Computer and Information Science; Vol. 14, No. 2; 2021
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Efficient and Privacy-Preserving Multi-User Outsourced K-Means Clustering

Na Li[1], Lianguan Huang[2], Yanling Li[2] & Meng Sun[2]

[1] College of Information Science and Technology, Jinan University, China

[2] College of Cyber Security, Jinan University, China

Correspondence: Na Li, College of Information Science and Technology, Jinan University, China.

Received: January 22, 2021 Accepted: February 26, 2021 Online Published: March 10, 2021

doi:10.5539/cis.v14n2p26 URL: https://doi.org/10.5539/cis.v14n2p26

Abstract

In recent years, with the development of the Internet, the data on the network presents an outbreak trend. Big data mining aims at obtaining useful information through data processing, such as clustering, clarifying and so on. Clustering is an important branch of big data mining and it is popular because of its simplicity. A new trend for clients who lack of storage and computational resources is to outsource the data and clustering task to the public cloud platforms. However, as datasets used for clustering may contain some sensitive information (e.g., identity information, health information), simply outsourcing them to the cloud platforms can't protect the privacy. So clients tend to encrypt their databases before uploading to the cloud for clustering. In this paper, we focus on privacy protection and efficiency promotion with respect to k-means clustering, and we propose a new privacy-preserving multi-user outsourced k-means clustering algorithm which is based on locality sensitive hashing (LSH). In this algorithm, we use a Paillier cryptosystem encrypting databases, and combine LSH to prune off some unnecessary computations during the clustering. That is, we don't need to compute the Euclidean distances between each data record and each clustering center. Finally, the theoretical and experimental results show that our algorithm is more efficient than most existing privacy-preserving k-means clustering.

Keywords: k-means clustering, privacy protection, homomorphic encryption, locality sensitive hashing

1. Introduction

Clustering analysis is one of the most commonly used tasks in data mining area (Kriegel et al., 2009). It is worth noting that our clustering analysis is very different from clustering coefficient (Li, Y. , Shang, Y. , & Yang, Y. ,2017). The former is a method for data analysis, while the latter is an important concept in network topology, describing the degree to which vertices in a graph cluster together. The traditional clustering analysis methods mainly include hierarchical clustering, partitioning clustering, density-based clustering, grid-based clustering and model-based clustering algorithm. Actually, k-means clustering is a kind of partitioning clustering, it is popular owing to its simple characteristics. In addition, k-means algorithm is of great importance in various fields, including image retrieval (Yin & Zhang, 2017), machine learning, pattern recognition, social participatory sensing (Xing, Hu, Yu, Cheng & Zhang, 2017), knowledge discovery and text mining (MUstafi & Sahoo, 2019).

K-means clustering (Jain, Murty & Flynn, 1999) is a typical distance-based clustering algorithm, whose goal is to assign each data record into a cluster which has the shortest distance from it. As a result, similar records will be classified into the same group while different records into different groups. But as the amount of data increases, clients with limited storage and computation resources usually prefer to outsource their datasets and clustering task to the cloud service providers. However, the cloud platform can't be fully trusted, and they may want to gather clients' sensitive information. Thus, we prefer to encrypt the datasets before sending them to the cloud, on which performs clustering without getting any useful information in the end.

The original privacy-preserving k-means clustering schemes (Xia, Hua, Tong & Zhong, 2020; Yu, Luo, Chen & Ding, 2016; Chen, Wang, Zhang, Zhong & Chen, 2018; Su, Cao, Li, Bertiino, Lyu & Jin, 2017) protected the data privacy by adding noises. Then some scholars constructed privacy schemes for k-means clustering (Jiang, Guo, Jin, Lv, Wu, Liu, Fang, Yiu, & Wang, 2020; Rong, Wang, Liu, Hao & Xian, 2017; Yuan & Tian, 2017; Zou, Zhao, Shi, Wang, Peng, Ping & Wang, 2020) with different encryption cryptosystems to improve the accuracy of k-means clustering, but also leaded to low efficiency. To address this problem, we propose a new

privacy-preserving multi-user outsourced k-means algorithm, which is based on locality sensitive hashing (LSH) (Datar, Immorlica, Indyk & Mirrokni, 2004) for pruning off unnecessary computations in clustering. Our contributions are showed as follows:

1) We propose a LSH-based privacy-preserving multi-user outsourced k-means clustering (LSH-PPMOC) algorithm, whose main idea is to prune off the unnecessary computations during clustering task.

2) We combine the privacy-preserving techniques with LSH to guarantee the security of users' datasets and clustering results.

3) The experiments on real-world datasets show that we improve the clustering efficiency greatly.

The remainder of this paper is organized as follows. In Section 2, we discuss the existing related work. Section 3 presents some definitions and properties related to k-means clustering algorithm, Paillier cryptosystem and Locality Sensitive Hashing as a background. In Section 4, we discuss our proposed LSH-PPMOC solutions in detail. Also, we analyze the security guarantees and complexities of our solution in Section 5. Section 6 presents our experimental results on a real-world dataset under different parameter settings. Finally, we conclude the paper along with the scope for future research in Section 7.

2. Related Work

Privacy-preserving k-means clustering has been widely used in data mining.The private data mining algorithms can be categorized as, 1) data perturbation based, 2) secure multiparty computation (SMC) based and 3)cryptography technologies based. In perturbation, data records are randomized by adding noises, and the work by (Liu, Kargupta & Ryan, 2005) showed that the clustering results based on multiplicative perturbation only had below 5% error rate compared to the results on the original data. However, data perturbation cannot guarantee privacy in any formal sense (Liu, Giannella & Kargupta, 2006; Wong, Cheung, Kao & Mamoulis, 2009). For example, if an adversary gets a few data records in plaintext, he may recover the rest records though they are perturbated (Liu, Giannella & Kargupta, 2006).

Algorithms based on secure multiparty computation (SMC) (Goldreich, 2009) can preserve the security and privacy of users' data. There existed literatures utilizing SMC like (Lindell & Pinkas, 2008; Mohassel, Rosulek & Trieu, 2020; Upmanyu, Namboodiri, Srinathan & Jawahar, 2010), in which algorithms were proposed to perform distributed data mining without revealing private inputs of participants. In addition, Clifton et al. (Clifton, Kantarcioglu, Vaidya, Lin & Zhu, 2002) proposed that a relatively small set of cryptography primitive should be utilized to generate the SMC protocol. However, these solutions suffered from high levels of communication and computing cost. We can conclude that, though the private clustering algorithms based on SMC can provide better security protection, it is too costly to applied in practical applications (Jagannathan & Wright, 2005).

There are also literatures those propose privacy-preserving clustering algorithms using the semantically secure additive or multiplicative homomorphic encryption schemes. Liu et al. (Liu, Bertino & Yi, 2014) first leveraged fully homomorphic encryption technique (Gentry, 2009) to perform clustering outsourcing. However, the encryption scheme adopted in their scheme was not secure according to (Wang, 2015). Besides, this algorithm needed clients to participate and provide some information during each iteration, thus leading to heavy overhead on clients. To reduce the interaction with users, Almutairi et al. (Almutairi, Coenen & Dures, 2017) presented an efficient mechanism by using the concept of an Updatable Distance Matrix (UDM). But, their work revealed partial privacy to the cloud servers, such as the size of each cluster and the distances between data objects and centroids.

Similarly, Samanthulat et al.(Samanthula, Rao, Bertino, Yi & Liu, 2014) proposed a secure and outsourced k-means clustering scheme using the Paillier cryptosystem. Unfortunately, this scheme has a high computation cost because of using the bit array-based comparison. Then Rong et al. (Rong, Wang, Liu, Hao & Xian, 2017) put forward a privacy outsourced k-means clustering under multiple keys based on public key cryptosystem with double decryption (Youn, Park, Kim & Lim, 2005). However, the addition protocol in their scheme was not secure because the assistant server could extract the ratio of messages. To address this problem, (Zou, Zhao, Shi, Wang, Peng, Ping & Wang, 2020) proposed a highly secure outsourced k-means clustering scheme using BCP cryptosystem which had the additive homomorphic property.

To better the clustering performance in cloud computing environment, many scholars proposed MapReduce (Dean & Ghemawat, 2008) based k-means clustering schemes to handle large-scale dataset in parallel (Cui, Zhu, Yang, Li & Ji, 2014; Sardar & Ansari, 2020), but they all didn't consider privacy protection. Yuan et al. (Yuan & Tian, 2017) proposed a privacy-preserving scheme using a lightweight cryptosystem basing on the hardness of

learning with errors (LWE) (Brakerski, Gentry & Halevi, 2013), and incorporated MapReduce to improve the efficiency. However, the scheme was not fully outsourced and didn't support multiple users. Similarly, in (Rong, Wang, Liu, Hao & Xian, 2017), the clustering was executed under the Spark framework.

Besides, there are also other ways to improve efficiency for k-means clustering. Bhaskara et al. (Bhaskara & Wijewardena, 2018) proposed a variant of Locality sensitive hashing (LSH) (Indyk & Motwani, 1998) to speed up the clustering. Similarly, (Li, Wang, Wang, Hu, Li & Li, 2014) utilized the local sensitive property of LSH to prune off unnecessary computations during clustering and carried out the experiment with the help of MapReduce. However, the both schemes did not take privacy into consideration.

Inspired by literature (Li, Wang, Wang, Hu, Li & Li, 2014), we propose a new and efficient LSH-based privacy-preserving outsourced k-means clustering. We consider a scenario, in which there are two cloud service providers and multiple users. We explicitly assume the two cloud servers will never collude during clustering, then our proposed scheme protects all users' data confidentiality.

3. Preliminaries

In this section, we will first introduce the definition of typical k-means clustering. Then we give a brief overview of additive homomorphic Paillier cryptosystem and some basic cryptographic preliminaries for securely performing clustering. At last, we briefly introduce LSH which we use as the basis for our solution.

3.1 K-means Clustering

K-means clustering is an unsupervised clustering algorithm, and it is widely used in various application scenarios. Supposing given d-dimension data records $\boldsymbol{t}\_1, \dots, \boldsymbol{t}\_n$, the goal of k-means clustering algorithm is to divide these records into k disjoint groups. That is to say, the objects in the same group are similar, while objects in different groups have low similarity. We use $c\_1, \dots, c\_k$ to denote k clusters, and $\boldsymbol{\mu}\_1, \dots, \boldsymbol{\mu}\_k$, the corresponding centroids. To cluster one data record $\boldsymbol{t}\_i, i \in [1, n]$ into correct cluster, what we should do first is to compute the distance between it and k centroids $\boldsymbol{\mu}\_j$, j $\in [1, k]$. We use squared Euclidean distance here, which is given as

$$Dist(\boldsymbol{t}\_i, \boldsymbol{\mu}\_j) = \sum\_(m = 1)\^l \llbracket (\boldsymbol{t}\_i[m] - \boldsymbol{\mu}\_j[m]) \rrbracket \^2 . \quad (1)$$

Then we assign t_i to the cluster c_h, if t_i has the smallest distance with $\boldsymbol{\mu}\_h, h \in [1, k]$ among k distances.

A traditional k-means clustering has three stages, (1) Initialization; (2) Clustering; (3) Updating new centroids. For Stage (1), initial k records are selected randomly as cluster centroids $\boldsymbol{\mu}\_1, \dots, \boldsymbol{\mu}\_k$. In Stage (2), we first compute k distances between every one record $\boldsymbol{t}\_i$ and k centroids, then cluster them according to the distances. Later, new centroids $\boldsymbol{\mu}\_i'$ can be derived as the mean values of attributes of records belonging to c_i in Stage (3). If the given maximum iteration number is not reached or the cluster results differ from that getting before, next iteration will restart from Stage (2) to (3) with the new centroids, otherwise, the algorithm terminates.

3.2 The Paillier Cryptosystem

In this paper, we use the Paillier encryption (Paillier, 1999), which is a probabilistic asymmetric encryption scheme. Without loss of generality. Let $Enc(\cdot)$ and $Dec(\cdot)$ denote the encryption and decryption function under the Paillier cryptosystem and N denote the RSA modulus.

We denote the encryption scheme as a triple $\{KeyGen, Enc, Dec\}$.

$$\boldsymbol{KeyGen}(\boldsymbol{1\^\kappa}) \rightarrow (\boldsymbol{pk}, \boldsymbol{sk})$$

Choose two large prime numbers p and q which satisfy that $\gcd(pq, (p-1)(q-1)) = 1$.

Calculate $N = pq$, and $\lambda = lcm(p-1, q-1)$.

Randomly choose an integer $g \in Z\_(N\^2)$.

Check whether there exists $u = \llbracket (L(g\^\lambda \bmod N\^2)) \rrbracket \^(-1) \bmod N$ where function L is $L(\mu) = (\mu - 1)/N$. Then pk is (N, g) and sk is (λ, μ).

$$\boldsymbol{Enc}(\boldsymbol{pk}, \boldsymbol{x}) \rightarrow \boldsymbol{c}$$

Select a random value $r \in Z\_N\^*$ for the message x and the ciphertext is c=$g\^x\, r\^N \bmod N\^2$.

$$\boldsymbol{Dec}(\boldsymbol{sk}, \boldsymbol{c}) \rightarrow \boldsymbol{x}$$

Decrypt the message by $x = L(c\^\lambda \bmod \llbracket N\^2 \rrbracket)\mu \bmod n$.

Then, for any $a, b \in Z\_N$, the encryption scheme is additive homomorphic:

$$E(a+b) = E(a) \times E(b) \ mod\ N\text{^}2 = E(a+b \ mod\ N),$$

$$〚E(a)〛\text{^}b \ mod\ N\text{^}2 = E(a \cdot b \ mod\ N)$$

Where the symbol "$\times$" denotes the multiplication on ciphertexts, and "$\cdot$" means the multiplication on plaintext. Besides, we will omit the term $mod\ N\text{^}2$ from homomorphic operations in the rest of the paper.

3.3 Basic Cryptographic Primitives

In this part, we propose some cryptographic primitives as the basis of our methods.

(1) Secure Multiplication (SM) Protocol

Given that Alice holds $\langle E\_(\ 〚pk〛\ \_u\ )\ (x), E\_(\ 〚pk〛\ \_u\ )\ (y)\rangle$, and Bob holds $〚sk〛\ \_u$, the goal of Alice and Bob is to securely compute $E\_(\ 〚pk〛\ \_u\ )\ (x \cdot y)$. During the execution of SM, no information regarding the contents of x and y is revealed to Alice or Bob. More details are shown in Algorithm 1.

Algorithm 1: $SM(E\_(\ 〚pk〛\ \_u\ )\ (x), E\_(\ 〚pk〛\ \_u\ )\ (y)) \rightarrow E\_(\ 〚pk〛\ \_u\ )\ (x \cdot y)$

Input: Alice has $E\_(\ 〚pk〛\ \_u\ )\ (x), E\_(\ 〚pk〛\ \_u\ )\ (y)$, Bob has $〚sk〛\ \_u$

Output: $E\_(\ 〚pk〛\ \_u\ )\ (x \cdot y)$

1. Alice: (a) Picks any two different numbers $r\_x,\ r\_y \in Z\_N$ randomly.

(b) Computes $x' \leftarrow E\_(\ 〚pk〛\ \_u\ )\ (x) \times E\_(\ 〚pk〛\ \_u\ )\ (r\_x)$ $y' \leftarrow E\_(\ 〚pk〛\ \_u\ )\ (y) \times E\_(\ 〚pk〛\ \_u\ )\ (r\_y)$

(c) Sends x', y' to Bob

2. Bob: (a) Computes $h\_x \leftarrow Dec(\ 〚sk〛\ \_u, x\text{^}'),\ h\_y \leftarrow Dec(\ 〚sk〛\ \_u, y\text{^}'), h \leftarrow h\_x \cdot h\_y \ mod\ N,$

$h\text{^}' \leftarrow Enc(\ 〚pk〛\ \_u, h)$

(b) Sends $h\text{^}'$ to Alice

3. Alice: (a) Computes $s \leftarrow h\text{^}' \times 〚E\_(\ 〚pk〛\ \_u\ )\ (x)〛\text{^}(N - r\_y)$, $s' \leftarrow s \times 〚E\_(\ 〚pk〛\ \_u\ )\ (y)〛\text{^}(N - r\_x)$

(b) Gets $E\_(\ 〚pk〛\ \_u\ )\ (x \cdot y) \leftarrow s'\ 〚E\_(\ 〚pk〛\ \_u\ )\ (r\_x\ r\_y)〛\text{^}(N-1)$

(2) Secure Squared Euclidean Distance (SSED) Protocol

For k-means algorithm, there are many ways to compute the similarity scores between the data record $\boldsymbol{t}\_i, 1 \le i \le n$ and cluster centroid $\boldsymbol{\mu}\_j, 1 \le j \le k$, such as Euclidean distance, Cosine similarity and Jaccard coefficient. In this paper, we use squared Euclidean distance for its simplicity, denoted by $\|\boldsymbol{t}\_i - \boldsymbol{\mu}\_j\|\text{^}2$.

Note that $\boldsymbol{\mu}\_j$ is a vector, and its components may be rational numbers. However, the ring Z_N doesn't support rational division operation, so we use a new form of expression to represent the cluster center like (Rong, Wang, Liu, Hao & Xian, 2017). Let $\langle \boldsymbol{s}\_j, |c\_j| \rangle$ denotes the new form of cluster center, where $\boldsymbol{s}\_j$ and $|c\_j|$ represent the sum vector and the total number of records belonging to cluster c_j, respectively. Now we define $\Omega\_(i,j)$ as the scaled squared Euclidean distance between $\boldsymbol{t}\_i$ and $\boldsymbol{\mu}\_j$, which satisfies that $\|\boldsymbol{t}\_i - \boldsymbol{\mu}\_j\| = \sqrt{(\Omega\_(i,j))}/(|c\_j|)$. So $\Omega\_(i,j)$ can be calculated as follows:

$$\Omega\_(i,j) = 〚(\|\boldsymbol{t}\_i - \boldsymbol{\mu}\_j\| \cdot |c\_j|)〛\text{^}2 = \sum\_(l=1)\text{^}d 〚(|c\_j| \cdot \boldsymbol{t}\_i[l] - \boldsymbol{s}\_j[l])〛\text{^}2, \quad (2)$$

where $1 \le i \le m, 1 \le j \le k$, m and d are the total number of objects and dimension size, respectively. Taking $E\_(\ 〚pk〛\ \_u\ )\ (\boldsymbol{t}\_i)$ and $\langle E\_(\ 〚pk〛\ \_u\ )\ (\boldsymbol{s}\_i), E\_(\ 〚pk〛\ \_u\ )\ (|c\_j|)\rangle$ as inputs, the output of SSED protocol is $\langle E_(\ 〚pk〛\ _u\ )\ (\Omega_(i,j)), E_(\ 〚pk〛\ _u\ )\ (|c_j|)\rangle$which is actually the distance between record $\boldsymbol{t}\_i$ and the centroid of j_th cluster. We present this protocol in Algorithm 2.

Algorithm 2: $SSED$

Input: Alice has $E\_(\llbracket pk \rrbracket\_u)(\boldsymbol{t}\_i)$, $E\_(\llbracket pk \rrbracket\_u)(\boldsymbol{\mu}\_j) = \langle E\_(\llbracket pk \rrbracket\_u)(\boldsymbol{s}\_i), E\_(\llbracket pk \rrbracket\_u)(|c\_j|)\rangle, i \in [1,n], j \in [1,k]$,

Bob has $\llbracket sk \rrbracket\_u$

$E\_(\llbracket pk \rrbracket\_u)(\Omega\_(i,j)) = E\_(\llbracket pk \rrbracket\_u)(0)$

for $l = 1$ *to* d

Alice and Bob jointly compute $\Gamma \leftarrow SM(E\_(\llbracket pk \rrbracket\_u)(|c\_j|), E\_(\llbracket pk \rrbracket\_u)(t\_i[l]))$,

$\Gamma\text{^}' \leftarrow \Gamma \times E\_(\llbracket pk \rrbracket\_u)(s\_j[m])\text{^}(N-1), \Gamma\text{^}'' \leftarrow SM(\Gamma\text{^}', \Gamma\text{^}')$

$E\_(\llbracket pk \rrbracket\_u)\ \llbracket(\Omega \rrbracket\_(i,j)) = E\_(\llbracket pk \rrbracket\_u)\ \llbracket(\Omega \rrbracket\_(i,j)) \times \Gamma\text{^}''$

end for

Alice and Bob jointly compute $E\_(\llbracket pk \rrbracket\_u)(|c\_j|) \leftarrow \mathrm{SM}(E\_(\llbracket pk \rrbracket\_u)(|c\_j|), E\_(\llbracket pk \rrbracket\_u)(|c\_j|))$

Output $(E\_(\llbracket pk \rrbracket\_u)\ \llbracket(\Omega \rrbracket\_(i,j)), E\_(\llbracket pk \rrbracket\_u)(|c\_j|))$

(3) Secure Comparison (SC) Protocol

Given that Alice has two encrypted values $E\_(\llbracket pk \rrbracket\_u)(x), E\_(\llbracket pk \rrbracket\_u)(y)$, and Bob has sk_u, the goal of SC protocol is to securely compare x and y without knowing both of them. To be specific, Alice generates an arbitrary value r firstly, and then encrypts it as $E\_(pk\_u)(r)$. After that, Alice computes $E\_(\llbracket pk \rrbracket\_u)(t\_0) = E\_(\llbracket pk \rrbracket\_u)(x + r)$ and $E\_(\llbracket pk \rrbracket\_u)(t\_1) = E\_(\llbracket pk \rrbracket\_u)(y + r)$ utilizing the additive homomorphic property of the Paillier cryptosystem. Alice flips a coin b where $b = 0/1$, then sends two ciphertexts in the order of $E\_(\llbracket pk \rrbracket\_u)(t\_b), E\_(\llbracket pk \rrbracket\_u)(t\_(1-b))$ to Bob. Bob decrypts with sk_u and gets t_b and $t\_(1-b)$. If $t\_b < t\_(1-b)$, Bob sends $E\_(\llbracket pk \rrbracket\_c)(\gamma) = E\_(\llbracket pk \rrbracket\_c)(1)$ to Alice, else sends $E\_(\llbracket pk \rrbracket\_c)(\gamma) = E\_(\llbracket pk \rrbracket\_c)(0)$. This protocol will return $E\_(\llbracket pk \rrbracket\_c)(\gamma)$ if $b = 0$, otherwise $SBN(E\_(\llbracket pk \rrbracket\_c)(\gamma))$.

(4) Privacy-preserving Minimum (PMIN) Protocol

In this protocol, we aim at comparing two squared Euclidean distances and output the minimum one. To achieve this, we use the privacy-preserving maximum protocol proposed by Liu et al. (Liu, Lu, Ma, Chen & Qin, 2015). However, we will give a slight change to it to transfer the maximum value to the minimum value.

To be specific, Alice has two encrypted squared Euclidean distances between one record $\boldsymbol{t}\_i, i \in [1,m]$ and two clusters c_a and c_b,
$E\_(\llbracket pk \rrbracket\_u)(d\_a) = (E\_(\llbracket pk \rrbracket\_u)(\Omega\_a), E\_(\llbracket pk \rrbracket\_u)(|c\_a|)), E\_(\llbracket pk \rrbracket\_u)(d\_b) = (E\_(\llbracket pk \rrbracket\_u)(\Omega\_b), E\_(\llbracket pk \rrbracket\_u)(|c\_b|))$, their secrets are
$E\_(\llbracket pk \rrbracket\_u)(\lambda\_a) = E\_(\llbracket pk \rrbracket\_u)(a), E\_(\llbracket pk \rrbracket\_u)(\lambda\_b) = E\_(\llbracket pk \rrbracket\_u)(b)$. Bob has sk_u. After finishing the protocol, Alice will get $E\_(\llbracket pk \rrbracket\_u)(d\_min)$ whose corresponding distance is relatively small, along with $E\_(\llbracket pk \rrbracket\_u)(\lambda\_min)$.

To get d_min, Alice first computes
$E\_(\llbracket pk \rrbracket\_u)(\xi\_1) = SM(E\_(\llbracket pk \rrbracket\_u)(\Omega\_a), E\_(\llbracket pk \rrbracket\_u)(|c\_b|\text{^}2)), E\_(\llbracket pk \rrbracket\_u)(\xi\_2) = SM(E\_(\llbracket pk \rrbracket\_u)(\Omega\_b), E\_(\llbracket pk \rrbracket\_u)(|c\_a|\text{^}2))$, then
$E\_(\llbracket pk \rrbracket\_u)(H\_1) = \llbracket E\_(\llbracket pk \rrbracket\_u)(\xi\_1) \rrbracket\text{^}2 \times E\_(\llbracket pk \rrbracket\_u)(1) = E\_(\llbracket pk \rrbracket\_u)(2\Omega\_a \cdot |c\_b|\text{^}2 + 1), E\_(\llbracket pk \rrbracket\_u)(H\_2) = E\_(\llbracket pk \rrbracket\_u)(2\Omega\_b \cdot |c\_a|\text{^}2)$. After that, Alice chooses some random values $R, r\_1, r\_2, r\_3, r\_4 \in Z\_N\text{^}*, \theta \in G, 2\text{^}L < N\text{^}2$. If θ is an odd number, Alice computes
$C\_1 = \llbracket(E\_(\llbracket pk \rrbracket\_u)(H\_1))\rrbracket\text{^}R \times \llbracket(E\_(\llbracket pk \rrbracket\_u)(H\_2))\rrbracket\text{^}(N-R) \times g\text{^}(2\text{^}L)\ \llbracket r\_4 \rrbracket\text{^}N = E\_(\llbracket pk \rrbracket\_u)(R(H\_1 - H\_2) + 2\text{^}L)$,
$C\_2 = E\_(\llbracket pk \rrbracket\_u)(\Omega\_a - \Omega\_b + r\_1), C\_3 = E\_(\llbracket pk \rrbracket\_u)(|c\_a|\text{^}2 - |c\_b|\text{^}2 + r\_2), C\_4 =$

$E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_a-\lambda\_b+r\_3\ )$. else if θ is even, Alice computes $C\_1=\ [\![(E\_(\ [\![pk]\!]\ \_u\ )\ (H\_2\ ))]\!]\ \hat{}\ R\times\ [\![(E\_(\ [\![pk]\!]\ \_u\ )\ (H\_1\ ))]\!]\ \hat{}\ (N-R)\times g\hat{}(2\hat{}L\ )\ [\![r\_4]\!]\ \hat{}\ N=$ $E\_(\ [\![pk]\!]\ \_u\ )\ (R(H\_2-H\_1\ )+2\hat{}L\ ), C\_2=E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_b-\Omega\_a+r\_1), C\_3=$ $E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_b\ |\hat{}2-\ [\![|c\_a\ |]\!]\ \hat{}\ 2+r\_2), C\_4=E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_b-\lambda\_a+r\_3\ )$. Then C_1, C_2, C_3 and C_4 are sent to Bob.

After receiving C_1, C_2, C_3 and C_4, Bob generates $r\_5, r\_6\in Z\_N$ randomly, and decrypts C_1 to get M. If $M>2\hat{}L$, assuming $\alpha=0$, Bob computes $E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)$, $A=E\_(\ [\![pk]\!]\ \_u\ )\ (0)$, $B=E\_(\ [\![pk]\!]\ \_u\ )\ (0)$, $C=E\_(\ [\![pk]\!]\ \_u\ )\ (0)$, else if $M<2\hat{}L$,assuming $\alpha=1$, Bob computes $E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha), A=C\_2\times r\_5\hat{}N$, $B=C\_3\times r\_6\hat{}N$, $C=C\_4\times r\_7\hat{}N$. After that, Bob sends $E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha),\ A, B, C$ back to Alice.

For Alice, if θ is odd, she will compute $E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_min\ )=E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_b\ )\times A\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_1\ )$, $E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\mathrm{c}]\!]\ \_min\ |)=$

$E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_b\ |\hat{}2\ )\times C\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_2\ )$,

$E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_min)=E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_b\ )\times B\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_3\ )$. Else if θ is even, Alice will compute

$E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_min\ )=$

$E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_a\ )\times A\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_1\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\mathrm{c}]\!]\ \_min\ |)=$

$E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_a\ |\hat{}2\ )\times C\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_2\ )$,

$E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_min)=E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_b\ )\times B\times\ [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\alpha)]\!]\ \hat{}\ (N-r\_3\ )$. Then the output of this protocol is $[\![E\_(\ [\![pk]\!]\ \_u\ )\ (d]\!]\ \_min)=(E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_min\ ),\ E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_min\ |))$, $E\_(\ [\![pk]\!]\ \_u\ )\ (\lambda\_min)$.

Algorithm 3: ESDC

Input: Alice has $(E\_(\ [\![pk]\!]\ \_u\ )\ [\![(\Omega]\!]\ \_a), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_a\ |)), (E\_(\ [\![pk]\!]\ \_u\ )\ [\![(\Omega]\!]\ \_b), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_b\ |))$, $[\![sk]\!]\ \_c$, Bob has $[\![sk]\!]\ \_u$, $[\![pk]\!]\ \_c$

Output: $E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_2)$

Alice and Bob jointly compute:

$$E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_b\ |]\!]\ \hat{}\ 2\ )\leftarrow SM(E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_b\ |), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_b\ |)),$$

$$E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_a\ |]\!]\ \hat{}\ 2\ )\leftarrow SM(E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_a\ |), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_a\ |))$$

$E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_a)\leftarrow SM(E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_a\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_b\ |]\!]\ \hat{}\ 2\ ))$, $E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_b)\leftarrow$ $SM(E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_b\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_a\ |]\!]\ \hat{}\ 2\ ))$

$E\_(\ [\![pk]\!]\ \_u\ )\ (\Theta)\leftarrow SM(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_b\ |]\!]\ \hat{}\ 2\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c\_a\ |]\!]\ \hat{}\ 2\ ))$, $E\_(\ [\![pk]\!]\ \_u\ )\ (\Theta')\leftarrow$ $[\![(SM(E\_(\ [\![pk]\!]\ \_u\ )\ (\Theta), E\_(\ [\![pk]\!]\ \_u\ )\ (\varepsilon)))]\!]\ \hat{}\ (N-2)$

$E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_1)\leftarrow SC(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_a\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_b\ ))$

Alice decrypts $\lambda\_1\leftarrow Dec(\ [\![sk]\!]\ \_c, E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_1\ ))$

if $\lambda\_1=1$, computes $E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_2)=$

$SC(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_b)\times\ [\![(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega\hat{}']\!]\ \_a))]\!]\ \hat{}\ (N-1), E\_(\ [\![pk]\!]\ \_u\ )\ (\Theta'))$

else, computes $E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_2)=$

$SC(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega']\!]\ \_a)\times\ [\![(E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![\Omega\hat{}']\!]\ \_b))]\!]\ \hat{}\ (N-1), E\_(\ [\![pk]\!]\ \_u\ )\ (\Theta'))$

(5) Enhanced Secure Distance Comparison (ESDC) Protocol

However, the PMIN protocol can't meet the requirement we need for pruning strategies in section 3.5.3, so we design a protocol, enhanced secure distance comparison (ESDC) protocol, the goal of which is to compare the absolute value of the difference between two encrypted Euclidean distance, $〚E\_(〚pk〛\_u)(d〛\_a)$, $〚E\_(〚pk〛\_u)(d〛\_b)$ and an encrypted constant value ε, where $〚E\_(〚pk〛\_u)(d〛\_a) = (E\_(〚pk〛\_u)(\Omega\_a), E\_(〚pk〛\_u)(|c\_a|))$, $〚E\_(〚pk〛\_u)(d〛\_b) = (E\_(〚pk〛\_u)(\Omega\_b), E\_(〚pk〛\_u)(|c\_b|))$. As a result, we can get λ, where $\lambda = 1$ means $|d\_a - d\_b| > \varepsilon$, and vice versa. We show it in Algorithm 3.

(6) Privacy-Preserving Minimum Among k Distances ($PMIN\_k$) Protocol

The goal of $PMIN\_k$ protocol is to gain the encrypted minimum distance and its corresponding secret from k encrypted square Euclidean distances. Assume Alice has k distances in ciphertext along with their secrets $,\{\langle E\_(〚pk〛\_u)(d\_1), E\_(〚pk〛\_u)(1)\rangle, \dots, \langle E\_(〚pk〛\_u)(d\_k), E\_(〚pk〛\_u)(k)\rangle\}$, where $〚E\_(〚pk〛\_u)(d〛\_i) = (E\_(〚pk〛\_u)(\Omega\_i), E\_(〚pk〛\_u)(|c\_i|)), 1 \le i \le k$ and Bob has the corresponding secret key sk_u. To get the minimum value, Alice and Bob need to execute PMIN with two inputs in a sequential fashion. This process is straightforward, so we omit it here.

3.4 Locality Sensitive Hashing

Locality sensitive hashing (LSH) was introduced in the seminal work (Indyk & Motwani, 1998), which gave an efficient algorithm for nearest neighbor searching in high dimensional space. The main idea of LSH is to use a few special hash functions, called function family to map objects into different buckets, and the objects that are close to each other may be mapped into the same bucket (collision) with higher probability than objects those are far apart. Assume S is the domain of objects, and D is the distance measure between objects.

Definition 1. *A function family* $\mathcal{H} = \{h: S \to U\}$ *is called* (r_1, r_2, p_1, p_2)*-sensitive for D if for any* $a, b \in S$*, it holds*

if $\|a - b\| \le r\_1$*, then* $p[h(a) = h(b)] \ge p\_1$

if $\|a - b\| \ge r\_2$*, then* $p[h(a) = h(b)] \le p\_2$

where p_1, p_2 are both probability value, and $0 < r\_1 \le r\_2, 0 \le p\_2 < p\_1 \le 1$.

We know different LSH families are adopted for different distance functions. (Datar, Immorlica, Indyk & Mirrokni, 2004) has proposed LSH families for l_p norms based on p-stable distributions. If p is 1, it is Manhattan distance, while p is equal to 2, it is Euclidean distance. As we use Euclidean distance in this paper, we use the hash function in Eq. (3).

$$\boldsymbol{h}\_(\boldsymbol{a}, \boldsymbol{b})(\boldsymbol{v}) = \lfloor (\boldsymbol{a} \cdot \boldsymbol{v} + \boldsymbol{b})/\boldsymbol{r} \rfloor \tag{3}$$

3.5 LSH Based Privacy-Preserving Multi-User Outsourced K-Means Clustering for Massive Datasets

This section discusses a new LSH based privacy-preserving multi-user outsourced k-means clustering algorithm (LSH-PPMOC) which includes two optimization strategies for large-scale high-dimensional data clustering. Our target is to minimize the cost in the process for k-means clustering while we won't reveal any information about users' data records or distances between data records and centroids. Firstly, we use a LSH function family which is determined by all users to get the data skeleton, where the similar points are reduced to the same bucket. Then we can save unnecessary computations or comparisons between k encrypted distances during the work through performing clustering on the data skeleton and utilizing efficient strategies.

3.5.1 Data Skeleton

In this section, we will firstly briefly introduce data skeleton proposed by Li et al. (Li, Wang, Wang, Hu, Li & Li, 2014), and then illustrate what we can do on data skeleton.

Each element of data skeleton is a 2-tuple (p_r, L_p), where p_r represents a list of points in L_p. To get data skeleton, we first use m hash functions to map the original datasets into different buckets, where the bucket value is an m-dimension value as the result of m hash functions. Then we select the center point of each bucket as a representative data point p_r. Initially, we set L_p for each representative data point to null. The distances

between p_r and all other points in this bucket are computed, and if the distance is smaller than d, the according point will be added into L_p, where d is a user-defined threshold. For the point which further away from p_r is also an element of data skeleton, whose corresponding L_p is null. An example is given as Figure 1. It is worth noting that all the data skeletons in each bucket form a hash table(HT).

It is well known that, as the dataset grows, the cost of distance computing between each point and centroid pair becomes unbearable, especially when the number of clusters is large. And we know that the points in one data skeleton are close in distance, and they may belong to the same cluster with high probability. Then we might save some unnecessary calculations based on this.

We demonstrate that in Figure 2. We say that c_1 and c_2 are two center points, p_1 and p_2 are two different data records, r_1, r_2 are the distances between p_1 and c_1, c_2, while r_1', r_2' are distances of p_2, d is the distance between p_1 and p_2. If we know that r_1 is much smaller than r_2, and d is near to zero, it is very likely that $r\_1^{\wedge\prime} < r\_2'$, which means that p_2 shares the same cluster with p_1. We can generalize it to Theorem 1.

Theorem 1. ***Given c_1 and c_2 as two center points, p_1 and p_2 are two points with the distance d. r_1, $r\_1^{\wedge\prime}$, r_2 and r_2' are distances between p_1, p_2 and c_1, c_2 respectively. If $r\_1 < r\_2$ and $r\_2 - r\_1 > 2 * d$, then it holds that $r\_1^{\wedge\prime} < r\_2'$.***

The proof was given in (Li, Wang, Wang, Hu, Li & Li, 2014), and we wouldn't go into details here.

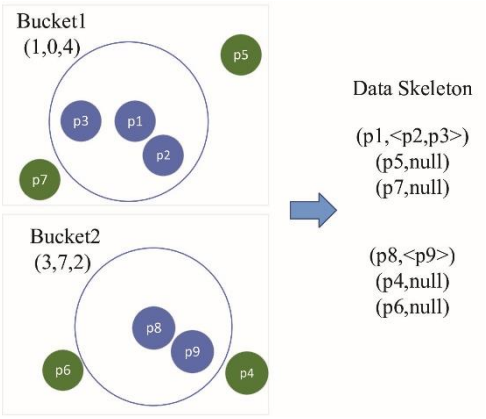


Figure 1. Data Skeleton

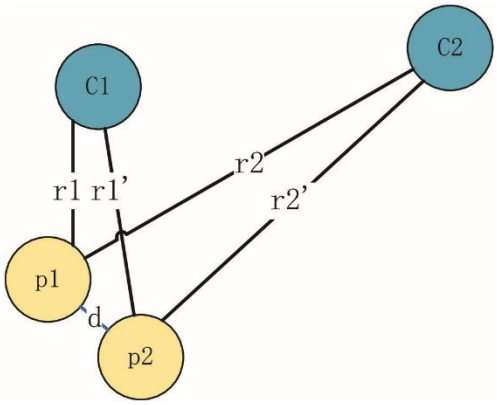


Figure 2. Points and Centers

3.5.2 Privacy-Preserving Pruning Strategies Using LSH

In this section, we explain how we make use of the low bound property of LSH to prune off the unnecessary computations during privacy k-means clustering.

There are two pruning strategies, and both of them are based on Theorem 1. The first one aims at reducing the number of data records that need to find the nearest centers. For each representative point p_r in one bucket, it represents the points in L_p, and the distance between any p_i and p_r is less than d. We first compute k distance between p_r and k centroids $\{c\_1, \dots, c\_k\}$ using SSED protocol. Let $d(p\_r, c\_1)$ and $d(p\_r, c\_2)$ be the smallest and the second smallest distances among k distances. If $d(p\_r, c\_2) - d(p\_r, c\_1) \geq 2d$, we can say that all points p_i represented by p_r have a shortest distance with c_1. In this case, we need not compute the distances between p_i and k centroids in this iteration.

The second strategy is to reduce the number of centroids to be compared for each point p_i in L_p. Let's think about a situation, we have computed k distances for p_r, represented by $r\_1, \dots, r\_k$ and r_1 is the smallest distance, then p_i is a point represented by p_r. When we tend to assign p_i to a proper cluster, we need to compare $r\_i - r\_1$ and $2d, i \in [2, k]$. Only the cluster c_i which holds $r\_i - r\_1 \leq 2d$ will be marked, and added into a set closeSet. That is to say, for p_i, we only need compute distances between it and centroids in the closeSet which is obviously the subset of k centroids.

In Algorithm 4, we give the pseudo-code of privacy-preserving LSH-based multi-user outsourced k-means clustering.

Algorithm 4. Privacy-Prserving LSH-Based Multi-User Outsourced K-Means Clustering

Input: $E\_(\ [\![pk]\!]\ \_u\ )\ (D) = \{E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{p}\_i\ )\}, i \in [1, m], C = \{E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{\mu}\_j\ )\} = \langle E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}\_j\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (\ [\![|c]\!]\ \_j\ |)\rangle, j \in [1, k]$

Combined blinded hash table (HT), $E\_(\ [\![pk]\!]\ \_u\ )\ (\epsilon)$

for $\boldsymbol{p}\_r$ in each bucket in HT:

for each center $\boldsymbol{\mu}\_j$ in C:

compute the distance between $\boldsymbol{p}\_r$ and $\boldsymbol{\mu}\_j$

$$[\![dis]\!]\ \_j = (E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_(p\_r, j)\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_j\ |)) = SSED(E\_(\ [\![pk]\!]\ \_u\ )\ (p\_r), E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{\mu}\_j)),$$

get closest encrypted center $\boldsymbol{\mu}\_j'$ for $\boldsymbol{p}\_r$ in C

$$\{min, label\} = \{(E\_(\ [\![pk]\!]\ \_u\ )\ (\Omega\_(p\_r, j\hat{}')\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_(j\hat{}')\ |)), label\} \leftarrow [\![PMIN]\!]\ \_k\ (E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{p}\_r\ ), C)$$

$$[\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j') = [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j') \times E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{p}\_r\ ), E\_(\ [\![pk]\!]\ \_u\ )\ |c\_j'\ | = E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_j'\ |) \times E\_(\ [\![pk]\!]\ \_u\ )\ (1)$$

Set $closeSet = null$

for $j \in [1, k]$ and $j \neq j'$

CS and AS jointly compute $E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_1) \leftarrow ESDC(min,\ [\![dis]\!]\ \_j, E\_(\ [\![pk]\!]\ \_u\ )\ (d))$

CS gets $\lambda\_1 \leftarrow Dec(\ [\![sk]\!]\ \_c, E\_(\ [\![pk]\!]\ \_c\ )\ (\lambda\_1\ ))$

if $\lambda\_1 = 0$, compute $closeSet = closeSet + \{E\_(\ [\![pk]\!]\ \_u\ )\ (\mu\_j)\}$

if $closeSet = null$

for each $\boldsymbol{p}\_i$ in L_p do

$$[\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j') = [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j') \times E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{p}\_i\ ), E\_(\ [\![pk]\!]\ \_u\ )\ |c\_j'\ | = E\_(\ [\![pk]\!]\ \_u\ )\ |c\_j'\ | \times E\_(\ [\![pk]\!]\ \_u\ )\ (1)$$

else

for each $\boldsymbol{p}\_i$ in L_p do

get closet center μ_j'' for each record p_i from $closeSet$

$$[\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j'') = [\![E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{s}]\!]\ \_j'') \times E\_(\ [\![pk]\!]\ \_u\ )\ (\boldsymbol{p}\_i\ ), E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_j''\ |) = E\_(\ [\![pk]\!]\ \_u\ )\ (|c\_j''\ |) \times E\_(\ [\![pk]\!]\ \_u\ )\ (1)$$

Then CS and AS will update the bitmap matrix according to $abel$, and they cooperatively judge the termination conditions.

4. The Proposed Solution

4.1 System Model

In our setting, we assume that there are two types of entities: users and cloud service providers as given in Figure 3. There are n users denoted by $U\_1, \ldots, U\_n$. Suppose that U_i holds a dataset T_i with d-dimension m_i data records, $\sum\_(i = 1)\hat{}n$▒$m\_i = m$. We use a Computation Server (CS) and an Assistant Server (AS) to excute the k-means clustering task, which are both semi-honest and don't collude. Let AS generate two public-secret key pair $(\ [\![pk]\!]\ \_u,\ [\![sk]\!]\ \_u)$, $(\ [\![pk]\!]\ \_c,\ [\![sk]\!]\ \_c)$ based on the Paillier cryptosystem and the public key $[\![pk]\!]\ \_u$ is sent to all users and CS, while AS remains $[\![sk]\!]\ \_u$, while $(\ [\![pk]\!]\ \_c,\ [\![sk]\!]\ \_c\ )$ is sent to CS.

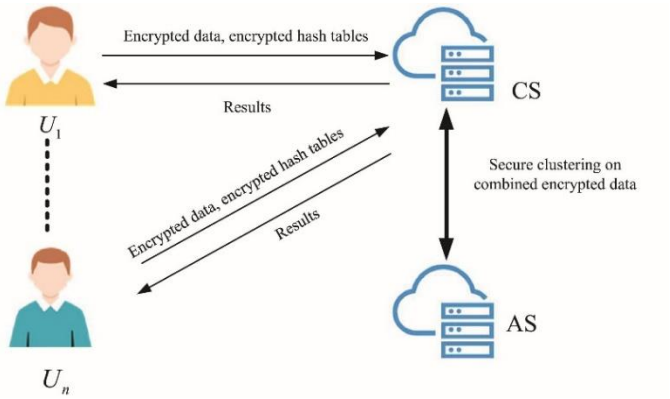


Figure 3. System architecture

When clustering starts, all users first use the same LSH function family to get their own hash table (Data Skeleton) , and then blind the bucket values through MD5 to preserve the privacy of buckets. After that, each user U_i encrypts his/her own dataset attribute-wise using $〚pk〛\_u$, and sends hash table and encrypted dataset to CS. Then CS will first aggregate all received hash tables into one hash table (HT), in this process, buckets with the same value will be merged. Next, CS performs k-means clustering on ciphertexts with the help of AS. When the clustering results do not change anymore or a predefined number of the iteration is reached, the clustering is finished. The results will be send to users.

4.2 The Proposed LSH-PPMOC Protocol

In this section, we will discuss our proposed LSH-PPMOC algorithm. There are three steps in our proposed algorithm. Specifically, our algorithm is composed of the following: 1) Data uploading, 2) LSH-based pruning clustering, 3) Termination.

Data uploading. At the beginning, AS generates two public/private key pairs ($〚pk〛\_c$, $〚sk〛\_c$), ($〚pk〛\_u$, $〚sk〛\_u$) based on the Paillier cryptosystem, then $〚pk〛_u$is sent to all users and CS, $〚sk〛\_u$ remains secret, while ($〚pk〛\_c$, $〚sk〛\_c$) are sent to CS for decrypting intermediate results. Then each user generates their own key pair ($〚pk〛\_v$, $〚sk〛\_v$) for decrypting the final clustering centroids. Before clustering, all users agree on a hash function family, and generate their own hash tables. The table is made up of different buckets. Unfortunately the bucket value is gotten from users' original data records, it needs to be blinded by a one-way hash function (SHA5, MD5 and so on) to protect information about data records from leaking. The hash table will be sent to CS. Besides, each user encrypts their dataset using $〚pk〛\_u$ before sending to CS.

LSH-based pruning clustering. After receiving encrypted hash tables, and encrypted original datasets, CS will aggregate tables firstly and then perform the clustering work cooperating with AS. This process can be divided into four steps.

(1) CS chooses initial k centroids randomly from all encrypted objects for k clusters, denoted as $E\_(〚pk〛\_u)(\mu\_i) = \langle E\_(〚pk〛\_u)(s\_i), E\_(〚pk〛\_u)(|c\_i|)\rangle, (i \in [1,k])$, where s_i equals $p\_j, j \in [1,m]$ and $E\_(〚pk〛\_u)(|c\_i|) = E\_(〚pk〛\_u)(1)$.

(2) For every representative point p_r in each bucket, CS and AS perform original privacy-preserving k-means clustering on them. In other words, they compute k squared Euclidean distances as $r\_1, \ldots, r\_k$ for each point using SSED, where we always assume r_1 is the smallest distance, then we assign this point to proper cluster through running PMIN_k protocol on the k distances. After clustering, we update the bitmap vector.

(3) For any point p_i in L_p in each bucket, whose distance to representative point p_r in this bucket is smaller than ε, we add cluster j into a set closeSet if $r\_j - r\_1 \leq 2d, j \in [2,k]$. The next step we should do is to compute distances between p_i and centroids in closeSet, cluster p_i accurately, and update according bitmap vector.

(4) Update the new cluster centroids.

Termination.The Step (2)-(4) in LSH-based pruning clustering process will be executed repeatedly until the clustering results do not change any more or the given maximum iteration number is reached.

5. Security Analysis

In this section, we will give a security proof of our LSH-PPMOC protocol under the semi-honest model. Since the proof of preliminaries we utilize in this paper are similar, we only take the SM protocol as an example and give a formal proof under ``Real-versus-Ideal" framework (Goldreich, 2009).

Theorem 2. The SM Protocol securely computes the multiplication on ciphertexts using the Paillier cryptosystem under two semi-honest but passive cloud servers.

Proof. Our SM protocol is performed by two semi-honest parties, Alice and Bob. We need to prove that SM is secure against both of semi-honest attacker Alice $\mathcal{A}\_A$ and Bob $\mathcal{A}\_B$.

(1) Security against $\mathcal{A}\_A$: In Step (1), the real world view of $\mathcal{A}\_A$ includes inputs $\{E\_(\ 〚pk〛\_u\ )\ (x), E\_(\ 〚pk〛\_u\ )\ (y)\}$, random values $\{r\_x, r\_y\}$ and outputs $E\_(\ 〚pk〛\_u\ )\ (x + r\_x), E\_(\ 〚pk〛\_u\ )\ (y + r\_y)$. In Step (2), the real world view includes $E\_(\ 〚pk〛\_u\ )\ ((x + r\_x)(y + r\_y))$ and $E\_(\ 〚pk〛\_u\ )\ (x \cdot y)$. However, $\mathcal{A}\_A$ can't decrypt these ciphertexts without $〚sk〛\_u$. Then we can construct a simulator $\mathcal{S}\_A$ in the ideal world. $\mathcal{S}\_A$ generates ciphertexts $\{E\_(\ 〚pk〛\_u\ )\ (m\_1), E\_(\ 〚pk〛\_u\ )\ (m\_2), E\_(\ 〚pk〛\_u\ )\ ((m\_1)\hat{}),$

$E\_(\ 〚pk〛\_u\ )\ ((m\_2)\hat{})\}$ and random values r_1, r_2 by randomly selecting from Z_N. Then $\mathcal{S}\_A$ executes SM protocol. Considering the Paillier cryptosystem is semantic secure, it is computationally difficult for $\mathcal{S}\_A$ to distinguish ideal world and the real world, which means that

$$〚〚Ideal〛\_(\ 〚f, \mathcal{S}〛\_A\ )\ 〛\_ \approx \hat{}c\ \ 〚\ Real\ 〛\_(SM\ 〚, \mathcal{S}〛\_A\ ) \tag{4}$$

where the symbol f means multiplication function on $E\_(\ 〚pk〛\_u\ )\ (m\_1), E\_(\ 〚pk〛\_u\ )\ (m\_2), E\_(\ 〚pk〛\_u\ )\ ((m\_1)\hat{}), E\_(\ 〚pk〛\_u\ )\ ((m\_2)\hat{}), r\_1, r\_2$ and

$\_ \approx \hat{}c$ means computationally indistinguishable.

(2) Security against $\mathcal{A}\_B$: In Step (2), the real world view of $\mathcal{A}\_B$ comprises of input $E\_(\ 〚pk〛\_u\ )\ (x + r\_x)$, $E\_(\ 〚pk〛\_u\ )\ (y + r\_y)$, and output $E\_(\ 〚pk〛\_u\ )\ ((x + r\_x)(y + r\_y))$. Though $\mathcal{A}\_B$ can decrypt the ciphertexts and get the messages $x + r\_x, y + r\_y$ with $〚sk〛\_u$, it is still hard for $\mathcal{A}\_B$ to get any useful information about the original data records x and y. Since r_x, r_y are random values which are chosen by Alice, they are random values in the point of view of $\mathcal{A}\_B$. Then we can build a simulator $\mathcal{S}\_B$ in ideal world, where we just choose random messages as ciphertexts. There is no doubt that it is computationally hard for $\mathcal{A}\_B$ to distinguish the ideal world and the real world. That is to say

$$〚〚Ideal〛\_(\ 〚f, \mathcal{S}〛\_B\ )\ 〛\_ \approx \hat{}c\ \ 〚\ Real\ 〛\_(SM\ 〚, \mathcal{S}〛\_B\ ) \tag{5}$$

Combine the above two analyses, we prove the correctness of Theorem 2.

In the data uploading process, even though CS holds the data records in ciphertexts form, it can't get any information about original datasets due to the semantic security of Paillier cryptosystem. As for blinded hash tables, though it may disclose the relationship among user's data records, it is a compromise for efficiency. During the clustering process, AS with decryption key will assist CS to perform various computation operations. It is worth noting that the plaintexts obtained by AS are randomized. Considering Paillier cryptosystem is semantic secure, and blinding factors are all randomly chosen, no additional information regarding users' data or clusters is revealed to the cloud servers. Besides, (Liu, Lu, Ma, Chen & Qin, 2015) has given security proof of PMIN and PMIN_k. Then preliminaries utilized in our paper are proven to be privacy-protected, according to Composition Theorem (Goldreich, 2009), we conclude the sequential composition in proposed LSH-PPMOC protocol is secure against the semi-honest cloud servers.

6. Performance Analysis

In this section, we give the performance analysis of LSH-PPMOC protocol from the view of theory and experiment.

6.1 Theoretical Analysis

To better illustrate the computational cost and communicational cost of our LSH-PPMOC protocol, we use the

symbol exp, mul being the modular exponentiation and multiplication operations. We assume $g = N + 1$ in the Paillier cryptosystem which is a common setting like (Samanthula, Rao, Bertino, Yi & liu, 2014), then

$$E\_pk\ (a) = (N + 1)\hat{}a \cdot r\hat{}N\ \ mod\ N\hat{}2 = (a \cdot N + 1) \cdot r^N\ mod\ N^2 \qquad (6)$$

From Equation (6), we can conclude that the computational cost for one encryption is $1exp + 2mul$,, also the same cost for decryption. The computational overheads and communicational cost in one iteration for our main preliminaries are given in Table 1. Here d is the dimension of a data record, k is the number of predefined clusters and m is number of all users' records. Besides, we use N to denote the size (in bits) of the Paillier encryption key. It is worth noting that the Stage 1 of our proposed algorithm is run only once, while the Stage 2 and Stage 3 are run in an iterative fashion until the clustering is finished.

Table 1. Computational and communication costs of primary algorithms

Algorithm	Computational cost	Communicational cost (in bits)
SM	$7exp + 6mul$	$3\|N\|$
SSED	$15l\ exp + 14l\ mul$	$4\|N\|$
SC	$3exp + 6mul$	$6\|N\|$
ESDC	$57exp + 111mul$	$55\|N\|$
PMIN	$19exp + 38mul$	$14\|N\|$

6.2 Experimental Analysis

In this part, we will conduct our experiments on local terminals, the server running Windows10 has Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19GHz and 8.00 GB RAM. We compare our proposed LSH-PPOMC with PPOCM (Rong, Wang, Liu, Hao & Xian, 2017), because our models are similar and both under public key cryptosystem, what's more, we have the same security goal for outsourced k-means clustering. We choose the key size $|N|$ as 1024-bit.

For authenticity of our experiments, we choose a real dataset, KEGG Metabolic Reaction Network dataset, which consists of 65,554 data records and 28 attributes, to perform our algorithm. But we find some data records are missing attribute values or repeating, we delete these corresponding records. And because some of the attribute values are decimal, we normalize all records so that each attribute value is scaled into integer which maintains [0,1000].

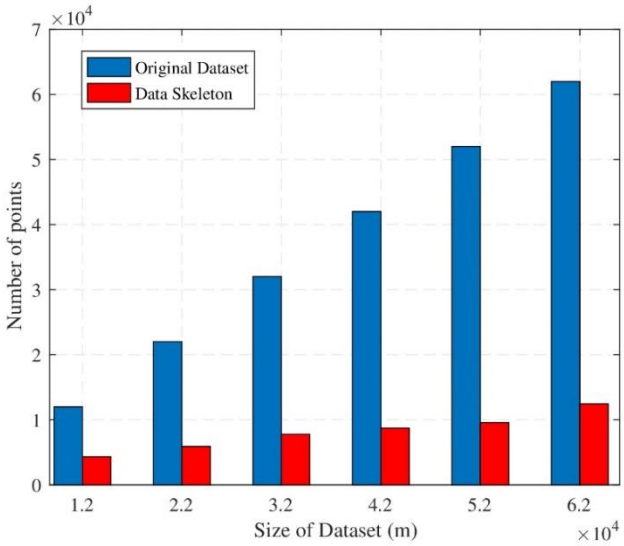


Figure 4. Original Dataset and Data Skeleton for KEGG

Before we perform k-means clustering work on the dataset, we want to illustrate the significant decrease between data points in original dataset and data skeleton in the form of a chart. The results are shown in Figure 4. From

this chart, we can see that, after users' preprocessing, the number of data points falls off drastically.

As for the secure k-means clustering on KEGG dataset, there are few factors affecting the final effect: the number of clusters (k), the size of aggregated dataset (m), and the number of attributes (d). We first asset the performance of PPOCM and LSH-PPMOC on the datasets of different sizes with dimension d=10, while k=10 and 20. The results was given in Figure 5. We can see that, the cloud running time grows almost linearly with k for both algorithms. Besides, we can also see that our LSH-PPOMC outperforms PPOCM with an increase in the number of clusters k owing to our pruning strategies.

Then we compare two k-means algorithms, with or without pruning strategies, while keeping everything else the same. Notably, the algorithm with pruning technique is exactly our LSH-PPMOC. More specifically, we change the size of aggregated dataset and the size of attribute values d respectively, and we assume k=10. The result is given in Figure 6. From the former diagram, we can learn that the pruning strategies can save about half the cloud running time compared with the algorithm without pruning strategies, while from the latter, we observe that our LSH-PPMOC can save nearly 70% time in terms of distance calculation. For example, when m=10000, d=10 and k=10, the time of distance calculation is 92min after pruning, while 275min without pruning operation. The difference can be attributed to the pruning strategies, whose goal is to reduce unnecessary distance calculation during the clustering. Besides, we can also learn that, as the amount of data increasing, the pruning is more effective.

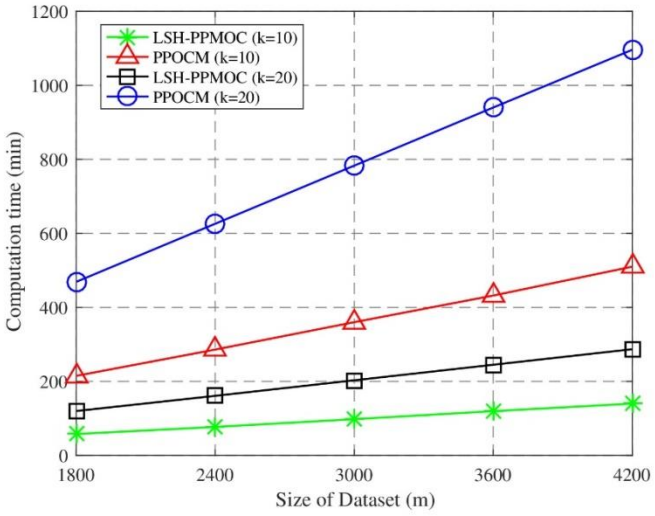
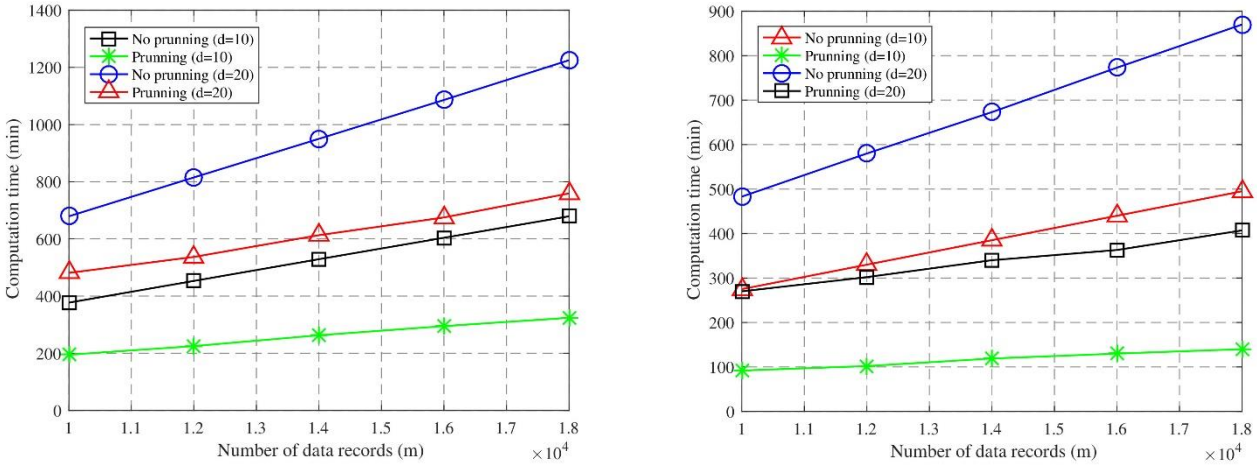


Figure 5. Cloud Running Time



(a) Time for Complete Algorithm (b) Time for Distance Computing

Figure 6. Cloud Running Time for Complete Algorithm and Distance Computing

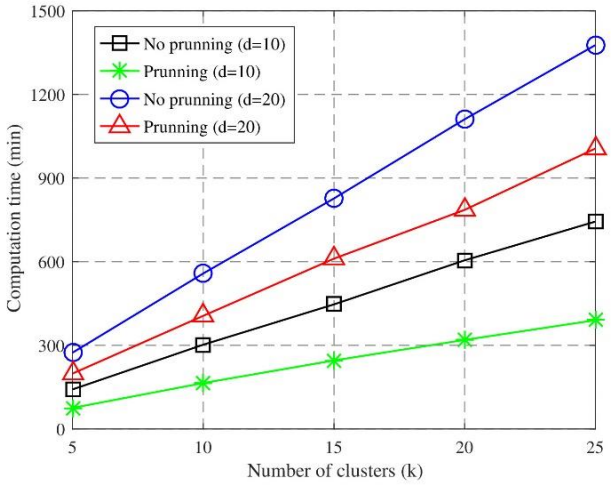


Figure 7. Cloud Running Time

Similarly, we give Figure 7 to show the effect of various k on performing clustering with two methods. It is obvious that the cloud running time saved is proportion to the number of clusters k, that is to say, when k is larger, the pruning effect is more obvious. So our LSH-PPMOC is suitable for k-means clustering on big dataset even though k is large.

7. Conclusion

In this paper, we proposed an efficient and privacy-preserving outsourced k-means clustering algorithm (LSH-PPMOC) for big data mining. For that purpose, we gave a series of building blocks to help achieve ciphertext multiplication, squared Euclidean distances computation, comparison and so on, which would never leak any useful information. The main idea in our algorithm is utilizing the local sensitivity of locally sensitive hash to prune some unnecessary computations when clustering. Besides, the experiments on KEGG dataset show that our algorithm is more efficient than some existing job. In the future, we are more willing to focus on the data integrity verification during k-means clustering and achieve strong secure under malicious model.

References

Jain, A. K., Murty, M. N., &Flynn, P. J. (1999). *Data clustering: a review.* ACM Comput. Surv. 31, 3 (Sept. 1999), 264-323. https://doi.org/10.1145/331499.331504

Almutairi, N., Coenen, F., & Dures, K. (2017, August). *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 (pp. 274-285). Springer, Cham. https://doi.org/10.1007/978-3-319-64283-3_20

Bhaskara, A., & Wijewardena, M. (2018, July). *Distributed clustering via lsh based data partitioning*. In International Conference on Machine Learning (pp. 570-579).

Brakerski, Z., Gentry, C., & Halevi, S. (2013, February). *Packed ciphertexts in LWE-based homomorphic encryption*. In International Workshop on Public Key Cryptography (pp. 1-13). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-36362-7_1

Chen, Z., Wang, Y., Zhang, S., Zhong, H., & Chen, L. (2018). *Differentially private user-based collaborative filtering recommendation based on K-means clustering*. arXiv preprint arXiv:1812.01782.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., & Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining. *ACM Sigkdd Explorations Newsletter, 4*(2), 28-34. https://doi.org/10.1145/772862.772867

Cui, X., Zhu, P., Yang, X., Li, K., & Ji, C. (2014). Optimized big data K-means clustering using MapReduce. *The Journal of Supercomputing, 70*(3), 1249-1259. https://doi.org/10.1007/s11227-014-1225-7

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). *Locality-sensitive hashing scheme based on p-stable distributions*. In Proceedings of the twentieth annual symposium on Computational geometry (pp. 253-262). https://doi.org/10.1145/997817.997857

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113. https://doi.org/10.1145/1327452.1327492

Gentry, C. (2009, May). *Fully homomorphic encryption using ideal lattices*. In Proceedings of the forty-first annual ACM symposium on Theory of computing (pp. 169-178). https://doi.org/10.1145/1536414.1536440

Goldreich, O. (2009). *Foundations of cryptography: volume 2, basic applications*. Cambridge university press.

Indyk, P., & Motwani, R. (1998, May). *Approximate nearest neighbors: towards removing the curse of dimensionality* . In Proceedings of the thirtieth annual ACM symposium on Theory of computing. https://doi.org/10.1145/276698.276876

Jagannathan, G., & Wright, R. N. (2005, August). *Privacy-preserving distributed k-means clustering over arbitrarily partitioned data*. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 593-599). https://doi.org/10.1145/1081870.1081942

Jiang, Z. L., Guo, N., Jin, Y., Lv, J., Wu, Y., Liu, Z., ... Wang, X. (2020). Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. *Information Sciences, 518,* 168-180. https://doi.org/10.1016/j.ins.2019.12.051

Kriegel, H. P., Kröger, P., & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD), 3*(1), 1-58. https://doi.org/10.1145/1497577.1497578

Li, Q., Wang, P., Wang, W., Hu, H., Li, Z., & Li, J. (2014, April). *An efficient K-means clustering algorithm on MapReduce*. In International Conference on Database Systems for Advanced Applications (pp. 357-371). Springer, Cham. https://doi.org/10.1007/978-3-319-05810-8_24

Li, Y., Shang, Y., & Yang, Y. (2017). Clustering coefficients of large networks. *Information Sciences, s 382-383,* 350-358. https://doi.org/10.1016/j.ins.2016.12.027

Lindell, Y., & Pinkas, B. (2008). *Secure Multiparty Computation for Privacy-Preserving Data Mining*. *IACR* Cryptol. ePrint Arch., 2008, 197. https://doi.org/10.29012/jpc.v1i1.566

Liu, D., Bertino, E., & Yi, X. (2014, June). *Privacy of outsourced k-means clustering*. In Proceedings of the 9th ACM symposium on Information, computer and communications security (pp. 123-134). https://doi.org/10.1145/2590296.2590332

Liu, K., Giannella, C., & Kargupta, H. (2006). *An attacker's view of distance preserving maps for privacy preserving data mining*. In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 297-308). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11871637_30

Liu, K., Kargupta, H., & Ryan, J. (2005). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering, 18*(1), 92-106. https://doi.org/10.1109/TKDE.2006.14

Liu, X., Lu, R., Ma, J., Chen, L., & Qin, B. (2015). Privacy-preserving patient-centric clinical decision support system on naive Bayesian classification. *IEEE journal of biomedical and health informatics, 20*(2), 655-668. https://doi.org/10.1109/JBHI.2015.2407157

Mohassel, P., Rosulek, M., & Trieu, N. (2020). Practical privacy-preserving k-means clustering. *Proceedings on Privacy Enhancing Technologies, 2020*(4), 414-433. https://doi.org/10.2478/popets-2020-0080

Mustafi, D., & Sahoo, G. (2019). A hybrid approach using genetic algorithm and the differential evolution heuristic for enhanced initialization of the k-means algorithm with applications in text clustering. *Soft Computing, 23*(15), 6361-6378. https://doi.org/10.1007/s00500-018-3289-4

Paillier, P. (1999, May). *Public-key cryptosystems based on composite degree residuosity classes*. In International conference on the theory and applications of cryptographic techniques (pp. 223-238). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48910-X_16

Rong, H., Wang, H., Liu, J., Hao, J., & Xian, M. (2017). *Outsourced k-Means clustering over encrypted data under multiple keys in spark framework*. In International Conference on Security and Privacy in Communication Systems (pp. 67-87). Springer, Cham. https://doi.org/10.1007/978-3-319-78813-5_4

Samanthula, B. K., Rao, F. Y., Bertino, E., Yi, X., & Liu, D. (2014). *Privacy-preserving and outsourced multi-user k-means clustering*. arXiv preprint arXiv:1412.4378.

Sardar, T. H., & Ansari, Z. (2020). An Analysis of Distributed Document Clustering Using MapReduce Based

K-Means Algorithm. *Journal of The Institution of Engineers (India): Series B, 101*(6), 641-650. https://doi.org/10.1007/s40031-020-00485-2

Su, D., Cao, J., Li, N., Bertino, E., Lyu, M., & Jin, H. (2017). Differentially private k-means clustering and a hybrid approach to private optimization. *ACM Transactions on Privacy and Security (TOPS), 20*(4), 1-33. https://doi.org/10.1145/3133201

Upmanyu, M., Namboodiri, A. M., Srinathan, K., & Jawahar, C. V. (2010, June). Efficient privacy preserving k-means clustering. *In Pacific-Asia Workshop on Intelligence and Security Informatics* (pp. 154-166). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13601-6_17

Wang, Y. (2015). *Notes on Two Fully Homomorphic Encryption Schemes Without Bootstrapping*. IACR Cryptol. ePrint Arch., 2015, 519.

Wong, W. K., Cheung, D. W. L., Kao, B., & Mamoulis, N. (2009). *Secure kNN computation on encrypted databases*. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (pp. 139-152). https://doi.org/10.1145/1559845.1559862

Xia, C., Hua, J., Tong, W., & Zhong, S. (2020). Distributed K-Means clustering guaranteeing local differential privacy. *Computers & Security, 90*, 101699. https://doi.org/10.1016/j.cose.2019.101699

Xing, K., Hu, C., Yu, J., Cheng, X., & Zhang, F. (2017). Mutual privacy preserving $ k $-means clustering in social participatory sensing. *IEEE Transactions on Industrial Informatics, 13*(4), 2066-2076. https://doi.org/10.1109/TII.2017.2695487

Yin, C., & Zhang, S. (2017). *P*arallel implementing improved k-means applied for image retrieval and anomaly detection. *Multimedia Tools and Applications, 76*(16), 16911-16927. https://doi.org/10.1007/s11042-016-3638-1

Youn, T. Y., Park, Y. H., Kim, C. H., & Lim, J. (2005, August). An efficient public key cryptosystem with a privacy enhanced double decryption mechanism. In *International Workshop on Selected Areas in Cryptography* (pp. 144-158). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11693383_10

Yu, Q., Luo, Y., Chen, C., & Ding, X. (2016). Outlier-eliminated k-means clustering algorithm based on differential privacy preservation. *Applied Intelligence, 45*(4), 1179-1191. https://doi.org/10.1007/s10489-016-0813-z

Yuan, J., & Tian, Y. (2017). *Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset.* IEEE Transactions on Cloud Computing.

Zou, Y., Zhao, Z., Shi, S., Wang, L., Peng, Y., Ping, Y., & Wang, B. (2020). *Highly Secure Privacy-Preserving Outsourced k-Means Clustering under Multiple Keys in Cloud Computing*. Security and Communication Networks, 2020. https://doi.org/10.1155/2020/1238505