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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

Privacy-Preserving Outsourced Association Rule Mining on Vertically Partitioned Databases

Lichun Li, Rongxing Lu, *Senior Member, IEEE,* Kim-Kwang Raymond Choo, *Senior Member, IEEE,* Anwitaman Datta, and Jun Shao

***Abstract*—Association rule mining and frequent itemset mining are two popular and widely studied data analysis techniques for a range of applications. In this paper, we focus on privacy-preserving mining on vertically partitioned databases. In such a scenario, data owners wish to learn the association rules or frequent itemsets from a collective dataset, and disclose as little information about their (sensitive) raw data as possible to other data owners and third parties. To ensure data privacy, we design an efficient homomorphic encryption scheme and a secure comparison scheme. We then propose a cloud-aided frequent itemset mining solution, which is used to build an association rule mining solution. Our solutions are designed for outsourced databases that allow multiple data owners to efficiently share their data securely without compromising on data privacy. Our solutions leak less information about the raw data than most existing solutions. In comparison to the only known solution achieving a similar privacy level as our proposed solutions, the performance of our proposed solutions is 3 to 5 orders of magnitude higher. Based on our experiment findings using different parameters and datasets, we demonstrate that the run time in each of our solutions is only one order higher than that in the best non-privacy-preserving data mining algorithms. Since both data and computing work are outsourced to the cloud servers, the resource consumption at the data owner end is very low.**

***Index Terms*—Association rule mining, frequent itemset mining, privacy-preserving data mining**

I. Introduction

Frequent itemset mining and association rule mining, two widely used data analysis techniques, are generally used for discovering frequently co-occurring data items and interesting association relationships between data items respectively in large transaction databases. These two techniques have been employed in applications such as market basket analysis [1], health care [2], web usage mining [3], bioinformatics [4] and prediction [5]. A transaction database is a set of transactions, and each transaction is a set of data items with a unique TID (Transaction ID). An itemset Z is regarded frequent if and only if $Supp(Z) \geq T_s$, where T_s is a threshold specified by the data miner. $Supp(Z)$ is Z's support, which is defined as Z's occurrence count in the database. An association rule is expressed using $X \Rightarrow Y$, where X and Y are two disjoint itemsets. $X \Rightarrow Y$ indicates that X's occurrence implies Y's occurrence in the same transaction with a certain confidence. We will use a supermarket's transaction database as an example, where a transaction is some customer's shopping list. A customer buying "bread" and "butter" will also buy "milk". Then {bread, butter} $\Rightarrow$ milk is a possible association rule. $X \Rightarrow Y$ is meaningful and useful if the confidence is high and $X \cup Y$ is frequent. More specifically, $X \Rightarrow Y$ is regarded as an association rule if and only if $Supp(X \cup Y) \geq T_s$ and $Conf(X \Rightarrow Y) \geq T_c$. We define $Conf(X \Rightarrow Y)$ as the confidence of $X \Rightarrow Y$. The latter is the probability of Y's occurrence given X's occurrence (i.e. $Conf(X \Rightarrow Y) = Supp(X \cup Y)/Supp(X)$). T_c denotes the threshold specified by the data miner. We also remark that the values of T_s and T_c are generally configured based on the type of transactions, the usage of the mining result, the size of database, etc. It is easy to mine association rules after mining frequent itemsets and obtaining their supports. Most association rule mining algorithms are built based on frequent itemset mining algorithms.

Classic frequent itemset mining and association rule mining algorithms, such as Apriori [6], Eclat [7] and FP-growth [8], were designed for a centralized database setting where the raw data is stored in the central site for mining. Privacy concerns were not considered in this setting. Vaidya and Clifton [9] and Kantarcioglu and Clifton [10] are the first to identify and address privacy issues in horizontally / vertically partitioned databases. Due to an increased understanding of the importance of data privacy (e.g. in the aftermath of the revelations by Edward Snowden, a former NSA contractor), a number of privacy-preserving mining solutions have been proposed in recent times. In their settings, there are multiple data owners wishing to learn association rules or frequent itemsets from their joint data. However, the data owners are not willing to send their raw data to a central site due to privacy concerns. If each data owner has one or more rows (i.e. transactions) in the joint database, we say that the database is *horizontally partitioned*. If each data owner has one or more columns in the joint database, the database is considered *vertically partitioned*. This paper focuses on

L. Li, and R. Lu are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. 639798. E-mail: lilichun@gmail.com; rxlu@ntu.edu.sg.

K.K.-R. Choo is with (1) Department of Information Systems and Cyber Security, University of Texas at San Antonio, USA, (2) School of Information Technology & Mathematical Sciences, University of South Australia, Australia, and (3) School of Computer Science, China University of Geosciences, Wuhan, China. E-mail: raymond.choo@fulbrightmail.org.

A. Datta is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. 639798. E-mail: Anwitaman@ntu.edu.sg.

J. Shao is with the Department of Information Security, Zhejiang Gongshang University, Hangzhou 310018, China. E-mail: chn.junshao@gmail.com.

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

vertically partitioned databases, and as explained in [11], such databases are useful for market basket analysis. For example, different businesses, such as a fashion designer and a luxury watch designer, sell different products to the same community. These businesses collaborate to mine customer buying patterns from the joint database. A transaction of the database contains the products that a customer had bought from one or more of the participating businesses, and attributes such as the customer credit card number and date of purchase are used as TIDs. Therefore, each of the businesses (i.e. data owners) will own some transaction partitions in the joint database. However, these businesses may not wish to disclose such data, which include trade secrets (e.g. there may be other competing businesses sharing the same joint database) and customer privacy (e.g. due to regulations in existing privacy regime). Therefore, a privacy-preserving mining solution must be applied. Other use cases can also be found in areas such as automotive safety [9] and national security [12].

In this paper, we propose a cloud-aided privacy-preserving frequent itemset mining solution for vertically partitioned databases, which is then used to build a privacy-preserving association rule mining solution. Both solutions are designed for applications where data owners have a high level of privacy requirement. The solutions are also suitable for data owners looking to outsource data storage – i.e. data owners can outsource their encrypted data and mining task to a semi-trusted (i.e. curious but honest) cloud in a privacy-preserving manner. To the best of our knowledge, this is the first work on outsourced association rule mining and frequent itemset mining for vertically partitioned databases. The key underlying techniques in our solutions are an efficient homomorphic encryption scheme and a secure outsourced comparison scheme.

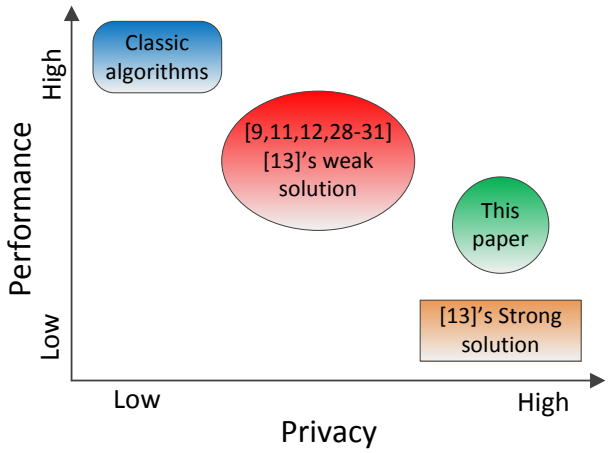

Fig. 1: Design space of association rule and frequent itemset mining solutions

The contributions of this paper are three-fold:

- This paper proposes privacy-preserving mining solutions for high privacy requirements. As shown in Figure 1, the proposed solutions are uniquely located in the design space. Compared with most solutions, our solutions achieve a higher privacy level, as most existing solutions require either the sharing / exposure of raw data or the disclosure of the exact supports to data owners. Such requirements result in the leakage of sensitive information of the raw data [11]. Our solutions are designed to avoid such complications. We note that one of the frequent itemset mining solutions in [13] can achieve the same privacy level as our proposed solutions. However, an association rule mining solution cannot be built based on the frequent itemset mining solution in [13]. In contrast, we present solutions for both frequent itemset mining and association rule mining. Moreover, as shown in Section VII, our frequent itemset mining solution is 3 to 5 orders faster. Our solution is significantly more efficient due to our customized homomorphic encryption scheme. The introduction of a semi-trusted third party (i.e. the cloud) also allows us to securely compute supports and compare supports with a threshold T_s more efficiently – see Sections VIII and VII for related work and comparative summary, respectively.
- This paper proposes an efficient homomorphic encryption scheme and a secure outsourced comparison scheme. To avoid the disclosure of supports / confidences, we design an efficient homomorphic encryption scheme to facilitate secure outsourced computation of supports / confidences, as well as a secure outsourced comparison scheme for comparing supports / confidences with thresholds. The proposed (symmetric homomorphic) encryption scheme is tailored for the proposed comparison. The scheme only requires modular additions and multiplications, and is more efficient than the homomorphic encryption schemes used in other association rule mining and frequent itemset mining solutions. For example, encryption computing in our scheme is three orders of magnitude faster than [14] and [15] respectively. To the best of our knowledge, the proposed secure comparison scheme is the first scheme based on symmetric homomorphic encryption. The proposed schemes are designed for the data mining solutions outlined in this paper, but they can potentially be adopted in a wide range of secure computation applications.
- This paper proposes a ciphertext tag approach for canceling out fictitious data's effect on mining result. To "hide" the data owner's raw data from the cloud, we adapt the concept outlined in [16] by encrypting items with a substitution cipher, and adding fictitious transactions as a mitigation against frequency analysis attacks on the substitution cipher. To allow secure and accurate computation of supports, we design a ciphertext tag approach to cancel out fictitious transactions in a privacy-preserving manner. Although our approach is designed for the data mining solutions outlined in this paper, it has potential applications in other secure computation contexts, such as secure data aggregation.

Organization. The remainder of this paper is organized as follows. In Section II, we formalize the system model and security model considered in this paper, and identify the design goals. The required background material on cryptographic techniques is provided in Section III. In Section IV, we present

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

our proposed homomorphic encryption scheme and secure comparison scheme, which will serve as the basis of our solutions. In Section V, we present our privacy-preserving solutions for association rule mining and frequent itemset mining on vertically partitioned databases, followed by the security analysis and performance evaluation in Section VI and Section VII, respectively. Related work is discussed in Section VIII. We conclude the paper in Section IX.

II. Models and Design Goals

In this section, we formalize the system model and security model used in this paper, and identify the design goals.

A. System Model

The system model (see Fig. 2) is comprised of two or more data owners and a cloud. Each data owner has a private database, and the data owners encrypt their private databases prior to outsourcing the encrypted databases to the cloud. Data owners can also request the cloud to mine association rules or frequent itemsets from the joint database on their behalf. The (honest but curious) cloud is tasked with the compiling and storing of databases received from different data owners, the mining of association rules or frequent itemsets for data owners, and the sending of the mining result to relevant data owners.

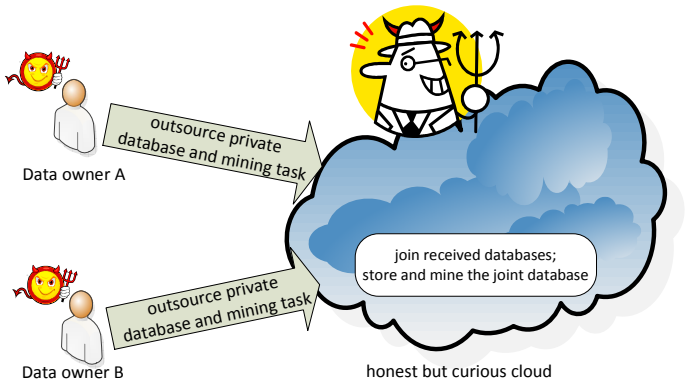

Fig. 2: System model of outsourced data mining on joint database

B. Security Model

The cloud is considered honest but curious in this paper. Firstly, the cloud honestly stores and mines data for data owners. Data owners pay for the cloud's services, and they will naturally choose a cloud believed to be honest (e.g. a cloud provider with a trusted reputation). There are also techniques to detect dishonest clouds [17], [18], and dishonest clouds could be detected by simply comparing the mining results from different clouds. Secondly, although the cloud is not malicious, it is motivated to learn the data of data owners for financial gains (e.g. for paid advertisement). In other words, the cloud will attempt to learn the raw data of the private databases and the mining results.

The cloud is assumed to have some background knowledge of some items and their frequencies; hence, it can launch frequency analysis attacks. For each of these items, the cloud may also know which data owner's private database the item belongs to. This makes such attacks easier because the cloud only needs to analyze frequencies in a private database to determine the ciphertext of a targeted item. However, we do not further assume the cloud has background knowledge of itemset frequency. This is reasonable for many cases; otherwise, even the data owners themselves need to learn frequent itemsets. Also, we do not assume the cloud to be colluding with any data owner. Hence, colluding attacks and insider attacks are not considered in this paper.

Data owners are also considered collaborative but curious in this paper. In the typical system setup for vertically partitioned databases, data owners participate in the collaborative data mining in order to obtain the mining result. For example, a fashion designer and a luxury watch designer located in a major capital city contributing their datasets regarding consumer buying habits and preferences, etc, so that they can better understand the consumer and suggest products to suit their preferences. Thus, data owners wish to learn the mining result, and are willing to collaborate with each other. However, some data owners may deviate from the cooperative mining protocol in order to learn the data of other data owner(s) as long as the deviation does not sabotage the mining result. As mentioned earlier, there are financial gains in doing so.

We assume each data owner has some knowledge of other owners' private databases. This is not surprising. Data owners are willing to participate in the collaborative mining, which indicates that they may already have such knowledge. With some knowledge of other owners' private databases, these data owners believe they can benefit from the collaborative mining. We assume each data owner knows the items and the size of any other data owner's private database. As vertically partitioned databases are being mined, we also assume that each owner has more knowledge about other private databases' TIDs. We remark that it is not required that all owners' private databases share the same set of TIDs, although it is expected that they share a lot of TIDs. Again, we do not assume any data owner to be colluding with the cloud.

C. Design Goals

The goals of our proposed privacy-preserving association rule mining and frequent itemset mining for vertically partitioned databases are as follow:

- *Privacy*. Data owners should learn as little information about databases belonging to other data owners as possible. More specifically, a data owner's raw transaction details should not be disclosed, and the supports should be concealed to avoid leakage of information about the raw data [11]. Similarly, exact confidences should be concealed as they could be used to infer some information about the raw data. The proposed solutions should also protect the mining results from the cloud.
- *Efficiency*. Privacy-preserving measures usually result in decreased performance of data mining, and therefore,

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

any trade-off has to be realistic. In our context, the data mining latency should be acceptable compared with the latencies of non-privacy-preserving data mining algorithms.

III. Preliminary: Substitution Cipher, Cryptographic Hash Function and Homomorphic Encryption

In this section, we outline the substitution cipher, cryptographic hash function and homomorphic encryption, which serve as the building blocks in our privacy-preserving data mining solutions.

A. Substitution Cipher and Frequency Analysis

A substitution cipher encrypts a message by substituting the units of the message with ciphertext units according to a substitution alphabet. Substitution cipher has been used in outsourced association rule mining and frequent itemset mining [19], [20], [21], [16]. In such use cases, the message units are the items in a transaction database. Let Σ be the plaintext alphabet including all unique items appearing in a database. Every item in Σ has a corresponding unique ciphertext. A substitution alphabet example is shown in Table I. To encrypt the database, all items in the database are replaced with their corresponding ciphertexts.

Substitution cipher is subject to frequency analysis attack if the frequencies of message units are different. *Frequency analysis*, the analysis of frequencies of ciphertext units or unit groups, has been used to break classical ciphers such as substitution ciphers. Attackers with some knowledge of the frequencies of message units or unit groups can recover some plaintext through frequency analysis. For example, if an attacker knows that bread and milk are the most and second most frequent items in a transaction database, the attacker can infer that the most and second most frequent ciphertext units in the encrypted database correspond to bread and milk, respectively. To counter frequency analysis attack, fictitious items or transactions can be added to hide item frequency.

TABLE I: An example of substitution alphabet

plaintext alphabet	milk	butter	bread
ciphertext alphabet	0110	1101	0010

B. Cryptographic Hash Function

A cryptographic hash function $H()$ has the properties of pre-image resistance and collision resistance. The former is related to a one-way function. In other words, when given a hash value h, it is computationally infeasible to find a message m satisfying $H(m) = h$. The property of collision resistance means the collision of hash values are very rare – i.e. it is computationally infeasible to find two distinct messages m_1 and m_2 satisfying $H(m_1) = H(m_2)$. Commonly used cryptographic hash functions include SHA-1 [22] and SHA-2 [23].

C. Homomorphic Encryption

Homomorphic encryption scheme allows one or more plaintext operations (e.g addition and multiplication) to be carried out on the ciphertexts. If the addition operation is allowed, then the scheme is known as additive homomorphic encryption. If the multiplication operation is allowed, then the scheme is known as multiplicative homomorphic encryption.

In an additive homomorphic encryption scheme, the ciphertext of the sum of two plaintexts, $m_1 + m_2$, can be obtained using some computation "$\bullet$" on the ciphertexts of m_1 and m_2, without first decrypting m_1 and m_2 or requiring the decryption key. Additive homomorphic encryption also allows the user to obtain the ciphertext of $m_1 \times m_2$ by performing m_2 times of "$\bullet$" computation on m_1's ciphertext. The most common additive homomorphic encryption schemes are Paillier encryption [14] and a variant of ElGamal encryption [15]. For example, in Paillier encryption, let $E_{PK}()$ be the function of encrypting with the public key, and "$\bullet$" is modular multiplication in Paillier. Given $E_{PK}(m_1)$, $E_{PK}(m_2)$ and the public key used in the encryption, one can compute $E_{PK}(m_1 + m_2)$ by performing a modular multiplication of $E_{PK}(m_1)$ and $E_{PK}(m_2)$. Similarly, given $E_{PK}(m_1), m_2$ and the public key, one can compute $E_{PK}(m_1 \times m_2)$ by performing a modular exponentiation $E_{PK}(m_1)^{m_2}$.

$$E_{PK}(m_1 + m_2) = E_{PK}(m_1) \times E_{PK}(m_2)$$

$$\begin{aligned} & E_{PK}(m_1 \times m_2) \\ =& \underbrace{E_{PK}(m_1) \times E_{PK}(m_1) \times ... E_{PK}(m_1)}_{(m_2 \text{ multiplications})} \\ =& E_{PK}(m_1)^{m_2} \end{aligned}$$

In the remainder of this paper, $\bullet$ denotes *homomorphic addition*, and the computing of the ciphertext of $m_1 \times m_2$ based on homomorphic addition is referred to as *homomorphic scalar multiplication*.

In a multiplicative homomorphic encryption scheme, the ciphertext of the product of two plaintexts, $m_1 \times m_2$, can be obtained with some kind of computation "$\otimes$" on the ciphertexts of m_1 and m_2, without first decrypting m_1 and m_2 or requiring the decryption key. The most common multiplicative homomorphic encryption scheme is ElGamal encryption [24], which is an asymmetric scheme.

IV. Proposed Homomorphic Encryption and Secure Outsourced Comparison Schemes

In this section, we propose an efficient homomorphic encryption scheme. Using the proposed homomorphic encryption scheme, we construct a secure outsourced comparison scheme. Both schemes will then serve as the basis of our privacy-preserving mining solutions.

A. Proposed Homomorphic Encryption Scheme

Existing homomorphic encryption schemes are generally asymmetric [14], [15]. In this paper, we propose a symmetric homomorphic encryption scheme (using only modular

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

additions and multiplications), which is significantly more efficient than asymmetric schemes. The scheme supports many homomorphic additions and limited number of homomorphic multiplications, and comprises the following three algorithms:

- **Key generation algorithm** $KeyGen()$

  $$(s, q, p) \leftarrow KeyGen(\lambda)$$

  The key generation algorithm $KeyGen()$ is a probabilistic algorithm, which takes a security parameter λ as input and outputs a secret key $SK = (s, q)$ and a public parameter p. Both p and q are big primes, and $p \gg q$. The bit length of q depends on the security parameter, and s is a random number from $\mathbb{Z}_p^*$.
- **Encryption algorithm** $E()$

  $$E(SK, m, d) = s^d(rq + m) \bmod p$$

  The encryption algorithm $E()$ is a probabilistic algorithm, which takes a secret key SK, a plaintext $m \in \mathbb{F}_q$ and a parameter d as inputs. The algorithm outputs a ciphertext $c \leftarrow E(SK, m, d)$. The parameter d is a small positive integer called *ciphertext degree*, and we say the ciphertext is a *d-degree ciphertext*. Let r denote a big random positive integer, and the bit length of r, $|r|$, satisfies $|r| + |q| < |p|$. We say r is the *random ingredient* of c. The encryption of a plaintext m is denoted by $E(m)$ for short.
- **Decryption algorithm** $D()$

  $$D(SK, c, d) = (c \times s^{-d} \bmod p) \bmod q$$

  The decryption algorithm $D()$ is a deterministic algorithm, which takes a secret key SK, a ciphertext $c \in \mathbb{F}_p$ and the ciphertext's degree d as inputs. The algorithm outputs a plaintext $m \leftarrow D(SK, c, d)$. Let s^{-d} denote the multiplicative inverse of s^d in the field $\mathbb{F}_p$. The correctness proof of the decryption algorithm is given below.

$$\begin{aligned} D(SK, c, d) &= (c \times s^{-d} \bmod p) \bmod q \\ &= ((s^d(rq + m) \bmod p) \times s^{-d} \bmod p) \bmod q \\ &= (rq + m) \bmod q \\ &= m \end{aligned}$$

B. Property of the proposed homomorphic encryption

Homomorphic multiplication. Let c_1, c_2 be the ciphertexts of two plaintexts m_1, m_2. Then, we have $c_1 = s^{d_1}(r_1q + m_1) \bmod p$ and $c_2 = s^{d_2}(r_2q + m_2) \bmod p$ for some random ingredients r_1 and r_2. As shown below, given d_1-degree ciphertext c_1 and d_2-degree ciphertext c_2, the $d_1 + d_2$-degree ciphertext of $m_1 \times m_2$ can be computed with a modular multiplication. To correctly decrypt $m_1 \times m_2$ from its ciphertext, $(r_1r_2q + r_1m_2 + m_1r_2)q + m_1 \times m_2 < p$ must be satisfied where $(r_1r_2q + r_1m_2 + m_1r_2)$ is the random ingredient. Therefore, in Section V, we choose the bit lengths which satisfy the condition and ensure the correctness of decryption. It is not hard to do so, as $|q| > |m_1|$ and $|q| > |m_2|$ and we have $|r_1r_2q| > |r_1m_2| + |m_1r_2|$. We only need to ensure $|r_1| + |r_2| + 2|q| + 1 < |p|$.

$$\begin{aligned} &(c_1 \times c_2) \bmod p \\ =&s^{d_1}(r_1q + m_1) \bmod p \times s^{d_2}(r_2q + m_2) \bmod p \\ =&s^{d_1+d_2}(r_1r_2q^2 + r_1qm_2 + m_1r_2q + m_1 \times m_2) \bmod p \\ =&s^{d_1+d_2}((r_1r_2q + r_1m_2 + m_1r_2)q + m_1 \times m_2) \bmod p \end{aligned}$$

Homomorphic addition. As shown below, the ciphertext of $m_1 + m_2 \bmod q$ can be computed by a modular addition of c_1 and c_2 if $d_1 = d_2$. To correctly decrypt $m_1 + m_2$ from its ciphertext, $(r_1 + r_2)q + m_1 + m_2 < p$ must be satisfied. Therefore, in Section V, we choose the bit lengths of p, q and random ingredients to ensure that all ciphertexts in our privacy-preserving mining solutions can be decrypted correctly.

$$\begin{aligned} &c_1 + c_2 \bmod p \\ =&s^{d_1}(r_1q + m_1) \bmod p + s^{d_2}(r_2q + m_2) \bmod p \\ =&s^{d_1}((r_1 + r_2)q + m_1 + m_2) \bmod p \quad \text{if } d_1 = d_2 \end{aligned}$$

Homomorphic subtraction. Similarly, as shown below, homomorphic subtraction can be achieved with a modular subtraction. To correctly decrypt $m_1 - m_2$ from its ciphertext, $r_1 - r_2$ must be satisfied.

$$\begin{aligned} &(c_1 - c_2) \bmod p \\ =&(s^{d_1}(r_1q + m_1) - s^{d_2}(r_2q + m_2)) \bmod p \\ =&s^{d_1}((r_1 - r_2)q + m_1 - m_2) \bmod p \quad \text{if } d_1 = d_2 \end{aligned}$$

Degree alignment for homomorphic addition/subtraction. Homomorphic addition and subtraction requires ciphertexts sharing the same degree. If c_1 and c_2 have different ciphertext degrees, homomorphic addition/subtraction can be performed after upgrading the lower-degree ciphertext to a ciphertext of the higher degree. Suppose c_2's degree d_2 is lower. A d_1-degree ciphertext of m_2, c_2', can be computed by doing a homomorphic multiplication of c_2 and a $(d_1 - d_2) - degree$ ciphertext of 1. Then we can do homomorphic addition/subtraction of c_1 and c_2'.

Homomorphic scalar multiplication. Given m_1's ciphertext c_1 and a plaintext m_2, the ciphertext of $m_1 \times m_2$ can be computed with a modular multiplication. To correctly decrypt $m_1 \times m_2$ from its ciphertext, $r_1m_2q + m_1 \times m_2 < p$ must be satisfied. Therefore, in Section V, we choose the bit lengths that the condition is satisfied which will ensure the correctness of decryption.

$$\begin{aligned} &(c_1 \times m_2) \bmod p \\ =&s^{d_1}(r_1q + m_1) \bmod p \times m_2 \bmod p \\ =&s^{d_1}(r_1m_2q + m_1 \times m_2) \bmod p \end{aligned}$$

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

C. Proposed Secure Outsourced Comparison Scheme

The proposed secure comparison scheme is based on the symmetric homomorphic encryption scheme discussed in Section IV-A. In our privacy-preserving data mining solutions, data owners require the cloud to compare supports / confidences with thresholds. However, both supports and confidences must be kept secret from the cloud and data owners, while the comparison results must be kept secret from the cloud. As shown in Section V, these secure comparison problems can be transformed to the same form below.

Let m be a secret integer unknown to the cloud and data owners, where m is in the range $[-\alpha, \beta]$ known to the cloud, and $q \gg \alpha, \beta > 0$. Data owners are indexed from 1 to t. Let μ_i be a ciphertext of "1" generated by the i-th data owner. The cloud holds $c = E(m \bmod q)$, p and q's bit length $|q|$ and $\{\mu_i : t \geq i \geq 1\}$. All these ciphertexts share the same degree d. (In our privacy-preserving data mining solutions, c is computed by the cloud from outsourced encrypted data utilizing the homomorphic property.) The data owners hold SK, and want to know whether $m \geq 0$. The data owners need the cloud to compare m with 0 in a privacy-preserving manner. However, as described in the preceding paragraph, m must be kept secret from the cloud and data owners, while the comparison result is kept secret from the cloud.

The secure comparison scheme for the above problem is as follows:

- Firstly, the cloud generates random integers, $u, \{v_i : t \geq i \geq 1\}$, meeting the following four requirements.

$$u \gg \sum_{i=1}^{t} v_i$$
$$v_i \gg max(\alpha, \beta) \qquad (t \geq i \geq 1)$$
$$(q-1)/2 \gg \beta \times u + \sum_{i=1}^{t} v_i$$
$$-\alpha \times u + \sum_{i=1}^{t} v_i \gg -(q-1)/2$$

  Note: The above requirements do not require q to be disclosed to the cloud, and the cloud can generate the $u, \{v_i : t \geq i \geq 1\}$ meeting above requirements as long as it knows $\alpha, \beta, |q|$.
- Secondly, the cloud computes $\Omega = c^u + \sum_{i=1}^{t} \mu_i^{v_i} \bmod p$, and sends Ω to data owners.
  Note: Due to the homomorphic property of the underlying encryption scheme, Ω is the ciphertext of $(m \times u + \sum_{i=1}^{t} v_i) \bmod q$.
- Thirdly, each data owner computes $\varphi = D(SK, \Omega, d) = (m \times u + \sum_{i=1}^{t} v_i) \bmod q$, and compares φ with $(q-1)/2$. If $\varphi < (q-1)/2$, $m \geq 0$. Otherwise, $m < 0$.
  Note: $u, \{v_i : t \geq i \geq 1\}$ are used to mask the value of m, while preserving the sign of m.

Correctness. Let us now consider following two cases.

Case of $m \geq 0$. Since $u \gg \sum_{i=1}^{t} v_i \gg \beta$ and $(q-1)/2 \gg \beta \times u + \sum_{i=1}^{t} v_i$, we have

$(q-1)/2 \gg \beta \times u + \sum_{i=1}^{t} v_i \geq m \times u + \sum_{i=1}^{t} v_i \gg \beta \Rightarrow (m \times u + \sum_{i=1}^{t} v_i) \bmod q = m \times u + \sum_{i=1}^{t} v_i \ll (q-1)/2$

Case of $m \leq -1$. Since $u \gg \sum_{i=1}^{t} v_i \gg \alpha$ and $-\alpha \times u + \sum_{i=1}^{t} v_i \gg -(q-1)/2$, we have

$-\alpha \gg m \times u + \sum_{i=1}^{t} v_i \geq -\alpha \times u + \sum_{i=1}^{t} v_i \gg -(q-1)/2 \Rightarrow (m \times u + \sum_{i=1}^{t} v_i) \bmod q = q - m \times u - \sum_{i=1}^{t} v_i \gg q - (q-1)/2 = (q+1)/2$

Thus, we have $m \geq 0 \Rightarrow (m \times u + \sum_{i=1}^{t} v_i) \bmod q \ll (q-1)/2$ and $m \leq -1 \Rightarrow (m \times u + \sum_{i=1}^{t} v_i) \bmod q \gg (q+1)/2$. Therefore, a data owner can detect whether $m \geq 0$ or not by comparing $\varphi = (m \times u + \sum_{i=1}^{t} v_i) \bmod q$ to $(q-1)/2$.

V. Privacy-preserving Outsourced Mining

In this section, we present our privacy-preserving association rule mining and frequent itemset mining solutions using the homomorphic encryption scheme and secure comparison scheme proposed in Section IV as building blocks.

A. Main Idea

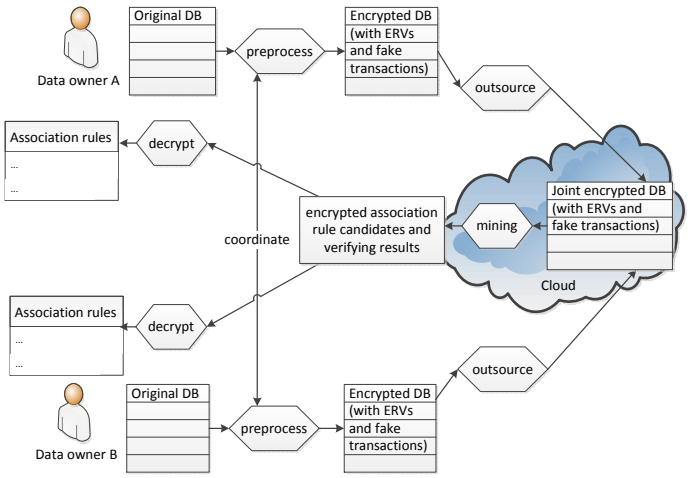

Fig. 3: Privacy-preserving outsourced association rule mining

As shown in Fig. 3, in our association rule mining solution, each data owner owns a private database, and data owners collaboratively mine their joint database's association rules with the assistance of the cloud. Our association rule mining solution includes two stages, namely: preprocessing and mining.

In the preprocessing stage, data owners and the cloud collaborate to generate an encrypted joint database at the cloud's end and some auxiliary data for privacy-preserving mining. Each data owner inserts fictitious transactions to his private database, and encrypts items in the database with a substitution cipher. The fictitious transactions are used to mitigate frequency analysis attacks (due to the inherent weakness of the substitution cipher). Once the databases have been encrypted, they are outsourced to the cloud as part of the joint database maintained by the cloud. To allow the cloud to accurately mine the database (which has fictitious transactions), data owners tag each transaction in their outsourced databases and joint database with an encrypted realness value (ERV) using our customized homomorphic encryption scheme. A realness value (RV for short) is either 0 or 1, which indicates that

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

the transaction is fictitious or real, respectively. All ERVs are sent to the cloud. Please note that the cloud is still unable to determine whether a transaction is fictitious or not, even having ERVs.

In the mining stage, the cloud mines association rules for data owners in a privacy-preserving manner. The cloud mines association rule candidates from the encrypted joint database. Because of the existence of fictitious transactions, some candidates will be "false positives". To allow data owners detecting false positives, the cloud verifies candidates in a privacy-preserving manner. The cloud computes each candidate's encrypted verifying result from the ERVs, utilizing our homomorphic encryption and secure comparison schemes. The cloud returns all candidates and their encrypted verifying results to the data owners. Finally, data owners decrypt the encrypted verifying results and association rule candidates to recover the real association rules.

The main idea of our frequent itemset mining solution is similar, and the only differences are in the mining stage. In the mining stage, the cloud mines frequent itemset candidates (i.e. the seemingly frequent itemsets are defined later) instead of association rule candidates. The data owners then decrypt the encrypted verifying results and frequent itemset candidates to recover the real frequent itemsets.

B. Frequent Itemset Mining Solution

We describe our frequent itemset mining solution in the t-data-owner setting below, and an example is shown in Tables II, III and IV.

Preprocessing stage:

1. *Initialization for homomorphic encryption*
   Let $D_1, D_2 ... D_t$ be the data owners. A data owner, say D_1, runs $KeyGen(\lambda)$ to generate a secret key SK and a public parameter p of the proposed homomorphic encryption scheme. p is shared with other data owners and the cloud, while SK is shared only with data owners.
   To use the proposed homomorphic encryption and outsourced secure comparison schemes in our solutions, the bit lengths of keys and parameters must be carefully selected based not only on the security parameter λ but also estimated maximum ciphertext degree and joint database size. The selection rules are shown in Table V. D_1 will select these bit lengths, and the other data owners will verify whether the lengths satisfy the selection rules.
2. *Initialization for secure threshold comparison*
   To enable outsourced secure comparison, each data owner computes a 1-degree ciphertext of "1", and sends it to the cloud. Let μ_i be the ciphertext generated by the i-th data owner.
   A data owner, say D_1, computes $c_s = E(SK, -T_s \bmod q, 1)$ and $c_e = E(SK, 1, 1)$. The owner sends c_s, c_e along with T_s to the cloud. To prevent D_1 deviating from the cooperative mining protocol, the cloud sends the received c_s, c_e and T_s to other data owners for correctness verification.
3. *Each data owner hides data item frequencies by inserting fictitious transactions to his private database.*
   [16]'s algorithm is used to insert fictitious transactions (see Appendix A). After inserting the fictitious transactions, each item shares the same frequency with at least $k-1$ other items in the same private database. The higher the value of k, the harder it is for the cloud to launch a frequency analysis attack. Data owners need to agree on k's value. Data owners exchange their desirable values of k, and the highest value will be used for all private databases.
4. *Each data owner tags his private database's transactions with* 1*-degree ERVs.*
   If a transaction is fictitious, its RV is 0. Otherwise, the RV is 1. The homomorphic encryption scheme proposed in Section IV is used to encrypt RVs to obtain ERVs. We remark that any two ERVs are different even if their plaintexts are the same because of the probabilistic property of the encryption scheme. Therefore, the cloud cannot determine whether any two ERVs share the same plaintext or not.
5. *Each data owner encrypts items in his private database with a substitution cipher.*

TABLE II: Original databases (before preprocessing stage)

A's database

TID	transaction
1	A1 A3
3	A2 A4
4	A3 A4
8	A1
9	A3

B's database

TID	transaction
1	B1 B2 B4
3	B1 B2
5	B3
8	B2
9	B4

TABLE III: Preprocessed databases for outsourcing (after step 5 of preprocessing stage)

A's database

TID	transaction	ERV
H(1)	S(A1) S(A3)	E(1)
H(2)	*S(A1)*	E(0)
H(3)	S(A2) S(A4)	E(1)
H(4)	S(A3) S(A4)	E(1)
H(6)	*S(A2)*	E(0)
H(8)	S(A1)	E(1)
H(9)	S(A3)	E(1)

B's database

TID	transaction	ERV
H(1)	S(B1) S(B2) S(B4)	E(1)
H(2)	*S(B4)*	E(0)
H(3)	S(B1) S(B2)	E(1)
H(4)	*S(B3)*	E(0)
H(5)	S(B3)	E(1)
H(8)	S(B2)	E(1)
H(9)	S(B4)	E(1)

H(): cryptographic hash function
S(): encryption with a substitution cipher
E(): probabilistic homomorphic encryption function
$k = 2$ and fictitious transactions are in italics.
Supp(S(A1))=Supp(S(A3))=3; Supp(S(A2))=Supp(S(A4))=2;
Supp(S(B1))=Supp(S(B3))=2; Supp(S(B2))=Supp(S(B4))=3;
If concealing original TIDs from the cloud is required, their hash values will replace them as the TIDs in encrypted databases.

6. *Database outsourcing*.
   Each data owner sends his encrypted database along with ERVs to the cloud, and the cloud joins received transactions by TIDs to create a joint database. Note that a transaction in a data owner's private database is a transaction partition of data owners' joint database.
7. *Aggregated verification of ERVs.*
   Let Θ be the set of all ERVs. The cloud computes $\Gamma = \sum_{c \in \Theta} c \bmod p$, and sends Γ to all data owners. Because of the homomorphic property of the encryption scheme, Γ is

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

TABLE IV: Joint database in the cloud (after preprocessing stage)

TID	A's partition	B's partition	ERV	
			A's DB	B's DB
H(1)	S(A1) S(A3)	S(B1) S(B2) S(B4)	E(1)	E(1)
H(2)	*S(A1)*	*S(B4)*	E(0)	E(0)
H(3)	S(A2) S(A4)	S(B1) S(B2)	E(1)	E(1)
H(4)	S(A3) S(A4)	*S(B3)*	E(1)	E(0)
H(5)	–	S(B3)	–	E(1)
H(6)	*S(A2)*	–	E(0)	–
H(8)	S(A1)	S(B2)	E(1)	E(1)
H(9)	S(A3)	S(B4)	E(1)	E(1)

Suppose $T_s = 2$ and $T_c = 0.8$. The mining result is as below.

Seemingly frequent itemsets with two or more items: S(A1) S(B2);S(A1) S(B4); S(A3) S(B4); S(B1) S(B2)

Association rule candidates: $S(A1) \Rightarrow S(B2)$; $S(B2) \Rightarrow S(A1)$; $S(A1) \Rightarrow S(B4)$; $S(B4) \Rightarrow S(A1)$; $S(A3) \Rightarrow S(B4)$; $S(B4) \Rightarrow S(A3)$; $S(B1) \Rightarrow S(B2)$; $S(B2) \Rightarrow S(B1)$;

Real association rules: $S(A1) \Rightarrow S(B2)$; $S(B4) \Rightarrow S(A3)$; $S(B1) \Rightarrow S(B2)$

the ciphertext of all RVs' sum, and Γ's random ingredient is the sum of all ERVs' random ingredients. Every data owner decrypts Γ to verify the bit length of random ingredient (explained in Section VI-B).

Mining stage:

1. *The cloud runs a classic frequent itemset mining algorithm for centralized database named Eclat [7] to find out all frequent itemsets of the joint database*.
   As the joint database contains fictitious data, an itemset's real support is the same as or lower than its support seen by the cloud. Therefore, a "frequent" itemset located here may not be real frequent itemset. Therefore, we refer to the frequent itemsets located here as "*seemingly frequent itemsets*", which contain all real frequent itemsets.
   The Eclat algorithm [7] is chosen over other classic algorithms here because it can generate the TID sets required in the next step as a byproduct.
2. *The cloud computes the encrypted support for each seemingly frequent itemset*.
   An itemset X's encrypted support, $E(Supp(X))$, is computed using the ERVs of the transactions containing X. The TID set of such transactions is generated in the previous step as a byproduct. Let the set $V(X)$ be the indices of transactions containing X, and the set $M(X)$ be the indices of the data owners involving X. (All items in X are from these data owners, and each of these owners has at least one item in X.) Let $ERV_{i,j}$ be the ERV for data owner D_j's partition of the i-th transaction. The i-th transaction, which may contain fictitious data, truly contains X if $ERV_{i,j}$ is a ciphertext of "1" for every $j \in M(x)$. Due to the properties of the proposed homomorphic encryption scheme, the cloud can compute

$$E(Supp(X)) = (\sum_{i \in V(x)} \prod_{j \in M(x)} ERV_{i,j}) \bmod p$$

   without knowing the plaintexts. In the above equation, $\prod_{j \in M(x)} ERV_{i,j} \bmod p$ is a ciphertext of "1" if and only if each of the data owners $M(X)$ has a real partition in the i-th transaction. Otherwise, it is a ciphertext of "0".
3. *For each seemingly frequent itemset, the cloud verifies whether it is real frequent or not in a privacy-preserving manner, and computes its ESVR.*
   Suppose Z is such an itemset. Recall that $E(Supp(Z))$ has been computed in the previous step. Utilizing the homomorphic property of our encryption scheme, the cloud computes

$$E(Supp(Z) - T_s) = (E(Supp(Z)) + c_s) \bmod p$$

   and compares $Supp(Z) - T_s$ with 0 using our secure comparison scheme. The (encrypted) comparison result is the ESVR.
   Note: As $Supp(Z) - T_s \geq 0 \Leftrightarrow Supp(Z) \geq T_s$, the data owners can decrypt Z's ESVR to determine whether Z is a real frequent or not.
4. *The cloud returns seemingly frequent itemsets and their ESVRs to involved data owners.*
5. *The data owners decrypt the received seemingly frequent itemsets' ESVRs to determine the real frequent itemsets, and decrypt the revealed real frequent itemsets (encrypted with a substitution cipher).*
   For each seemingly frequent itemset, a data owner first decrypts its ESVR to determine whether the itemset is really a real frequent. If it is real frequent, then the data owner will decrypt it.

In the first step of preprocessing stage, using exact maximum ciphertext degree and joint database size to select bit lengths could reduce ciphertext size. These exact values could be obtained from the cloud after the first step of mining stage. The maximum ciphertext degree in our solutions is the maximum number of data owners involved in one seemingly frequent itemset.

To reduce ciphertext size, data owners can defer the step of initialization for homomorphic encryption until the exact values are obtained. Subsequently, ERV generation and initialization for secure comparison will also be deferred. The only drawback is an increase in the communication rounds required to obtain the above exact values and to send ERVs to the cloud separately. (Currently, ERVs are sent along with the databases.)

C. Association Rule Mining Solution

Based on the above frequent itemset mining solution, we can build a cloud-aided privacy-preserving association rule mining solution for vertically partitioned databases. The solution is given below.

1. *All data owners and the cloud run the above frequent itemset mining solution to mine frequent itemsets.*
   Eventually, the cloud finds out all seemingly frequent itemsets, and obtains encrypted supports and ESVRs. If data owners only want to learn the association rules, without learning the frequent itemsets, the cloud does not need to return these itemsets and ESVRs in the running. In addition, data owners do not need to decrypt frequent itemsets.
2. *Initialization for secure threshold comparison*

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

A data owner, say D_1, computes $c_z = E(SK, 0)$, and sends c_z along with $n1$ and $n2$ to the cloud where $n1$ and $n2$ are two integers and $n_2/n_1 = T_c$. To prevent D_1 deviating from the cooperative mining protocol, the cloud sends these received ciphertexts and plaintexts to other data owners for correctness verification.

To reduce communication rounds, this step can be performed together with the "initialization for secure threshold comparison" step in frequent itemset mining.

3. *The cloud generates association rule candidates*.
   An association rule candidate $X \Rightarrow Y$ that satisfies $X \cap Y = \emptyset$ and $X \cup Y$, where X, Y and $X \cup Y$ are seemingly frequent itemsets.
4. *For each association rule candidate, the cloud verifies its confidence in a privacy-preserving manner, and computes its ECVR (Encrypted Confidence Verifying Result)*.
   For any association rule candidate $X \Rightarrow Y$, both $X \cup Y$ and X are seemingly frequent itemsets. Using their encrypted supports computed earlier, the cloud computes

$$\begin{aligned} &E(Supp(X \cup Y) \times n_1 - Supp(X) \times n_2) \\ =&(E(Supp(X \cup Y)) \times n1 + (c_z - E(Supp(X))) \times n2) \bmod p \end{aligned}$$

   and compares $Supp(X \cup Y) \times n_1 - Supp(X) \times n_2$ with 0 using our secure comparison scheme. The encrypted comparison result is the ECVR.

   Note: From (1), we observe that the ECVR can be used to determine whether the confidence is over T_c or not.

$$\begin{aligned} &Supp(X \cup Y) \times n_1 - Supp(X) \times n_2 \geq 0 \\ \Leftrightarrow &Supp(X \cup Y)/Supp(X) \geq n_2/n_1 \\ \Leftrightarrow &Conf(X \Rightarrow Y) \geq T_c \end{aligned} \quad (1)$$

5. *The cloud returns association rule candidates, ESVRs and ECVRs to involved data owners*.
6. *The data owners decrypt association rule candidates, ESVRs and ECVRs to find out real association rules*.
   For a candidate $X \Rightarrow Y$, the data owners will need to first decrypt the ESVR of $X \cup Y$ to determine whether $Supp(X \cup Y) \geq T_s$ or not. If yes, the data owners will decrypt the ECVR of $X \Rightarrow Y$ to determine whether $Conf(X \Rightarrow Y) \geq T_c$ or not. If yes, the data owners will then decrypt X and Y to recover the real association rule in plaintext (as items in the joint database are encrypted with a substitution cipher).

VI. Security Analysis

In this section, we analyze the security properties of the proposed solutions, focusing on how our solutions can protect a data owner's data from the cloud and the other data owners.

A. Security under the Cloud's Attacks

Confidentiality of transactions under frequency analysis attack. In our solutions, items are encrypted with a substitution cipher. Recall that substitution cipher is subject to frequency analysis attacks. To counter such an attack, item frequency is hidden by adding fictitious transactions to data owners' private databases based on [16]'s algorithm. In contrast to the approach in [16], we apply this algorithm in vertically partitioned databases instead of a single database, and we tag transactions with ERVs. These differences will not, however, undermine the security under a frequency analysis attack, as explained below.

(1) Applying this algorithm in vertically partitioned databases will not undermine the security. Data owners' encrypted databases are uploaded to the cloud, and these databases do not share any items. So cracking one of these databases by item frequency analysis is independent of other databases. Then, from Theorem 4 of [16], we know that the crack probability for an item or itemset in such a database is no more than $1/k$. Now, let us analyze the security of the joined database. The cloud joins the above-mentioned databases. Let X be an itemset of the joined database, and suppose X's item(s) are from b data owner(s). Then, X can be divided into b itemset(s), and each has only one data owner's item(s). To crack X, the attacker has to crack all b itemset(s). Because cracking one of these itemsets is independent of the other itemsets, the probability of cracking X is no more than $(1/k)^b$. As $t \geq b \geq 1$, the crack probability for an item or itemset in our solutions is still no more than $1/k$.

(2) ERVs will not undermine the security. The use of ERVs cancels out fictitious transactions in the mining result, but the cloud cannot detect fictitious transactions and reveal real item frequency based on ERVs. ERVs are encrypted with a probabilistic encryption scheme, and any two ERVs are different even if their plaintexts are the same. Therefore, the cloud cannot distinguish whether a transaction is real or not from its ERV. The cloud is not able to tell whether any two given transactions share the same realness value. Without real item frequency, the cloud cannot launch frequency analysis attack on the substitution cipher. Without knowing the plaintexts of encrypted items, the cloud cannot learn any sensitive information about the transaction data and mining result of the outsourced database.

Confidentiality of TIDs. The original TIDs of some databases may contain sensitive information. To hide such information, the TIDs in the outsourced databases are replaced by the hash values of the original TIDs. Because of the pre-image resistance property of cryptographic hash function, the cloud cannot recover original TIDs from the TIDs used in the outsourced databases.

B. Security under Data Owners' Attacks

Attack via chosen random ingredients. In our solutions, data owners are not required to exchange any plaintext or ciphertext (i.e., encryption of their private data). In addition, due to the use of the underlying homomorphic encryption scheme and secure comparison scheme, the exact supports and confidences are concealed from any data owner. However, there is still a possible attack via chosen random ingredients. A data owner may use ERVs' random ingredients to tag some transactions, and verify whether such a transaction contains a given itemset or not. For example, the data owner could compute an ERV with a customized random ingredient. The random ingredient's bit length is longer than any other random

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

ingredient's in the preprocessing stage. For each received seemingly frequent itemset, the data owner decrypts its ESVR, and obtains the ESVR's random ingredient. If the random ingredient matches the pattern of the customized random ingredient's most significant bits, it is an indication that the ERV with the customized random ingredient has been used to compute the ESVR. Thus, the data owner knows the transaction associated with the ERV contains the itemset.

To prevent such an attack, we carefully configure the bit lengths of random ingredients. An example is presented in Table V. The random ingredients in $\mu_1, \mu_2 ... \mu_t$ are used to mask chosen random ingredients in ERVs. Also, in the preprocessing stage, each data owner must verify c_s, c_z, c_e and that ERVs are generated honestly with random ingredients of the right bit lengths. In the last step of preprocessing stage, the random ingredients of ERVs are verified in an aggregated manner. By examining the sum of these random ingredients, data owners can be assured that any chosen random ingredient will be masked.

C. Security of Underlying Homomorphic Encryption Scheme

The security of underlying homomorphic encryption scheme depends on the hardness of solving nonlinear systems.

In the proposed mining solutions, a number of plaintexts and their corresponding ciphertexts (i.e. c_s, c_z, $\{\mu_i : t \geq i \geq 1\}$) are disclosed to the cloud. Therefore, the underlying homomorphic encryption scheme should be secure under known-plaintext attacks. In our homomorphic encryption scheme, to encrypt a plaintext m_i and obtain the corresponding ciphertext c_i, we require the use of two secrets q, s and a random ingredient r_i. From a known (m_i, c_i) pair, the attacker can get a nonlinear equation of three unknowns q, s, r_i:

$$s(q \times r_i + m_i) = c_i$$

From w known pairs, the attacker can generate an underdetermined nonlinear system of w equations, and the number of unknowns are $w + 2$. If the attacker can solve this system, he can learn the secret key of the encryption scheme.

Security can be achieved by increasing hardness of solving above nonlinear system. Solving underdetermined nonlinear systems is NP-hard, while solving overdetermined systems can be done in polynomial time. The attacker may attack a nonlinear system by guessing the values of some unknowns [25] if guessing the correct values is not very hard. The attacker can generate many overdetermined systems from the targeted underdetermined nonlinear system by fixing some unknowns (3 unknowns for the system above) to all possible values. These unknowns will be viewed as constants in the generated systems. By solving the generated overdetermined systems, the attacker finds the solution to the targeted underdetermined nonlinear system. To prevent such attacks, we can configure large ranges or bit lengths for q, s, r_i to compound the challenges of guessing the correct values. An example of the configuration is presented in Table V.

D. Security of Underlying Secure Comparison Scheme

The proposed secure comparison scheme is used in our solutions to conceal the exact support values and confidence values from the data owners. From the correctness proof in Section IV-C, we know that

$$m_i \geq 0 \Rightarrow (q-1)/2 \gg \varphi \gg \beta$$
$$m_i \leq -1 \Rightarrow q - \alpha \gg \varphi \gg (q+1)/2$$

If $\varphi < (q-1)/2$, m could be any value in the range $[0, \beta]$. Otherwise, m could be any value in $[-\alpha, 0]$. By observing φ, a data owner knows whether $m \geq 0$ or not, but m's range cannot be deduced. As our design goal is to conceal exact supports, it is sufficient to conceal m's value in many possible values. In other words, we do not need a secure comparison scheme to conceal more information.

VII. Performance Evaluation

In this section, we evaluate the computational complexity, communication complexity and storage cost of our association rule mining and frequent itemset mining solutions. In the evaluation, we choose one of [13]'s solutions and classic non-privacy-preserving algorithms as the baseline. The former is chosen because it and our solutions achieve the same privacy level. In contrast, other solutions achieve lower privacy levels. Classic algorithms are chosen as baselines because they are the most efficient known solutions.

A. Computational Complexity

1) Comparing with Classic Algorithms: We used the running time to evaluate the computational complexity. To demonstrate the feasibility of our solutions, we compare our solutions with classic non-privacy-preserving algorithms, which are the most efficient known solutions.

We evaluated our solutions and three classic non-privacy-preserving algorithms (Apriori, Eclat and FP-growth) using two datasets (retail and pumsb) from [26]. The retail dataset from Tom Brijs contains anonymous retail market basket data in a Belgian retail store, while the pumsb dataset contains census data for population and housing. The retail and pumsb datasets contain 88,162 and 49,046 transactions, respectively. To simulate t data owners, we vertically partitioned each dataset into t databases randomly.

Our solutions are implemented in JAVA, and we use a JAVA implementation [27] of Apriori, Eclat and FP-growth algorithms in our experiments. All implementations are single-threaded implementations. To ensure a fair comparison, in all experiments, machines playing the roles of cloud or data owner have the same hardware and software settings. The settings and the parameters of our solutions are shown in Table V.

The results are shown in Figs. 4, 5 and 6. As the running time of association rule mining is only slightly higher than that of frequent itemset mining, only the results of association rule mining are presented. We show our solutions' running time at the cloud's end (i.e. running time of mining) and data owner side (i.e. running time of preprocessing and decrypting) separately. As expected, our solutions are not as

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

TABLE V: Experimental settings

CPU	Intel i7 2620M (dual-core 2.7 GHz)	
memory	8 GB	
software	Windows 7 64bit and Java SE 8	
λ	80	λ is a security parameter.
plaintext bit length	< 72 bits	
$\|r\|$ for c_s, c_e and ERVs	$= \lambda/2 = 40$ bits	
$\|r\|$ for c_z	64 bits	Must be larger than $\lambda/2 + \|n'\|$.
$\|r\|$ for μ_i	120 bits	Must be larger than $\lambda/2 + \|r\|$ for $c_z + 10$
$\|q\|$	80 bits	Must be larger than plaintext bit length and not smaller than λ.
$\|p\|$	$120d_{max} + 88$ bits	Must be larger than $\|q\|d_{max} + \lambda(d_{max} - 1)/2 + \|r\|$ for μ_i

$|\ \ |$: bit length operator.
n' is the size of the joint database with fictitious transactions.
$|r|$ could be smaller than λ as an attacker needs to search 3 random ingredients exhaustively.
d_{max} is the maximum ciphertext degree, which is less than t.

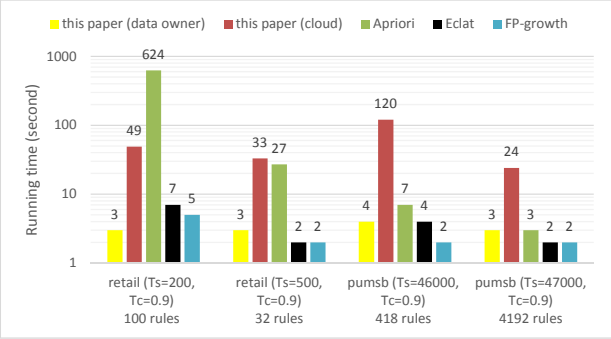

Fig. 4: Running time comparison ($t = 4$ and $k = 12$)

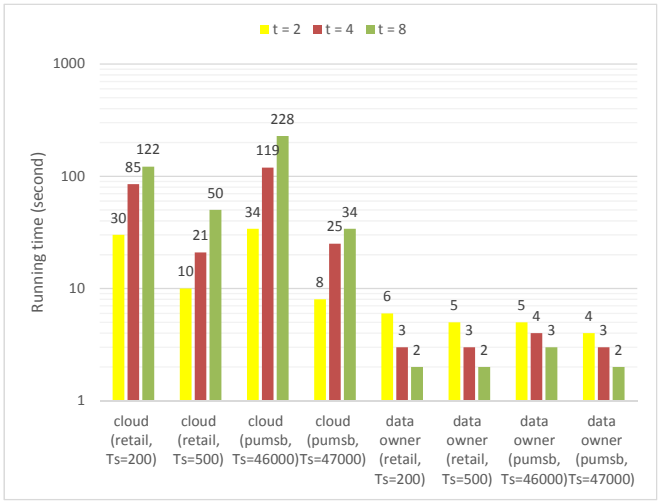

Fig. 5: Running time under different data owner count t (k is fixed to 16)

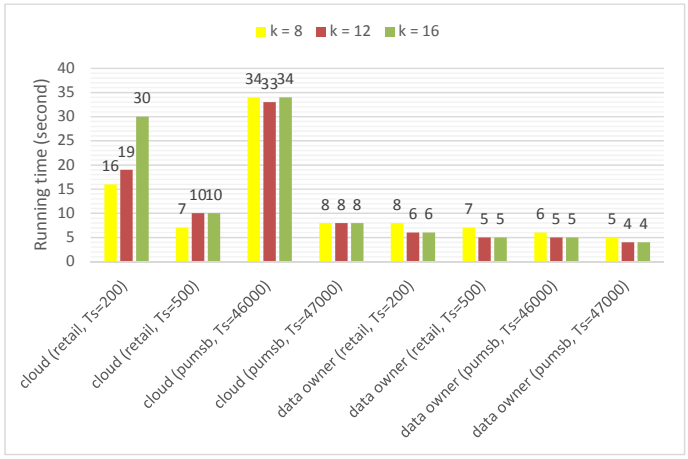

Fig. 6: Running time under different k (t is fixed to 2)

efficient as the most efficient algorithms / solutions of low privacy levels. However, they achieve a higher privacy level with an acceptable running time. Compared with the fastest algorithm's running time, the cloud's running time is about one order higher for most cases, while data owner's running time is very low. This is the classic trade-off between privacy-preserving and efficiency.

From Figs. 5 and 6, we also observe that running time changes with increasing values of k and t. The cloud's running time increases with t, as d_{max} increases with t and a larger d_{max} results in larger ciphertext size and more computations. The cloud's running time increases with k for the retail dataset, but barely changes for the pumsb dataset. The increase in running time for retail dataset is due to the increase in fictitious data. However, the pumsb dataset is very dense, and the supports are already very high without including fictitious data. Thus, adding more fictitious data hardly changes the number of seemingly frequent itemsets and their supports. We can also observe that data owner's running time decreases when t increases. The reason is simple. If the same joint database is vertically partitioned to more data owners, each data owner's dataset is smaller. Preprocessing a smaller dataset requires less time. Data owner's running time doesn't increase with k either. Such a phenomenon can also be explained using [16]'s algorithm for adding fictitious transactions. Specifically, data owner's running time is dominated by the time to run this algorithm, which is hardly affected by changes in the values of k. In summary, increasing t and k usually results in a higher running time at the cloud end, without resulting in an increase in data owner's running time.

2) Comparing with the solution achieving the Same Privacy Level: To the best of our knowledge, the only existing privacy-

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

preserving solution that does not leak sensitive information of the raw data is one of [13]'s frequent itemset mining solutions (hereafter referred to as "[13]'s strong solution" in this paper). This solution cannot be used for association rule mining, whilst we have solutions for both association rule mining and frequent itemset mining. Similar to our solutions, [13]'s strong solution uses homomorphic encryption. However, it needs to use asymmetric homomorphic encryption scheme, which is computationally expensive. [13]'s strong solution requires about $n \times F$ encryptions as well as $(n + T_s) \times F$ homomorphic additions and scalar multiplications. F is the number of frequent itemsets. This solution's running time is dominated by these expensive operations. We evaluate its running time by estimating the required time to undertake the operations – see Table VI. To estimate the running time, we measure the speeds of two popular asymmetric additive homomorphic encryption schemes (i.e., Paillier and ElGamal) implemented in Java, and we remark that ElGamal is the scheme suggested by [13]. The speeds are measured using a machine of the same specifications in our evaluations (see Table V), and so are the datasets and parameters. Compared with the result of our solution in Fig. 4, the running time of [13]'s strong solution is several orders of magnitude higher.

TABLE VI: Estimated running time of [13]'s solution

	retail $T_s = 500$	retail $T_s = 200$	pumsb $T_s = 47000$	pumsb $T_s = 46000$
Paillier	77 hours	1233 hours	30 hours	128 hours
ElGamal	43 hours	691 hours	17 hours	73 hours

The key length is 1024 bits, which is in favor of [13]'s solution.
The ElGamal scheme for estimation uses the additive homomorphic variant [15].

B. Communication Complexity and Storage Cost

Similar to most other solutions, our solutions require constant communication rounds. In the preprocessing stage, some keys and parameters are shared among t data owners. These keys and parameters can be sent to data owners in parallel. In the mining stage, all mining results can be sent to data owners in a communication round. Therefore, the number of communication rounds does not grow if t, frequent itemsets, or association rules increases.

TABLE VII: Transaction count of joint database (retail)

	$t = 2$	$t = 4$	$t = 8$
$k = 8$	202997	218400	245862
$k = 12$	269132	285996	315996
$k = 16$	335789	354743	383776

Original transaction count without fictitious data: 88162

TABLE VIII: Transaction count of joint database (pumsb)

	$t = 2$	$t = 4$	$t = 8$
$k = 8$	98692	108139	118648
$k = 12$	108223	122767	162697
$k = 16$	120301	156828	211529

Original transaction count without fictitious data: 49046

The communication traffic and storage cost in our solutions are dominated by the joint database size and the size of all ERVs. Let n and n' be the transaction count of the joint database with and without fictitious transactions, respectively. Tables VII and VII list the joint database size of two datasets under different settings of k and t. The joint database in our solutions contains fictitious transactions, while the one in class algorithms does not. We can observe from Tables VII and VII that the joint database size in our solutions is a few times larger and the size grows with k and t.

The number of ERVs is at most $n' \times t = O(n \times t)$. A ciphertext's size is $O(\lambda \times d_{max}) \leq O(\lambda \times t)$, where λ is the security parameter (typically 80). Normally, d_{max} can be viewed as a small constant. d_{max} is not larger than 4 for most settings in the above experiments, and the largest d_{max} observed in the above experiments is 6. Let m be the average transaction size. Then the communication traffic and storage cost in our solutions are both $O(n \times m + n \times \lambda \times t \times d_{max})$. The traffic and cost in classic algorithms are $O(n \times m)$. For most reasonable transaction sizes, our solutions resulted in an increase in communication traffic and storage cost only by a few times. The storage cost in [13]'s strong solution is $O(n \times m)$, while the communication traffic is $O(T_s \times t \times F \times C)$ where F is the number of frequent itemsets. C is the ciphertext size in [13]'s strong solution, which is at least 2048 bits. In many settings, our solutions' traffic is not any higher than the traffic of [13]'s strong solution.

VIII. Related Work

Privacy-preserving Association Rule Mining and Frequent Itemset Mining on Vertically Partitioned Databases. In [9], the first work to identify and address privacy issues in vertically partitioned databases, a secure scalar product protocol is presented and used to build a privacy-preserving frequent itemset mining solution. Association rules can then be found given frequent itemsets and their supports. Since the publication of this seminal work, a number of privacy-preserving association rule mining or frequent itemset mining solutions have been published in the literature (see [11], [12], [13], [28], [29], [30], [31]).

The most relevant work is the privacy-preserving association rule mining solution presented in [11]. In this solution, a data owner known as the master is responsible for the mining. The other data owners (known as slaves) insert fictitious transactions to their respective datasets, and send the datasets to the master. Each data owner will also send his set of real transactions' IDs to a semi-trusted third-party server. The third-party server is assumed not to be colluding with any data owner, but it cannot be trusted to hold the raw data. The master generates association rule candidates from the joint database containing fictitious data. For each rule candidate $X \Rightarrow Y$, the master sends the ID lists of the transactions containing $X \cup Y$ and the transactions containing X to the third-party server. The server verifies if the rule is qualified or not. Similar to our solutions, a semi-trusted third-party is utilized for the mining. However, unlike our solutions, a data owner (i.e. the master) does the majority of the computational

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

work. Therefore, we can hardly say that such a solution is an outsourced mining solution. Though fictitious data are added in datasets to lower data usability, the master is able to learn significant information about other data owners' raw data from the received datasets. In contrast, our solutions do not leak such information as we do not rely on one particular data owner to undertake the computations and we also encrypt the datasets.

All existing solutions, with the exception of [11], do not utilize a third-party server to compute the mining result. Some solutions [12], [13] use asymmetric homomorphic encryption to compute the supports of itemsets, while other solutions [9], [28], [29], [30] use a secure scalar product protocol, a set intersection cardinality protocol or a secret sharing scheme to perform these computations. A majority of these solutions expose exact supports to all data owners, resulting in the leakage of information about the data owners' raw data [11]. The only exception is one of [13]'s solutions. In [13], there are two privacy-preserving solutions for frequent itemset mining. The first solution exposes exact supports, which is not desirable. The second solution does not expose exact supports. However, association rules cannot be mined based on the result of second solution because confidences cannot be computed without the exact supports. In addition, this solution's method cannot be used to mine association rules because securely computing confidence is more complicated than computing support. In comparison with this solution, our frequent itemset mining solution's computational complexity is significantly lower. Our solutions do not expose exact supports or confidences to data owners. Different from existing solutions based on homomorphic encryption, we use symmetric homomorphic encryption instead of asymmetric homomorphic encryption, and the manner in which we use homomorphic encryption also differs from existing solutions. In our approach, we use homomorphic encryption to create ERVs and build our secure outsourced comparison scheme.

Privacy-preserving Outsourced Association Rule Mining and Frequent Itemset Mining. Privacy-preserving outsourced frequent itemset mining and association rule mining have been studied in the setting of a single data owner [19], [20], [21], [16]. In existing solutions, the data owner outsources their data and the mining task to the cloud, but at the same time, wish to keep the raw data secret from the cloud. Generally, data items in the database are encrypted using a substitution cipher prior to outsourcing. [19] proposed a solution to counter frequency analysis attack on substitution cipher. However, a later work [20] demonstrated that [19]'s solution is not secure. Giannotti et al. proposed a solution based on k-anonymity frequency [21], [16]. To counter frequency analysis attack, the data owner inserts fictitious transactions in the encrypted database to conceal the item frequency. After inserting the fictitious transactions, any item in the encrypted database will share the same frequency with at least $k-1$ other items. The data owner sends the encrypted database of both the real and fictitious transactions to the cloud. The cloud runs a classic frequent itemset mining algorithm, and returns the result (frequent itemsets and their supports) to the data owner. The data owner revises these itemsets' supports by subtracting them with these itemsets' corresponding occurrence counts in the fictitious transactions respectively. Finally, the data owner decrypts the received itemsets with the revised supports higher than the frequency threshold, and generates association rules based on found frequent itemsets. Our solutions use their techniques to conceal the raw data from the cloud and mitigate frequency analysis attack that can be undertaken by the cloud. Using these techniques alone, however, is not sufficient to protect data privacy in the vertically partitioned database setting. To cancel out fictitious transactions, both [21], [16] require the data owner to count itemset occurrences in fictitious transactions. In the vertically partitioned database setting, data owners are unable to perform such calculation using the techniques described in [21], [16]. In our solutions, the cloud rather than the data owners cancels out fictitious transactions in a privacy-preserving manner, and the underlying techniques are our homomorphic encryption, secure comparison and ciphertext tag schemes.

Another recent work [32] proposed a privacy-preserving outsourced association rule mining solution based on predicate encryption. This solution is resilient to chosen-plaintext attacks on encrypted items, but it is vulnerable to frequency analysis attacks. Applying this solution to vertically partitioned databases will also result in the leakage of the exact supports to data owners. In this paper, our adversary model is different. We assume the cloud has knowledge of the item frequencies instead of chosen plaintext-ciphertext pairs, and our solutions are resilient to frequency analysis attacks.

Other related work. Other than the settings of vertically partitioned databases and cloud/third-party-aided mining, privacy-preserving frequent itemset mining and association rule mining have been studied in the settings of horizontally partitioned databases [10], [33], [34], [35], data publishing [36] and differential privacy [37]. These settings are beyond the scope of this paper.

IX. Concluding Remarks

In this paper, we proposed a privacy-preserving outsourced frequent itemset mining solution for vertically partitioned databases. This allows the data owners to outsource mining task on their joint data in a privacy-preserving manner. Based on this solution, we built a privacy-preserving outsourced association rule mining solution for vertically partitioned databases. Our solutions protect data owner's raw data from other data owners and the cloud. Our solutions also ensure the privacy of the mining results from the cloud. Compared with most existing solutions, our solutions leak less information about the data owners' raw data. Our evaluation has also demonstrated that our solutions are very efficient; therefore, our solutions are suitable to be used by data owners wishing to outsource their databases to the cloud but require a high level of privacy without compromising on performance.

To realize our solutions, an efficient homomorphic encryption scheme and a secure outsourced comparison scheme were presented in this paper. Both schemes have potential usage in other secure computation applications, such as secure data aggregation, beyond the data mining solutions described in this

paper. Demonstrating the utility of the proposed homomorphic encryption scheme and outsourced comparison scheme in other settings will be the focus of future research.

ACKNOWLEDGMENT

The authors would like to thank Quach Vinh Thanh, the Associate Editor, and the three anonymous reviewers for providing constructive and generous feedback. Despite their invaluable assistance, any errors remaining in this paper are solely attributed to the authors. This work is supported by Nanyang Technological University under Grant MOE Tier 1 (M4011450) and J. Shao is supported by NSFC No. 61472364 and NSFZJ No. LR13F020003.

REFERENCES

[1] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, "Using association rules for product assortment decisions: A case study," in *SIGKDD 1999*.

[2] S. E. Brossette, A. P. Sprague, J. M. Hardin, K. B. Waites, W. T. Jones, and S. A. Moser, "Association rules and data mining in hospital infection control and public health surveillance," *Journal of the American medical informatics association*, vol. 5, no. 4, pp. 373–381, 1998.

[3] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," in *WIDM 2001*.

[4] C. Creighton and S. Hanash, "Mining gene expression databases for association rules," *Bioinformatics*, vol. 19, no. 1, pp. 79–86, 2003.

[5] X. Yin and J. Han, "Cpar: classification based on predictive association rules." in *SIAM SDM 2003*.

[6] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *VLDB 1994*.

[7] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372–390, 2000.

[8] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM SIGMOD 2000*.

[9] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *SIGKDD 2002*.

[10] 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.

[11] B. Rozenberg and E. Gudes, "Association rules mining in vertically partitioned databases," *Data & Knowledge Engineering*, vol. 59, no. 2, pp. 378–396, 2006.

[12] J. Zhan, S. Matwin, and L. Chang, "Privacy-preserving collaborative association rule mining," in *DBSEC 2005*.

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

[14] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT 1999*.

[15] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[16] F. Giannotti, L. 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.

[17] B. Dong, R. Liu, and W. H. Wang, "Result integrity verification of outsourced frequent itemset mining," in *Data and Applications Security and Privacy XXVII - 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Newark, NJ, USA, July 15-17, 2013. Proceedings*, 2013, pp. 258–265. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39256-6_17

[18] R. Liu and W. H. Wang, "Result integrity verification of outsourced privacy-preserving frequent itemset mining," in *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015*, 2015, pp. 244–252. [Online]. Available: http://dx.doi.org/10.1137/1.9781611974010.28

[19] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "Security in outsourcing of association rule mining," in *VLDB 2007*.

[20] I. Molloy, N. Li, and T. Li, "On the (in) security and (im) practicality of outsourcing precise association rule mining," in *ICDM 2009*.

[21] F. Giannotti, L. V. Lakshmanan, A. Monreale, D. Pedreschi, and H. W. Wang, "Privacy-preserving data mining from outsourced databases," in *CPDP 2011*.

[22] National Institute of Standards and Technology, "Fips publication 180-1: Secure hash standard," 1995.

[23] ——, "Fips publication 180-2: Secure hash standard," 2002.

[24] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985. [Online]. Available: http://dx.doi.org/10.1109/TIT.1985.1057074

[25] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, "Efficient algorithms for solving overdefined systems of multivariate polynomial equations," in *EUROCRYPT*. Springer, 2000, pp. 392–407.

[26] Real-life datasets in SPMF format, http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php.

[27] SPMF (Sequential Pattern Mining Framework), http://www.philippe-fournier-viger.com/spmf/.

[28] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *Journal of Computer Security*, vol. 13, no. 4, pp. 593–622, 2005.

[29] X. Ge, L. Yan, J. Zhu, and W. Shi, "Privacy-preserving distributed association rule mining based on the secret sharing technique," in *SEDM 2010*.

[30] R. Kharat, M. Kumbhar, and P. Bhamre, "Efficient privacy preserving distributed association rule mining protocol based on random number," in *Intelligent Computing, Networking, and Informatics*. Springer, 2014, pp. 827–836.

[31] C. Dong and L. Chen, "A fast secure dot product protocol with application to privacy preserving association rule mining," in *Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I*, 2014, pp. 606–617. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-06608-0_50

[32] J. Lai, Y. Li, R. H. Deng, J. Weng, C. Guan, and Q. Yan, "Towards semantically secure outsourcing of association rule mining on categorical data," *Information Sciences*, vol. 267, pp. 267–286, 2014.

[33] T. Fukasawa, J. Wang, T. Takata, and M. Miyazaki, "An effective distributed privacy-preserving data mining algorithm," in *IDEAL 2004*.

[34] C. Su and K. Sakurai, "A distributed privacy-preserving association rules mining scheme using frequent-pattern tree," in *ADMA 2008*.

[35] M. G. Kaosar, R. Paulet, and X. Yi, "Secure two-party association rule mining," in *ACSW-AISC 2011*.

[36] J. Lin and J. Y. Liu, "Privacy preserving itemset mining through fake transactions," in *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007*, 2007, pp. 375–379. [Online]. Available: http://doi.acm.org/10.1145/1244002.1244092

[37] B. N. Keshavamurthy, A. M. Khan, and D. Toshniwal, "Privacy preserving association rule mining over distributed databases using genetic algorithm," *Neural Computing and Applications*, pp. 1–14, 2013.

APPENDIX A
INSERTING FICTITIOUS TRANSACTIONS ([16]'S ALGORITHM)

An algorithm to counter frequency analysis attacks on the outsourced database encrypted with a substitution cipher was proposed in [16]. For the purpose of concealing the item frequency, this algorithm inserts fictitious transactions in the database to be oursourced. The goal is to ensure that each item share the same frequency with at least $k-1$ items. The algorithm is summarized as follows (also see [16]).

- Firstly, the data owner scans the database to count each individual item's support.
- Secondly, the data owner groups items considering the supports and co-occurrence of items.
  The data owner sorts items in decreasing order of support. Starting from the first of the sorted item list (i.e. the item with the highest support), the data owner assigns every k adjacent items to a new created group. If there are less

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/TIFS.2016.2561241, IEEE Transactions on Information Forensics and Security

than k unassigned items remaining, these items will be assigned to the last created group. The data owner swaps items from different groups to ensure that all items in the same group do not occur together in the same transaction.

- Thirdly, for each item in each group, the data owner calculates the difference between the item's support and the highest support in the group.
  The difference is defined as the "noise" of the item.
- Fourthly, to achieve k-anonymity frequency, the data owner generates fictitious transactions based on the result of the third step.
  The number of an item's occurrences in the fictitious transactions is equal to its noise calculated in the third step. After inserting the fictitious transactions, all items in the same group share the same support.

Lichun Li received the PhD. degree in computer science from Beijing University of Posts and Telecommunications in 2009, master degree in communication and information systems in China Academy of Telecommunication Technology in 2006, and bachelor degree in information engineering in in Beijing University of Posts and Telecommunications in 2002. He is currently a Postdoctoral Research Fellow with the INFINITUS laboratory, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include privacy and security in cloud and big data.

Rongxing Lu (S'09-M'11-SM'15) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. From May 2012 to April 2013, he was a Postdoctoral Fellow with the University of Waterloo. Since May 2013, he has been an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include computer network security, mobile and wireless communication security, and applied cryptography. Dr. Lu was the recipient of the Canada Governor General Gold Metal.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He is currently a cloud technology endowed associate professor at University of Texas at San Antonio, an associate professor at the University of South Australia, and a guest professor at China University of Geosciences. He was named one of 10 Emerging Leaders in the Innovation category of The Weekend Australian Magazine / Microsoft's Next 100 series in 2009, and is the recipient of ESORICS 2015 Best Research Paper Award, 2015 Winning Team of Germany's University of Erlangen-Nuremberg Digital Forensics Research Challenge, 2014 Australia New Zealand Policing Advisory Agency's Highly Commended Award, 2010 Australian Capital Territory Pearcey Award, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award.

Anwitaman Datta is an associate professor in the School of Computer Science and Engineering in NTU Singapore. He leads the Self-* and Algorithmic aspects of Networked Distributed Systems (SANDS) research group at NTU.

Jun Shao received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a Postdoctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, State College, PA, USA, from 2008 to 2010. He is currently a full Professor with the Department of Information Security, Zhejiang Gongshang University, Hangzhou, China. His research interests include network security and applied cryptography.