Hindawi
Security and Communication Networks
Volume 2017, Article ID 3910126, 19 pages
https://doi.org/10.1155/2017/3910126

Research Article

Privacy-Preserving k-Means Clustering under Multiowner Setting in Distributed Cloud Environments

Hong Rong, Huimei Wang, Jian Liu, Jialu Hao, and Ming Xian

State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, National University of Defense Technology, Changsha, China

Correspondence should be addressed to Hong Rong; r.hong_nudt@hotmail.com

Received 16 June 2017; Revised 15 September 2017; Accepted 27 September 2017; Published 13 November 2017

Academic Editor: Xiangyang Luo

With the advent of big data era, clients who lack computational and storage resources tend to outsource data mining tasks to cloud service providers in order to improve efficiency and reduce costs. It is also increasingly common for clients to perform collaborative mining to maximize profits. However, due to the rise of privacy leakage issues, the data contributed by clients should be encrypted using their own keys. This paper focuses on privacy-preserving k-means clustering over the joint datasets encrypted under multiple keys. Unfortunately, existing outsourcing k-means protocols are impractical because not only are they restricted to a single key setting, but also they are inefficient and nonscalable for distributed cloud computing. To address these issues, we propose a set of privacy-preserving building blocks and outsourced k-means clustering protocol under Spark framework. Theoretical analysis shows that our scheme protects the confidentiality of the joint database and mining results, as well as access patterns under the standard semihonest model with relatively small computational overhead. Experimental evaluations on real datasets also demonstrate its efficiency improvements compared with existing approaches.

1. Introduction

With tremendous amount of data collected each day, it is increasingly difficult for resource-constrained clients to perform computationally intensive tasks locally. There are numerous cases regarding this, such as mobile phones or sensors in the IoT system with limited battery and small companies that lack hardware and software infrastructures. Thus, it is a viable option to outsource data mining tasks to Cloud Service Provider (CSP) which provides massive storage and computation power in a cost-efficient way [1]. By leveraging the cloud platforms, a great many giant IT companies have offered machine learning services to help clients to train and deploy their own models, for example, Amazon Machine Learning [2], Google Cloud Machine Learning Engine [3], and IBM Watson [4]. Despite the advantages, privacy issues are the critical concerns for cloud users to employ these services. For many data records may contain sensitive information, such as health condition, financial records, and location information, outsourcing them in plain form inevitably reveals personal privacy to CSP which may be untrustworthy or even malicious. For instance, it is favorable to improve the diagnosis accuracy by utilizing data mining techniques over medical records gathered from multiple patients [5], while releasing such information to public directly is prohibited by laws in many countries, for example, HIPAA [6]. Thereby, appropriate mechanisms should be designed to guarantee that the outsourced mining tasks are executed in a privacy-preserving manner.

In this paper, we focus on privacy protection techniques on outsourced k-means clustering, which is a widely used data mining algorithm in the fields of image analysis, information retrieval, pattern recognition, and so on. The outsourcing datasets are contributed by multiple data owners who are willing to collaborate in outsourced clustering in order to obtain more accurate results. Normally, the data records are encrypted via cryptographic tools to prevent them from being disclosed to other parties. The goal of our solution is to let cloud servers perform clustering over the encrypted data.

Traditional privacy-preserving clustering schemes cannot be directly adopted to address the privacy issues for outsourcing. They aim at computing clusters through interactions among different data holders without revealing respective data to others [7–9], whereas, in our case, the data are stored and processed by the cloud rather than clients themselves.

Most existing works on outsourced privacy-preserving clustering require cloud clients to utilize the same key for data encryption [10–15]. In practice, sharing the same encryption key has some disadvantages: (1) for symmetric encryption scheme, compromised data owners can easily decrypt other owners' encrypted data if they launch eavesdropping attacks, which suites the case in [10–14]; (2) for asymmetric encryption, if the datasets are encrypted under cloud's public key, data owners cannot decrypt their uploaded data due to not knowing the private key [15]. One way to solve this is allowing data owners to encrypt their data by their own keys, but this calls for computation over encrypted data under multiple keys (denoted by multikey). The only work [16] concerning multikey clustering was constructed on the multiplicative transformation method proposed in [17], whereas their proposed solutions expose all owners' keys to the query user, which causes security risks if the user is compromised.

Another issue of current research is that their threat models do not suffice higher level attacks. The essence of underlying schemes in [10, 14, 16] is to apply random matrix to encrypt data. These are secure against known-sample attack [17] (namely, attacker only knows some instances). But if attacker also knows the corresponding encrypted values of some data (i.e., chosen plaintext attack), the remaining instances may be recovered by setting up enough equations. In addition, the fully homomorphic encryption (FHE) used in [12, 13] is not secure according to [18]. Furthermore, access patterns (assignment of data objects, etc.) are disclosed to cloud servers [14, 16]. They may be used to derive sensitive information regardless of data encryption, as indicated by [19, 20]. Last but not least, few works consider combining their privacy-preserving techniques with large-scale data processing frameworks for boosting efficiency.

To address these challenges, we propose a novel solution for Privacy-Preserving k-means Clustering Outsourcing under Multikeys (PPCOM), which enables distributed cloud servers to perform clustering collaboratively over the aggregated datasets with no privacy leakage. Specifically, the major contributions of this paper are fourfold:

(i) Firstly, our solution allows the cloud to perform arithmetic computations over encrypted data under multiple keys. It is achieved by transforming ciphertexts under different keys into ones under the unified key through the double decryption cryptosystem. Since the encryption scheme is only partially homomorphic, we propose a secure addition subprotocol under the noncolluding two-server model, which does not reveal anything about input or output, including the input ratio. Based on these, the cloud can compute Euclidean distances between records and cluster centers.

(ii) Secondly, we propose two secure primitives to evaluate equality test and compare ciphertexts, addressing the problem that encrypted data are incomparable because of probabilistically random distribution. These two are further utilized in other privacy-preserving building blocks, including minimum Euclidean distance computing and encrypted bitmap converting, as well as cluster center updating.

(iii) Thirdly, on the basis of those privacy-preserving building blocks, we design PPCOM protocol by integrating Spark framework to accelerate the outsourcing process, which works in distributed cloud environments. Moreover, PPCOM requires no clients' participation after they upload their encrypted datasets under their own private keys.

(iv) Fourthly, theoretical analysis demonstrates that the proposed protocol not only protects the privacy of aggregated data records and clustering centers, but also hides access patterns under the semihonest model. Experimental results on real dataset shows that PPCOM is much more efficient than existing methods in terms of computational overhead.

The rest of the paper is organized as follows. Section 2 reviews the related works. In Section 3, we describe the preliminaries required in the understanding of our proposed PPCOM. In Section 4, we formalize the system model, threat model, and design objectives. The design details of the proposed privacy-preserving building blocks as well as outsourcing protocol are presented in Section 5. We provide security analysis in Section 6. Section 7 shows the theoretical and experimental evaluation results. Finally, we conclude the paper and outline future work in Section 8.

2. Related Work

There have been a lot of works on privacy-preserving distributed k-means clustering [7–9]. These works have different security requirements and design goals compared with our work. In the distributed setting where data are partitioned among multiple parties, the clustering task is undertaken by data holders instead of centralized servers. Generally, their schemes exploit secure multiparty computation (SMC) techniques so as to preclude one's data from being disclosed to the others except the final results. Whereas, in terms of clustering outsourcing, data owners intend to transfer the major workloads to cloud servers for the sake of reducing costs and improving efficiency. During the entire outsourcing process, all the inputs and outputs, as well as intermediate results, are supposed to be encrypted to ensure confidentiality.

As for outsourced k-means clustering, Liu et al. [12] first leveraged FHE technique to perform outsourced clustering. To compare encrypted distances, their approach requires data owner to provide trapdoor information during each iteration, which entails heavy overhead on clients. To reduce the amount of data owner participation, Almutairi et al. [13] presented an efficient mechanism by using the concept of an Updatable Distance Matrix (UDM). Nevertheless, both works

reveal partial privacy to cloud servers, such as the size of each cluster and the distance between data object and centroid. Moreover, the encryption scheme adopted in [12, 13] is not secure according to [18].

Another line of research for outsourced clustering is to use distance-preserving data perturbation or transformation techniques to encrypt dataset [21]. Keeping the distance after encryption enables the cloud to update clusters independently without data owner's involvement, which achieves approximate efficiency compared with unencrypted data. However, as [17, 22] pointed out, these solutions are weak in security. Specifically, if attackers obtain some original instances (i.e., known-sample attack (KSA) [17]), the rest may be recovered by identifying corresponding encrypted ones and setting up enough equations. The work by Lin [11] focused on kernel k-means instead of standard k-means to avoid the preserving-distance vulnerability of random transformation. For outsourced collaborative data mining, Huang et al. [16] utilized asymmetric scalar-product-preserving encryption (ASPE) proposed in [17] that is resilient to KSA to compare distances. However, Yao et al. [23] demonstrated that ASPE is vulnerable to the linear analysis attack (LAA) [23]. To accelerate clustering efficiency, much research [24, 25] has been done to integrate MapReduce into k-means algorithm and optimize its performance, while few of them take privacy protection into account. Most recently, a secure scheme based on MapReduce to support large-scale dataset was proposed by Yuan and Tian [14]. By preserving the sign of encrypted distance difference like ASPE, their approach enables the cloud to assign data object to its closest cluster. It is resistant against both KSA and LAA. Unfortunately, none of the previously mentioned encryption schemes were formally proved to be secure against chosen plaintext attack (CPA); meanwhile some sensitive information, such as assignment of data objects and size of clusters, is directly disclosed to cloud servers.

To further reenforce security, Rao et al. [15] proposed a semantic secure scheme for outsourced distributed clustering over the aggregated encrypted data from multiple users. Based on Paillier cryptosystem, their solution protects not only confidentiality of data contents, but also access patterns from cloud servers and other users. In addition, participation of users is no longer required during outsourcing. Their design objective is similar to ours, except that their protocol does not support computation under multikeys and Spark framework. Besides, the cost of secure comparison is too heavy since each input has to be decomposed into encrypted bits by calling SBD subroutine [26]. This will be demonstrated in the experimental evaluations in Section 7. In regard to computation under multikey setting, López et al. [27] proposed a new FHE; however its efficiency suffers from complex key-switching and heavy interactions among users. There are other works that utilize double decryption mechanism [28] or proxy reencryption technique [29] to convert ciphertext keys, allowing two servers to conduct addition and multiplication operations under multikeys. Nevertheless, these basic operations still cannot fulfill the need to perform more sophisticated data mining computations, for example, similarity measurement.

3. Preliminaries

In this section, we briefly introduce the typical k-means clustering algorithm and public key cryptosystem with double decryption mechanism, serving as the basis of our solution.

3.1. k-Means Clustering. Given records $t_1, \ldots, t_l$, the k-means clustering algorithm partitions them into k disjoint clusters, denoted by $c_1, \ldots, c_k$. Let μ_i be the centroid value of c_i. Record t_j assigned to c_i has the shortest distance to μ_i compared with its distances to other centroids, where $i \in [1, k]$ and $j \in [1, l]$. Let $V_{l\times k}$ be the matrix defining the membership of records, in which $V_{i,j} \in \{0, 1\}$, for $1 \le i \le l$, $1 \le j \le k$. Note that the ith record belongs to c_j if $V_{i,j} = 1$; otherwise, $V_{i,j} = 0$.

Initial k records are selected randomly as cluster centers $\mu_1, \ldots, \mu_k$. Then the algorithm executes in an iterative fashion. For t_i, the algorithm computes Euclidean distance between t_i and every centroid μ_j for $1 \le j \le k$ and updates V according to $\arg\min_j \|t_i - \mu_j\|^2$, that is, assigns t_i to the closest cluster c_j. Later, the centroid μ_j is derived by computing the mean values of attributes of records belonging to c_j. With the updated $c_1, \ldots, c_k$, the clustering algorithm begins the next iteration. Finally, the algorithm terminates if the matrix V does not vary any more or if a predefined maximum count of iterations is reached [10].

3.2. Public Key Cryptosystem with Double Decryption. Public key cryptosystem with double decryption mechanism (denoted by PKC-DD) allows an authority to decrypt any ciphertext by using the master secret key without consent of corresponding owner. In this paper, we use the scheme proposed by Youn et al. [30] as our secure primitive, which is more efficient than the scheme in [31] in that Youn et al.'s approach applies smaller modulus in cryptographic operations. The major steps are shown in the following.

(i) *Key generation* ($\mathsf{KeyGen}(\kappa) \rightarrow N, g, \mathrm{msk}, \mathrm{pk}, \mathrm{sk}$): given a security parameter k, the master authority chooses two primes p and q ($|p| = |q| = \kappa$) and defines $N = p^2 q$. Then it chooses a random number g in $\mathbb{Z}_N^*$ such that the order of $g_p = g^{p-1} \bmod p^2$ is p. The master secret key $\mathrm{msk} = (p, q)$ is known only to the authority. The public parameters are N, g. A cloud user picks a random integer $\mathrm{sk} \in \{0, 1, \ldots, 2^{\kappa-1} - 1\}$ as secret key and computes $\mathrm{pk} = g^{\mathrm{sk}} \bmod N$ as public key.

(ii) *Encryption* ($\mathsf{Enc}(\mathrm{pk}, m) \rightarrow C$): the encryption algorithm takes the message $m \in \mathbb{Z}_N$ and pk as inputs and outputs $C = (A, B)$, where $A = g^r \bmod N$, $B = \mathrm{pk}^r \cdot m \bmod N$, and r is a random $\kappa - 1$ bit integer.

(iii) *Decryption with user key* ($\mathsf{uDec}(\mathrm{sk}, C) \rightarrow m$): the decryption algorithm takes ciphertext C and sk as inputs, and outputs the message m by computing $m \leftarrow B/A^{\mathrm{sk}} \bmod N$.

(iv) *Decryption with master key* ($\mathsf{mDec}(\mathrm{msk}, \mathrm{pk}, C) \rightarrow m$): given msk, pk, and C, the authority decrypts C by factorizing N. The secret key of C can be obtained by computing $\mathrm{sk} \leftarrow L(\mathrm{pk}^{p-1})/L(g_p)$, where function

L is defined as $L(x) = (x-1)/p$. Then, m is recovered by computing $m \leftarrow B/A^{\mathrm{sk}} \bmod N$.

By applying the general conversion method in [32], the scheme was claimed to be IND-CCA2 secure under the hardness of solving the p-DH Problem [30]. However, Galindo and Herranz [33] have constructed an attack by generating invalid public keys and querying for the master decryption, which may lead to factorization of N. To solve this, we adopt a slight modification of the scheme by checking the validity of the secret key, as proposed in [33]. If $\mathrm{sk} \geq 2^{\kappa-1}$, the master entity outputs a rejection message; otherwise, the decryption proceeds as usual.

4. Problem Statement

In this section, we formally describe our system model, threat model, and design objectives.

4.1. System Model. In our system model depicted in Figure 1, there are two types of entities, that is, Cloud Users and Cloud Service Provider. Cloud Users consist of Data Owners and Query Client. The Cloud Service Provider can be divided into Storage and Computation Provider and Cryptographic Service Provider. Storage and Computation Provider is composed of dozens of Executing Servers, whereas Cryptographic Service Provider comprises a Key Authority Server and a group of Assistant Servers. The description of each party is given as follows.

(1) Data owner (DO): DO is the proprietor of a large dataset. Due to lack of hardware and software resources, DO prefers to outsource his data to the cloud for storage and collaborative data mining. There are $\mathrm{DO}_1, \ldots, \mathrm{DO}_n$ in the system. DO_i has dataset D_i which contains m attributes and l_i records, for $i \in [1, n]$. The total number of records is $L = \sum_i^n l_i$.

(2) Query client (QC): QC is an authorized party requesting k-means clustering tasks over the federate datasets. QC should not be involved in outsourced computation and should be able to decrypt the result with his own secret key.

(3) Executing worker (EW): EW server is a cluster node within Storage and Computation Provider, which is responsible for storing DO's dataset and performing computation over them. There are $\mathrm{EW}_1, \ldots, \mathrm{EW}_\theta$ in the system. They together constitute a parallel Spark cluster, working on the same distributed file system like HDFS and providing cloud users with massive storage and computing power.

(4) Key authority (KA): KA belongs to Cryptographic Service Provider, which is assigned with distribution and management of public parameters and public/private keys, as well as the master key of the cryptosystem.

(5) Assistant worker (AW): AW is also part of Cryptographic Service Provider. AW server holds the public/private keys generated by KA, with which AW is able to assist EW to execute a series of privacy-preserving building blocks. We assume that there are ϑ AWs, that is, $\mathrm{AW}_1, \ldots, \mathrm{AW}_\vartheta$. All AWs and KA constitute the cluster of Cryptographic Service Provider. Note that they offer sufficient computing power for temporal tasks, while they do not store the combined database.

Previous study has shown that it is impossible to implement a noninteractive protocol in the single server setting under the partially homomorphic encryption scheme [34]. So at least two servers are required to complete the outsourced computation [35]. In design of the system model, we take into account the situation that there are usually a large number of servers in one CSP. Moreover, it is feasible to propose secure outsourcing protocols through the cooperation between cloud servers from different CSPs. $\forall i \in [1, n]$, DO_i generates its own key pair $\mathrm{pk}_i/\mathrm{sk}_i$ using the parameters produced by KA and encrypts D_i with pk_i before uploading it to EW. With the joint datasets as inputs, the distributed cloud servers are scheduled to perform k-means clustering algorithm in a privacy-preserving manner.

4.2. Threat Model. In our threat model, all cloud servers and clients are assumed to be semihonest, which means that they strictly follow the prescribed protocol but try to infer private information using the messages they receive during the protocol execution. This assumption is consistent with existing works [11–16] on privacy-preserving clustering in cloud environment. Furthermore, the cloud servers have some prior knowledge regarding distribution of owners' datasets that may be used to launch inference attacks by analyzing access patterns [19]. DOs, QCs, EWs, AWs, and KA are also interested in learning plain data belonging to other parities. Therefore, a CPA adversary $\mathscr{A}$ is introduced in the threat model. The target of $\mathscr{A}$ is to decrypt the ciphertexts from the challenge DO and challenge QC with the following abilities:

(i) $\mathscr{A}$ may compromise all the EWs to guess the plaintexts of received ciphertexts from DOs and AWs during the execution of the protocol.

(ii) $\mathscr{A}$ may compromise all the AWs and KA to guess the plaintext values of ciphertexts sent from EWs during the protocol interactions.

(iii) $\mathscr{A}$ may compromise one or more DOs and QCs except the challenge DO and the challenge QC to decrypt the ciphertexts belonging to the challenge party.

Nevertheless, we assume that the adversary $\mathscr{A}$ cannot compromise EWs and AWs and KA simultaneously; otherwise, $\mathscr{A}$ is able to decrypt any ciphertext stored on Storage and Computation Provider with the keys from Cryptographic Service Provider. In other words, there is no collusion between these two cloud providers, whereas servers within the provider itself may collude with each other. We remark that such assumptions are typical in adversary models used in cryptographic protocols (e.g., [15, 28, 36]), in that cloud providers are mostly competitors and economically driven by different business models.

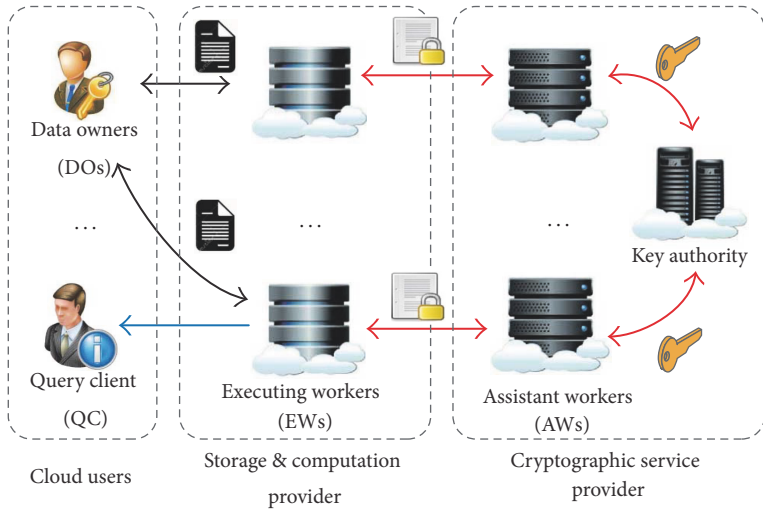


Figure 1: System model.

4.3. Design Objectives. Given the aforementioned system model and threat model, our design should achieve the following objectives:

(i) *Correctness*: if the cloud users and servers both follow the protocol, the final decrypted result should be the same as in the standard k-means algorithm.

(ii) *Data confidentiality*: nothing regarding the contents of datasets $D_1, \ldots, D_n$ and cluster centers $\mu_1, \ldots, \mu_k$, as well as the size of each cluster, should be revealed to the semihonest cloud servers.

(iii) *Access pattern hidden*: access patterns of clustering process, such as which records are assigned to which clusters, should not be revealed to the cloud to prevent any inference attacks [37].

(iv) *Efficiency*: most computation should be processed by cloud in an efficient way, while DOs and QCs are not required to be involved in the outsourced clustering.

5. The PPCOM Solution

In this section, we first discuss a set of privacy-preserving building blocks. Then the complete protocol of PPCOM is presented.

Recall that, in Section 3.1, the semihonest but noncolluding cloud servers need to cooperate to perform computation over encrypted data under PKC-DD scheme. At first, KA takes a security parameter κ as input and generates public parameter (N, g) and master secret key msk by executing $\mathsf{KeyGen}(\kappa)$. Also, KA generates a key pair $\mathrm{pk}_u/\mathrm{sk}_u$ used to unify ciphertext encryption key. After that, $\mathrm{pk}_u/\mathrm{sk}_u$ and msk are sent to AWs, while (N, g) is distributed to DOs and QC, which are used to produce their own key pair $\mathrm{pk}_i/\mathrm{sk}_i$ for $i \in [1, n]$. Their generated public keys are sent back to KA for management. Hereafter, let $\mathsf{Enc}_{\mathrm{pk}}(\cdot)$ denote the underlying encryption and $\mathsf{uDec}_{\mathrm{sk}}(\cdot)$ and $\mathsf{mDec}_{\mathrm{sk}}(\cdot)$ denote user-side decryption and master-side decryption, respectively. $|x|$ represents the bit length of x.

5.1. Privacy-Preserving Building Blocks. We present eight privacy-preserving building blocks under multikeys. Five of them aim at solving basic operations over ciphertexts, including ciphertext transformation, multiplication, addition, equality test, and minimum, while the rest are especially designed for outsourced clustering.

It can be apparently observed that the underlying encryption scheme is multiplicatively homomorphic due to the following equation:

$$\begin{aligned}&(g^{r_1}, \mathrm{pk}^{r_1} \cdot m_1) \times (g^{r_2}, \mathrm{pk}^{r_2} \cdot m_2)\\ &\quad = (g^{r_1+r_2}, \mathrm{pk}^{r_1+r_2} \cdot m_1 \cdot m_2) = \mathsf{Enc}_{\mathrm{pk}}(m_1 \cdot m_2),\end{aligned} \tag{1}$$

where $\mathsf{Enc}_{\mathrm{pk}}(m_i) = (g^{r_i}, \mathrm{pk}^{r_i} \cdot m_i)$, for $i = 1, 2$. This property is so critical that multiplication over ciphertexts can be evaluated by one EW server independently, as long as the encryptions are under the same public key. Hereafter, "$\times$" denotes multiplication operation in the encrypted domain while "$\cdot$" represents multiplication in the plaintext domain.

5.1.1. Secure Ciphertext Transformation (SCT) Protocol. Given that EW holds $\mathsf{Enc}_{\mathrm{pk}_x}(m)$, and AW holds $(\mathrm{msk}, \mathrm{pk}_y)$, the goal of the **SCT** protocol is to transform encrypted message m under public key pk_x into another ciphertext under public key pk_y. During execution of **SCT**, the plaintext m should not be revealed to EW or AW; meanwhile the output $\mathsf{Enc}_{\mathrm{pk}_y}(m)$ is only known to EW. The complete steps are shown in Algorithm 1.

To start with, EW generates a number $r \in_R \mathbb{Z}_N$, which means that r is randomly picked in $\mathbb{Z}_N$. Note that $|r| < |\kappa - 1|$ ensures that r is invertible in $\mathbb{Z}_N$ due to $\mathrm{GCD}(r, N) = 1$. Then we exploit multiplicative homomorphic property of PKC-DD to blind m from AW, even if it is able to decrypt $\mathsf{Enc}_{\mathrm{pk}_x}(r \cdot m)$ via using msk. Finally, EW removes the randomness by multiplying the encrypted inverse of r due to $\mathsf{Enc}_{\mathrm{pk}_y}(m) = \mathsf{Enc}_{\mathrm{pk}_y}(r \cdot m \cdot r^{-1} \bmod N)$. **SCT** protocol is especially useful for converting data under different encryption keys into ones

Require: EW has $\mathsf{Enc}_{\mathrm{pk}_x}(m)$, pk_x, and pk_y; AW has msk, pk_x, and pk_y.
(1) EW:
  (a) Generate a random number: $r \in_R \mathbb{Z}_N$, s.t., $|r| < |\kappa - 1|$;
  (b) Compute $\mathsf{Enc}_{\mathrm{pk}_x}(r \cdot m) \leftarrow \mathsf{Enc}_{\mathrm{pk}_x}(m) \times \mathsf{Enc}_{\mathrm{pk}_x}(r)$;
  (c) Send $\mathsf{Enc}_{\mathrm{pk}_x}(r \cdot m)$ to AW;
(2) AW:
  (a) Decrypt $r \cdot m \leftarrow \mathsf{mDec}(\mathrm{msk}, \mathrm{pk}_x, \mathsf{Enc}_{\mathrm{pk}_x}(r \cdot m))$;
  (b) Encrypt $\mathsf{Enc}_{\mathrm{pk}_y}(r \cdot m) \leftarrow \mathsf{Enc}(\mathrm{pk}_y, r \cdot m)$;
  (c) Send $\mathsf{Enc}_{\mathrm{pk}_y}(r \cdot m)$ to EW;
(3) EW:
  (a) Compute $\mathsf{Enc}_{\mathrm{pk}_y}(m) \leftarrow \mathsf{Enc}_{\mathrm{pk}_y}(r \cdot m) \times \mathsf{Enc}_{\mathrm{pk}_y}(r^{-1})$;

Algorithm 1: $\mathrm{SCT}(\mathsf{Enc}_{\mathrm{pk}_x}(m), \mathrm{pk}_y) \rightarrow \mathsf{Enc}_{\mathrm{pk}_y}(m)$.

under the same key so that EW can perform homomorphic operation.

5.1.2. Secure Addition (SA) Protocol. It takes $\mathsf{Enc}_{\mathrm{pk}_u}(m_1)$ and $\mathsf{Enc}_{\mathrm{pk}_u}(m_2)$ held by EW and sk_u held by AW as inputs. The output is the encrypted addition of m_1 and m_2, that is, $\mathsf{Enc}_{\mathrm{pk}_u}(m_1 + m_2)$, which is only known to EW. As the encryption scheme is not additively homomorphic, it requires interactions between EW and AW.

To preclude AW from obtaining m_1 and m_2, a straightforward solution for EW is to blind the inputs with a random value r by multiplying $\mathsf{Enc}_{\mathrm{pk}_u}(r)$, where $r \in_R \mathbb{Z}_N$ and $\mathrm{GCD}(r, N) = 1$. Then the encrypted randomized data are sent to AW. Since AW holds the secret key, it is able to get $r \cdot m_1$, $r \cdot m_2$ through decryption. The randomized addition (denoted by ω) is computed by $r \cdot m_1 + r \cdot m_2 \bmod N$. After that, AW encrypts ω and sends it back to EW who can get the desired output by running $\mathsf{Enc}_{\mathrm{pk}_u}(\omega) \times \mathsf{Enc}_{\mathrm{pk}_u}(r^{-1})$. Nevertheless, it is very possible that partial privacy is revealed to AW. This is because the ratio of inputs can be calculated via $m_1/m_2 \leftarrow m_1 \cdot r/m_2 \cdot r$, which may be utilized to distinguish inputs. As our threat model assumes the semihonest servers have some background knowledge of dataset distribution, it is effortless for AW to find correlations between encrypted records and known samples. Therefore, to achieve the privacy-preserving guarantees, disclosing input ratio should be prohibited during **SA** execution.

We propose an enhanced **SA** protocol still under two-server model, which protects confidentiality of inputs and outputs as well as intermediate results. There are five steps in **SA**, the details of which are presented in Algorithm 2.

During Step (1), the cloud EW generates a set of random numbers, namely, $\{r_i \in \mathbb{Z}_N, \rho_j \in \mathbb{Z}_N \mid i = 1, \ldots, 3, j = 1, \ldots, 4\}$, and calculates their corresponding encryptions under pk_u. By exploiting (1), EW computes several intermediate results, such as S_1, S_2, H_1, H_2, and outputs $\alpha_1, \ldots, \alpha_6$. It can be easily verified that $S_1 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^3)$, $S_2 = \mathsf{Enc}_{\mathrm{pk}_u}(m_2^3)$, $H_1 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^2 \cdot m_2)$, $H_2 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1 \cdot m_2^2)$, $\alpha_1 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^3 \cdot r_1)$, $\alpha_2 = \mathsf{Enc}_{\mathrm{pk}_u}(m_2^3 \cdot r_2)$, $\alpha_3 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^2 \cdot m_2 \cdot \rho_1 \cdot r_1)$, $\alpha_4 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1 \cdot m_2^2 \cdot \rho_3 \cdot r_2)$, $\alpha_5 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^2 \cdot m_2 \cdot \rho_2 \cdot r_3)$, $\alpha_6 = \mathsf{Enc}_{\mathrm{pk}_u}(m_1 \cdot m_2^2 \cdot \rho_4 \cdot r_3)$.

During Step (2), AW decrypts α_i using sk_u, for $1 \le i \le 6$. It then computes additions: $\beta_1, \beta_2, \beta_3$ and their corresponding encryptions. It can be verified that $\beta_1' = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^3 \cdot r_1 + m_1^2 \cdot m_2 \cdot \rho_1 \cdot r_1)$, $\beta_2' = \mathsf{Enc}_{\mathrm{pk}_u}(m_2^3 \cdot r_2 + m_1 \cdot m_2^2 \cdot \rho_3 \cdot r_2)$, $\beta_3' = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^2 \cdot m_2 \cdot \rho_2 \cdot r_3 + m_1 \cdot m_2^2 \cdot \rho_4 \cdot r_3)$.

During Step (3), the blinding factors in $\beta_1', \beta_2', \beta_3'$ are removed by multiplying the encrypted values of $r_1^{-1}, r_2^{-1}, r_3^{-1}$. Then, EW generates a random number r_4, which is used to blind K_1, K_2, K_3. At Step (4), AW decrypts K_1', K_2', K_3' as L_1, L_2, L_3, and calculates their addition γ. It can be verified that $\gamma' = \mathsf{Enc}_{\mathrm{pk}_u}(m_1^3 \cdot r_4 + m_1^2 \cdot m_2 \cdot \rho_1 \cdot r_4 + m_2^3 \cdot r_4 + m_1 \cdot m_2^2 \cdot \rho_3 \cdot r_4 + m_1^2 \cdot m_2 \cdot \rho_2 \cdot r_4 + m_1 \cdot m_2^2 \cdot \rho_4 \cdot r_4)$.

In the end of Step (5), EW gets rid of randomness by $\chi \leftarrow \gamma' \times \mathsf{Enc}_{\mathrm{pk}_u}(r_4^{-1})$. So we can verify that

$$\chi = \gamma' \times \mathsf{Enc}_{\mathrm{pk}_u}\left(r_4^{-1}\right) = \mathsf{Enc}_{\mathrm{pk}_u}\left(m_1^3 + \rho_1 \cdot m_1^2 \cdot m_2 + \rho_2 \cdot m_1^2 \cdot m_2 + \rho_3 \cdot m_1 \cdot m_2^2 + \rho_4 \cdot m_1 \cdot m_2^2 + m_2^3\right). \tag{2}$$

After that, EW computes the inverse of 3 modulo $\mathbf{ord}(\mathbb{G})$. Note that $\mathbf{ord}(\mathbb{G})$ can be obtained due to $\phi(N) = p(p-1)(q-1)$. We have $\mathrm{GCD}(3, \phi(N)) = 1$, if p and q are primes of the forms $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes, which means that 3^{-1} exists in the degree domain. The desired output is calculated by $\mathsf{Enc}_{\mathrm{pk}_u}(m_1 + m_2) \leftarrow \chi^{3^{-1}}$. The correctness of SA protocol can be proven by the following equation:

$$\begin{aligned}
\chi^{3^{-1}} &= \mathsf{Enc}_{\mathrm{pk}_u}\left(m_1^3 + \rho_1 \cdot m_1^2 \cdot m_2 + \rho_2 \cdot m_1^2 \cdot m_2 + \rho_3 \cdot m_1 \cdot m_2^2 + \rho_4 \cdot m_1 \cdot m_2^2 + m_2^3\right)^{3^{-1}} = \mathsf{Enc}_{\mathrm{pk}_u}\left(m_1^3 + (\rho_1 + \rho_2) \cdot m_1^2 \cdot m_2 + (\rho_3 + \rho_4) \cdot m_1 \cdot m_2^2 + m_2^3\right)^{3^{-1}} \\
&= \mathsf{Enc}_{\mathrm{pk}_u}\left(m_1^3 + 3 \cdot m_1^2 \cdot m_2 + 3 \cdot m_1 \cdot m_2^2 + m_2^3\right)^{3^{-1}} \\
&= \mathsf{Enc}_{\mathrm{pk}_u}\left((m_1 + m_2)^3\right)^{3^{-1}} = \mathsf{Enc}_{\mathrm{pk}_u}(m_1 + m_2).
\end{aligned} \tag{3}$$

Executing Algorithm 2 requires two rounds of interactions between EW and AW, which incurs more computational

Require: EW has $\mathsf{Enc}_{\mathrm{pk}_u}(m_1)$ and $\mathsf{Enc}_{\mathrm{pk}_u}(m_2)$; AW has the secret key sk_u.

(1) EW:

(a) Generate random numbers: $r_1, r_2, r_3 \in_R \mathbb{Z}_N$, s.t., $|r_i| < |\kappa - 1|$ for $1 \le i \le 3$;

(b) Compute $S_1 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(m_1)^3$, and $\alpha_1 \leftarrow S_1 \times \mathsf{Enc}_{\mathrm{pk}_u}(r_1)$;

(c) Compute $S_2 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(m_2)^3$, and $\alpha_2 \leftarrow S_2 \times \mathsf{Enc}_{\mathrm{pk}_u}(r_2)$;

(d) Generate random numbers: $\rho_1, \rho_2, \rho_3, \rho_4 \in_R \mathbb{Z}$, s.t., $\rho_1 + \rho_2 \equiv 3 \bmod N$, $\rho_3 + \rho_4 \equiv 3 \bmod N$;

(e) Compute $H_1 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(m_1)^2 \times \mathsf{Enc}_{\mathrm{pk}_u}(m_2)$, and $\alpha_3 \leftarrow H_1 \times \mathsf{Enc}_{\mathrm{pk}_u}(\rho_1) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_1)$;

(f) Compute $H_2 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(m_1) \times \mathsf{Enc}_{\mathrm{pk}_u}(m_2)^2$, and $\alpha_4 \leftarrow H_2 \times \mathsf{Enc}_{\mathrm{pk}_u}(\rho_3) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_2)$;

(g) Compute $\alpha_5 \leftarrow H_1 \times \mathsf{Enc}_{\mathrm{pk}_u}(\rho_2) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_3)$, and $\alpha_6 \leftarrow H_2 \times \mathsf{Enc}_{\mathrm{pk}_u}(\rho_4) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_3)$;

(h) Send $\{\alpha_i \mid i = 1, \ldots, 6\}$ to AW;

(2) AW:

(a) Receive $\{\alpha_i \mid i = 1, \ldots, 6\}$ from AW;

(b) Decrypt $F_i \leftarrow \mathsf{uDec}(\mathrm{sk}_u, \alpha_i)$, for $1 \le i \le 6$;

(c) Compute $\beta_1 \leftarrow F_1 + F_3 \bmod N$, $\beta_2 \leftarrow F_2 + F_4 \bmod N$, and $\beta_3 \leftarrow F_5 + F_6 \bmod N$;

(d) Encrypt $\beta_i' \leftarrow \mathsf{Enc}(\mathrm{pk}_u, \beta_i)$, for $1 \le i \le 3$;

(e) Send $\{\beta_1', \beta_2', \beta_3'\}$ to EW;

(3) EW:

(a) Receive $\{\beta_1', \beta_2', \beta_3'\}$ from EW;

(b) Generate a random number: $r_4 \in_R \mathbb{Z}_N$, s.t., $|r_4| < |\kappa - 1|$;

(c) **for** $i = 1$ to 3 **do**

(i) Compute $K_i \leftarrow \beta_i' \times \mathsf{Enc}_{\mathrm{pk}_u}(r_i^{-1})$;

(ii) Compute $K_i' \leftarrow K_i \times \mathsf{Enc}_{\mathrm{pk}_u}(r_4)$;

(d) Send $\{K_1', K_2', K_3'\}$ to AW;

(4) AW:

(a) Receive $\{K_1', K_2', K_3'\}$ from AW;

(b) Decrypt $L_i \leftarrow \mathsf{uDec}(\mathrm{sk}_u, K_i')$, for $1 \le i \le 3$;

(c) Compute $\gamma \leftarrow L_1 + L_2 + L_3 \bmod N$;

(d) Encrypt $\gamma' \leftarrow \mathsf{Enc}(\mathrm{pk}_u, \gamma)$; Send γ' to EW;

(5) EW:

(a) Receive γ' from EW;

(b) Compute $\chi \leftarrow \gamma' \times \mathsf{Enc}_{\mathrm{pk}_u}(r_4^{-1})$;

(c) Compute $d \leftarrow 3^{-1} \bmod \mathbf{ord}(\mathbb{G})$;

(d) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(m_1 + m_2) \leftarrow \chi^d \bmod N$;

Algorithm 2: $\mathrm{SA}(\mathsf{Enc}_{\mathrm{pk}_u}(m_1), \mathsf{Enc}_{\mathrm{pk}_u}(m_2)) \rightarrow \mathsf{Enc}_{\mathrm{pk}_u}(m_1 + m_2)$.

and communication overhead than the simple solution. However, it reveals no privacy to both cloud servers. The formal security analysis of **SA** is given in Section 6.

5.1.3. Secure Equality Test (SET) Protocol. Given that EW holds two encrypted values $\mathsf{Enc}_{\mathrm{pk}_u}(m_1)$ and $\mathsf{Enc}_{\mathrm{pk}_u}(m_2)$ while AW holds the secret key sk_u, the goal of **SET** is to test whether m_1 and m_2 are equal without revealing them to cloud servers. The detailed steps are presented in Algorithm 3.

At the first step, EW computes the fractions of two input ciphertexts. Supposing that $(A_1, B_1) = (g^{r_1}, m_1 \cdot \mathrm{pk}^{r_1})$ and $(A_2, B_2) = (g^{r_2}, m_2 \cdot \mathrm{pk}^{r_2})$, it can be verified that $\xi' = (g^{r\cdot(r_1-r_2)}, (m_1 \cdot m_2^{-1})^r \cdot \mathrm{pk}^{r\cdot(r_1-r_2)})$, where $r, r_1, r_2 \in_R \mathbb{Z}_N$. During Step (2), AW decrypts ξ' using sk_u as follows:

$$\begin{aligned}\zeta &= \frac{\left(m_1 \cdot m_2^{-1}\right)^r \cdot \mathrm{pk}^{r\cdot(r_1-r_2)}}{g^{\mathrm{sk}_u \cdot r\cdot(r_1-r_2)}} \bmod N \\ &= \left(m_1 \cdot m_2^{-1}\right)^r \bmod N.\end{aligned} \tag{4}$$

Since r is randomly selected in $\mathbb{Z}_N$, ζ is a random value if and only if $m_1 \neq m_2$. For $m_1 = m_2$, it is obvious to infer that $\zeta = 1$. Thus, if AW obtains $\zeta = 1$, the returned value ρ is set to be true (denoted by T); otherwise, ρ is false (denoted by F). It is worth noting that neither m_1, m_2 nor the intermediate result m_1/m_2 is revealed to the cloud during execution of **SET**.

5.1.4. Secure Squared Euclidean Distance (SSED) Protocol. For k-means algorithm, we use squared Euclidean distance to measure the distance between the data record and cluster centroid, denoted by $\|t_i - \mu_j\|^2$, supposing that EW holds the ciphertext of ith data record t_i, and the ciphertext of jth cluster centroid μ_j, while AW holds the secret key sk_u, for $1 \le i \le L$ and $1 \le j \le k$.

Note that μ_j is a vector composed of m attributes which may be rational numbers. However, the ring $\mathbb{Z}_N$ does not support rational division operation, so a new form of expression is required to represent the cluster center. Let $\langle s_j, \overline{c_j} \rangle$ denote the new form of cluster center, where s_j and $\overline{c_j}$ represent the

Require: EW holds two encrypted values $\mathsf{Enc}_{\mathrm{pk}_u}(m_1)$ and $\mathsf{Enc}_{\mathrm{pk}_u}(m_2)$; AW holds the secret key sk_u.
(1) EW:
  (a) Compute $A' \leftarrow A_1 \cdot A_2^{-1} \bmod N$, $B' \leftarrow B_1 \cdot B_2^{-1} \bmod N$, where $\mathsf{Enc}_{\mathrm{pk}_u}(m_1) = (A_1, B_1)$, $\mathsf{Enc}_{\mathrm{pk}_u}(m_2) = (A_2, B_2)$;
  (b) Generate a random number $r \in_R \mathbb{Z}_N$ that $r \neq 0$;
  (c) Compute $\xi' \leftarrow \xi^r$, where $\xi = (A', B')$; Send ξ' to AW;
(2) AW:
  (a) Decrypt $\zeta \leftarrow \mathsf{Dec}(\mathrm{sk}_u, \xi')$;
  (b) **if** $\zeta == 1$ **then** $\rho \leftarrow T$; **else** $\rho \leftarrow F$;
(3) **return** ρ;

Algorithm 3: $\mathrm{SET}(\mathsf{Enc}_{\mathrm{pk}_u}(m_1), \mathsf{Enc}_{\mathrm{pk}_u}(m_2)) \rightarrow T$ or F.

sum vector and the total number of records belonging to c_j, respectively. It is easily observed that $s_j = \Sigma_{h=1}^{L}(V_{h,j} \cdot t_h) \in \mathbb{Z}_N^m$ and $\overline{c_j} = \Sigma_{h=1}^{L} V_{h,j} \in \mathbb{Z}_N$. Furthermore, $\Omega_{i,j}$ is defined as the scaled squared distance between t_i and μ_j, which satisfies $\|t_i - \mu_j\| = \sqrt{\Omega_{i,j}}/\overline{c_j}$. Thus, $\Omega_{i,j}$ can be calculated as follows:

$$\begin{aligned}\Omega_{i,j} &= \left(\left\|t_i - \mu_j\right\| \cdot \overline{c_j}\right)^2 \\ &= \left(\sqrt{\sum_{h=1}^{m}\left(t_i[h] - \frac{s_j[h]}{\left|c_j\right|}\right)^2} \cdot \overline{c_j}\right)^2 \\ &= \sum_{h=1}^{m}\left(\overline{c_j} \cdot t_i[h] - s_j[h]\right)^2,\end{aligned} \tag{5}$$

where $i \in [1, L]$, $j \in [1, k]$, and m is the dimension size.

Taking encrypted record $[t_i]$ and encrypted center $[\mu_j]$ as inputs, EW and AW jointly execute **SSED** by invoking **SA** and output encryption of squared Euclidean distance $[d(i, j)]$, in which $[t_i] = \mathsf{Enc}_{\mathrm{pk}_u}(t_i)$, $[\mu_j] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(s_j), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_j})\rangle$, and $[d(i, j)] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,j}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_j})\rangle$. The **SSED** protocol should reveal neither the contents of t_i and μ_j nor the Euclidean distance $d(i, j)$ between them to cloud servers. Since implementation of **SSED** is straightforward by **SA** scheme, the design details are omitted.

5.1.5. Secure Squared Distance Comparison (SSDC) Protocol. Suppose that EW holds $[d(i, a), d(i, b)]$ and AW holds sk_u, where $i \in [1, L]$, $a, b \in [1, k]$, $a \neq b$. Apart from these, EW also has encrypted secrets associated with the distances, that is, s_a', s_b'. The output of **SSDC** is the encrypted minimum squared distance and its corresponding secret. Since our encryption scheme is probabilistic and does not preserve the order of messages, EW and AW should jointly compute the minimum without revealing $\langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_a})\rangle$, $\langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,b}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_b})\rangle$, and s_a, as well as s_b, to both servers.

Our key idea is to compute the fraction value between the two squared Euclidean distances, based on which AW is able to judge its relationship and returns EW as an encrypted identifier that indicates the minimum value. The fraction between the two squared Euclidean distances can be calculated as follows:

$$\frac{\left\|t_i - \mu_a\right\|^2}{\left\|t_i - \mu_b\right\|^2} = \frac{\Omega_{i,a}}{\overline{c_a}^2} \cdot \left(\frac{\Omega_{i,b}}{\overline{c_b}^2}\right)^{-1} = \frac{\Omega_{i,a} \cdot \overline{c_a}^2}{\Omega_{i,b} \cdot \overline{c_a}^2} = \lambda. \tag{6}$$

Since $\Omega_{i,a} \cdot \overline{c_b}^2$ and $\Omega_{i,b} \cdot \overline{c_a}^2$ are integers within $\mathbb{Z}_N$, the ratio λ may be a rational value in $\mathbb{Q}$, according to (6). It can be observed that if $\lfloor \lambda \rfloor < 1$ ($\lfloor \cdot \rfloor$ truncates the decimal fraction while keeping the integer part), we deduce that $\|t_i - \mu_a\|^2 < \|t_i - \mu_b\|^2$; otherwise, $\|t_i - \mu_a\|^2 \geq \|t_i - \mu_b\|^2$. The overall steps of **SSDC** are given in Algorithm 4.

To begin with, EW computes encryptions of randomized distances, that is, τ_1, τ_2. It can be verified that $\tau_1 = \mathsf{Enc}_{\mathrm{pk}_u}((r_1 \cdot \Omega_{i,a} \cdot \overline{c_b})^{r_2})$ and $\tau_2 = \mathsf{Enc}_{\mathrm{pk}_u}((r_1 \cdot \Omega_{i,b} \cdot \overline{c_a})^{r_2})$. During Step (2), AW obtains η_1, η_2 through decrypting τ_1, τ_2 with sk_u. Next, it calculates η_1/η_2 in the rational field.

$$\frac{\eta_1}{\eta_2} = \frac{\left(r_1 \cdot \Omega_{i,a} \cdot \overline{c_b}^2\right)^{r_2}}{\left(r_1 \cdot \Omega_{i,b} \cdot \overline{c_a}^2\right)^{r_2}} = \left(\frac{\Omega_{i,a} \cdot \overline{c_b}^2}{\Omega_{i,b} \cdot \overline{c_a}^2}\right)^{r_2} = (\lambda)^{r_2}. \tag{7}$$

From (7), it is easy to infer that AW locates the minimum value by $\delta = (\lambda)^{r_2}$, for the blinding factor r_2 does not alter the relationship between $\|t_i - \mu_a\|^2$ and $\|t_i - \mu_b\|^2$. If $\lfloor \delta \rfloor = 0$, it means $\eta_1 < \eta_2$ and AW assigns the indicator σ' to be $\mathsf{Enc}_{\mathrm{pk}_u}(2)$; otherwise, $\eta_1 \geq \eta_2$ and $\sigma' = \mathsf{Enc}_{\mathrm{pk}_u}(1)$. One may prefer to use $\mathsf{Enc}_{\mathrm{pk}_u}(1)$ and $\mathsf{Enc}_{\mathrm{pk}_u}(0)$ to represent the indicator, which is more straightforward, whereas it is not secure to utilize encryption of "0", because $C_2 = 0 \cdot \mathrm{pk}^r \bmod N = 0$, in which $\mathsf{Enc}_{\mathrm{pk}_u}(0) = (C_1, C_2)$. In that case, EW can obviously infer that the encrypted message is zero by observing the ciphertext part.

During Step (3), EW takes the received σ and encrypted squared distances as well as secrets as inputs and computes the target minimum values. It invokes a secure subroutine called **ComputeMin**, as shown in Algorithm 5. The correctness of **SSDC** protocol is proven as follows. Let us take $\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min})$ as an example; if $s \bmod 2 = 1$, EW and AW jointly execute **ComputeMin** and get the following:

$$\begin{aligned}&\mathsf{Enc}_{\mathrm{pk}_u}\left(\Omega_{\min}\right) \\ &\quad = \mathsf{Enc}_{\mathrm{pk}_u}\left(\sigma \cdot \Omega_{i,a} + 2 \cdot \Omega_{i,b} - \Omega_{i,a} - \sigma \cdot \Omega_{i,b}\right) \\ &\quad = \mathsf{Enc}_{\mathrm{pk}_u}\left((\sigma - 1) \cdot \Omega_{i,a} + (2 - \sigma) \cdot \Omega_{i,b}\right).\end{aligned} \tag{8}$$

Require: EW has two encrypted squared distances along with their corresponding encrypted secrets, i.e., $([d(i,a)], s_a')$ $([d(i,b)], s_b')$, while AW holds secret key sk_u, where $[d(i,a)] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_a}) \rangle$, $[d(i,b)] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,b}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_b}) \rangle$, $s_a' = \mathsf{Enc}_{\mathrm{pk}_u}(s_a)$, $s_b' = \mathsf{Enc}_{\mathrm{pk}_u}(s_b)$.

(1) EW:
- (a) Generate a random number $s \in_R \mathbb{Z}$, two non-zero random numbers $r_1, r_2 \in_R \mathbb{Z}_N$;
- (b) Compute $E_1 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}) \times \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_b}) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_1)$;
- (c) Compute $E_2 \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}) \times \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_a}) \times \mathsf{Enc}_{\mathrm{pk}_u}(r_1)$;
- (d) **if** $s \bmod 2 == 1$ **then**
  - (i) Compute $\tau_1 \leftarrow E_1^{r_2} \bmod N$;
  - (ii) Compute $\tau_2 \leftarrow E_2^{r_2} \bmod N$;
- (e) **else**
  - (i) Compute $\tau_1 \leftarrow E_2^{r_2} \bmod N$;
  - (ii) Compute $\tau_2 \leftarrow E_1^{r_2} \bmod N$;
- (f) Send τ_1, τ_2 to AW;

(2) AW:
- (a) Decrypt $\eta_i \leftarrow \mathsf{uDec}(\mathrm{sk}_u, \tau_i)$, for $i = 1, 2$;
- (b) Compute $\delta \leftarrow \eta_1 / \eta_2$;
- (c) **if** $\lfloor \delta \rfloor == 0$ **then**
  - (i) Compute $\sigma \leftarrow 2$;
- (d) **else if** $\lfloor \delta \rfloor >= 1$ **then**
  - (i) Compute $\sigma \leftarrow 1$;
- (e) Encrypt $\sigma' \leftarrow \mathsf{Enc}(\mathrm{pk}_u, \sigma)$; Send σ' to EW;

(3) EW:
- (a) **if** $s \bmod 2 == 1$ **then**
  - (i) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}), \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,b}), \sigma')$;
  - (ii) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_{\min}}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_a}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_b}), \sigma')$;
  - (iii) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(s_{\min}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(s_a), \mathsf{Enc}_{\mathrm{pk}_u}(s_b), \sigma')$;
- (b) **else**
  - (i) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,b}), \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a}), \sigma')$;
  - (ii) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_{\min}}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_b}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_a}), \sigma')$;
  - (iii) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(s_{\min}) \leftarrow$ **ComputeMin**$(\mathsf{Enc}_{\mathrm{pk}_u}(s_b), \mathsf{Enc}_{\mathrm{pk}_u}(s_a), \sigma')$;

(4) **return** $([d_{\min}], s'_{\min}) \leftarrow (\langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min}), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_{\min}}) \rangle, \mathsf{Enc}_{\mathrm{pk}_u}(s_{\min}))$;

Algorithm 4: $\mathrm{SSDC}(\{[d(i,a)], s_a'\}, \{[d(i,b)], s_b'\}) \rightarrow \{[d_{\min}], s'_{\min}\}$.

Require: EW holds two encrypted values v', w', and an encrypted identifier φ', where $v' = \mathsf{Enc}_{\mathrm{pk}_u}(v)$, $w' = \mathsf{Enc}_{\mathrm{pk}_u}(w)$, $\varphi' = \mathsf{Enc}_{\mathrm{pk}_u}(\varphi)$; AW has sk_u;

(1) EW computes $\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi) \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(v) \times \mathsf{Enc}_{\mathrm{pk}_u}(\varphi)$;
(2) EW computes $\mathsf{Enc}_{\mathrm{pk}_u}(2 \cdot w) \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(2) \times \mathsf{Enc}_{\mathrm{pk}_u}(w)$;
(3) EW computes $\mathsf{Enc}_{\mathrm{pk}_u}(w \cdot \varphi) \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(w) \times \mathsf{Enc}_{\mathrm{pk}_u}(\varphi)$;
(4) EW and AW jointly compute $\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi + 2 \cdot w) \leftarrow \mathbf{SA}(\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi), \mathsf{Enc}_{\mathrm{pk}_u}(2 \cdot w))$;
(5) EW and AW jointly compute $\mathsf{Enc}_{\mathrm{pk}_u}(v + w \cdot \varphi) \leftarrow \mathbf{SA}(\mathsf{Enc}_{\mathrm{pk}_u}(v), \mathsf{Enc}_{\mathrm{pk}_u}(w \cdot \varphi))$;
(6) EW computes $\mathsf{Enc}_{\mathrm{pk}_u}(-v - \mathrm{w} \cdot \varphi) \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(v + w \cdot \varphi) \times \mathsf{Enc}_{\mathrm{pk}_u}(-1)$;
(7) EW and AW jointly compute $\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi + 2 \cdot w - v - w \cdot \varphi) \leftarrow \mathbf{SA}(\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi + 2 \cdot w), \mathsf{Enc}_{\mathrm{pk}_u}(-v - w \cdot \varphi))$;
(8) **return** $\min(u', v') \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi + 2 \cdot w - v - w \cdot \varphi)$;

Algorithm 5: $\mathrm{ComputeMin}(v', w', \varphi') \rightarrow \min(u', v')$.

Apparently, it can be observed that if $\sigma = 2$, then we have $\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min}) = \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,a})$; otherwise, $\mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{\min}) = \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,b})$. Likewise, $\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_{\min}})$ and $\mathsf{Enc}_{\mathrm{pk}_u}(s_{\min})$ are calculated in a similar way.

5.1.6. Secure Minimum among k Squared Distances (SMkSD) Protocol. We assume that EW holds a set of encrypted squared Euclidean distances $[d(i,1)], \ldots, [d(i,k)]$, where $[d(i,j)] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{i,j}), \overline{\mu_j} \rangle$, $i \in [1, L]$, $j \in [1, k]$, and AW holds the secret key sk_u. Besides, EW also has the encryptions of secrets corresponding to their distances, that is, $s_1', \ldots, s_k'$. The goal of **SMkSD** is to compute the encryption of the shortest squared distance along with its encrypted secret, denoted by $[d_{\min}], s'_{\min}$, respectively. To execute **SMkSD**, we compute the minimum by utilizing **SSDC** with two inputs each time in a sequential fashion. The computation

Require: EW holds encrypted index $\mathsf{Enc}_{\mathrm{pk}_u}(\nu)$; AW has secret key sk_u.
(1) EW:
  (a) **for** $i = 1$ to k **do**
    (i) Generate non-zero random numbers $\alpha_i, \beta \in_R \mathbb{Z}_N$;
    (ii) Compute $A_i \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(\alpha_i) \times \mathsf{Enc}(\mathrm{pk}_u, i)^{\beta}$;
    (iii) Compute $B_i \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(\alpha_i) \times \mathsf{Enc}_{\mathrm{pk}_u}(\nu)^{\beta}$;
  (b) Generate a random permutation function π;
  (c) Compute $\Gamma' \leftarrow \pi(\Gamma)$, where $\Gamma = \{\langle A_i, B_i\rangle \mid i = 1, \ldots, k\}$;
  (d) Send Γ' to AW;
(2) AW:
  (a) **for** $i = 1$ to k **do**
    (i) Decrypt $E_i \leftarrow \mathsf{Dec}(\mathrm{sk}_u, A_i')$, $F_i \leftarrow \mathsf{Dec}(\mathrm{sk}_u, B_i')$, where $\Gamma' = \{\langle A_i', B_i'\rangle \mid i = 1, \ldots, k\}$;
    (ii) Compute $\eta_i \leftarrow E_i/F_i$;
    (iii) **if** $\eta_i == 1$ **then**
      (A) Compute $W[i] \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(2)$;
    (iv) **else**
      (A) Compute $W[i] \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(1)$;
  (b) Send W to EW;
(3) EW:
  (a) Compute $\Lambda \leftarrow \pi^{-1}(W)$;

Algorithm 6: $\mathrm{SIBC}(\mathsf{Enc}_{\mathrm{pk}_u}(\nu)) \rightarrow \Lambda$.

complexity of this algorithm is $O(k)$. Also, it can be executed in a binary tree hierarchy like SMINn in [15], which takes at most $\lceil \log_2 k \rceil$ iterations.

5.1.7. Secure Index to Bitmap Conversion (SIBC) Protocol. Given that EW has the encrypted index of the closest cluster denoted by $\mathsf{Enc}_{\mathrm{pk}_u}(\nu)$ ($\nu \in [1, k]$), and AW holds the private key sk_u, the output of **SIBC** is a bitmap vector Λ composed of k encrypted elements. During execution of **SIBC**, neither the index nor the bitmap should be revealed to both servers. If $i = \nu \wedge i \in [1, k]$, **SIBC** denotes $\Lambda[i] = \mathsf{Enc}_{\mathrm{pk}_u}(2)$; otherwise, $\Lambda[i] = \mathsf{Enc}_{\mathrm{pk}_u}(1)$. This indicates that the position of $\mathsf{Enc}_{\mathrm{pk}_u}(2)$ in Λ is the index of the nearest cluster. The typical form of Λ is as follows:

$$\Lambda = \left\langle \overbrace{\mathsf{Enc}_{\mathrm{pk}_u}(1), \ldots, \mathsf{Enc}_{\mathrm{pk}_u}(1)}^{1 \rightarrow \nu - 1}, \overbrace{\mathsf{Enc}_{\mathrm{pk}_u}(2)}^{\nu}, \overbrace{\mathsf{Enc}_{\mathrm{pk}_u}(1), \ldots, \mathsf{Enc}_{\mathrm{pk}_u}(1)}^{\nu + 1 \rightarrow k} \right\rangle. \tag{9}$$

The complete steps are presented in Algorithm 6.

During Step (1), EW generates nonzero random numbers and uses them to blind i and ν, for $1 \le i \le k$. It can be verified that $A_i = \mathsf{Enc}_{\mathrm{pk}_u}(\alpha_i \cdot (i)^{\beta})$ and $B_i = \mathsf{Enc}_{\mathrm{pk}_u}(\alpha_i \cdot (\nu)^{\beta})$. For $i = 1, \ldots, k$, $\langle A_i, B_i\rangle$ constitutes an ordered set Γ, which is further permutated by a random permutation function π.

During Step (2), AW decrypts the permutated set Γ' and computes the fraction for each part. It is easy to infer the following equation:

$$\eta_j = \frac{E_j}{F_j} = \frac{\alpha_i \cdot i^{\beta}}{\alpha_i \cdot \nu^{\beta}} = \left(\frac{i}{\nu}\right)^{\beta}, \tag{10}$$

where $i, j \in [1, k]$ and i, j may not be the same. Regardless of β, if $i = \nu$, then we have $\eta_j = 1$; otherwise, η_j is a random number in $\mathbb{Q}$. Note that AW cannot know the relationship between E_i and E_j for $i \neq j$, since they are randomized by blinding factors. Besides, the ratio $E_i/E_j = (\alpha_i/\alpha_j) \cdot (i/j)^{\beta}$ is also a random number. Thus, as long as π is kept confidential, the index of closest cluster is not revealed to AW. In the end, EW recovers the true sequence of Λ by running the inverse permutation $\pi^{-1}(W)$.

Furthermore, it should be emphasized that the method based on (10) can also be used in other scenarios where equality test is required.

5.1.8. Secure New Cluster Computation (SNCC) Protocol. Given that EW holds the assignment membership matrix V, the encrypted dataset D', and target cluster c_i, the goal of **SNCC** is to calculate the new cluster centroid denoted by $[\mu_i]$, where $[\mu_i] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(s_i), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_i})\rangle$. During execution of **SNCC**, nothing regarding the data record, sum of attributes, and cluster size should be disclosed to cloud servers. The complete steps are shown in Algorithm 7.

During Step (1), EW computes the encryption of summation for each attribute, that is, $\mathsf{Enc}_{\mathrm{pk}_u}(s_{i,h})$ for $1 \le i \le k$ and $1 \le h \le m$. Supposing that $V[j, i] = \mathsf{Enc}_{\mathrm{pk}_u}(\nu_{j,i})$, where $\nu_{j,i} \in \{1, 2\}$ and $j \in [1, L]$, EW obtains $H_j = \mathsf{Enc}_{\mathrm{pk}_u}(t_{j,h} \cdot \nu_{j,i})$. After L rounds of iterations, the important observation is

$$\begin{aligned} \mathsf{Enc}_{\mathrm{pk}_u}(s_{i,h}) &= \mathsf{Enc}_{\mathrm{pk}_u}\left(\sum_{j=1}^{L} t_{j,h} \cdot \nu_{j,i} - \sum_{j=1}^{L} t_{j,h}\right) \\ &= \mathsf{Enc}_{\mathrm{pk}_u}\left(\sum_{j=1}^{L} t_{j,h} \cdot \left(\nu_{j,i} - 1\right)\right). \end{aligned} \tag{11}$$

Require: EW holds the assignment matrix V, the encrypted dataset D', the cluster id i, while AW has the secret key sk_u, where $V = \langle \Lambda_1^T, \ldots, \Lambda_L^T \rangle$, $D' = \{\mathsf{Enc}_{\mathrm{pk}_u}(t_{j,h}) \mid j \in [1, L], h \in [1, m]\}$, $i \in [1, k]$.

(1) EW:
  (a) **for** $h = 1$ to m **do**
    (i) **for** $j = 1$ to L **do**
      (A) Compute $H_j \leftarrow \mathsf{Enc}_{\mathrm{pk}_u}(t_{j,h}) \times V[j, i]$;
      (B) Compute $H_h' \leftarrow \mathbf{SA}(H_j, H_h')$ with AW;
    (ii) Compute $S_h' \leftarrow S_h \times \mathsf{Enc}_{\mathrm{pk}_u}(-1)$, where $S_h = \sum_{j=1}^{L} t_{j,h}$;
    (iii) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(s_{i,h}) \leftarrow \mathbf{SA}(H_h', S_h')$ with AW;
(2) EW:
  (a) **for** $j = 1$ to L **do**
    (i) Compute $C \leftarrow \mathbf{SA}(V[j, i], C)$ with AW;
  (b) Compute $\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_i}) \leftarrow \mathbf{SA}(C, \mathsf{Enc}_{\mathrm{pk}_u}(-L))$ with AW;
(3) **return** $[\mu_i] \leftarrow \langle \mathsf{Enc}_{\mathrm{pk}_u}(s_i), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_i}) \rangle$;

Algorithm 7: $\mathrm{SNCC}(V, D', i) \rightarrow [\mu_i]$.

Recall that $v_{j,i} = 2$ means that t_j belongs to the cluster c_i, while $v_{j,i} = 1$ denotes $t_j \notin c_j$. Therefore, (11) adds all the records that are assigned to the target cluster. Moreover, S_h only needs to be calculated only once during the entire outsourcing period, for $1 \le h \le m$.

During Step (2), EW and AW jointly compute the encryption of cluster size $\mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_i})$ by invoking **SA** subprotocol. It can be verified that

$$\begin{aligned} \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_i}) &= \mathsf{Enc}_{\mathrm{pk}_u}\left(\sum_{j=1}^{L} v_{j,i} - L\right) \\ &= \mathsf{Enc}_{\mathrm{pk}_u}\left(\sum_{j=1}^{L} (v_{j,i} - 1)\right). \end{aligned} \tag{12}$$

Obviously, (12) sums up those whose $v_{j,i}$ equals 2 and discards those with $v_{j,i} = 1$, through which the final result is the size of the cluster c_i.

5.2. The Complete PPCOM Protocol. In this subsection, we present our proposed PPCOM protocol for the standard k-means algorithm which works in the distributed cloud environments.

The primary goal of PPCOM is to schedule a group of cloud servers to perform clustering task over the joint datasets encrypted under multiple keys, meanwhile no privacy of data records, intermediate results, and final clusters should be revealed to the semihonest servers. In order to improve the overall performance, we leverage a large-scale data processing engine called Spark [38]. It develops a data structure called the resilient distributed dataset (RDD) for data parallelism and fault-tolerance, which facilitates iterative algorithms in machine learning. Though it provides a scalable machine learning library—MLlib—which includes k-means algorithm [39], it does not take privacy protection into account and cannot process encrypted data either. Therefore, it is essential to combine our proposed building blocks in Section 5.1 and Spark computing framework to design PPCOM.

PPCOM is composed of four stages, that is, Data Uploading, Ciphertext Transformation, Clustering Computation, and Result Retrieval, the details of which are described in the following.

5.2.1. Data Uploading Stage. We assume that all data records should have been preprocessed by data owners. One essential preprocessing step is to normalize data values, because different attributes have different value domains, which possibly leads to the case that those attribute values whose domain is large have greater impacts on accuracy of distance-based clustering. Normalization enables records to fall into the common range by endowing all attributes with equal weights. In this paper, we adopt Mix-Max Normalization [40]. Suppose that attribute A owns s observed values $v_1, v_2, \ldots, v_s$ and $[\min_A, \max_A]$ is the range of A, while $[\mathrm{new\_min}_A, \mathrm{new\_max}_A]$ is the target range. Mix-Max Normalization maps v_i into v_i' in $[\mathrm{new\_min}_A, \mathrm{new\_max}_A]$ by calculating the following equation:

$$\begin{aligned} v_i' = \frac{v_i - \min_A}{\max_A - \min_A} (\mathrm{new\_max}_A - \mathrm{new\_min}_A) \\ + \mathrm{new\_min}_A. \end{aligned} \tag{13}$$

After the preprocessing step, DO_i encrypts its dataset D_i with pk_i by calculating $\mathsf{Enc}(\mathrm{pk}_i, t_{j,h}^i)$ for $i \in [1, n]$. Let $t_{j,h}^i$ denote the hth attribute value of jth record t_j^i in D_i for $h \in [1, m]$ and $j \in [1, L]$. D_i' denotes the encryption of D_i. After all DOs complete uploading their datasets to EWs, the cloud aggregates the distributed datasets into a joint database $D' = \bigcup_{i=1}^{n} D_i'$. Under this circumstance, DO_i is still able to retrieve its data and decrypt it with its private key sk_i, whereas DO_i cannot decrypt D_j without corresponding sk_i for $i \neq j$.

5.2.2. Ciphertext Transformation Stage. Upon receiving clustering request from QC, EWs initiate ciphertext transformation procedure which aims at converting ciphertexts under pk_i into encryptions under the unified key pk_u, for $i \in [1, n]$. EW first replicates D' into D_r' to ensure DOs' accessibility to their original dataset. Then **SCT** subprotocol is executed

to output the converted dataset (denoted by D'_u). This stage is important for two reasons: (1) EW is able to perform homomorphic operation merely under the same public key; (2) user-decryption is much more efficient than master decryption.

5.2.3. Clustering Computation Stage. With all the converted records $\mathsf{Enc}_{\mathrm{pk}_u}(t_{i,j})$, for $i \in [1, L]$, $j \in [1, m]$, the objective of this stage is to compute the cluster centroids $[\mu_1], \ldots, [\mu_k]$ and the membership matrix $V_{L\times k}$ without compromising privacy. The outsourcing process is not only protected by the proposed secure building blocks, but also accelerated by Spark framework. The stage includes four steps, namely, Job Assignment, Map Execution, Reduce Execution, and Update Judgement, as shown in Figure 2.

Step 1 (Job Assignment). Firstly, CSPs select ς minimum computing units (denoted by MCUs). Each MCU is composed of one server from $\{\mathrm{EW}_1, \ldots, \mathrm{EW}_\theta\}$ and one from $\{\mathrm{AW}_1, \ldots, \mathrm{AW}_\vartheta\}$; that is, MCU $= \{\mathrm{EW}, \mathrm{AW}\}$. Obviously, MCU is able to perform cryptographic building blocks on its own. Since the workload of AW is relatively light compared to EW, it is preferable to share one AW within several MCUs to maximize resource usage. We assume that every cloud node possesses sufficient storage and computational power for its assigned job. The set $\{\mathrm{MCU}_1, \ldots, \mathrm{MCU}_\varsigma\}$ is divided into two disjoint sets, that is, Map set and Reduce set. Without loss of generality, Map consists of $\mathrm{MCU}_1, \ldots, \mathrm{MCU}_f$, while Reduce comprises $\mathrm{MCU}_{f+1}, \ldots, \mathrm{MCU}_{f+k+1}$. Then D'_u is divided into f uniformly distributed partitions $P_1, \ldots, P_f$, which are transferred to their corresponding MCU nodes in Map set. In this paper, we assume that the k initial cluster centers are chosen by DOs in advance and they are also encrypted. $\forall i \in [1, f]$, Map$[i]$ is aware of the initial cluster centroid set $U = \{[\mu_1], \ldots, [\mu_k]\}$, where $[\mu_j] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(s_j), \mathsf{Enc}_{\mathrm{pk}_u}(\overline{c_j})\rangle$, for $1 \le j \le k$.

Step 2 (Map Execution). Given P_i and U as inputs, Map$[i]$ ($i \in [1, f]$) outputs a key-value table T_i, in which the key is the record and the value is the encryption of bitmap that indicates the closest cluster. Suppose that P_i includes z data records $[t_1], \ldots, [t_z]$. Here, $[t_j]$ ($j \in [1, z]$) is an m-dimension vector $\langle \mathsf{Enc}_{\mathrm{pk}_u}(t_{j,1}), \ldots, \mathsf{Enc}_{\mathrm{pk}_u}(t_{j,m})\rangle$.

As presented in Algorithm 8, each MCU in Map executes the following steps in parallel: (1) computes the encryption of squared Euclidean distance between $[t_j]$ and cluster center $[\mu_h]$, for $1 \le j \le z$ and $1 \le h \le k$ (Steps (1)–(5)); (2) computes the encrypted index of the minimum among k distances for each record by calling **SMkSD** (Step (6)); (3) converts the index into an encrypted bitmap via **SIBC** scheme (Step (7)). The final output for Map$[i]$ is table T_i with z entries. Each entry consists of a data record as the key and its corresponding assignment bitmap as value.

Step 3 (Reduce Execution). Taking $\{T_1, \ldots, T_f\}$ from Map MCUs and the aggregated dataset D'_u as inputs, Reduce$[i]$ computes the cluster center for c_i based on the assignment membership matrix V. The major steps are presented in Algorithm 9. It can be observed that each MCU in Reduce concurrently executes the following: (1) converges the assignment vectors in $\{T_1, \ldots, T_f\}$ into the complete membership matrix V (Step (4)); (2) computes the new cluster center for the target cluster by invoking **SNCC** (Step (7)). The final output for Reduce$[i]$ is the encrypted centroid $[\mu_i]$ for c_i and the matrix V.

Step 4 (Update Judgement). This step is to determine whether the predefined termination condition of k-means algorithm is satisfied. In this paper, we consider that the membership matrix is not changing as the termination condition. Given that EW holds the previous matrix V and the current matrix V' and AW holds the key sk_u, our strategy is to find out whether the elements in V and V' are equal one by one by utilizing **SET** subprotocol. Once a mismatch is detected, EW replaces V with V' and goes to Step 2 of Clustering Computation Stage. Otherwise, these servers continue comparing till the end of matrix. If the outcome is $V = V'$ which means that the assignment of clusters does not vary any more, EW then terminates the iteration and activates Result Retrieval Stage.

5.2.4. Result Retrieval Stage. Since the cluster center set U and assignment matrix V are encrypted under pk_u, QC cannot decrypt them without sk_u. Firstly, **SCT** scheme is invoked to transform the encryption key of U and V from pk_u to pk_Q. After that, QC is able to download them and decrypt the final result with sk_Q. The final cluster center of c_i is recovered by $\mu_i \leftarrow s_i/\overline{c_i}$, for $i \in [1, k]$.

6. Security Analysis

We first analyze the security of the privacy-preserving building blocks. Since all parties are semihonest, security in this model can be proven under "Real-versus-Ideal" framework [41]: there is an ideal model where all computations are performed by a trusted third party. The protocol is secure if all adversarial behaviors in the real world can be simulated in the ideal world. Considering that the proofs of the proposed building blocks are basically the same, we just take the formal proof of **SA** subprotocol as an example.

Theorem 1. *The SA protocol described in Section 4.1 securely computes the addition over ciphertexts by using the PKC-DD cryptosystem in presence of two semihonest but noncolluding cloud servers.*

Proof. Since this algorithm is collaboratively completed by EW and AW, we need to prove that **SA** is not only secure against adversary $\mathscr{A}_{\mathrm{EW}}$ corrupting EW, but also against $\mathscr{A}_{\mathrm{AW}}$ corrupting AW, respectively.

(1) *Security against* $\mathscr{A}_{EW}$: in Step (1), the real world view of $\mathscr{A}_{\mathrm{EW}}$ in **SA** includes inputs $\{\mathsf{Enc}_{\mathrm{pk}_u}(m_1), \mathsf{Enc}_{\mathrm{pk}_u}(m_2)\}$, random values $\{r_1, r_2, r_3, \rho_1, \rho_2, \rho_3, \rho_4\}$, and outputs $\{\alpha_i, S_1, S_2, H_1, H_2 \mid 1 \le i \le 6\}$. In Step (2), the real world view includes inputs $\{\beta_1, \beta_2, \beta_3\}$, outputs $\{K'_1, K'_2, K'_3\}$, and intermediate results $\{K_1, K_2, K_3, r_4\}$. In Step (3), the real world view is composed of input γ', outputs $\{\chi, \chi^d\}$, and a fixed value d. Without the secret key sk_u, $\mathscr{A}_{\mathrm{EW}}$ is unable to

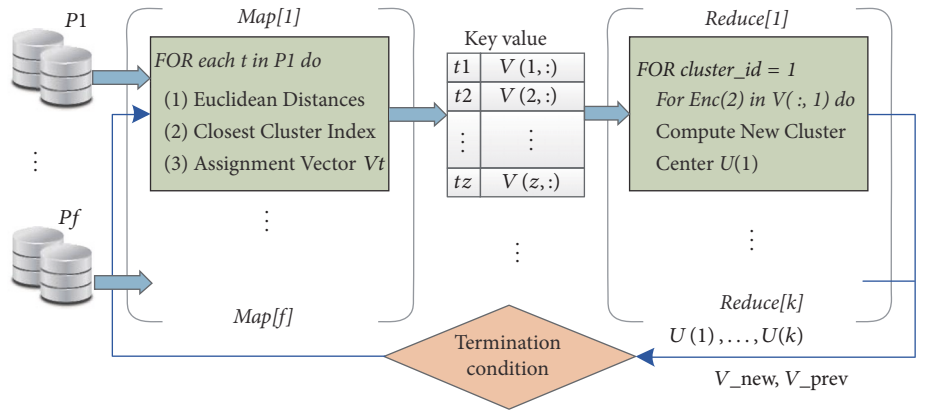


Figure 2: PPCOM under Spark framework.

Require: Map MCUs have datasets $P_1, \ldots, P_f$ and cluster centroid set U.
(1) $\forall i \in [1, f]$, Map[i] concurrently executes:
(2) **for** $j = 1$ to z **do**
(3) **for** $h = 1$ to k **do**
(4) {**Compute the squared Euclidean distance**}
$[d(j,h)] \leftarrow$ **SSED**$([t_j], [\mu_h])$, where $[d(j,h)] = \langle \mathsf{Enc}_{\mathrm{pk}_u}(\Omega_{j,h}), \overline{c_h} \rangle$;
(5) **end for**
(6) {**Compute the index of minimum among k distances**}
$\{[d_{\min}], s'_{\min}\} \leftarrow$ **SMkSD**$(\{[d(j,1)], s'_1\}, \ldots, \{[d(j,k)], s'_k\})$, where $s'_v = \mathsf{Enc}_{\mathrm{pk}_u}(v)$, for $v \in [1, k]$;
(7) {**Convert the index into encrypted vector**}
$\Lambda_j \leftarrow$ **SIBC**$(s'_{\min})$;
(8) $key_j \leftarrow [t_j]$ and $value_j \leftarrow \Lambda_j$;
(9) **end for**
(10) **return** $T_i = \{\langle key_1, value_1 \rangle, \ldots, \langle key_z, value_z \rangle\}$ for $i = 1, \ldots, f$;

Algorithm 8: Map$(P_1, \ldots, P_f, U) \rightarrow \{T_1, \ldots, T_f\}$.

Require: Reduce MCUs have $T_1, \ldots, T_f$ and the aggregated dataset D'_u.
(1) $\forall i \in [1, k]$, Reducer[i] concurrently executes:
(2) **for** $j = 1$ to f **do**
(3) **for** $h = 1$ to z **do**
(4) {**Converge the assignment vectors into matrix**}
$V[(j-1) \cdot z + h] \leftarrow T_j.value_h$;
(5) **end for**
(6) **end for**
(7) {**Compute the new cluster center for c_i**}
$[\mu_i] \leftarrow$ **SNCC**(V, D'_u, i);
(8) **return** U, V, where $U = \{[\mu_1], \ldots, [\mu_k]\}$;

Algorithm 9: Reduce$(T_1, \ldots, T_f, D'_u) \rightarrow \{U, V\}$.

decrypt those ciphertexts. Thus, we can build a simulator $\mathcal{S}_{\mathrm{EW}}$ in the ideal world. It generates encryptions of inputs $\{\widehat{m}_1, \widehat{m}_2, \widehat{\beta}_1, \widehat{\beta}_2, \widehat{\beta}_3, \widehat{\gamma}'\}$ and random numbers $\{\widehat{r}_i, \widehat{\rho}_i \mid i = 1, \ldots, 4\}$ by randomly selecting from $\mathbb{Z}_N$. Then, $\mathcal{S}_{\mathrm{EW}}$ follows the protocol to compute the corresponding outputs. Due to the semantic security of the PKC-DD, it is computationally difficult for $\mathcal{A}_{\mathrm{EW}}$ to distinguish the views from the real world and the ideal world. Therefore, we can prove that

$$\mathrm{Ideal}_{f,\mathcal{S}_{\mathrm{EW}}}\left(\mathsf{Enc}_{\mathrm{pk}_u}(m_i)\right) \overset{c}{\approx} \mathrm{Real}_{\mathrm{SA},\mathcal{A}_{\mathrm{EW}}}\left(\mathsf{Enc}_{\mathrm{pk}_u}(m_i)\right), \quad (14)$$

where $i \in \{1, 2\}$. Herein, $\overset{c}{\approx}$ means computationally distinguishable.

Table 1: Computational and communication costs of primary algorithms.

Algorithm	Computational cost	Communication cost (in bits)
SCT	8Exp + 9Mul	$4\,\|N\|$
SA	51Exp + 55Mul	$26\,\|N\|$
SET	3Exp + 3Mul	$2\,\|N\|$
SSDC	478Exp + 572Mul	$84\,\|N\|$
SIBC	$(8k+2)$Exp + $8k$Mul	$6k\,\|N\|$
Map	$kz(51m+486)$Exp + $kz(59m+580)$Mul	$kz\,(26m+90)\,\|N\|$
Reduce	$(102mL+51m+51L)$Exp + $(111mL+55m+55L)$Mul	$(52mL+26m+26L)\,\|N\|$
Update	$3mL$Exp + $3mL$Mul	$2mL\,\|N\|$

(2) *Security against* $\mathscr{A}_{AW}$: in Step (2), the real world view of $\mathscr{A}_{\mathrm{AW}}$ includes inputs $\{\alpha_i \mid i \le i \le 6\}$ and outputs $\{F_i, \beta_j, \beta_j' \mid 1 \le i \le 6, 1 \le j \le 3\}$. Note that the output values are blinded by random $r_1, \ldots, r_3, \rho_1, \ldots, \rho_4$. Besides, it is extremely hard for $\mathscr{A}_{\mathrm{AW}}$ to guess the correct ratio, that is, m_1/m_2. Suppose that $\rho_1 = 3 + k_1 N - \rho_2$ and $\rho_3 = 3 + k_2 N - \rho_4$, where $k_1, k_2 \in_R \mathbb{Z}$. There are 8 unknown factors $(r_1, \ldots, r_3, \rho_1, \ldots, \rho_4, m_1/m_2)$, whereas $\mathscr{A}_{\mathrm{AW}}$ can only set up 6 equations $(\alpha_1, \ldots, \alpha_6)$ that are not enough to crack m_1/m_2. As for Step (4), the real world view includes inputs $\{K_i' \mid 1 \le i \le 3\}$ and outputs $\{L_i, \gamma, \gamma' \mid 1 \le i \le 3\}$. Note that the inputs are randomized by r_4. Therefore, we can build a simulator $\mathscr{S}_{\mathrm{AW}}$ in the ideal world, which generates random numbers as inputs, that is, $\{\widehat{\alpha}_i, \widehat{K}_j' \in_R \mathbb{Z}_N \mid i \le i \le 6, 1 \le j \le 3\}$. $\mathscr{S}_{\mathrm{AW}}$ then executes the protocol to produce output $\{\widehat{\beta}_j', \widehat{\gamma}' \mid 1 \le i \le 3\}$ through $\mathsf{uDec}(\cdot)$, $\mathsf{Enc}(\cdot)$, and modular addition operations. Due to the semantic security of the PKC-DD and randomness of blinding factors, the views of $\mathscr{A}_{\mathrm{AW}}$ in the real world and ideal world are computationally distinguishable. Therefore, for $i \in \{1, 2\}$, we have

$$\mathrm{Ideal}_{f,\mathscr{S}_{\mathrm{AW}}}\left(\mathsf{Enc}_{\mathrm{pk}_u}\left(r_i m_i^3\right)\right) \overset{c}{\approx} \mathrm{Real}_{\mathrm{SA},\mathscr{A}_{\mathrm{AW}}}\left(\mathsf{Enc}_{\mathrm{pk}_u}\left(r_i m_i^3\right)\right). \tag{15}$$

□

During outsourcing process, cloud servers invoke the proposed building blocks as subroutines and all transmitted data are encrypted for each step. Note that the data records are held by cloud parties without decryption key. The assistant parties with the key can decrypt the received data, but the real data are randomized. Since PKC-DD is semantically secure and blinding factors are randomly selected, nothing regarding the data contents and computed clusters is revealed to the servers. Moreover, the access patterns of which encrypted input denotes the minimum Euclidean distance (as shown in **SSDC**, Algorithm 4) and of which record is assigned to which cluster are protected from the cloud due to encryption of V (as shown in **SNCC**, Algorithm 7). By the Composition Theorem [41], the sequential composition of four stages in PPCOM is secure under the semihonest model.

Discussions. Note that the order of computing addition via **SA** cannot be altered in **ComputeMin** called **SSDC** even though the final outcome is the same. Suppose that the clouds choose to compute $\mathsf{Enc}_{\mathrm{pk}_u}(v \cdot \varphi - v)$ or $\mathsf{Enc}_{\mathrm{pk}_u}(2 \cdot w - \varphi \cdot w)$; if the result is $\mathsf{Enc}_{\mathrm{pk}_u}(0)$, it will inevitably reveal $\varphi = 1$ or $\varphi = 2$ to both servers. Similarly, the order of computation steps in Algorithm 7 in **SNCC** ought to be kept unchanged.

7. Performance Analysis

In this section, we analyze the performance of PPCOM protocol from both theoretical and experimental perspectives.

7.1. Theoretical Analysis. Let Exp, Mul denote the modular exponentiation and multiplication operations, respectively. Let $|N|$ represent the key size of the double decryption scheme. The encryption of the underlying cryptosystem incurs 2Exp + 1Mul. The cost of normal decryption is 1Exp + 1Mul, while that of authority decryption is 2Exp+2Mul. Recall that m is the dimension size of a record, and L is size of the joint dataset. The computational and communication overheads for the major building blocks and clustering algorithms in one iteration are given in Table 1. It can be observed that the addition and comparing operations incur a lot of Exp operations at the cost of hiding access patterns. For Data Uploading Stage, it takes each DO $2lm$Exp + lmMul computational cost and $2lm|N|$ bits. The computational overhead of Ciphertext Transformation Stage is $8Lm$Exp + $9Lm$Mul while its communication cost is $4Lm|N|$. We stress that Stage 1 and Stage 2 of PPCOM protocol are executed only once. These overheads are amortized through a number of iterations. Furthermore, the number of MCU in Map set is closely related to the costs, while L affects the performance of Reduce more significantly (usually $L \gg m$). It is easy to find that the larger the computing cluster is, the less the tasks (z) are distributed to each unit. This is because the k-means jobs can be parallelized under Spark. As for the Update Judgement step, the worst case is to compare every element of the matrix, the computational and communication costs of which are related to L and m.

7.2. Experimental Analysis. The experiments are conducted on our local cluster, in which each server running CentOS6.5 has Intel Xeon E5-2620 @ 2.10 GHz with 12 GB memory. We implemented all the outsourcing protocols using the Crypto++ 5.6.3 library and Spark framework. In this paper,

Table 2: Performance of each subprotocol (1000 times for average).

Subprotocol	Computation	Communication
SA	40.851 ms	4.875 KB
SET	5.543 ms	0.375 KB
SSED ($m = 20$)	766.154 ms	88.860 KB
SSDC	267.815 ms	33.750 KB
SMkSD ($k = 4$)	1.107 s	134.997 KB
SIBC ($k = 4$)	49.509 ms	4.500 KB
SNCC ($L = 100$, $m = 20$, $k = 4$)	80.364 s	9.997 MB

key size of similar methods (Paillier used in PPODC [15] and BCP encryption in [28]) is 1024 bits, which is commonly acceptable. To achieve the same security level, $|N|$ of PKC-DD should be 500~600 bits more than RSA modulus [42]. During all tests, we choose the security parameter $\kappa = 512$, so the key size $|N| = 1536$.

To facilitate comparisons, we use KEGG Metabolic Reaction Network dataset [43]. The dataset includes 65554 instances and 29 attributes. Before clustering, all records are first normalized into integers in $[0, 1000]$ to prevent impacts of large unit values as mentioned in Data Uploading Stage of Section 5.2. Note that the first attribute is excluded from tests, since it is just the identifier of pathway. All of the testing records are randomly selected from KEGG dataset.

7.2.1. Privacy-Preserving Building Blocks' Performance. We first measure the execution time of each privacy-preserving building block on a single server through 1000 times, the average costs of which are shown in Table 2. It can be seen that costs of the compound protocol (made up of basic primitives, e.g., **SSED**, **SMkSD**, and **SNCC**) are relatively high, since they are made up of several **SA** operations, which involve many encryptions and decryptions, as well as rounds of interactions to preserve data privacy. This is consistent with theoretical analysis.

We then evaluate the performance of **SCT** scheme. Table 3 shows the ciphertext transformation time for varying dataset size (L) in **SCT** and **KeyProd** in [28]. It can be seen that the cloud running time grows with increasing value of L. Our scheme executes about 4 times faster than **KeyProd** in that PKC-DD works more efficiently than theirs. Besides, we remark that schemes in [28] are designed for basic arithmetic operations under multikey rather than complex mining tasks like k-means algorithm.

We also compare the cloud running time for **SSED** and **SMkSD** with counterpart methods in PPODC [15], respectively. As shown in Figures 3(a) and 3(b), computation time of both grows with increase of dataset size and our schemes outperform PPODC's. Let w denote the bit length of plaintext message. In Figure 3(b), it is easy to find that, with growth of w, the computation time of PPODC grows more rapidly. It is because ciphertexts have to be decomposed into encryption of bits before comparison in [15].

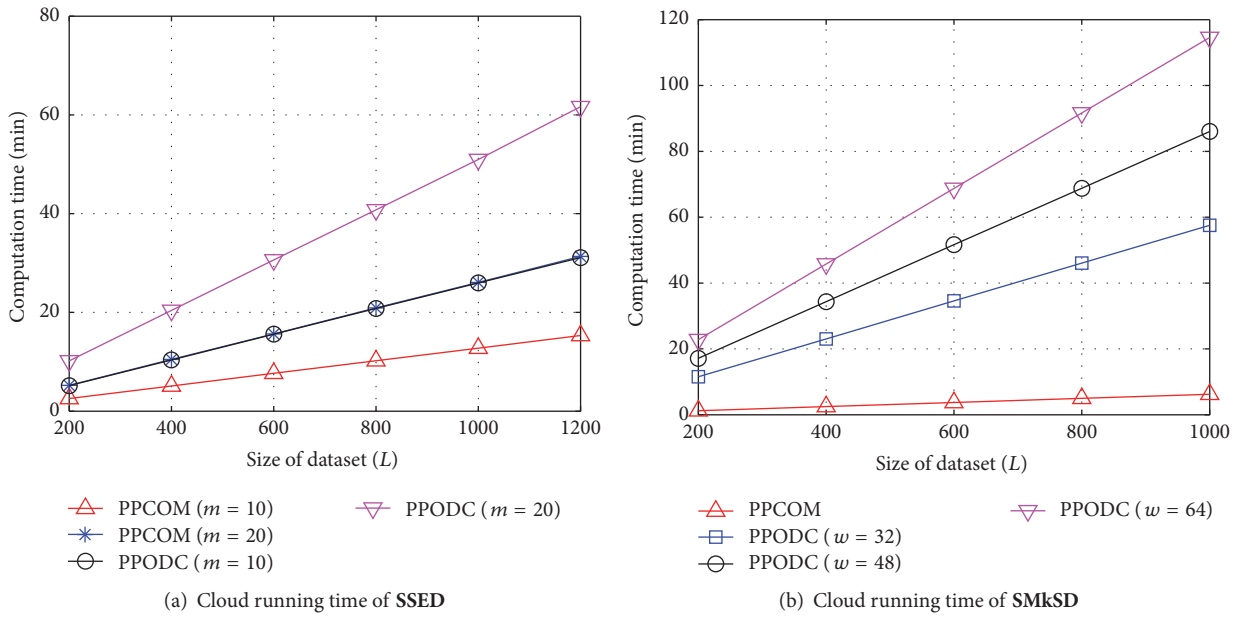
7.2.2. Factors Affecting PPCOM's Performance. There are three major factors affecting the outsourced performance: (1) the number of clusters (k); (2) the number of parallelized MCUs (f); (3) the size of aggregated dataset (L).

First, we evaluate the overhead on cloud servers with varying k and m when $L = 2000$ and $f = 8$, in comparison with the optimized PPODC with 8 parallelized server pairs. The results are given in Figures 4(a) and 4(b). It can be seen that the computation costs of both protocols grow almost linearly with k and PPCOM protocol outperforms PPODC; for example, when $m = 10$, $k = 12$, the cloud computation time of PPODC is 381.798 min, that is, 4.33 times that of our scheme. The efficiency is gained not only by improved secure primitives, but also by Spark framework. Nevertheless, the communication overhead of PPCOM is relatively high, which is mainly caused by frequent interactions during **SA** process. Furthermore, the growth of dimension size also increases the computational and communication overhead of both protocols.

Next, we evaluate the overhead on cloud servers with varying f when $k = 4, m = 20$. As shown in Figure 5(a), the computation time decreases with the growth of f. It can be derived that (1) the scaling of parallelized servers can accelerate the outsourced clustering task; (2) it takes PPCOM less computational cost than PPODC to accomplish the same amount of work. Figure 5(b) shows that the communication cost of both schemes remains unchanged regardless of f, because the total amount of clustering task is fixed. However, PPCOM incurs heavier communication overhead to protect privacy and access patterns.

Moreover, we evaluate the impact of L on cloud servers' performance with $k = 4$, $m = 10$. From Figure 6(a), we observe that the running time of both protocols increases with L, as more data need to be clustered. It is obvious that the cost of PPODC grows more sharply than ours. Figure 6(b) shows the computation overhead of Map stage and Reduce stage during execution of PPCOM, respectively. Map stage takes larger proportion of total cost than Reduce, while they scale linearly with L. In addition, the doubled parallelized MCUs save almost 40% of Map execution time.

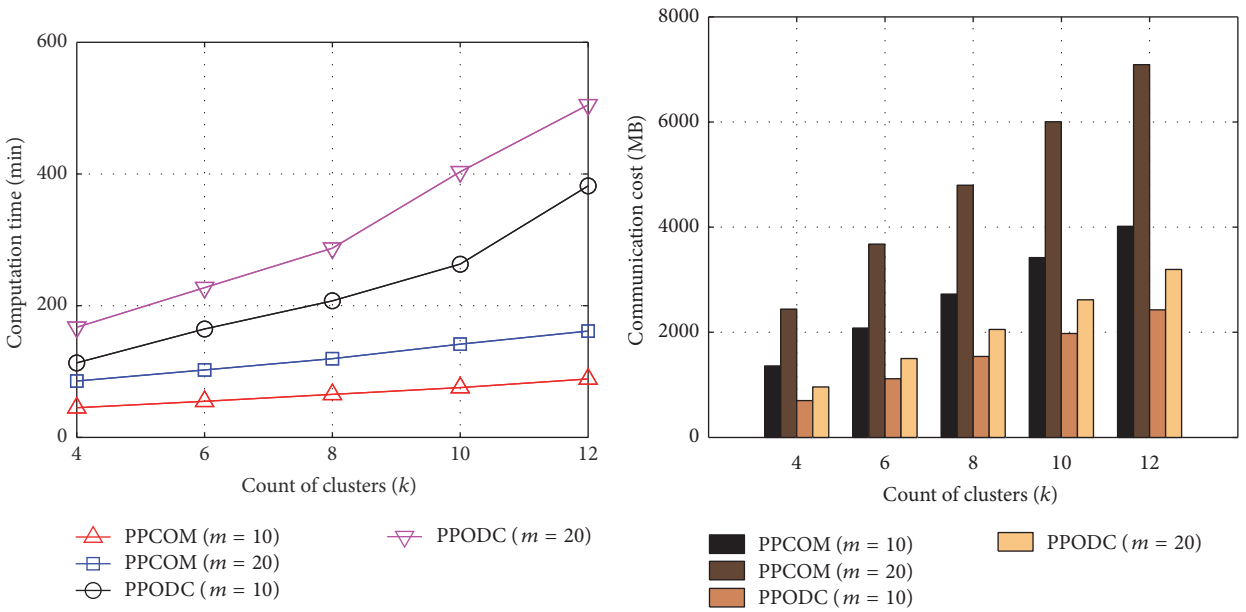
7.3. Comparative Summary. As shown in Table 4, we summarize qualitative comparisons with existing outsourced k-means protocols. All protocols are claimed to protect input data privacy. The encryption schemes of first three are constructed on random transformation, such as randomized kernel matrix [11] and random invertible matrices [14, 16]. They were proven to be secure against KSA (known-sample attack) that the attacker knows a set of plain data objects in the dataset, but not the corresponding encrypted values. Yuan and Tian's work [14] can also defeat LAA (linear analysis attack) introduced by [23]. However, these schemes are weak considering that attacker gets both some data objects and their encryptions. The latter four outsourcing protocols are proposed based on homomorphic encryption techniques, which can resist CPA (chosen plaintext attack). References [12, 13] adopt Liu's FHE as the underlying encryption scheme, which yet may not be secure enough as illustrated by [18]. Rao et al.'s [15] and our encryption schemes achieve semantic



(a) Cloud running time of **SSED**

(b) Cloud running time of **SMkSD**

FIGURE 3: Performance of **SSED** and **SMkSD** with varying size of datasets (L).



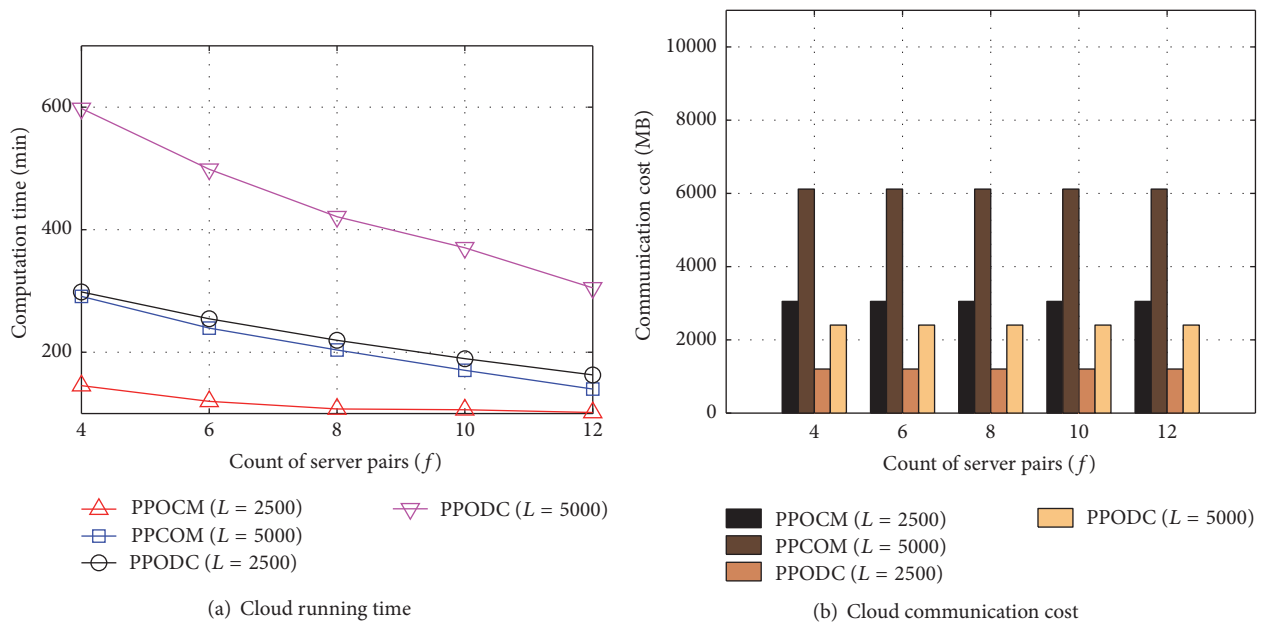
(a) Cloud running time

(b) Cloud communication cost

FIGURE 4: Experiment analysis with varying number of clusters (k) in the real dataset.

security, relying on hardness of Decisional Composite Residuosity [44] and Diffie-Hellman Problem [30], respectively.

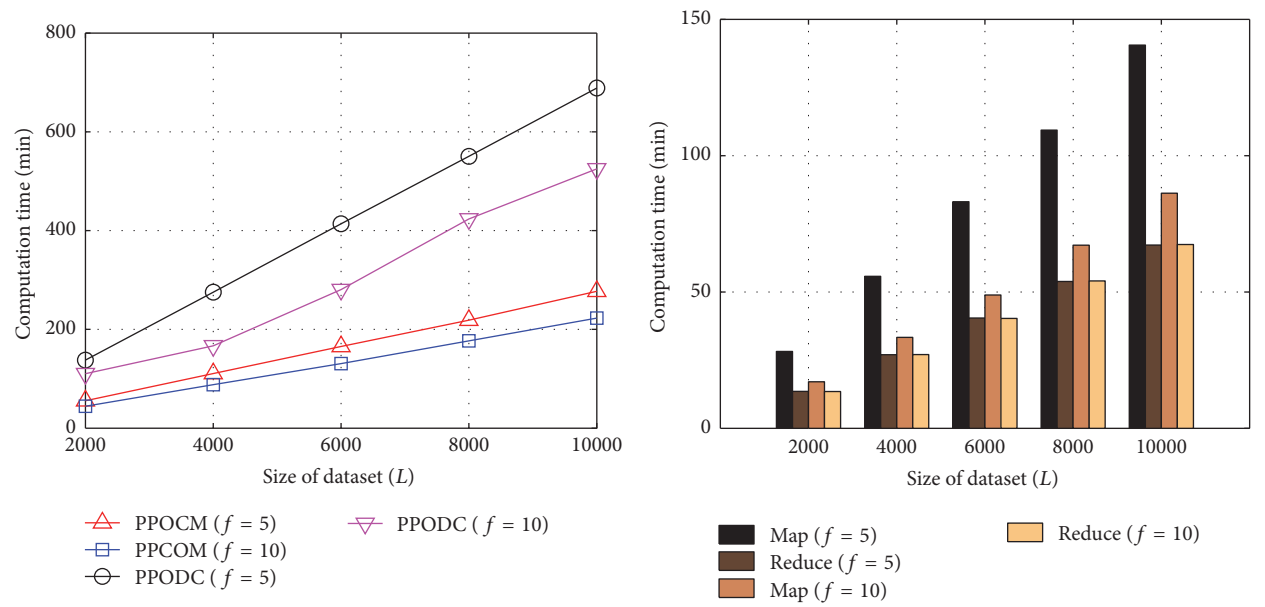
From this table, it can be seen that only [16] and ours support computation over encrypted data under multikeys, whereas one drawback in [16] is that they either reveal data owners' keys to query user or reveal query user's key to owners. Rao et al.'s [15] and ours hide access patterns by executing clustering in an oblivious way, preventing cloud servers from knowing the assignment membership of encrypted records which may be used to launch inference attack. Several schemes require data owners to participate in the mining process so as to update cluster centroids or to assist similarity comparison, except those from [11, 15] and ours. Almost all works allow the cloud to perform comparison operation between encrypted distances, while approach in [13] adopts a plain updatable matrix to compare. What is more, only Yuan and Tian's [14] and our researches consider how to integrate big data processing framework into privacy-preserving protocols. As a consequence, our solution achieves the most comprehensive security requirements and



(a) Cloud running time

(b) Cloud communication cost

FIGURE 5: Experiment analysis with varying number of parallelized MCUs (f) in the real dataset.



(a) Cloud running time

(b) Running time for Map and Reduce

FIGURE 6: Experiment analysis with varying size of joint datasets (L) from the real dataset.

feasibility for clustering outsourcing compared with current works.

8. Conclusion

In this paper, we proposed an efficient privacy-preserving protocol for outsourced k-means clustering over joint datasets encrypted under multiple data owners' keys. By utilizing double decryption cryptosystem, we proposed a series of privacy-preserving building blocks to transform ciphertexts and evaluate addition, multiplication, equality, and comparison, and so on, over encrypted data. Our protocol not only protects privacy of the aggregated database, but also hides access patterns under the semihonest model. Another improvement is that the outsourced clustering works under big data processing framework, which can be scaled to process big data. Experiments on real dataset show that our scheme is more efficient than existing approaches. However, the computation and communication costs of PPCOM are still heavy for large datasets. Our future work will focus

Table 3: Cloud running time for ciphertexts transformation (in min).

Subprotocol	$L = 2000$	$L = 4000$	$L = 6000$	$L = 8000$	$L = 10000$
SCT	11.6	23.2	35.3	46.5	57.9
KeyProd [28]	43.9	87.9	138.9	175.3	219.4

Table 4: Comparative summary of existing solutions for outsourced k-means.

Protocol	Security model	Data privacy protection	Support multikeys	Hide access patterns	No owner involvement	Encrypted distance comparison	Big data engine
Lin's [11]	KSA	√	×	×	√	√	×
Huang et al.'s [16]	KSA	√	√	×	×	√	×
Yuan and Tian's [14]	KSA & LAA	√	×	×	×	√	√
Liu et al.'s [12]	CPA	√	×	×	×	√	×
Almutairi et al.'s [13]	CPA	√	×	×	×	×	×
Rao et al.'s [15]	CPA	√	×	√	√	√	×
Ours	CPA	√	√	√	√	√	√

on improving the efficiency of outsourced protocols while protecting data privacy to withstand advanced attacks, for example, collusion attack between two CSPs.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Hajjat, X. Sun, Y.-W. E. Sung et al., "Cloudward bound: planning for beneficial migration of enterprise applications to the cloud," in *Proceedings of the 7th International Conference on Autonomic Computing, SIGCOMM 2010*, pp. 243–254, September 2010.

[2] Amazon Machine Learning, https://aws.amazon.com/machine-learning.

[3] Cloud Machine Learning Engine, https://cloud.google.com/ml-engine.

[4] "Do your best work with Watson," https://www.ibm.com/watson.

[5] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses," *The Computer Journal*, vol. 45, no. 1, Article ID 6127995, pp. 39–45, 2012.

[6] Health Information Privacy, U.S, Department of Health & Human Services, https://www.hhs.gov/hipaa/for-individuals/guidance-materials-for-consumers/index.html.

[7] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over arbitrarily partitioned data," in *ACM KDD*, 2003.

[8] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proceedings of the KDD-2005: 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 593–599, August 2005.

[9] S. Jha, L. Kruger, and P. McDaniel, "Privacy preserving clustering," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 3679, pp. 397–417, 2005.

[10] K.-P. Lin, "Privacy-preserving kernel k-means outsourcing with randomized kernels," in *Proceedings of the 13th IEEE International Conference on Data Mining Workshops, ICDMW 2013*, pp. 860–866, December 2013.

[11] K.-P. Lin, "Privacy-preserving kernel k-means clustering outsourcing with random transformation," *Knowledge and Information Systems*, vol. 49, no. 3, pp. 885–908, 2016.

[12] 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, ASIA CCS 2014*, pp. 123–133, June 2014.

[13] 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 *International Conference on Big Data Analytics and Knowledge Discovery (DaWak)*, pp. 274–285, 2017.

[14] J. Yuan and Y. Tian, "Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–12, 2017.

[15] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user k-means clustering," in *Proceedings of the 1st IEEE International Conference on Collaboration and Internet Computing, CIC 2015*, pp. 80–89, October 2015.

[16] Y. Huang, Q. Lu, and Y. Xiong, "Collaborative outsourced data mining for secure cloud computing," *Journal of Networks*, vol. 9, no. 9, pp. 2655–2664, 2014.

[17] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems (SIGMOD-PODS '09)*, pp. 139–152, Providence, RI, USA, July 2009.

[18] Y. Wang, "Notes on two fully homomorphic encryption schemes without bootstrapping. Cryptology ePrint Archive," Report 2015/519, 2015.

[19] S. De Capitani Di Vimercati, S. Foresti, and P. Samarati, "Managing and accessing data in the cloud: privacy risks and approaches," in *Proceedings of the 7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012*, pp. 1–9, October 2012.

[20] P. Williams, R. Sion, and B. Carbunar, "Building castles out of mud: practical access pattern privacy and correctness on untrusted storage," in *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS'08*, pp. 139–148, October 2008.

[21] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, pp. 439–450, 2000.

[22] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Knowledge Discovery in Databases: PKDD 2006*, vol. 4213 of *Lecture Notes in Computer Science*, pp. 297–308, Springer, Berlin, Germany, 2006.

[23] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE '13)*, pp. 733–744, IEEE, Brisbane, Australia, April 2013.

[24] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[25] W. Zhao, H. Ma, and Q. He, "Parallel *k*-means clustering based on mapreduce," in *Journal of Cloud Computing*, vol. 5931, pp. 674–679, Springer, Berlin, Germany, 2009.

[26] B. K. K. Samanthula, H. Chun, and W. Jiang, "An efficient and probabilistic secure bit-decomposition," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, ASIA CCS 2013*, pp. 541–546, May 2013.

[27] A. López, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the 2012 (ACM) Symposium on Theory of Computing*, pp. 1219–1234, 2012.

[28] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 12, pp. 2046–2058, 2013.

[29] B. Wang, M. Li, S. S. M. Chow, and H. Li, "A tale of two clouds: computing on data encrypted under multiple keys," in *Proceedings of the 2014 IEEE Conference on Communications and Network Security, CNS 2014*, pp. 337–345, October 2014.

[30] T.-Y. Youn, Y.-H. Park, C. H. Kim, and J. Lim, "An efficient public key cryptosystem with a privacy enhanced double decryption mechanism," in *Selected Areas in Cryptography*, vol. 3897, pp. 144–158, Springer, Berlin, Germany, 2006.

[31] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Advances in Cryptology—(ASIACRYPT) 2003*, vol. 2894, pp. 37–54, Springer, Berlin, Germany, 2003.

[32] E. Kiltz and J. Malone-Lee, "A general construction of IND-CCA2 secure public key encryption," in *Cryptography and Coding*, vol. 2898, pp. 152–166, Springer, Berlin, Germany, 2003.

[33] D. Galindo and J. Herranz, "On the security of public key cryptosystems with a double decryption mechanism," *Information Processing Letters*, vol. 108, no. 5, pp. 279–283, 2008.

[34] M. D. Van and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," *HotSec'10*, pp. 1–8, 2010.

[35] S. S. M. Chow, J. H. Lee, and M. Strauss, "Two-party computation model for privacy-preserving queries over distributed databases," *NDSS*, 2009.

[36] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[37] 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, no. 5, pp. 1–14, 2015.

[38] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita, "Big data analytics in the cloud: spark on hadoop vs MPI/OpenMP on Beowulf," *Procedia Computer Science*, vol. 53, no. 1, pp. 121–130.

[39] X. Meng, J. Bradley, B. Yavuz, E. Sparks, and S. Venkataraman, "MLlib: machine learing in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–12414, 2015.

[40] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 3rd edition, 2011.

[41] O. Goldreich, "Basic Applications," in *The Foundations of Cryptography*, vol. 2, Cambridge University Press, 2004.

[42] R. Peralta, Report on Integer Factorization, 2001, http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1025_report.pdf.

[43] M. Naeem and S. Asghar, "KEGG Metabolic Reaction Network Data Set," The UCI KDD Archive, 1999, http://archive.ics.uci.edu/ml/datasets/KEGG+Metabolic+Reaction+Network+(Undirected).

[44] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—(EUROCRYPT) '99 (Prague)*, vol. 1592, pp. 223–238, Springer, Berlin, Germany, 1999.