Received: 18 August 2019 | Revised: 3 November 2019 | Accepted: 24 November 2019
DOI: 10.1002/cpe.5641



WILEY

RESEARCH ARTICLE

# A privacy-preserving density peak clustering algorithm in cloud computing

Liping Sun[1,2] | Shang Ci[1,2] | Xiaoqing Liu[1,2] | Xiaoyao Zheng[1,2] | Qingying Yu[1,2] | Yonglong Luo[1,2]

[1]School of Computer and Information, Anhui Normal University, Wuhu, China
[2]Anhui Provincial Key Laboratory of Network and Information Security, Anhui Normal University, Wuhu, China

**Correspondence**
Yonglong Luo, School of Computer and Information, Anhui Normal University, Wuhu 241000, China.
Email: ylluo@ustc.edu.cn

**Funding information**
National Natural Science Foundation of China, Grant/Award Number: 61602009, 61972439, and 61702010; Anhui Provincial Natural Science Foundation of China, Grant/Award Number: 1808085MF172

## Summary

Aiming at preventing the privacy disclosure of sensitive information, issues related to privacy protection in cloud computing have attracted the interest of researchers. To protect the privacy of users during clustering in a cloud computing environment, we present a privacy-preserving density peak clustering (PPDPC) algorithm that neither discloses personal privacy information nor leaks the cluster centers. Our scheme contains two steps of density peak clustering: First, a cloud service provider calculates the cluster centers without knowing each participant's private data and without disclosing any cluster center information to the other participants, and second, participant allocation is secure and every participant is prevented from identifying the other members of the same cluster. Security analysis and comparison experiments show that the proposed PPDPC algorithm not only obtains good accuracy with respect to density peak clustering but also resists collusion attacks even if the cloud service provider is collaborating with all except one participant. Both theoretical analysis and experimental results confirm the security and accuracy of our method.

**KEYWORDS**

cloud computing, data mining, density peak clustering, homomorphic encryption, privacy preservation

## 1 | INTRODUCTION

With the rapid development of mobile social networks and computer technology, all types of mobile terminals and servers have started generating huge amounts of data at all times, presenting a serious challenge to the computing ability of enterprises.[1,2] Cloud computing technology, which is used to address this challenge, is growing. More and more enterprises are storing data in cloud servers to save economic costs; its powerful computing power is convenient for handling huge amounts of data.[3-5] Additionally, data mining technology can help users analyze and extract key value information from a large amount of data in scientific research and business applications. The analysis of these data allows the prediction of future development trends and directions.[6] Clustering, as one of the important research methods of data mining, aims to divide data objects into several clusters such that object similarity in a cluster is high, while the similarity between each cluster is low.

In the process of clustering analysis, a large amount of user-based privacy data, such as geographical location, electricity consumption data, and spatiotemporal sensing data, is collected and analyzed.[7-9] The security and privacy of this data depends on the security of cloud services. The sensitive data is directly outsourced to the cloud server for calculation, at which point the user's privacy may be leaked if the cloud service provider is malicious or dishonest. If multiple users collude with each other, they combine their own information to calculate their respective distances and then calculate the cluster centers by distance. If user's privacy and cluster centers are disclosed, serious consequences can result. Therefore, the development of a data mining technology for preserving the mutual privacy of users and the cloud server is important.[10-12] Such a technology must allow a cloud server to extract information about a social network without accessing the privacy data of any user, while also preventing all users from obtaining any information about other users or social networks. The existing privacy-preserving technology focuses

  

on protecting the privacy of a single party, such as participating users; this technology may leak intermediate results from a social networking cluster to potential privacy attackers or be unable to resist collusion attacks. For example, a collusion participant can identify the cluster centers at each iteration of the *k*-means cluster.[13] The exposure of these intermediate data and ultimately the characteristic data of a social network can put personal privacy at risk, leading to panic and extreme behavior among social participants.

Cloud computing is a computing model that uses cloud computing clustering capabilities to decompose complex computing requests into smaller subrequests that are processed by multiple infrastructures and then output after integration. Cloud computing is also a business model that uses cloud technology to deliver cloud service providers large-scale IT equipment to cloud service consumers over the network. This service is characterized by high reliability, easy expansion, low price, and on-demand service. It is generally believed that the characteristics of cloud computing itself are the main cause of cloud security. Additionally, cloud computing is built on the existing technology environment, which involves network security issues pertaining to virtual machine security, hardware security, and others; these issues then also appear in cloud computing security.

Density peak clustering (DPC) is a simple and efficient clustering algorithm.[14] This algorithm obtains the decision diagram by measuring the local density and distance of each sample point, selects the best clustering center points on the decision diagram according to the characteristics of the clustering center points, and assigns the remaining sample points to the sample cluster closest to them with high density. The algorithm is simple and efficient, eliminating the need for an iterative process. By using the DPC algorithm, the distance information, which requires the participant's privacy data, is first calculated, thus increasing the risk of disclosing privacy.

In cloud computing security, an effective method to protect user privacy is to encrypt data before outsourcing it. The homomorphic encryption technology obtains accurate calculation result and better security through strict cryptographic principles. However, the computational cost of homomorphic encryption is high. Therefore, for calculation, this paper uses homomorphic encryption combined with cloud computing platforms.

To solve these problems, this paper presents a privacy-preserving density peak clustering (PPDPC) algorithm. The proposed PPDPC algorithm obtains sufficient accuracy with respect to DPC algorithm. Our method can resist collusion attacks, although the cloud server is collaborating with all except one participant. Both privacy analysis and experimental results confirm the security and accuracy of our algorithm. The main contributions of this study to relevant literature are summarized as follows.

1. In the process of calculating the clustering centers through the DPC algorithm, we first calculated the distance between the participants in the social network, and proposed a privacy-preserving clustering algorithm. Security and privacy-preserving analyses show that the proposed PPDPC method can resist collusion attacks.
2. The proposed algorithm not only can protect the privacy of personal information but also can prevent any clustering center information from leaking to the participants; this preserves privacy in the cluster and limits each participant in their ability to identify other members of the cluster.
3. The basic idea of the DPC, which here is proposed to improve secrecy and mitigate the collusion attack, can be extended to other data mining algorithms for privacy preservation.
4. We validated the proposed method through extensive experimental research. The effectiveness of the PPDPC in resisting collusion attacks is tested, even if the cloud service provider is collaborating with all except one participant.

The rest of the paper is organized as follows. Section 2 introduces the most related existing work on privacy-preserving clustering. Section 3 presents preliminary background information about DPC and introduces the assumptions adopted by this paper. Our privacy-preserving DPC algorithm is detailed in Section 4, and its corresponding privacy and cost analyses are presented in Section 5. Experimental studies are reported in Section 6. We conclude this paper with an outline of intended future research in Section 7.

## 2 | RELATED WORK

Conventional privacy-preserving methods are divided into three categories: *k*-anonymous technology,[15] perturbation technology,[16] and data encryption technology.[17,18] The *k*-anonymous technology includes generalization and suppression methods and can directly hide information to effectively protect user privacy data. However, *k*-anonymous technology does not provide high security and cannot effectively resist link attacks. Perturbation techniques include factors such as data cleansing, data exchange and randomization interference, and differential privacy techniques. By randomly scrambling or adding noise to the original data to protect user privacy, this interference to the data affects the accuracy of calculation results. The most typical data encryption techniques include secure multiparty and homomorphic encryption algorithms; for these, results obtained by rigorous cryptography are accurate and safe, but the computational overhead is high. Here, a homomorphic encryption algorithm was used, calculated with the help of a cloud computing platform.

According to the nature of homomorphic encryption, we summarized the work of addition and multiplication in the existing privacy-preserving clustering algorithm for homomorphic encryption. As a classical clustering algorithm, the additive homomorphic encryption scheme based on *k*-means research is gaining increasing attention. Unlike the DPC algorithm, *k*-means clustering does not involve the distance between two sample points; instead, only the sample points are assigned to the nearest cluster center. A safe allocation sample pointing to the nearest clustering

center algorithm was proposed by Xing.[19] Vaidya and Clifton[20] proposed a privacy-preserving *k*-means algorithm for vertically segmenting data. Du and Atallah[21] proposed a secure permutation algorithm to calculate the distance between participants based on the homomorphic encryption scheme; however, the agreement required three incompatible participants, a hypothesis not easily guaranteed in many practical applications. Jha et al[22] presented a privacy-preserving *k*-means clustering algorithm based on oblivious polynomial evaluation and homomorphic encryption. Although this approach can be applied in multiparty environments, its clustering centers are often exposed to potential privacy attacks. Bunn and Ostrovsky[23] proposed a mutual *k*-means clustering protocol based on homomorphic encryption to guarantee the privacy of any partition data, in which the protocol does not expose the intermediate results of a cluster and sample allocation. Additionally, a secure protocol was designed to randomly select *k* initial cluster centers. However, if the protocol extends to multiparty *k*-means clustering, it can introduce new security and privacy risks. For example, when more than half of the participants engage in collusion, the agreement is not resistant to conspiracy attacks. To address this, Rao et al[24] proposed a two-party *k*-means clustering protocol that can be achieved through the semantic security of the homomorphic encryption scheme. Liu et al[25] proposed an outsourcing *k*-means clustering scheme using a cryptographic algorithm to generate trap information; however, this scheme only protects the privacy of one participant.

Liu et al[25] elaborated upon the sum of the data of sample points in the cluster by using the additive homomorphic encryption scheme in the *k*-means algorithm to find the average of each cluster and update the clustering center. In the current study, as the method of multiplication was used to calculate the distance by using the DPC algorithm, the multiplicative homomorphic encryption scheme was considered. This scheme ensures that only the cloud service provider knows the distance information by computing the cluster centers based on the DPC algorithm; in this manner, the participants cannot obtain the intermediate information. A previous study had proposed the multiplicative homomorphic encryption algorithm[26]; multiplicative homomorphism was shown to assist the Rivest-Shamier-Adleman (RSA) attack to some extent (such as the selection of plaintext attacks). Another study[27] presented a new method for protecting the privacy of the *k*-means clustering mining algorithm based on RSA multiplicative homomorphic encryption scheme; this method uses the RSA public key cryptosystem and homomorphic encryption system to protect the security of each participant's data. Each participant first computes the clustering process through the *k*-means clustering algorithm in the local center and then encrypts the results. The local clustering results are then received from the central site, and the remaining data mining work in the cloud is completed.

Recently, methods of privacy preservation have been increasingly applied to the DPC algorithm. Guo and Meng[28] proposed a DPC-based differential privacy-preserving model that can protect privacy by adding Laplace noise to local density $\rho$ and distance $\delta$. However, this scheme only protects the cluster centers and adversely data validity. Zhang et al[29] proposed a tensor distance to calculate the multimedia data set sample point distance through homomorphic encryption to upload data to the cloud for DPC. However, the algorithm cannot be applied to a multiparty environment. Here, we propose the use of the homomorphic encryption algorithm in multiple environments and a PPDPC algorithm to resist collusion attacks.

## 3 | PRELIMINARIES AND ASSUMPTIONS

The symbols used in this paper and their semantic meanings are listed in Table 1.

### 3.1 | DPC algorithm

The DPC algorithm can achieve efficient clustering of arbitrary-shape data sets. The algorithm process is based on (1) and (2). Suppose there are $n$ participants and each participant $a_i$ has a $q$-dimensional sample data $a_i^{(\tau)}$. The core idea of the DPC algorithm is to characterize cluster centers, which should simultaneously have the following two characteristics:

1. Large density: The density is not more than that of its surrounding sample points.
2. Large distance: The distance from other data points with greater density is relatively larger.

**TABLE 1** Notations and their semantic meanings

| Notations | Means |
|---|---|
| $\rho$ | Local density |
| $\delta$ | Distance of each object |
| $d_{ij}$ | Euclidean distance between samples *i* and *j* |
| $d_c$ | Cutoff distance |
| $E(\cdot)$ | Encryption operation |
| $D(\cdot)$ | Decryption operation |
| $PK$ | Public key |
| $SK$ | Private key |

Here, we used Euclidean distance as the criterion for representing distance of sample points $a_i$ and $a_j$, ie, $d_{ij} = \text{dist}(a_i^{(\tau)}, a_j^{(\tau)})$. For any participant $a_i$, we can define quantities $\rho_i$ and $\delta_i$ (these two amounts correspond to the two characteristics of the cluster centers).

$$\rho_i = \sum_{j \neq i} \chi(d_{ij} - d_c), \tag{1}$$

where $1 \leq i \leq n, 1 \leq j \leq n$ and when $x < 0$, $\chi(x) = 1$ or $\chi(\text{x}) = 0$ otherwise.

$$\delta_i = \min_{\substack{j \\ \rho_j > \rho_i}} (d_{ij}) \tag{2}$$

For the largest sample $a_i$ of local density, $\rho_i, \delta_i = \max_j d_{ij}$.

According to (1), the cutoff distance $d_c$ is affected by the sample local density introduced by Rodriguez and Laio,[14] who showed that when the data set size (ie, the number of samples) is small, the cutoff distance positively affects the performance of the DPC algorithm, whereas the performance is negatively affected if the data set size is large. To avoid the effect of cutoff distances on sample local density and clustering performance, the DPC algorithm calculates the sample local density for smaller data sets by using Gaussian cores as

$$\rho_i = \sum_{j \neq i} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right). \tag{3}$$

The DPC algorithm has relatively large $\rho_i$ and $\delta_i$ as the simultaneous cluster centers. This method uses a qualitative analysis rather than a quantitative analysis to determine the clustering center, including subjective factors. A decision-making graph consisting of the decision value $\gamma$(calculated using (4)) was presented by Rodriguez and Laio[14] to determine the cluster centers (density peak points).

$$\gamma_i = \rho_i \cdot \delta_i \tag{4}$$

The remaining sample $a_j$ is assigned by the DPC algorithm to a cluster with larger density than that of $a_j$ and the nearest samples; then, $a_j$ is allocated in one step without iteration.

To avoid the effect of outliers on cluster results, the DPC algorithm defines the boundary region, as shown in the cluster halo, according to (5). The boundary area of a cluster refers to the sample that belongs to the cluster (number of samples, $n_b$); however, the distance from the other cluster samples is less than the cutoff distance $d_c$, and the outliers are distributed in the boundary region. The density of a sample larger than the average density of the cluster's boundary region is defined as the threshold $\rho_b$ in (6). Additionally, the threshold $\rho_{b_i}$ is defined for each cluster, and the core area of the cluster, as shown in (7) is a sample of the density larger than $\rho_{b_i}$, while the other samples of the cluster are outlier points.

$$\text{clusterhalo} = \left\{a_i | d_{ij} < d_c, 1 \leq i \leq n_b, 1 \leq j \leq n\right\} \tag{5}$$

$$\rho_b = \left\{ \rho_{b_i} | \rho_{b_i} > \frac{1}{2}(\rho_i + \rho_j), 1 \leq i \leq n_b, 1 \leq j \leq n_b \right\} \tag{6}$$

$$\text{clustercore} = \{a_i | \rho_i > \rho_b, 1 \leq i \leq n_b\} \tag{7}$$

## 3.2 | Homomorphic encryption

Homomorphic encryption allows certain calculations of encrypted data; this is equivalent to ciphertext operations after the encryption of a plaintext operation. Rivest et al[26] used number theory to design a well-known RSA encryption scheme. The RSA cryptosystem is a well-known multiplication homomorphic encryption scheme, which provides fast encryption and decryption. The security of the RSA scheme depends on the difficulty of large integer decomposition.[30,31] The longer the RSA key, the more secure it is; it is presently widely used in banks and other fields. The RSA cryptography system is briefly described as follows.

1. *Key generation*. An entity selects two large primes, $p$ and $q$, and computes $n = pq$. According to Euler's theorem, $L(n) = (p - 1)(q - 1)$. The entity then randomly chooses integers $\lambda$ and $e$ such that $\gcd(\lambda, L(n)) = 1$ and $e\lambda \equiv 1(L(n))$, where *gcd* indicates the greatest common divisor. The public and private keys are then $\{n, e\}$ and $\{\lambda\}$, respectively.

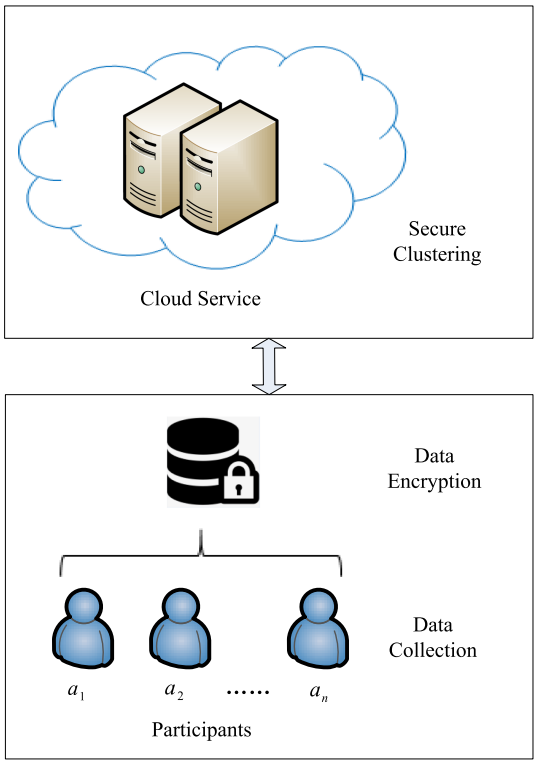


**FIGURE 1** Clustering model

2. *Encryption*. Let $m \in \mathbb{Z}_N^*$ be a plaintext. Then, the ciphertext of $m$ is computed as

$$E(m) = m^e \bmod n, \tag{8}$$

where $E(\cdot)$ denotes the encryption operation using public key $\{n, e\}$.

3. *Decryption*. For ciphertext $E(m)$, the corresponding plaintext can be computed as

$$D(E(m)) = E(m)^{\lambda} \bmod n \tag{9}$$

where $D(\cdot)$ denotes the decryption operation using the private key $\{\lambda\}$.

4. *Homomorphic*. The RSA cryptography system is a multiplicative homomorphism because it satisfies the following conditions given $\{m_1, m_2\} \in \mathbb{Z}_N^*$

$$E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2). \tag{10}$$

## 3.3 | Clustering model

We consider a clustering problem consisting of $n$ participants $a_1, \ldots, a_n$. Each participant has its own privacy information $a_i^{(\tau)}$, which is a $q$-dimensional vector describing its characteristics or attributes. Additionally, as shown in Figure 1, there also exists a cloud-service provider $A$, who is responsible for grouping the participants with similar attributes into a cluster. However, owing to privacy issues, the cloud service provider cannot access the privacy information of any participant.

When calculating the distance between the private data of participants $a_1, \ldots, a_n$, our goal is to establish privacy protection between cloud service $A$ and participants $a_i$. As a result of the DPC clustering method, neither $A$ nor the participants can deduce any private information from the other party, except the published information. Cloud service provider $A$ cannot learn any private information owned by the participants because a compromised $A$ may betray the data to others. Similarly, malicious participants are unable to obtain the data of cloud service $A$ (distance and clustering centers) and should not be able to disrupt the calculations of $A$.

We assume that

1. any pair of participants $a_i$ and $a_j$ share a unique pairwise key;

2. cloud service $A$ shares a unique pairwise key with each participant;
3. cloud service $A$ generates a public and private key pair $(PK, SK)$, which can be used for multiplicative homomorphic encryption, and distributes its public key $PK$ to all the participants.

Messages from each participant $a_i^{(\tau)}$ to $A$ should be encrypted with $PK$. Let $m$ denote the plaintext and $m'$ denote the corresponding ciphertext. We have $m' = E(PK, m)$ and $m = D(SK, m')$. Owing to the homomorphic property,

$$E(PK, m_1 \cdot m_2) = E(PK, m_1) \cdot E(PK, m_2). \tag{11}$$

### 3.4 | Security model

To maximize the protection of the privacy of all parties, the proposed scheme utilizes a semihonest model in secure multiparty computing. By this model, the parties are semihonest, ie, each party follows the rules of the security protocol but records intermediate results to infer private information. Here, the parties refer to cloud service providers and participants.

## 4 | PPDPC ALGORITHM

In this section, we present our PPDPC algorithm, which comprises two steps: (1) the safe calculation of the cluster centers and (2) the assignment of the remaining points to the cluster comprising the cluster centers.

In the first step, participants must encrypt their private data and send it to cloud service $A$ for the calculation of the distance between the participants. Then, the cluster centers are safely computed based on the distance. In the second step, $A$ first eliminates outliers corresponding to the participants so as not to affect the clustering results. The remaining participants are then assigned to their nearest cluster according to the DPC algorithm; then, the participants of outliers are allocated.

### 4.1 | Stage 1: Computation of clustering centers

First, the distance between the current participant and other participants is calculated, and $D_{ij}$ represents the distance between the $i$th and $j$th participants in the data set.

$$D_{ij} = (a_i - a_j)^T (a_i - a_j) = {a_i}^T a_i - 2{a_i}^T a_j + a_j^T a_j \tag{12}$$

The privacy data of participants and are all $q$-dimensional vectors; their distinction expressions are given as $a_i = \{a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(q)}\}$ and $a_j = \left\{a_j^{(1)}, a_j^{(2)}, \ldots, a_j^{(q)}\right\}$, respectively. Now, considering the value of $a_i^T a_j$, we have

$$a_i^T a_j = a_i^{(1)} \cdot a_j^{(1)} + a_i^{(2)} \cdot a_j^{(2)} + \ldots + a_i^{(q)} \cdot a_j^{(q)}. \tag{13}$$

Equation (12) calculates the distance between each participant and other participants, as specifically shown in (14).

$$\begin{cases} a_1^T a_1 - 2\left(a_1^{(1)} \cdot a_2^{(1)} + a_1^{(2)} \cdot a_2^{(2)} + \ldots + a_1^{(q)} \cdot a_2^{(q)}\right) + a_2^T a_2 = D_{12} \\ a_1^T - 2\left(a_1^{(1)} \cdot a_3^{(1)} + a_1^{(2)} \cdot a_3^{(2)} + \ldots + a_1^{(q)} \cdot a_3^{(q)}\right) + a_3^T a_3 = D_{13} \\ \ldots\ldots \\ a_1^T a_1 - 2\left(a_1^{(1)} \cdot a_n^{(1)} + a_1^{(2)} \cdot a_n^{(2)} + \ldots + a_1^{(q)} \cdot a_n^{(q)}\right) + a_n^T a_n = D_{1n} \\ a_2^T a_2 - 2\left(a_2^{(1)} \cdot a_3^{(1)} + a_2^{(2)} \cdot a_3^{(2)} + \ldots + a_2^{(q)} \cdot a_3^{(q)}\right) + a_3^T a_3 = D_{23} \\ a_2^T a_2 - 2\left(a_2^{(1)} \cdot a_4^{(1)} + a_2^{(2)} \cdot a_4^{(2)} + \ldots + a_2^{(q)} \cdot a_4^{(q)}\right) + a_4^T a_4 = D_{24} \\ \ldots\ldots \\ a_2^T a_2 - 2\left(a_2^{(1)} \cdot a_n^{(1)} + a_2^{(2)} \cdot a_n^{(2)} + \ldots + a_2^{(q)} \cdot a_n^{(q)}\right) + a_n^T a_n = D_{2n} \\ \ldots\ldots \\ \ldots\ldots \\ a_{n-1}^T a_{n-1} - 2\left(a_{n-1}^{(1)} \cdot a_n^{(1)} + a_{n-1}^{(2)} \cdot a_n^{(2)} + \ldots + a_{n-1}^{(q)} \cdot a_n^{(q)}\right) + a_n^T a_n = D_{(n-1)n} \end{cases}. \tag{14}$$

We present a safe computing distance for the method of multiplicative homomorphic encryption. The distance is calculated by combining three values, as shown in (12). For $a_i^T a_i$ and $a_j^T a_j$, the results can be calculated by the participants themselves and sent to cloud service $A$. For the $a_i^T a_j$ value, participants $a_i$ and $a_j$ are required to send their respective private data to $A$, which then calculates the three values and finally obtains the distance results.

Next, a privacy-preserving program was implemented by sending individual privacy data to $A$ in $a_i^{(\tau)}$ and $a_j^{(\tau)}$. We encrypted a participants $q$-dimensional privacy data separately. For example, for $a_i^{(1)} \in a_i$ and $a_j^{(1)} \in a_j, A$ generates the corresponding random numbers $R_i^{(1)}$ and $R_j^{(1)}$ that satisfy

$$R_i^{(1)} \cdot R_j^{(1)} = 1 \tag{15}$$

and sends them to $a_i^{(1)}$ and $a_j^{(1)}$, respectively.

Next, the participants encrypt $a_i^{(1)}$ and $a_j^{(1)}$, for which the encryption is calculated using the public key $PK$ of the homomorphic encryption system. For $a_i^{(\tau)} \in a_i, a_i$ encrypts $a_i^{(\tau)}$ after receiving the random number $R_i^{(\tau)}$. For example, $a_i^{(1)}$ encryption is obtained after

$$Y_i^{(1)} = E\left(PK, a_i^{(1)} \cdot R_i^{(1)}\right). \tag{16}$$

For $a_j^{(\tau)} \in a_j, a_j$ encrypts dimension data $a_j^{(\tau)}$ after it receives the random number $R_j^{(\tau)}$. For example, $a_j^{(1)}$ encryption is obtained after

$$Y_j^{(1)} = E\left(PK, a_j^{(1)} \cdot R_j^{(1)}\right). \tag{17}$$

When the encryption operation is complete, the participant shares a portion of the ciphertext with the other person as follows. First, participants randomly divide the encrypted privacy data (computed using (16) and (17)) into $p$ components $Y_{i1}^{(1)}, Y_{i2}^{(1)}, \ldots, Y_{ip}^{(1)}$ and $Y_{j1}^{(1)}, Y_{j2}^{(1)}, \ldots, Y_{jp}^{(1)}$, satisfying

$$Y_{i1}^{(1)} \cdot Y_{i2}^{(1)} \cdot \cdots \cdot Y_{ip}^{(1)} = Y_i^{(1)}$$
$$\text{and} \quad Y_{j1}^{(1)} \cdot Y_{j2}^{(1)} \cdot \cdots \cdot Y_{jp}^{(1)} = Y_j^{(1)}.$$

The participants then send $q$ components to each other randomly via the secure channel, where $0 < q < p$. Note that privacy data $a_i^{(1)}$ and $a_j^{(1)}$ must retain $p - q$ components.

Furthermore, each dimension of a participant's privacy data should be completed according to ciphertexts in (16) and (17) to complete the above-mentioned partitioning and distribution process. Simultaneously, each dimension of the current participant's privacy data can receive ciphertext fragments from the privacy data of the corresponding dimensions of other participants. The participant then multiplies all the received ciphertext components and fragments they retain by using the homomorphism operation to obtain $r$. For example, $a_i^{(1)}$ divides $Y_i^{(1)}$ into three parts: $Y_i 1^{(1)}, Y_i 2^{(1)}$, and $Y_i 3^{(1)}$; and $a_j^{(1)}$ divides $Y_j^{(1)}$ into two parts: $Y_{j1}^{(1)}$ and $Y_j 2^{(1)}$. Then, $a_i^{(1)}$ sends $Y_i 2^{(1)}$ and $Y_{i3}^{(1)}$ to $a_j^{(1)}$, whereas $a_j^{(1)}$ sends $Y_{j2}^{(1)}$ to $a_i^{(1)}$; then,

$$r_i^{(1)} = Y_{i1}^{(1)} \cdot Y_{j2}^{(1)}$$
$$r_j^{(1)} = Y_{j1}^{(1)} \cdot Y_{i2}^{(1)} \cdot Y_{i3}^{(1)}.$$

After the calculation is completed, $r$ should be sent to the cloud service provider $A$.

$A$ multiplies all received data and obtains the following results according to (11) and (15):

$$\begin{aligned} Y_{i1}^{(1)} \cdot Y_{j2}^{(1)} \cdot Y_{j1}^{(1)} \cdot Y_{i2}^{(1)} \cdot Y_{i3}^{(1)} &= Y_i^{(1)} \cdot Y_j^{(1)} \\ &= E\left(PK, a_i^{(1)} \cdot R_i^{(1)}\right) \cdot E\left(PK, a_j^{(1)} \cdot R_j^{(1)}\right) \\ &= E\left(PK, a_i^{(1)} \cdot R_i^{(1)} \cdot a_j^{(1)} \cdot R_j^{(1)}\right) \\ &= E\left(PK, a_i^{(1)} \cdot a_j^{(1)}\right). \end{aligned} \tag{18}$$

Then, $A$ obtains $a_i^{(1)} \cdot a_j^{(1)}$ through decryption by using its private key $SK$ and adds all the values of the dimensions of private data $a_i^{(\tau)} \cdot a_j^{(\tau)}$ according to (13). Participants $a_i$ and $a_j$ calculate values for $a_i^T a_i$ and $a_j^T a_j$, respectively, and send the values to $A$, which then calculates the distance $D_{ij}$ between each participant according to (12).

**Algorithm 1** Computing cluster centers

**Input:** There are $n$ participants and cloud service $A$. Each participant $a_i$ has $q$-dimensional privacy data $a_i = \left\{ a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(q)} \right\}$.
**Output:** cluster centers $U$.

**for** $i \leftarrow 1$ to $n$-1
  **for** $j \leftarrow i + 1$ to $n$
    **for** $\tau \leftarrow 1$ to $q$
    Cloud service $A$ generates random number $R$ according to (15) and sends it to $a_i^{(\tau)} : R_i^{(\tau)}$ and $a_j^{(\tau)} : R_j^{(\tau)}$ ;
    $a_i$ encrypts current privacy data $a_i^{(\tau)}$ according to (16), $a_j$ encrypts current privacy data $a_j^{(\tau)}$ according to (17).
    $a_i$ and $a_j$ divide the ciphertext into $p$ components, randomly send $q$ components to each other and each retains $p$-$q$ components;
    $a_i$ and $a_j$ multiply their own ciphertexts and receive part of the ciphertexts to obtain $r$;
    $a_i$ and $a_j$ send $r$ to cloud service $A$;
    $A$ multiplies all received data and obtains $a_i^{(\tau)} \cdot a_j^{(\tau)}$ through decryption;
    Cloud service $A$ adds all $a_i^{(\tau)} \cdot a_j^{(\tau)}$ values according to (13);
    **end for**
  $a_i$ and $a_j$ calculate the values of $a_i^T a_i$ and $a_j{}^T a_j$, respectively, and send them to $A$;
  $A$ calculates distance $D_{ij}$ between each participant according to (12);
  **end for**
**end for**
$A$ calculates local density $\rho$ according to (3);
$A$ calculates distance $\delta$ of the local density and the nearest participant according to (2);
$A$ calculates decision value $\gamma = \{\gamma_1, \ldots, \gamma_k\}$ according to (4) ($k$ is the number of decision values);
$A$ to $\gamma$ in descending to select cluster centers $U$.

Finally, the cluster centers are computed by $A$ according to (2), (3), and (4). The entire process is summarized in Algorithm 1.

Figure 2 illustrates the detailed process of computing the cluster centers via the PPDPC algorithm. According to the above process, the cloud service provider can calculate the cluster centers based on distance. The advantage of this algorithm is that it ensures privacy preservation.

1. Cloud providers can calculate distances without accessing the privacy information of each participant.
2. The participants do not know each other's privacy information. Particularly, they do not know who exists in the same cluster.
3. The participants do not know the information of the cluster centers. This information is calculated and protected by the cloud service.

## 4.2 | Stage 2: Assign participants to their nearest centers

After calculating the cluster centers $U$ ($U_1, U_2, \ldots, U_k$ are sorted by $\gamma$ values in descending order) via Algorithm 1, cloud service $A$ assigns the remaining participants to the nearest cluster. After Stage 1, $A$ directs the participants to $k$ cluster centers, with the label of the corresponding participants, as shown in Table 2, where $s_i \in \{a_1, \ldots, a_n\}$ indicates $s_i$ as the number corresponding to the $i$th cluster center.

As the privacy data of some participants may be different from those of other participants, they can become outliers throughout the data set, thus affecting the clustering effect. Therefore, we decided to discard these outliers (according to (5) to (7) to find the outlier sample) and number the resulting sample as $o_i \in \{a_1, \ldots, a_n\}$.

If $a_1, \ldots, a_n$ does not comprise numbered participant $a_j$, $A$ classifies it into a cluster of participants with a density greater than $a_j$ and a distance nearest to $a_j$. The $\rho$ values calculated in Stage 1 were then sorted in ascending order. If numbered participant $s_i$ (cluster center) corresponds to the $\rho$ value, no further allocation is performed. The remaining participants are in the order of $\rho$ from large to small, and $A$ is assigned to the cluster in which the participant is closest according to the distance calculated using (12). Next, we merged outlier $o_i$ into the cluster in which the nearest assigned participant is located. The entire process is summarized in Algorithm 2.

The labels of the postallocation participants are listed in Table 3, where $a_{(j,l)} \in \{a_1, \ldots, a_n\}$ represents the $l$th participant in the $j$th cluster, $o_{(j,z)} \in \{a_1, \ldots, a_n\}$ represents the participant of the $z$th outliers in the $j$th cluster, where $1 \le j \le k$, $1 \le l \le n_k$, and $1 \le z \le m_k$.

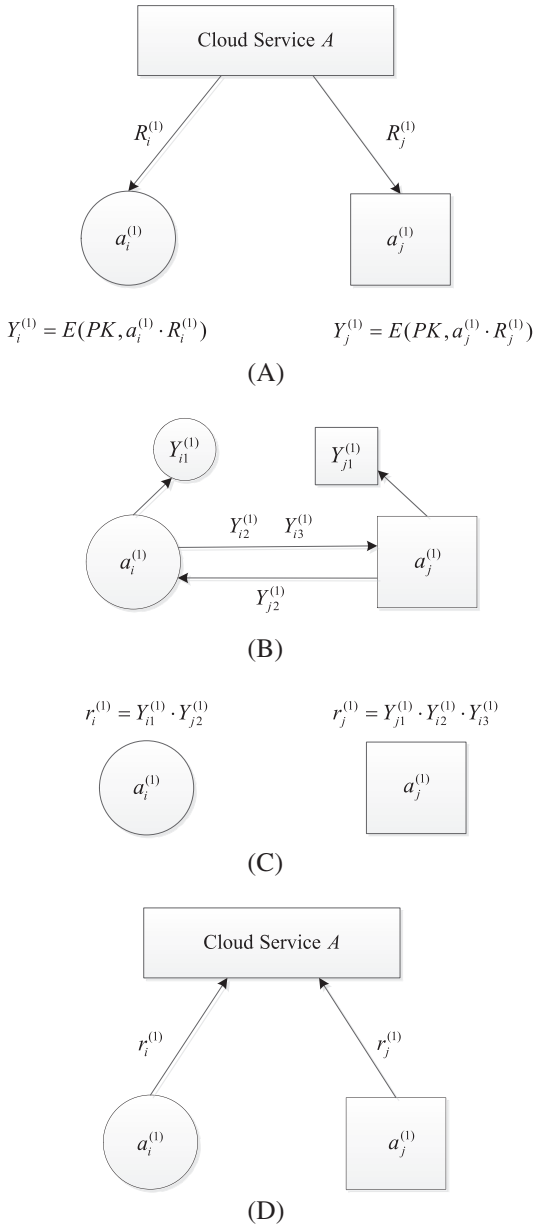


**FIGURE 2** Detailed process of computing cluster centers in the PPDPC algorithm. A, The cloud service provider sends a random number to two participants, satisfying $R_i^{(1)} \cdot R_j^{(1)} = 1$; B, Each participant slices its encrypted data. That is, the encrypted data is randomly divided into $p$ components, $q$ of them are sent to other participants and the $p - q$ components are reserved; C, Each participant multiplies the data received and the data it retains. D, All participants send the results to the cloud service provider that can compute the cluster centers

**TABLE 2** Cluster center labels

| Participants | |
|---|---|
| **Cluster** | **Participants Label** |
| $U_1$ | $s_1$ |
| $U_2$ | $s_2$ |
| … | … |
| $U_k$ | $s_k$ |

**Algorithm 2** Assigning participants to their nearest centers

**Input:** $k$ cluster center $U$; cluster $U_i$ corresponding to participant number $s_i$.

**Output:** Label of participants after allocation.

1: Cloud service $A$ excludes outliers according to (5)-(7), and numbers the result as $o_i \in \{a_1, \ldots, a_n\}$;

2: $A$ sorts the $\rho$ values in Stage 1 from largest to smallest;

3: $A$ assigns unnumbered participants $a_i$ in the order of $\rho$ values from large to small, and assigns them according to the distance calculated by (12) to the cluster in which the participant is the smallest;

4: $A$ assigns the participant corresponding to the outlier to the cluster in which the nearest assigned participant is located;

5: $A$ informs each cluster of where the participants are located.

**TABLE 3** Participant labels after assignment

| Participant cluster | Participant label after assignment |
|---|---|
| $U_1(1+n_1+m_1\ participants)$ | $s_1, a_{(1,1)}, \ldots, a_{(1,n_1)}, o_{(1,1)}, \ldots, o_{(1,m_1)}$ |
| $U_2(1+n_2+m_2\ participants)$ | $s_2, a_{(2,1)}, \ldots, a_{(2,n_2)}, o_{(2,1)}, \ldots, o_{(2,m_2)}$ |
| … | … |
| $U_k(1+n_k+m_k participants)$ | $s_k, a_{(k,1)}, \ldots, a_{(k,n_k)}, o_{(k,1)}, \ldots, o_{(k,m_k)}$ |

## 5 | PRIVACY AND EFFICIENCY ANALYSIS

In this section, we discuss the use of our proposed PPDPC algorithm to prevent potential passive and active attacks in privacy preservation. Before delving into the details, we first define the privacy objectives of this study; participants should be unable to obtain the following information:

1. cluster centers;
2. other participant's in the same cluster;
3. the private data of the other participants.

Cloud service $A$ knows the location of the cluster centers but cannot access the private information of any participant. In other words, $A$ knows that participant $a_i$ belongs to the $j$th cluster but does not know the private data $a_i^{(\tau)}$ of $a_i$.

### 5.1 | Privacy analysis for computing clustering centers

In Stage 1 (see Section 4.1), we used the public key–based multiplicative homomorphic encryption scheme to compute the clustering center. Encryption data $Y_i^{(1)} = E(PK, a_i^{(1)} \cdot R_i^{(1)})$ and $Y_j^{(1)} = E(PK, a_j^{(1)} \cdot R_j^{(1)})$ owned by participants $a_i$ and $a_j$, respectively, are semantically safe because only the cloud service knows private key $SK$. As $Y_i^{(1)}$ and $Y_j^{(1)}$ are sliced, shared, and eventually sent to $A$, the process is safe for participants and $A$. Data $Y_i^{(1)}$ is mixed with the random number $R_i$ known only by $A$ and protected by the public key cryptography system, and $Y_i^{(1)}$ is safe for participants $a_j$. As the data obtained by $A$ is encrypted by the participants, $A$ cannot obtain the participant's privacy data. Additionally, the participants are unable to obtain information about the cluster centers as they are not able to know the distance information and $\gamma$ value after sending their calculated information to $A$. This information is protected by the private key of $A$.

In Algorithm 1, cloud service $A$ sends a random number to each participant. Participant $a_i$ then slices encrypted data $Y_i^{(1)}$ and randomly sends the $q$ component to participant $a_j$, while participant $a_i$ retains the remaining $p - q$ parts. As a result, both parties do not know the other's privacy data. If an attacker wants to obtain the private data held by $a_i$, then all of the sent components $q$ and the other received components must be cracked. As $0 < q < p$, the participant receives the largest slice component $p - 1$ and the largest transmission component $q - 1$. The participant's privacy data can only be disclosed if the attacker has the ability to destroy all $2p - 2$ slice components. Let $m$ denote the probability of a single slice component being disclosed to an attacker. Then, the probability that

$$P_i = m^{p-1}.$$

When $p$ is larger and the probability increases exponentially, the smaller the probability $P_i$ and the less likely it is for the participant's privacy data to be disclosed.

*Cost Analysis*. In Stage 1, the cloud service performs the primary computing tasks; the participants only need to perform very few calculations. Consider the computational complexity of Algorithm 1, which is calculated $\frac{n(n-1)}{2}$ times when the distance between each participant is calculated. Further, each participant has a $q$-dimensional privacy data. In the process of encryption and decryption (Steps 5 and 9), the cloud service must perform the corresponding steps in Algorithm 1 $q$ times. Therefore, the complexity of calculating all cluster centers is $O(\frac{n(n-1)}{2} \cdot q)$.

### 5.2 | Privacy analysis for assigning participants to their nearest centers

Cloud service $A$ processes outliers in Stage 2 and allocates participants to the nearest cluster based on the cluster centers and processes $\rho$ obtained from Stage 1, where $A$ does not need to know the participant's privacy data. Once the participants have been assigned, the number of participants in each cluster can be obtained along with the clusters to which each participant belongs. This process does not disclose the privacy information of participants and cloud service $A$ related cluster centers, and participants are not aware of other participant's in the same cluster. Therefore, this process is safe.

### 5.3 | Security against collusion attacks

The proposed PPDPC algorithm can resist collusion attacks. To illustrate this, we assume that there exist dishonest cloud service A and participants who follow the following protocols.

***1. Collusion between the cloud service and participants***. Our method ensures that participants can hide their private data safely, even if the cloud service is an adversary. In Algorithm 1, cloud service $A$ receives $r_i^{(\tau)}$ and $r_j^{(\tau)}$ from $n$ participants. Although $A$ can decrypt data $r_i$ and $r_j$, no further information can be derived, as $r_i$ and $r_j$ are the results of the sliced components mixed with other random data.

Consider the scenario in which cloud service $A$ colludes with one or more participants. At maximum, there can be as many as $n-1$ participants colluding with $A$ to deduce the private information of the remaining participant $a_i$. We assume $a_i$ has sliced its encrypted data into $p$ splices; then, $q$ of the $p$ components (where $0 < q < p$) will be acquired by the other participants. Although $A$ knows the decryption key and random data $R_i^{(\tau)}$ held by $a_i$, $A$ cannot provide the private information of $a_i$ as $p-q$ of the $p$ slices is reserved by $a_i$. Thus, collusion alliance still cannot obtain private information of $a_i$; this illustrates that our algorithm is robust even in the worst case comprising as many as $n-1$ participants colluding with the cloud service. Therefore, we conclude that our protocol can protect the privacy of each participant against the collusion of the cloud service and its participants.

***2. Collusion among the participants***. Consider the worst case of $n-1$ participants colluding with each other to detect the cluster centers. Without loss of generality, we assume that these $n$-1 participants are denoted by $a_1, a_2, \ldots, a_{n-1}$; they combine their information to compute the distance, after which they can calculate the cluster center according to distance $U_1$. All of these $n$-1 colluding participants can construct the following (19) by using the received data from cloud service $A$ and their own data.

$$\begin{cases} a_1^T a_1 - 2a_1^T a_2 + a_2^T a_2 = D_{12} \\ a_1^T a_1 - 2a_1^T a_3 + a_3^T a_3 = D_{13} \\ \cdots\cdots \\ a_1^T a_1 - 2a_1^T a_n + a_n^T a_n = D_{1n} \\ a_2^T a_2 - 2a_2^T a_3 + a_3^T a_3 = D_{23} \\ a_2^T a_2 - 2a_2^T a_4 + a_4^T a_4 = D_{24} \\ \cdots\cdots \\ a_2^T a_2 - 2a_2^T a_n + a_n^T a_n = D_{2n} \\ \cdots\cdots \\ \cdots\cdots \\ a_{n-2}^T a_{n-2} - 2a_{n-2}^T a_{n-1} + a_{n-1}^T a_{n-1} = D_{(n-2)(n-1)} \end{cases} \tag{19}$$

For the participants, values $\{a_1^T a_2, a_1^{Ta_3}, \ldots, a_1^T a_n, a_2^T a_3, a_2^T a_4, \ldots, a_2^T a_n, \ldots, a_{n-2}^T a_{n-1}\}$ are unknown in (19) and are calculated by the cloud service through the private key. Then, $\frac{(n-1)(n-2)}{2}$ equations can be obtained when $n$-1 collusion participants combine all available information; however, there exist $\frac{n(n-1)}{2}$ unknown numbers. This proves that it is impossible for participants to calculate the distance and obtain cluster centers $\{U_1, U_1, U_3, \ldots, U_k\}$. Therefore, our method can resist any number of collusion attacks initiated by participants without revealing any information about the cluster centers.

## 6 | EXPERIMENTAL STUDY

This experiment was conducted on a cloud platform including ten computers, each with 2.0 GHz of Intel Xeon CPU E5-2620 and 4 GB physical memory, implemented in Python programming. In considering the performance and time efficiency of the typical clustering algorithm and the comparability of this study, we compared and analysed the proposed PPDPC algorithm with the k-means algorithm,[32] DPC algorithm,[14] and spectral clustering (SC) algorithm.[33]

### 6.1 | Data sets and evaluation metrics

To evaluate the performance of the PPDPC algorithm, we selected six representative data sets from the UCI machine learning library[34] for simulation experiments. The basic characteristics of the data sets are shown in Table 4.

Here, two real data sets were used to further verify the effectiveness of the PPDPC. The first data set involves predicting a user's demographic characteristics based on the user's application usage behavior on a mobile phone. Here, the attribute data includes event records of the user using the mobile phone, for example, downloading an application, location information, application type, and application event, to predict the gender and age of the mobile phone users. The data set contains 1 048 575 samples, 4 attributes, and 2 clusters.[35] The second data set lends

**TABLE 4** UCI data sets used in experiments

| Data sets | Number of samples | Number of attributes | Number of cluster's |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Wpbc | 198 | 33 | 2 |
| Heart | 303 | 13 | 2 |
| Balance | 625 | 4 | 3 |
| German | 1000 | 24 | 2 |
| Adult | 32561 | 14 | 2 |

data of club credit loan defaulters; this mainly includes loan status and repayment information. The clustering results can help the credit platform to judge customer credit rating. The data set contains 887 379 samples, 74 attributes, and 7 clusters.[36]

We used accuracy (ACC)[37] and F-measure[38] to evaluate the results of the algorithm; the scope of these two evaluation indicators is [0,1]. Considering that the algorithm generates random numbers when calculating cluster centers, nuanced clustering results may occur. Thus, the experiment runs 20 times for each algorithm in each data set and takes an average of 20 repetition tests.

The accuracy is measured as a percentage of the total number of samples that are correctly categorized in the cluster results compared to the predefined category labels in the data sets; the higher the accuracy value, the better the clustering quality. The definition of accuracy is as follows:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\omega(l_i, c_i), \tag{20}$$

where $l_i$ indicates that sample $a_i$ is the correctly clustered class label, $c_i$ indicates that the algorithm achieves the class label of sample $a_i$, when $l_i = c_i, \omega(l_i, c_i) = 1$, otherwise $\omega(l_i, c_i) = 0$.

The F-measure combines the characteristics of both precision and recall; the higher the F-measure, the more the clustering algorithm can not only distinguish the data according to category but also can divide the same data into the same cluster. The F-measure can also measure the overall performance of the clustering algorithm to a large extent, as compared to the accuracy and recall rate. The precision evaluation of the accuracy of the clustering results is calculated as shown in (21). The recall rate evaluates the completeness of the experimental results, as shown in (22), and the F-measure is calculated as in (23):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + FP} \tag{21}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{22}$$

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{23}$$

In (21)-(23), TP correctly groups a pair of similar data objects into one cluster, TN correctly partitions a pair of dissimilar data objects into two clusters, FP incorrectly groups a pair of dissimilar data objects into one cluster, and FN incorrectly partitions a pair of similar data objects into two clusters.

## 6.2 | Performance evaluation in terms of time

Before the cloud server clusters the data sets, we must calculate the distance between each sample. To measure the impact of the data set size on the calculation of the encryption time for the distance, the time taken for a quarter, half, three-fourth and all of the samples of the data sets was counted, as shown in Figure 3. For different subsets of these eight data sets, Figure 3 shows that the data amount has a significant impact on the time taken for encryption. That is, for the Iris data set, encryption time varies from 0.62 s to 1.83 s; for the Wpbc data set, from 3.67 s to 9.84 s; for the Heart data set, the encryption time varies from 2.08 s to 4.62 s; for the Balance data set, the encryption time varies from 2.11 s to 5.35 s; for the German data set, the encryption time varies from 8.86 s to 28.46 s; and for the Adult data set, the encryption time varies from 243.47 s to 635.58 s. Further, for the Mobile User data set, the encryption time ranges from 36.95 min to 97.45 min and for the Lending Club data set, from 615.86 min to 1548.14 min.

For the five UCI data sets with small sample sizes, the running times of the PPDPC algorithm and the comparison algorithms mentioned earlier are shown in Figure 4. The running time of the PPDPC algorithm is lower than those of the comparison algorithms, and no significant increase is observed when compared with the original DPC algorithm.

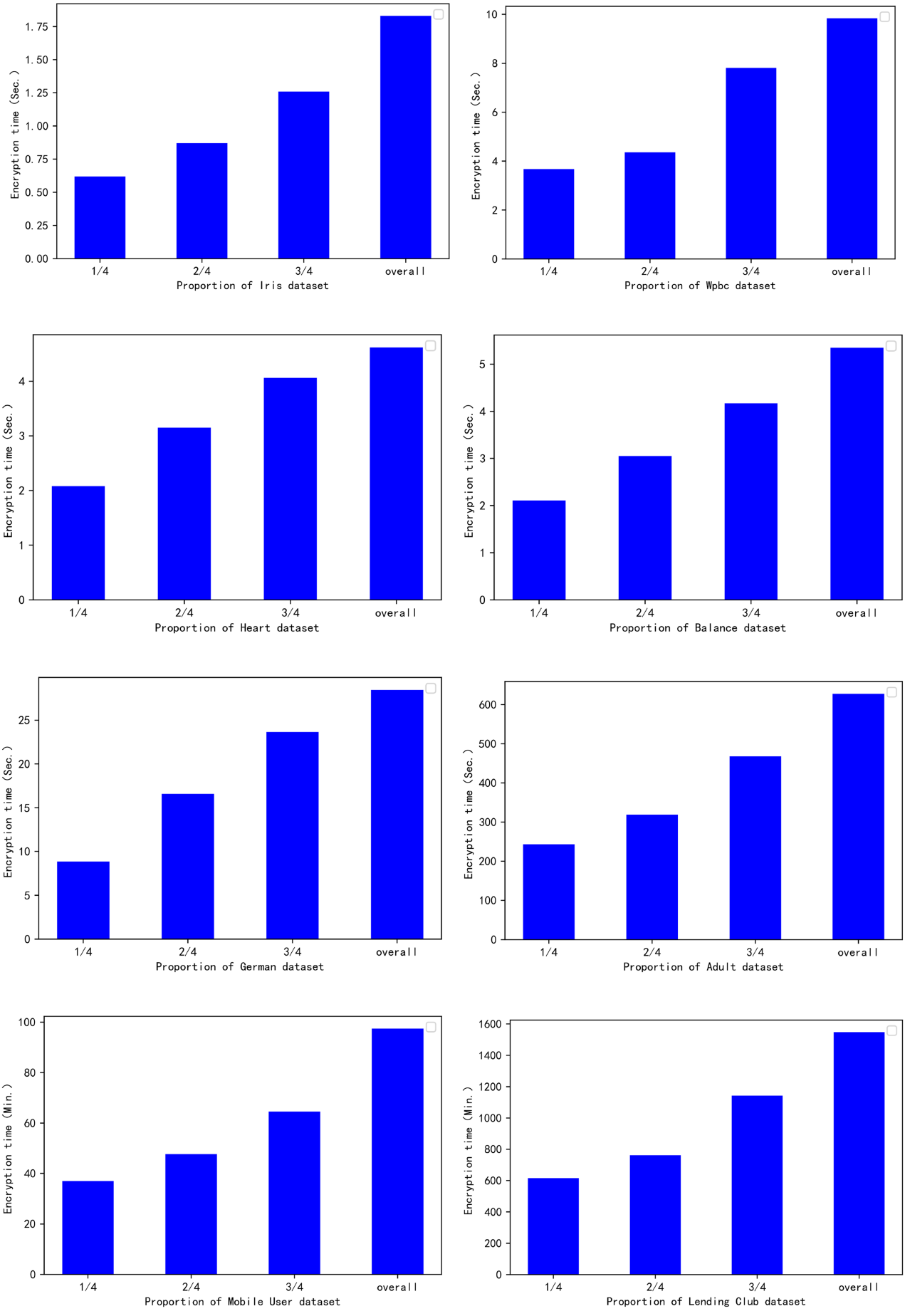


**FIGURE 3** Results of different sizes of data sets at encryption times

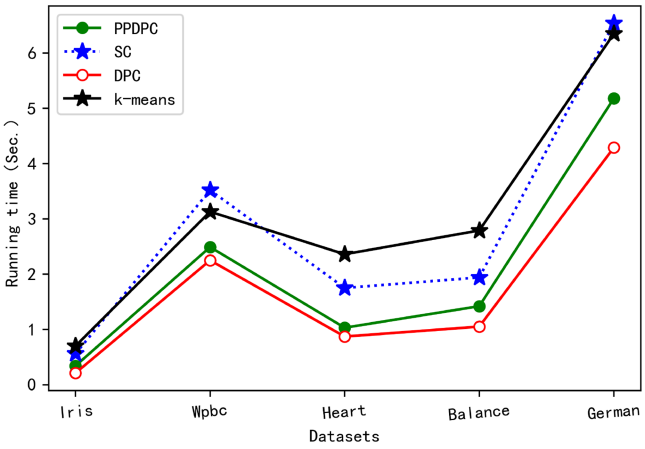


**FIGURE 4** Running times of comparison algorithms on different data sets

**TABLE 5** Running times of comparison algorithms on different data sets

| Algorithm / Data sets | *k*-means | SC | DPC | PPDPC |
|---|---|---|---|---|
| Iris | 0.72 *s* | 0.56 *s* | 0.21 *s* | 0.34 *s* |
| Wpbc | 3.13 *s* | 3.52 *s* | 2.25 *s* | 2.49 *s* |
| Heart | 2.36 *s* | 1.75 *s* | 0.87 *s* | 1.03 *s* |
| Balance | 2.79 *s* | 1.94 *s* | 1.05 *s* | 1.42 *s* |
| German | 6.35 *s* | 6.54 *s* | 4.29 *s* | 5.18 *s* |
| Adult | 131.25 *s* | 124.16 *s* | 103.92 *s* | 112.37 *s* |
| Mobile User | 19.69 min | 17.87 min | 16.52 min | 16.94 min |
| Lending Club | 280.12 min | 256.65 min | 239.83 min | 243.14 min |

For eight data sets, the running times of the PPDPC and comparison algorithms are shown in Table 5. As shown in the table, the PPDPC algorithm shows an increase in uptime compared to that in the DPC algorithm, but not more than 15.9%; it does not exceed the running times of the *k*-means and SC algorithms. The use of the PPDPC algorithm reduces the running times of data sets Iris, Adult, and Lending Club by 52.1%, 29.1% and 13.2%, respectively, compared to those when using the *k*-means algorithm. Additionally, compared to the SC algorithm, the proposed algorithm reduces the running time of the Wpbc, German and Mobile User data set by 29.3%, 20.8%, and 15.9%, respectively.

The experimental results show that in the cloud computing platform, the use of the proposed algorithm affects the time performance according to the size of the data sets, ie, the greater the scale, the more time required. For data sets with large dimensions but small sizes, the proposed algorithm shows good time performance. For sample sizes with large data sets, eg, Mobile User and Lending Club, the PPDPC algorithm calculates the time consumption of the distance between samples which has larger growth than the smaller data sets, because the algorithm needs to process more data in the running process, and thus takes more time. Compared to the original DPC algorithm, the running time of the PPDPC algorithm is not significantly increased and is superior to other comparison algorithms.

## 6.3 | Performance evaluation of clustering

For the eight aforementioned data sets, the experimental results of the privacy-preserving DPC algorithm and the comparison algorithms are shown in Table 6.

As shown in Table 6, by using two evaluation metrics (ACC and the F-measure) to evaluate the clustering results, our proposed algorithm outperforms other algorithms on average; this shows that the PPDPC algorithm produces more accurate clustering centers. For the Iris data set, the ACC index of the PPDPC algorithm is 46.9% higher than that of the DPC algorithm. For the Wpbc data set, the F-measure index of the PPDPC algorithm is 28.7% higher than that of the SC algorithm. For the Heart data set, the ACC index of the PPDPC algorithm was increased by 11.9% compared to the *k*-means algorithm. For the Balance data set, the ACC index of the PPDPC algorithm is 26% higher than that obtained by the DPC algorithm. For the German data set, the ACC index of PPDPC algorithm is 11.7% higher than that of the *k*-means algorithm. For the Adult data set, the F-measure index of the PPDPC algorithm is 11.5% higher than that of the SC algorithm. For the Mobile User data set, the ACC index of the PPDPC algorithm is 6.3% higher than that of the SC algorithm. For the Lending Club data set, the F-measure index of the PPDPC algorithm is 17.8% higher than that of the *k*-means algorithm.

As the PPDPC algorithm recognizes outliers before allocating the sample points, clusters the sample points besides the outliers, and then allocates the outliers, the clustering performance is improved compared to that of the DPC algorithm. The experimental results show that the proposed privacy-preserving clustering method has good accuracy in computing cluster centers and the clustering effect. Additionally, it can obtain the clustering result without obtaining the specific attribute information of data sets; this can better protect the privacy information of data sets.

**TABLE 6** Experimental results of different algorithms on data sets

| Data set | Algorithms | ACC | F-Measure | Dataset | Algorithms | ACC | F-Measure |
|---|---|---|---|---|---|---|---|
| Iris | *k*-means | 0.793 | 0.783 | Wpbc | *k*-means | 0.594 | 0.578 |
| | SC | 0.629 | 0.645 | | SC | 0.534 | 0.572 |
| | DPC | 0.576 | 0.712 | | DPC | 0.675 | 0.695 |
| | PPDPC | 0.846 | 0.853 | | PPDPC | 0.714 | 0.736 |
| Heart | *k*-means | 0.653 | 0.521 | Balance | *k*-means | 0.534 | 0.465 |
| | SC | 0.649 | 0.569 | | SC | 0.521 | 0.472 |
| | DPC | 0.694 | 0.614 | | DPC | 0.450 | 0.473 |
| | PPDPC | 0.731 | 0.616 | | PPDPC | 0.567 | 0.549 |
| German | *k*-means | 0.597 | 0.735 | Adult | *k*-means | 0.736 | 0.635 |
| | SC | 0.482 | 0.578 | | SC | 0.657 | 0.576 |
| | DPC | 0.645 | 0.543 | | DPC | 0.759 | 0.614 |
| | PPDPC | 0.667 | 0.621 | | PPDPC | 0.786 | 0.642 |
| Mobile User | *k*-means | 0.693 | 0.648 | Lending Club | *k*-means | 0.756 | 0.579 |
| | SC | 0.637 | 0.583 | | SC | 0.721 | 0.624 |
| | DPC | 0.712 | 0.606 | | DPC | 0.703 | 0.658 |
| | PPDPC | 0.739 | 0.635 | | PPDPC | 0.715 | 0.682 |

## 7 | CONCLUSION AND FUTURE RESEARCH

Aiming at solving the problem of privacy leakage in clustering processes, this paper presented an efficient privacy-preserving DPC algorithm, which uses a cloud computing service to safely calculate the nearest clustering centers for each participant without disclosing any clustering information to the participants. Additionally, the participant does not know the privacy information of other participants in the same cluster. Through in-depth security analysis, even if colluding participants exist, the privacy information of the remaining participants is not disclosed. No participant can obtain the private information of other participants or cluster centers. Additionally, cloud service providers calculate cluster centers without knowing participant's private information. Experimental results on the eight data sets showed that the proposed algorithm shows good time performance and clustering effects compared to the classical clustering algorithms and primitive DPC algorithm. In our future studies, we will consider the privacy preservation of other clustering algorithms and apply them to practical problems.

### ACKNOWLEDGMENTS

The authors would like to express gratitude to all the reviewers for their helpful suggestions. This work was supported in part by the National Natural Science Foundation of China under grant 61602009, grant 61972439, and grant 61702010, and in part by the Anhui Provincial Natural Science Foundation of China under grant 1808085MF172.

### ORCID

*Liping Sun* https://orcid.org/0000-0002-6678-9759

### REFERENCES

1. Torrecilla JL, Romo J. Data learning from big data. *Stat Probab Lett*. 2018;136:15-19.
2. Cheng Y, Chen K, Sun H, Zhang Y, Fei T. Data and knowledge mining with big data towards smart production. *J Ind Informat Integr*. 2018;9:1-13.
3. Armbrust M, Fox A, Griffith R. A view of cloud computing. *Commun ACM*. 2010;53(4):50-58.
4. Bhatia T, Verma AK, Sharma G. Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing. *Concurrency Computat Pract Exper*. 2019. In press.
5. Woodworth JW, Salehi MA. S3BD: secure semantic search over encrypted big data in the cloud. *Concurrency Computat Pract Exper*. 2018;31(11):1-18.
6. Wu X, Zhu X, Wu G, Ding W. Data mining with big data. *IEEE Trans Knowl Data Eng*. 2014;26(1):97-107.
7. Liu C, Beaugeard N, Yang C, Zhang X, Chen J. HKE-BC: hierarchical key exchange for secure scheduling and auditing of big data in cloud computing. *Concurrency Computat Pract Exper*. 2016;28(3):646-660.
8. Xiong J, Ren J, Chen L, et al. Enhancing privacy and availability for data clustering in intelligent electrical service of IoT. *IEEE Internet Things J*. 2018;6(2):1530-1540.
9. Xiong J, Ma R, Chen L, et al. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans Ind Informat*. 2019. In press.
10. Esposito C, Castiglione A, Martini B, Choo K-KR. Cloud manufacturing: security, privacy, and forensic concerns. *IEEE Cloud Comput*. 2016;3(4):16-22.
11. Huang X, Du X. Achieving big data privacy via hybrid cloud. Paper presented at: 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2014; Toronto, ON.
12. Lindell Y, Pinkas B. Privacy preserving data mining. Paper presented at: Annual International Cryptology Conference; 2000; Santa Barbara, CA.
13. Jagannathan G, Wright RN. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining; 2005; Chicago, IL.
14. Rodriguez A, Laio A. Clustering by fast search and find of density peaks. *Science*. 2014;344(6191):1492-1496.

15. Samarati P. Protecting respondents identities in microdata release. *IEEE Trans Knowl Data Eng*. 2001;13(6):1010-1027.
16. Dwork C. Differential privacy. In: Proceedings of the 33rd International Colloquium Automata Languages Programming; 2006; Venice, Italy.
17. Du W, Zhan Z. A practical approach to solve secure multi-party computation problems. In: Proceedings of the Workshop New Security Paradigms; 2002; Virginia Beach, VA.
18. Fontaine C, Galand F. A survey of homomorphic encryption for nonspecialists. *Eurasip J Inf Security*. 2007;2007(1):1-10.
19. Xing K, Hu C, Yu J, Chen X. Mutual privacy preserving *k*-means clustering in social participatory sensing. *IEEE Trans Ind Informat*. 2017;13(4):2066-2076.
20. Vaidya J, Clifton C. Privacy-preserving k-means clustering over vertically partitioned data. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery Data Mining; 2003; Washington, DC
21. Du W, Atallah MJ. Privacy-preserving cooperative statistical analysis. Paper presented at: Seventeenth Annual Computer Security Applications Conference; 2001; New Orleans, LA.
22. Jha S, Kruger L, Mcdaniel P. Privacy preserving clustering. In: *Computer Security-ESORICS 2005*. Berlin, Germany: Springer; 2005:397-417.
23. Bunn P, Ostrovsky R. Secure two-party *k*-means clustering. In: Proceedings of the 14th ACM Conference on Computer and Communication Security; 2007; Alexandria, VA.
24. Rao F-Y, Samanthula BK, Bertino E, Yi X, Liu D. Privacy-preserving and outsourced multi-user k-means clustering. Paper presented at: 2015 IEEE Conference on Collaboration and Internet Computing (CIC); 2015; Hangzhou, China.
25. Liu D, Bertino E, Yi X. Privacy of outsourced k-means clustering. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security; 2014; Hong Kong.
26. Rivest RL, Shamir A, Adleman LM. A method for obtaining digital signatures and public-key cryptosystems. In: Proceedings of the ACM Conference on Communications; 1978; Nagoya, Japan.
27. Yuan W, Ren X. Research on privacy preserving clustering method for horizontal partitioned data. *J Comput Technol Develop*. 2015;5:115-117.
28. Guo S, Meng X. Density peaks clustering with differential privacy. In: Proceedings of the 8th ACM Conference on Biennial Innovative Data Systems Research; 2017; Chaminade, CA.
29. Zhang Q, Zhong H, Yang L, Chen Z, Bu F. PPHOCFS: privacy preserving high-order CFS algorithm on the cloud for clustering multimedia data. *ACM Trans Multimedia Comput Commun Applications*. 2016;12(4):1-15.
30. Adleman LM. Factoring numbers using singular integers. In: Proceedings of the 23th Annual ACM Symposium on Theory Computing; 1991; New Orleans, LA.
31. Buhler JP, Lenstra HW, Pomerance C. Factoring integers with the number field sieve. In: *The Development of the Number Field Sieve*. Berlin, Germany: Springer; 1993:50-93.
32. Macqueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability; 1967; Berkeley, CA.
33. Luxburg UV. A tutorial on spectral clustering. *Statistics Computing*. 2007;17(4):395-416.
34. Most Popular Data Sets. http://archive.ics.uci.edu/ml/
35. Mobile User. https://www.kaggle.com/c/talkingdata-mobile-user-demographics/
36. Lending Club. https://www.kaggle.com/wendykan/lending-club-loan-data/
37. Carpaneto G, Toth P. Algorithm 548: solution of the assignment problem. *ACM Trans Math Software*. 1980;6(1):104-111.
38. Conrad JG, Al-Kofahi K, Zhao Y, Karypis G. Effective document clustering for large heterogeneous law firm collections. In: Proceedings of the International Conference on Artificial Intelligence and Law; 2005; Bologna, Italy.

**How to cite this article:** Sun L, Ci S, Liu X, Zheng X, Yu Q, Luo Y. A privacy-preserving density peak clustering algorithm in cloud computing. *Concurrency Computat Pract Exper*. 2020;32:e5641. https://doi.org/10.1002/cpe.5641