:** If \(|x| < \beta^n\), then \(t_x \leq 0\). We should expand \(x\) to \(\eta\) digits.

Based on the above idea, the overall steps of SITF protocol are shown as follows:

**Step-1:** First, CP and CSP jointly determine whether \(x\) is equal to \(0\), this can be achieved by the following calculations:

\[
[Z] = \text{SEQ}(\langle x \rangle, \langle 0 \rangle); \quad [1 - Z] = [1] \cdot [Z]^{N-1}
\]

**Step-2:** Then, CP and CSP jointly determine whether \(|x|\) is greater than \(0\).

\[
[S_1] = [\lfloor \frac{N}{2} \rfloor]; \quad [s] = \text{SLT}(\langle S_1 \rangle, \langle x \rangle); \\
[1 - s] = [1] \cdot [s]^{N-1}; [1 - x] = [x]^{N-1}; \\
|x| = \text{RSM}(\langle s \rangle, \langle x \rangle) \cdot \text{RSM}(\langle 1 - s \rangle, \langle -x \rangle)
\]

**Step-3:** CP and CSP jointly determine whether \(|x|\) is greater than or equal to \(\beta^n\).

\[
[B_0] = \text{SLT}(\langle [x], \beta^n \rangle); \quad [1 - B_0] = [1] \cdot [B_0]^{N-1}
\]

**Step-4:** Then, CP and CSP jointly solve the first case that \(|x| \geq \beta^n\).

\[
[u_1] = [0]; [T_1] = [0]; [m_1] = [0]; [A_1] = [\langle x \rangle];
\]

for \((i = 0\) to \(\eta_{max} - 1\)) { 

\[
[M_1] = \text{SMOD}(\beta^{n-i-1}, \langle [x] \rangle); \\
[E_1] = [\eta_{max} - i - 1]; \\
[u_1^*] = \text{SEQ}(\langle M_1 \rangle, \langle [x] \rangle); [t_1] = \text{SXOR}(\langle u_1^* \rangle, \langle u_1 \rangle); \\
[T_1] = [T_1] \cdot \text{RSM}(\langle t_1 \rangle, \langle E_1 \rangle); \\
[v_1] = [u_1^*]; [v_1^*] = \text{RSM}(\langle v_1 \rangle, \langle I_2 \rangle); \\
[V_1] = \text{RSM}(\langle v_1 \rangle, \langle v_1^* \rangle); [m_1] = [m_1]; [V_1];
\]

\}

**Step-5:** After that, CP and CSP jointly solve the second case that \(0 < |x| < \beta^n\).

\[
[u_2] = [0]; [T_4] = [0]; [m_2] = [0]; [A_2] = [\langle x \rangle];
\]

for \((i = 0\) to \(\eta - 1\)) { 

\[
[E_2] = [\eta - i - 1]; [M_2] = \text{SMOD}(\beta^{n-i-1}, \langle [x] \rangle); \\
[u_2^*] = \text{SEQ}(\langle M_2 \rangle, \langle [x] \rangle); [t_2] = \text{SXOR}(\langle u_2^* \rangle, \langle u_2 \rangle); \\
[T_5] = [T_5] \cdot \text{RSM}(\langle t_2 \rangle, \langle E_2 \rangle); [v_2] = [u_2^*]; \\
[v_2^*] = [A_2]; [M_2]^{N-1}; [A_2] = [A_2] \cdot [v_2^*]; \\
[v_2^*] = [v_2];
\]

\}

\}

Let \([T_6] = [\eta - i - 1]; [T_6] = [T_6]^{N-1}\) and \([B_3] = \text{SEXP}(\beta, [T_6])\).

for \((i = 0\) to \(\eta - 1\)) { 

\[
[E_2] = [\eta - i - 1]; [R_2] = \text{SLT}(\langle T_5 \rangle, \langle E_2 \rangle);
\]
First, we determine whether \( Z_m = 0 \), this can be achieved by the following calculations:

\[
[x] = \text{RSM}([1 \cdot Z_m], [0]); \quad [t] = \text{RSM}([1 \cdot B_0], [t_0]); \quad [m] = \text{RSM}([1 \cdot m_0], [0]);
\]

\{ T_2 \} = \{ T_2 \}_{0}^{N-1}.

Step-6: Then CP and CSP jointly calculate:

\[
[m] = \text{RSM}([1 \cdot B_0], [m_0]) \cdot \text{RSM}([B_0], [m_2]);
\]

\[ [t] = \text{RSM}([1 \cdot B_0], [t_0]) \cdot \text{RSM}([B_0], [t_2]); \]

Step-7: Finally, CP and CSP jointly calculate the final result:

\[
0 = \{ [s_0], [m_0], [t_0] \};
\]

\[
[s_i] = \text{RSM}([1 - Z], [s_0]) \cdot \text{RSM}([Z, [1 - s_i]]);
\]

\[
[t_i] = \text{RSM}([1 - Z], [t_0]) \cdot \text{RSM}([Z, [t_i]]);
\]

\[
[m_i^t] = \text{RSM}([1 - Z], [m_0]) \cdot \text{RSM}([Z, [m_i]]);
\]

Let \( \langle f \rangle = \{ [s_i^t], [m_i^t], [t_i^t] \} \). Then, CP and CSP can jointly compute \( \langle x \rangle = \text{SIFT}(\langle x \rangle) \).

V. SECURE AND EVALUATIVE CLUSTERING

In this section, we first introduce Square Euclidean Distance, which will be used as a measure of the distance between samples. Next, we give the implementation of the homomorphic algorithm of K-means. To evaluate the performance of the clustering results on ciphertext, we design and implement the homomorphic silhouette coefficient.

A. SECURE SQUARE EUCLIDEAN DISTANCE (SED)

Given two arrays of the same length, \( x = \{ x_0, \cdots, x_{n-1} \} \) and \( y = \{ y_0, \cdots, y_{n-1} \} \), then the square euclidean distance \( \text{SED}(x, y) = \sum_{i=0}^{n-1} (x_i - y_i)^2 \). Therefore, given two integer ciphertext arrays of the same length, \( x = \{ X_0, \cdots, X_{n-1} \} \) and \( y = \{ Y_0, \cdots, Y_{n-1} \} \), the square euclidean distance on integer ciphertext is defined in Algorithm 1.

Algorithm 1 Secure Square Euclidean Distance (SSED)

Input: \( x = \{ X_0, \cdots, X_{n-1} \}, y = \{ Y_0, \cdots, Y_{n-1} \} \)

Output: the integer ciphertext \( [D] \) of square euclidean distance between \( x \) and \( y \)

1. Initialize \( [D] = [0] \);
2. for \( i = 0 \) to \( n - 1 \) do
3. \( [d_i] = [x_i] \cdot [y_i]^{-N-1}; [s_i] = \text{RSM}([d_i], [d_i]); \)
4. \( [D] = [D] \cdot [s_i]; \)
5. end for
6. return \([D]\);

B. SECURE K-MEANS CLUSTERING

K-means is a commonly-used cluster algorithm. Given a set of samples (i.e. the observations) \( X = \{ x_0, \cdots, x_{m-1} \} \) where each feature \( x_i \) has \( n \) features, i.e. \( x_i = \{ x_{i0}, \cdots, x_{in-1} \} \). The algorithm partitions the samples into \( K \) disjoint subsets \( S = \{ S_1, S_2, \cdots, S_K \} \) so as to minimize the within-cluster sum of squares. Here, we will introduce K-means algorithm module by module. Then, we will introduce the implementation of secure K-means clustering based on our toolkit MPoCT.

1) Secure Sample Partition

For each sample \( x_i (0 \leq i < m) \) in the dataset, the K-means algorithm calculates the distance \( D_i,k \) from each sample \( x_i \) to each cluster center \( C_k \), and assigns the sample \( x_i \) to the cluster to which the nearest cluster center belongs. Suppose \( C_i \) is the index of the cluster to which sample \( x_i \) belongs.
The implementation of the above partition of samples on ciphertext is shown in Algorithm 2.

Algorithm 2 Secure Sample Partition(SSP)

**Input:** i) encrypted samples $X = \{x_0, \ldots, x_{m-1}\}$;
i) encrypted centroids of cluster $C = \{C_0, \ldots, C_{K-1}\}$;

**Output:** i) encrypted cluster index of samples $c = \{(c_0), \ldots, (c_{K-1})\}$

1. for $j = 0$ to $m - 1$ do
2. for $k = 0$ to $K - 1$ do
3. Calculate $[d_{j,k}] = \text{SSED}(x_j, C_k)$;
4. end for
5. end for
6. Let $d_j = \{[d_{j,0}], \ldots, [d_{j,K-1}]\}$;
7. Let $c_j = \min \{[d_{j,0}, d_{j,1}, \ldots, d_{j,K-1}]\} = \text{SIAN}(d_j)$;
8. end for
9. return $c$

2) Secure Cluster Updating

For each partitioned cluster set $S_k$, K-means algorithm will recalculate its cluster center $C_k = (C^0_k, C^1_k, \ldots, C^{n-1}_k)$ where the $i$-th center of the $j$-th feature are computed as follows:

$$C^j_k = \frac{1}{|S_k|} \sum_{x_i \in S_k} x^j_i, 0 \leq j < n$$  \hspace{1cm} (3)

The implementation of the above cluster updating approach on ciphertext is shown in Algorithm 3.

Algorithm 3 Secure Cluster Updating(SCU)

**Input:** i) encrypted samples $X = \{x_0, \ldots, x_{m-1}\}$;
i) encrypted cluster index of samples $c = \{(c_0), \ldots, (c_{K-1})\}$;

**Output:** i) encrypted cluster centroids $C = \{C_0, \ldots, C_{K-1}\}$

1. for $k = 0$ to $K - 1$ do
2. Let $[n_k] = [0]$;
3. for $j = 0$ to $m - 1$ do
4. $[E_1] = \text{SEQ}([c_j], [k]); \quad [1 - E_1] = [1] \cdot [E_1]^{N-1}$;
5. $C^k_n = [0](0 \leq l < n)$;
6. for $l = 0$ to $n - 1$ do
7. $[t^l_j] = \text{RSM}([1 - E_1], [x^j_l]); \quad [C^k_l] = [C^k_l] \cdot [t^l_j]$;
8. end for
9. $[n_k] = [n_k] \cdot [1 - E_1]$;
10. end for
11. for $l = 0$ to $n - 1$ do
12. $[C^k_l] = \text{SDIV}([C^k_n], [n_k])$;
13. end for
14. Let $C_k = \{(C^0_k), (C^1_k), \ldots, (C^{n-1}_k)\}$
15. end for
16. Let $C = \{C_0, C_1, \ldots, C_{K-1}\}$
17. return $C$

3) Secure K-means

Based on the above two modules, for a given number of iterations $E$, the implementation of K-means clustering algorithm on ciphertext is shown in Algorithm 4.

Algorithm 4 Secure K-means(SKM)

**Input:** i) encrypted samples $X = \{x_0, \ldots, x_{m-1}\}$;
i) encrypted cluster centroids $C = \{C_0, \ldots, C_{K-1}\}$;
i) encrypted cluster index of samples $c = \{(c_0), \ldots, (c_{K-1})\}$

**Output:** i) encrypted cluster centroids $C = \{C_0, \ldots, C_{K-1}\}$;

1. Random choice $k(k \leq m)$ samples $x_{r_1}, \ldots, x_{r_k}$ as cluster centroids $C = \{C_0, \ldots, C_{K-1}\}$.
2. for $i = 0$ to $E - 1$ do
3. Partition the samples to $K$ clusters: $c = \text{SSP}(X, C)$;
4. Updating cluster centroids: $C = \text{SCU}(X, c)$;
5. end for
6. return $C, c$

C. EVALUABLE CLUSTERING ON CIPHERTEXT

Since the previous schemes did not provide an evaluation index on the ciphertext of the clustering result, DOs can only obtain the clustering results returned by the server. However, it is impossible to evaluate the clustering results due to DOs’ limited local resources. Here, we choose silhouette coefficient [28] as the evaluation index of clustering. Here we describe the silhouette coefficient module by module and give the corresponding implementation on ciphertext based on our toolkit MPOCT.

1) Secure Distance Calculation between Samples

First, the silhouette coefficient needs to calculate the distance between two samples $d_{i,j}$. This is easy to implement on ciphertext, which is shown in algorithm 5.

Algorithm 5 Secure Distance Calculation between Samples(SDCS)

**Input:** i) sample number $m$;
i) cluster number $K$;
i) encrypted sample set $X = \{x_0, \ldots, x_{m-1}\}$

**Output:** i) the encrypted distance between samples $d = \{d_{i,j}\}$

1. Let $d_{i,j}$ is the distance between $x_i$ and $x_j$, then $[d_{i,j}] = [0], (0 \leq i \leq m - 1)$.
2. for $i = 0$ to $m - 1$ do
3. for $j = i + 1$ to $m - 1$ do
4. Calculate $[d_{i,j}] = \text{SSED}(x_i, x_j)$;
5. Let $[d_{j,i}] = [d_{i,j}]$
6. end for
7. end for
8. Let $d = \{d_{i,j}\}$
9. return $d$
2) Secure Distance Calculation between Sample and Cluster

Next, the silhouette coefficient needs to calculate the distance between each sample \(x_i\) and each cluster \(S_k\), which is defined as follows:

\[
D_{i,k} = \sum_{x_j \in S_k} d_{i,j} \quad (4)
\]

Let \(N_k\) be the number of samples contained in each cluster, \(E_{i,k} \in \{0,1\}\) is the relationship between sample \(x_i\) and cluster \(c_k\), \(T\) is the product of all \(N_k\), i.e. \(T = \prod_{k=0}^{K-1} N_k\). The implementation of the above process on the ciphertext is shown in Algorithm 6.

Algorithm 6 Secure Distance Calculation between Sample and Cluster (SDCSC)

**Input:** i) sample number \(m\); 
ii) cluster number \(K\); 
iii) the encrypted distance between samples to clusters \(D = \{D_{i,k}\}\); 
iv) encrypted relationship matrix between samples and clusters \(E = \{E_{i,k}\}\); 
v) the encrypted number of samples contained in each cluster \(N = \{N_0, N_1, \cdots, N_{K-1}\}\).

**Output:** i) encryption of all samples \(A = \{A_0, A_1, \cdots, A_{m-1}\}\); 
ii) encrypted cluster index of samples \(\{e_0, e_1, \cdots, e_{K-1}\}\); 
iii) encrypted product \(P = \{P_0, P_1, \cdots, P_{m-1}\}\); 
iv) encrypted product \(Q = \{Q_0, Q_1, \cdots, Q_{m-1}\}\); 
v) encrypted separation \(V = \{v_i, k\}\).

1: for \(k = 0\) to \(K-1\) do
2: for \(i = 0\) to \(m-1\) do
3: \([e_0] = SEQ([k], [e_0])\);
4: Let \([E_{i,k}] = [1]\cdot [e_0]^{N-1} = [1 - e_0]d_{i,j}\);
5: Update \(N_k:\ [N_k] = N_k\cdot [E_{i,k}]\);
6: for \(j = 0\) to \(m-1\) do
7: \([U] = RSM([E_{i,k}], [d_{i,j}])\);
8: Update \(D_{j,k}: [D_{j,k}] = [D_{j,k}]\cdot [U]\);
9: end for
10: end for
11: end for
12: Initial \([T] = [1]\), which is used to calculate the product of all \(N_k\);
13: for \(k = 0\) to \(K-1\) do
14: \([T] = RSM([T], [N_k])\);
15: end for
16: Let \(D = \{D_{i,k}\}\);
17: Let \(E = \{E_{i,k}\}\);
18: Let \(N = \{N_0, N_1, \cdots, N_{K-1}\}\);
19: return \(D, N, E, T\);

3) Secure Cohesion Calculation

Then, we come to the calculation of cohesion. For sample \(x_i\), its cohesion \(A_i\) is defined as follows:

\[
A_i = \frac{\sum_{x_j \in S_i} SSED(x_i, x_j)}{|S_{c_i}| - 1} \quad (5)
\]

where 
\(S_{c_i}\) is the cluster to which \(x_i\) belongs and \(|S_{c_i}|\) is number of samples in \(S_{c_i}\). The implementation process of the above formula on the ciphertext is shown in Algorithm 7.

Algorithm 7 Secure Cohesion Calculation (SCOC)

**Input:** i) sample number \(m\); 
ii) cluster number \(K\); 
iii) encrypted distance between samples to clusters \(D = \{D_{i,k}\}\); 
iv) the encrypted relationship matrix between samples and clusters \(E = \{E_{i,k}\}\); 
v) the encrypted number of samples contained in each cluster \(N = \{N_0, N_1, \cdots, N_{K-1}\}\).

**Output:** i) encrypted cohesion of all samples \(A = \{A_0, A_1, \cdots, A_{m-1}\}\); 
ii) encrypted cluster index of samples \(\{e_0, e_1, \cdots, e_{K-1}\}\); 
iii) encrypted product \(P = \{P_0, P_1, \cdots, P_{m-1}\}\); 
iv) encrypted product \(Q = \{Q_0, Q_1, \cdots, Q_{m-1}\}\); 
v) encrypted separation \(V = \{v_i, k\}\).

1: \([P_i] = [0], [Q_i] = [0], 0 \leq i < m\); 
2: for \(k = 0\) to \(K-1\) do
3: \([e_1] = SEQ([1], [N_k])\);
4: \([e_2] = [1] \cdot [e_1]^{N-1} = [1 - e_1]d_{i,j}\);
5: \([n_k] = [N_k] \cdot [-1] = [N_k - 1]\);
6: for \(i = 0\) to \(m-1\) do
7: \([u_0] = RSM([E_{i,k}], [n_k])\);
8: \([u_0] = RSM([u_0], [e_1]) \cdot RSM([1], [e_2])\);
9: \([Q_i] = [Q_i] \cdot [u_0] = [Q_i + u_0]\);
10: \([v_i, k] = RSM([D_{i,k}], [E_{i,k}])\);
11: \([v_i, k] = RSM([v_i, k], [e_1])\);
12: \([P_i] = [P_i] \cdot [v_i, k] = [P_i + v_i, k]\);
13: end for
14: end for
15: for \(i = 0\) to \(m-1\) do
16: \([P_i] = SITF([P_i]); [Q_i] = SITF([Q_i]);\)
17: \([A_i] = SFPD([P_i], [Q_i]);\)
18: end for
19: Let \(A = \{A_0, A_1, \cdots, A_{m-1}\}\);
20: return \(A\);

4) Secure Separation Calculation

Then we come to the calculation of separation \(b_i\). For sample \(x_i\), separation \(b_i\) is defined as follows:

\[
b_i = \min_{0 \leq k < K, k \neq c_i} \left\{ \frac{\sum_{x_j \in S_k} d(x_i, x_j)}{|S_k|} \right\} \quad (6)
\]

The implementation process of the above formula on the ciphertext is shown in Algorithm 8.

5) Secure Silhouette Coefficient

Silhouette coefficient combines cohesion and separation. Based on the above modules, we can calculate the silhouette coefficient of \(x_i\) and the overall silhouette coefficient \(sc\), i.e.

\[
sc_i = \frac{B_i - A_i}{\max(A_i, B_i)} \quad (7)
\]

\[
sc = \frac{1}{m} \sum_{i=1}^{m} sc_i \quad (8)
\]
Algorithm 8 Secure Separation Calculation (SSEC)

**Input:**
1. sample number $m$;
2. cluster number $K$;
3. encrypted distance between samples to clusters $D = \{(D_{i,k})\}$;
4. the encrypted relationship matrix between samples and clusters $E = \{(E_{i,k})\}$;
5. the encrypted number of samples contained in each cluster $N = \{N_0, N_1, \ldots, N_{K-1}\}$;
6. the encrypted product $T$;

**Output:**
1. encrypted separation of all samples $B = \{[B_0], [B_1], \ldots, [B_{m-1}]\}$;

1. Let $[G] = [10^{64}]$
2. for $k = 0$ to $K-1$ do
3. $[V_k] = \text{SINV}([N_k]); [R_k] = \text{RSM}([T], [V_k])$
4. for $i = 0$ to $m$ do
5. $[e_{3i}] = \{1 \cdot [E_{i,k}]\}^N - 1; [H] = \text{RSM}([D_{i,k}], [R_k])$
6. $[z_0] = \text{RSM}([e_{3i}], [G])$
7. $[S_1] = \text{RSM}([z_0], [H])$
8. $[Z_{i,k}] = [z_0] \cdot [z_1]$
9. end for
10. end for
11. for $i = 0$ to $m$ do
12. Let $[Z_i] = \{[Z_{i,0}], [Z_{i,1}], \ldots, [Z_{i,K-1}]\}$
13. $[I_i], [m_i] = \text{SIAN}([Z_i])$
14. $[m_i] = \text{SIAS}([m_i]); (T) = \text{SIAS}([T])$
15. $[B_i] = \text{SFPC}([m_i], (T))$
16. end for
17. Let $B = \{[B_0], [B_1], \ldots, [B_{m-1}]\}$
18. return $B$

Thus we can implement the silhouette coefficient on the ciphertext based on our toolkit MPOCT, which is shown in Algorithm 9.

VI. SECURITY ANALYSIS

In this section, we first analyze the security of our outsourcing computation protocols. Then, we demonstrate the security of our toolkit MPOCT.

A. THE SECURITY OF PROTOCOLS

Before we explain the security of our protocols, we first give the definition of the semantic security of a public key encryption system that supports partial decryption. As the semantic security of PCDP has been proven in [5], we will use it to demonstrate that our outsourcing computation protocols are secure.

Definition 1 (Semantic Security). Let $E = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme supporting partial decryption. We say that $E$ is semantically secure if for any polynomial-time adversary $A$ it has a negligible advantage (in the security parameter) in the following experiment (between the challenger and the adversary):

1) The challenger runs $\text{KeyGen}(1^k)$ to obtain a public and private key pair $(pk, sk)$ and splits the private key $sk$ into two parts $sk_1$ and $sk_2$. Then the challenger sends the public key $pk$ as well as one part of the secret key, e.g. $sk_1$ to the adversary $A$.
2) The adversary $A$ chooses two equal-length messages $m_0$ and $m_1$. Then the adversary $A$ sends them to the challenger.
3) The challenger chooses a random bit $b \in \{0, 1\}$ and sends the ciphertext $c' = \text{Enc}(m_b)$ to $A$.
4) The adversary $A$ outputs a bit $b'$ as a guess of $b$.

The adversary’s advantage in the above experiment is defined as $\text{Adv}_E(k) := |Pr[b' = b] - \frac{1}{2}|$.

Theorem 1. [5] PCDP satisfies Semantic Security, i.e., $\text{Adv}_{PCPD}(k) := |Pr[b' = b] - \frac{1}{2}| < \frac{1}{2^k}$.

Here, we discuss the security model for securely implementing an ideal functionality in the presence of non-colluding semi-honest adversaries. For simplicity, we operate according to the specific scenario of our functionality, which involves three parties, challenger DO (i.e. $D_0$), CP (i.e. $S_1$ and CSP i.e. $S_2$). Therefore, we need to construct three simulators $\text{Sim}_0 = \text{Sim}_{D_0}, \text{Sim}_{S_1}, \text{Sim}_{S_2}$ to against three kinds of adversaries ($A_{D_0}, A_{S_1}, A_{S_2}$) that respectively corrupt $D_0, S_1$ and $S_2$.

Theorem 2. The SCDP protocol described in Section IV is to securely calculate the ciphertext result of division on the premise that the ciphertext of the dividend and the plaintext...
of the divisor are known in the presence of semi-honest (non-colluding) adversaries $A = (A_{D_0}, A_S, A_{S_2})$.

**Proof.** We now demonstrate how to construct three independent simulators $\text{Sim}_{D_0}, \text{Sim}_S, \text{Sim}_{S_2}$.

$\text{Sim}_{D_0}$ receives $x$ and $y$ as input and simulates $A = A_{D_0}$ as follows: it generates a public integer $y$ and ciphertext $[x] = \text{Enc}(x)$ of $x$. Finally, it returns $[x]$ and $y$ to $A_{D_0}$ and outputs $A_{D_0}$’s entire view.

The view of $A_{D_0}$ consists of the encrypted data. And the views of $A_{D_0}$ in the real and ideal executions are indistinguishable due to the semantic security of PCPD.

$\text{Sim}_S$ simulates $A_S$ as follows: First, it generates (fictitious) a public number $y$ and encryption $[x]$ and $[y]$ by running $\text{Enc}(\cdot)$ on randomly chosen $x$ and above $y$. Then it uses $[x]$ and $[y]$ as the inputs of $\text{Sim}_{S_1}$ and generates $[M_1]$. Next, it uses $[y]$ as the input of $\text{Sim}_{S_1}^{\text{SMOD}(\cdot)}$ and generates $[I_1]$. Then, it calculates $[M_2] = [x] [M_1]^{N-1}$, uses $[M_2]$ and $[I_1]$ as the inputs of $\text{Sim}_{S_1}^{\text{SINV}(\cdot)}$ and generates $[f]$. If $y < 0$, it calculates $[f^*] = [f]^{N-1}$. Otherwise, $[f^*] = [f]$. Finally, $\text{Sim}_{S_1}$ sends the encryption $[M_1], [I_1], [M_2], [f], [f^*]$ to $A_S$. If $A_S$ replies with $\bot$, then $\text{Sim}_{S_1}$ returns $\bot$.

$\text{Sim}_{S_2}$ is analogous to $\text{Sim}_{S_1}$.

The security proofs of $\text{SDIV}, \text{SIMF}, \text{SPOW}, \text{SLOG}$ are similar to that of the $\text{SCDP}$ under the semi-honest (non-colluding) adversaries $A = (A_{D_0}, A_S, A_{S_2})$. For the encrypted floating point number calculations (include $\text{SFPR}, \text{SFPO}, \text{SFPU}, \text{SFTI}$ and $\text{SITF}$), the security relies on the basic encrypted integer calculation (the prove method is similar to that of $\text{SCDP}$), which has been proven. Due to the semantic security of PCPD, it is secure for all calculations to be performed in the ciphertext domain.

**B. THE SECURITY OF MPOCT**

Here, we give an analysis to show that our toolkit MPOCT can resist the system attackers defined in Section 2. The specific analysis is as follows:

- If adversary $A^*$ eavesdrops on the transmission between DO and CP, then $A^*$ can obtain the original encrypted data and the final result. In addition, adversary $A^*$ can obtain the encrypted result transmitted between CP and CSP through eavesdropping. However, these data are encrypted during transmission. Based on the semantic security of the PCPD cryptosystem, adversary $A^*$ will not be able to decrypt the ciphertext without knowing the private key of DO. Since the public key and private key in the system are distributed securely to each participant by KGC, our system model will not be affected by...
TABLE 2. The Performance of protocols for Integer (80-bit Security Level)

| Protocol | CP compute. | CSP compute. | Total compute. | Commu. |
|----------|-------------|--------------|----------------|-------|
| RSM      | 13.927 ms   | 7.698 ms     | 1.262 ms       | 1.265 KB |
| SLT      | 5.750 ms    | 7.698 ms     | 1.262 ms       | 1.002 KB |
| SXOR     | 32.278 ms   | 16.570 ms    | 2.510 ms       | 4.514 KB |
| SEQ      | 49.091 ms   | 26.850 ms    | 75.941 ms      | 0.750 KB |
| SEXP     | 5.461 ms    | 5.341 ms     | 10.802 ms      | 0.750 KB |
| SINV     | 3.863 ms    | 6.293 ms     | 10.156 ms      | 0.750 KB |
| SMOD     | 16.376 ms   | 10.244 ms    | 26.620 ms      | 1.752 KB |
| SCDP     | 34.795 ms   | 24.678 ms    | 59.473 ms      | 3.758 KB |
| SIMF     | 49.089 ms   | 32.596 ms    | 81.685 ms      | 5.014 KB |
| SPOW     | 138.281 ms  | 76.443 ms    | 214.724 ms     | 11.304 KB |
| SLOG     | 1.136 s     | 0.610 s      | 1.746 s        | 95.360 KB |
| SDIV     | 7.294 s     | 3.324 s      | 10.618 s       | 260.226 KB |

TABLE 3. The Performance of protocols for FPNs (80-bit Security Level)

| Protocol | CP compute. | CSP compute. | Total compute. | Commu. |
|----------|-------------|--------------|----------------|-------|
| SFPF     | 5.542 s     | 3.749 s      | 9.102 s        | 513.285 KB |
| SFPM     | 1.497 s     | 0.957 s      | 2.545 s        | 140.731 KB |
| SFPC     | 1.623 s     | 0.867 s      | 2.490 s        | 136.893 KB |
| SFPS     | 1.103 s     | 0.601 s      | 1.704 s        | 98.010 KB |
| FPEQ     | 0.163 s     | 0.085 s      | 0.248 s        | 14.329 KB |
| SFPO     | 3.60 s      | 0 ms         | 3.60 s         | 0 KB |
| SFPU     | 5.548 s     | 3.470 s      | 9.018 s        | 513.285 KB |
| SFPR     | 16.770 s    | 9.312 s      | 26.082 s       | 1.194 MB |
| SFPD     | 18.270 s    | 10.266 s     | 28.536 s       | 1.331 MB |
| SFTI     | 1.368 s     | 0.702 s      | 2.070 s        | 97.888 KB |
| SITF     | 8.609 s     | 4.579 s      | 13.188 s       | 667.670 KB |

Man-in-the-middle attack.
- Suppose adversary $A^*$ has compromised the CP (or CSP) to obtain the challenge RU’s partially private key. As the private key is randomly split by executing KeyS algorithm of PCPD, adversary $A^*$ is unable to recover the private key of the challenger DO to decrypt the ciphertext. In addition, adversary $A^*$ cannot obtain useful information even if the CSP is compromised, because our protocols use the known technique of “blinding” the plaintext \([29]\), given the ciphertext of the message, we use the additive homomorphism of the PCPD cryptosystem to add random messages to it. Therefore, the original plaintext becomes blinded.

VII. EXPERIMENTAL EVALUATION

In this section, we first evaluate the performance of protocols in MPOCT. Then we analyze the performance of SKM and SSC.

A. EXPERIMENTS OF MPOCT

First, we let $N$ be 1024 bits to achieve 80-bit security \([30]\). The computation cost and communication cost of the proposed MPOCT are evaluated using a Python program, and the experiments are performed on a single server with 2.3 GHz one-core processor and 8 GB RAM memory. The performances of protocols for both integers and FPNs in MPOCT are respectively shown in Table 2 and Table 3, while some protocols are evaluated under specific parameters: \(\text{SPOW}(y = 31)\), \(\text{SLOG}(\delta = 20)\), \(\text{SDIV}(\beta_0 = 3, \eta_0 = 32)\), \(\text{SITF}(\eta_{\max} = 32)\). We can find that the protocols of integer ciphertext are faster in computing time and lower in communication overhead than protocols of FPNs. Then, we discuss the factors that affect the performance of these protocols. From Figure 3(a)-(h), we can easily find that both the running time and communication overhead of the protocols increase with $N$. This is because the running time required for basic operations (modular multiplication and exponential) increases while $N$ increases. Meanwhile, more bits need to be transmitted. In addition, the running time of some protocols will also change according to specific parameters, which is shown in Figure 3(i)-(l). We find that SDIV has the highest computational efficiency, when $\beta_0$ is equal to 3. In addition, the calculation efficiency of SLOG changes linearly with $\delta$ and the calculation efficiency of SITF also changes linearly with $\eta_{\max}$. Since SPOW is based on the idea of Binary Exponentiation, the calculation efficiency only changes logarithmically with $y$.

B. EXPERIMENTS OF SKM AND SSC

1) Computational Efficiency Analysis

According to the algorithm process of SKM and SSC, we found that the main factors affecting the computational efficiency are the bit size of the public key $|N|$, the number of clusters $k$, the number of samples $m$ and the number of features $n$. From Figure 4(a)(b)(c), we find that the calculation time of SKM is linearly related to $k$, $m$ and $n$. From Figure 4(d)(e)(f), we find that the calculation time of SSC is non-linearly related to $m$, but it is linearly related to $k$ and $n$. Since SSC needs to calculate the distance between paired samples, the calculation time of SSC is related to $m^2$. Therefore, the calculation time of SSC will increase significantly when the number of samples $m$ increases. In addition, we have also compared with the previous scheme PPODC \([7]\) and PPCOM \([9]\). We fix the bit size $N$ of public key in our Paillier cryptosystem to be 1024. When $m = 10000$, $n = 10$, $k = 4$, the running time of PPODC on Python is 13360 minutes per iteration and 1856 minutes on PPCOM, while our solution SKM only needs 577 minutes, which is more than 20 times faster than PPODC and 3 times faster than PPCOM.

2) Performance Analysis

To demonstrate the effectiveness of SKM and SSC, we conduct extensive experiments over three public datasets. The dataset description is shown in Table 4.

First, we compare the performance of three different K-means algorithms, i.e. SKM, K-means on plaintext, K-means using differential privacy (DP-K-means \([22]\)). Parameter configuration: the number of iterations $E$ is set as 10. The...
privacy-preserving budget of DP-K-means \( \epsilon \) is set as 1. Since the datasets have provided label information, we conduct our experiments in two aspects: with and without labels.

Table 5 shows the classification accuracy of three methods with labels. We can find that the K-means on plaintext outperforms the others with the highest accuracy. Our scheme SKM achieves the second place, and it is competitive with the performance of K-means on plaintext. This is because SKM requires a given fixed number of iterations \( E \) and thus may not fully converge. To ensure convergence, it is only necessary to set a larger number of iterations. However, the computational cost increases linearly as the number of iterations \( E \) increases. Therefore, in order to balance the computational efficiency and the usability of the algorithm, we choose a small number of iterations \( E \) for experiments.

DP-K-means performs relatively inferior due to the added noise in data.

Since ground truth labels are hard to obtain in a real application scenario, we use silhouette coefficient for evaluation. As an effective index to evaluate the performance of clustering, the silhouette coefficient ranges from \(-1\) to \(1\). The closer the it is to \(1\), the better the clustering performance is; on the contrary, the closer the silhouette coefficient is to \(-1\), the worse the clustering performance is. Table 6 shows the silhouette coefficient of three methods. The situation is similar to those with labels. K-means on plaintext surpasses the others with the highest silhouette coefficient. The performance of our scheme SKM is competitive with that on K-means. DP-K-means, however, performs relatively inferior for the same reason above.

Last, we validate the effectiveness of secure silhouette coefficient. Based on the clustering results obtained by SKM, we evaluate the performance of silhouette coefficient on the decrypted plaintext (SC) and secure silhouette coefficient on the ciphertext (SSC). We can observe from Table 7 that the proposed SSC in this paper can ensure the same availability as that on the plaintext.
| Dataset          | SC  | SSC |
|------------------|-----|-----|
| Iris             | 0.480 | 0.480 |
| Wine             | 0.284 | 0.284 |
| Breast Cancer    | 0.345 | 0.345 |

VIII. CONCLUSION AND FUTURE WORK

In this paper, a novel multifunctional and privacy-preserving outsourcing computation toolkit (MPOCT) is proposed to support several homomorphic computing protocols including division and power on ciphertext of integers and floating point numbers. Concretely, we first extend several homomorphic operations on the ciphertext of floating-point numbers based on the previous framework POCF. However, due to the low efficiency and high communication overhead on the ciphertext of floating-point numbers, we further extend the secure outsourcing computation protocols on the ciphertext of integers. After that, homomorphic mutual conversion protocols between integer and floating-point ciphertext are proposed to balance the efficiency and feasibility of computation. Next, we implement a homomorphic K-means algorithm based on MPOCT for clustering and design the homomorphic silhouette coefficient as the evaluation index, providing an informative cluster assessment for local users with limited resources. Comprehensive experimental results and security analysis have proved the proposed MPOCT can achieve efficiency and utility without privacy leakage to unauthorized parties.

In the future, MPOCT is expected to resort to homomorphic neural network modules such as homomorphic convolution and homomorphic pooling. In addition, MPOCT can only be applied to the algorithms with certain termination conditions (i.e. the homomorphism of the while loop cannot be implemented) since the servers cannot be aware of the comparison results of ciphertext. An improved scheme of solving this problem is to be explored in the future.

REFERENCES

[1] C. Butpheng, K. Yeh, and H. Xiong, “Security and privacy in iot-cloud-based e-health systems - A comprehensive review,” Symmetry, vol. 12, no. 7, p. 1191, 2020.

[2] A. Wood, K. Najarian, and D. Kahrobaei, “Homomorphic encryption for machine learning in medicine and bioinformatics,” ACM Comput. Surv., vol. 53, no. 4, pp. 70:1–70:35, 2020.

[3] Z. A. Almusaylim and N. Z. Jianjhi, “Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing,” Wirel. Pers. Commun., vol. 111, no. 1, pp. 541–564, 2020.

[4] Y. Yang, X. Huang, X. Liu, H. Cheng, J. Weng, X. Luo, and V. Chang, “A comprehensive survey on secure outsourced computation and its applications,” IEEE Access, vol. 7, pp. 159,426–159,465, 2019.

[5] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, “Privacy-preserving outsourced calculation on floating point numbers,” IEEE Trans. Inf. Forensics Secur., vol. 11, no. 11, pp. 2513–2527, 2016.

[6] X. Liu, K. R. Choo, R. H. Deng, R. Lu, and J. Weng, “Efficient and privacy-preserving outsourced calculation of rational numbers,” IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 27–39, 2018.

[7] F. Yao, B. K. Samantha, B. Bertino, X. Yi, and D. Liu, “Privacy-preserving and outsourced multi-user k-means clustering,” in IEEE Conference on Collaboration and Internet Computing, CIC 2015, Hangzhou, China, October 27–30, 2015, pp. 80–89.

[8] 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 Big Data Analytics and Knowledge Discovery - 19th International Conference, DaWAK 2017, Lyon, France, August 28-31, 2017, Proceedings, 2017, pp. 274–285.

[9] H. Rong, H. Wang, J. Liu, J. Hao, and M. Xian, “Privacy-preserving k-means clustering under multi-tier setting in distributed cloud environments,” Secur. Commun. Networks, vol. 2017, pp. 3910126:1–3910126:19, 2017.

[10] P. Sun, “Security and privacy protection in cloud computing: Discussions and challenges,” J. Netw. Comput. Appl., vol. 160, p. 106242, 2020.

[11] J. Domingo-Ferrer, O. Farràs, J. Ribes-González, and D. Sánchez, “Privacy-preserving cloud computing on sensitive data: A survey of methods, products, and challenges,” Comput. Commun., vol. 140–141, pp. 38–60, 2019.

[12] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Advances in Cryptology - EUROCRYPT ’99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceedings, 1999, pp. 223–238.

[13] T. E. Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” IEEE Trans. Inf. Theory, vol. 31, no. 4, pp. 469–472, 1985.

[14] L. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, pp. 169–178.

[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” ACM Trans. Comput. Theory, vol. 6, no. 3, pp. 13:1–13:36, 2014.

[16] I. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” IACR Cryptol. ePrint Arch., vol. 2012, p. 144, 2012.

[17] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I, 2017, pp. 409–437.

[18] Y. Liu, Y. Luo, Y. Zhu, Y. Liu, and X. Li, “Secure multi-label data classification in cloud by additionally homomorphic encryption,” Inf. Sci., vol. 468, pp. 89–102, 2018.

[19] X. Liu, R. H. Deng, K. R. Choo, Y. Yang, and H. Pang, “Privacy-preserving outsourced calculation toolkit in the cloud,” IEEE Trans. Dependable Secur. Comput., vol. 17, no. 5, pp. 898–911, 2020.

[20] J. Li, X. Xiang, S. Lin, X. Ma, and Y. Tang, “Privacy preservation for machine learning training and classification based on homomorphic encryption schemes,” Inf. Sci., vol. 526, pp. 166–179, 2020.

[21] H. Zong, H. Huang, and S. Wang, “Secure outsourcing of matrix determinant based on fully homomorphic encryption,” IEEE Access, vol. 9, pp. 22,651–22,661, 2021.

[22] C. Xia, J. Hua, W. Tong, and S. Zhong, “Distributed K-means clustering guaranteeing local differential privacy,” Comput. Secur., vol. 90, p. 101699, 2020.

[23] P. Mohassel, M. Rosulek, and N. Trieu, “Practical privacy-preserving k-means clustering,” Cryptology ePrint Archive, 2019.

[24] Y. Zou, Z. Zhao, S. Shi, L. Wang, Y. Peng, Y. Ping, and B. Wang, “Highly secure privacy-preserving outsourced k-means clustering under multiple keys in cloud computing,” Security and Communication Networks, vol. 2020, 2020.

[25] S. Z. Selim and M. A. Ismail, “K-means-type algorithms: A generalized convergence theorem and characterization of local optimality,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 6, no. 1, pp. 81–87, 1984.

[26] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie, “Edge-based differential privacy computing for sensor cloud systems,” J. Parallel Distrib. Comput., vol. 17, no. 5, pp. 75–85, 2020.

[27] Q. Do, B. Martini, and K. R. Choo, “A forensically sound adversary model for mobile devices,” CoRR, abs/1509.06815, 2015.

[28] R. P. J., “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” Journal of computational and applied mathematics, vol. 20, pp. 53–65, 1987.

[29] P. Weyer, E. Tews, and S. Katzenbeisser, “Efficiently outsourcing multiparty computation under multiple keys,” IEEE Trans. Inf. Forensics Secur., vol. 8, no. 12, pp. 2046–2058, 2013.

[30] A. D and S. A. M, “Security evaluation of paillier homomorphic encryption scheme,” J. Asian Journal of Research in Computer Science, vol. 2020, pp. 12–17, 2020.
JIALIN LI received her bachelor’s degree in Software Engineering from Jilin University in 2019. She is currently pursuing master’s degree in Software Engineering from East China Normal University. Her research interests include semantic segmentation, graph data processing and privacy-preserving related topics.

PENGHAO LU received his bachelor’s degree in Science from East China University of Science and Technology. He is currently pursuing master’s degree in Software Engineering from East China Normal University, Shanghai, China. His research interests mainly include machine learning and privacy-preserving.

XUEMIN LIN is a UNSW distinguished Professor-Scientia Professor, and the head of database and knowledge research group in the school of computer science and engineering at UNSW. He is a concurrent professor in the School of Software, East China Normal University. He is also a distinguished visiting Professor at Tsinghua University and visiting Chair Professor at Fudan University. He is a fellow of IEEE. Xuemin Lin is working in the areas of scalable processing and mining of large-scale data, including graph, spatial-temporal, streaming, text and uncertain data. Xuemin Lin currently serves as the editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2017-now). He was an associate editor of ACM Transactions Database Systems (2008-2014) and IEEE Transactions on Knowledge and Data Engineering (2013-2015), and an associate editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2015-2016).