Outsourced k-Means Clustering over Encrypted Data Under Multiple Keys in Spark Framework

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
r.hong_nudt@hotmail.com, freshcdwhm@163.com, ljabc730@gmail.com, haojialupb@163.com, qwertmingx@sina.com

Abstract. As the quantity of data produced is rapidly rising in recent years, clients lack of computational and storage resources tend to outsource data mining tasks to cloud service providers in order to improve efficiency and save costs. It's also increasing 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 under their own keys. This paper focuses on privacy-preserving k-means clustering over the joint datasets from multiple sources. Unfortunately, existing secure outsourcing protocols are either restricted to a single key setting or quite inefficient because of frequent client-to-server interactions, making it impractical for wide application. To address these issues, we propose a set of secure building blocks and outsourced clustering protocol under Spark framework. Theoretical analysis shows that our scheme protects the confidentiality of the joint database and mining results in the standard threat model with small computation and communication overhead. Experimental results also demonstrate its significant efficiency improvements compared with existing methods.

Keywords: Outsourced k-means clustering · Multiple keys Cloud environment · Spark framework

1 Introduction

With tremendous amount of data gathered each day, it's increasingly difficult for resource-constrained clients (e.g., mobile devices) to perform computationally intensive task locally. It is a reasonable option to outsource data mining tasks to cloud service provider 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 facilitate clients to train and deploy their own models, e.g., Amazon Machine Learning [2], Google Cloud Machine Learning Engine [3], IBM Watson [4], etc. Despite these advantages,

X. Lin et al. (Eds.): SecureComm 2017, LNICST 238, pp. 67–87, 2018.
https://doi.org/10.1007/978-3-319-78813-5_4

privacy issues impede clients from migrating to cloud due to concerns of privacy breach. For example, by collecting medical health records from multiple patients and social networks, hospitals may build more accurate models to improve diagnosis or to predict disease outbreaks [5]. It is, however, crucial to guarantee the security and privacy of e-health records which usually contain a lot of sensitive information, such as personal identity and health condition. A straightforward solution is allowing patients to encrypt the data with their own keys before outsourcing. Whereas it still remains a big challenge for current cloud-based services to perform machine learning operations over encrypted data. Thus, in this paper, we try to solve the above issues by focusing on privacy protection techniques regarding a typical data mining algorithm–k-means clustering [6].

Traditional privacy-preserving clustering schemes cannot be directly adopted to address the privacy issues during outsourcing. Their target is to compute clusters through interactions among participating data holders without revealing respective data to others [7,8], 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 employ the same key data for encryption [9–11]. It is apparent that the single key restriction has some drawbacks: (1) a compromised data owner can easily decrypt others' ciphertexts if they share the identical symmetric or asymmetric keys as methods in [9,10]; (2) without knowing the secret key, owners cannot retrieve their own data downloaded from the cloud if the datasets are encrypted under cloud's public key [11]. To overcome these limitations, data owners should encrypt their datasets with their own keys, which calls for computation over encrypted data under multiple keys. The recent work [12] concerning multi-key scenario is built on geometric transformation to preserve the dot product as KNN scheme [13]. However, this method is weak in security, for all instances may be recovered if the attacker can setup a group of equations with enough linearly independent instances. Furthermore, only one or two cloud servers are adopted in the existing works, the great computing power of distributed cloud environment is not fully exploited to accelerate the outsourcing process.

In this paper, we present a method for Privacy-Preserving Outsourced Clustering under Multiple keys (PPOCM), which enables distributed cloud servers to perform clustering collaboratively over the aggregated datasets encrypted under multiple keys with no privacy leakage. Specifically, the major contributions of this paper are three folds.

- Firstly, we propose a set of privacy-preserving building blocks for basic arithmetic operations. Based on the cryptosystem with double decryption property, our schemes allows to evaluate addition and multiplication over inputs encrypted under different keys. Through these primitives, cloud servers are able to compute Euclidean distances between records and cluster centers.
- Secondly, as the encryptions are probabilistically randomized and incomparable, we propose an efficient method to compare encrypted Euclidean distances

in a privacy-preserving manner. In addition, clients are not required to participate in the comparison operations during k-means outsourcing.
- Thirdly, based on the proposed secure building blocks, we design PPOCM protocol by taking advantage of a big data analytic framework–Spark and distributed cloud resources. Theoretical analysis demonstrates the proposed protocol protects the content of data records, intermediate results as well as the privacy of clustering result in the semi-honest model. Experimental results on real dataset shows that PPOCM is much more efficient than existing methods in terms of computation and communication overhead.

A comparative summary of existing outsourced k-means protocols is presented in Table 1.

Table 1. Comparative summary of existing solutions for outsourced k-means

Protocol	Encryption type	Data privacy protection	Multi-key support	Minimal owner participation	Ciphertext comparison	Big data engine
Lin's [9]	Symmetric	$\surd$	$\times$	$\surd$	$\times$	$\times$
Liu's [10]	Symmetric	$\surd$	$\times$	$\times$	$\surd$	$\times$
Huang's [12]	Symmetric	$\surd$	$\surd$	$\times$	$\times$	$\times$
Rao's [11]	Asymmetric	$\surd$	$\times$	$\surd$	$\surd$	$\times$
Ours	Asymmetric	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

The rest of the paper is organized as follows. In Sect. 2, we review k-means clustering algorithm and the underlying encryption scheme. The system model, threat model, and design goals are presented in Sect. 3. The design details of our proposed protocol–PPOCM are described in Sect. 4. We also analyze the security of the protocol in Sect. 5. Section 6 shows the theoretical and experimental evaluations. Section 7 discusses related work. Finally, we conclude the paper and outline future work in Sect. 8.

2 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.

2.1 k-Means Clustering

Given records $t_1, ..., t_l$, the k-means clustering algorithm partitions them into k disjoint clusters, denoted by $c_1, ..., c_k$. 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 $1 \leq i \leq l$, $1 \leq j \leq k$. Note that the i^{th} record belongs to c_j if $V_{i,j} = 1$; otherwise, $V_{i,j} = 0$.

Initial k records are selected randomly as cluster centers $\mu_1, ..., \mu_k$. 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 $1 \leq j \leq k$, and updates V according to $\arg\min_j ||t_i - \mu_j||^2$, i.e., 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 $c_1, ..., c_k$, the clustering algorithm begins next iteration. Finally, the algorithm terminates if the matrix V does not vary any more, or a predefined maximum count of iterations is reached [9].

2.2 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. [14] as our secure primitive, which is more efficient than the scheme in [15] in that Youn's approach applies smaller modulus in cryptographic operations. The major steps are shown in the following.

- **Key Generation** ($\mathsf{KeyGen}(\kappa) \rightarrow N, g, msk, pk, sk$): 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 $msk := (p, q)$ is known only to the authority. The public parameters are N, g. A cloud user picks a random integer $sk \in \{0, 1, ..., 2^{\kappa-1} - 1\}$ as secret key and computes $pk := g^{sk} \bmod N$ as public key.
- **Encryption** ($\mathsf{Enc}(pk, m) \rightarrow C$): The encryption algorithm takes the message $m \in \mathbb{Z}_N$ and pk as inputs, and outputs ciphertext $C = (A, B)$, where $A := g^r \bmod N$, $B := pk^r \cdot m \bmod N$, and r is a random $\kappa - 1$ bit integer.
- **Decryption with user key** ($\mathsf{uDec}(sk, C) \rightarrow m$): The decryption algorithm takes ciphertext C and sk as inputs, and outputs the message m by computing $m \leftarrow B/A^{sk} \bmod N$.
- **Decryption with master key** ($\mathsf{mDec}(msk, pk, C) \rightarrow m$): Given msk, pk, and C, the authority decrypts C by factorizing N. The secret key of C can be obtained by computing $sk \leftarrow L(pk^{p-1})/L(g_p)$, where function L is defined as $L(x) = \frac{x-1}{p}$. Then, m is recovered by computing $m \leftarrow B/A^{sk} \bmod N$.

By applying the general conversion method in [16], the scheme was claimed to be IND-CCA2 secure under the hardness of solving the p-DH Problem [14]. However, Galindo et al. [17] has 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 sk during master decryption proposed in [17]. If $sk \geq 2^{\kappa-1}$, the master entity outputs a rejection message; otherwise, the decryption proceeds as usual.

3 Problem Statement

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

3.1 System Model

In our system model as depicted in Fig. 1, there are three types of entities, i.e., Cloud Users, Computation Service Provider, and Cryptographic Service Provider. Cloud Users consist of Data Owners and Query Clients. Computation Service Provider is composed of one Coordinating Server and a set of Executing Servers; Cryptographic Service Provider comprises a Key Management Server and a set of Assistant Servers.

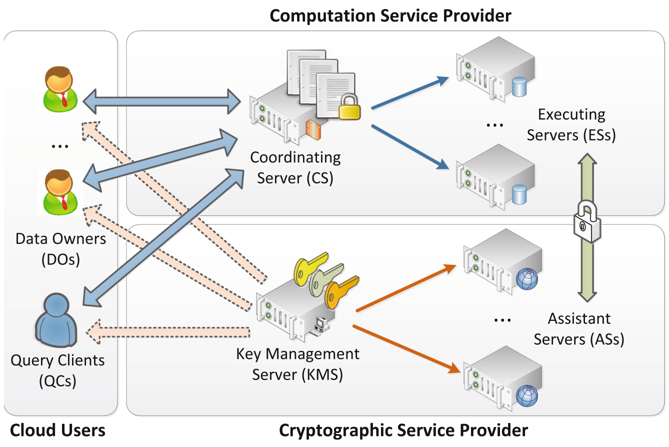


Fig. 1. System model

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 DO$_1$,...,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 $L = \sum_i^n l_i$. Let $t_{j,h}^i$ be the h^{th} attribute value of j^{th} record in D_i for $h \in [1, m]$ and $j \in [1, L]$. We assume DO$_i$ does not collude with the cloud to breach privacy.
2. Query Client (QC): QC is an authorized party requesting k-means clustering tasks over the aggregated datasets. QC should not involve in outsourced computation and is able to decrypt the result with his own secret key.
3. Coordinating Server (CS): CS not only stores and manages combined datasets from multiple DOs, but also deploys cloud computing resources to perform clustering jobs and returns the final calculated clusters to QC.

4. Executing Server (ES): ES is the task node that undertakes the workload assigned by CS. There are $\text{ES}_1, ..., \text{ES}_\theta$ with massive computing power, making it feasible to implement parallel processing model like Spark paradigm.
5. Key Management Server (KMS): KMS generates and distributes public key parameters of the underlying cryptosystem. It holds the master secret key of PKC-DD, which is used to convert ciphertext's encryption key.
6. Assistant Server (AS): AS holds decryption key generated by KMS. With that key, AS assists ES to execute a series of privacy-preserving building blocks. There are λ ASs, i.e., $\text{AS}_1, ..., \text{AS}_\lambda$ in the system.

The major workflow of PPOCM is summarized as follows. For $\forall i \in [1, n]$, DO_i generates its own key pair pk_i/sk_i using the parameters produced by KMS, and encrypts D_i with pk_i before outsourcing to CS. With the joint datasets as inputs, the distributed cloud servers are scheduled to perform k-means clustering algorithm in a privacy-preserving manner. The cloud returns QC the encrypted cluster centers under QC's public key after clustering iteration terminates.

3.2 Threat Model

In our threat model, all cloud servers and clients are assumed to be semi-honest, which means that they strictly follow the prescribed protocol but try to infer private information using the messages they receive during the protocol execution. DO, QC, ES, AS and KMS are interested in learning plain data belonging to other parities. Therefore, we introduce an active adversary $\mathcal{A}$ in the threat model. The target of $\mathcal{A}$ is to decrypt the ciphertexts from the challenge DO and challenge QC with the following capabilities:

– $\mathcal{A}$ may compromise all the ESs to guess the plaintexts of received ciphertexts from DOs and ASs during the execution of the protocol.
– $\mathcal{A}$ may compromise all the ASs and KMS to guess the plaintext values of ciphertexts sent from ESs during the protocol interactions.
– $\mathcal{A}$ 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.

However, we assume the adversary $\mathcal{A}$ cannot compromise two cloud providers simultaneously; otherwise, $\mathcal{A}$ is able to decrypt any ciphertext stored on CS and ES with secret keys from KMS and AS. In other words, there's no collusion between these two cloud providers, whereas servers from the same provider may collude. We remark that such assumptions are typical in adversary models used in cryptographic protocols (e.g., [11,20]), in that cloud providers are mostly competitors and not willing to disclose business info to others. $\mathcal{A}$ is also assumed to have no prior knowledge about samples for unpublished data.

3.3 Design Objectives

Given the aforementioned system model and threat model, our design should achieve the following objectives:

- **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.
- **Confidentiality**. Nothing regarding the contents of datasets $D_1, ..., D_n$ and cluster centers $\mu_1, ..., \mu_k$ should be revealed to the semi-honest cloud servers.
- **Efficiency**. The most computation should be processed by cloud in a highly efficient way while DOs and QCs are not required to involve in the outsourced clustering.

4 The PPOCM Solution

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

Recall that in Sect. 3.1, the semi-honest but non-colluding cloud servers need to cooperate to perform computation over encrypted data under PKC-DD scheme. In the first place, KMS takes a security parameter κ as input, and generates public parameter (N, g) for all parties and master secret key msk for itself by executing $\mathsf{KeyGen}(\kappa)$. After KMS generates a key pair pk_u/sk_u used for ciphertext transformation, pk_u is distributed to cloud ES while sk_u is sent to cloud AS. With N and g, DO$_i$ produces its own public/private key pair pk_i/sk_i and broadcasts pk_i to cloud servers, for $i = 1, ..., n$. Hereafter, let $\mathsf{Enc}_{pk}(\cdot)$ denote the underlying encryption, $\mathsf{uDec}_{sk}(\cdot)$ and $\mathsf{mDec}_{sk}(\cdot)$ denote user-side decryption and master-side decryption, respectively.

4.1 Privacy-Preserving Building Blocks

We present a set of privacy-preserving building blocks in the distributed cloud environment, aiming at solving basic operations on ciphertexts which include secure ciphertext transformation, multiplication, addition, Euclidean distance computation, comparison, etc.

Secure Ciphertext Transformation (SCT) Protocol. Given that CS holds $\mathsf{Enc}_{pk_x}(m)$, and KMS holds (msk, pk_y), the goal of the SCT protocol is to transform encrypted 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 KMS or CS, meanwhile the output $\mathsf{Enc}_{pk_y}(m)$ is only known to CS. The complete steps are shown in Algorithm 1.

To start with, CS generates an invertible random number $r \in_R \mathbb{Z}_N$, which denotes r is randomly picked in $\mathbb{Z}_N$. Note that the condition $r < 2^{\kappa-1}$ ensures r is invertible in $\mathbb{Z}_N$ due to $|r| < |p|$. It's obvious that the PKC-DD scheme is multiplicative homomorphic, so we have $\mathsf{Enc}_{pk}(m_1) \times \mathsf{Enc}_{pk}(m_2) \rightarrow \mathsf{Enc}_{pk}(m_1 \cdot m_2)$. Then we exploit this to blind m so that KMS does not know m even if it is able to decrypt $\mathsf{Enc}_{pk_x}(r \cdot m)$ via using msk. Hereafter, "$\times$" denotes multiplication operation in the encrypted domain while "$\cdot$" represents multiplication in the plaintext domain. Finally, CS removes the randomness by multiplying the encrypted inverse of r due to $\mathsf{Enc}_{pk_y}(m) = \mathsf{Enc}_{pk_y}(r \cdot m \cdot r^{-1} \bmod N)$.

Algorithm 1. SCT($\mathsf{Enc}_{pk_x}(m), pk_y$) $\rightarrow \mathsf{Enc}_{pk_y}(m)$

```
Require: CS has $\mathsf{Enc}_{pk_x}(m)$, $pk_x$, and $pk_y$; KMS has $msk$, $pk_x$, and $pk_y$.
1: CS:
    a) Generate a random number $r \in_R \mathbb{Z}_N$, which satisfies $r < 2^{\kappa-1}$;
    b) Compute $\mathsf{Enc}_{pk_x}(r \cdot m) \leftarrow \mathsf{Enc}_{pk_x}(m) \times \mathsf{Enc}_{pk_x}(r)$;
    c) Send $\mathsf{Enc}_{pk_x}(r \cdot m)$ to KMS;
2: KMS:
    a) Decrypt $r \cdot m \leftarrow \mathsf{mDec}(msk, pk_x, \mathsf{Enc}_{pk_x}(r \cdot m))$;
    b) Encrypt $\mathsf{Enc}_{pk_y}(r \cdot m) \leftarrow \mathsf{Enc}(pk_y, r \cdot m)$;
    c) Send $\mathsf{Enc}_{pk_y}(r \cdot m)$ to CS;
3: CS:
    a) Compute $\mathsf{Enc}_{pk_y}(m) \leftarrow \mathsf{Enc}_{pk_y}(r \cdot m) \times \mathsf{Enc}_{pk_y}(r^{-1})$;
```

Secure Addition (SA) Protocol. It takes $\mathsf{Enc}_{pk_u}(m_1)$ and $\mathsf{Enc}_{pk_u}(m_2)$ held by ES and sk_u held by AS as inputs. The output is the encrypted addition of m_1 and m_2, i.e., $\mathsf{Enc}_{pk_u}(m_1 + m_2)$, which is only known to ES. As the encryption scheme is not additively homomorphic, it requires interactions between ES and AS. The major steps are shown in Algorithm 2.

In this protocol, cloud server ES first generates a random number $r \in_R \mathbb{Z}_N$. The ciphertexts of m_1 and m_2 are blinded with r. Using the secret key sk_u, AS is able to decrypt the encrypted randomized inputs $\mathsf{Enc}_{pk_u}(r \cdot m_1)$, $\mathsf{Enc}_{pk_u}(r \cdot m_2)$. AS then computes the sum of two decrypted messages denoted by α, and sends the encryption of α back to ES. Finally, ES obtains $\mathsf{Enc}_{pk_u}(m_1 + m_2)$ by multiplying $\mathsf{Enc}_{pk_u}(\alpha)$ with $\mathsf{Enc}_{pk_u}(r^{-1})$ based on multiplicative homomorphism, since $\mathsf{Enc}_{pk_u}(m_1 + m_2) = \mathsf{Enc}_{pk_u}((m_1 + m_2) \cdot r \cdot r^{-1} \bmod N)$.

Note that m_1 and m_2 are blinded by the same random value, so AS can easily compute the ratio by $m_1/m_2 \leftarrow m_1 r/m_2 r$, which may be used to distinguish inputs. However, our security model is based on the assumption that the adversary has no background knowledge about the raw data distribution, which is common for unpublished data. Hence, the adversary cannot deduce sensitive information about users' data.

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 $||t_i - \mu_j||^2$. Suppose ES holds the ciphertext of i^{th} data record t_i, and the ciphertext of j^{th} cluster centroid μ_j, while AS holds the secret key sk_u.

Note that μ_j is a vector composed of fractional values which may be rational numbers. However, ring $\mathbb{Z}_N$ supports no rational operation, so a new form of expression is required to represent the cluster center. Let $<s_j, |c_j|>$ denote the new form of cluster center, where s_j and $|c_j|$ represent the sum, the total

Algorithm 2. SA($\mathsf{Enc}_{pk_u}(m_1), \mathsf{Enc}_{pk_u}(m_2)) \rightarrow \mathsf{Enc}_{pk_u}(m_1 + m_2)$

```
Require: ES has $\mathsf{Enc}_{pk_u}(m_1)$ and $\mathsf{Enc}_{pk_u}(m_2)$; AS has $sk_u$.
1: ES:
    a) Generate a random number $r \in_R \mathbb{Z}_N$ and $r < 2^{\kappa-1}$;
    b) Compute $\mathsf{Enc}_{pk_u}(r \cdot m_1) \leftarrow m_1' \times \mathsf{Enc}_{pk_u}(r)$;
    c) Compute $\mathsf{Enc}_{pk_u}(r \cdot m_2) \leftarrow m_2' \times \mathsf{Enc}_{pk_u}(r)$;
    d) Send $\mathsf{Enc}_{pk_u}(r \cdot m_1)$, $\mathsf{Enc}_{pk_u}(r \cdot m_2)$ to AS;
2: AS:
    a) Decrypt $r \cdot m_1 \leftarrow \mathsf{uDec}(sk_u, \mathsf{Enc}_{pk_u}(r \cdot m_1))$;
    b) Decrypt $r \cdot m_2 \leftarrow \mathsf{uDec}(sk_u, \mathsf{Enc}_{pk_u}(r \cdot m_2))$;
    c) Compute $\alpha \leftarrow r \cdot m_1 + r \cdot m_2$;
    d) Encrypt $\mathsf{Enc}_{pk_u}(\alpha) \leftarrow \mathsf{Enc}(pk_u, \alpha)$;
    e) Send $\mathsf{Enc}_{pk_u}(\alpha)$ to ES;
3: ES:
    a) Compute $\mathsf{Enc}_{pk_u}(m_1 + m_2) \leftarrow \mathsf{Enc}_{pk_u}(\alpha) \times \mathsf{Enc}_{pk_u}(r^{-1})$;
```

number of the records belonging to c_j, respectively. It's easily observed that $s_j = \Sigma_{h=1}^{L}(V_{h,j} \cdot t_h)$ and $|c_j| = \Sigma_{h=1}^{L} V_{h,j}$, where $V_{h,j}$ denotes the membership between t_h and c_j. $\Omega_{i,j}$ is defined as the scaled squared distance between t_i and μ_j, which satisfies that $||t_i - \mu_j|| = \frac{\sqrt{\Omega_{i,j}}}{|c_j|}$. So $\Omega_{i,j}$ can be calculated as follows:

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

where $i \in [1, L]$, $j \in [1, k]$, and m is the dimension size. Taking $\mathsf{Enc}_{pk_u}(t_i)$ and $<\mathsf{Enc}_{pk_u}(s_j), |c_j|>$ as inputs, ES and AS jointly execute SSED by invoking SA subprotocol and output $<\mathsf{Enc}_{pk_u}(\Omega_{i,j}), |c_j|>$. We omit the implementation details of SSED since the steps are straightforward. In addition, although the count of data records is directly revealed to cloud server, the numerator of average attribute, i.e., s_j is still encrypted. Thus it's impossible to infer the real centroid value as long as the underlying encryption scheme is semantically secure.

Secure Distance Comparison (SDC) Protocol. Supposing ES holds $<\mathsf{Enc}_{pk_u}(\Omega_{i,a}), |c_a|>$, $<\mathsf{Enc}_{pk_u}(\Omega_{i,b}), |c_b|>$ and AS holds sk_u, where $i \in [1, L]$, $a, b \in [1, k]$, $a \neq b$, the output of SDC is the minimum distance. Since the encryption scheme is probabilistic and does not preserve the order of plaintexts, ES and AS should jointly compute the minimum without revealing $\Omega_{i,a}$ and $\Omega_{i,b}$ to both parties.

Our basic idea is to compute the encrypted difference between the two inputs, based on which AS is able to judge its sign and returns an identifier that indicates the minimum value. It is commonsense that the maximum size of message is normally far smaller than modulus N. Let ε be the maximum size of plaintext. The maximum value is $2^\varepsilon - 1$ and minimum value is $-2^\varepsilon + 1$. After modular computation, the positive difference falls into range $[1, 2^\varepsilon - 1]$ while the negative difference is in the range $[N - 2^\varepsilon + 1, N - 1]$. Normally, if we get a value that is larger than $2^\varepsilon - 1$, then the value can be considered as a negative. The difference between the two squared Euclidean distances can be calculated as follows:

$$\begin{aligned}\mathsf{Enc}_{pk_u}\left(||t_i - c_a||^2 - ||t_i - c_b||^2\right) &= \mathsf{Enc}_{pk_u}\left(\frac{\Omega_{i,a}}{|c_a|^2} - \frac{\Omega_{i,b}}{|c_b|^2}\right) \\ &\propto \mathsf{Enc}_{pk_u}\left(\Omega_{i,a} \cdot |c_b|^2 - \Omega_{i,b} \cdot |c_a|^2\right).\end{aligned} \tag{2}$$

By observation from Eq. (2), it is only required to determine the sign of $\Omega_{i,a} \cdot |c_b|^2 - \Omega_{i,b} \cdot |c_a|^2$, defined as $\delta_{a,b}$. The overall steps are given in Algorithm 3.

As revealing distance difference δ directly to AS may violate privacy, it's necessary to blind δ with random number r, which is selected randomly from a special range. Suppose η is the threshold for sign judgement, which is chosen according to $2^\varepsilon - 1 < \eta < N + 2^\varepsilon - 1$. To preserve the original sign of δ, the blinding factor r should suffice conditions in Eq. (3). They ensure that the scaled positive and negative ranges can still be judged with η. It can be verified that $1 < r < \min\{N - \eta, \lfloor \frac{\eta - \phi N}{2^\varepsilon - 1} \rfloor\}$, where $\phi \in \mathbb{Z}$.

$$\begin{cases}(2^\varepsilon - 1) \cdot r \bmod N < \eta \\ (N - 1) \cdot r \bmod N > \eta \\ (N + 1 - 2^\varepsilon) \cdot r \bmod N > \eta\end{cases} \tag{3}$$

Secure Minimum Among *k* Distances (SMkD) Protocol. SMkD aims at computing the encrypted minimum value from k encrypted Euclidean distances. Assume that ES holds $d_1, d_2, ..., d_k$, where $d_j = <\mathsf{Enc}_{pk_u}(\Omega_{i,j}), |c_j|>$, $i \in [1, L]$, $j \in [1, k]$, and AS holds the secret key sk_u. The output of SMkD is encryption of the shortest distance among $d_1, d_2, ..., d_k$. Let $d_{\min} = <\mathsf{Enc}_{pk_u}(\Omega_{\min}), |c_{\min}|>$ represent the minimum. To execute SMkD, we compute the minimum by utilizing SDC with two inputs each time in a sequential fashion. The computation complexity of this algorithm is $O(k)$.

4.2 The Proposed PPOCM Protocol

In this subsection, we present our proposed PPOCM protocol for the standard k-means algorithm working in the distributed cloud environment.

The primary goal of PPOCM is to schedule a group of cloud servers to perform clustering task over the joint datasets encrypted under multiple keys, meanwhile no information regarding the content of record attributes should be revealed to the semi-honest servers. In order to improve the performance, we

Algorithm 3. $\mathrm{SDC}(\psi_{i,a}, \psi_{i,b}) \rightarrow < \mathsf{Enc}_{pk_u}(\Omega_{\min}), |c_{\min}| >$

```
Require: ES has encrypted distances $\psi_{i,a}, \psi_{i,b}$; AS has $sk_u$, where $\psi_{i,a} =< \mathsf{Enc}_{pk_u}(\Omega_{i,a}), |c_a| >$, $\psi_{i,b} =< \mathsf{Enc}_{pk_u}(\Omega_{i,b}), |c_b| >$.
1: ES:
    a) Compute $\mathsf{Enc}_{pk_u}(\Omega'_{i,a}) \leftarrow \mathsf{Enc}_{pk_u}(\Omega_{i,a}) \times \mathsf{Enc}_{pk_u}(|c_b|^2)$;
    b) Compute $\mathsf{Enc}_{pk_u}(\Omega'_{i,b}) \leftarrow \mathsf{Enc}_{pk_u}(\Omega_{i,b}) \times \mathsf{Enc}_{pk_u}(|c_a|^2)$;
    c) Compute $\mathsf{Enc}_{pk_u}(\Omega''_{i,b}) \leftarrow \mathsf{Enc}_{pk_u}(\Omega'_{i,b}) \times \mathsf{Enc}_{pk_u}(-1 \bmod N)$;
2: ES and AS:
    a) Compute $\mathsf{Enc}_{pk_u}(\delta_{a,b}) \leftarrow \mathrm{SA}(\mathsf{Enc}_{pk_u}(\Omega'_{i,a}), \mathsf{Enc}_{pk_u}(\Omega''_{i,b}))$;
3: ES:
    a) Generate a random number $r \in_R \mathbb{Z}_N$ according to Eq. (3);
    b) Compute $\mathsf{Enc}_{pk_u}(\delta'_{a,b}) \leftarrow \mathsf{Enc}_{pk_u}(\delta_{a,b}) \times \mathsf{Enc}_{pk_u}(r)$;
    c) Send $\mathsf{Enc}_{pk_u}(\delta'_{a,b})$ to AS;
4: AS:
    a) Decrypt $\delta'_{a,b} \leftarrow \mathsf{uDec}(sk_u, \mathsf{Enc}_{pk_u}(\delta'_{a,b}))$;
    b) if $\delta'_{a,b} > \eta$ then
        – Encrypt $sn \leftarrow \mathsf{Enc}_{pk_{ES}}(1)$;
    c) else
        – Encrypt $sn \leftarrow \mathsf{Enc}_{pk_{ES}}(r')$, where $r' \in_R \mathbb{Z}_N \wedge r' \neq 1$;
    d) Send $sn$ to ES;
5: ES:
    a) if $\mathsf{uDec}(sk_{ES}, sn) == 1$ then
        – Compute $\mathsf{Enc}_{pk_u}(\Omega_{\min}) \leftarrow \mathsf{Enc}_{pk_u}(\Omega_{i,a})$, $|c_{\min}| \leftarrow |c_a|$;
    b) else
        – Compute $\mathsf{Enc}_{pk_u}(\Omega_{\min}) \leftarrow \mathsf{Enc}_{pk_u}(\Omega_{i,b})$, $|c_{\min}| \leftarrow |c_b|$;
```

leverage a fast engine called Spark for large-scale data processing [21]. Spark uses 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 [22], it does not take privacy protection into consideration and cannot process encrypted data directly. So it's necessary to integrate our proposed building blocks in Sect. 4.1 and the idea of Spark computing framework into designing PPOCM.

The PPOCM protocol is composed of four phases, namely, Data Uploading, Ciphertext Transformation, Clustering Computation, as well as Result Retrieval, the details of which are described in the following.

Data Uploading Phase. To start with, $\mathrm{DO}_i(i \in [1,n])$ generates its own public/private key pair, i.e., pk_i/sk_i, by using public parameter N, g [14]. DO_i encrypts D_i with pk_i by calculating $\mathsf{Enc}(pk_i, t^i_{j,h})$. Recall that $t^i_{j,h}$ means the h^{th} attribute value of j^{th} record t^i_j in D_i for $h \in [1,m]$ and $j \in [1,L]$. Without loss of generality, we assume the sizes of all datasets are equal to be l, so $L = nl$. Let D'_i denote the encrypted D_i. After DO_i uploads the D'_i to CS for $\forall i \in [1,n]$, the server obtains the joint database D', where $D' = \cup^n_{i=1} D'_i$. With D' storing in the cloud, DO_i is able to retrieve its data and decrypt them with its private key sk_i, whereas DO_i cannot decrypt D'_j without sk_i for $i \neq j$.

Ciphertext Transformation Phase. Upon receiving clustering request from QC, CS initiates ciphertext transformation process which aims at converting ciphertexts under pk_i into encryptions under the unified key pk_u, for $i \in [1,n]$. CS first replicates D' into D'_r to ensure DOs' accessibility to their original dataset. Then KMS and CS jointly execute SCT subprotocol. The output of converted dataset (denoted by D'_u) is known only to CS while no privacy is revealed to KMS. This phase is essential for two reasons: (1) multiplicative homomorphic operation can be performed by ES independently only under the same key; (2) it no longer requires the key authority (KMS) to decrypt different ciphertexts for non-homomorphic operations during the entire outsourcing period, since KMS may risk broader attack surface and also become the bottleneck for efficiency.

Clustering Computation Phase. With all the converted records $\mathsf{Enc}_{pk_u}(t_{i,j})$ held by CS for $i \in [1,L]$, $j \in [1,m]$, the goal of this phase is to compute the cluster centroids $\mathsf{Enc}_{pk_u}(\mu_1), ..., \mathsf{Enc}_{pk_u}(\mu_k)$ 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 phase includes four steps, namely, Job Assignment, Map Execution, Reduce Execution, and Update Judgement. The last three steps are performed in an iterative fashion as shown in Fig. 2.

Step 1. Job Assignment. In this step, the CSP assigns various jobs to different computing nodes according to the cloud resource scheduling policy. First, CS selects τ minimum computing units (denoted by MCU) from $\{\mathrm{ES}_1, ..., \mathrm{ES}_\theta\}$ and $\{\mathrm{AS}_1, ..., \mathrm{AS}_\lambda\}$ respectively. In other words, $\mathrm{MCU} = \{\mathrm{ES}, \mathrm{AS}\}$. Each unit is able to perform cryptographic building blocks independently. We assume that each ES node provides adequate storage space and computation power for its assigned mission. The set $\{\mathrm{MCU}_1, ..., \mathrm{MCU}_\tau\}$ is divided into two disjoint sets, i.e., Map and Reduce. Without loss of generality, the Map has $\mathrm{MCU}_1, ..., \mathrm{MCU}_f$ while the Reduce has $\mathrm{MCU}_{f+1}, ..., \mathrm{MCU}_{f+k+1}$. Then CS divides D'_u into f uniformly distributed partitions $P_1, ..., P_f$, which are sent to their corresponding MCU

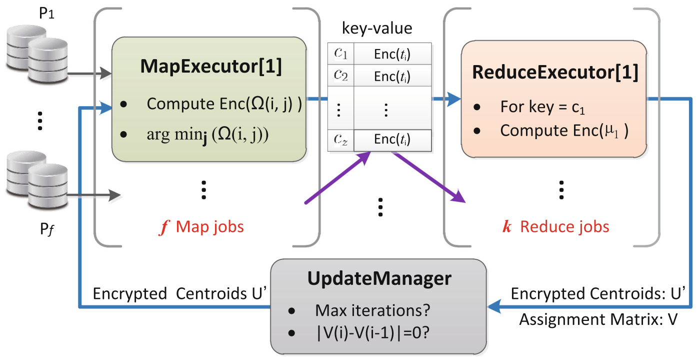


Fig. 2. PPOCM under Spark framework

nodes in Map. In this paper, we assume the k initial clusters are randomly selected from D'_u. Therefore, for $\forall i \in [1, f]$, Mapper$[i]$ knows the vector $U' = <\mu'_1, ..., \mu'_k>$, where $\mu'_j = <\mathsf{Enc}_{pk_u}(s_j), |c_j|>$, for $j \in [1, k]$.

Step 2. Map Execution. As for $1 \leq i \leq f$, given dataset P_i and centroids U' as inputs, Map$[i]$ independently computes encrypted Euclidean distances between data records and centroids, and outputs a key-value table, in which the key is the closest cluster id and the value is the encryption of corresponding record. Suppose P_i has z data records $t'_1, ..., t'_z$, in which $t'_j (j \in [1, z])$ is a m-dimension vector $<\mathsf{Enc}_{pk_u}(t_{j,1}), ..., \mathsf{Enc}_{pk_u}(t_{j,m})>$. The major steps are presented in Algorithm 4.

Algorithm 4. Map$(P_1, ..., P_f, U') \rightarrow \{T_1, ..., T_f\}$

```
Require: Mappers have $P_1, ..., P_f$ and centroids $U'$.
1: $\forall i \in [1, f]$, Mapper$[i]$:
2: for $j = 1$ to $z$ do
3:    for $h = 1$ to $k$ do
4:       Compute $d_h \leftarrow \mathrm{SSED}(t'_j, \mu'_h)$, where $d_h =< \mathsf{Enc}_{pk_u}(\Omega_{j,h}), |c_h| >$;
5:    end for
6:    Compute $d_{\min} \leftarrow \mathrm{SMkD}(d_1, d_2, ..., d_k)$, where $d_{\min} =< \mathsf{Enc}_{pk_u}(\Omega_{\min}), |c_{\min}| >$;
7:    Compute $key_j \leftarrow \mathsf{Indexof}(c_{\min})$ and $value_j \leftarrow t'_j$;
8: end for
9: Send $T_i = \{< key_1, value_1 >, ..., < key_z, value_z >\}$ to Reducer$[j]$, for $j \in [1, k]$;
```

Step 3. Reduce Execution. Upon receiving the key-value table from Map set, Reducer$[i]$ locates the item where *key* equals index of c_i, and computes the encryption of updated centroid μ'_i for $i \in [1, k]$. The output is μ'_i as long with an assignment vector $V_{L\times 1}$, in which $V[j] \in \{0, 1\}$ indicates whether j^{th} record belongs to c_i for $j \in [1, L]$. The major steps are presented in Algorithm 5.

Step 4. Update Judgement. CS takes cluster centers $U' = \{\mu'_1, ..., \mu'_k\}$ and assignment matrix $V_{L\times k} = <V_1^T, ..., V_k^T>$ from overall Reducers as inputs. Its

Algorithm 5. Reduce$(T_1, ..., T_f) \to \{W_1, ..., W_k\}$

```
Require: Reducers have $T_1, ..., T_f$.
 1: $\forall i \in [1, k]$, Reducer[$i$]:
 2: Initialize $s'_i \leftarrow \mathsf{Enc}_{pk_u}(0)$, $|c_i| \leftarrow 0$, $V_i \leftarrow \{0, ..., 0\}$;
 3: for $j = 1$ to $f$ do
 4:     for $h = 1$ to $z$ do
 5:         if $T_j[h].key == i$ then
 6:             Compute $s'_i \leftarrow \mathrm{SA}(T_{j,h}[w].value, s'_i[w])$, for $1 \leq w \leq m$; $|c_i| \leftarrow |c_i| + 1$;
 7:             Compute $V_i[(j-1) \cdot f + h] \leftarrow 1$;
 8:         end if
 9:     end for
10: end for
11: Send $W_i = \{\mu'_i, V_i\}$ to CS, where $\mu'_i =< \mathsf{Enc}_{pk_u}(s_i), |c_i| >$;
```

target is to determine whether the predefined termination conditions are satisfied. In PPOCM, there are two termination conditions: (1) the maximum iteration $\phi_{\max}$; (2) the matrix V does not vary any more. Therefore, CS not only needs to record the iteration count ϕ during updating clusters each time, but also judges whether the difference $\delta = V_{\phi+1} - V_\phi$ is zero matrix or $\phi \geq \phi_{\max}$. If either termination condition is met, the last phase is activated; otherwise, the cloud moves onto Step 2 to start next iteration, taking U' as inputs.

Result Retrieval Phase. To enable QC to obtain the final clusters, CS and KMS invoke SCT to compute $\{<\mathsf{Enc}_{pk_Q}(s_i), |c_i|>|i = 1, ..., k\}$, which are sent back to QC along with V. After that, QC is able to decrypt the result by his sk_Q. Since s_i and $|c_i|$ are not real center point, QC calculates the final centroids by $\mu_i \leftarrow \frac{s_i}{|c_i|}$, where $i \in [1, k]$. Furthermore, the assignment matrix V is in plain form, which does not require client-side decryption.

5 Security Analysis

We first analyze the security of the privacy-preserving building blocks. Since all parties are semi-honest, security in this model can be proven under "Real-vs.-Ideal" framework [23]: all adversarial behavior in the real world can be simulated by trusted party in the ideal world. We take SDC security proof as an example and the rests can be proved in a similar way.

Since there are two parties i.e., ES and AS, we need to prove SDC is secure not only against semi-honest adversary $\mathcal{A}_{ES}$ corrupting ES, but also against semi-honest adversary $\mathcal{A}_{AS}$ corrupting AS, respectively.

1. **Security Against ES.** The real world view of $\mathcal{A}_{ES}$ in SDC includes input $\{\psi_{i,a}, \psi_{i,b}\}$, a random r, ciphertexts $\{\mathsf{Enc}_{pk_u}(\Omega'_{i,a}), \mathsf{Enc}_{pk_u}(\Omega'_{i,b}),$ $\mathsf{Enc}_{pk_u}(\delta'_{a,b})\}$ and output sn. $\psi_{i,a}$ consists of $\mathsf{Enc}_{pk_u}(\Omega_{i,a})$ and $|c_a|$. From Eq. (1), $|c_a|^2$ is the denominator of $||t_i - \mu_a||^2$, whereas the numerator $\Omega_{i,a}$ is

encrypted under pk_u. Without the decryption key sk_u, $||t_i - \mu_a||^2$ is unknown to $\mathcal{A}_{ES}$. Likewise, $||t_i - \mu_b||^2$ is not revealed to $\mathcal{A}_{ES}$. Note that sn indicates the minimum of inputs, but it cannot be used to infer the actual distances directly. Thus, we can build a simulator $\mathcal{S}_{ES}$ in the ideal world by using encryptions of values randomly distributed in $\mathbb{Z}_N$ and sn is selected from $\{0, 1\}$ randomly. By the semantic security of the PKC-DD scheme, it's computationally difficult for $\mathcal{A}_{ES}$ to distinguish from the real world and the ideal world.

$$\text{Ideal}_{f,\mathcal{S}_{ES}}(\mathsf{Enc}_{pk_u}(\Omega_{i,j})) \overset{c}{\approx} \text{Real}_{\text{SDC},\mathcal{A}_{ES}}(\mathsf{Enc}_{pk_u}(\Omega_{i,j})),$$

where $i \in [1, L], j \in [1, k]$, and $\overset{c}{\approx}$ means computationally distinguishable.

2. **Security Against AS.** The real world view of $\mathcal{A}_{AS}$ in SDC includes input $\mathsf{Enc}_{pk_u}(\delta'_{a,b})$, blinded message $\delta'_{a,b}$, output sn and randomized $\Omega'_{i,a}, \Omega'_{i,b}$ during SA execution. Note that the blinding factors in SA are randomly distributed in $\mathbb{Z}_N$ and r in SDC is randomly selected in range according to Eq. (3), so we can build a simulator $\mathcal{S}_{AS}$ to simulate the ideal world view of $\mathcal{A}_{AS}$ by using random values in $\mathbb{Z}_N$. Even though $\mathcal{A}_{AS}$ is able to judge the sign of randomized distance, the actual distance and corresponding inputs are still unknown to $\mathcal{A}_{AS}$. Therefore, $\mathcal{A}_{AS}$ is not able to distinguish from the real world and the ideal world.

$$\text{Ideal}_{f,\mathcal{S}_{AS}}(\mathsf{Enc}_{pk_u}(\Omega_{i,j})) \overset{c}{\approx} \text{Real}_{\text{SDC},\mathcal{A}_{AS}}(\mathsf{Enc}_{pk_u}(\Omega_{i,j})),$$

where $i \in [1, L], j \in [1, k]$.

The PPOCM protocol includes 4 phases. In the first phase, D_i is encrypted under pk_i for $i \in [1, n]$. In the second phase, SCT subprotocol is invoked to transform ciphertexts. During the clustering phase, SA, SSED, SMkD are invoked as subroutines. At last, the encrypted centroids are converted by SCT. Note that the data held by parties without secret (i.e., CS and ES) key are encrypted while the data held by parties with secret key (i.e., KMS and AS) are randomized. Since the encryption scheme is semantically secure and blinding factors are randomly selected, nothing regarding the data content are revealed to the cloud servers or other owners. Matrix V is known to the server, but it is insufficient to deduce data records using the assignment membership. According to the Composition Theorem [23], the sequential compositions of those phases is secure. In conclusion, PPOCM is secure under the semi-honest model.

6 Performance Analysis

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

6.1 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 $2\texttt{Exp}+1\texttt{Mul}$. The cost of normal decryption is $1\texttt{Exp} + 1\texttt{Mul}$, while that of authority decryption is $2\texttt{Exp} + 2\texttt{Mul}$. The encryption and decryption of PKC-DD in [14] are claimed to be 3 times faster than BCP scheme in [15]. We stress that Phase 1 and Phase 2 of PPOCM protocol are executed only once. These overheads are amortized through a number of iterations. As for Clustering Computation Phase, the Map and Reduce steps undertake the most workload, the costs of which in one iteration are given in Table 2. It can be observed that the number of MCU in Map and Reduce sets are closely related to the outsourcing costs, that is, the larger the computing cluster is, the less overheads are exerted on each unit. This is because the Map and Reduce jobs can be parallelized and boosted under Spark.

Table 2. Computational and communication costs of clustering phase

Algorithm	Computational costs	Communication costs (in bits)
Map	$kz(10m + 13)\texttt{Exp} + kz(16m + 20)\texttt{Mul}$	$kz(6m + 9)\|N\|$
Reduce	$8fz\texttt{Exp} + 11fz\texttt{Mul}$	$6fz\|N\|$

6.2 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 compare our work with PPODC [11], because the system models are alike, and both protocols are constructed on public key cryptosystem and achieve the same security goals. We implemented all the outsourcing protocols using the Crypto++ 5.6.3 library and Spark framework. The key size $|N|$ is chosen to be 1536-bit, because to achieve the same security level with 1024-bit Paillier encryption used in PPODC and 1024-bit BCP encryption scheme in [20], $|N|$ of PKC-DD should be 500–600 bit more than RSA modulus [24].

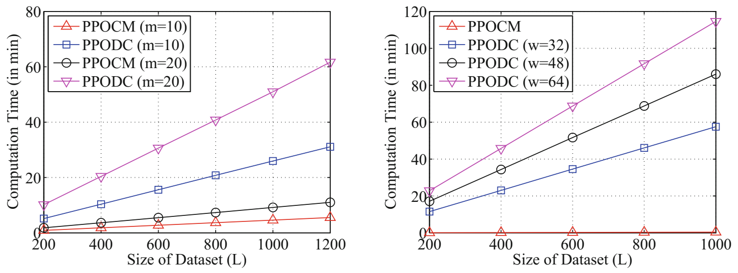
To facilitate comparisons, we use KEGG Metabolic Reaction Network dataset [11, 25]. The dataset includes 65554 instances and 29 attributes. Before clustering, all records are normalized into integers to prevent impacts of large unit values. Note that the first attribute is excluded from tests, since it is just the identifier of pathway. We assume there are 20 data owners in the system, each dataset of whom is randomly selected from KEGG dataset. They encrypt their data using own keys before outsourcing to the servers. There are three major factors that affect the outsourced clustering performance: (1) the ciphertext transformation scheme; (2) the number of clusters (k); (3) the number of parallelized MCUs (f).

We first evaluate the performance of transforming encrypted datasets under owners' keys into ciphertexts under the unified key. Table 3 shows the ciphertext transformation time for varying size of aggregated datasets (L) in our PPOCM scheme and KeyProd in [20]. It can be seen that the cloud running time grows with increasing value of L. However, our scheme executes about 4 times faster than KeyProd in that our underlying cryptosystem is much more efficient than theirs. Note that [20] aims at privacy-preserving arithmetic operations (i.e., addition and multiplication) rather than k-means algorithm, while PPODC scheme cannot be used to cluster data encrypted under multiple keys.

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

Protocol	$L = 2000$	$L = 4000$	$L = 6000$	$L = 8000$	$L = 10000$
PPOCM	11.6	23.2	35.3	46.5	57.9
KeyProd	43.9	87.9	138.9	175.3	219.4

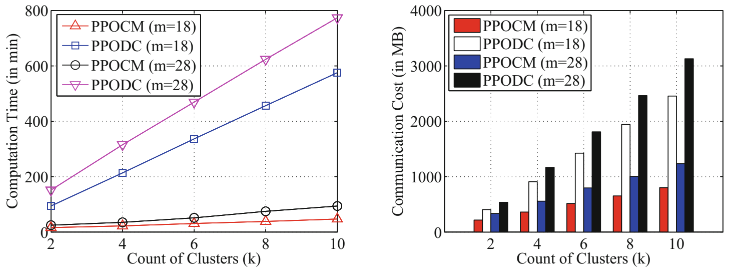
We then conduct tests on SSED and SMkD to evaluate the performance of the proposed secure building blocks, which utilize SA and SDC as primitives and are frequently invoked during k-means outsourcing. Figure 3(a) shows that the computation cost of both schemes increase with growth of dataset size, but SSED of PPOCM executes much faster. In addition, the increase of dimension size (m) has more impact on PPODC. As shown in Fig. (3)(b), it's observed that with growth of w, the computation time of SMkD in PPODC grows rapidly, where w denotes the bit length of plaintext message. The reason is that every ciphertext should be decomposed into a w-length vector of encrypted bits during execution of SMIN in [11], whereas in contrast, PPOCM's comparison operation is much more efficient by preserving the sign of randomized value.



(a) Cloud running time on SSED (b) Cloud running time on SMkD

Fig. 3. Experiment analysis on SSED and SMkD over samples from the real dataset

Next, we assess the overhead of complete protocols with varying k and m when $L = 2000$ and $f = 4$. PPOCM is compared with the optimized version of PPODC with 4 parallelized server pairs. The results are given in Fig. 4(a) and (b). It can be seen that both the computation time and communication cost grows almost linearly with the count of clusters. It is because more encrypted Euclidean distances need to be calculated and compared with increasing k. Our

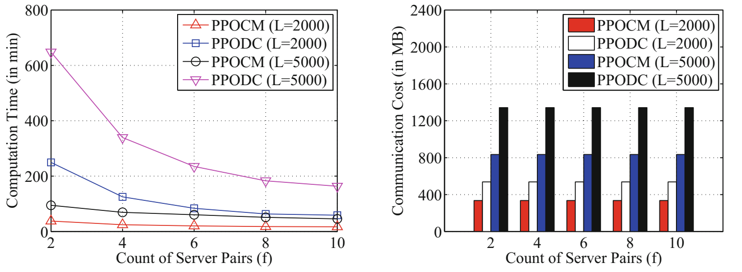


(a) Cloud running time (b) Cloud communication cost

Fig. 4. Experiment analysis with varying number of clusters (k) over the real dataset

method obviously outperforms PPODC. For instance, when $k = 6$ and $m = 28$, the execution time of PPODC is 469.1 min, whereas that of PPOCM is 51.3 min, almost 10 times faster. The communication cost of PPODC and PPOCM are 1809.1 MB and 796.1 MB, respectively. Furthermore, the growth of dimension size also increases the computational and communication overhead.

Moreover, we evaluate the overhead on cloud servers with varying f when $k = 2$. As shown in Fig. 5(c), the computation time decreases with the growth of f. It can be derived that: (1) the more parallelized MCUs or server-pairs participate in outsourcing, the shorter time it takes both schemes to complete the entire clustering task; (2) with growing size of dataset, it takes PPODC longer time to complete the same amount of work. The reason why PPOCM has better performance should be attributed to the excellent scaling capability of Spark engine and efficient primitiv. Figure 5(d) shows that the communication cost of both schemes remain invariable regardless of f. Though each server pair only handles partial jobs, the total amount of clustering task is fixed. Hence, the mount of transmitted data remain unchanged. In addition, the communication cost of PPOCM accounts for 62.2% of that of parallelized PPODC.



(a) Cloud running time (b) Cloud communication cost

Fig. 5. Experiment analysis with varying number of parallelized MCUs (f) over the real dataset

7 Related Work

There have been a lot of works on privacy-preserving distributed k-means clustering [7,8]. These works assume clustering task is performed through interactions among different data holders instead of third parties, resulting in different security requirements and design goals compared to our work.

As for outsourced clustering, Lin [9] proposed a privacy-preserving method for kernel k-means based on random linear transformation and perturbation of kernel matrix. But this scheme is neither fit for the standard k-means without kernel function, nor computes the cluster centers. Works in [10,18] leveraged fully homomorphic encryption to perform clustering on a single server and proposed to compare ciphertexts with trapdoor information, while their approach requires data owner's participation in each iteration, which affects the outsourcing performance. PPODC scheme proposed by Rao et al. [11] enables the cloud to perform clustering over the combined encrypted databases from multiple users, which is similar with our scheme. However, their solution does not support database encrypted under multiple keys. Besides, the overhead of secure comparison is too heavy since each inputs have to be decomposed into encrypted bits by calling SBD subroutine. As for arithmetic computation over data encrypted under multiple keys, López et al. [19] studied the FHE under multiple keys. Unfortunately, the efficiency of their scheme suffers from complex key-switching technique and heavy interactions among users. The recent work [20] utilized BCP encryption scheme with double trapdoor decryption [15] to address basic computations under multi-key setting, which yet cannot be used to compare ciphertexts. Besides, none of existing works have utilized big data analytic techniques.

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, comparison, etc. over encrypted data. Our protocol protects privacy of the combined database under the semi-honest model and requires no cloud client's participation. Another improvement is that the outsourced clustering works under big data processing framework, which significantly boosts the system performance. Experiments on real dataset show that our scheme is more efficient than existing approaches. As future work, we will focus on privacy protection and integrity verification techniques to withstand advanced attacks under malicious model during k-means outsourcing.

References

1. Hajjat, M., Sun, X., Sung, Y.E., Maltz, D., Rao, S., Sripanidkulchai, K., Tawarmalani, M.: Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. In: ACM SIGCOMM, pp. 243–254 (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. Eubank, S., Guclu, H., Kumar, V.S.A., Marathe, M.V., Srinivasan, A., Toroczkai, Z., Wang, N.: Modeling disease outbreaks in realistic urban social networks. Nature **429**, 180–184 (2004)
6. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**, 264–323 (1999)
7. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over arbitrarily partitioned data. In: ACM KDD (2003)
8. Jagannathan, G., Gehrke, J., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: KDD, pp. 593–599 (2005)
9. Lin, K.: Privacy-preserving kernel k-means outsourcing with randomized kernels. In: ICDM Workshop, pp. 860–866 (2013)
10. Liu, D., Bertino, E., Yi, X.: Privacy of outsourced k-means clustering. In: ASIA CCS, pp. 123–133 (2014)
11. Rao, F., Samanthula, B.K., Bertino, E., Yi, X., Liu, D.: Privacy-preserving and outsourced multi-user k-means clustering. In: IEEE Conference on Collaboration and Internet Computing, pp. 80–89 (2015)
12. Huang, Y., Lu, Q., Xiong, Y.: Collaborative outsourced data mining for secure cloud computing. J. Netw. **9**(9), 2655–2664 (2014)
13. Wong, W.K., Cheung, D.W., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted database. In: SIGMOD, pp. 139–152 (2009)
14. Youn, T.-Y., Park, Y.-H., Kim, C.H., Lim, J.: An efficient public key cryptosystem with a privacy enhanced double decryption mechanism. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 144–158. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_10
15. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_3
16. Kiltz, E., Malone-Lee, J.: A general construction of IND-CCA2 secure public key encryption. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 152–166. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40974-8_13
17. Galindo, D., Herranz, J.: On the security of public key cryptosystems with a double decryption mechanism. Inf. Process. Lett. **108**(2008), 279–283 (2008)
18. Liu, X., Jiang, Z.L., Yiu, S.M., et al.: Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks. In: The 11th International Conference on Mobile Ad-Hoc and Sensor Networks, pp. 124–133 (2016)
19. López, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)
20. Peter, A., Tews, E., Katzenbeisser, S.: Efficiently outsourcing multiparty computation under multiple keys. IEEE Trans. Inf. Forensics Secur. **8**(12), 2046–2058 (2013)

21. Ortiz, J.R., Oneto, L., Anguita, D.: Big data analytics in the cloud: spark on hadoop vs MPI/OpenMP on Beowulf. Procedia Comput. Sci. **53**(1), 121–130 (2015)
22. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S.: MLlib: machine learing in apache spark. J. Mach. Learn. Res. **17**(1), 1235–1241 (2015)
23. Goldreich, O.: The Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, Cambridge (2004)
24. Peralta, R.: Report on Integer Factorization (2001). http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1025_report.pdf
25. Naeem, M., Asghar, S.: KEGG Metabolic Reaction Network Data Set. The UCI KDD Archive (2011). https://archives.ics.uci.edu/ml/datasets/KEGG+Metabolic+Reaction+Network+(Undirected)