Computer and Information Science; Vol. 12, No. 2; 2019
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Secure Multi-User *k*-Means Clustering Based on Encrypted IoT Data

Yanling Li[1], Chuansheng Wang[1], Qi Wang[1], Jieling Dai[1] & Yushan Zhao[1]

[1] Department of Computer Science, Jinan University, China

Correspondence: Chuansheng Wang, Department of Computer Science, Jinan University, China. Tel: 86-131-2934-7321. E-mail: chueng0828@126.com

Received: February 2, 2019 Accepted: March 10, 2019 Online Published: March 25, 2019

doi:10.5539/cis.v12n2p35 URL: https://doi.org/10.5539/cis.v12n2p35

Abstract

IoT technology collects information from a lot of clients, which may relate to personal privacy. To protect the privacy, the clients would like to encrypt the raw data with their own keys before uploading. However, to make use of the information, the data mining technology with cloud computing is used for the knowledge discovery. Hence, it is an emergent issue of how to effectively performing data mining algorithm on the encrypted data. In this paper, we present a *k*-means clustering scheme with multi-user based on the IoT data. Although, there are many privacy-preserving *k*-means clustering protocols, they rarely focus on the situation of encrypting with different public keys. Besides, the existing works are inefficient and impractical. The scheme we propose in this paper not only solves the problem of evaluation on the encrypted data under different public keys but also improves the efficiency of the algorithm. It is semantic security under the semi-honest model according to our theoretical analysis. At last, we evaluate the experiment based on a real dataset, and comparing with previous works, the result shows that our scheme is more efficient and practical.

Keywords: *k*-means clustering, privacy preservation, homomorphic encryption, proxy cryptography

1. Introduction

With the growing up of Internet of Things technology, the application of Internet of Things will spread to all walks of life. IoT technology collects information through a variety of smart devices or sensors, according to the agreed protocol, transfers the information to the application platform for processing and achieves the goal of intelligent control. For example, hospital wristbands can identify patients undergoing medical care, and sport trackers can log physical activities. All of those smart devices and sensors will produce large amounts of data in its running, and to reduce cost, users would like to benefit from the outsourced services, which is one of the fundamental advantages of cloud computing.

Cloud computing is a new business computing model. As an emerging computing model, cloud computing attracts a large number of users for its characteristics such as good scalability, low-cost and pay-on-demand. More and more enterprises and users store their services and data into the cloud. That is, the users with resource-constraint devices can delegate the heavy workloads into untrusted cloud servers and enjoy the unlimited computing resources. However, such a large quantity of data always involves sensitive information, such as medical records or locations information, and it is high risk to store such information directly in the cloud servers. The security challenge is an emergent problem in the cloud computing development.

Due to the security threats to the cloud, users have to take action to protect their sensitive information. The common method is to encrypt raw data before uploading to the cloud server. Generally speaking, users do not trust others easily, and they encrypt their own data with their own keys. In other words, except securely storing data in the cloud server, it allows users to retrieve cloud data without revealing confidential data to other users or the service providers. Meanwhile, it brings new challenges to evaluate over ciphertexts under multiple keys.

Previous works have contributed on performing evaluation over encrypted data, such as Full Homomorphic Encryption (FHE) (Gentry, 2009; Dijk, Gentry, Halevi, & Vaikuntanathan, 2010; Brakerski & Vaikuntanathan, 2014), which allows performing arbitrary operations on encrypted data without decryption. Partial Homomorphic Encryption (PHE) (e.g. Elgamal HE (Elgamal, 1982), Paillier HE (Pailliar, 1982), Goldwasser–Micali HE (Goldwasser & Micali, 1982)) supports either additive or multiplicative homomorphic operation. One common problem for the previous works is that they just work on the cipehrtexts encrypted under the single key. Different from these approaches, López-Alt et al. (López-Alt, Bleumer, & Strauss, 1998) proposed a scheme that is capable of operating on inputs encrypted under multiple keys. Unfortunately, it requires heavy communication cost during the decryption of the final results, and its efficiency is far from practice. Then, Peter, Tews and Katzenbeisser (2013) represented a scheme that allows evaluating any dynamically chosen function on inputs encrypted under different independent public keys utilizing Bresson-Catano-Pointcheval (BCP) encryption (Bresson, Catalano, & Pointcheval, 2003), which is an additive homomorphic encryption with a double trapdoor decryption mechanism. The disadvantage of this solution is that it requires the complex interactions between two non-colluding cloud servers during the ciphertexts transformation phase. Therefore, it is not suitable for the system in the real word.

In this paper, to avoid all these drawbacks and achieve a better balance between efficiency and security, we construct a more efficient secure *k*-means clustering scheme under multiple keys based on two non-colluding cloud servers by utilizing the ElGamal-based Proxy Re-encryption (PRE) (Ateniese, Fu, Green, & Hohenberger, 2006). Generally speaking, it is reasonable to assume the existence of two non-colluding servers to perform the secure computation. According to Van and Juels (2010), there is a clear indication that it is impossible to realize a completely non-interactive solution in a single server setting. Hence, if we aim to implement non-interaction between data owners, we need at least two servers, just like Peter et al. (2013). Moreover, as a way to enhance efficiency, we make extensive use of ElGamal-based PRE to transform ciphertexts encrypted under multiple keys into ciphertexts under the same key before computation. Compared to Peter et al. (2013), we reduce the interaction between two servers, and increase the computing efficiency.

In brief, we summarize our main contributions as follows:

1) In order to protect privacy information, we construct a new efficiency privacy-preserving *k*-means clustering scheme. In our setting, the scheme can both preserve the privacy of the data owners' sensitive information and the calculation results.
2) Existing works require the inputs to be encrypted under the same public key, which is very limited in practice. To avoid these problems, we take advantage of proxy re-encryption to construct a scheme that is based on distributed data encrypted under multiple keys.
3) We utilize the two non-colluding servers model to complete large calculation in the learning phase. Except computing proxy keys, the data owners should do nothing during the learning process. In the end, the data owners only need to do decryption to obtain the result.
4) Finally, we evaluate our scheme based on a real dataset. Besides, we make comparison with other works, and the results show that our scheme is more efficient and more practical.

Organization. The rest paper is organized as follows. In Section 2, we introduce the related work about our work. Next, we represent the setting of our scheme and analyze the threat model in Section 3. Section 4 describes the preliminary knowledge and privacy-preserving building blocks. We introduce our scheme in detail in Section 5, while analyzing security in Section 6. We summarize our experimental results in Section 7. Finally, we conclude in Section 8.

2. Related Work

Previous works have focused on the issues of privacy-preserving clustering algorithm. In the early years, researchers mainly focused on the security *k*-means clustering based on a single databased, and made some achievements. Recently, the focus has shifted to the multiple data sources setting to obtain more precise clustering result. In the scheme (Bunn & Ostrovsky, 2007), Bunn and Ostrovsky (2007) proposed a secure two-party *k*-means clustering protocol that guaranteed privacy of each database without revealing the intermediate values, which was based on secure two-party computation. This scheme extends the clustering algorithm to an algorithm that works in the two-database setting. However, as we all know, secure multiparty computation will increase communication cost among participating parties. Besides, it is a heavy work for users to perform the data mining algorithm, because of its limited computation resources.

To address this issue, researchers have started to focus on the data mining task in an outsourced environment (Rao, Samanthula, Bertino, Yi, & Liu, 2015; Jiang et al., 2018; Rong, Wang, Liu, Hao, & Xian, 2017; Samanthula, Rao, Bertino, Yi, & Liu, 2014; Xing, Hu, Yu, Cheng, & Zhang, 2017). The works of Rao et al. (2015) and Samanthula et al. (2014) outsourced all computation to two non-colluding cloud servers. In their works, users encrypt their own raw data under a cloud server's public key and upload them to the other cloud server. The two cloud servers collaboratively perform the clustering task on the combined data in a privacy-preserving manner. Since all data are encrypted under a unified key, and only the cloud server who holds the secret key can decrypt the ciphertext, it is hard for data owners to retrieve data from the cloud servers.

Recently, there are many researches about privacy-preserving data mining based on encrypted database encrypted under multiple keys. Jiang et al. (2018) proposed a secure k-means clustering protocol with the benefit of two non-colluding cloud servers to support storage and computation outsourcing. The raw data are encrypted under the data owner's public key, and it is convenient for them to retrieve its data and decrypt with its secret key. While, during the clustering procedures, the data owners should online all the time, and help work out the result. Similarly, data owners need participate in the whole clustering process in scheme Xing et al. (2017). They allow data users to compute the nearest cluster locally and update the cluster centers with the help of cloud server. Obviously, it will increase the communication cost between each entity and there exists a large number of calculations for data owners. Beyond that, there are some potential problems about data security in the update cluster centers phase. Rong et al. (2017) presented a privacy-preserving k-means clustering over a joint database encrypted under multiple keys in distributed cloud environments. They transformed ciphertexts under different keys into ones under a common key through a double decryption cryptosystem (Youn, Park, Kim, & Lim, 2005), which allows an authority to decrypt any ciphertext by using the master key without consent of corresponding owner. The problem with this method is that it will occur without the data owners' prior consent, which may against data owners' wishes. In addition, at the stage of ciphertext transformation, the ciphertexts are converted by two non-colluding servers, and it may significantly decrease the efficient computation. Obviously, it is still a problem to be solved imperatively that achieves efficient privacy-preserving k-means clustering algorithm under multiple keys.

3 Setting and Threat Model

3.1 Architecture and Entities

In order to solve the existing problems, we present a secure and efficient privacy-preserving k-means clustering protocol. In our setting, as shown in Figure 1, we consider n data owners, and each of them with an d-dimensional object $\boldsymbol{a}_i$ $(1 \leq \mathrm{i} \leq n)$. Due to the security concerns, data owners carefully store data in an encrypted form. Once they want to get some information from the data, they will send request to the cloud server. In other words, data mining will be executed based on the cloud model in a secure manner. We only discuss k-means clustering as the data mining method in this paper, and it obviously can be extended to other data mining algorithm.

There are two type of entities in our system model: data owners and cloud servers.

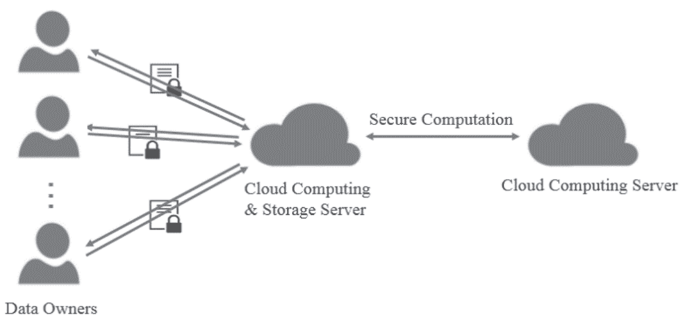

Figure 1. System Model under Consideration

1) **Data Owners (DO)**: Data owner encrypts data under his own public key, and he is the only person who can decrypt the ciphertext by the secret key. In general, the ciphertexts would be centralized into storage service provider. Data owners have the right to decide whether their data will be involved in the k-means clustering algorithm.
2) **Cloud Computing and Storage Server (S)**: The cloud computing and storage server provides storage service to all data owners, and it will perform computation on the data when receiving request from the clients.
3) **Cloud Computing Server (C)**: Cloud computing server is a temporary server with only computing service. Its main work is to assist the cloud server S to perform data mining algorithm in a privacy-preserving manner.

3.2 Threat Model

In this paper, we prefer outsourcing the data and computation to a server provider, such as the cloud server. While, due to many reasons, the clouds are unreliable, and they may try to collude with others to obtain uncorrupted parties' private information as many as possible. During the evaluation process, the corrupted parties may also deviate from the protocol specification according to the adversary instruction. Under these circumstances, the adversaries are called malicious adversaries. In our study, we mainly focus on the semi-honest model. In other words, the entities in our setting are all semi-honest adversaries. That is to say, they will execute the protocol correctly, but they also attempt to obtain some information about the users' private information. In addition, we assume that the entities do not collude with each other.

The design goal of our scheme is to ensure the data owners obtain the results of clustering, while the clouds do not learn any information from the clustering algorithm, even the intermediate value. It is worth noting that the ciphertexts stored in the cloud server are all encrypted under different public keys. Hence, we mainly aim to effectively solve the privacy-preserving k-means clustering under multi-keys. We make benefits from the proxy re-encryption to convert the ciphertexts into the ones under a unified key. During the transformation process, the entities learn nothing about the data owners' information. The clustering process is based on a two-cloud model. They execute the protocol correctly and do not learn any extra information. Thus, we say that our system is private. Informally, we also say that our protocol is correct. It is easy to verify.

In addition, during the clustering process, we make sure that the communication cost and computation is minimized as possible. In the study, we consider that the data owners are not willing to help to run collaborative analysis and to spend too much resources to execute it. Consequently, we outsource the computation to computation service providers, and ensure that the communication cost is minimized between the two cloud servers.

4. Preliminaries

4.1 k-Means Clustering Algorithm

k-means clustering algorithm as one of the main data mining methods is widely used in practice. It can be used to partition a set of data objects into clusters. We assume that there are n participants and each participant holds an d-dimensional object $\boldsymbol{a}_i$ $(1 \leq i \leq n)$. k-means clustering algorithm aims to divide the n objects into k clusters $\boldsymbol{c}_j$ $(1 \leq j \leq k)$, and to ensure a large degree of similarity within the same class, but little similarity between different classes. The clustering process is comprised of two steps. The first step is to assign objects into different clusters. The criterion of classification is to measure the distance between a sample $\boldsymbol{a}_i$ and the related cluster center $\boldsymbol{c}_j$. There are many methods for this criterion, but in this paper, we adopt the Euclidean distance. At each iteration of the first step, k-means clustering algorithm assigns the object to the nearest cluster, which is labeled by c_i, and it follows

$$c_i = \underset{j}{\operatorname{argmin}} ||\boldsymbol{a}_i - \boldsymbol{c}_j||^2 \tag{1}$$

where $1 \leq i \leq n,\ 1 \leq j \leq k$. All objects will be divided into k clusters during the first step, and the second step is to update the cluster centers. The new cluster center is defined as the center of each cluster, and we assume that there exists n_j objects in the j-th cluster. The updating algorithm is given by

$$\boldsymbol{u}_j = \frac{\sum_{i=1}^{n_j} \boldsymbol{a}_i}{n_j} \tag{2}$$

where $1 \leq \mathrm{i} \leq n_j,\ 1 \leq \mathrm{j} \leq k$. The clustering process is terminated if the cluster centers are sufficiently close to the previous ones or the iterations reach a certain number of times.

4.2 Additively Homomorphic Proxy Re-Encryption

Proxy Re-Encryption (PRE) allows an honest-but-curious proxy to transform a ciphertext computed under Alice's public key into one that can be opened by Bob's secret key, without disclosing the plaintext. In 2006, Ateniese et al. (2006) proposed a unidirectional PRE, which is an improvement over Blaze, Bleumer and Strauss (1998) where the keys are bidirectional relied on pairing-based cryptography. In this paper, to enable the additive homomorphic property, we use the algebraic structure of elliptic curves over finite fields, similar to (Wang, M Li, Chow, & H Li, 2014; Shafagh, Hithnawi, Burkhalter, Fischli, & Duquennoy, 2017). The PRE scheme is also based on the bilinear map (Boneh & Franklin, 2001), which, given a cyclic group $\mathbb{G}$ of prime order q, has the following property for $a, b \in \mathbb{Z}_q$ and $g, h \in \mathbb{G}$: $\mathrm{e}(g^a, h^b) = e(g, h)^{ab}$. The additively homomorphic proxy re-encryption (AHPRE) scheme can be described as follows:

- $\mathrm{Setup}(1^\kappa) \rightarrow \{g, \mathbf{e}, \mathrm{Z}, \mathbb{G}, \mathbb{G}_T\}$: Input 1^κ, where κ is a security parameter. Choose a random generator $g \in \mathbb{G}$. $\mathbf{e}$ as the map: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and $\mathrm{Z} = \mathbf{e}(g, g) \in \mathbb{G}_T$.
- $\mathrm{KeyGen} \rightarrow (\mathrm{pk}, \mathrm{sk})$: Choose a random number $a \in \mathbb{Z}_q$, and set the public key as $\mathrm{pk}_a = g^a$ with secret key $\mathrm{sk}_a = a$.
- $\mathrm{ReKeyGen}(\mathrm{sk}_a, \mathrm{pk}_b) \rightarrow \mathrm{rk}_{a \rightarrow b}$: A user A delegates to B with public key $\mathrm{pk}_b = g^b$, the re-encryption key is computed as $\mathrm{rk}_{a \rightarrow b} = {\mathrm{pk}_b}^{1/a} = g^{b/a} \in \mathbb{G}$.

- $\mathrm{Enc}_1(\mathrm{pk}, m) \to c$: Present the message m as $M = Z^m \in \mathbb{G}_T$. To encrypt M under pk_a in such a way that it can only decrypted by the holder of sk_a, output the first-level ciphertext $c_a = (Z^{ak}, Z^{m+k})$.
- $\mathrm{Enc}_2(\mathrm{pk}, m) \to c$: To encrypt $M = Z^m \in \mathbb{G}_T$ under pk_a in such a way that it can only decrypted by A and her delegates, output the second-level ciphertext $\mathrm{c} = (g^{ak}, Z^{m+k})$.
- $\mathrm{ReEnc}(\mathrm{rk}, c) \to c_b$: Given a second-level ciphertext $\mathrm{c} = (g^{ak}, Z^{m+k})$, compute $\mathbf{e}(g^{ar}, g^{b/a}) = Z^{br}$ and output $c_b = (Z^{bk}, Z^{m+k})$.
- $\mathrm{Dec}_1(\mathrm{sk}, c) \to M$: To decrypted a first-level ciphertext $\mathrm{c} = (Z^{ak}, Z^{m+k})$ with $\mathrm{sk}_a = a$, compute
$$M = \frac{Z^{m+k}}{(Z^{ak})^{1/a}} = \frac{Z^{m+k}}{Z^k} = Z^m.$$
- $\mathrm{Dec}_2(\mathrm{sk}, c) \to M$: To decrypted a second-level ciphertext $\mathrm{c} = (g^{ak}, Z^{m+k})$ with $\mathrm{sk}_a = a$, compute
$$M = \frac{Z^{m+k}}{\mathbf{e}(g^{ak}, \mathrm{g})^{1/a}} = \frac{Z^{m+k}}{Z^k} = Z^m.$$

Note that in final decryption, we need to map back M to m, where the message is a finite and relatively small number, which can be obtained by solving a discrete log problem.

Homomorphic Properties The AHPRE is an additively homomorphic cryptosystem, and it satisfies the following properties:

- Given two ciphertexts $\mathrm{Enc}(m_1)$ and $\mathrm{Enc}(m_2)$, we have
$$\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 + m_2). \tag{3}$$
- Given a ciphertext $\mathrm{Enc}(m)$ and a constant c, we have
$$\mathrm{Enc}(m)^c = \mathrm{Enc}(cm). \tag{4}$$

4.3 Basic Cryptographic Primitives

In this section, we mainly introduce a group of cryptographic primitives that will be used in our privacy preserving scheme.

- **Secure Multiplication Protocol (SMP):**

This protocol aims to compute the multiplication of two ciphertexts. Assume that S has two ciphertexts $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$, it will obtain the encrypted multiplication $\mathrm{Enc}(xy)$ with the help of C who has the corresponding secret key sk. The details of SMP is described in **Protocol 1**.

Protocol 1 Secure Multiplication Protocol

```
      Server's (S) Input: two ciphertexts $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$;
      Server's (C) Input: the secret key sk.
      Output S: $\mathrm{Enc}(xy)$
1: S randomly chooses two numbers $r_1, r_2 \in \mathbb{Z}_q$ and sends $\mathrm{Enc}(xr_1) \leftarrow \mathrm{Enc}(x)^{r_1}$ and $\mathrm{Enc}(yr_2) \leftarrow \mathrm{Enc}(y)^{r_2}$ to C.
2: C decrypts $\mathrm{Enc}(xr_1)$ and $\mathrm{Enc}(yr_2)$, and computes the multiplication $xyr_1r_2$, then re-encrypts it, and sends the ciphertext $\mathrm{Enc}(xyr_1r_2)$ to S.
3: S computes $\mathrm{Enc}(xy) \leftarrow \mathrm{Enc}(xyr_1r_2)^{(r_1r_2)^{-1}}$.
4: S outputs $\mathrm{Enc}(xy)$.
```

- **Secure Squared Euclidean Distance Protocol (SSEDP):**

Let $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$ and $\boldsymbol{y} = (y_1, y_2, \dots, y_d)$ denote two d-dimentional vectors. In this protocol, S with the inputs $\mathrm{Enc}(\boldsymbol{x})$ and $\mathrm{Enc}(\boldsymbol{y})$ and C with the corresponding secret key sk. They compute the squared Euclidean distance based on the SMP. At last, C obtains the encrypted result $\mathrm{Enc}(|\boldsymbol{x} - \boldsymbol{y}|^2)$. The process of SSEDP is described in **Protocol 2**.

Protocol 2 Secure Squared Euclidean Distance Protocol

```
      Server's (S) Input: two ciphertexts $\mathrm{Enc}(\boldsymbol{x})$ and $\mathrm{Enc}(\boldsymbol{y})$;
      Server's (C) Input: the secret key sk.
      Output S: $\mathrm{Enc}(|\boldsymbol{x} - \boldsymbol{y}|^2)$
1: for $i = 1$ to $d$ do
2:     S computes $\mathrm{Enc}(x_i - y_i) \leftarrow \mathrm{Enc}(x_i) \cdot \mathrm{Enc}(y_i)^{-1}$.
3:     S and C execute SMP to compute $\mathrm{Enc}((x_i - y_i)^2)$.
4: S outputs $\mathrm{Enc}(|\boldsymbol{x} - \boldsymbol{y}|^2) \leftarrow \prod_{i=1}^{d} \mathrm{Enc}((x_i - y_i)^2)$.
```

- **Secure Minimum out of *2* Numbers Protocol (SMINP$_2$):**

This protocol considers the S with inputs $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$ and the server C with the secret key sk. The SMINP$_2$ can be used to determine the relationship between two encrypted data. The **Protocol 3** shows the detailed process.

Protocol 3 Secure Minimum out of *2* Numbers Protocol

```
      Server's (S) Input: two ciphertexts $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$;
      Server's (C) Input: the secret key sk.
      Output S: $\mathrm{Enc}(\mathrm{argmin}(x,y))$
 1: S randomly chooses a non-zero random numbers $r \in \mathbb{Z}_q$.
 2: S flips a coin $s$.
 3: if $s == 1$ then
 4:     S computes $\mathrm{Enc}(l) = (\mathrm{Enc}(x) \cdot \mathrm{Enc}(y)^{q-1})^r$.
 5: else
 6:     S computes $\mathrm{Enc}(l) = (\mathrm{Enc}(y) \cdot \mathrm{Enc}(x)^{q-1})^r$.
 7: S sends $\mathrm{Enc}(l)$ to C.
 8: C decrypts $\mathrm{Enc}(l)$ and obtains $l$.
 9: if $|l| > |q|/2$ then
10:     Set $\alpha \leftarrow 2$.
11: else
12:     Set $\alpha \leftarrow 1$.
13: C encrypts $\alpha$ by computing $\alpha' = \mathrm{Enc}(\mathrm{pk}, \alpha)$, and sends $\alpha'$ to S.
14: S:
15: if $s == 1$ then
16:     S computes $\mathrm{Enc}(\min) \leftarrow \mathbf{MinFun}(\mathrm{Enc(x)}, \mathrm{Enc(y)}, \alpha')$.
17:     S computes $\mathrm{Enc}(\min) \leftarrow \mathbf{MinFun}(\mathrm{Enc}_2(c_x), \mathrm{Enc}_2(c_y), \alpha')$.
18: else
19:     S computes $\mathrm{Enc}(\min) \leftarrow \mathbf{MinFun}(\mathrm{Enc(y)}, \mathrm{Enc(x)}, \alpha')$.
20:     S computes $\mathrm{Enc}(\min) \leftarrow \mathbf{MinFun}(\mathrm{Enc}_2(c_y), \mathrm{Enc}_2(c_x), \alpha')$.
21: return Enc(min).
```

MinFun$(\mathrm{Enc}(x), \mathrm{Enc}(y), \alpha)$

```
      Server's (S) Input: three ciphertexts $\mathrm{Enc}(x)$, $\mathrm{Enc}(y)$ and $\mathrm{Enc}(\alpha)$;
      Server's (C) Input: the secret key sk.
      Output S: $\mathrm{Enc}(\mathrm{argmin}(x,y))$
1: S computes:
2:     $\mathrm{Enc}(x\alpha) \leftarrow \mathbf{SMP}(\mathrm{Enc}(x), \mathrm{Enc}(\alpha))$,
3:     $\mathrm{Enc}(y\alpha) \leftarrow \mathbf{SMP}(\mathrm{Enc}(y), \mathrm{Enc}(\alpha))$
4:     $\mathrm{Enc}(2y) \leftarrow \mathrm{Enc(y)}^2$
5:     $\mathrm{Enc}(x\alpha + 2y) \leftarrow \mathrm{Enc}(xa) \cdot \mathrm{Enc}(2y)$
6:     $\mathrm{Enc}(x + y\alpha) \leftarrow \mathrm{Enc}(x) \cdot \mathrm{Enc}(y\alpha)$
7:     $\mathrm{Enc}(-x - y\alpha) \leftarrow \mathrm{Enc}(x + y\alpha)^{q-1}$
8:     $\mathrm{Enc}(x\alpha + 2y - x - y\alpha) \leftarrow \mathrm{Enc}(x + y\alpha) \cdot \mathrm{Enc}(-x - y\alpha)$
9: return $\mathrm{Enc}(\mathrm{argmin}(x,y)) \leftarrow \mathrm{Enc}(x\alpha + 2y - x - y\alpha)$
```

5. The Procedures of Our Scheme

In this section, we describe our privacy-preserving scheme for the k-means clustering algorithm in detail. We consider that there are n data owners, and each of them holds an d-dimensional object $\boldsymbol{a}_i$ $(1 \leq \mathrm{i} \leq n)$. They encrypt raw data with their own public key pk_i $(1 \leq i \leq n)$ to the second-level ciphertexts and upload to the cloud server S. Under this circumstance, data owners are still able to retrieve their data and decrypt them under their own secret keys sk_i without leaking any information to other participants. Once receiving the request, the cloud server S will perform the clustering algorithm with the cloud server C in a privacy preserving manner.

We divide our scheme into three stages: (1) Ciphertexts transformation; (2) Assigning records to the nearest cluster; (3) Computing the new clustering centers.

5.1 Ciphertexts Transformation

Due to the ciphertexts are encrypted under different public keys, it is hard to perform evaluation on these data. Although, López-Alt et al. (2012) have presented a multi-key fully homomorphic encryption cryptosystem, it is really inefficient in the practical applications. To realize data encryption and communication protection and improve the calculation efficiency, we design a ciphertexts transformation method based on the proxy re-encryption (Ateniese et al., 2006). This method aims to convert the ciphertexts into the ones under a unified key.

Noting that the ciphertexts stored in the cloud server are all encrypted in the form of the second-level ciphertext. The data owners have the right to decide whether they will participant into the data mining. If they would like to participant the k-means clustering to comparison with others, which may help them to reacquainted themselves, they will send the request to the cloud server S with a proxy re-encryption key $\mathrm{rk}_{i\to C}$, which is computed based on the data owners' secret key and the cloud server C's public key.

$$\mathrm{rk}_{i\to C} \leftarrow \mathrm{ReKeyGen}(\mathrm{sk}_i, \mathrm{pk}_\mathrm{C}),$$

where the sk_i is the i-th participant's secret key and pk_C is the C's public key that is broadcast to all entities. Once receiving clustering request and the proxy re-encryption key $\mathrm{rk}_{i\to C}$ from data owners, the cloud server S begins to execute the re-encryption function $\mathrm{ReEnc}(\mathrm{Enc}_2(\mathrm{pk}_i, \boldsymbol{a}_i), \mathrm{rk}_{i\to C})$ to convert the ciphertext computed under DO_i's public key pk_i into the encryption under the cloud server C's public key pk_C.

$$\mathrm{Enc}_1(\mathrm{pk}_C, \boldsymbol{a}_i) \leftarrow \mathrm{ReEnc}(\mathrm{Enc}_2(\mathrm{pk}_i, \boldsymbol{a}_i), \mathrm{rk}_{i\to C}).$$

Next work will be executed on the ciphertexts under the cloud server C's public key. To simply, we denote $\mathrm{Enc}(\boldsymbol{a}_i)$ as the first-level ciphertexts under pk_C.

5.2 Assigning Records to the Nearest Cluster

The second step is to assign records to their nearest clusters by computing the minimum squared Euclidean distance. The cloud server S's first task is to initialize k cluster centers $(\boldsymbol{u}_1, \dots, \boldsymbol{u}_k)$. There are many initialization methods. The general method is to initialize centers with randomly generated values. Alternatively, to reduce the number of iterations required in the process of clustering, we can adopt an optimized manner proposed in (Ostrovsky, Rabani, Schulman, & Swamy, 2012). In this paper, we randomly choose k data records as the cluster centers.

Let D_{ij} denotes the squared Euclidean distance between the record $\boldsymbol{a}_i$ and the cluster center $\boldsymbol{u}_j$. It is easy for S to compute D_{ij} $(1 \le i \le n, 1 \le j \le k)$ by performing the protocol SSEDP defined in Section 3. To assign a record to the nearest cluster, it needs to compare the squared Euclidean distance D_{ij} between the record $\boldsymbol{a}_i$ and the k cluster centers $\boldsymbol{u}_j$ $(1 \le j \le k)$. We have present a protocol SMINP$_2$ for securely get the minimum, and we assign the record to the cluster based on the minimum squared Euclidean distance and update the cluster label corresponding to the record $\boldsymbol{a}_i$ to $c_i = j$ at the same time. It is worth mentioning that the cluster label is encrypted with the cloud C's public key under the second-level encryption in our scheme. Its good point is the convenience for returning the final results to the data owners. Just like before, it only needs performing the proxy re-encryption to convert the ciphertexts to ones that encrypted under data owners' public keys.

This process repeats until the cluster center is figured out. The data set will be divided into k subsets. Note that when this procedure terminates, the cloud servers will learn nothing about the raw data and which cluster $\boldsymbol{a}_i$ belongs to.

5.3 Computing the New Clustering Centers

After assigning the records to the nearest cluster, the cloud server S needs to reevaluate the cluster center $\boldsymbol{u}_j$ for each cluster. We assume that there are n_j records in the j-th cluster. The new cluster center is simply defined as the center of each cluster, and the updating process as follows:

$$\mathrm{Enc}(\boldsymbol{u}_j) = \frac{\prod_{i=1}^{n_j} \mathrm{Enc}(\boldsymbol{a}_i)}{n_j}.$$

The stage-2 and stage-3 is an iterative process. They will repeat until the termination condition holds. In this paper, we set two termination conditions:

1) We set a reasonable number β, and execute the algorithm verifies whether the sum of the squared Euclidean distances between the current and new clusters is upper-bounded by β.

2) If the number of iterations reaches a set value, the iteration will be terminated too.

If the termination condition holds, the clustering will halt and return the final result. Otherwise, the algorithm continues to the next iteration with the new clusters as input.

6. Security Analysis

In this section, we mainly analyze the security about our privacy-preserving k-means clustering scheme. Obviously, the data confidentiality of our scheme is achieved by the additively homomorphic proxy re-encryption. Beyond that, we assume that there is no collusion between the cloud server S and the cloud server C. And our scheme aims to achieve the privacy of the raw data, the intermediate values and the final results under the semi-honest model.

The additively homomorphic proxy re-encryption (AHPRE) is semantic security if the following problem is hard in $(\mathbb{G}, \mathbb{G}_T)$:

$$\text{Given } (g, g^a, g^b, Q), \text{ for } g \leftarrow \mathbb{G},\ a, b \leftarrow \mathbb{Z}_q$$
$$\text{and } Q \in \mathbb{G}_T, \text{ decide if } Q = e(g,g)^{a/b}.$$

Theorem 1. If the decisional bilinear Diffie-Hellman inversion assumption (DBDHI) is hard, the AHPRE is semantically security.

Proof. While, Dodis and Yampolskiy (2005) have proved that the above assumption is hard in the generic group model. In (Dodis & Yampolskiy 2005), Dodis and Yampolskiy (2005) address a stronger version called q-Decisional Bilinear Diffie-Hellman Inversion (q-DBDHI). The q-DBDHI problem asks: given the tuple $(g, g^x, \dots, g^{(x^q)})$ as input, distinguish $e(g,g)^{1/x}$ from random. The q-DBDHI assumption holds if no probabilistic polynomial time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DBDHI problem in $\mathbb{G}$.

$$\left|\Pr\left[\mathcal{A}\left(g, g^x, \dots, g^{(x^q)}, e(g,g)^{\frac{1}{x}}\right) = 1\right] - \Pr\left[g, g^x, \dots, g^{(x^q)}, \Gamma\right] = 1\right| \le \epsilon,$$

Where the probability is taken over the internal coin tosses of $\mathcal{A}$ and choices of $x \in \mathbb{Z}_p^*$ and $\Gamma \in \mathbb{G}$.

Definition 1. *(q-DBDHI Assumption). We say that the (t, q, ϵ) –DBDHI assumption holds in $\mathbb{G}$ if no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DBDHI problem in $\mathbb{G}$.*

Privacy of Data. To protect the data privacy of the data owners, we allow the data owners encrypt the sensitive information with its own pair of public and private keys under the additively homomorphic proxy re-encryption (AHPRE) cryptosystem. According to the **Theorem 1** and the assumption of non-colluding between two cloud servers, we can easily achieve the protection of individual privacy.

Lemma 1. Without any collusion, our scheme is privacy-preserving for the training model under the additively homomorphic proxy re-encryption.

Proof. Recall that in our basic scheme, the cloud server C does not communicate with the cloud server S until the training phase. It's worth mentioning that data owners do not need to communicate with the cloud servers. During the training phase, the cloud server S always perform the evaluation on the ciphertexts, and it can't obtain any information about the intermediate values, because the additively homomorphic proxy re-encryption scheme is semantically secure. Although the cloud server C keeps the secret key, it receives the blinded ciphertexts and can only be able to get the blinded messages. Hence, the cloud server C also cannot obtain the learning result. Therefore, the privacy of the training model is confidential.

7. Evaluation and Implementation

7.1 Scheme Evaluation

In this subsection, we will analyze the communication and computation overhead incurred in each stage of the proposed scheme. The results regarding computational costs are given in Table 1. Here d denotes the number of attributes, n denotes the sum numbers of data records of participants, k denotes the number of clusters. Besides, **Map** represents the bilinear map, **Mul** represents the multiplication, **Exp** represents the exponentiation. It is important to note that Stage 1 is run only once whereas Stage 2 and Stage 3 are run in an iterative manner until the termination condition holds. In addition, it clearly shows that the computational and communication costs of Stage 2 are significantly higher than the costs incurred in Stage 3 in each iteration.

7.2 Implementation and Dataset Description

We implemented the proposed scheme in Python using the GNU Multiple Precision Arithmetic (GMP) library

(The GNU MP Bignum Library). The experiments are conducted on Ubuntu 16.04.3 LTS with Intel(R) Core(TM) i7-6700 CPU 3.40GHz and 4GB RAM. To facilitate comparisons, we also use KEGG Metabolic Reaction Network dataset (Naeem & Asghar, 1999) from the UCI KDD archive. The dataset consists of 65554 data records and 29 attributes. As part of the pre-processing, we normalized the attribute values and scaled them into the integer domain.

Table 1. Computational and communication costs of the scheme

Stage	Computational Costs	Communication Costs (in bits)
Stage 1	$d * n$ **Map**	$n\|q\|$
Stage 2 (per iteration)	$n * (kd(d+4) + 25(k-1))$ **Mul** $n * (8kd + 40(k-1))$ **Exp**	$n * (6kd + 28(k-1))\|q\|$
Stage 3 (per iteration)	$n + 25k$ **Mul** $40k$ **Exp**	$28k\ \|q\|$

In this paper, we focus on performing evaluation of arbitrary functions on inputs that are encrypted under different independent public keys. We compare the performance of the ciphertexts transformation with two previous works. The result is showed in Figure 1. It clearly shows that the scheme we proposed in this paper is more efficient than the other two schemes. In addition, the communication costs in Table 1 is also less than the scheme of Rong et al. (2017) and Peter et al. (2013). To apply the privacy-preserving scheme to complex practical system, it is obvious that our scheme is the best choice to convert the ciphertexts into ones under a unified key.

Furthermore, we implemented the secure *k*-means clustering algorithm based on the KEGG Metabolic Reaction Network dataset. The performance of the scheme is shown in Figure 2. We tested our scheme with different dimensional data (i.e $d = 29, d = 20, d = 10$). As we can see, the computational time depends on the size of the dataset. The most of the costs are resulted by the partial homomorphic property, since it needs much time to perform the **SMP** which is involved many decryption and encryption algorithm.

Generally speaking, the proposed scheme in this paper is more efficient and more practical according to the result. Furthermore, we note that the data owners need not participate in the learning phase, and all the hard computations are outsourced to cloud servers, which is lightweight for them.

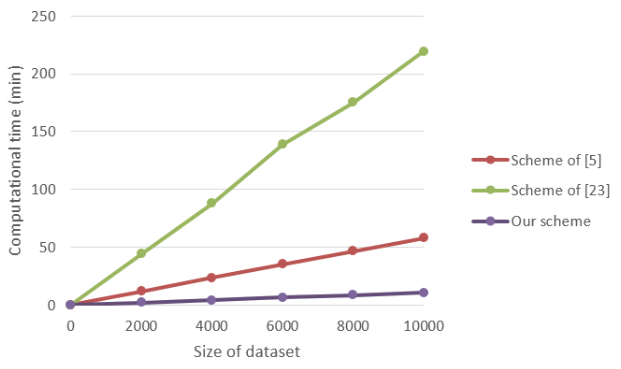

Figure 1. Cloud running time for ciphertexts transformation (in min)

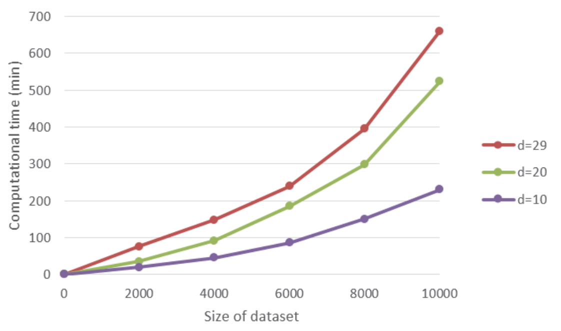

Figure 2. Performance of the proposed scheme with varying size of datasets

8. Conclusion and Future Work

In this paper, we aim to solve the problems of *k*-means clustering algorithm on inputs that are encrypted under different independent public keys. Our scheme is based on two non-colluding cloud servers, and during the process, there is no interaction between the cloud servers and the data owners. We have proved that our scheme is semantically secure in the semi-honest model. Moreover, we highlight its efficiency by giving the experimental results and comparing with previous works.

To meet the needs of practical application, we will continue improve the efficiency of the learning algorithm. Furthermore, we are planning to experiment with other machine learning algorithms used in other application scenario.

References

Ateniese, G., Fu, K., Green, M., & Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage[J]. *ACM Transactions on Information and System Security (TISSEC), 9*(1), 1-30.

Bellare, M., Hoang, V. T., & Rogaway, P. (2012). Foundations of garbled circuits[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, 784-796.

Blaze, M., Bleumer, G., & Strauss, M. (1998). Divertible protocols and atomic proxy cryptography[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 127-144.

Boneh, D., & Franklin, M. (2001). Identity-based encryption from the Weil pairing[C]//Annual international cryptology conference. Springer, Berlin, Heidelberg, 213-229.

Brakerski, Z., & Vaikuntanathan, V. (2014). Efficient fully homomorphic encryption from (standard) LWE[J]. *SIAM Journal on Computing, 43*(2), 831-871.

Bresson, E., Catalano, D., & Pointcheval, D. (2003). A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2003, 37-54.

Bunn, P., & Ostrovsky, R. (2007). Secure two-party k-means clustering[C]//Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007, 486-497.

Dodis, Y., & Yampolskiy, A. (2005). A verifiable random function with short proofs and keys[C]//International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2005, 416-431.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms[J]. *IEEE transactions on information theory, 31*(4), 469-472.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the 41st annual ACM symposium on Symposium on theory of computing-STOC'09. Vol. 9[J]. 2009.

Goldwasser, S., & Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information[C]//Proceedings of the fourteenth annual ACM symposium on Theory of computing. ACM, 1982, 365-377.

Jiang, Z. L. et al. (2018). Efficient Two-Party Privacy Preserving Collaborative k-means Clustering Protocol Supporting both Storage and Computation Outsourcing[C]//International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2018, 447-460.

López-Alt, A., Tromer, E., & Vaikuntanathan, V. (2012). On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption[C]//Proceedings of the forty-fourth annual ACM symposium on Theory of computing. ACM, 2012: 1219-1234.

Naeem, M., & Asghar, S. (1999). "KEGG Metabolic Reaction Network Data Set," The UCI KDD Archive, 1999, http://archive.ics.uci.edu/ml/datasets/KEGG+Metabolic+Reaction+Network+(ndirected).

Ostrovsky, R., Rabani, Y., Schulman, L. J., & Swamy, C. (2006). The effectiveness of Lloyd-type methods for the k-means problem[C]//Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on. IEEE, 2006: 165-176.

Paillier P. (1999). Public-key cryptosystems based on composite degree residuosity classes[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1999: 223-238.

Peter, A., Tews, E., & Katzenbeisser, S. (2013). Efficiently outsourcing multiparty computation under multiple keys[J]. *IEEE transactions on information forensics and security, 8*(12), 2046-2058.

Rao, F. Y., Samanthula, B. K., Bertino, E., & Yi, X. (2015). Privacy-preserving and outsourced multi-user k-means clustering[C]//Collaboration and Internet Computing (CIC), 2015 IEEE Conference on. IEEE, 2015, 80-89.

Rong, H., Wang, H., Liu, J., Hao, J., & Xian, M. (2017). Privacy-Preserving-Means Clustering under Multiowner Setting in Distributed Cloud Environments[J]. *Security and Communication Networks,* 2017.

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

Shafagh, H., Hithnawi A., Burkhalter L., Fischli P., & Duquennoy S. (2017). Secure Sharing of Partially Homomorphic Encrypted IoT Data[C]//Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems. ACM, 2017: 29.

The GNU MP Bignum Library. http://gmplib.org/.

Van Dijk, M., & Juels, A. (2010). On the impossibility of cryptography alone for privacy-preserving cloud computing[J]. *HotSec, 10*, 1-8.

Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 24-43.

Wang, B., Li, M., Chow, S. S. M., & Li, H. (2014). A tale of two clouds: Computing on data encrypted under multiple keys[C]//Communications and Network Security (CNS), 2014 IEEE Conference on. IEEE, 337-345.

Xing, K., Hu, C., Yu, J., Cheng, X., & Zhang, F. (2017). Mutual Privacy Preserving $ k $-Means Clustering in Social Participatory Sensing[J]. *IEEE Transactions on Industrial Informatics, 13*(4), 2066-2076.

Youn, T. Y., Park, Y. H., Kim, C. H., & Lim, J. (2005). An efficient public key cryptosystem with a privacy enhanced double decryption mechanism. *International Workshop on Selected Areas in Cryptography.* Springer, Berlin, Heidelberg, 144-158.

Copyrights