Peer-to-Peer Networking and Applications
https://doi.org/10.1007/s12083-020-00881-x

# Privacy-preserving *k*-means clustering with local synchronization in peer-to-peer networks

**Youwen Zhu**[1,2] · **Xingxin Li**[1]

Received: 22 September 2019 / Accepted: 31 January 2020


**Abstract**

*k*-means clustering, which partitions data records into different clusters such that the records in the same cluster are close to each other, has many important applications such as image segmentation and genes detection. While the *k*-means clustering has been well-studied by a significant amount of works, most of the existing schemes are not designed for peer-to-peer (P2P) networks. P2P networks impose several efficiency and security challenges for performing clustering over distributed data. In this paper, we propose a novel privacy-preserving *k*-means clustering scheme over distributed data in P2P networks, which achieves local synchronization and privacy protection. Specifically, we design a secure aggregation protocol and a secure division protocol based on homomorphic encryption to securely compute clusters without revealing the privacy of individual peer. Moreover, we propose a novel massage encoding method to improve the performance of our aggregation protocol. We formally prove that the proposed scheme is secure under the semi-honest model and demonstrate the performance of our proposed scheme.

**Keywords** Privacy-preserving · *k*-means clustering · Peer-to-peer networks

## 1 Introduction

*k*-means clustering [4, 25] is a simple and well-studied technique in machines learning areas which aim to group data records into *k* clusters. Informally, the clustering algorithm partitions data records into different clusters such that records in the same cluster are close to each other based on some distance metrics. Clustering has been applied in many domains, such as image segmentation and genes detection [9, 11, 45]. The traditional *k*-means clustering algorithm is assumed that data is centralized in a single location, which cannot be directly applied to distributed situations where each party holds a part of the dataset.

As a typical distributed system consisting of thousands of connected nodes, peer-to-peer (P2P) networks are emerging as a good choice for many technologies [6, 33, 46]. Performing clustering in P2P networks arises more and more attention. Among this, designing a distributed clustering algorithm in P2P networks instead of collecting data in a single point is a practical option. There are many important applications of distributed clustering in P2P networks. For example, in a P2P file-sharing network, we can utilize the distributed clustering algorithm to mine browsing history of users and find users with common interest [18]. However, P2P networks impose several challenges for performing clustering over distributed data. First, since peer-to-peer networks are highly decentralized and always contain millions of nodes, it is not practical to centralize the whole dataset and achieve global synchronization in the network. Second, there are frequent topology changes in P2P networks. Third, each peer in P2P networks suffers from frequent data update. Fourth, peers may not reveal their local data for the privacy concern.

Many schemes [2, 7, 27, 29] have been proposed to solve the above challenges. Among these works, Datta et al. [7]

This article is part of the Topical Collection: *Special Issue on Security and Privacy in Machine Learning Assisted P2P Networks* Guest Editors: Hongwei Li, Rongxing Lu and Mohamed Mahmoud

✉ Xingxin Li
lixingxin@nuaa.edu.cn

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[2] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, 541004, China

Published online: 03 March 2020

proposed two algorithms to perform $k$-means clustering over distributed data in peer-to-peer networks, which only require that each node achieves local synchronization with its neighbors. Mashayekhi et al. [27] presented a general fully decentralized method to perform clustering over dynamic and distributed data in P2P networks. Their scheme constructed a summarized view of the distributed dataset and utilizes the partition-based and density-based clustering methods to learn the clusters over the summarized view, which achieves effective clustering accuracy and efficient communication costs. Although the above works achieve comparable accuracy with centralized methods, each peer in their schemes is required to send local information to its neighbors for collaborated computations. Since the local data in each peer may contain sensitive information about itself, directly sending local data in the existing distributed clustering schemes compromises the privacy of peers [21, 42–44]. Some recent works [3, 5] designed several privacy-preserving feature selection schemes in P2P networks. Unfortunately, their schemes cannot be directly applied to solve privacy-preserving $k$-means clustering in P2P networks. Among another line of work, many secure schemes [36, 40, 48] are proposed to handle $k$-means clustering over horizontally partitioned or vertically partitioned data. However, these privacy-preserving clustering schemes either only consider the two parties situation or require global synchronization with all parties, which is not suitable for the topology of P2P networks. There still lacks a practical privacy-preserving $k$-mean clustering scheme in P2P networks.

In this paper, we propose a novel privacy-preserving $k$-means clustering scheme over distributed data in P2P networks. Each peer in our scheme iteratively updates the clusters and achieves local synchronization at each iteration, i.e., it only requires to synchronize with its neighbors to learn the clusters. Specifically, each peer implements our secure aggregation protocol to obtain local centers and cluster counts from its neighbors and computes the novel clusters by using our secure division protocol. As a result, each peer can learn the clustering result without revealing its local information. Our main contributions in this paper can be summarized as follows.

- We propose a novel privacy-preserving $k$-means clustering scheme over distributed data in P2P networks, which simultaneously achieves local synchronization and protects the privacy of each peer.
- To protect the privacy of local data in each peer, we design a secure aggregation protocol and a secure division protocol based on homomorphic encryption [31]. In addition, we design a novel message encoding mechanism to improve the performance of our aggregation protocol.
- We formally prove that the proposed scheme is secure under the semi-honest model. We also theoretically analyze the performance of our proposed scheme.

The remainder of this paper is organized as follows. We introduce the system model and threat model in Section 2. We present some preliminaries used in our scheme in Section 3. We describe the proposed scheme in Section 4. Then, we analyze the security and computational complexity of our proposed scheme in Section 5. We review the related work in Section 6. Finally, we conclude the paper in Section 7.

## 2 Problem statement

### 2.1 System model

The dataset consisting of $n$ data records $\{p_1, p_2, \cdots, p_n\}$ is distributed over different nodes in peer-to-peer networks. Each node denotes a user and holds a part of the dataset. In other words, the dataset $D$ is horizontally distributed over different nodes. The nodes try to learn $k$ collaborated clusters $C' = \{c'_1, c'_2, \cdots, c'_k\}$ over the distributed dataset by using the $k$-means clustering algorithm. Nevertheless, the standard clustering algorithm only works on centralized datasets. It is still a challenge to perform $k$-means clustering over distributed datasets in peer-to-peer networks. In this paper, we consider the network as a connected, undirected graph where each peer represents a vertex and each edge between two nodes denotes they can communicate with each other. Our scheme is based on local node synchronization and each node is only synchronizing with its neighbors in each iteration. Thus we simply assume that each node can only communicate with its neighbors and each node has a unique identity. Given a node $N_i$, we use the notion $\Gamma_i$ to denote its neighbors and $|\Gamma_i|$ to represent the number of neighbors. The all notations used in this paper are summarized in Table 1.

### 2.2 Threat model

We consider the security of our proposed scheme under the semi-honest (honest-but-curious) model [10]. That is, each party will correctly follow the protocol, but try to learn other's inputs using what he legally receives during the protocol. A protocol $\Pi$ is secure under the semi-honest

**Table 1** Notation Table

| | |
|---|---|
| $N_i$ | the $i$-the node in peer-to-peer networks. |
| $D_i$ | the local dataset hold by $N_i$. |
| $\Gamma_i$, $\lvert\Gamma_i\rvert$ | the neighbors of a node $N_i$, the number of neighbors. |
| $N_{s_i}$ | the assistant neighbor node of $N_i$. |
| $(pk_i, sk_i)$ | the public and secret key pair of Paillier cryptosystem. |
| $\epsilon$ | the termination threshold of $k$-means clustering algorithm. |
| $C^{(1)} = \left\{c_1^{(1)}, c_2^{(1)}, \cdots, c_k^{(1)}\right\}$ | $k$ initial clusters. |
| $C_i^{(l)} = \left\{c_{i1}^{(l)}, \cdots, c_{ik}^{(l)}\right\}$ | the local clusters of the node $N_i$ at $l$-th iteration. |
| $m_i^{(l)} = \left(m_{i1}^{(l)}, m_{i2}^{(l)}, \cdots, m_{il}^{(l)}\right)$ | the number of records in $N_i$ assigned to $C_i^{(l)}$ at $l$-th iteration. |
| $w_i^{(l)} = \left(w_{i1}^{(l)}, w_{i2}^{(l)}, \cdots, w_{ik}^{(l)}\right)$ | $k$ local centers in the node $N_i$ at $l$-th iteration. |
| $E_{pk}(\cdot)$, $D_{sk}(\cdot)$ | the encryption and decryption functions. |

model if each party's view during the protocol can be simulated given only its input and output. The semi-honest model has been adopted by several existing woks [16, 20, 22, 41]. A formal definition of security against semi-honest adversaries can be described as follows.

**Definition 1** Let $F$ be a functionality and $\Pi$ be a $n$-party protocol for computing $F$. $F_i$ represents the computation on parity $i$. The view of party $i$ during the execution of $\Pi$ is denoted by $View_i$ and equals to $(x_i, r_i, m_1, \cdots, m_t)$ where $x_i$ represents the input, $r_i$ represents the randomness and $m_j$ represents the $j$-th received message. We say that the protocol $\Pi$ is secure under the semi-honest model if there exist a probabilistic polynomial-time simulator $Sim_i$ for each party $i$ such that

$$Sim_i(x_i, F_i(x_1, x_2, \cdots, x_n)) \stackrel{c}{\equiv} view_i(x_i, F_i(x_1, x_2, \cdots, x_n)), \quad (1)$$

where $\stackrel{c}{\equiv}$ represents computational indistinguishability.

### 2.3 Design goal

In our scheme, the dataset $D$ is horizontally distributed over different peers, so each peer can learn the content of different data records. The nodes try to learn $k$ clusters over the whole dataset, thus we assume each node learns the whole clusters during the computation process. The main privacy issues we consider in this paper are listed below.

- A node cannot learn the content of data records possessed by other nodes.
- A node cannot know the closest clusters of data records possessed by other nodes.
- A node cannot learn the number of data records assigned to a cluster for all of the clusters.

## 3 Preliminaries

### 3.1 *k*-means clustering

Given a dataset of $n$ data records $D = \{p_1, p_2, \cdots, p_n\}$, $k$-means clustering partitions the dataset into $k$ disjoint subsets called clusters, where $k$ is a user-defined parameter and $p_i$ is an element of $\mathcal{R}^d$. The goal of $k$-means clustering is to find clusters that achieve the minimized sum of the distances between clusters and data records. In this paper, we use Euclidean distance as the distance metric, i.e., $Dist(p, q) = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2}$, and represent each cluster as its centroid. The $k$-means clustering algorithm [4, 25] can be described as follows.

We first set $l = 1$ and randomly selects $k$ data records $C^{(l)} = \left\{c_1^{(l)}, c_2^{(l)}, \cdots, c_k^{(l)}\right\}$ in the dataset $D$ as the initial clusters and iteratively refine the $k$ potential clusters until reaching a termination condition. Specifically, in each iteration, we assign each data record $p_i$ to the cluster $c_j^{(l)}$ which is closet to it and count the number of such data records, which is represented as $m_j^{(l)}$. Then we compute the novel clusters $C^{(l+1)} = \left\{c_1^{(l+1)}, c_2^{(l+1)}, \cdots, c_k^{(l+1)}\right\}$ based on the cluster assignment, i.e., the new clusters are computed by the arithmetic mean of the points in the clusters. We check whether the difference between the novel and the old clusters is within a predefined threshold. If the condition holds, we terminate the iteration and outputs $C^{(l+1)}$ as the final results; otherwise, we will replace $C^{(l)}$ with $C^{(l+1)}$and repeat the above process. The main steps is illustrated in Fig. 1.

The above algorithm only works on the centralized situation where the whole dataset $D$ is stored in a single place. Nevertheless, the dataset $D$ in peer-to-peer networks is distributed over different nodes. Each node $N_i$ holds a part of the dataset. The specific topology of P2P networks makes it difficult to design a distributed $k$-means clustering

**Fig. 1** $k$-means clustering algorithm

**Input:** A dataset with $n$ records $D = \{p_1, p_2, \cdots, p_n\}$, the number of clusters $k$.
**Output:** Clusters $C = \{c_1, c_2, \cdots, c_k\}$.

1. Randomly selects $k$ data records $C^{(l)} = \{c_1^{(l)}, c_2^{(l)}, \cdots, c_k^{(l)}\}$ as the initial clusters where $l = 1$.
2. For each record $p_i$, find its closest cluster $c_j^{(l)}$ and assign it to the cluster $c_j^{(l)}$.
3. Compute the novel clusters $C^{(l+1)} = \{c_1^{(l+1)}, c_2^{(l+1)}, \cdots, c_k^{(l+1)}\}$ based on the cluster assignment in Step 2.
4. If the difference between the novel and the old clusters is within a predefined threshold, terminate the iteration and outputs $C'^{(l+1)}$ as the final results.
5. If not, replace $C^{(l)}$ with $C^{(l+1)}$ and repeat Step 2 and 3.

algorithm. Moreover, the subset $D_i$ holds by each node may contain sensitive information about himself, thus the node is not willing to reveal this information to others. Designing a distributed $k$-means clustering algorithm maintaining the privacy of each node is still a challenge.

### 3.2 Homomorphic encryption

Paillier cryptosystem [31] is an efficient public key cryptosystem with semantic security (indistinguishability under chosen plaintext attack, IND-CPA). The encryption scheme is additively homomorphic, i.e.,

$$E_{pk}(x_1) \times E_{pk}(x_2) = E_{pk}(x_1 + x_2), \tag{2}$$

$$E_{pk}(x)^{\alpha} = E_{pk}(\alpha x). \tag{3}$$

Here, $E$ denotes the encryption function, $x_1$, $x_2$ and $\alpha$ are arbitrary messages in the plaintext space, $pk$ is the public key, $E_{pk}(x_1)$ represents the ciphertext of $x_1$, and $D$ denote the decryption function. The main steps of the Paiilier encryption system is shown as follows.

**Key Generation**. Choose two large enough primes $p$ and $q$. Then, the secret key $sk = \text{lcm}(p-1, q-1)$, that is, the least common multiple of $p-1$ and $q-1$. The public key $pk = (N, g)$, where $N = pq$ and $g \in \mathbb{Z}_{N^2}^*$ such that $\gcd\left(L(g^s \mod N^2), N\right) = 1$, that is, the maximal common divisor of $L(g^s \mod N^2)$ and $N$ is equivalent to 1. Here, $L(x) = (x-1)/N$, the same below.

**Encryption**. Let $x_0$ be a number in plaintext space $\mathbb{Z}_N$. Select a random $r \in \mathbb{Z}_N^*$ as the secret parameter, then the ciphertext of $x_0$ is $c_0 = \mathrm{E}_{pk}(x_0) = g^{x_0} r^N \bmod N^2$.

**Decryption**. Let $c_0 \in \mathbb{Z}_{N^2}$ be a ciphertext. The plaintext hidden in $c_0$ is

$$x_0 = \mathrm{Dec}_{sk}(c_0) = \frac{L(c_0^s \mod N^2)}{L(g^s \mod N^2)} \mod N.$$

Note that Pailler encryption only supports positive integers, we need to transform real numbers into integers through multiplying them by a large integer $\delta$ ($\delta > 0$) as previous works [23, 24, 35].

## 4 Privacy-preserving $k$-means clustering in P2P networks

### 4.1 Overview

Our proposed privacy-preserving $k$-mean clustering scheme does not require that all nodes in a peer-to-peer network achieve global synchronization. The scheme only requires each node to synchronize with its neighbors, i.e., it only requires local synchronization. Each node moves on to the next generation once it receives responses from all its neighbors. To protect the privacy of each node, we design a secure aggregation protocol and a secure division protocol between each node and its neighbors.

In the initial stage of our scheme, a single node randomly generates $k$ initial clusters $C^{(1)} = \left\{c_1^{(1)}, c_2^{(1)}, \cdots, c_k^{(1)}\right\}$ and a termination threshold $\epsilon > 0$. Then the node sends $(1, C^{(1)}, \epsilon)$ to all its neighbors and starts the iteration 1. When a node $N_i$ receives the message $(1, C^{(1)}, \epsilon)$ form it neighbors for the first time, it randomly chooses a neighbor $N_{s_i}$ as its assistant node. $N_{s_i}$ generate its public and secret key pair $(pk_i, sk_i)$ and sends the public key $pk_i$ to $N_i$. The node $N_i$ then sends $(1, C^{(1)}, \epsilon, pk_i)$ to the remainder of its neighbors and starts the iteration 1. Eventually, all nodes finish the initial stage and enter the iteration 1 with the same initial clusters $C^{(1)}$ and threshold $\epsilon$.

In each iteration of our scheme, each node $N_i$ securely aggregates clusters from his neighbors and computes novel clusters. Specifically, $N_i$ first implement a $k$-means cluster algorithm on its local dataset $D_i$ with location clusters $C_i^{(l)}$. For each data record in $D_i$, $N_i$ calculates the distances between it and each cluster and assign it to the closest cluster $c_{ij}^{(l)}$. Then $N_i$ counts the number of records in his dataset assigned to the cluster $c_{ij}^{(l)}$, which is denoted as $m_{ij}^{(l)}$ and computes $k$ local centers $w_i^{(l)} = \left(w_{i1}^{(l)}, w_{i2}^{(l)}, \cdots, w_{ik}^{(l)}\right)$ where $w_{ij}^{(l)}$ is a $d$-dimensional point and $w_{ij}^{(l)} = \sum_{p \in c_{ij}^{(l)}} p$. $N_i$ stores $\left\{w_i^{(l)}, m_i^{(l)} = \left(m_{i1}^{(l)}, m_{i2}^{(l)}, \cdots, m_{il}^{(l)}\right)\right\}$ in its history table. It also sends a request $(i, l)$ to its neighbor nodes $\Gamma_i$. This request requires all its neighbors

to implement the *secure aggregation protocol* to securely respond their local centers and cluster counts in the iteration $l$. Then, $N_i$ and its assistant node $N_{s_i}$ will learn the secret shares of $\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)}$ and $\sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)}$, i.e., $N_i$ gets $\alpha_{j1}, \beta_{j1}$ and $N_{s_i}$ gets $\alpha_{j2}, \beta_{j2}$ where $\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)} = \alpha_{j1} + \alpha_{j2} \mod N$ and $\sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)} = \beta_{j1} + \beta_{j2} \mod N$. The details about the *secure aggregation protocol* will be described in the next part. $N_i$ then utilizes the *secure division protocol* with the assistant node $N_{s_i}$ to update its clusters. For each cluster $c_{ij}^{(l+1)}$, $N_i$ obtains

$$c_{ij}^{(l+1)} = \frac{\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)}}{\sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)}}.$$

$N_i$ computes $Dist\left(c_{ij}^{(l)}, c_{ij}^{(l+1)}\right)$ and finds the max distance among them. $N_i$ compares $max\left\{Dist\left(c_{ij}^{(l)}, c_{ij}^{(l+1)}\right)\right\}_{1 \le j \le k}$ with $\epsilon$. If $max\left\{Dist\left(c_{ij}^{(l)}, c_{ij}^{(l+1)}\right)\right\}_{1 \le j \le k} > \epsilon$, it continues to next iteration $l+1$; otherwise, it moves on to the termination state and $C_i^{(l)} = \left\{c_{i1}^{(l+1)}, c_{i2}^{(l+1)}, \cdots, c_{ik}^{(l+1)}\right\}$ is the final clusters. The main steps of each iteration in our proposed scheme are illustrated in Fig. 2.

## 4.2 Secure aggregation protocol

In our secure aggregation protocol, $N_i$ securely sums centers and cluster counts from its neighbors $\Gamma_i$. In the protocol, each node $N_a \in \Gamma_i$ holds a local dataset $D_a$, local centers, and cluster counts $\left\{\left(w_a^{(l)}, m_a^{(l)}\right)\right\}$ in its history table. Once receiving a request $(i, \hat{l})$ from $N_i$, $N_a$ first compares its current iteration $l$ with $\hat{l}$. If $\hat{l} \le l$, $N_a$'s history table contains local centers and cluster counts for the iteration $\hat{l}$. $N_a$ finds $\left\{\left(w_a^{(\hat{l})}, m_a^{(\hat{l})}\right)\right\}$ from its history table and encrypts them with the pubic key of $N_{s_i}$. Concretely, for $w_a^{(\hat{l})} = \left(w_{aj1}^{(\hat{l})}, w_{aj2}^{(\hat{l})}, \cdots, w_{ajd}^{(\hat{l})}\right)$ and $m_{aj}^{(\hat{l})}$ $(1 \le j \le k)$, $N_a$ encrypts them as $E_{pk_i}\left(w_{aj}^{(\hat{l})}\right) = \left(E_{pk_i}\left(w_{aj1}^{(\hat{l})}\right), E_{pk_i}\left(w_{aj2}^{(\hat{l})}\right), \cdots, E_{pk_i}\left(w_{ajd}^{(\hat{l})}\right)\right)$, and $E_{pk_i}(m_{aj}^{(\hat{l})})$. $N_a$ then responses

$$E_{pk_i}(w_a^{(\hat{l})}) = \left\{E_{pk_i}\left(w_{a1}^{(\hat{l})}\right), E_{pk_i}\left(w_{a2}^{(\hat{l})}\right), \cdots, E_{pk_i}\left(w_{ak}^{(\hat{l})}\right)\right\},$$

$$E_{pk_i}\left(m_a^{(\hat{l})}\right) = \left\{E_{pk_i}\left(m_{a1}^{(\hat{l})}\right), E_{pk_i}\left(m_{a2}^{(\hat{l})}\right), \cdots, E_{pk_i}\left(m_{ak}^{(\hat{l})}\right)\right\}$$

to $N_i$. If $\hat{l} > l$, $N_a$ puts $(i, \hat{l})$ into its wait table. Then $N_a$ checks the wait table during each iteration $l$. Once $l$ reaches $\hat{l}$, it computes and sends the response $E_{pk_i}(w_a^{(\hat{l})})$ and $E_{pk_i}(m_a^{(\hat{l})})$ based on the above procedure to the node $N_i$.

After receiving all responses from its neighbors $\Gamma_i$,

**Fig. 2** An overview of our privacy-preserving $k$-means clustering algorithm

**Assumption:** Each node $N_i$ holds a local subset $D_i$, the termination condition $\epsilon$, and $k$ initial clusters $C_i^{(l)} = \{c_{i1}^{(l)}, c_{i2}^{(l)}, \cdots, c_{ik}^{(l)}\}$ where $l = 1$.
For each interaction $l$, the node $N_i$ does:

1. For each data record $p$ in $D_i$, assign it to the closest cluster $c_{ij}^{(l)}$. Compute the counts $m_{ij}^{(l)}$ of each cluster $c_{ij}^{(l)}$ $(1 \le j \le k)$ and $k$ local centers $w_i^{(l)} = (w_{i1}^{(l)}, w_{i2}^{(l)}, \cdots, w_{ik}^{(l)})$ where $w_{ij}^{(l)}$ is a $d$-dimensional point and,
$$w_{ij}^{(l)} = \sum_{p \in c_{ij}^{(l)}} p.$$
2. Store $\{(w_i^{(l)}, m_i^{(l)})\}$ in the history table where $m_i^{(l)} = (m_{i1}^{(l)}, m_{i2}^{(l)}, \cdots, m_{il}^{(l)})$.
3. Send a request $(i, l)$ to its neighbor nodes $\Gamma_i$. Then $N_i$ and its assistant node $N_{s_i}$ learn the secret shares of $\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)}$ and $\sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)}$ by using the *secure aggregation protocol*, where $\Gamma_i^* = \Gamma_i \cup N_i$.
4. Utilize the *secure division protocol* with the assistant node $N_{s_i}$ to update its clusters. For each cluster $c_{ij}^{(l+1)}$, we have
$$c_{ij}^{(l+1)} = \frac{\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)}}{\sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)}}.$$
5. If $max\{Dist(c_{ij}^{(l)}, c_{ij}^{(l+1)})\}_{1 \le j \le k} > \epsilon$, the node $N_i$ continues to next iteration $l+1$; otherwise, it moves on to the termination state.

$N_i$ aggregates all messages based on the additively homomorphic property of Paillier encryption. It computes

$$\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_a^{(\hat{l})}\right) = \left\{\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{a1}^{(\hat{l})}\right), \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{a2}^{(\hat{l})}\right), \cdots, \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{ak}^{(\hat{l})}\right)\right\},$$

$$\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_a^{(\hat{l})}\right) = \left\{\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_{a1}^{(\hat{l})}\right), \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_{a2}^{(\hat{l})}\right), \cdots, \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_{ak}^{(\hat{l})}\right)\right\}.$$

Then $N_i$ randomly selects $d$ values $\{r_{j1}, r_{j2}, \cdots, r_{jd}\} \in \mathbb{Z}_N^d$ for each $\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{aj}^{(\hat{l})}\right) = \left(\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{aj1}^{(\hat{l})}\right), \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{aj2}^{(\hat{l})}\right), \cdots, \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{ajd}^{(\hat{l})}\right)\right)$ and computes

$$\alpha_{js} = \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(w_{ajs}^{(\hat{l})}\right) * E_{pk_i}^{-1}(r_{js}),$$

where $1 \le s \le d$. For each $\prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_{aj}^{(\hat{l})}\right)$, $N_i$ randomly selects a value $R_j \in \mathbb{Z}_N$ and calculates

$$\beta_j = \prod_{N_a\in\Gamma_i^*} E_{pk_i}\left(m_{aj}^{(\hat{l})}\right) * E_{pk_i}^{-1}(R_j).$$

$N_i$ sends $\{\alpha_j, \beta_j\}_{1\le j\le k}$ to its assistant node $N_{s_i}$, who will decrypts $\{\alpha_j, \beta_j\}_{1\le j\le k}$ with its secret key $sk_i$ and obtains

$$D_{sk_i}(\alpha_{js}) = \sum_{N_a\in\Gamma_i^*} w_{ajs}^{(\hat{l})} - r_{js} \mod N,$$

$$D_{sk_i}(\beta_j) = \sum_{N_a\in\Gamma_i^*} m_{aj}^{(\hat{l})} - R_j \mod N.$$

After executing the secure aggregation protocol, $N_i$ gets $r_{js}, R_j$ while $N_{s_i}$ gets $D_{sk_i}(\alpha_{js}), D_{sk_i}(\beta_j)$ satisfying $r_{js} + D_{sk_i}(\alpha_{js}) = \sum_{N_a\in\Gamma_i^*} w_{aj}^{(l)} \mod N$, $R_j + D_{sk_i}(\beta_j) = \sum_{N_a\in\Gamma_i^*} m_{aj}^{(l)} \mod N$, i.e., $N_i$ and $N_{s_i}$ learn the secret shares of $\sum_{N_a\in\Gamma_i^*} w_{aj}^{(l)}, \sum_{N_a\in\Gamma_i^*} m_{aj}^{(l)}$ $(1 \le j \le k)$ (Fig. 3).

#### 4.2.1 Optimization

In this part, we propose a novel message encoding mechanism to improve the performance of our secure aggregation protocol. Our encoding method is based on Horner's rule [19]. A formal definition about it is shown as follows.

**Horner's rule** Given a $n$-degree polynomial $p = a_nR^n + a_{n-1}R^{n-1} + \cdots + a_1R + a_0$, we can represent the polynomial as $p = (\cdots(a_nR + a_{n-1})R + \cdots)R + a_0$.

In our encoding method, we construct a polynomial $p = a_nR^n + a_{n-1}R^{n-1} + \cdots + a_1R + a_0$ and ensure $R > max\{a_n, a_{n-1}, \cdots, a_1\}$. If we know $p$ and $R$, we can get $n+1$ coefficients $a_n, a_{n-1}, \cdots, a_0$ based on the Horner's rule, which costs $n+1$ division operations and $n+1$ modulo operations. The detail of our message encoding mechanism in the secure aggregation protocol is described as follows.

We take $m_a^{(\hat{l})} = \left\{m_{a1}^{(\hat{l})}, m_{a2}^{(\hat{l})}, \cdots, m_{ak}^{(\hat{l})}\right\}$ as an example to show the main steps of our novel encoding method. In the original secure aggregation protocol, $N_a$ encrypts $m_a^{(\hat{l})}$ as $E_{pk_i}(m_a^{(\hat{l})}) = \left\{E_{pk_i}(m_{a1}^{(\hat{l})}), E_{pk_i}(m_{a2}^{(\hat{l})}), \cdots, E_{pk_i}(m_{ak}^{(\hat{l})})\right\}$ and sends $E_{pk_i}(m_a^{(\hat{l})})$ to $N_i$. Then $N_i$ randomly selects $k$ values $\{R_1, R_2, \cdots, R_k\} \in \mathbb{Z}_N$ and computes $\beta = \{\beta_1, \beta_2, \cdots, \beta_k\}$ where $\beta_j = \prod_{N_a\in\Gamma_i^*} E_{pk_i}(m_{aj}^{(\hat{l})}) * E_{pk_i}^{-1}(R_j)$. $N_i$ sends $\beta$ to $N_{s_i}$, who decrypts $\beta$ and obtains $\{\sum_{N_a\in\Gamma_i^*} m_{a1}^{(\hat{l})} - R_1 \mod N, \cdots, \sum_{N_a\in\Gamma_i^*} m_{ak}^{(\hat{l})} - R_k \mod N\}$. In the original protocol, we encrypt a $k$-dimensional vector $m_a^{(\hat{l})}$ element-wise, which costs $k$ encryption operations. Note that the plaintext space $N$ of Paillier encryption is usually much larger than $m_{aj}^{(\hat{l})}$, we can integrate a $k$-dimensional vector into an integer and then encrypt the integer, such that the encryption cost can be reduced from $k$ to 1. In our novel encoding method, we choose a value $R$ and encodes $m_a^{\hat{l}} = \left\{m_{a1}^{(\hat{l})}, m_{a2}^{(\hat{l})}, \cdots, m_{ak}^{(\hat{l})}\right\}$ as a $(k-1)$-degree polynomial $p(R)$ as

$$p(m_a^{(\hat{l})}) = m_{a1}^{(\hat{l})} + m_{a2}^{(\hat{l})}R + \cdots + m_{ak}^{(\hat{l})}R^{k-1}.$$

Then we encrypt $p(m_a^{(\hat{l})})$ with public key $pk_i$ and sends $E_{pk_i}(p(m_a^{(\hat{l})}))$ to $N_i$. After receiving all $E_{pk_i}(p(m_a^{\hat{l}}))$ from its neighbors, $N_i$ randomly selects $k$ values $\{r_1, r_2, \cdots, r_k\}$ and encodes it as $p(r) = r_1 + r_2R + \cdots + r_kR^{k-1}$. Then $N_i$ computes

$$\beta = \prod_{N_a\in\Gamma_i^*} E_{pk_i}(p(m_a^{(\hat{l})})) * E_{pk_i}(p(r)).$$

$N_i$ sends $\beta$ to its assistant node $N_{s_i}$. $N_{s_i}$ decrypts $\beta$ and utilizes Horner's rule to get $\{\sum_{N_a\in\Gamma_i^*} m_{a1}^{(\hat{l})} + r_1 \mod N, \cdots, \sum_{N_a\in\Gamma_i^*} m_{ak}^{(\hat{l})} + r_k \mod N\}$.

We require $R > max\{a_n, a_{n-1}, \cdots, a_1\}$ to ensure the correctness of Horner's rule. In our encoding method, we have $a_k = \sum_{N_a\in\Gamma_i^*} m_{ak}^{(\hat{l})} + r_k$. We select proper parameters $R$ and $r_j$ based on the following strategy. Assume $m_{ak}^{(\hat{l})}$ is $\sigma$-bit and the maximum number of $N_a \in \Gamma_i$ is $\theta$. we can set

**Fig. 3** Secure aggregation protocol

**Input:** Each node $N_a \in \Gamma_i$ holds his local dataset $D_a$ and receives a request $(i, \hat{l})$ from $N_i$.
**Output:** $N_i$ and the assistant node $N_{s_i}$ learn the secret shares of $\sum_{N_a \in \Gamma_i^*} w_{aj}^{(l)}, \sum_{N_a \in \Gamma_i^*} m_{aj}^{(l)}$ $(1 \le j \le k)$.
For each node $N_a \in \Gamma_i$

1. If $\hat{l} \le l$,
   - $N_a$ finds $\{(w_a^{(\hat{l})}, m_a^{(\hat{l})})\}$ from its history table, and encrypts them with the pubic key of $N_{s_i}$, i.e.,

$$E_{pk_i}(w_a^{(\hat{l})}) = \{E_{pk_i}(w_{a1}^{(\hat{l})}), E_{pk_i}(w_{a2}^{(\hat{l})}), \cdots, E_{pk_i}(w_{ak}^{(\hat{l})})\},$$

$$E_{pk_i}(m_a^{(\hat{l})}) = \{E_{pk_i}(m_{a1}^{(\hat{l})}), E_{pk_i}(m_{a2}^{(\hat{l})}), \cdots, E_{pk_i}(m_{ak}^{(\hat{l})})\},$$

     where $E_{pk_i}(w_{aj}^{(\hat{l})}) = (E_{pk_i}(w_{aj1}^{(\hat{l})}), E_{pk_i}(w_{aj2}^{(\hat{l})}), \cdots, E_{pk_i}(w_{ajd}^{(\hat{l})}))$ $(1 \le j \le k)$.
   - $N_a$ then sends the response $E_{pk_i}(w_a^{(\hat{l})})$ and $E_{pk_i}(m_a^{(\hat{l})})$ to the node $N_i$.
2. If $\hat{l} > l$, $N_a$ puts $(i, \hat{l})$ into its wait table. Then $N_a$ checks the wait table during each iteration $l$. Once $l$ reaches $\hat{l}$, it computes and sends the response $E_{pk_i}(w_a^{(\hat{l})})$ and $E_{pk_i}(m_a^{(\hat{l})})$ to the node $N_i$.

Once $N_i$ receives responses form all its neighbors,

1. computes

$$\prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_a^{(\hat{l})}) = (\prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{a1}^{(\hat{l})}), \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{a2}^{(\hat{l})}), \cdots, \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{ak}^{(\hat{l})})),$$

$$\prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_a^{(\hat{l})}) = (\prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{a1}^{(\hat{l})}), \prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{a2}^{(\hat{l})}), \cdots, \prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{ak}^{(\hat{l})}))$$

2. For each $\prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{aj}^{(\hat{l})}) = (\prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{aj1}^{(\hat{l})}), \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{aj2}^{(\hat{l})}), \cdots, \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{ajd}^{(\hat{l})}))$, randomly selects $d$ values $\{r_{j1}, r_{j2}, \cdots, r_{jd}\} \in \mathbb{Z}_N^d$ and calculates $\alpha_j = (\alpha_{j1}, \cdots, \alpha_{jd})$, where

$$\alpha_{js} = \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{ajs}^{(\hat{l})}) * E_{pk_i}^{-1}(r_{js}).$$

3. For each $\prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{aj}^{(\hat{l})})$, randomly selects a value $R_j \in \mathbb{Z}_N$ and calculates

$$\beta_j = \prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{aj}^{(\hat{l})}) * E_{pk_i}^{-1}(R_j).$$

4. Sends $\{\alpha_j, \beta_j\}_{1 \le j \le k}$ to the assistant node $N_{s_i}$.

$N_{s_i}$ receives messages from $N_i$ and decrypts $\{\alpha_j, \beta_j\}_{1 \le j \le k}$ with its secret key $sk_i$,

$$D_{sk_i}(\alpha_{js}) = \sum_{N_a \in \Gamma_i^*} w_{ajs}^{(\hat{l})} - r_{js} \mod N,$$

$$D_{sk_i}(\beta_j) = \sum_{N_a \in \Gamma_i^*} m_{aj}^{(\hat{l})} - R_j \mod N.$$

$R$ to a $(\sigma+\theta+\kappa)$-bit integer and selects $r_j$ from the range $[0, 2^{\kappa}]$.

### 4.3 Secure division protocol

In our secure division protocol, values $a, b$ are shared between two nodes $N_i, N_j$ where $N_i$ holds $a_1, b_1$ and $N_j$ holds $a_2, b_2$ satisfying $a = a_1 + a_2 \mod N, b = b_1 + b_2 \mod N$. After executing the protocol, $N_i$ learns the value $\frac{a}{b}$ while $N_j$ cannot learn any useful information about $a, b$ and $\frac{a}{b}$.

$N_i$ first generates the public and secret key pair $(pk, sk)$ of Paillier cryptosystem and encrypts $a_1, b_1$ with the public key $pk$. Then $N_i$ sends encrypted values $(E_{pk}(a_1), E_{pk}(b_1))$ to $N_j$. After receiving messages from $N_i$, $N_j$ selects a non-zero random value $\lambda \in \mathbb{Z}_n$ and computes

$$E_{pk}(\lambda a) = (E_{pk}(a_1) * E_{pk}(a_2))^{\lambda},$$

$$E_{pk}(\lambda b) = (E_{pk}(b_1) * E_{pk}(b_2))^{\lambda}.$$

$N_j$ then sends $E_{pk}(\lambda a)$ and $E_{pk}(\lambda b)$ to $N_i$, who decrypts received values with the secret key $sk$ and obtains $\lambda a \mod N$, $\lambda b \mod N$. Finally, $N_i$ obtains the division by computing $\frac{a}{b} = \frac{\lambda a}{\lambda b}$ (Fig. 4).

## 5 Evaluation

### 5.1 Security analysis

We consider the security of our scheme under the semi-honest model [10]. The security of our scheme can be proved based on the following theorems.

**Theorem 1** *Our secure aggregation protocol is secure under the semi-honest model. Each participant cannot learn any useful information about others.*

*Proof* Our secure aggregation protocol involves participants with three different types: neighbor nodes $N_a \in \Gamma_i$, the node $N_i$, and the assistant node $N_{s_i}$. We prove the theorem by considering each participant is corrupted in turn by an adversary. We show that we can construct a computationally indistinguishable simulator to simulate the corrupted party's view.

When the node $N_i$ is corrupted, we construct a simulator $Sim_i$ to simulate $N_i$'s view as follows. In a real execution, the $N_i$'s view $View_i$ is as follow:

$$View_i = \Big\{w_i^{(\hat{l})}, m_i^{(\hat{l})}, (E_{pk_i}(w_a^{(\hat{l})}), E_{pk_i}(m_a^{(\hat{l})}))_{N_a \in \Gamma_i}, (\alpha_j, \beta_j)_{1 \le j \le k}, \times (r_{js})_{1 \le j \le k, 1 \le s \le d}, (R_j)_{1 \le j \le k}\Big\}$$

In the above $View_i$, $w_i^{(\hat{l})}, m_i^{(\hat{l})}$ are the input, $(E_{pk_i}(w_a^{(\hat{l})}), E_{pk_i}(m_a^{(\hat{l})}), \alpha_j, \beta_j)$ are the ciphertexts of Paillier encryption, and $r_{js}, R_j$ are random value selected from $\mathbb{Z}_N$. To simulate the $N_i$'s view $View_i$, $Sim_i$ randomly selects $w_a^*, m_a^*$ from $\mathbb{Z}_N$ and encrypts them with the public key $pk_i$. Then it randomly selects $(r_{js}^*, R_j^*)_{1 \le j \le k}$ from $\mathbb{Z}_N$ and computes $\alpha_j^*, \beta_j^*$ as $\alpha_{js}^* = \prod_{N_a \in \Gamma_i^*} E_{pk_i}(w_{ajs}^*) * E_{pk_i}^{-1}(r_{js}^*), \beta_j^* = \prod_{N_a \in \Gamma_i^*} E_{pk_i}(m_{aj}^*) * E_{pk_i}^{-1}(R_j^*)$. The simulator $Sim_i = \{w_i^{(\hat{l})}, m_i^{(\hat{l})}, (E_{pk_i}(w_a^*), E_{pk_i}(m_a^*))_{N_a \in \Gamma_i}, (\alpha_j^*, \beta_j^*)_{1 \le j \le k}, (r_{js}^*)_{1 \le j \le k, 1 \le s \le d}, (R_j^*)_{1 \le j \le k}\}$. In both $View_i$ and $Sim_i$, the input are identical. $(E_{pk_i}(w_a^{(\hat{l})}), E_{pk_i}(m_a^{(\hat{l})}), \alpha_j, \beta_j)$ and $(E_{pk_i}(w_a^*), E_{pk_i}(m_a^*)), (\alpha_j^*, \beta_j^*)$ are the ciphertexts of Paillier encryption. Since Paillier encryption is

**Fig. 4** Secure Division protocol

**Input:** $N_i$ holds $a_1, b_1$ while $N_j$ holds $a_2, b_2$
**Output:** $N_i$ learns the value $\frac{a}{b}$ where $a = a_1 + a_2 \mod N, b = b_1 + b_2 \mod N$.

1. $N_i$ generates the public and secret key pair $(pk, sk)$ of Paillier cryptosystem.
2. $N_i$ encrypts $a_1, b_1$ with his public key and sends $(E_{pk}(a_1), E_{pk}(b_1))$ to $N_j$.
3. $N_j$ selects a non-zero random value $\lambda \in \mathbb{Z}_n$, computes

$$E_{pk}(\lambda a) = (E_{pk}(a_1) * E_{pk}(a_2))^{\lambda},$$

$$E_{pk}(\lambda b) = (E_{pk}(b_1) * E_{pk}(b_2))^{\lambda},$$

   and returns them to $N_i$.
4. After receiving responses from $N_j$, $N_i$ decrypts them with his secret ke $sk$ and obtains $\lambda a$ and $\lambda b$.
5. $N_i$ computes $\frac{a}{b} = \frac{\lambda a}{\lambda b}$.

semantically secure, $(E_{pk_i}(w_a^{(\hat{l})}), E_{pk_i}(m_a^{(\hat{l})}), \alpha_j, \beta_j)$ and $(E_{pk_i}(w_a^*), E_{pk_i}(m_a^*)), (\alpha_j^*, \beta_j^*)$ are computationally indistinguishable. Since $r_{js}, R_j$ and $r_{js}^*, R_j^*$ are all random values in $\mathbb{Z}_N$, they are indistinguishable. Therefore, we can claim that $Sim_i$ is computationally indistinguishable from $View_i$.

When the assistant node $N_{s_i}$ is corrupted, we can construct a simulator $Sim_{s_i}$ as follows. The $N_{s_i}$'s view is

$$View_{s_i} = \{(D_{sk_i}(\alpha_{js}))_{1\le j\le k, 1\le s\le d}, (D_{sk_i}(\beta_j))_{1\le j\le k}\}.$$

To simulate $View_{s_i}$, $Sim_{s_i}$ randomly chooses $\gamma_{js}^*$ and $\eta_j^*$ from $\mathbb{Z}_N$ once $N_{s_i}$ obtains $D_{sk_i}(\alpha_{js})$ and $D_{sk_i}(\beta_j)$. The simulator $Sim_{s_i} = \{(\gamma_{js})_{1\le j\le k, 1\le s\le d}, (\eta_j^*)_{1\le j\le k}\}$. Since both $D_{sk_i}(\alpha_{js})$ and $D_{sk_i}(\beta_j)$ are masked with random values, $\gamma_{js}^*, \eta_j^*$ are indistinguishable from $D_{sk_i}(\alpha_{js})$, $D_{sk_i}(\beta_j)$. Therefore, we can conclude that $Sim_{s_i}$ is computationally indistinguishable from $View_{s_i}$.

The case where neighbors $N_a \in \Gamma_i$ is corrupted. The node $N_a$ in our protocol only receives $(i, \hat{l})$ from $N_i$. These messages are all public parameters, thus an adversary cannot learn any useful information once he corrupts the neighbor $N_a \in \Gamma_i$.

Combining the above, we can conclude that the secure aggregation protocol is secure under the semi-honest model. □

**Theorem 2** *Our secure division protocol is secure under the semi-honest model.*

*Proof* Our secure division protocol involves two participants $N_i$ and $N_j$. We construct a computationally indistinguishable simulator to simulate the corrupted party's view as follows.

The $N_j$'s view during the protocol is $View_j = \{E_{pk}(a_1), E_{pk}(b_1), \lambda, E_{pk}(\lambda a), E_{pk}(\lambda b)\}$. Once $N_j$ receives $E_{pk}(a_1), E_{pk}(b_1)$, $Sim_j$ randomly selects $a_1^*, b_1^*$ from $\mathbb{Z}_N$ and encrypts them with the public key $pk$. Then $Sim_j$ randomly selects a non-zero value $\lambda^*$ from $\mathbb{Z}_N$ and computes $E_{pk}(\lambda^* a^*) = (E_{pk}(a_1^*) * E_{pk}(a_2^*))^{\lambda^*}$ and $E_{pk}(\lambda^* b^*) = (E_{pk}(b_1^*) * E_{pk}(b_2^*))^{\lambda^*}$. The simulator $Sim_j = \{E_{pk}(a_1^*), E_{pk}(b_1^*), \lambda^*, E_{pk}(\lambda^* a^*), E_{pk}(\lambda^* b^*)\}$. Since $\lambda$ and $\lambda^*$ are randomly selected values and Pallier cryptosystem is semantically secure, $Sim_j$ is computationally indistinguishable from $View_j$.

The $N_i$'s view in this protocol is $View_i = \{\lambda a, \lambda b\}$. To simulate it, $Sim_i$ selects a non-zero value $a^*$ from $\mathbb{Z}_N$ and computes $\frac{a}{b}a^*$. The simulator $Sim_i = \{a^*, \frac{a}{b}a^*\}$. Since $\lambda a, \lambda b$ are masked with random values, they are indistinguishable from $a^*, \frac{a}{b}a^*$. Thus, $Sim_i$ is computationally indistinguishable from $View_i$.

Based on the above analysis, we can claim that our secure division protocol is secure under the semi-honest model. □

**Theorem 3** *Our privacy-preserving k-means clustering protocol is secure under the semi-honest model if the secure aggregation protocol and the secure division protocol are secure under the semi-honest model.*

*Proof* In our $k$-means clustering protocol, each node $N_i$ first computes local centers $w_i^{(l)}$ and cluster counts $m_i^{(l)}$. Then it implements the secure aggregation protocol with its neighbors $\Gamma_i$ to securely sum local centers and cluster counts. We have learned that the secure aggregation protocol is secure under the semi-honest model in Theorem 1. Thus $N_i$ cannot learn any useful information about other node's local data. Then $N_i$ runs the secure division protocol with $N_{s_i}$ to compute the next iteration clusters. Theorem 2 has proved that the secure division protocol is secure under the semi-honest model. Therefore, $N_i$ cannot learn any useful information about the aggregation result expect the novel clusters. Based on the above analysis, we can claim that our $k$-means clustering protocol is secure under the semi-honest model. □

## 5.2 Complexity analysis

In this part, we analyze the computational complexity of our proposed scheme. For simplicity, we omit the cost of operations over plaintexts and focus the time-consuming operations on ciphertexts including encryption operations $E$, exponentiation operations $Exp$, and decryption operations $D$. The computational complexity of our proposed protocols is shown in Table 2. The detailed analysis is described as follows.

We first analyze the cost of our original secure aggregation protocol. In the proposed protocol, each node $N_a \in \Gamma_i$ first encrypts $(w_a^{(\hat{l})}, m_a^{(\hat{l})})$, which requires $(dk + k)$ encryption operations. $N_i$ computes $\prod_{N_a\in\Gamma_i^*} E_{pk_i}(w_a^{(\hat{l})})$ which costs $|\Gamma_i^*|kd$ exponentiation operations and $\prod_{N_a\in\Gamma_i^*} E_{pk_i}(m_a^{(\hat{l})})$ which takes $|\Gamma_i^*|k$ exponentiation operations. Then $N_i$ selects random values and computes $\{\alpha_j, \beta_j\}_{1\le j\le k}$, which requires $(kd + k)$ encryption operations and $(kd + k)$ exponentiation operations. $N_{s_i}$ decrypts $\{\alpha_j, \beta_j\}_{1\le j\le k}$which takes $(kd+k)$ decryption operations. The overall computational cost in our original secure aggregation protocol is $O(|\Gamma_i^*|kd)$ encryption operations, $O(kd)$ decryption operations, and $O(|\Gamma_i^*|kd)$ exponentiation operations. In our optimized secure aggregation protocol, we encode vectors $w_{aj}^{(\hat{l})}$ and $m_a^{(\hat{l})}$ into two integers respectively. Thus $N_a \in \Gamma_i$ only requires $(k + 1)$ encryption operations to encrypt $(w_a^{(\hat{l})}, m_a^{(\hat{l})})$. $N_i$ takes $|\Gamma_i^*|k$ exponentiation operations to compute $\prod_{N_a\in\Gamma_i^*} E_{pk_i}(w_a^{(\hat{l})})$ and $|\Gamma_i^*|$ exponentiation operations to

**Table 2** The computation complexity of proposed protocols

| | Encryption | Decryption | Exponentiation |
|---|---|---|---|
| Original secure aggregation protocol | $O(\|\Gamma_i^*\|kd)$ | $O(kd)$ | $O(\|\Gamma_i^*\|kd)$ |
| Optimized secure aggregation protocol | $O(\|\Gamma_i^*\|k)$ | $O(k)$ | $O(\|\Gamma_i^*\|k)$ |
| Secure division protocol | $O(1)$ | $O(1)$ | $O(\lambda)$ |
| PP$k$M$_b$ | $O((d+\|\Gamma_i^*\|)kdl)$ | $O(kdl)$ | $O((d\lambda+\|\Gamma_i^*\|)kdl)$ |
| PP$k$M$_e$ | $O((d+\|\Gamma_i^*\|)kl)$ | $O(kl)$ | $O((d\lambda+\|\Gamma_i^*\|)kl)$ |

compute $\prod_{N_a\in\Gamma_i^*} E_{pk_i}(m_a^{(\hat{l})})$. Then $N_i$ encodes random values and computes $\{(\alpha_1,\alpha_2,\cdots,\alpha_k),\beta\}$ which requires $(k+1)$ encryption operations and $(k+1)$ exponentiation operations. $N_{s_i}$ decrypts $\{(\alpha_1,\alpha_2,\cdots,\alpha_k),\beta\}$ to learn the result which takes $(k+1)$ decryption operations. The overall computational cost in our optimized secure aggregation protocol is $O(|\Gamma_i^*|k)$ encryption operations, $O(k)$ decryption operations, and $O(|\Gamma_i^*|k)$ exponentiation operations.

In our secure division protocol, $N_i$ encrypts $a_1, b_2$ which requires 2 encryption operations. $N_j$ encrypts $a_2, b_2$ and computes $E_{pk}(\lambda a)$ and $E_{pk}(\lambda b)$ based on the additively homomorphic property of Paillier encryption, which needs 2 encryption operations and $2\lambda$ exponentiation operations. $N_i$ then receives responses from $N_j$ and decrypts the messages which requires 2 decryption operations. The overall computational cost of our secure division protocol is $O(1)$ encryption operations, $O(1)$ decryption operations, and $O(\lambda)$ exponentiation operations.

In each iteration of our privacy-preserving $k$-means clustering protocol, the node $N_i$ implements the secure aggregation protocol to gather centers and clustering counts from all its neighbors and securely share this information with its assistant node $N_{s_i}$. $N_i$ then implements the secure division protocol to compute local clusters. The update procedure needs to invoke the secure division protocol $kd$ times to calculate the new clusters $C_i^{(l+1)}$. Therefore, the computational cost of single node in our privacy-preserving $k$-means clustering scheme with the original secure aggregation protocol denoted by PP$k$M$_b$ in each iteration is $O((d+|\Gamma_i^*|)kd)$ encryption operations, $O(kd)$ decryption operations, and $O(kd(d\lambda+|\Gamma_i^*|))$ exponentiation operations. Assume the number of iteration is $l$, the overall computational cost of single node in our PP$k$M$_b$ scheme is $O((d+|\Gamma_i^*|)kdl)$ encryption operations, $O(kdl)$ decryption operations, and $O((d+|\Gamma_i^*|)kdl)$ exponentiation operations. Similarly, the computational cost of single node in our privacy-preserving $k$-means clustering scheme with the optimized secure aggregation protocol denoted by PP$k$M$_e$ is $O((d+|\Gamma_i^*|)kl)$ encryption operations, $O(kl)$ decryption operations, and $O((d+|\Gamma_i^*|)kl)$ exponentiation operations.

### 5.3 Experiments

In this part, we evaluate the performance of our proposed scheme under various parameter setting. The dataset we used is a real dataset from the UCI repository,[1] which consists of 56,000 data records and each record has 8 attributes. We assume that the number of data records in each peer node is same and is fixed to 500 in our experiments. We implement the Paillier encryption by using Paillier library[2] and conducts all experiments on a 4-core CPU, 16GB RAM machine. The key size of Paillier encryption is set to 2048 bits and the number of clusters $k$ in the experiments is set to 8. We vary the number of neighbors of each node $|\Gamma_i|$ to evaluate computation time of single node at each iteration. The experiment results are shown in Table 3. We can learn from the results that the computation time of both PP$k$M$_b$ and PP$k$M$_e$ increases with the value $|\Gamma_i|$ and the cost of PP$k$M$_e$ is smaller than that of PP$k$M$_b$. For example, when $|\Gamma_i|$ is 5, the computation time of PP$k$M$_b$ is 0.72 seconds while the corresponding time of PP$k$M$_e$ is only 0.12 seconds; when $|\Gamma_i|$ grows to 30, the computation time of PP$k$M$_b$ ups to 4.63 seconds while the corresponding time of PP$k$M$_e$ is only 0.74 seconds. The experiment results indicate that our novel encoding method based on Horner's rule can effectively reduce the computation costs.

## 6 Related work

### 6.1 Distributed machine learning in peer-to-peer networks

A large number of works have been proposed to perform machine learning over distributed data in peer-to-peer networks. Luo et al. [26] proposed an ensemble scheme for distributed classification in peer-to-peer networks. In

[1] http://archive.ics.uci.edu/ml

[2] http://python-paillier.readthedocs.io/en/stable/index.html.

**Table 3** The computation time of single node at one iteration. (Seconds)

| | $\|\Gamma_i\| = 5$ | $\|\Gamma_i\| = 10$ | $\|\Gamma_i\| = 15$ | $\|\Gamma_i\| = 20$ | $\|\Gamma_i\| = 25$ | $\|\Gamma_i\| = 30$ |
|---|---|---|---|---|---|---|
| PP$k$M$_b$ | 0.72 | 1.63 | 2.53 | 3.27 | 4.01 | 4.63 |
| PP$k$M$_e$ | 0.12 | 0.25 | 0.39 | 0.51 | 0.62 | 0.74 |

the proposed scheme, they constructed local classifiers on each peer by using the learning algorithm of pasting bites and proposed a distributed plurality voting protocol to combine the decisions by local classifiers. Wolff et al. [39] presented a generic algorithm to calculate any ordinal function of the average data in a large peer-to-peer network and proposed a general framework to consequently update any model of distributed data. Datta et al. [7] proposed two algorithms to perform $k$-means clustering over distributed data in peer-to-peer networks. Their scheme avoids large-scale synchronization or data centralization and only requires that each node achieves local synchronization with its neighbors. Ang et al. [1] combined cascade support vector machine (SVM) and reduced SVM methods to construct SVM classifiers in peer-to-peer networks. Their scheme achieves comparable classification accuracy with centralized classifiers. Kan et al. [17] presented a collaborative classification method to build SVM classifiers in scale-free peer-to-peer networks. The proposed method improves the local classification accuracy through propagating SVM models with the most influence. Ormándi et al. [30] proposed a general approach named gossip learning to combine local classification models based on multiple models taking random walks and virtual weighted voting mechanisms. Papapetrou et al. [32] presented a collaborative approach for document classification in peer-to-peer networks. Their scheme constructs local classification models on each node and combines the most discriminative model to construct the collaborative classification model. Mashayekhi et al. [27] presented a general fully decentralized method to perform clustering over dynamic and distributed data in peer-to-peer networks. Their scheme first constructs a summarized view of the distributed dataset through decentralized gossip-based communication and then utilizes the partition-based and density-based clustering methods to learn the clusters over the summarized view. The above solutions mainly focus on designing machine learning algorithms over distributed data, which maintain high accuracy and low computation and communication costs without centralizing distributed data in a single node. Although some of the above works achieve comparable accuracy with centralized methods, they fail to consider the privacy issues when performing machine learning over distributed data. Some nodes may not want to reveal his local data or model to others for privacy consideration.

### 6.2 Privacy-preserving distributed machine learning

Privacy-preserving machine learning over distributed data has been investigated by several secure schemes, such as privacy-preserving naive Bayes classification [37, 38, 49], secure support vector machine [15, 47, 50], and privacy-preserving deep learning [28, 34]. For the $k$-means clustering, Vaidya et al. [36] utilized secure permutation protocol and homomorphic encryption to construct the first privacy-preserving $k$-means algorithm for vertically partitioned data where each party holds a portion of the attributes of data records. The work [8] presented a secure multi-party clustering scheme over vertically partitioned data based on additively secret sharing, which achieves better computational performance than existing works. works [14] proposed secure protocols based on oblivious polynomial evaluation and homomorphic encryption to perform $k$-means clustering over horizontally partitioned data where each party holds different data records in the dataset. Yu et al. [48] proposed a secure multi-party $k$-means clustering scheme where they consider both horizontally partitioned and vertically partitioned datasets. Jagannathan et al. [13] proposed a secure scheme to perform $k$-means clustering over arbitrarily partitioned data based on random shares and Yao's garbled circuits [12]. Xing et al. [40] designed a mutual privacy-preserving $k$-means clustering scheme in social participatory sensing environments, which preserves both each party's private information and global clusters.

There are also some works considered the privacy-preserving machine learning in peer-to-peer networks. Das et al. [5] proposed a privacy-preserving feature selection scheme over distributed data in large peer-to-peer networks. Their scheme incorporated misclassification gain, Gini index, and entropy feature measurement and combines the secure sum protocol with the Bayes optimal privacy model to aggregate features without revealing the privacy of each node. Bhuyan et al. [3] utilized the fuzzy methodologies technique to design a privacy-preserving sub-feature section scheme in distributed environments.

## 7 Conclusion

In this paper, we proposed a novel privacy-preserving $k$-means clustering scheme in peer-to-peer networks. In our

scheme, we designed a secure aggregation protocol to learn the sum of centers and clustering counts from neighbors and a secure division protocol to perform division operations over shared values. Moreover, we presented a novel message encoding method based on Horner's rule to improve the performance of our aggregation protocol. Compared with existing solutions, our scheme achieves local synchronization and privacy protection in each peer. We also formally proved the security of our proposed scheme and analyzed the computational complexity of our proposed scheme.

**Acknowledgements** This work is partly supported by the National Key Research and Development Program of China (No. 2017YFB0802300), the Natural Science Foundation of China (No. 61602240), the Postgraduate Research & Practice Innovation Program of Jiangsu Province (No. KYCX18_0305), and the Research Fund of Guangxi Key Laboratory of Trusted Software (No. kx201906).

## Compliance with Ethical Standards

**Conflict of interests** The authors declare that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Informed Consent** Informed consent was obtained from all individual participants included in the study.

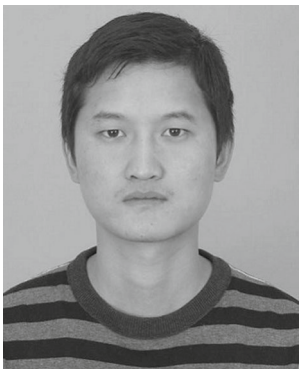
## References

1. Ang HH, Gopalkrishnan V, Hoi SC, Ng WK (2008) Cascade rsvm in peer-to-peer networks. In: Joint European conference on machine learning and knowledge discovery in databases, pp 55–70. Springer
2. Bandyopadhyay S, Giannella C, Maulik U, Kargupta H, Liu K, Datta S (2006) Clustering distributed data streams in peer-to-peer environments. Inform Sci 176(14):1952–1985
3. Bhuyan HK, Kamila NK (2015) Privacy preserving sub-feature selection in distributed data mining. Appl Soft Comput 36:552–569
4. Chien Y (1974) Pattern classification and scene analysis. IEEE Trans Autom Control 19(4):462–463
5. Das K, Bhaduri K, Kargupta H (2010) A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. Knowl Inf Syst 24(3):341–367
6. Datta S, Bhaduri K, Giannella C, Wolff R, Kargupta H (2006) Distributed data mining in peer-to-peer networks. IEEE Internet Comput 10(4):18–26
7. Datta S, Giannella C, Kargupta H (2008) Approximate distributed k-means clustering over a peer-to-peer network. IEEE Trans Knowl Data Eng 21(10):1372–1388
8. Doganay MC, Pedersen TB, Saygin Y, Savaş E, Levi A (2008) Distributed privacy preserving k-means clustering with additive secret sharing. In: Proceedings of the 2008 international workshop on privacy and anonymity in information society, pp 3–11. ACM
9. Gligorijević V, Pržulj N (2015) Methods for biological data integration: perspectives and challenges. J R Soc Interface 12(112):20150,571
10. Goldreich O (2004) Foundations of cryptography: Volume II, Basic Applications. Cambridge University Press, Cambridge
11. Hao M, Li H, Luo X, Xu G, Yang H, Liu S (2019) Efficient and privacy-enhanced federated learning for industrial artificial intelligence. IEEE Transactions on Industrial Informatics pp 1–1. https://doi.org/10.1109/TII.2019.2945367
12. Huang Y, Evans D, Katz J, Malka L (2011) Faster secure two-party computation using garbled circuits. In: USENIX security symposium, vol 201, pp 331–335
13. Jagannathan G, Wright RN (2005) Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery in data mining, pp 593–599. ACM
14. Jha S, Kruger L, McDaniel P (2005) Privacy preserving clustering. In: European symposium on research in computer security, pp 397–417. Springer
15. Jia Q, Guo L, Jin Z, Fang Y (2018) Preserving model privacy for machine learning in distributed systems. IEEE Trans Parallel and Distrib Syst 29(8):1808–1822
16. Jiang W, Li H, Xu G, Wen M, Dong G, Lin X (2019) Ptas: Privacy-preserving thin-client authentication scheme in blockchain-based pki. Futur Gener Comput Syst 96:185–195
17. Khan U, Schmidt-Thieme L, Nanopoulos A (2017) Collaborative svm classification in scale-free peer-to-peer networks. Expert Syst Appl 69:74–86
18. Koskela T, Kassinen O, Harjula E, Ylianttila M (2013) P2p group management systems: A conceptual analysis. ACM Comput Surv (CSUR) 45(2):20
19. Levitin A (2012) Introduction to the design & analysis of algorithms. Pearson Education
20. Li H, Liu D, Dai Y, Luan TH, Yu S (2018) Personalized search over encrypted data with efficient and secure updates in mobile clouds. IEEE Trans Emerg Topics Comput 6(1):97–109
21. Li H, Yang Y, Dai Y, Yu S, Xiang Y (2017) Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Transactions on Cloud Computing pp 1–1. https://doi.org/10.1109/TCC.2017.2769645
22. Li X, Zhu Y, Wang J (2019) Highly efficient privacy preserving location-based services with enhanced one-round blind filter. IEEE Transactions on Emerging Topics in Computing. https://doi.org/10.1109/TETC.2019.2926385
23. Li X, Zhu Y, Wang J, Liu Z, Liu Y, Zhang M (2018) On the soundness and security of privacy-preserving svm for outsourcing data classification. IEEE Trans Depend Secure Comput 15(5):906–912
24. Liu Y, Zhao Q (2019) E-voting scheme using secret sharing and k-anonymity. World Wide Web 22(4):1657–1667
25. Lloyd S (1982) Least squares quantization in pcm. IEEE Trans Inf Theory 28(2):129–137
26. Luo P, Xiong H, Lü K, Shi Z (2007) Distributed classification in peer-to-peer networks. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 968–976. ACM
27. Mashayekhi H, Habibi J, Khalafbeigi T, Voulgaris S, Van Steen M (2015) Gdcluster: A general decentralized clustering algorithm. IEEE Trans Knowl Data Eng 27(7):1892–1905
28. Mohassel P, Zhang Y (2017) Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP), pp 19–38. IEEE
29. Muller WT, Eisenhardt M, Henrich A (2003) Efficient content-based p2p image retrieval using peer content descriptions. In: Internet Imaging V, vol 5304, pp. 57–68. International Society for Optics and Photonics
30. Ormándi R, Hegedűs I, Jelasity M (2013) Gossip learning with linear models on fully distributed data. Concurr Comput Pract Exp 25(4):556–571

31. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques, pp. 223–238. Springer
32. Papapetrou O, Siberski W, Siersdorfer S (2015) Efficient model sharing for scalable collaborative classification. Peer-to-Peer Netw Appl 8(3):384–398
33. Ren H, Li H, Dai Y, Yang K, Lin X (2018) Querying in internet of things with privacy preserving: Challenges, solutions and opportunities. IEEE Netw 32(6):144–151
34. Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp 1310–1321. ACM
35. Song J, Liu Y, Shao J, Tang C (2019) A dynamic membership data aggregation (dmda) protocol for smart grid. IEEE Systems Journal. https://doi.org/10.1109/JSYST.2019.2912415
36. Vaidya J, Clifton C (2003) Privacy-preserving k-means clustering over vertically partitioned data. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 206–215. ACM
37. Vaidya J, Clifton C (2004) Privacy preserving naive bayes classifier for vertically partitioned data. In: Proceedings of the 2004 SIAM international conference on data mining, pp 522–526. SIAM
38. Vaidya J, Kantarcıoğlu M, Clifton C (2008) Privacy-preserving naive bayes classification. The VLDB J 17(4):879–898
39. Wolff R, Bhaduri K, Kargupta H (2008) A generic local algorithm for mining data streams in large distributed systems. IEEE Trans Knowl Data Eng 21(4):465–478
40. Xing K, Hu C, Yu J, Cheng X, Zhang F (2017) Mutual privacy preserving *k*-means clustering in social participatory sensing. IEEE Trans Ind Inform 13(4):2066–2076
41. Xu G, Li H, Dai Y, Yang K, Lin X (2019) Enabling efficient and geometric range query with access control over encrypted spatial data. IEEE Trans Inf Forensics Secur 14(4):870–885
42. Xu G, Li H, Liu S, Wen M, Lu R (2019) Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. IEEE Trans Veh Technol 68(4):3854–3865
43. Xu G, Li H, Liu S, Yang K, Lin X (2020) Verifynet: Secure and verifiable federated learning. IEEE Trans Inf Forensics Secur 15(1):911–926
44. Xu G, Li H, Ren H, Yang K, Deng RH (2019) Data security issues in deep learning: Attacks, countermeasures and opportunities. IEEE Commun Mag 57(11):116–122. https://doi.org/10.1109/MCOM.001.1900091
45. Xu M, Guo M, Shang L, Jia X (2016) Multi-value image segmentation based on fcm algorithm and graph cut theory. In: 2016 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1333–1340. IEEE
46. Xue Q, Zhu Y, Wang J (2019) Joint distribution estimation and naïve bayes classification under local differential privacy. IEEE Transactions on Emerging Topics in Computing. https://doi.org/10.1109/TETC.2019.2959581
47. Yu H, Vaidya J, Jiang X (2006) Privacy-preserving svm classification on vertically partitioned data. In: Pacific-asia conference on knowledge discovery and data mining, pp 647–656. Springer
48. Yu TK, Lee D, Chang SM, Zhan J (2010) Multi-party k-means clustering with privacy consideration. In: International symposium on parallel and distributed processing with applications, pp 200–207. IEEE
49. Zhu Y, Li X, Wang J, Liu Y, Qu Z (2017) Practical secure naïve bayesian classification over encrypted big data in cloud. Int J Found Comput Sci 28(06):683–703
50. Zhu Y, Zhang Y, Li X, Yan H, Li J (2018) Improved collusion-resisting secure nearest neighbor query over encrypted data in cloud. Concurrency and Computation Practice and Experience. https://doi.org/10.1002/cpe.4681




**Youwen Zhu** received his B.E. degree and Ph.D. degree in Computer Science from University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively. From 2012 to 2014, he is a JSPS postdoc in Kyushu University, Japan. He is currently an Associate Professor at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He has published more than 40 papers in refereed international conferences and journals, and has served as program committee member in several international conferences. His research interests include identity authentication, information security and data privacy.



**Xingxin Li** received the M.S. degree in Nanjing University of Aeronautics and Astronautics, China in 2017. He is currently pursuing the Ph.D. degree at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include cloud computing and data privacy.