Hindawi
Security and Communication Networks
Volume 2021, Article ID 3630312, 11 pages
https://doi.org/10.1155/2021/3630312

*Research Article*

# Multi-Party Verifiable Privacy-Preserving Federated *k*-Means Clustering in Outsourced Environment

**Ruiqi Hou,[1] Fei Tang,[1] Shikai Liang,[2] and Guowei Ling[2]**

[1]*School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing 400 065, China*
[2]*College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400 065, China*

Correspondence should be addressed to Fei Tang; tangfei@cqupt.edu.cn

Received 5 November 2021; Accepted 9 December 2021; Published 28 December 2021

Academic Editor: Youwen Zhu



As a commonly used algorithm in data mining, clustering has been widely applied in many fields, such as machine learning, information retrieval, and pattern recognition. In reality, data to be analyzed are often distributed to multiple parties. Moreover, the rapidly increasing data volume puts heavy computing pressure on data owners. Thus, data owners tend to outsource their own data to cloud servers and obtain data analysis results for the federated data. However, the existing privacy-preserving outsourced *k*-means schemes cannot verify whether participants share consistent data. Considering the scenarios with multiple data owners and sensitive information security in an outsourced environment, we propose a verifiable privacy-preserving federated *k*-means clustering scheme. In this article, cloud servers and participants perform *k*-means clustering algorithm over encrypted data without exposing private data and intermediate results in each iteration. In particular, our scheme can verify the shares from participants when updating the cluster centers based on secret sharing, hash function and blockchain, so that our scheme can resist inconsistent share attacks by malicious participants. Finally, the security and experimental analysis are carried out to show that our scheme can protect private data and get high-accuracy clustering results.

## 1. Introduction

Data mining technology can be used to analyze and extract potentially valuable information from large collections of data. Clustering algorithms are widely used in data mining and have an important role in research for medical, scientific, and commercial applications in practical life. In brief, clustering [1, 2] algorithms can divide data items into groups according to their features and attributes, such that the data items are sufficiently similar in the same group. As a well-known clustering algorithm, *k*-means clustering [3] algorithm has the advantages of simple process and good clustering results and it can assign data into *k* clusters based on the distances from cluster centers.

Data analysis can extract a lot of useful information, but in the process of data analysis, a large amount of personal privacy data will be collected and analyzed, such as living habits, criminal records, and medical records. Furthermore, privacy data breaches often result in financial losses and panic for society and companies. Data privacy has gained more attention than before. There are some researches about privacy-preserving in [4–6]. Moreover, a lot of privacy-preserving data mining schemes can be found in [7–9] in recent years.

The traditional privacy-preserving *k*-means schemes are primarily achieved through the interaction of participants. Vaidya and Clifton [10] firstly proposed the multi-party privacy-preserving *k*-means clustering protocol on vertically partitioned data, where the secure distance computation and comparison are supported by the secure permutation scheme and homomorphic encryption. Jha et al. [11] introduced a two-party privacy-preserving *k*-means clustering protocol based on oblivious polynomial evaluation and homomorphic encryption. Bunn and Ostrovsky [12]

presented an efficient two-party clustering protocol on arbitrarily partitioned data, where the intermediate values are not disclosed by using division protocol and random values protocol. Yi and Zhang [13] introduced an equally contributory privacy-preserving $k$-means clustering protocol based on ElGamal, plaintext equivalence test protocol, and mix networks. Xing et al. [14] proposed a mutual privacy-preserving $k$-means scheme in the social scene where the parties are grouped with the help of a data analyst, and the scheme can resist collusion attacks. Zhang et al. [15] combined secure multi-party computing and differential privacy technology to train a privacy-preserving $k$-means clustering model. After that, privacy-preserving $k$-means schemes in the malicious model are proposed in [16, 17].

Recently, explosive growth data poses a challenge for data owners in storing and computing, and a cloud server with high storage capacity and strong computing power is a good solution to the problem. Privacy protection and audit research in the cloud environment have also been studied in [18–22]. So there are some privacy-preserving $k$-means schemes in an outsourced environment. Liu et al. [23], following the framework in [24], presented a privacy-preserving outsourced $k$-means clustering protocol that one party outsourced the distance computation to a cloud server without revealing both the data and clustering results to any party and cloud server. Jiang et al. [25] introduced an efficient two-party privacy-preserving $k$-means clustering protocol, and this scheme can compute distance safely using subprotocols in [26] and update cluster centers using garbled circuit proposed in [27]. Zou et al. [28] proposed a highly secure privacy-preserving outsourced $k$–means clustering scheme using BCP homomorphic encryption and AES encryption under multiple keys. Sakellariou and Gounaris [29] introduced a privacy-preserving outsourced $k$-means scheme with low client-side load based on switch key and Paillier encryption. However, the existing privacy-preserving outsourced $k$-means schemes cannot verify whether participants share consistent data. In this article, we propose a multi-party verifiable privacy-preserving federated $k$-means scheme for horizontally partitioned data.

Our main contributions can be summarized as follows:

(1) We propose a privacy-preserving $k$-means scheme based on Paillier cryptosystem, secret sharing, hash function, and blockchain. In the multi-party scenario, we outsource the main computing task to the cloud server and reduce the computing overhead of participants.

(2) Our scheme can protect the participants' information and avoid leaking the clustering centers to participants in each iteration. Furthermore, the malicious participant can be detected using hash function and blockchain in the process of updating cluster centers.

(3) We carry out an experimental study to validate our scheme, and the experimental results show that the cloud server's running time can reach 77 % in our scheme. Moreover, our scheme can obtain high-accuracy clustering results.

The rest of the article is organized as follows. Section 2 introduces the preliminaries about $k$-means clustering and cryptography knowledge. Section 3 presents the framework and specific entities in our scheme. Basic secure protocol and our scheme are detailed in Section 4. Security analysis is carried out in Section 5. Performance analysis is presented in Section 6. Moreover, we conclude this article in Section 7.

## 2. Preliminaries

For better elaboration, the notations used in this article and their semantic meanings are presented in Table 1.

*2.1. k-Means Clustering Algorithm.* The $k$-means clustering algorithm is one of the most well-performed unsupervised clustering algorithms. The process of $k$-means clustering algorithm is described as follows. Assume that there is a set of samples and each sample $a_i$ is an $\ell$-dimensional data. Suppose that the samples need to be grouped into $k$ clusters $\{C_1, \ldots, C_k\}$, where the cluster center of $j$th cluster is denoted by $\mu_j$. Initially, randomly select $k$ samples $\{\mu_1, \ldots, \mu_k\}$ as the initial cluster centers. There are many iterations to measure the distances between each sample and $k$ cluster centers. In this article, we adopt Euclidean distance as the criterion. Sample $a_i$ belongs to cluster $C_j$ if the cluster center $\mu_j$ is the closest to $a_i$. In each iteration, each sample is reassigned to the nearest cluster and recompute the cluster centers as in the following equation (1). The iteration terminates when there is no or little change in the cluster centers. The specific description of the $k$-means clustering algorithm is shown in Algorithm 1.

$$\mu_j = \frac{\sum_{a_i \in C_j} a_i}{\left|C_j\right|}. \tag{1}$$

*2.2. Homomorphic Encryption.* Homomorphic encryption allows certain computation over encrypted data. Paillier [30] cryptosystem is a popular homomorphic encryption scheme based on the decisional composite residuosity class problem. Furthermore, the Paillier cryptosystem can provide fast encryption and decryption, and it is widely used in privacy-preserving data mining. We adopt Paillier cryptosystem in our scheme. The Paillier cryptosystem is briefly introduced as follows:

(i) *Key generation:* an entity selects two large primes $p$ and $q$ and compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. Then randomly choose an integer $g \in \mathbb{Z}_N^*$ and check whether $\gcd(L(g^\lambda \bmod n^2), n) = 1$, where $L(x) = (x-1)/n$. The public key is $pk = (n, g)$ and the secret key $sk = (\lambda)$.

(ii) *Encryption:* let $m \in \mathbb{Z}_N^*$ be a message and $r \in \mathbb{Z}_N^*$ be a random number. The ciphertext of $m$ is computed by

Table 1: Notations and their semantic meanings.

| Notations | Meanings |
|---|---|
| $C_i$ | The $i$th cluster, $1 \le i \le k$ |
| $\mu_i$ | The $i$th cluster center |
| $E(\cdot)$ | Encryption operation |
| $D(\cdot)$ | Decryption operation |
| $C$ | Cloud server |
| $P_i$ | The $i$th participant |
| $B$ | Blockchain |
| $n$ | Number of participants |
| $d_{p_i}$ | Sample size of the $i$th participant |
| $S_i$ | Sample of the $i$th participant |
| $hv$ | Hash value |
| $S$.sum | The sum of samples |
| $S$.no | The number of samples |

Randomly select $k$ cluster centers $\{\mu_1, \ldots, \mu_k\}$ from the dataset
Repeat
(1) Calculate the distances between each sample and $k$ cluster centers $\{\mu_1, \ldots, \mu_k\}$
(2) Assign each sample to the closest cluster
(3) Replace each cluster centers $\mu_i$ with the mean of the $i$ th cluster
**until** Cluster centers do not change

Algorithm 1: $k$-means clustering.

$$E(m) = g^m \cdot r^n \bmod n^2, \tag{2}$$

where $E(\cdot)$ denotes the encryption with the $pk = (n, g)$.

(iii) *Decryption:* decrypt the ciphertext of $m$ by

$$D(E(m)) = \frac{L\left(E(m)^\lambda \bmod n^2\right)}{L\left(g^\lambda \bmod n^2\right)} \bmod n, \tag{3}$$

where $D(\cdot)$ denotes the decryption with the $sk = (\lambda)$.

(iv) *Homomorphic:* the Paillier cryptosystem is additive homomorphic, which satisfies the following equation:

$$\begin{aligned} E(m_1) \cdot E(m_2) &= E(m_1 + m_2), \\ E(m)^c &= E(c \cdot m). \end{aligned} \tag{4}$$

*2.3. Blockchain.* Blockchain is the underlying technology of Bitcoin [31], which is essentially a distributed database. Blockchain is a very new network form, which uses cryptography, hash function, and proof of work (Pow). There are a lot of nice features of blockchain, such as decentralization, tamper resistance, and transparency.

(i) Decentralization: the data on the blockchain are maintained by all nodes in the peer-to-peer networks. Moreover, all nodes compete to generate a block of block without relying on a centralized third party to record transactions.

(ii) Tamper resistance: each node in the peer-to-peer networks saves a copy of data on the blockchain, so it is impossible to tamper with the data once the data has been recorded on the blockchain.

(iii) Transparency: the records on the blockchain are transparent to all nodes, and anyone can access the data on the blockchain.

## 3. Scheme Model

The scheme model is illustrated in Figure 1, where the multi-party verifiable privacy-preserving federated $k$-means model includes three entities. The first entity is participants; the data owners provide the original data for the $k$-means algorithm, such as hospitals, scientific research companies, and government agencies. The second entity is a cloud server with adequate storage and computing resources, where the cloud server is responsible for storing the encrypted samples of the participants and undertakes the main computational task in privacy-preserving $k$-means clustering. The last entity is blockchain, which is used to store hash values of secret shares. Because of the tamper-resistant nature of blockchain, once the hash values have been uploaded to the blockchain, it can be guaranteed that they will not change.

In the multi-party verifiable privacy-preserving federated $k$-means scheme, the specific descriptions of entities are shown below.

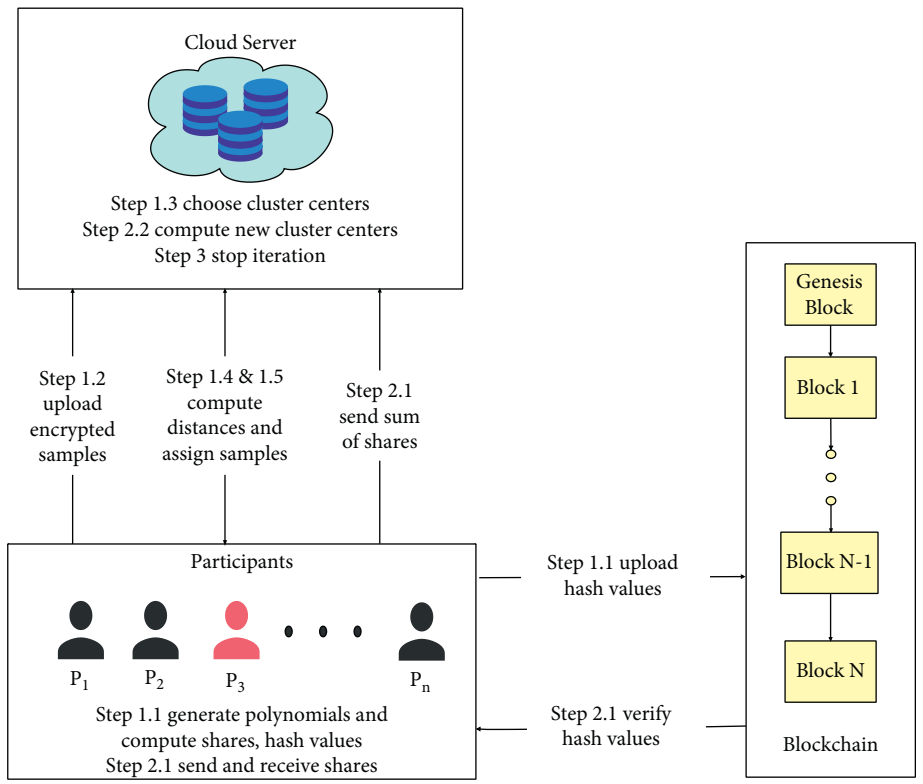


Figure 1: Model of multi-party verifiable privacy-preserving federated $k$-means clustering.

(i) *Participants:* the samples data to be analyzed are horizontally distributed on the participants, where the $n$ participants are denoted by $P_1, \ldots, P_n$. Moreover, each participant $P_i$ holds $d_{P_i}$ samples, where the samples are $\ell$-dimensional. Participants in red represent malicious participants. Malicious participants may not follow the protocol and share inconsistent data information. In our scheme, participants generate their public key and secret key and upload the encrypted samples to the cloud server. Besides, participants need to share data with a secret sharing scheme when updating the cluster centers. In order to verify the secret shares, participants need to compute the hash values of secret shares and upload them to the blockchain in advance.

(ii) *Cloud Server:* all encrypted samples are stored on the cloud server. The cloud server is responsible for interacting with participants and calculating and comparing the distances between encrypted samples and cluster centers. Then, the cloud server assigns samples to different clusters.

(iii) *Blockchain:* blockchain is responsible for storing hash values generated by the participants.

The aim of our scheme is to build a multi-party verifiable federated $k$-means scheme. In our scheme, participants only get the final $k$-means clustering results, and the cluster centers in each iteration are only known to the cloud server. Furthermore, participants and cloud servers cannot know or infer private information about the other side. In the process of updating the cluster centers, participants send secret shares to other participants and verify the secret shares received from other participants with hash values. Once there exists a secret share that fails to verify, it means that malicious participants have sent inconsistent secret shares so that malicious participants can be detected in time and avoid losses due to incorrect $k$-means results.

## 4. Our Construction

We construct a multi-party verifiable privacy-preserving federate $k$-means scheme. In our scheme, we use Paillier encryption to protect the information of participants. Furthermore, secret sharing, hash function, and blockchain technology are used to verify and update new cluster centers.

*4.1. Basic Secure Protocol.* We present a set of subprotocols that will be used in constructing the multi-party verifiable privacy-preserving federate $k$-means scheme.

(i) *Secure Multiplication (SM) Protocol.* In this protocol, participants $P$ have input $(E_{pk}(x), E_{pk}(y))$ and output $(E_{pk}(x * y))$ to cloud server $C$, where neither $P$ nor $C$ knows $x$ and $y$. Furthermore, information concerning $x$ and $y$ is not leaked to $P$ or $C$.

(ii) *Secure Squared Euclidean Distance (SSED) Protocol.* There exist two $\ell$-dimensional vectors denoted by $X = (x_1, \ldots, x_\ell)$ and $Y = (y_1, \ldots, y_\ell)$, where

$E(X) = <E(x_1), \ldots, E(x_\ell)>$ and $E(Y) = <E(y_1), \ldots, E(y_\ell)>$ denote the encrypted components sets of $X$ and $Y$. The cloud server $C$ with the input $(E(X), E(Y))$ and participants $P$ with $sk$ securely compute the encrypted value of the squared Euclidean distance between vectors $X$ and $Y$.

(iii) *Secure Bit-Decomposition (SBD) Protocol.* SBD [32] protocol considers cloud server $C$ with input $E_{\text{pk}}(z)$ and participants $P$ securely compute encrypted values of the individual bits of $z$, where $0 \le z \le 2^\alpha - 1$ and $z$ is not known to $C$ and $P$. No information regarding output $[z] = \langle E_{\text{pk}}(z_1), \ldots, E_{\text{pk}}(z_\alpha)\rangle$ is revealed to $P$. Here, $(z_1, z_\alpha)$ are the most and least significant bits of $z$.

(iv) *Secure Minimum out of 2 Numbers (SMIN2) Protocol.* We assume that $u \in \{0,1\}^\alpha$ and $v \in \{0,1\}^\alpha$ are two $\alpha$-bit strings and $0 \le u, v \le 2^\alpha - 1$. Let $[u] = \{E(u_1), \ldots, E(u_\alpha)\}$ and $[v] = \{E(v_1), \ldots, E(v_\alpha)\}$ represent the encrypted bits of $u$ and $v$, respectively, where $u_i$ and $v_i$ denote each bit of $u$ and $v$. This protocol considers cloud server $C$ with input $([u], [v])$ and participants $P$ with $sk$ securely output the encrypted values of the individual bits of $\min(u, v)$, where $(u_1, u_\alpha)$ and $(v_1, v_\alpha)$ are the most and least significant bits of $u$ and $v$, respectively.

(v) *Secure Minimum out of k Numbers (SMINk) Protocol.* In this protocol, $d_i \in \{0,1\}^\alpha (1 \le i \le k)$ is a distance and $d_{i,j} (1 \le j \le \alpha)$ denotes a bit of $d_i$. Let $[d_i] = \{E(d_{i,1}), \ldots, E(d_{i,\alpha})\}$, where $(d_{i,1}, d_{i,\alpha})$ is the most and least significant bits of $d_i$. Consider cloud server $C$ with $k$ encrypted vectors $\{[d_1], \ldots, [d_k]\}$ and participants $P$ with sk. The goal of this protocol is to compute the $\min([d_1], \ldots, [d_k])$ without revealing any information regarding $d_i (1 \le i \le k)$ to $C$ and $P$.

(vi) *Secure and Verifiable Vector Share (SVVS) Protocol.* Participant $P_i$ has a dataset, $S_i = \{S_{i,j} | 1 \le j \le d_{p_i}\} (1 \le i \le n)$. Samples $S_{i,j} = \{s_{i,j,t} | 1 \le t \le \ell\}$ are $\ell$-dimensional vectors. In this protocol, $P_i$ securely share samples using a secret sharing scheme. Firstly, $P_i$ generate polynomial $f_{i,j}(x) = a_{i,j,} \ (n-1) * x^{n-1} + \cdots + a_{i,j,1} * x + a_{i,j,0}, 1 \le i \le n, 1 \le j \le d_{p_i}$ and compute shares $f_{i,j}(x_h) (1 \le h \le n)$. Then, $P_i$ computes the hash values $hv$ of $f_{i,j}(x_h)$ and upload them to the blockchain.

(vii) *Verify and Recovery Secret (VRS) Protocol.* In this protocol, participants $P_i$ with input clustering results $C^i = \{C^i_j | 1 \le j \le k\}$. $P_i$ sends shares $f(x_j)$ to $P_j$ and receives shares $f(x_j)$ from $P_j$. Then, $P_i$ computes hash values of $f(x_j)$ and verifies whether hash values are on the blockchain. If all hash values exist on the blockchain, $P_i$ computes the sum of shares $F(x_i) = \sum_{j=1}^{n} f(x_i) (1 \le i \le n)$ , where $F(x_i)$ is not known to other participants. $P_i$ sends $F(x_i)$ to cloud server $C$, and $C$ solves the set of $F(x)$ using Lagrange's interpolation to recovery the sum of the secret values.

*4.2. The Proposed Scheme.* This section presents the detailed steps of the multi-party verifiable privacy-preserving federated $k$-means scheme.

*4.2.1. Step 1: Assign Samples to Clusters.* This step aims to assign each sample to its nearest cluster. Where the cluster centers are initialized by cloud server $C$.

(i) *Step 1.1: participants generate polynomials and compute hash values.* For each sample $S_{i,j}$ of $P_i$, participant $P_i$ runs the SVVS protocol to generate polynomial values $f_{i,j}(x)$ and computes hash values $hv$ of $f_{i,j}(x)$. We use SHA256 in the SVVS protocol. Then, participants upload hash values to blockchain $B$.

(ii) *Step 1.2: participants encrypt samples and upload ciphertext to cloud server.* Each participant $P_i$ encrypts their samples, and $CT_i$ represents encrypted samples of $P_i$, where $\text{CT}_i = \{\text{CT}_{i,j} | 1 \le j \le d_{p_i}\}$, $\text{CT}_{i,j} = \{\text{CT}_{i,j,t} = E_{pk}(s_{i,j,t}) | 1 \le t \le \ell\}$ and uploads ciphertext to cloud server.

(iii) *Step 1.3: cloud server randomly chooses k cluster centers.* Cloud server $C$ randomly chooses $k$ cluster centers $\varphi = \{\mu_c | 1 \le c \le k\}$, where $\mu_c = \{\mu_{c,t} | 1 \le t \le \ell\}$. Cloud server encrypts $k$ cluster centers using the pk of each participant to get $C^i_\mu (1 \le i \le n)$, where $C^i_\mu = \{C^i_{\mu,c} | 1 \le c \le k\}$, $C^i_{\mu,c,t} = \{E_{pk}(\mu_{c,t}) | 1 \le t \le \ell\}$.

(iv) *Step 1.4: cloud server and participants compute distances.* Cloud server $C$ has $P_i$'s encrypted samples $\text{CT}_i$ and cluster centers $C^i_{\mu,c}$, and $P_i$ has the secret key $sk$. $C$ and $P_i$ run SSED protocol to compute encrypted distances between each sample and $k$ cluster centers. The distances are represented as $\text{dis}^i = \{\text{dis}^i_j = \{\text{dis}^i_{j,c} = \text{SSED}(\text{CT}_{i,j}, C^i_{\mu,c}) | 1 \le i \le n, 1 \le j \le d_{p_i}, 1 \le c \le k\}\}$. The $\text{dis}^i_j$ denotes the encrypted distances between $P_i$'s sample $S_{i,j}$ and $k$ cluster centers in encrypted form.

(v) *Step 1.5: cloud server assigns samples to k clusters.* To compare the encrypted distances, $C$ and $P_i$ run SMINk protocol to get the minimum distance $\min_i = \{\min_{i,j} = \text{SMINk}(\text{dis}^i_j) | 1 \le i \le n, 1 \le j \le d_{p_i}\}$ of sample $S_{i,j}$. The sample $S_{i,j}$ will be assigned to the $c$th cluster if the minimum distance $\min_{i,j}$ is equal to $\text{dis}^i_{j,c}$. Finally, the cloud server gets the clustering results of each participant's samples $C = \{C^i | 1 \le i \le n\}$, where $C^i = \{C^i_j | 1 \le j \le k\}$ and $C^i_j$ denotes the set of $P_i$'s samples in the $j$th cluster. Then, cloud server sends samples clustering results to the corresponding participants. In other words, participants only know the clustering results of their own samples, and they do not know the clustering results of other participants' samples.

**Require:** $C$ has $E_{pk}(x), E_{pk}(y)$; $P$ has $sk$

(1) $C$:

(a) Pick two random numbers $r_x, r_y \in Z_N$
(b) $x' \longleftarrow E(x) * E(r_x)$, $y' \longleftarrow E(y) * E(r_y)$
(c) Send $x', y'$ to $P$

(2) $P$:

(a) $h_x \longleftarrow D(x'), h_y \longleftarrow D(y')$, $h \longleftarrow h_x * h_y \mathrm{mod} N, h' \longleftarrow E(h)$
(b) Send $h'$ to $C$

(3) $C$:

(a) $s \longleftarrow h' * E(x)^{N-r_y}$, $s' \longleftarrow s * E(y)^{N-r_x}$
(b) Compute $E(x * y) \longleftarrow s' * E(r_x * r_y)^{N-1}$

Algorithm 2: SM $(E_{pk}(x), E_{pk}(y)) \longrightarrow E_{pk}(x * y)$.

**Require:** $C$ has $E_{\mathrm{pk}}(X), E_{\mathrm{pk}}(Y)$; $P$ has $sk$

(1) $C$:
**for** $1 \le i \le \ell$ **do:**
$E_{\mathrm{pk}}(x_i - y_i) \leftarrow E_{\mathrm{pk}}(x_i) * E_{\mathrm{pk}}(y_i)^{N-1}$

(2) $C$ and $P$:
**For** $1 \le i \le \ell$ **do:**
Use SM to compute $E_{\mathrm{pk}}((x_i - y_i)^2)$ with $E_{\mathrm{pk}}(x_i - y_i)$

(3) $C$:
Compute $E_{\mathrm{pk}}(|X - Y|^2) \leftarrow \prod_{i=1}^{\ell} E_{\mathrm{pk}}((x_i - y_i)^2)$

Algorithm 3: SSED $(E_{\mathrm{pk}}(X), E_{\mathrm{pk}}(Y)) \longrightarrow E_{\mathrm{pk}}(|X - Y|^2)$.

*4.2.2. Step 2: Update Cluster Centers.* This step aims to compute the new cluster centers.

(i) *Step 2.1: participants send and receive shares of samples.* After assigning all samples to $k$ clusters, participants run VRS protocol to update cluster centers. Because each sample's shares have been calculated by participants through the SVVS protocol in Step 1, it is very easy to get the shares of each sample. Suppose we want to update the $c$th cluster center, participants send shares of their samples in $c$th cluster to corresponding participants and receive shares of other participants' samples in $c$th cluster. After receiving shares from other participants, participants need to compute hash values of shares and verify whether these hash values exist on the blockchain. If the hash values exist on the blockchain, it indicates that other participants share the consistent values. Then, participants compute the sum of shares $F(x_i)$ and send $F(x_i)$ to the cloud server.

(ii) *Step 2.2: calculate the new centers.* The cloud server receives the sum of shares $F(x_i)(1 \le i \le n)$ from participants and uses Lagrange's interpolation to find the sum of samples $S.sum$. Because the cluster server has the clustering results of all samples, it can easily get the number of samples $S.no$ in each cluster. Then, the cluster server computes the new centers $\mu' = S.\mathrm{sum}_c / S.\mathrm{no}_c$. Moreover, the new cluster centers cannot be leaked to participants.

*4.2.3. Step 3: Stop Iteration.* This step aims to determine whether to terminate the privacy-preserving federated $k$-means algorithm. After Step 1 and Step 2, the cloud server should compare the new cluster centers $\mu'$ with previous cluster centers $\mu$. If they are close enough (e.g., the difference is no more than the threshold set earlier), the $k$-means ends and the clustering results have been sent to participants in Step 1.5. Otherwise, the cloud server encrypts the $k$ new cluster centers using each participants' public key. Then, go to Step 1.4 and iterate.

## 5. Security Analysis

In this section, we discuss the privacy protection capabilities offered by our scheme. Firstly, we define the goals in privacy protection for different entities.

For each participant in the federated $k$-means scheme, it should not access to the following information:

(1) Samples' information from other participants
(2) Clustering results for other participants' samples
(3) Clustering centers in each iteration

For the cloud server, it can only get the cluster centers in each iteration and cannot obtain the samples' information of participants.

The blockchain is only used to store hash values, which is publicly available and does not interact with the participants or the cloud server, so it does not access any private information about participants or the cloud server.

*5.1. Privacy-Preserving Analysis for Assigning Samples.* In the stage of assigning samples, each participant encrypts samples with the Paillier cryptosystem. Because Paillier's cryptosystem is semantically secure, other participants and cloud servers cannot decrypt and deduce additional information from the encrypted data.

According to our scheme, we should compute the distances between samples and cluster centers with the SM and SSED protocols. In SM protocol, the cloud server has the encrypted values $E(x)$ and $E(y)$, and the participants hold the secret key $sk$. The aim of the SM protocol is to return encrypted values of $x * y$ (e.g., $E(x * y)$) to cloud server with the interaction between cloud server and participants. The idea of SM protocol is based on the following property:

$$x * y = (x + r_x) * \left(y + r_y\right) - x * r_y - y * r_x - r_x * r_y. \quad (5)$$

The detailed SM protocol is shown in Algorithm 2. Firstly, the cloud server chooses two random numbers $r_x, r_y$, which are only known to the cloud server. Then, the cloud server computes $x' = E(x + r_x) = E(x) * E(r_x)$, $y' = E(y + r_y) = E(y) * E(r_y)$ and sends $x', y'$ to the participant. After receiving $x'$ and $y'$, participant decrypts $h_x = D(x'), h_y = D(y')$ and computes $h = h_x * h_y = (x + r_x) * (y + r_y)$. Then, the participant sends $h' = E(h)$ to the cloud server. Finally, the cloud server computes $E(x * y) = h' * E(x)^{N-r_y} * E(y)^{N-r_x} * E(r_x * r_y)^{N-1} = E((x + r_x) * (y + r_y)) * E(x)^{-r_y} * E(y)^{-r_x} * E(r_x * r_y)^{-1}$. Note that $N - x$ is equivalent to $-x$ for any $x \in Z_N$. During the SM protocol, no information about $x$ and $y$ except for $E(x * y)$ is revealed to the cloud server.

In the SSED protocol shown in Algorithm 3, the cloud server has the encrypted vectors $E(X) = <E(x_1), \ldots, E(x_\ell)>, E(Y) = <E(y_1), \ldots, E(y_\ell)>$ and the participants hold the secret key sk. The goal of the SSED protocol is to compute the distance between $X$ and $Y$. Cloud server computes $E((x_i - y_i)^2)$ using the SM protocol and then computes $E(|X - Y|^2) = \prod_{i=1}^{\ell} E((x_i - y_i)^2)$ because the Paillier cryptosystem has additive homomorphism. Since the SM protocol does not reveal private information to the cloud server, SSED does not leak information either.

After computing the distances between samples and each cluster center, each sample is assigned to the closest cluster with SBD, SMIN2, and SMINk protocols. In SBD protocol, for random $z (0 \le z \le 2^\ell)$, the cloud server only can get the individual bits of the binary representation of $z$. During the process, the cloud server has $E(z)$ and participants have $sk$, and $z$ is not known to both of them. The detailed SBD protocol is described in [32].

The SMIN2 protocol is described in Algorithm 4, and this protocol can help cloud server get the encrypted results of the individual bits of $\min(u, v)$. The participant can not only directly obtain $u$ or $v$ but also send an identifier $\alpha$ to the cloud server. Then, cloud server can access $\min(u, v)$ without decrypting or knowing $u, v$. Similarly, the SMINk protocol shown in Algorithm 5 is private and secure.

*5.2. Privacy-Preserving Analysis for Updating Cluster Centers.* In the step of updating the cluster centers, we mainly used Shamir's secret sharing. The SSVS protocol is described in Algorithm 6. In the SSVS protocol, each participant generates a polynomial for each sample $f(x) = a_{n-1} * x^{n-1} + \cdots + a_1 * x + a_0$. Participants compute shares and hash values of shares. Then, participants upload hash values to the blockchain. The VRS protocol is described in Algorithm 1. In the VRS protocol, participants send and receive shares with each other. The secret values of a participant cannot be revealed even if all remaining participants exchange their shares. Because each participant executes Shamir's secret sharing algorithm with a random polynomial of degree $n - 1$. In order to compute the coefficients of the corresponding polynomial, at least $n$ values of the polynomial are needed. In theVRS protocol, each participant actually computes $n$ values of polynomial but only sends $n - 1$ polynomial values to the other participants, keeping one polynomial value for itself. Thus, as long as the participants do not reveal the polynomial value they hold, the secret value cannot be deduced even if the remaining participants combine their shares. So the VRS protocol can protect the privacy of the participants.

*5.3. Resisting Attacks from Malicious Participants.* In our multi-party verifiable privacy-preserving federated $k$-means scheme, we make no assumption that all participants are semihonest. Thus, there will be malicious participants in the privacy-preserving $k$-means scheme. In the following, we will show that our scheme can resist inconsistent sharing attacks from malicious participants. We assume that the cloud server is semihonest and will follow the proposed protocol.

Because a malicious participant may send inconsistent shares (e.g., shares of samples that do not belong to him) to other participants in the process of updating cluster centers. When participants receive shares from others, they should compute the hash values of shares and verify whether the hash values are on the blockchain. Furthermore, once the information has been agreed and added to the blockchain, it is recorded by all nodes together and is cryptographically guaranteed to be interlinked backward and forward, making tampering very difficult and costly. Therefore, if participants receive a share that cannot be verified, it indicates that other participants have sent inconsistent data share. And we can conclude that the participant does not follow the proposed protocol, and it is considered to be a malicious participant. So incorrect $k$-Means results due to inconsistent shares by malicious participants can be avoided.

**Require:** $C$ has $[u]$ and $[v]$, where $0 \le u, v \le 2^{\alpha-1}$; $P$ has sk

(1) $C$:

(a) Randomly choose the functionality $F$

(b) **for** $1 \le i \le \alpha$ **do:** $E_{\text{pk}}(u_i * v_i) \longleftarrow SM(E_{\text{pk}}(u_i), E_{\text{pk}}(v_i))$

**if** $F$: $u > v$ **then:**

$W_i \longleftarrow E_{\text{pk}}(u_i) * E_{pk}(u_i * v_i)^{N-1}$, $\Gamma_i \longleftarrow E_{\text{pk}}(v_i - u_i) * E_{\text{pk}}(\widehat{r_i})$; $\widehat{r_i} \in Z_N$

else

$W_i \longleftarrow E_{\text{pk}}(v_i) * E_{\text{pk}}(u_i * v_i)^{N-1}$, $\Gamma_i \longleftarrow E_{\text{pk}}(u_i - v_i) * E_{\text{pk}}(\widehat{r_i})$; $\widehat{r_i} \in Z_N$

$G_i \longleftarrow E_{\text{pk}}(u_i \oplus v_i)$, $H_i \longleftarrow H_{i-1}^{r_i} * G_i$; $r_i \in Z_N$ and $H_0 = E_{\text{pk}}(0)$

$\Phi_i \longleftarrow E_{\text{pk}}(-1) * H_i$, $L_i \longleftarrow W_i * \Phi_i^{r_i}$; $r_i' \in Z_N$

(c) $\Gamma' \longleftarrow \pi_1(\Gamma)$, $L' \longleftarrow \pi_2(L)$. Send $\Gamma'$ and $L'$ to $P$

(2) $P$:

(a) $M_i \longleftarrow D_{\text{sk}}(L_i')$; for $1 \le i \le \alpha$

(b) **if** $\exists j$ such that $M_j = 1$ **then:** $\lambda \longleftarrow 1$

**else** $\lambda \longleftarrow 0$

(c) $M_i' \longleftarrow \Gamma_i'^{\lambda}$; for $1 \le i \le \alpha$, Send $M_i'$ and $E_{\text{pk}}(\lambda)$ to $C$

(3) $C$:

(a) $\widetilde{M} \longleftarrow \pi_1^{-1}(M')$

(b) **for** $1 \le i \le \alpha$ **do:** $\eta_i \longleftarrow \widetilde{M} * E_{\text{pk}}(\lambda)^{N-r_{i'}}$

**if** $F$: $u > v$ **then:** $E_{\text{pk}}(\min(u, v)_i) \longleftarrow E_{\text{pk}}(u_i) * \eta_i$

**else** $E_{\text{pk}}(\min(u, v)_i) \longleftarrow E_{\text{pk}}(v_i) * \eta_i$

(c) Get the $E_{pk}(\min(u, v))$

Algorithm 4: SMIN2 $([u], [v]) \longrightarrow [\min(u, v)]$.

**Require:** $C$ has $([d_1], \ldots, [d_k])$; $P$ has $sk$

(1) $C$:

$[d_i'] \leftarrow [d_i]$; for $1 \le i \le k$, num$\leftarrow k$

(2) $C$ and $P$:

(a) **for** $1 \le i \le \lceil \log_2 k \rceil$ **do**:

**for** $1 \le j \le \lfloor \text{num}/2 \rfloor$ **do**:

**if** $i = 1$ **then:**

$[d_{2j-1}'] \leftarrow$ SMIN2 $([d_{2j-1}'], [d_{2j}'])$, $[d_{2j}'] \leftarrow 0$

else

$[d_{2i(j-1)+1}'] \leftarrow$ SMIN2 $([d_{2i(j-1)+1}'], [d_{2ij-1}'])$, $[d_{2ij-1}'] \leftarrow 0$

(b) num$\leftarrow \lceil \text{num}/2 \rceil$

(3) $C$:

$[d_{\min}] \leftarrow [d_1']$, Get $[d_{\min}]$

Algorithm 5: SMINk $([d_1], \ldots, [d_k]) \longrightarrow [d_{\min}]$.

## 6. Performance Analysis

*6.1. Experimental Analysis.* We use two datasets of different sizes for the experiments. The first one is a 2-dimensional synthetic dataset S1 [33], which is a clustering benchmark. S1 contains 2000 samples and 7 cluster centers. The second dataset is the HCV [34], which contains laboratory values of blood donors and hepatitis C patients. HCV consists of 588 10-dimensional samples and 4 cluster centers.

To show that our scheme is suitable for multi-party scenarios, we assume that samples are randomly distributed among three participants, and we choose SHA256 as the required hash function. In the process of calculating metadata, the blockchain needs to run a consensus algorithm. The consensus time depends on the specific blockchain system selected. Therefore, this section only conducts experimental analysis on the proposed scheme and does not consider the execution cost of the blockchain itself.

Running time of the experiment on datasets S1 and HCV is shown in Tables 2 and 3, respectively, where $n$, $k$, $C$, and $P$ represent the data size, number of clusters, cloud server, and participants. The unit of time is seconds. We record the running time under different data size in one iteration, and it is obvious that the running time increases proportionally with the number of data size increasing.

**Require:** $C$ has a random number generate; $P$ has $sk$

(1) $C$:

(a) Choose $n$ publicly known random numbers $\{x_i, \ldots, x_n\}$

(b) Send $x_i$ to $P_i$

(2) $P$:

(a) **for** $1 \le i \le n$ **do:**

**For** $1 \le j \le d_{p_i}$ **do:**

$P_i$ chooses a random polynomial $f_{i,j}(x) = a_{i,j,(n-1)} * x^{n-1} + \cdots + a_{i,j,1} * x + a_{i,j,0}$, where $a_{i,j,0} \leftarrow S_{i,j}$, $S_{i,j} = \{s_{i,j,t} | 1 \le t \le \ell\}$

(b) **for** $1 \le h \le n$ **do:**

$P_i$ computes shares $f_{i,j}(x_h) = a_{i,j,(n-1)} * x_h^{n-1} + \cdots + a_{i,j,1} * x_h + a_{i,j,0}$

$P_i$ computes $hv = SHA256(f_{i,j}(x_h))$ and uploads $hv$ to blockchain $B$

Algorithm 6: SSVS $(S \longrightarrow f(x), hv)$.

**Require:** $P$ has the clustering results $C^i = \{C_j^i | 1 \le j \le k\}$

(1) $P$:

In $c$ th cluster, $P_1, P_2, \ldots, P_n$ have the set of samples $C_c^1, C_c^2, \ldots, C_c^n$. For ease of expression, we assume here that $C_c^i = \{S_{i,1}\}$. In fact, $C_c^i$ may contains a lot of samples.

(a) **for** $1 \le i \le n$ **do:**

$P_i$ has computed the polynomial values of $S_{i,1}$ and hash values in SVVS protocol, where the polynomial values are $f_{i,1}(x_h), 1 \le h \le n$

**for** $1 \le h \le n$ **do:**

$P_i$ sends $f_{i,1}(x_h)$ to $P_h$ and receive $f_{h,1}(x_i)$ form $P_h$.

$P_i$ computes $hv = SHA256(f_{h,1}(x_i))$.

**while** $hv$ is not on B **do:**

Call $P_h$ resend $f_{h,1}(x_i)$.

(b) **for** $1 \le i \le n$ **do:**

$P_i$ computes $F(x_i) = f_{1,1}(x_i) + f_{2,1}(x_i) + \cdots + f_{n,1}(x_i)$.

$P_i$ sends $F(x_i)$ to C.

(2) $C$:

Solves the set of $F(x)$ using Lagrange's interpolation to find sum of samples $S.sum$, where $S.sum = S_{1,1} + S_{2,1} + \cdots + S_{n,1}$.

Algorithm 7: VRS $(f(x) \longrightarrow S.\text{sum})$.

Table 2: Running time of privacy-preserving $k$-means clustering on S1 dataset in one iteration.

| $n$ | $k$ | $C$ (S) | $P$ (S) |
|---|---|---|---|
| 100 | 7 | 14.24 | 4.18 |
| 200 | 7 | 29.17 | 8.53 |
| 500 | 7 | 74.77 | 20.82 |
| 1000 | 7 | 142.41 | 39.54 |
| 2000 | 7 | 278.79 | 80.56 |

Table 3: Running time of privacy-preserving $k$-means clustering on HCV dataset in one iteration.

| $n$ | $k$ | $C$ (S) | $P$ (S) |
|---|---|---|---|
| 100 | 4 | 4.36 | 1.37 |
| 200 | 4 | 8.46 | 2.69 |
| 588 | 4 | 28.22 | 8.78 |

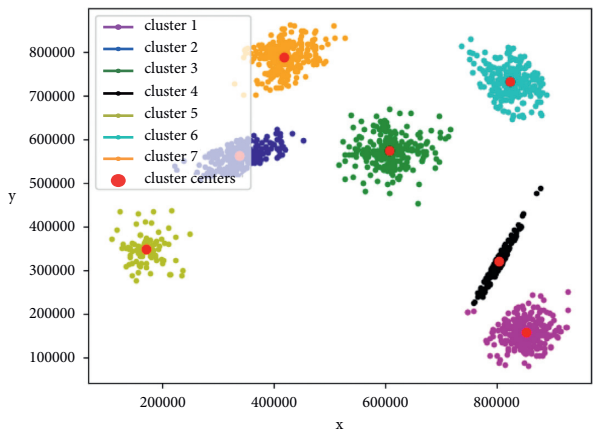


Figure 2: Result of privacy-preserving $k$-means clustering on S1 dataset.

Table 4: Function comparison with [13, 14, 25].

| Scheme | Party | Participants' information protection | Cluster centers protection | Verifiable |
|---|---|---|---|---|
| [25] | Two | ✓ | × | × |
| [14] | Multi | ✓ | ✓ | × |
| [13] | Two | ✓ | × | × |
| Our scheme | Multi | ✓ | ✓ | ✓ |

We calculate the respective time proportions of cloud server and participants to judge the performance of our scheme. In the process of our scheme, the cloud server undertakes the main computing tasks, and running time accounts for 77%.

*6.2. Accuracy Analysis.* We calculate the percentage of correctly classified samples as the standard for the accuracy of our scheme. For dataset S1, we compare the result of our privacy-preserving $k$-means clustering with plaintext $k$-means clustering to measure the accuracy of our scheme. The result of privacy-preserving $k$-means clustering on S1 is shown in Figure 2 and the accuracy of the S1 dataset is 99.25%. In the same way, we calculated the accuracy of the HCV dataset to be 99.14%. So, we can conclude that our scheme can get high-accuracy clustering results.

*6.3. Functional Analysis.* We analyze our scheme's function and compare it with schemes proposed in [13, 14, 25] from different properties, including parties, participants' information protection, cluster centers protection, and verifiability. The result is shown in Table 4. In [25], Jiang et al. combined homomorphic encryption and garbled circuit to design an outsourced two-party privacy-preserving $k$-means clustering scheme. Compared with the scheme in [25], our scheme extends to multi-party. Furthermore, our scheme can verify the shares from other participants with secret sharing, hash function, and blockchain technology. In the step of updating cluster centers, the shares are verified by hash values calculated in advance, and the blockchain is used to ensure that once the hash value is verified, the participants can be guaranteed to share the consistent data. From Table 4, the significant advantages of our scheme compared to the other three schemes are verifiable.

## 7. Conclusion

In this article, we propose a multi-party verifiable privacy-preserving federated $k$-means scheme that provides a validation mechanism in an outsourced environment. By computing hash values of secret shares in advance, we can detect malicious participants that send inconsistent shares and avoid incorrect $k$-means clustering results. The security and experimental analysis show that our scheme can protect privacy and get high clustering results.

## Data Availability

The dataset used to support the findings of this study comes from https://cs.joensuu.fi/sipu/datasets/and https://archive-beta.ics.uci.edu/ml/datasets/hcv+data.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no. 61702067) and in part by the National Natural Science Foundation of Chongqing (no. cstc2020jcyj-msxmX0343).

## References

[1] M. A. Dalal and N. D. Harale, "A survey on clustering in data mining," in *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, pp. 559–562, Mumbai, India, February 2011.

[2] A. Saxena, M. Prasad, A. Gupta et al., "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017.

[3] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Oakland, CA, USA, June 1967.

[4] J. Chen, Y. Liu, Y. Xiang, and K. Sood, "RPPTD: robust privacy-preserving truth discovery scheme," *IEEE Systems Journal*, vol. 2021, Article ID 3099103, 8 pages, 2021.

[5] S. Lv and Y. Liu, "PLVA: privacy-preserving and lightweight V2I authentication protocol," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2021, Article ID 3059638, 7 pages, 2021.

[6] P. Wang and Y. Liu, "SEMA: secure and efficient message authentication protocol for VANETs," *IEEE Systems Journal*, vol. 15, no. 1, pp. 846–855, 2021.

[7] M. Liu, F. Zhao, X. Jiang, H. Zhang, and H. Zhou, "Parallel binary image cryptosystem via spiking neural networks variants," *International Journal of Neural Systems*, Article ID 2150014, 2021.

[8] F. Zhao, M. Liu, K. Wang, and H. Zhang, "Color image encryption via Henon-zigzag map and chaotic restricted Boltzmann machine over Blockchain," *Optics & Laser Technology*, vol. 135, Article ID 106610, 2021.

[9] Y. Liu, Z. Ma, Z. Yan, Z. Wang, X. Liu, and J. Ma, "Privacy-preserving federated k-Means for proactive caching in next generation cellular networks," *Information Sciences*, vol. 521, pp. 14–31, 2020.

[10] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 206–215, ACM, Washington, USA, August 2003.

[11] S. Jha, L. Kruger, and P. Mcdaniel, "Privacy preserving clustering," in *Proceedings of the 10th European Symposium on Research in Computer Security*, pp. 593–599, Milan, Italy, September 2005.

[12] P. Bunn and R. Ostrovsky, "Secure two-party k-means clustering," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 486–497, ACM, Alexandria, Virginia, USA, October 2007.

[13] X. Yi and Y. Zhang, "Equally contributory privacy-preserving k-means clustering over vertically partitioned data," *Information Systems*, vol. 38, no. 1, pp. 97–107, 2013.

[14] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, "Mutual privacy preserving $k$ -means clustering in social participatory sensing," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2066–2076, 2017.

[15] E. Zhang, H. Li, Y. Huang, S. Hong, Le Zhao, and C. Ji, "Practical multi-party private collaborative k-means clustering," *Neurocomputing*, vol. 467, pp. 256–265, 2021.

[16] S. Patel, V. Patel, and D. Jinwala, "Privacy preserving distributed k-Means clustering in malicious model using zero knowledge proof," in *Proceedings of the International Conference on Distributed Computing and Internet Technology*, pp. 420–431, Springer, Bhubaneswar, India, February 2013.

[17] S. Patel, M. Sonar, and D. C. Jinwala, "Privacy preserving distributed k-Means clustering in malicious model using verifiable secret sharing scheme," *International Journal of Distributed Systems and Technologies*, vol. 5, no. 2, pp. 44–70, 2014.

[18] J. Chang, B. Shao, Y. Ji, M. Xu, and R. Xue, "Secure network coding from secure proof of retrievability," *Science China Information Sciences*, vol. 64, no. 12, pp. 229301:1–229301:2, 2021.

[19] H. Cheng, M. Shojafar, M. Alazab, R. Tafazolli, and Y. Liu, "PPVF: privacy-preserving protocol for vehicle feedback in cloud-assisted VANET," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2021, Article ID 3117950, 13 pages, 2021.

[20] Y. Ji, B. Shao, J. Chang, and G. Bian, "Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property," *Cluster Computing*, vol. 24, no. 3, pp. 1–13, 2021.

[21] H. Wang, L. Feng, Y. Ji, B. Shao, and R. Xue, "Toward usable cloud storage auditing, revisited," *IEEE Systems Journal*, vol. 20218 pages, 2021.

[22] R. Zhou, X. Zhang, X. Wang, G. Yang, H.-N. Dai, and M. Liu, "Device-oriented keyword searchable encryption scheme for cloud-assisted industrial IoT," *IEEE Internet of Things Journal*, vol. 2021, Article ID 3124807, 1 page, 2021.

[23] D. Liu, B. Elisa, and X. Yi, "Privacy of outsourced k-Means clustering," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, pp. 123–134, ACM, Kyoto, Japan, June 2014.

[24] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the ACM International Conference on Management of Data and Symposium on Principles of Database Systems*, pp. 439–450, ACM, Dallas, Texas, USA, May 2000.

[25] Z. L. Jiang, N. Guo, Y. Jin et al., "Efficient two-party privacy-preserving collaborative k-Means clustering protocol supporting both storage and computation outsourcing," *Information Sciences*, vol. 518, pp. 168–180, 2020.

[26] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proceedings of the International Conference on Data Engineering*, pp. 664–675, Chicago, USA, March 2014.

[27] P. Mohassel, M. Rosulek, and Y. Zhang, "Fast and secure three-party computation," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pp. 519–602, ACM, Denver, Colorado, USA, October 2015.

[28] Y. Zuo, Z. Zhao, Z. Shi et al., "Highly secure privacy-preserving outsourced k-Means clustering under multiple keys in cloud computing," *Security and Communication Networks*, vol. 2020, pp. 1–11, 2020.

[29] G. Sakellariou and A. Gounaris, "Homomorphically encrypted k-means on cloud-hosted servers with low client-side load," *Computing*, vol. 101, pp. 813–836, 2019.

[30] P. Pallie, "Public-key cryptosystems based on composite degree residuosity classes,"vol. 1592, pp. 223–238, in *Proceedings of the Cryptology-EUROCRYPT'99*, vol. 1592, pp. 223–238, Springer, Prague, Czech Republic, May 1999.

[31] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," Available online: https://bitcoin.org/bitcoin.pdf, 2009.

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

[33] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761–775, 2006.

[34] Hcv data, "UCI machine learning repository," 2020, https://archive-beta.ics.uci.edu/ml/datasets/hcv+data.