SecEDMO: Enabling Efficient Data Mining with Strong Privacy Protection in Cloud Computing

Jiahui Wu, Nankun Mu, Xinyu Lei, Junqing Le, Di Zhang, and Xiaofeng Liao *Senior member, IEEE*

Abstract—Frequent itemsets mining and association rules mining are among the top used algorithms in the area of data mining. Secure outsourcing of data mining tasks to the third-party cloud is an effective option for data owners. However, due to the untrust cloud and the distrust between data owners, the traditional algorithms which only work over plaintext should be re-considered to take security and privacy concerns into account. For example, each data owner may not be willing to disclose their own private data to others during the cooperative data mining process. The previous solutions are either not sufficiently secure or not efficient. Therefore, we propose a Secure and Efficient Data Mining Outsourcing (SecEDMO) scheme for secure outsourcing of frequent itemsets mining and association rules mining over the joint database (i.e., database aggregated from multiple data owners) in the paradigm of cloud computing. Based on our customized lightweight symmetric homomorphic encryption algorithm and a secure comparison algorithm, SecEDMO can ensure strong privacy protection and low data mining latency simultaneously. Moreover, the well-designed virtual transaction insertion algorithm can hide the information of the original database while still preserving the cloud's ability to perform data mining over the obfuscated data. By evaluation of a numerical experiment and theoretical comparisons, the correctness, security, and efficiency of SecEDMO are confirmed.

Index Terms—Data mining, frequent itemsets mining, association rules mining, privacy protection, cloud computing.

1 Introduction

1.1 Motivations

Discovering the association relationships among the frequent itemsets from a given dataset is called association rules mining, which leverages the frequent itemsets mining process as a sub-routine. Association rules mining (along with frequent itemsets mining) is one of the most important algorithms in the field of data mining. For example, the classical association rules mining algorithm (i.e., Apriori [1]) is identified as a top 3 algorithm in data mining [2]. It is widely utilized in many fields including market supervision and management [3], web browsing preference prediction [4], intrusion detection [5], health-care services [6], to just name a few. The traditional algorithms for frequent itemsets mining and association rules mining mainly consider how to mine on plaintext domain, so they lack security and privacy concerns. However, the privacy-preserving property is strongly required in many scenarios. One typical setting is that there are multiple data owners wishing to learn association rules or frequent itemsets from their joint data, which is outsourced and stored in the public cloud. The security problem becomes a critical concern in this service model. First, the data owners may be unwilling to disclose their own data to other owners. For example, some data owners' data may have commercial value and they want to keep secret. Second, the public cloud cannot be fully trusted. These third-party clouds may have financial incentives to collect or infer their customer sensitive information. Moreover, these public clouds may be compromised and all of the stored information maybe further leaked by hackers. Therefore, we focus on developing a scheme to enable privacy-preserving frequent itemsets mining and association rules mining with multiple data owners in the cloud computing model.

- *This work is supported by National Key Research and Development Program of China (Grant no. 2016YF-B0800601), National Natural Science Foundation of China (Grant no. 61472331, 61772434, 61403121, 61806169) and Fundamental Research Funds for the Central Universities (Grant no. XDJK2018D005). (Corresponding author: Liao Xiaofeng)*
- *J. Wu, N. Mu, J. Le, and D. Zhang are with the College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China, and Chongqing Key Laboratory of Nonlinear Circuits and Intelligent Information Processing. E-mail: wjh2015@email.swu.edu.cn; nankun.mu@qq.com; lejunqing@163.com; zhangdiii@163.com*
- *X. Lei is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA. E-mail: leixinyu@msu.edu*
- *X. Liao is with the college of computer, Chongqing University, Chongqing 400044, China. E-mail: xfliao@cqu.edu.cn*

1.2 Problem Formulation

• **Frequent Itemsets Mining & Association Rules Mining.** Frequent itemsets mining refers to finding frequent patterns (including itemsets, sequences, substructures) that occur frequently in data sets. Support[1] represents how frequently the pattern appears in the data sets. With a pre-defined adjustable count threshold (i.e., minimum support), the aim of the mining algorithm is to find all itemsets with the count/support greater than or equal to the minimum support. And the motivation of association rules mining is to find out the frequently occurred association relationships between frequent itemsets. Therefore, association rules mining algorithm mainly consists of two steps: (1) find all frequent itemsets from the dataset (by using the frequent itemsets mining); (2) generate association rules from these frequent itemsets.

More concretely, let I represent the collection of data items and D represent the database that consists of a set of transactions. In general, a rule has the form of $X \Rightarrow Y$, where $X, Y \subseteq I$. The rule $X \Rightarrow Y$ with support s, i.e., $(X \Rightarrow Y).\text{sp} = s$, implies the occurrence count that X and Y co-occur in the database. The rule $X \Rightarrow Y$ with confidence c, i.e., $(X \Rightarrow Y).\text{cf} = (X \cup Y).\text{sp}/Y.\text{sp} = c$, implies that the probability of Y's occurrence given X's occurrence is c. We denote the minimum support and the minimum confidence as T_s and T_c, respectively. A rule $X \Rightarrow Y$ with support no less than

1. To facilitate the subsequent operation in this paper, we denote the support of a pattern as its count number.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

T_s and confidence no less than T_c, is identified as an association rule. Note that the thresholds T_s and T_c are configurable by data owners.

• **Privacy-Preserving Data Mining.** We consider the privacy-preserving frequent itemsets mining and association rules mining problem by taking advantage of cloud computing. The cloud can achieve on-demand high-quality data mining and reduces storage cost at the same time. To achieve privacy-preserving data mining, each data owner needs to encrypt their data before outsourcing to the untrusted cloud. The cloud should be able to perform mining over the encrypted data and then return the mining results (in encrypted form) to the data owners for decryption.

• **Horizontally Partitioned Database.** We consider horizontally partitioned databases in this paper. The horizontally partitioned database means that data outsourced by different owners are all in the same format. The other type of database is vertically partitioned, which means that outsourced data of distinct data owners are in different formats. For example, consider that the data owners A and B have databases with items I = {bread, milk, egg} and the data owner C has a database with items I = {pearl, agate}, then we have the database aggregated by A and B is horizontally partitioned and the database aggregated by A and C is vertically partitioned.

1.3 Limitation of Prior Art

The previous solutions for secure frequent itemsets mining or association rules mining suffer from three limitations as described below.

- Some schemes [7]–[11] are built based on public-key homomorphic encryption (HE) algorithms (e.g., elliptic curve-based ElGamal, Paillier encryption). Since the current HE lacks efficiency, these schemes suffer from high running time.
- Some schemes perform privacy-preserving data mining over a number of distributed servers (e.g., [10], [12], [13]). These schemes have a large number of communication rounds.
- Some symmetric key encryption-based schemes (e.g., [14]) can achieve low pruning time and low communication cost. However, they have a weak threat model, and therefore, they possess poor anti-attack capabilities (as analyzed in [15]).

Therefore, we believe that there is a critical need to develop a secure and effective association rules mining scheme on multi-owner databases with high-efficiency in a strong threat model.

1.4 Technical Challenges

There are three technical challenges that SecEDMO shall deal with.

- The first challenge is how to realize strong privacy-preserving, low data mining latency, and low communication complexity simultaneously. It is challenging to strike a well-balanced trade-off between them.
- The second challenge is how to design an effective algorithm to obfuscate some data for a large-sized database while still preserving the cloud's ability to perform data mining over the obfuscated data.
- The third challenge is how to prevent privacy leakage from encrypted databases and mining results in a strong threat model. It is desirable that (1) each data owner should not deduce other owners' data; (2) the scheme should be able to deal with attacks from a corrupted owner and multiple collusive owners; (3) the scheme should be able to resist attacks from an outside adversary.

1.5 Main Contributions

In this paper, we design SecEDMO to address the above challenges. The comparison between SecEDMO and the previous schemes is illustrated in Figure 1, where the red area represents the optimal solution in both security and performance (i.e., an ideal solution). SecEDMO (in green area) is the closest scheme to the optimum compared with other schemes. It is shown that our SecEDMO strikes a better trade-off between security and performance than the prior art. Moreover, SecEDMO can be applied in other secure data computing solution such as secure data aggregation (presented in Appendix B). In summary, this paper makes the following contributions.

- We design a customized symmetric HE algorithm and a secure comparison algorithm to address the secure association rules mining problem. The designed algorithms ensure strong privacy preserving and low data mining latency.
- We propose a virtual transaction insertion algorithm to obfuscate the support of each item in the original database. The proposed algorithm can hide the information of the original database while still preserving the cloud's ability to perform data mining over the obfuscated data.
- We design the scheme in a strong threat model. Various countermeasures (including secret-key dividing, ciphertext characterizing and mining share) are adopted to resist different kinds of attacks from the data owners, the cloud, and the outside adversary.

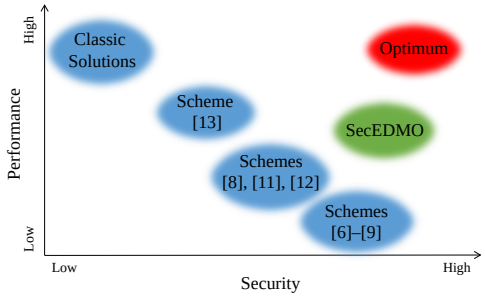

Fig. 1. Comparison between SecEDMO and the previous schemes in terms of performance and security.

1.6 Paper Organization

The remainder of this paper is organized as follows. Section 2 gives a formal description of our system model, threat model, and design goals. Section 3 introduces the preliminaries of the requested cryptographic knowledge. Section 4 presents the designs of HE, secure comparison, and virtual transaction insertion. In Section 5, the design of SecEDMO is elaborated. Section 6 performs correctness and security analysis of SecEDMO and compares its security with several other schemes. Section 7 presents performance evaluation of SecEDMO. Related work is described in Section 8, followed by conclusion in Section 9.

2 System Model, Threat Model, and Design goals

In this section, we introduce the adopted system model, threat model, and design goals of SecEDMO.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

2.1 System Model

Figure 2 shows the system model, which consists of three entities: an administration server, multiple data owners, and a cloud. To enable that distinct owners have distinct individual secret keys, a trusted administration is introduced to distribute keys to each owner separately. The administration server responds to a request of the concerted data owners to distribute secret keys and parameters. After obtaining secret keys and parameters, each data owner separately encrypts his own database and then outsources the encrypted database to the cloud. The cloud connects the received horizontally partitioned databases from multiple data owners and then mines frequent itemsets or association rules according to the requests of data owners. The returned mining results consist of frequent itemsets, association rules, and threshold comparisons' results. All of the returned results are ciphertexts in encrypted form. Upon receiving the returned ciphertexts from the cloud, the data owner decrypts them to obtain the corresponding mining results in plaintext.

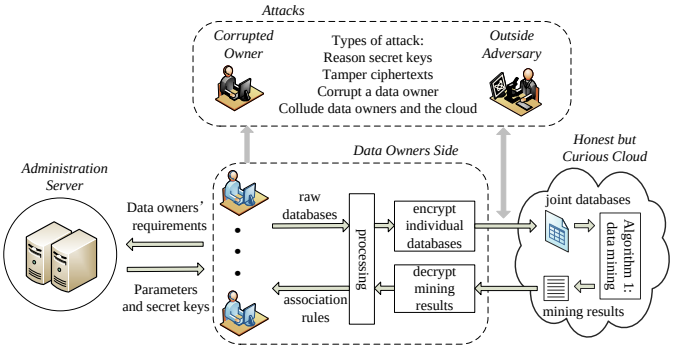

Fig. 2. System model of privacy-preserving data mining for the joint database.

2.2 Threat Model

We consider the possible threats of the system come from three entities: the cloud, the data owners, and the outside adversary, as depicted in Figure 2.

• **Cloud.** The cloud who performs data mining task over encrypted joint databases is assumed to be honest-but-curious [8], [14], [16]. That is, the cloud follows the designed scheme honestly but attempts to deduce the private information of encrypted databases, frequent itemsets, and association rules. Such an assumption of untrusted cloud server better approaches the real-world applications because most of the cloud service providers are commercial third parties. We have also noted that there is an alternative security model in which the cloud is assumed to be trusted [17].

• **Data Owners.** We consider two types of data owners. The first type of data owner is treated as a collaborative but curious entity. They honestly follow the scheme but are curious about other owners' private information. The second type of data owner is assumed to be corrupted. By corrupted data owners we mean that the adversary may deviate from the scheme specifications and attempt to deduce the other owners' private information (e.g., tamper the other owners' ciphertexts).

• **Outside Adversary.** We consider the outside adversary who has the capabilities of (1) eavesdropping and tampering the ciphertexts in the communication channel between the data owners and the cloud, (2) corrupting a data owner, and (3) colluding with the corrupted data owners and the cloud. With the above three capabilities, the adversaries can launch a variety of attacks (e.g., attacks on reasoning secret keys, tampering ciphertexts, corrupting a data owner, and colluding with data owners or the cloud). These attacks are described and analyzed in detail in Section 6.

2.3 Design Goals

SecEDMO should achieve the following design goals.

- *Correctness.* If both data owners and cloud follow the SecEDMO honestly, the data mining tasks can be indeed performed by the cloud, and the data owners can get the correct mining results.
- *Security.* (1) Data owners' databases should be confidential to one another. (2) The mining results should be secret to the cloud. (3) SecEDMO should resist attacks from an outside adversary with various capabilities.
- *Efficiency.* SecEDMO should have low computational complexity, communication traffic, storage cost, and end-to-end latency to guarantee the efficiency of secure association rules mining.
- *Accommodate lightweight client.* SecEDMO should keep as less as workload on the client side (i.e., the data owner side) and accommodate lightweight client.

3 Preliminaries

In this section, we introduce the background knowledge of homomorphic encryption, virtual transactions, and mathematical notations.

• **Homomorphic Encryption.** Homomorphic encryption was first proposed by Rivest *et al.*, closely after the creation of RSA [18], [19]. HE satisfies the homomorphism property so that the algebraic operations on the plaintext can be performed on the corresponding ciphertext, which prevents the plaintext from being cracked. In general, an HE algorithm includes four algorithms: (`GenKey`, `Enc`, `Dec`, `Eva`) [20], which represent the key generation algorithm, the encryption algorithm, the decryption algorithm, and the evaluation algorithm, respectively.

We denote an operation on ciphertexts (`Eva` algorithm) as F, whereas a required operation on plaintext as f. `Enc` is homomorphic, if $F(c_1, \cdots, c_n) = \texttt{Enc}_K(f(m_1, \cdots, m_n))$, where K is a set of secret keys generated by `GenKey`, and m_i, c_i $(i = 1, \cdots, n)$ are plaintexts and ciphertexts respectively. Its underlying meaning is: f is expected to act on owners' plaintexts but instead of operating f, the corresponding ciphertexts are outsourced to an untrusted third party and implement $F(c_1, \cdots, c_n)$. Then $f(m_1, \cdots, m_n)$ can be obtained by decrypting $F(c_1, \cdots, c_n)$ – the third party works for the owners, but knows nothing about the operative plaintexts.

• **Virtual Transactions.** Virtual transactions, also known as fake or fictitious transactions in [14], [21], are extra and nonexistent transactions inserted into the original database. They are often used to obfuscate the privacy of the real database when the database tends to be outsourced to the cloud. Conversely, without insertion of virtual transactions, the original database can be easily attacked by the cloud using the statistical attack although it is encrypted by a 1-1 substitution cipher. In general, after being inserted with virtual transactions, the database adds a tag attribute for each transaction which records the real transaction as 1 and the virtual transaction as 0. Table 1 gives an example of a database inserted with virtual transactions. The tag values of T_1, T_3 are 1 and T_2, T_4 are 0, that is, T_1, T_3 are real transactions and T_2, T_4 are virtual transactions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

TABLE 1. Database with inserted virtual transactions.

TID	bread	milk	apple	banana	Tag
T_1	0	0	1	0	1
T_2	1	1	0	1	0
T_3	0	0	1	1	1
T_4	0	1	1	1	0

• **Mathematical Notations.** Table 2 summarizes mathematical notations and their semantic meanings used throughout this paper.

TABLE 2. The notations and their semantic meanings.

Notations	Meanings
DB	Database
I	A set of items (item labels) in DB
T_i	The ith transaction in DB, $i \geq 1$
TID	Transaction ID
$X \cup Y$	The union of itemsets X and Y
$X \Rightarrow Y$	A rule that X and Y co-occur in DB
X.sp	The support of the itemset X
$(X \Rightarrow Y)$.cf	The confidence of the rule $X \Rightarrow Y$
l_i	The number of T in the ith database, $1 \leq i \leq n$
n_t, n_v	The number of real and virtual T in DB, respectively
n	The number of data owners
n_I	The number of items in I
N_T, n_T	The number of total and real T in joint database, respectively
λ	A security parameter
k, d, ε, N	Positive integers
r', r	Random positive integers
$\dot{r}$	A random real in [0,1]
$\mathbb{F}_p$	A finite field of integers modulo p
$\mathbb{Z}_p^*$	A cyclic group under the multiplicative operation modulo p
e_i	The element in $\mathbb{F}_{p_1}$, $1 \leq i \leq n$
K	Key set
N_d	The maximum denominator of T_c
GenKey	Key generation algorithm
Enc, Dec	Encryption and decryption algorithms
Eva	Evaluation algorithm
HE	Homomorphic encryption
$f(\cdot), F(\cdot)$	Functions operated on plaintexts and ciphertexts, respectively
$H(\cdot)$	Hash function

4 Proposed HE, Secure Comparison, and Virtual Transaction Insertion Algorithms

In this section, we present a symmetric HE, a secure comparison, and a virtual transaction insertion algorithms for data privacy protection. The three algorithms are the subroutines of SecEDMO and are designed as follows.

4.1 Symmetric HE Algorithm Design

4.1.1 HE Design Goals

Before designing HE, we clarify its design goals corresponding to secure association rules mining.

- In frequent itemsets and association rules mining, it is necessary to calculate supports and confidences of transactions. For privacy-preserving consideration, the confidentiality of supports and confidences should be ensured. Supports and confidences are worked out mainly using addition, subtraction and threshold comparison. Therefore, we design an HE algorithm to satisfy homomorphism of addition, subtraction, and scalar multiplication at the same time to secure the operation in calculating supports and confidences.
- Almost all privacy-preserving data computation based on HE cannot resist the chosen attacks (a class of attacks on reasoning secret keys) due to properties of HE, thus a secure HE against the chosen attacks should be designed.
- In order to prevent ciphertexts encrypted by HE from being tampered arbitrarily, these ciphertexts need to be characterized, so that an invalid modification on ciphertexts will lead to a failure in decrypting and sound an intrusion warning.
- Multi-owner joint data mining is to mine target information in the joint database of all the agreed owners, so all the owners' databases should be included in the mining, otherwise, the mining results will be biased. Therefore, a scheme that detects if all the joined databases are included before mining should be considered.
- In the process of data mining, in order to prevent attacks from a corrupted owner and multiple colluded participants (including the data owners and the cloud), the mining processing should be performed correctly only when all the owners' databases are combined together. Thus a corrupted owner only or several colluded participants cannot obtain the plaintexts of mining results.

4.1.2 HE Design

An HE algorithm generally includes four algorithms: (GenKey, Enc, Dec, Eva). Each of them in our designed HE is given below.

- $\texttt{GenKey}(\lambda, l_1, \cdots, l_n)$: Generate secret keys and parameters of n parties. λ is a security parameter and $l_i (i = 1, \cdots, n)$ represents the number of transactions provided by the ith data owner.
  - s1: Generate three big primes p_1, p_2, p_3 and a small positive integer d. p_3 depends on the security parameter. $p_1 \ll p_2 \ll p_3$, and the scope of the three large primes is further described in Section 4.3.
  - s2: Randomly choose a number $s \in \mathbb{Z}_{p_3}^*$, an expansive factor $k \ll p_1$, and n positive integers $e_1, e_2, \cdots, e_n \in \mathbb{F}_{p_1}$ with $\sum_{i=1}^{n} l_i e_i \bmod p_1 = 0$.
  - s3: p_1, p_3, d are distributed as public parameters, (s, p_2, k) as public secret keys across the n data owners, and $e_i (i = 1, \cdots, n)$ as an individual private key of the ith owners.
- $\texttt{Enc}_K(T)$: HE algorithm for a constant value $U \in \mathbb{F}_{p_2}$.
  - s1: Select a random number r.
  - s2: Generate the corresponding ciphertext C_U:

$$C_U = \texttt{Enc}_K(U) = s^d(rp_2 + kU) \bmod p_3. \tag{1}$$

- $\texttt{Enc}_K^{e_i}(m_{ij})$: HE algorithm for m_{ij} which is the jth data of the ith data owner $(j = 1, \cdots, l_i; i = 1, \cdots, n)$.
  - s1: Check if the plaintext $km_{ij} \in \mathbb{F}_{p_1}$.
  - s2: Select a random number r_{ij}.
  - s3: Generate the ciphertext of m_{ij}:

$$c_{ij} = \texttt{Enc}_K^{e_i}(m_{ij}) = s^d(r_{ij}p_2 + km_{ij} + e_i) \bmod p_3. \tag{2}$$

- $\texttt{Eva}(c_1, \cdots, c_n)$: Evaluation on owners' ciphertexts.
  - s1: Evaluate all the ith owner's ciphertexts $c_{ij} (j = 1, \cdots, l_i)$ as

$$\begin{aligned} c_i = \texttt{Eva}(c_{i1}, \cdots, c_{il_i}) &= \sum_{j=1}^{l_i} \texttt{Enc}_K^{e_i}(m_{ij}) \\ &\triangleq s^d(r_i p_2 + km_i + l_i e_i) \bmod p_3. \end{aligned} \tag{3}$$

  - s2: Evaluate all the n owners' ciphertexts $c_i (i = 1, \cdots, n)$ as:

$$\begin{aligned} c = \texttt{Eva}(c_1, \cdots, c_n) &= \sum_{i=1}^{n} s^d(r_i p_2 + km_i + l_i e_i) \\ &\triangleq s^d(rp_2 + km + \varepsilon p_1), \end{aligned} \tag{4}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

where ε is an integer.

Equations (3) and (4) present the addition homomorphism property of the designed HE. Subtraction and scalar multiplication homomorphism of the HE are presented as Equations (5) and (6) respectively.

$$\begin{aligned}&((c_1 - \cdots - c_n) - 2l_1Z_1) \bmod p_3\\ &= s^d((r_1p_2 + km_1 + l_1e_1) - \cdots - (r_np_2 + km_n + l_ne_n)\\ &\quad - (2r'p_2 + 2l_1e_1)) \bmod p_3\\ &\triangleq s^d(rp_2 + k(m_1 - \cdots - m_n) - \varepsilon p_1) \bmod p_3,\end{aligned} \tag{5}$$

where $Z_1 = \mathtt{Enc}_K^{e_1}(0)$ is a public available value issued by the owner of number 1.

$$\begin{aligned}&(N \times C_U) \bmod p_3\\ &= (N \times s^d(rp_2 + kU) \bmod p_3) \bmod p_3\\ &\triangleq s^d(rp_2 + kNU) \bmod p_3,\end{aligned} \tag{6}$$

where N is a positive integer.

- $\mathtt{Dec}_K(C_U)$: Homomorphic decryption algorithm to decrypt ciphertexts of constant values.
  s1: Compute $d_1 = (C_U \times s^{-d} \bmod p_3) \bmod p_2$.
  s2: If $d_1|k$ (i.e., d_1 can be divisible by k), the plaintext of C_U can be obtained by calculating $U = \mathtt{Dec}_K(C_U) = d_1/k$.
- $\mathtt{Dec}_K(c)$: Homomorphic decryption algorithm for the evaluated value c in Equation (4).
  s1: Compute d_1 as follows:

$$\begin{aligned}d_1 &= ((c \times s^{-d} \bmod p_3) \bmod p_2) \bmod p_1\\ &= ((rp_2 + km + \varepsilon p_1) \bmod p_2) \bmod p_1 = km.\end{aligned}$$

  s2: If $d_1|k$, the plaintext of c can be obtained by calculating $m = \mathtt{Dec}_K(c) = d_1/k$.

In the designed HE, we only decrypt the ciphertexts whose original plaintexts are constant values (i.e., plaintexts are encrypted by Equation (1)) or who are evaluated values of all the n owners' ciphertexts (i.e., ciphertexts are evaluated by Equation (4)). Therefore c_i, the evaluated ciphertext of the ith owner, cannot be decrypted. Such design we call mining sharing, analogous to secret sharing, is to ensure that the ciphertexts can be decrypted only when all owners' individual ciphertexts are combined together and individual ciphertexts are of no use on their own. In association rules mining, supports of itemsets are counted as HE evaluated values of all n owners' ciphertexts c, thus, the decryption of these evaluated values should be effective.

4.1.3 Discussion

We next discuss how the customized HE algorithm meets the design goals.

- Before encrypting, the plaintexts are characterized by multiplying an expansion factor k, so that the valid ciphertext margin takes up only part of the numerical space. This measure can prevent arbitrary tampering of ciphertexts.
- The confidentiality of supports and confidences in cloud-aided mining phase are ensured, because the designed HE owns properties of homomorphic addition and scalar multiplication at the same time.
- A mining sharing method analogous to secret sharing is introduced to (1) detect if all the agreed databases are included before mining and (2) prevent corruption and collusion attacks. It ensures that distinct data owners have distinct individual secret keys (e_i for the ith owner), and the encrypted mining results can be correctly decrypted only when the joint database is stitched by databases of all involved owners. Otherwise, an intrusion warning is sounded. Through this, a corrupted owner does not know the other owners' secret keys and thus cannot obtain privacy from the ciphertexts, and a joint database of partial data owners cannot be successfully mined by the cloud. Therefore, mining in all the agreed databases can be proceeded normally to get the correct results in response to the joint mining, while mining in partial databases cannot be implemented to prevent the partial owners from deriving the other owners' privacy.
- The designed scheme can resist attacks on reasoning secret keys, because the underlying HE algorithm is a probabilistic encryption algorithm, and distinct owners have distinct individual secret keys.

The detailed analysis of how the designed HE algorithm resist the distinct types of attacks is presented in Section 6.

4.2 Secure Comparison Algorithm Design

Based on the above HE algorithm, we design a secure comparison algorithm. In general solutions of association rules mining, it is necessary to compare supports and confidences with their corresponding thresholds. In our privacy-preserving solutions, the cloud performs the above comparisons in the joint database encrypted by the HE and is required to be unknown with the supports, the confidences, and the comparison results (i.e., the mining results). The secure comparison algorithm works as follows.

- First, the cloud receives a ciphertext $C_V = \mathtt{Enc}_K(V)$ from data owners, where V is a big number ($V > N_T$ and $V + N_T < p_1$, where N_T is the number of transactions in the joint database) provided by the trusted administration.
- Second, the cloud calculates $C = (C_V + c_\alpha - c_\beta)$ as the comparison result, where c_α and c_β are two ciphertexts (encrypted by Equation (4)) to be compared, and $\alpha, \beta \le N_T < V$ are the original plaintexts.
- Third, the comparison result (i.e., the mining result) in plain form is $\alpha - \beta = (C \times s^{-d} \bmod p_3 \bmod p_2 \bmod p_1)|k - V$, calculated by the data owners.

Correctness. C_V is in the form of Equation (1). Two ciphertexts c_α, c_β are in the form of Equation (4). Then the comparison result is

$$C = (C_V + c_\alpha - c_\beta) = s^d(rp_2 + k(V + \alpha - \beta) + \varepsilon p_1),$$

and its decrypted form is

$$\begin{aligned}&(C \times s^{-d} \bmod p_3 \bmod p_2 \bmod p_1)|k - V\\ &= (k(V + \alpha - \beta))|k - V\\ &= \alpha - \beta.\end{aligned}$$

A concrete example of the comparison algorithm utilized in frequent itemsets and association rules mining is shown in Section 6.

4.3 Discussion of HE Parameters

Proper HE parameters are needed to ensure that the encrypted comparison results (mining results) of the joint database can be successfully decrypted by data owners. According to the construction of the HE algorithm and the comparison algorithm, the HE parameters p_1, p_2, p_3 can be limited by the following three requests. First, the maximum evaluation value $\sum_{\mathrm{TID}} \mathtt{Enc}(m)$ should be decrypted successfully, where $\sum_{\mathrm{TID}}$ aggregates all transactions in a joint database. Second, since V is the maximum plaintext,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

$\mathrm{Enc}(V)$ should be decrypted successfully. Third, to effect the secure comparison, the scalar multiplication $N_d \sum_{\mathrm{TID}} \mathrm{Enc}(m)$ should be decrypted successfully, where ${N_d}^2$ is an integer.

All the above three requests can be satisfied when $\mathrm{Enc}(V) + N_d \sum_{\mathrm{TID}} \mathrm{Enc}(m)$ can be successfully decrypted.

$$\begin{aligned} &\mathrm{Enc}(V) + N_d \sum\nolimits_{\mathrm{TID}} \mathrm{Enc}(m) \\ &= s^d(r' p_2 + kV) + N_d \sum\nolimits_{\mathrm{TID}} s^d(r p_2 + e_i + km) \\ &= s^d(r' p_2 + kV + N_d(p_2 \sum r + \sum\nolimits_{i=1}^{n} l_i e_i + k \sum m)) \\ &< s^d((N_d N_T + 1) r_{max} p_2 + N_T N_d p_1 + k(N_d N_T + V)). \end{aligned} \tag{7}$$

Equation (7) can be successfully decrypted when the following inequalities hold.

$$\begin{aligned} &p_1 > k(N_d N_T + V), \\ &p_2 > N_T N_d p_1 + k(N_d N_T + V), \\ &p_3 > (N_d N_T + 1) r_{max} p_2 + N_T N_d p_1 + k(N_d N_T + V). \end{aligned} \tag{8}$$

Since $V > N_T$, Equation (8) can be simplified as

$$\begin{aligned} &p_1 > kV(N_d + 1), \\ &p_2 > V N_d p_1, \\ &p_3 > r_{max} V N_d p_2. \end{aligned} \tag{9}$$

Thus, the parameters of HE should meet Equation (9) to ensure that comparison results (mining results) in our scheme can be successfully decrypted.

4.4 Virtual Transaction Insertion Algorithm Design

4.4.1 Design Goals

Before outsourcing individual databases to the cloud, data owners insert virtual transactions into their raw databases to obfuscate the real transactions. Each item in the inserted database should be uniformly distributed, and the order of the transactions should be confused so that the plaintext database itself can conceal real information of all transactions in the original database.

4.4.2 Insertion Algorithm

We assume that the number of the raw database's transactions and the inserted transactions are n_t and n_v respectively, and a random decimal is $\dot{r} \in [0, 1]$. A transaction is known to contain multiple items, and each item has a value of 1 or 0. This algorithm generates each item in a virtual transaction independently, and the generation of an item X in the ith virtual transaction contains two steps: First, count the support of X and denote it as X.sp; second, calculate the virtual value of X in the nth virtual transaction by the following equations.

If $i < n_v$,

$$X_i = \begin{cases} 0, \dot{r} < \frac{X.\mathrm{sp}}{n_t} \\ 1, \dot{r} \geq \frac{X.\mathrm{sp}}{n_t} \end{cases} \tag{10}$$

otherwise,

$$X_i = \begin{cases} 0, \dot{r} < 0.5 \\ 1, \dot{r} \geq 0.5 \end{cases} \tag{11}$$

Each item value in the ith transaction is generated according to Equations (10) and (11). If the number of items in the generated transaction is greater than the maximum item number n_I in the original database, the transaction is then randomly divided into several sub-transactions with item numbers not exceeding n_I. We generate all n_v transactions by repeating this process. This insertion algorithm is designed to balance the support of each item as far as possible, and therefore the algorithm can obfuscate items' values of transactions in individual databases. After inserting with the virtual transactions, the algorithm confuses TIDs by using a pseudo-random number sequence as in [22]. The randomness of each item in a database including virtual transactions is analyzed in Section 7.

2. N_d in this scheme represents the maximum denominator of $T_c = \frac{T_{c_1}}{T_{c_2}}$, where T_{c_1}, T_{c_2} are integers. $N_d \geq max(T_{c_2})$ is usually set to be 100 for that the maximum denominator of confidence is 100.

5 SecEDMO Scheme Design

Referring to the mining model of a database with the virtual transaction in [14], SecEDMO is presented using the HE algorithm, secure comparison algorithm, and virtual transaction insertion algorithm proposed in Section 4 as building blocks.

5.1 Overview of SecEDMO

Figure 2 illustrates the proposed process of multi-user joint association rules mining. The process consists of three principal parts, i.e., an administration server, data owners and a cloud server, which mainly focus on parameter setting, data privacy processing, and data mining respectively. The entire process of SecEDMO contains five stages:

- Owner request stage: n data owners collate their data, insert virtual transactions and denote the number of transactions in their processed database as $l_i (i = 1, \cdots, n)$ separately, prior to requesting the administration server.
- Key generation stage: The administration server receives the request from data owners and executes $\mathrm{GenKey}(\lambda, l_1, \cdots, l_n)$ algorithm, represented in Section 4 to breed encryption keys and parameters.
- Data privacy preserving stage: After receiving secret keys and parameters from the administration server, the data owners encrypt item labels using a typical hash function like SHA-X [23], tag the virtual transactions in the joint database with 0 and the real with 1, and encrypt the tag values with HE (the encrypted tag values are denoted as ETVs). Section 5.2 introduces the detail of the data privacy preserving.
- Secure data mining stage: The cloud receives the owners' encrypted databases and then mines association rules using Algorithm 1 to output the mining results to the owners afterward.
- Result acquisition stage: The owners decrypt the mining results to obtain the ultimate plaintexts of frequent itemsets or association rules.

Owner request and key generation stages are the preparation of privacy-preserving data mining. Next, we elaborate on the processing of our privacy-preserving data mining, i.e., data privacy preserving and secure data mining.

5.2 Data Privacy Preserving

Data in our horizontally partitioned databases include finite item labels and transaction contents which record whether the items occur or not. In order to protect the privacy of the transaction contents and ensure the correctness of the data mining, hash function, virtual transaction, and the HE algorithm are used for the following reasons.

(1) *Replace item labels with their hash values:* Item labels in all the horizontally partitioned databases are the same, and can

be known to anyone (e.g., item labels in supermarkets are their commodity codes and are public). We replace them with their hash values to conceal the content of items in databases.

(2) *Obfuscate real transactions by inserting virtual transaction:* The contents of the real transactions are 1 or 0, which means a certain item occurs or not. To prevent adversaries from knowing the real contents, some virtual transactions are inserted to obfuscate the raw database, so that the adversaries cannot differentiate whether a certain transaction is real or not.

(3) *Encrypt tag values with HE:* The adversaries cannot tell which are the real transactions in the obfuscated database, and neither can the cloud. This makes it impossible for the cloud to mine the correct results based on real databases. To address this problem, two tag values (0 and 1) are used to differentiate the virtual and real transactions respectively, so that the cloud can eliminate the virtual one when executing data mining. But the tag values of the transactions should be unknown to the adversaries, thus the tag values are encrypted by the HE. Therefore, the encrypted tag values can not only resist attacks from the adversaries but also ensure that the cloud can directly calculate the correct mining results (in their encrypted form) without distinguishing which transactions are real.

5.3 Secure Data Mining

5.3.1 Data Mining on the Cloud

Algorithm 1 presents the general process of secure mining. Steps 2-4 are secure frequent itemsets mining and steps 5-6 are secure association rules mining.

In Algorithm 1, data owners provide their own threshold T_d. As the support threshold is encrypted before sending to the cloud, the cloud has to retain all comparison results between each itemset's support and the $\mathtt{Enc}(T_s)$. In order to receive as little unsatisfactory comparison results from the cloud as possible, data owners send their own threshold $T_d \le T_s$ to the cloud to cut off partially undesired results. The confidence threshold is $T_c = T_{c_1}/T_{c_2}$, where T_{c_1}, T_{c_2} are integers and $T_{c_2} \le N_d$. The maximum value of T_{c_2} is 100.

Algorithm 1 Secure association rules mining in cloud

Input: All n data owners' processed databases, $\mathtt{Enc}(T_s)$, T_{c_1}, T_{c_2}, T_d.

Output: Encrypted mining results RFIs/ARs, ECSs and ECCs.

1: Stitch all of the databases horizontally to form a joint database.

2: Calculate all frequent itemsets in the joint database with the support T_d and denote the set of these itemsets as FIs which contains all of the frequent itemsets and partial of the infrequent itemsets.

3: Obtain all of the real frequent itemsets in FIs using Equation (12) and denote them as a set RFIs.

4: Compare frequent itemsets in RFIs with the encrypted support $\mathtt{Enc}(T_s)$ using Equation (13) and denote the comparison results as ECSs.

5: Calculate all the association rules in RFIs and denote them as a set ARs which contains all real association rules and partial of false rules.

6: Compare all association rules in ARs with $T_c = T_{c_1}/T_{c_2}$ using Equation (14) and denote the comparison results as ECCs.

$$\begin{aligned}&\mathtt{Enc}((H(X_1),\cdots,H(X_t)).\mathrm{sp})\\&=\sum\nolimits_{\mathrm{DB=A}}^{\mathrm{B}}\mathrm{ETV}(H(X_1)==1\&\cdots\&H(X_t)==1)\\&\quad+\sum\nolimits_{\mathrm{A}}(H(X_1)==0|\cdots|H(X_t)==0)\cdot Z_{\mathrm{A}}\\&\quad+\sum\nolimits_{\mathrm{B}}(H(X_1)==0|\cdots|H(X_t)==0)\cdot Z_{\mathrm{B}},\end{aligned}\tag{12}$$

where & and | represent logic "and" and "or" operators respectively. Considering an itemset $X \in$ RFIs, its encrypted comparison result of support is calculated as

$$\begin{aligned}\mathrm{ECSs}&=\mathtt{Enc}(V+X.\mathrm{sp}-T_s)\\&=(\mathtt{Enc}(V)+\mathtt{Enc}(X.\mathrm{sp}))-\mathtt{Enc}(T_s))\bmod p_3.\end{aligned}\tag{13}$$

For the rule $H(X) \Rightarrow H(Y)$, the cloud computes its encrypted comparison result of confidence (ECCs) as

$$\begin{aligned}&T_{c_1}C_V-(T_{c_1}H(X).\mathrm{sp}-T_{c_2}(H(X)\cup H(Y)).\mathrm{sp})+T_{c_1}Z_e\\&=T_{c_1}s^d((r_1p_2+kV)-(r_2p_2+kX.\mathrm{sp}+\varepsilon_1p_1))\bmod p_3\\&\quad+T_{c_2}s^d(r_3p_2+k(X\cup Y).\mathrm{sp}+\varepsilon_2p_1))\bmod p_3\\&\quad+T_{c_1}s^d(r_4p_2+N_Tp_1)\bmod p_3\\&\triangleq s^d(rp_2+k(T_{c_1}(V-X.\mathrm{sp})+T_{c_2}(X\cup Y).\mathrm{sp})+\varepsilon p_1)\\&\quad\bmod p_3,\end{aligned}\tag{14}$$

where $\varepsilon = T_{c_1}\varepsilon_1 + T_{c_2}\varepsilon_2$.

5.3.2 Results Acquisition by the Data Owners

After receiving the encrypted mining outputs from the cloud (outputs in Algorithm 1), the owners process these ciphertexts to acquire real frequent itemsets and association rules.

• **Frequent Itemsets Acquisition.** The data owners decrypt the ECSs and determine whether inequality $\mathtt{Dec}(\mathrm{ECSs})-V \ge 0$ holds. If so, itemsets in RFIs are encrypted real frequent itemsets, which are then decrypted to recover the real plaintext frequent itemsets.

• **Association Rules Acquisition.** For an association rule $X \Rightarrow Y$ in ARs, the data owners firstly decrypt the support of $H(X)\cup H(Y)$ in ECSs to determine whether the itemset is a real frequent itemset. If so, the owners then decrypt the confidence comparison result of rule $H(X) \Rightarrow H(Y)$ in ECCs as

$$\begin{aligned}D&=(((\mathrm{ECC}\times s^{-d}\bmod p_3)\bmod p_2)\bmod p_1)|k-T_{c_1}V\\&=T_{c_2}(X\cup Y).\mathrm{sp}-T_{c_1}X.\mathrm{sp}.\end{aligned}$$

If

$$\begin{aligned}D\ge 0&\Rightarrow T_{c_2}(X\cup Y).\mathrm{sp}-T_{c_1}X.\mathrm{sp}\ge 0\\&\Rightarrow\frac{(X\cup Y).\mathrm{sp}}{X.\mathrm{sp}}\ge\frac{T_{c_1}}{T_{c_2}}=T_c,\end{aligned}$$

the rule $X \Rightarrow Y$ is a real association rule and then the owners decrypt its corresponding ciphertext in ARs to obtain its plaintext.

6 Correctness and Security Analysis

In this section, we first present attacks for secure data computation, and then under these attacks, we analyze the security of SecEDMO and some comparison schemes.

6.1 Correctness Analysis

We introduce a numerical example to present how SecEDMO scheme works correctly. The database A and B in Table 4 are respectively the original database of the data owners. The joint database in Table 5 is horizontal stitching of processed databases

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

TABLE 3. List of involved values in the numerical example

	Item	Owner A	Owner B
Owner request	Size of database	$l_1 = 6$	$l_2 = 8$
Key generation	Individual secret key	$e_1 = 1647$	$e_2 = 836$
	Public secret key	$s = 1245, p_2 = 248569, k = 10$	
	Public parameters	$p_1 = 1657, p_3 = 372853501, d = 2$	
	Constant values	$V = 15, N_d = 10$	
Database privacy preserving	Ciphertext of V	$C_V = s^d(p_2 + 15k) \bmod p_3 = 363001442$	
	Ciphertext of 0	$Z_A = \mathtt{Enc}_K(0)\|_{r=3} = 334409744$	$Z_B = \mathtt{Enc}_K(0)\|_{r=4} = 233179122$
	Random values	$r_1 = [8, 1, 10, 9, 6, 2]$	$r_2 = [5, 8, 2, 4, 1, 10, 6, 9]$
	ETVs of A	[256691452,88914610,129333085,14335643,353049319,219412302]	
	ETVs of B	[84041855, 118181680,80902530,326397664,323258339, 6323563,214539547,248679372]	
Threshold setting	Threshold	$T_d = 4, T_s = 4, T_c = 55/100$	
	Encrypted support	$C_{T_s} = \mathtt{Enc}_K(T_s)\|_{r=5} = 285835059$	

of A and B. The following process is to determine whether $X_1 \Rightarrow X_2$ in the joint database is an association rule.

(1) *Owner Request and Key Generation:* Owners A and B request an administration server to generate secret keys and parameters, whose assumed values are listed in Table 3.

(2) *Database Privacy Preserving:* The two owners respectively perform database privacy preserving with their own secret keys and parameters. The processed values of the databases are listed in Table 3. Z_A, Z_B are ciphertexts of digital 0 in Equation (12), encrypted by A and B respectively. Finally, the encrypted databases are received by the cloud and then are horizontally connected, as shown in Table 5.

(3) *Secure Data Mining:* The cloud finds out all frequent itemsets under support T_d. $H(X_1).\mathrm{sp} = 7, H(X_2).\mathrm{sp} = 8, H(X_1)H(X_2).\mathrm{sp} = 4$ are all in the selection. The cloud calculates the encrypted real support of $H(X_1), H(X_2)$ and $H(X_1) \cup H(X_2)$ as $C_{X_1.\mathrm{sp}} = (\sum_{\mathrm{TID}}(T_{X_1}ETV) + 3Z_A + 3Z_B) \bmod p3 = 320354872$ and $C_{X_2.\mathrm{sp}} = 12662363, C_{X_{12}.\mathrm{sp}} = 174356930$ respectively and compares the encrypted real support of $H(X_1) \cup H(X_2)$ with $\mathtt{Enc}(T_s)$ by calculating $C_{s_1} = (C_V + C_{X_{12}.\mathrm{sp}} - C_{T_s}) \bmod p_3$ to obtain the ECS of itemset $H(X_1) \cup H(X_2)$. The comparison of confidence between $X \Rightarrow Y$ and T_c is calculated as $C_{X_{12}}.\mathrm{cf} = (T_{c_1}C_V - (T_{c_1}H(X_1).\mathrm{sp} - T_{c_2}(H(X_1) \cup H(X_2)).\mathrm{sp}))) + T_{C_1}Z_e = 99364546$. The cloud then sends $H(X_1) \Rightarrow H(X_2), C_{X_{12}.\mathrm{sp}}$ and $C_{X_{12}.\mathrm{cf}}$ to the data owners.

(4) *Result Acquisition:* The data owners receive the comparison results and decrypt them to obtain $D_{X_{12}.\mathrm{sp}} - V = 0$ and $D_{X_{12}.\mathrm{cf}} - V = -1 < 0$ which mean that $X_1 \cup X_2$ is a frequent itemset but $X_1 \Rightarrow X_2$ is not an association rule.

TABLE 4. Original Databases of A and B.

DB	TID	X_1	X_2	X_3	X_4
A	T_1	0	0	1	0
	T_2	1	1	0	1
	T_3	0	0	1	1
	T_4	0	1	1	1
B	T_1	1	1	0	0
	T_2	0	1	0	1
	T_3	0	1	1	0
	T_4	1	0	1	1
	T_5	0	1	0	1

TABLE 5. The Joint Database of A and B.

DB	TID	$H(X_1)$	$H(X_2)$	$H(X_3)$	$H(X_4)$	ETV
A with Z_A	T_1	0	0	1	0	Enc(1)
	T_2	1	1	0	1	Enc(1)
	T_3	1	1	1	0	Enc(0)
	T_4	0	0	1	1	Enc(1)
	T_5	1	0	0	0	Enc(0)
	T_6	0	1	1	1	Enc(1)
B with Z_B	T_1	1	1	1	0	Enc(0)
	T_2	1	1	0	0	Enc(1)
	T_3	0	1	0	1	Enc(1)
	T_4	0	0	1	1	Enc(0)
	T_5	0	1	1	0	Enc(1)
	T_6	1	0	1	1	Enc(1)
	T_7	1	0	1	0	Enc(0)
	T_8	0	1	0	1	Enc(1)

6.2 Security Analysis

This subsection first gives various attacks and then analyses the security of the SecEDMO and the comparison schemes under these attacks.

6.2.1 Attack Categories

The various attacks which refer to Li *et al.*'s, Lin *et al.*'s and Vinodha *et al.*'s schemes [14], [24], [25], are adopted and modified to match our mining scheme. According to the threat model presented in Section 2, we divide these attacks into four different types as follows.

Type-I: Reasoning Secret Keys. An adversary attempts to reason secret keys (or equivalent keys) of an encryption scheme so that he can crack arbitrary encrypted data. Keys' security of an encryption scheme can be qualified by this type of attacks, which is made up of four attacks.

(1) *Ciphertext-only-attack.* An adversary has only ciphertexts and tries to deduce secret keys.
(2) *Known-plaintext-attack.* An adversary has some plaintexts along with their corresponding ciphertexts and tries to deduce secret keys.
(3) *Chosen-plaintext-attack.* The access of the encryption oracle is fetched by an adversary. Therefore he can encrypt certain plaintexts to obtain the corresponding ciphertexts before deducing secret keys.

(4) *Chosen-ciphertext-attack*. The access of the decryption oracle is fetched by an adversary. Therefore he can decrypt certain ciphertexts to obtain the corresponding plaintexts before deducing secret keys.

Type-II: Tampering Ciphertexts. Assuming an adversary does not possess the ability to reason secret keys, he then attempts to generate format-valid ciphertexts to tamper the original ciphertexts. Attacks in Type-II are just the response to the ability of ciphertexts tampering.

(1) *Unauthorized-mining-attack*. An adversary tries to generate format-valid ciphertexts to replace the cloud mining results; whereafter the forged results are received by data owners.
(2) *Malleability-attack*. An adversary tries to modify the contents of owners' ciphertexts (by adding a numeric value to a real ciphertext, appending a forged ciphertext to the raw encrypted database, or deleting partial ciphertexts in the encrypted database[3]), and ensure that all the modified ciphertexts are format-valid.

Type-III: Corrupting a Data Owner. With a corrupted data owner, an adversary knows the shared knowledge (e.g., shared secret keys) among all the data owners and thus owns stronger capabilities to tamper ciphertexts and obtain the other owners' private information. The following attacks are all in the premise that a data owner has been corrupted.

(1) *Unauthorized-encryption-attack*. With knowing the shared knowledge, the adversary attempts to imitate the other owners to encrypt their respective data into format-valid ciphertexts, so as to deviate the mining results.
(2) *Unauthorized-decryption-attack*. With knowing the shared knowledge, the adversary attempts to decrypt the other owners' ciphertext, so as to obtain their private information.

Type-IV: Colluding Data Owners and the Cloud. This type represents the attacks of collusion between the data owners and the cloud. The colluders are generally more powerful to deduce the other owners' secrets.

(1) *Owners-collusion-attack*. An adversary corrupts more than one data owner, and then these data owners collude with each other in an attempt to infer the secrets of the uncorrupted owners.
(2) *Cloud-owners-collusion-attack*. An adversary corrupts one or more data owners and the cloud, and then the owners and the cloud collude with each other in an attempt to infer the secrets of the uncorrupted owners. Note that the cloud is still only curious about the information and executes instructions sincerely.

6.2.2 Comparison under the Given Attacks

Based on the above-given attacks, we analyze the security of SecEDMO and some comparison schemes. The comparison schemes are three secure data mining schemes on horizontally partitioned databases (SDM_{HP}), one secure data mining scheme on virtually partitioned databases (SDM_{VP}), and one secure data aggregation scheme (SDA). SDM_{VP} and SDA whose underlying encryption algorithms are HE are chosen to compare with SecEDMO. We elaborate on the anti-attack performance of these schemes separately below.

3. As a special case, when all the ciphertexts in a database are deleted, this is equivalent to the owner of the database refuses to participate in the data mining.

• **Secure Data Mining Schemes on Horizontally Partitioned Databases (SDM_{HP})** [10], [12], [13]. These three schemes are all distributed multi-site association rules mining for horizontally partitioned data. Their underlying cryptosystems are asymmetric HE, Pohlig-Hellman encryption, and hash-based sum, respectively. HE generally cannot resist the chosen-ciphertext-attack [24]; Pohlig-Hellman encryption cannot resist the known-plaintext-attack (the proof is given in Appendix A); the hash-based sum is non-probabilistic and its security lies in mask values which can be easily deduced by comparing several plaintext/ciphertext pairs. Therefore, scheme in [10] is vulnerable to Type-I (4), and schemes in [12], [13] are vulnerable to attacks of Type-I (2-4). All the three schemes cannot resist attacks in Type-II and Type-III (1), because the forms of their ciphertexts are unrestricted, namely, any ciphertext generated by an adversary is format-valid; the three schemes are all distributed and own different secret keys, and thus, without considering other attack means, the three schemes are secure under Type-III (2). Collusion attacks in Type-IV in these three schemes are cooperative between sites. The scheme in [12] is resilient against the collusion attacks when the number of the non-collusive sites is more than one, or else, the secrets of the non-collusive site can be cracked. Schemes in [10], [13] cannot resist the collusion attacks for that a collusion between the $(i+1)$th site and the $(i-1)$th site can reveal the ith site's support value.
• **Secure Data Mining Scheme on Virtually Partitioned Databases (SDM_{VP})** [14]. This scheme is designed for mining association rule on vertically partitioned databases. The underlying encryption algorithm is a probabilistic symmetric HE and all the involved data owners share the same secret keys. No adversary can deduce secret keys with only ciphertexts, thus, the schemes are tight to the only-ciphertext-attack. The scheme is proved to be insecure [15] under the known-plaintext-attack and is breached utilizing only three arbitrary pairs of plaintext/ciphertext owing to some internal logic between its parameters. Thus, the scheme cannot resist the attacks of Type-I (2-4). The format of data owners' ciphertexts and encrypted mining results in this scheme is unrestricted, and any value produced by an adversary has the corresponding plaintext. Thus, the scheme cannot resist attacks in Type-II. When it comes to corruption attacks, presented in Type-III, an adversary owns only what the corrupted owner has, and can deduce the other owners' information because all owners share the same secret keys. Therefore, the scheme cannot resist Type-III attacks. In collision attacks in Type-IV, the scheme can resist the owners-collusion-attack when there are more than one non-collusive data owners. This makes sense: with accomplices' databases and the mining results, the adversary cannot deduce the other two or more non-collusive owners' information. But when the cloud collaborates with an owner, the scheme cannot sustain the security of other owners' information because the adversary has both secret keys and the other owners' encrypted individual database, so the scheme is vulnerable to the cloud-owners-collusion-attack.
• **Secure Data Aggregation Scheme (SDA)** [24]. This scheme is designed for concealed data aggregation in wireless sensor networks utilizing a probabilistic asymmetric HE. The scheme is tight to the attacks of Type-I (1-3) but is vulnerable to Type-I (4) for the same reason that [10] has. The scheme is secure to all of the attacks in Type-II and Type-III except Type-III (1) [24]. Two attacks in Type-IV in there respectively represent that more than one aggregators are compromised and one or more aggregators are compromised along with BS. Under these compromised scenarios,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

TABLE 6. Comparison of anti-attack performance of different schemes, where √ means well defense, ⊗ means partial defense, and × means no defense

Type	Attack	SDM_{HP} [10]	SDM_{HP} [12]	SDM_{HP} [13]	SDM_{VP} [14]	SDA [24]	SecEDMO
I	(1) Only-ciphertext-attack	√	√	√	√	√	√
	(2) Known-plaintext-attack	√	×	×	×	√	√
	(3) Chosen-plaintext-attack	√	×	×	×	√	√
	(4) Chosen-ciphertext-attack	×	×	×	×	×	⊗
II	(1) Unauthorized-mining-attack	×	×	×	×	√	√
	(2) Malleability-attack	×	×	×	×	√	√
III	(1) Unauthorized-encryption-attack	×	×	×	×	⊗	√
	(2) Unauthorized-decryption-attack	√	√	√	×	√	√
IV	(1) Owners-collusion-attack	×	⊗	×	⊗	×	⊗
	(2) Cloud-owners-collusion-attack				×	×	⊗

the scheme is insecure, since an aggregator can aggregate arbitrarily collected data, as long as the aggregation times are maintained.

• **Our SecEDMO.** The underlying HE of SecEDMO is to dispose of n owners' horizontally joint data, and its homomorphic properties are only adapted to the counted terms which are an addition of n owners' ETVs. Next, we describe the attack resistances of the HE algorithm under the given attacks.

- **Attacks in Type-I:** Suppose $c_i = \mathtt{Enc}(m_i) = s^d(rp_2 + km_i + e_i) \bmod p_3$ is a ciphertext from ith owner, where s, p_2, k are the public secret keys and e_i is the individual secret key of the ith owner, an adversary want to crack the ciphertext by reasoning secret keys (or equivalent keys). Since distinct owners have distinct individual secret keys, there are n distinct encryption/decryption oracles. We assume the adversary can access all the n encryption oracles. Since the underlying encryption algorithm of SecEDMO is probabilistic and is designed as a mining share form which ensures that a ciphertext from a single encrypted plaintext cannot be decrypted. Thus, the adversary cannot crack c_i although with n oracles' accesses. That is to say, SecEDMO can resist the chosen-plaintext-attack and weaker attacks such as the only/known-plaintext-attack. For the chosen-ciphertext-attack, the adversary can access the decryption oracle. Assuming that the adversary knows l_i, l_j (the number of transactions in the ith and the jth databases respectively) and a pair of plaintext/ciphertext of the jth owner as m_j, c_j, he calculates $c = l_i c_i + l_j c_j = s^d(rp_2 + k(l_i m_i + l_j m_j) + l_i e_i + l_j e_j) \bmod p_3$. The adversary then obtains m through decrypting c with the help of decryption oracle. If the number of total data owners is two, the adversary can obtain $m_i = (m - l_j m_j)|l_i$, or else, c cannot even be decrypted by the oracle and hence m_i cannot be cracked. Thus, SecEDMO can resist the chosen-ciphertext-attack partially.
- **Attacks in Type-II:** An adversary attempts to modify either the ciphertext of mining results in the cloud (unauthorized-mining-attack) or the encrypted tag values (ETVs) of data owners (malleability-attack). The mining results provided by the cloud are formed as $c_c = s^d(rp_2 + km + \varepsilon p_1)$, and can be decrypted by the data owners only when $(((c_c s^{-d} \bmod p_3) \bmod p_2) \bmod p_1)$ is exactly divisible by the secret key k. Thus, the modified mining results in Type-II (1) cannot be decrypted by the data owners, thus the data owners cannot be successfully tricked but warned of an intrusion. The owners' ciphertexts are formed as $c_d = s^d(rp_2 + km + e_i)$. Once any one of the ciphertexts is changed in data owners side, ε in c_c will take away part of km or leave the rest of itself to km. In either case, km is distorted, so it can not be decrypted by the owners, instead, it is a warning of intrusion. Therefore, Type-II (2) is also unadapted to cracking the HE algorithm.
- **Attacks in Type-III:** For attack Type-III (1), when a data owner is corrupted by an adversary, the owner's secret keys along with its database are compromised. The adversary attempts to forge a format-valid ciphertext to modify the data in the compromised database. If the forged ciphertext is appended to the database, the transaction number in the database will be changed which leads to $\sum l_i e_i \neq \varepsilon p_2$, then c_c cannot be decrypted but warns the owners of an intrusion. If the forged ciphertext is added to a raw ciphertext in the database, the condition is just the same due to the homomorphic addition property. Since distinct owners possess distinct individual secret keys, the compromised keys cannot help the adversary to reckon the other owners' secrets, that is to say, the scheme can resist Type-III (2-3).
- **Attacks in Type-IV:** An adversary in this type of attacks can only reckon the secret keys of one real data owner while the other $n - 1$ owners are all corrupted, otherwise, the scheme is tight to the owners-collusion-attack and the cloud-owners-collusion-attack.

Table 6 shows the results of the anti-attack performance of the analyzed schemes. In all of the above anti-attack analyses, we have assumed that the various types of adversaries are irrelevant to each other, for example, adversaries in Type-II, Type-III and Type-IV will not compare plaintext/ciphertext pairs to analyze secret keys, which is an attack in Type-I. However, adversaries who have the abilities to corrupt data owners and the cloud, definitely own pairs of plaintext/ciphertext and can make an attempt to reason secret keys. Therefore, the scheme that cannot resist the known-plaintext-attack in Table 6 will actually suffer the attacks in Type-III and Type-IV.

7 Performance Evaluation

This section presents performance evaluations of the proposed virtual transaction insertion and HE algorithms, compares SecEDMO with other schemes on computation and communication complexity and storage cost, and analyzes end-to-end latency from owner to cloud.

7.1 Randomness of the Inserted Virtual Transaction

In order to clearly demonstrate the randomness of virtual transactions, a small dataset "chess.dat" (prepared by Roberto Bayardo from the UCI datasets and PUMSB [26]) which has 3196

TABLE 7. Support ($\frac{\text{support}}{100\,n_t}\%$) of different items before and after inserting virtual transactions. The items "59, 47, 2, 58" are chosen from 3,196 items in the raw DB for that their percentages of the supports to the total transaction number are 0%, 20.0%, 47.8%, 100%, respectively.

n_v (δ) \ item	"59"	"47"	"2"	"58"
0 (0%)	1 (0.0%)	640 (20.0%)	1527 (47.8%)	3195 (100%)
792 (25.0%)	447 (11.3%)	1009 (25.5%)	1764 (44.5%)	3196 (80.7%)
1972 (57.2%)	1316 (25.6%)	1673 (32.4%)	2227 (43.1%)	3196 (61.8%)
2394 (74.9%)	1509 (27.1%)	1859 (33.4%)	2339 (42.0%)	3196 (57.0%)
3348 (104.8%)	1967 (30.1%)	2148 (32.8%)	2541 (38.8%)	3195 (48.8%)

transactions and 75 items is used as the experimental dataset DB. The transaction number of DB is n_t=3196. The number of inserted virtual transaction n_v is determined by data owners. Here, we insert different numbers of virtual transactions and then calculate the support of different items. Table 7 shows supports of items in the database before and after the database is inserted with virtual transactions.

In this table, $\delta = \frac{n_v}{100\,n_t}\%$ represents the percentage of the number of the inserted transactions to the raw transactions in DB. We select four items "59, 47, 2, 58" in the raw DB whose percentages of the supports to the total transaction number are $0\%, 20.0\%, 47.8\%, 100\%$, respectively. When we insert virtual transactions, the percentages are gradually balanced with the increasing of the inserted transaction number. Generally speaking, when the number of inserted transactions reaches that of the original transactions, the support of each item is approximately the same. TIDs in any database in our paper are dispensable, so the continuous 0 or 1 are not regarded as factors that affect the randomness of items. Therefore, our insertion algorithm is effective in obfuscating the raw items' support.

7.2 Computational Complexity, Storage Cost, and Communication Complexity

In SecEDMO, we assume there are n data owners, the number of items and the real transactions in the joint database are n_I and n_T respectively, the amount of the inserted transactions is δn_T, where δ is a small constant number, and the number of RFIs is n_F which contains all of the frequent itemsets and some of the unfrequent itemsets. Next, we analyze the complexities of SecEDMO and the comparison schemes.

7.2.1 Computational Complexity of SecEDMO

We present the computational complexity of data owners side and the cloud end in SecEDMO.

- For data owners, they insert virtual transactions, encrypt items and transaction tags using a hash function and the proposed HE algorithm respectively, and decrypt mining results received from the cloud. The computational complexity of inserting one virtual transaction is $O(1)$, and thus that of the proposed virtual transaction insertion algorithm is $O(n_T)$ which is linear. Since the time of encrypting each tag is roughly constant and the number of items is fixed, the computational complexity of encryption in owners side is $O(n_T)$. The maximum computational complexity of the decryption is $O(n_F)$, where we consider the extreme condition that all of the frequent itemsets and association rules provided by the cloud is real. Therefore, the overall computational complexity in owner side is linear to ensure a lightweight client.
- In the cloud end, the cloud extracts all of the frequent itemsets from the joint database and then finds out association rules according to the frequent itemsets. The underlying frequent itemsets mining algorithm is Apriori (FP-growth) [1], [27] and we denote its computational complexity as $O(M(n_I \times n_T))$. After obtaining the frequent itemsets, finding out the association rules is a simple calculation and comparison processing which owns computational complexity far less than $O(M(n_I \times n_T))$.

Therefore, the computational complexity of the whole mining processing containing data owners side and the cloud end is $O(M(n_I \times n_T) + n_F)$.

7.2.2 Storage Cost of SecEDMO

In SecEDMO, the storage cost for the cloud mainly lies in the joint database itself and the mining results. The joint database contains real and virtual transactions whose amount is varied and is determined by data owners. The storage capacity of transactions in the joint database is $n_I \times (1+\delta) n_T$ bits. The size of a ciphertext encrypted by HE is $O(\lambda)$, then the storage costs of encrypted ETVs and the encrypted mining results are $O(\lambda n_T)$ and $O(\lambda n_F)$ respectively. The total maximum storage cost is $O(n_I \times n_T + \lambda(n_T + n_F))$.

7.2.3 Communication Complexity of SecEDMO

The communication rounds of SecEDMO (a centralized association rules mining) are constant for data owners. At the first stage of SecEDMO, n data owners request an administration server to obtain the necessities. The owners then send their processed databases to the cloud who subsequently transmits mining results to all owners at once. Obviously, the communication rounds are fixed and will not change with the type of the database. The maximum communication traffic is $O(\lambda n_F)$, which is less than the storage cost because the cloud only transmits the mining results, not the joint database.

7.2.4 Complexities of the Comparison Schemes

Since SecEDMO is used to protect the privacy of horizontally partitioned databases, the comparison schemes [10], [12], [13] here are also designed aiming at horizontally partitioned databases, and they are distributed among n sites. n_T and n_F for the schemes are the number of all transactions and all frequent itemsets of the n sites respectively. The computational costs of [12], [10] and [13] are $O(M(n_I \times n_T) \times \lambda^3 \times |CG_{(k)}| \times n_T)$, $O(M(n_I \times n_T) + n_F \times (ElG + Pai))$ and $O(M(n_I \times n_T) + n_F)$ respectively, where $|CG_{(k)}|$ is a combinatorial number related to n_I, and ElG and Pai are computational complexities of elliptic curve ElGamal and Paillier cryptography. The communication rounds of the distributed mining systems in [10], [12], [13] are all varied with the number of sites n. Scheme in [12] has the communication

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2932065, IEEE Transactions on Cloud Computing

TABLE 8. Performance comparison between SecEDMO and the previous schemes

Scheme	Kantarcioglu *et al.* [12]	Adhvaryu *et al.* [10]	Lakshmi *et al.* [13]	SecEDMO
Underlying cryptosystem	Pohlig-Hellman	asymmetric HE	hash based sum	symmetric HE
Type of system and the miner	distributed multi sites	distributed multi sites	distributed multi sites	centralized a cloud
Computational complexity	high	high	low	low
Communication complexity	high	high	high	low
Storage cost	high	high	low	low

cost of $O(n_F \times n_I \times \lambda \times |CG_k| \times n^2)$. Its total storage cost of n sites is $O(n_T \times n_I + n_F \times n_I \times |CG_k|)$. Scheme in [10] has the communication cost of $O(n^2 \times n_F)$, and the storage cost of $O(2n_F \times (n-1) \times cs)$ for the total n sites, where cs represents the size of a ciphertext. Scheme in [13] has the maximum communication cost of $O(n^2 \times n_F)$, and the storage cost of $O(2n_F \times cs)$ for the total n sites.

The comprehensive comparison of computational complexity, communication complexity, and storage cost of different schemes are shown in Table 8. High and low complexities in the table are relative. From this table, we know that the communication complexity of a distributed data mining system is higher than that of a centralized system. Throughout all of the performance evaluations, SecEDMO is not higher than the comparison schemes in computational complexity, communication complexity and storage cost.

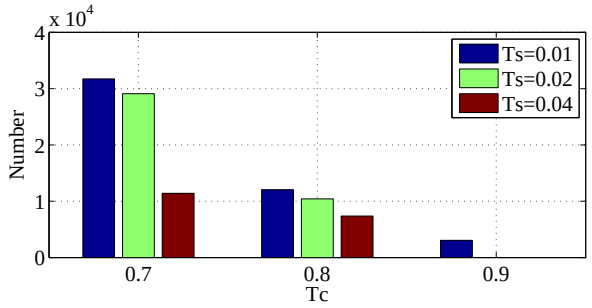

Fig. 3. The number of association rules under distinct T_c and T_s ($T_d = T_s, \delta = 0$).

7.3 End-to-end Latency

In this subsection, a simulation experiment that three clients (the owners) transmitting their databases to a server (the cloud) is set up to test the end-to-end latency of our SecEDMO.

TABLE 9. Tests of the end-to-end latency.

Parameter setting		Owner A	Owner B	Owner C
# of T in original database		39,000	27,100	22,062
# of T in inserted database		40,950	30,038	26,619
Size of inserted database		2.48MB	2.22MB	2.83MB
Size of encrypted tag value		2.50MB	1.88MB	1.62MB
Size of encrypted database		4.98MB	4.10MB	4.45MB
Latency of data transmission		14.845s	13.668s	12.669s
Latency of the HE	One item "0"	0.189s	0.061s	0.047s
	All items "0-16468"	673.774s	533.648s	581.090s

• **Datasets.** The experimental datasets come from a real-world retail market basket data – "retail.dat" [26], which owns 88,162 transactions and 16,469 items. Figure 3 presents the number of association rules with distinct support and confidence thresholds of the dataset. To simulate three-party joint mining, we divide the dataset into three parts which own 39000, 27100, and 22062 transactions, respectively.

• **Metrics.** End-to-end latency in SecEDMO mainly contains the time taken for (1) data transmission, i.e., transmitting an encrypted database across a network from the owner to the cloud and (2) the HE running in the cloud.

• **Implementation.** The implementations of SecEDMO are achieved by Java SE Development Kit. Three clients set up on jdk1.8.0-201 on a personal computer are served as three data owners, and a server set up on jdk1.7.0-80 in a desktop computer is served as a cloud. The three databases are inserted with 5%, 10% and 20% virtual transactions by three owners, respectively. Then the three inserted databases own 40980, 30038 and 26619 transactions with size 2.48MB, 2.22MB, and 2.83MB (uncompressed version). The bit length of p_3 in our experiment is assumed as 64 which meets the parameter requirement given in Section IV-C.

• **Experimental Results.** The size of the encrypted tag values and the latency of the data transmission and the HE are listed in Table 9. In the table, the transmission latencies within 15s are acceptable. The HE for one item can be used to determine whether a certain item is frequent. Its runtime is short, so real-time identifications of frequent itemsets can be realized. The runtime of the HE for all 16,469 items is around 10 minutes, which may mean a long latency when obtaining all the joint database's frequent itemsets with real-time response. But for market basket analysis, e.g., a supermarket gathers transactions on customer buying habits to determine which commodities are frequently purchased together and uses this information for marketing purposes [28], [29]. The 10 minutes' runtime is acceptable, because the supermarket may spend more time to reconstruct its commodity distribution according to the mining results.

8 Related Work

Generally speaking, solutions for all the existing privacy-preserving association rules mining can be classified into three categories: data disturbance-based solutions, query restriction-based solutions, and data encryption-based solutions [30].

8.1 Data Disturbance-based Privacy-Preserving Mining

Data Disturbance ensures that the disturbed data does not reveal private information when they are mined. Data disturbance algorithm in [31] proposes a random projection matrix based on multiplication to maintain privacy in distributed data mining

8.2 Query Restriction-based Privacy-Preserving Mining

Query restriction provides restricted access to original data to protect data privacy. A query restriction scheme in [32] is

to mine fussy association in databases which are generalized (a general and abstract description of data) so that the certain privacy information in the databases will not be leaked. But the mining results considered in this type of scheme are subject to many qualifications, e.g., an owner cannot get an association rule that is restricted.

8.3 Data Encryption-based Privacy-Preserving Mining

Both the above privacy-preserving schemes based on data disturbance and query restriction are used to mine fuzzy association rules (thus the mining results are inaccurate) while privacy-preserving mining schemes based on data encryption are accurate. We present two types of data encryption schemes in privacy-preserving data mining as follows.

• **SMC-based.** Secure multi-party summation between n owners in [33] is proposed to solve collusion attacks: the owners first randomly divide their private data into n pieces and distribute $n-1$ pieces to the other owners, then they separately calculate the sum of the data they owned and transmit the sum to all the other owners. Finally, the owners calculate the received partial sum and then calculate the total. This algorithm can solve collusion between less than $n-1$ owners, but it owns a high communication cost which is $O(n^2)$ in owners' side, which cannot meet a lightweight client requirement. Unlike this solution, SecEDMO owns fix communication rounds in the owners' side and can resist collusion attacks at the same time.

• **HE-based.** To realize low computational and communication cost, multi-owners outsource their databases to a semi-trusted third party (such as a cloud) to implement a secure mining algorithm. Homomorphic encryptions are very adapted to secure outsourcing computing. Li *et al.* [14] proposes a secure outsourcing association rules mining scheme based on a symmetric HE and all the involved data owners share the same secret keys. The scheme has good performance in computational and communication complexity but is vulnerable to most of the attacks shown in this paper. Our SecEDMO, different from this scheme, realizes a good trade-off between performance and security.

9 Conclusion

In this paper, we have explored the problem of privacy-preserving association rules mining on horizontally partitioned databases of multiple data owners in cloud computing. In contrast to prior works, SecEDMO is secure in presence with various attacks including secret keys attacks, ciphertexts tampering, data owners corruption and collusion between owners or even with the cloud. Moreover, the designed method in SecEDMO provides a new idea of secure data computations (e.g., data aggregation) with high efficiency and anti-attack at the same time.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *International Conference on Very Large Data Bases*, 1994, pp. 487–499.

[2] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. Mclachlan, A. Ng, B. Liu, and P. S. Yu, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[3] B. Yang, "Association rule mining in distributed databases for market supervision and management," *Informatization Research*, 2017.

[4] Y. Zhang and G. Chen, "A forensics method of web browsing behavior based on association rule mining," in *International Conference on Systems and Informatics*, 2015, pp. 927–932.

[5] Y. Wang, "Research on the technology of association rule mining in intrusion detection system," *Computer Science*, vol. 35, no. 10, pp. 81–84, 2008.

[6] A. Forkan, I. Khalil, A. Ibaida, and Z. Tari, "Bdcam: Big data for context-aware monitoring - a personalized knowledge discovery framework for assisted healthcare," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[7] S. Qiu, B. Wang, M. Li, J. Liu, Y. Shi, S. Qiu, B. Wang, M. Li, J. Liu, and Y. Shi, "Toward practical privacy-preserving frequent itemset mining on encrypted cloud data," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[8] X. Yi, F. Y. Rao, E. Bertino, and A. Bouguettaya, "Privacy-preserving association rule mining in cloud computing," in *ACM Symposium on Information, Computer and Communications Security*, 2015, pp. 439–450.

[9] M. V. Ahluwalia, A. Gangopadhyay, Z. Chen, and Y. Yesha, "Target-based, privacy preserving, and incremental association rule mining," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 633–645, 2017.

[10] R. Adhvaryu and N. Domadiya, *An Improved EMHS Algorithm for Privacy Preserving in Association Rule Mining on Horizontally Partitioned Database*. Springer Berlin Heidelberg, 2014.

[11] S. Zhong, "Privacy-preserving algorithms for distributed mining of frequent itemsets," *Information Sciences*, vol. 177, no. 2, pp. 490–503, 2007.

[12] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[13] N. V. M. Lakshmi and K. S. Rani, "Privacy preserving association rule mining without trusted party for horizontally partitioned databases," *International Journal of Data Mining and Knowledge Management Process*, vol. 2, no. 2, p. 0151, 2012.

[14] L. Li, R. Lu, K. K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on vertically partitioned databases," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1847–1861, 2016.

[15] B. Wang, Y. Zhan, and Z. Zhang, "Cryptanalysis of a symmetric fully homomorphic encryption scheme," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1460–1467, 2018.

[16] S. Sharma and S. Ahuja, "Privacy preserving data mining: A review of the state of the art," in *Harmony Search and Nature Inspired Optimization Algorithms. Springer, Singapore*, 2019, pp. 1–15.

[17] N. Paladi, C. Gehrmann, and A. Michalas, "Providing user security guarantees in public infrastructure clouds," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[18] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169–179, 1978.

[19] R. L. Rivest, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 26, no. 2, pp. 96–99, 1978.

[20] Gentry and Craig, "Fully homomorphic encryption using ideal lattices," *Stoc*, vol. 9, no. 4, pp. 169–178, 2009.

[21] F. Giannotti, L. V. S. Lakshmanan, A. Monreale, D. Pedreschi, and H. Wang, "Privacy-preserving mining of association rules from outsourced transaction databases," *IEEE Systems Journal*, vol. 7, no. 3, pp. 385–395, 2013.

[22] W. Zhang, H. Yu, Y. L. Zhao, and Z. L. Zhu, "Image encryption based on three-dimensional bit matrix permutation," *Signal Processing*, vol. 118, pp. 36–50, 2016.

[23] C. H. Lin, Y. S. Yeh, S. P. Chien, C. Y. Lee, and H. S. Chien, "Generalized secure hash algorithm: Sha-x," in *Eurocon - International Conference on Computer as a Tool*, 2011, pp. 1–4.

[24] Y. H. Lin, S. Y. Chang, and H. M. Sun, "Cdama: concealed data aggregation scheme for multiple applications in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1471–1483, 2013.

[25] D. Vinodha and E. A. M. Anita, "Secure data aggregation techniques for wireless sensor networks: a review," *Archives of Computational Methods in Engineering*, no. 8, pp. 1–21, 2018.

[26] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, "Using association rules for product assortment decisions," *Knowledge Discovery and Data Mining*, pp. 254–260, 1999, http://fimi.ua.ac.be/data/.

[27] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *Acm Sigmod Record*, vol. 29, no. 2, pp. 1–12, 2000.

[28] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.
[29] "Data mining," https://en.wikipedia.org/wiki/Data_mining.
[30] Y. H. Liu, B. R. Yang, M. A. Nan, and D. Y. Cao, "State-of-the-art in distributed privacy preserving data mining," *Application Research of Computers*, vol. 28, no. 10, pp. 545 – 549, 2011.
[31] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 92–106, 2006.
[32] I. Mguiris, H. Amdouni, and M. M. Gammoudi, "An algorithm for fuzzy association rules extraction based on prime number coding," in *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2017.
[33] Y. Luo, Z. Xu, and L. Huang, "Secure multi-party statistical analysis problems and their applications," *Computer Engineering and Applications*, vol. 41, no. 24, pp. 145–147, 2005.

Jiahui Wu received the B.S. degree in electronic information science and technology from Anqing Normal University, Anhui, China, in 2014, and the M.S. degree in signal and information processing from Southwest University, Chongqing, China, in 2018. At present, she is pursuing a Ph.D. degree in intelligent computing and information processing at Southwest University. Her research areas are including information security, cloud computing security, and data mining.

Nankun Mu received the B.S. degree in software engineering, the M.S. degree in computer systems and structures and the Ph.D. degree in computer science and technology from Chongqing University, Chongqing, China, in 2011, 2013 and 2015 respectively. His research areas including information security, multi-agent control system, evolutionary computation.

Xinyu Lei received the B.S. and M.S. degree in computing science from Chongqing University, China, in 2010 and 2014, respectively. He worked in Texas A&M University at Qatar and Ford Motor company in 2013 and 2017, respectively. He received Engineering Distinguished Fellowship Award in Michigan State University in 2015. He is now working toward the Ph.D. degree in Computer Science and Engineering at Michigan State University, USA. His current research focuses on IoT, blockchain, cloud computing, etc.

Junqing Le received the B.S. degree in software engineering from Southwest Jiaotong University, Chengdu, China, in 2014, and the M.S. degree in signal and information processing from Southwest University, Chongqing, China, in 2017. At present, he is pursuing a Ph.D. degree in intelligent computing and information processing at Southwest University. His research areas are including privacy protection, privacy machine learning, and intelligent transportation systems.

Di Zhang received the B.S. degree in communication engineering from Southwest University, Chongqing, China, in 2014. At present, she is pursuing the Ph.D. degree in Computational Intelligence and Information Processing with the College of Electronics and Information Engineering, Southwest University. Her research areas are including chaos, cryptography, cloud computing security, and blockchain.

Xiaofeng Liao received the BS and MS degrees in mathematics from Sichuan University, Chengdu, China, in 1986 and 1992, respectively, and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China in 1997. From 1999 to 2012, he was a professor at Chongqing University. At present, he is a professor at Southwest University and the Dean of School of Electronic and Information Engineering. He is also a Yangtze River Scholar of the Ministry of Education of China. From November 1997 to April 1998, he was a research associate at the Chinese University of Hong Kong. From October 1999 to October 2000, he was a research associate at the City University of Hong Kong. From March 2001 to June 2001 and March 2002 to June 2002, he was a senior research associate at the City University of Hong Kong. From March 2006 to April 2007, he was a research fellow at the City University of Hong Kong. Professor Liao holds 4 patents and published 4 books and over 400 international journal and conference papers. His current research interests include neural networks, nonlinear dynamical systems, bifurcation and chaos, information security and cryptography. He currently serves as an associate editor for IEEE Transactions on Cybernetics and IEEE Transactions on Neural Networks and Learning Systems.