Information Sciences 267 (2014) 267–286

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Towards semantically secure outsourcing of association rule mining on categorical data

Junzuo Lai [a,b,*], Yingjiu Li [a], Robert H. Deng [a], Jian Weng [a,b], Chaowen Guan [a], Qiang Yan [a]

[a] Department of Computer Science, Jinan University, China
[b] School of Information Systems, Singapore Management University, Singapore

ARTICLE INFO

Article history:
Received 23 October 2012
Received in revised form 24 July 2013
Accepted 26 January 2014
Available online 1 February 2014

Keywords:
Association rule mining
Outsourcing
Semantic security
Privacy
Soundness

ABSTRACT

When outsourcing association rule mining to cloud, it is critical for data owners to protect both sensitive raw data and valuable mining results from being snooped at cloud servers. Previous solutions addressing this concern add random noise to the raw data and/or encrypt the raw data with a substitution mapping. However, these solutions do not provide semantic security; partial information about raw data or mining results can be potentially discovered by an adversary at cloud servers under a reasonable assumption that the adversary knows some plaintext–ciphertext pairs. In this paper, we propose the first semantically secure solution for outsourcing association rule mining with both data privacy and mining privacy. In our solution, we assume that the data is categorical. Additionally, our solution is sound, which enables data owners to verify whether there exists any false data in the mining results returned by a cloud server. Experimental study shows that our solution is feasible and efficient.

1. Introduction

Data mining is a very popular research area in the field of computer science. The objective of data mining is to extract comprehensible, useful, and non-trivial knowledge from large datasets. Association rule mining, introduced in [1], which aims at finding close relationships between items in transactional data, has been the major focus in data mining research with enormous applications such as market basket analysis, network intrusion detection, and inventory control [2,17,20,26,36,48].

In the cloud computing era, data owners who lack expertise or computational resources may outsource association rule mining to cloud service providers. While outsourcing has the potential of reducing computation and software cost, it is critical to protect both sensitive raw data, which is the database of transactions, and valuable mining results from being snooped at cloud servers. To address this concern, data owners need to encrypt the original data using a suitable algorithm such that the mining tasks can be performed by cloud servers directly on the encrypted data. Though this topic has been intensively researched since the introduction of privacy preserving data mining [3], none of the previous works on outsourcing of association rule mining [14,33,41,50] ensure semantic security [16]. Informally, semantic security states that an adversary (at cloud servers) can gain no partial information about the target data even the adversary has the capability of

* Corresponding author at: Department of Computer Science, Jinan University, China.
E-mail addresses: junzuolai@smu.edu.sg (J. Lai), yjli@smu.edu.sg (Y. Li), robertdeng@smu.edu.sg (R.H. Deng), cryptjweng@gmail.com (J. Weng), cwguan@smu.edu.sg (C. Guan), qiang.yan.2008@phdis.smu.edu.sg (Q. Yan).

http://dx.doi.org/10.1016/j.ins.2014.01.040
0020-0255/

obtaining adaptively plaintext–ciphertext pairs. In this paper, we investigate semantically secure solutions for efficient outsourcing of association rule mining. To the best of our knowledge, this is the first attempt in this area.

Initial researches in association rule mining [1,2,18,19] have assumed that the data is categorical. We also start the study of semantically secure outsourcing of association rule mining from the categorical data. In practice, databases may contain much quantitative data and are not limited to categorical items only. Different evolutionary algorithms and especially genetic algorithms [4–6,8,21,25,30,35,43–45] have been proposed to extract association rules in quantitative data, and how to extend these algorithms to construct semantically secure outsourcing of association rule mining on quantitative data will be left as our future work. As most association rule mining algorithms [1,2,18,19] on categorical data, we split the problem of association rule mining into two subproblems. The first is to find all of the frequent itemsets in database. The second is to find the association rules from the discovered frequent itemsets. As showned in [1], it is straightforward to solve the second subproblem after the first subproblem is resolved. In this paper, we focus on the first subproblem.

It had been difficult to design a semantically secure solution for outsourcing association rule mining until the emergence of fully homomorphic encryption [12]. Since fully homomorphic encryption [12] enables arbitrary functions to be computed on encrypted data, it can be directly applied to association rule mining. However, we will show that this solution is extremely inefficient and thus impractical. In addition, it requires that most of the data mining task be performed by data owners rather than by cloud servers, which is contrary to the purpose of outsourcing.

In our method, the dominant task of association rule mining is to discover all frequent itemsets (i.e., itemsets whose frequencies appearing in transactions are higher than a pre-defined threshold). The basic operation for discovering all frequent itemsets is to decide whether or not a transaction contains an itemset. We discover that this basic operation can be transformed to an inner product operation, and it is possible to adopt the notion of symmetric-key predicate-only encryption for inner products [40] in the design of semantically secure and efficient schemes for outsourcing association rule mining. We notice that the existing symmetric-key predicate-only encryption scheme for inner products [40] is only selectively secure, meaning that the security is proven in a weaker model where part of the challenge strings must be revealed before the attacker receives the public parameters. To remove this constraint, based on the predicate encryption scheme for inner products proposed by [27] and the dual system encryption methodology introduced by [49], we propose a fully secure symmetric-key predicate-only encryption scheme for inner products, which enables us to construct a semantically secure and efficient scheme for outsourcing association rule mining.

Besides addressing the privacy concern for protecting both raw data and data mining results, we also consider the situation in which a cloud server may have incentive to insert false data into true mining results (e.g., for charging more fees). To address this concern, we introduce the soundness requirement so that data owners are able to verify whether the data mining results returned by the cloud server contain any itemsets that are not frequent and if so, the data owners are able to filter out such false data.

1.1. Our contribution

In this paper, we define a formal model of outsourcing association rule mining. Our model consists of four algorithms: SysSetup, AddEncTransRecord, RetrieveTransRecord, and GetFreqItemSets. Algorithm SysSetup is used to output public parameters and data owner's secret key. Algorithm AddEncTransRecord is used to encrypt transaction records before outsourcing data to a cloud server. Algorithm RetrieveTransRecord is not necessarily part of outsourcing association rule mining, and it is for the convenience of data owner to retrieve transaction records from their encrypted forms. Algorithm GetFreqItemSets is a 2-round protocol executed between data owner and a cloud server. In this protocol, data owner sends data mining requests together with necessary token keys to the cloud server, which returns to data owner with encrypted data mining results after performing major data mining tasks on encrypted data. At the end of this protocol, data owner retrieves all frequent itemsets from the encrypted data mining results.

We define three security requirements for outsourcing association rule mining. In addition to protecting raw data (*data privacy*) and mining results (*mining privacy*) under chosen plaintext attacks, we identify another important security requirement: *soundness*, which has not been considered in previous works. In brief, soundness enables data owner to check and filter out any false data from the data mining results returned by a cloud server.

Based on the predicate encryption scheme for inner products proposed by Lewko et al. [27], we propose an efficient solution for outsourcing association rule mining in our model. We prove that our solution achieves data privacy, mining privacy and soundness under chosen plaintext attacks. To the best of our knowledge, it is the first semantically secure solution for outsourcing association rule mining. We also implement our solution and show that it is efficient in experiments.

1.2. Organization

The rest of this paper is organized as follows. Section 2 introduces the related work. In Section 3, we give the preliminary knowledge for designing our solution. In Section 4, we define the model of outsourcing association rule mining and propose a concrete construction under our model with security proofs. In Section 5, we show performance results. Finally, in Section 6, we conclude this paper and point out some future directions.

2. Related work

Since the introduction of privacy preserving data mining [3], data perturbation has been used to protect sensitive information when outsourcing data mining of frequent itemsets [9,28,32,38,42]. This approach aims at protecting the raw data rather than the mining results. Due to the random noise added to the raw data, this approach may have unpredictable impacts on data mining precision. To provide better protection on both raw data and mining results, later works employ simple encryption, which is usually a substitution mapping of raw data items. For example, Wong et al. [50] proposed a solution that encrypts original transactions by a mapping function, and adds random "fake" items to the encrypted transactions. Unfortunately, Molloy et al. [33] showed that the random "fake" items can be removed by detecting the low correlations between items, and that top frequent items can be re-identified by attackers. Recently, Tai et al. [41] and Giannotti et al. [14] proposed similar methods to achieve k anonymity protection for both raw data and mining results against a knowledgable adversary with certain background knowledge. However, none of these works assume that the adversary at server has the capability of launching chosen plaintext attacks; that is, they do not provide semantic security, which means that the adversary with knowledge on chosen plaintext–ciphertext pairs is able to infer partial information about the raw data and mining results.

We clarify that our work is different from the line of work on secure multiparty computation (SMC) [15,54] and its application to association rule mining [22,29,34]. SMC provides a generic approach to solving problems in the distributed computing scenario, in which two or more parties compute collaboratively on an agreed function from their private inputs. When SMC is applied to association rule mining, it is assumed that multiple parties holding part of original data such as horizontally partitioned data [22] work collaboratively in discovering association rules without revealing each other's data. In our outsourcing scenario, however, the involving parties contribute differently in performing the task – while the original data is provided by data owner only, the mining task is mainly performed at cloud servers on encrypted data.

The notions of privacy-preserving data publishing and association rule hiding are also related to our work. Privacy-preserving data publishing [10,11,31,51–53] enables accurate data patterns such as aggregate statistics and association rules to be discovered from published data while any individual's privacy related to the underlying data is protected in data publication. In association rule mining context, this notion protects raw data (in terms of individual's privacy) but no mining results. On the other hand, association rule hiding [37,46,47,55] is concerned with how to transform the raw data such that certain sensitive associate rules cannot be discovered from the released dataset while other rules can. In other words, it aims at protecting part of mining results only. In comparison, our task is to protect both raw data and mining results at an outsourced server in terms of semantic security.

3. Notations and assumptions

In this section, we introduce some notations and backgrounds on dual pairing vector spaces, pseudorandom functions, predicate encryption, and association rule mining, which will be used in constructing our solution.

3.1. Notation

Given a set S, let $s \xleftarrow{\$} S$ denote the operation of picking an element s uniformly at random from S. Let $\mathbb{N}$ denote the natural numbers. For $\lambda \in \mathbb{N}$, let 1^λ denote the string of λ ones. Let $z \leftarrow \mathsf{A}(x, y, \ldots)$ denote the operation of running an algorithm A with inputs $(x, y, \ldots)$ and output z. A function $f(\lambda)$ is *negligible* if for every natural number $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. Let $GL(N, \mathbb{F}_q)$ denote the general linear group of degree N over $\mathbb{F}_q$, which is the set of $N \times N$ invertible matrices with entries from $\mathbb{F}_q$.

3.2. Dual pairing vector spaces

A dual pairing vector space (DPVS) $(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{A})$ is a tuple of a prime q, two cyclic (multiplicative) groups $\mathbb{G}$ and $\mathbb{G}_T$ of order q, a pairing $\hat{e}$, an element $g \neq 1 \in \mathbb{G}$, an N-dimensional vector space $\mathbb{V} = \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{N}$, and a canonical basis $\mathbb{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ of $\mathbb{V}$, where $\mathbf{a}_i = (\overbrace{1, \ldots, 1}^{i-1}, g, \overbrace{1, \ldots, 1}^{N-i})$. The pairing $\hat{e} : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$ satisfies the following conditions.

1. The pairing $\hat{e}$ is a polynomial-time computable non-degenerate bilinear pairing: $\hat{e}(s\mathbf{x}, t\mathbf{y}) = \hat{e}(\mathbf{x}, \mathbf{y})^{st}$; if $\hat{e}(\mathbf{x}, \mathbf{y}) = 1$ for all $\mathbf{y} \in \mathbb{V}$, then $\mathbf{x} = \mathbf{0}$. The pairing $\hat{e}$ is defined by $\hat{e}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{N} \hat{e}(g_i, h_i)$, where $\mathbf{x} = (g_1, \ldots, g_N) \in \mathbb{V}$ and $\mathbf{y} = (h_1, \ldots, h_N) \in \mathbb{V}$.
2. For all i and j, it holds $\hat{e}(\mathbf{a}_i, \mathbf{a}_j) = \hat{e}(g, g)^{\delta_{i,j}}$, where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ otherwise.
3. There exist polynomial-time computable endomorphisms $\phi_{i,j}$ of $\mathbb{V}$ that satisfy $\phi_{i,j}(\mathbf{a}_j) = \mathbf{a}_i$ and $\phi_{i,j}(\mathbf{a}_k) = \mathbf{0}$ if $k \neq j$.

Let $\mathcal{G}_{dpvs}$ be an algorithm that takes as input a security parameter $1^\lambda (\lambda \in \mathbb{N})$ and $N \in \mathbb{N}$, and outputs a tuple $(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{A})$. A random orthonormal basis generator $\mathcal{G}_{ob}$ is defined as follows

$$
\begin{aligned}
&\mathcal{G}_{ob}(1^{\lambda}, N):\\
&\quad (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{A}) \leftarrow \mathcal{G}_{dpvs}(1^{\lambda}, N),\\
&\quad X = (\chi_{i,j}) \xleftarrow{\$} GL(N, \mathbb{F}_q),\ (\vartheta_{i,j}) = (X^{\mathrm{T}})^{-1},\\
&\quad \mathbf{b}_i = \sum_{j=1}^{N} \chi_{i,j}\mathbf{a}_j,\ \mathbb{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_N),\\
&\quad \mathbf{b}_i^* = \sum_{j=1}^{N} \vartheta_{i,j}\mathbf{a}_j,\ \mathbb{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_N^*),\\
&\quad \text{return } (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*).
\end{aligned}
$$

where $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthonormal bases; that is, for all i and j, $\hat{e}(\mathbf{b}_i, \mathbf{b}_j^*) = \hat{e}(g,g)^{\delta_{i,j}}$, where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ otherwise.

We now state the complexity assumptions that we will rely onto prove security of our outsourcing association rule mining scheme. These assumptions are similar to those used by Lewko et al. [27]. It is easy to show that these assumptions hold in the generic group model by applying the theorems of Katz et al. [23], as did in [27].

Assumption 1. Let $\mathcal{G}_{ob}$ be a random orthonormal basis generator. We define the following distribution

$$
\begin{aligned}
&(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^{\lambda}, 2n+5),\\
&\widehat{\mathbb{B}} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}, \mathbf{b}_{2n+3}, \mathbf{b}_{2n+5}),\\
&\widehat{\mathbb{B}}^* = (\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*, \mathbf{b}_{n+1}^*, \mathbf{b}_{2n+3}^*, \mathbf{b}_{2n+4}^*),\\
&D = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*),\\
&\text{for } i, j = 1, \ldots, n+1,\\
&\quad \delta_1, \delta_{2,i} \xleftarrow{\$} \mathbb{F}_q, \rho \xleftarrow{\$} \mathbb{F}_q^*, (u_{ij}) \xleftarrow{\$} GL(n+1, \mathbb{F}_q),\\
&\text{for } i = 1, \ldots, n+1,\\
&\quad \mathbf{e}_{0,i} = \delta_1\mathbf{b}_i + \delta_{2,i}\mathbf{b}_{2n+5},\\
&\quad \mathbf{e}_{1,i} = \delta_1\mathbf{b}_i + \rho\sum_{j=1}^{n+1} u_{ij}\mathbf{b}_{n+1+j} + \delta_{2,i}\mathbf{b}_{2n+5}.
\end{aligned}
$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 is defined as

$$Adv_{\mathcal{A}}^{1} = |\Pr[\mathcal{A}(D, \{\mathbf{e}_{0,i}\}_{i=1,\ldots,n+1}) = 1] - \Pr[\mathcal{A}(D, \{\mathbf{e}_{1,i}\}_{i=1,\ldots,n+1}) = 1]|.$$

Assumption 2. Let $\mathcal{G}_{ob}$ be a random orthonormal basis generator. We define the following distribution

$$
\begin{aligned}
&(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^{\lambda}, 2n+5),\\
&\widehat{\mathbb{B}} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}, \mathbf{b}_{2n+3}, \mathbf{b}_{2n+5}),\\
&\widehat{\mathbb{B}}^* = (\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_{2n+3}^*, \mathbf{b}_{2n+4}^*),\\
&D = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*),\\
&\text{for } i, j = 1, \ldots, n+1,\\
&\quad \omega, \gamma_i, \delta \xleftarrow{\$} \mathbb{F}_q, \rho, \tau \xleftarrow{\$} \mathbb{F}_q^*,\\
&\quad (u_{ij}) \xleftarrow{\$} GL(n+1, \mathbb{F}_q),\ (z_{ij}) = ((u_{ij})^{-1})^{\mathrm{T}},\\
&\text{for } i = 1, \ldots, n+1,\\
&\quad \mathbf{h}_{0,i}^* = \omega\mathbf{b}_i^* + \gamma_i\mathbf{b}_{2n+4}^*,\\
&\quad \mathbf{h}_{1,i}^* = \omega\mathbf{b}_i^* + \tau\sum_{j=1}^{n+1} z_{ij}\mathbf{b}_{n+1+j}^* + \gamma_i\mathbf{b}_{2n+4}^*,\\
&\quad \mathbf{e}_i = \delta\mathbf{b}_i + \rho\sum_{j=1}^{n+1} u_{ij}\mathbf{b}_{n+1+j}.
\end{aligned}
$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 is defined as

$$Adv^2_{\mathcal{A}} = |\Pr[\mathcal{A}(D, \{\mathbf{h}^*_{0,i}, \mathbf{e}_i\}_{i=1,\ldots,n+1}) = 1] - \Pr[\mathcal{A}(D, \{\mathbf{h}^*_{1,i}, \mathbf{e}_i\}_{i=1,\ldots,n+1}) = 1]|.$$

Assumption 3. Let $\mathcal{G}_{ob}$ be a random orthonormal basis generator. We define the following distribution

$$(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^\lambda, 2n+5),$$
$$\widehat{\mathbb{B}} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}, \mathbf{b}_{2n+3}, \mathbf{b}_{2n+5}),$$
$$\widehat{\mathbb{B}}^* = (\mathbf{b}^*_1, \mathbf{b}^*_2, \ldots, \mathbf{b}^*_n, \mathbf{b}^*_{n+1}, \mathbf{b}^*_{2n+3}, \mathbf{b}^*_{2n+4}),$$
$$D = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*),$$
for $i, j = 1, \ldots, n+1$,
$$\omega, \gamma_i \xleftarrow{\$} \mathbb{F}_q, \tau \xleftarrow{\$} \mathbb{F}^*_q, \ (z_{i,j}) \xleftarrow{\$} GL(n+1, \mathbb{F}_q),$$
for $i = 1, \ldots, n+1$,
$$\mathbf{h}^*_{0,i} = \omega \mathbf{b}^*_i + \gamma_i \mathbf{b}^*_{2n+4},$$
$$\mathbf{h}^*_{1,i} = \omega \mathbf{b}^*_i + \tau \sum_{j=1}^{n+1} z_{i,j} \mathbf{b}^*_{n+1+j} + \gamma_i \mathbf{b}^*_{2n+4}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 is defined as

$$Adv^3_{\mathcal{A}} = |\Pr[\mathcal{A}(D, \{\mathbf{h}^*_{0,i}\}_{i=1,\ldots,n+1}) = 1] - \Pr[\mathcal{A}(D, \{\mathbf{h}^*_{1,i}\}_{i=1,\ldots,n+1}) = 1]|.$$

Assumption 4. Let $\mathcal{G}_{ob}$ be a random orthonormal basis generator. We define the following distribution

$$(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^\lambda, 2n+5),$$
$$\widehat{\mathbb{B}} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{2n+4}, \mathbf{b}_{2n+5}),$$
$$\widehat{\mathbb{B}}^* = (\mathbf{b}^*_1, \mathbf{b}^*_2, \ldots, \mathbf{b}^*_n, \mathbf{b}^*_{n+1}, \mathbf{b}^*_{2n+3}, \mathbf{b}^*_{2n+4}),$$
$$D = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*),$$
for $i, j = 1, \ldots, n+1$,
$$\delta_1, \delta_{2,i}, \omega \ \$ \leftarrow \mathbb{F}_q, \rho, \tau \xleftarrow{\$} \mathbb{F}^*_q,$$
$$(u_{i,j}) \xleftarrow{\$} GL(n+1, \mathbb{F}_q), \ (z_{i,j}) = ((u_{i,j})^{-1})^{\mathrm{T}},$$
for $i = 1, \ldots, n+1$,
$$\mathbf{e}_{0,i} = \delta_1 \mathbf{b}_i + \delta_{2,i} \mathbf{b}_{2n+5},$$
$$\mathbf{e}_{1,i} = \delta_1 \mathbf{b}_i + \rho \sum_{j=1}^{n+1} u_{i,j} \mathbf{b}_{n+1+j} + \delta_{2,i} \mathbf{b}_{2n+5},$$
$$\mathbf{h}^*_i = \omega \mathbf{b}^*_i + \tau \sum_{j=1}^{n+1} z_{i,j} \mathbf{b}^*_{n+1+j}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 is defined as

$$Adv^4_{\mathcal{A}} = |\Pr[\mathcal{A}(D, \{\mathbf{e}_{0,i}, \mathbf{h}^*_i\}_{i=1,\ldots,n+1}) = 1] - \Pr[\mathcal{A}(D, \{\mathbf{e}_{1,i}, \mathbf{h}^*_i\}_{i=1,\ldots,n+1}) = 1]|.$$

Definition 1. $\mathcal{G}_{ob}$ satisfies Assumption i if for any polynomial time algorithm $\mathcal{A}, Adv^i_{\mathcal{A}}$ is negligible, for $1 \leqslant i \leqslant 4$.

3.3. Pseudorandom functions

A function family is a mapping $F : \mathrm{Keys}(F) \times D \to R$, where D is the domain of F, R is the range of F, and $\mathrm{Keys}(F)$ is the set of keys (or indexes) of F. An instance of family F is a mapping $F_K : D \to R$, where $K \in \mathrm{Keys}(F)$ and $F_K(x) = F(K, x)$.

A pseudorandom function is a family of functions with the property that the input–output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function. Intuitively, anyone who has only black-box access to a function is not able to distinguish within polynomial time whether the function in question is a random instance of the family or a random function.

Definition 2. Let $F : \mathsf{Keys}(F) \times D \to R$ be a family of functions. The function family $\mathsf{Rand}^{D\to R}$ has domain D and range R. The set of instances of $\mathsf{Rand}^{D\to R}$ is the set of all functions mapping D to R. A game is defined between an attack algorithm A and a challenger as follows

Setup: The challenger randomly chooses a key $k \in \mathsf{Keys}(F)$ and a function $g \in \mathsf{Rand}^{D\to R}$. Then, it selects a random bit $\beta \in \{0,1\}$.

Query phase: The adversary $\mathcal{A}$ issues a query on $x \in D$. If $\beta = 0$, the challenger returns value $g(x)$. Else, it returns value $F_k(x)$.

Guess: The adversary $\mathcal{A}$ outputs its guess $\beta\prime \in \{0,1\}$ for β and wins the game if $\beta = \beta\prime$.

The adversary $\mathcal{A}$'s advantage in the game is defined as

$$Adv_{\mathcal{A}}(q) = \left|\mathsf{Pr}[\beta = \beta'] - \frac{1}{2}\right|.$$

A function family F is pseudorandom if all polynomial time adversaries have at most negligible advantages in the above game.

3.4. Predicate encryption

In predicate encryption (PE) [24], secret keys correspond to predicates and ciphertexts are associated with attributes. The secret key SK_f corresponding to a predicate f can be used to decrypt a ciphertext associated with attribute I if and only if $f(I) = 1$. In a special case of PE for inner product predicates, each attribute corresponds to a vector $\vec{x}$ and each predicate $f_{\vec{v}}$ corresponds to a vector $\vec{v}$ such that $f_{\vec{v}}(\vec{x}) = 1$ iff $\vec{x} \cdot \vec{v} = 0$, where $\vec{x} \cdot \vec{v}$ denotes the standard inner-product.

A stronger security notion, *attribute-hiding*, as well as the basic security requirement, *payload-hiding*, can be defined for PE [24]. The attribute-hiding requires that a ciphertext conceal the associated attribute as well as plaintext, while payload-hiding only requires plaintext being concealed. Another security notion of PE, *predicate privacy*, requires that a secret key in PE hides all information about the encoded predicate other than what can be implied from the ciphertext [40]. Predicate privacy, however, is inherently impossible to achieve in the public-key setting.

3.5. Association rule mining

Let $\mathcal{I}$ be a set of literals, called *items*. A set of items $X \subseteq \mathcal{I}$ is called an *itemset*. Let $\mathcal{D}$ be a database of transactions, where each transaction T is an itemset. A transaction T *contains* an itemset X if $X \subseteq T$.

The support of an itemset X in the database $\mathcal{D}$, denoted $supp(X)$, is the number of transactions in $\mathcal{D}$ that contain the itemset X. Given a support threshold $s\%$, we say that X is a *frequent* itemset if and only if $supp(X) \geqslant |\mathcal{D}| \times s\%$, where $|\mathcal{D}|$ is the number of transactions in $\mathcal{D}$.

An association rule is an implications of the form $X \Rightarrow Y$, where $X, Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. The association rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with $s\%$ *support* and $c\%$ *confidence* if $s\%$ of transactions in $\mathcal{D}$ contain $X \cup Y$ and $c\%$ of transactions in $\mathcal{D}$ that contain X also contain Y.

The problem of association rule mining was initially presented in [1]. The authors in [1] discovered that the problem can be decomposed into two subproblems. The first is to find all of the frequent itemsets in $\mathcal{D}$. The second is to find the association rules from the discovered frequent itemsets. As discovered in [1], it is straightforward to solve the second subproblem after the first subproblem is solved. In this paper, we focus on the first subproblem.

4. Outsourcing association rule mining

In general, a system of outsourcing association rule mining consists of four algorithms: SysSetup, AddEncTransRecord, RetrieveTransRecord and GetFreqItemSets. As illustrated by Fig. 1, first of all, data Owner runs algorithm SysSetup to output public parameters and a secret key for himself. Then, data Owner executes algorithm AddEncTransRecord to encrypt transaction records before outsourcing data to a cloud server. Algorithm RetrieveTransRecord is dedicated to the convenience of data owner to retrieve transaction records from their encrypted forms. Algorithm GetFreqItemSets is a 2-round protocol executed between data Owner and a cloud Server. In this protocol, firstly, data Owner sends data mining requests together with necessary token keys to the cloud Server. Secondly, the cloud Server performs major data mining tasks on encrypted data, and then returns the encrypted data mining results to data Owner. At the end of this protocol, data Owner retrieves all frequent itemsets from the encrypted data mining results.

Formally, outsourcing association rule mining from data Owner to cloud Server can be modeled as a set of polynomial algorithms shown below:

- $\mathsf{SysSetup}(1^{\lambda})$: An algorithm run by Owner which, given a security parameter λ, outputs the system's public parameters PK and secret key SK.

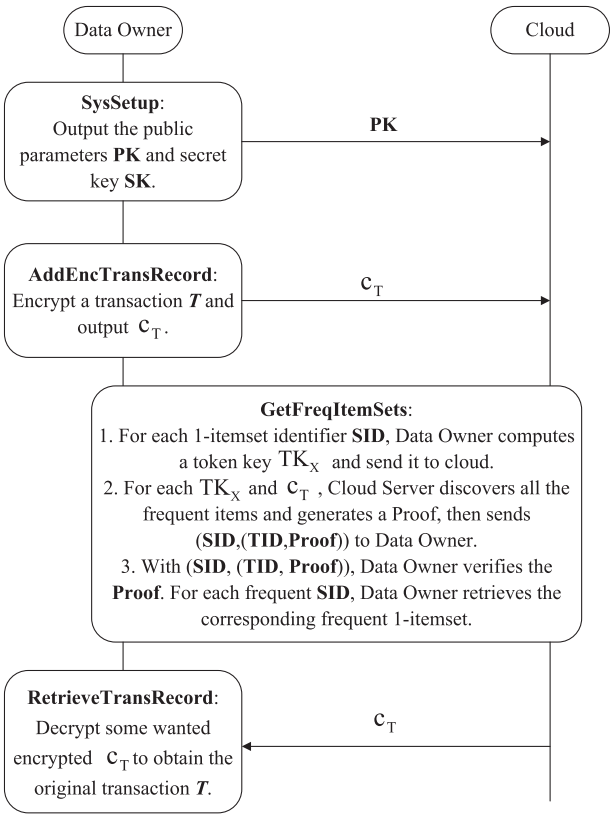

Fig. 1. Outsourcing association rule mining.

$$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{SysSetup}(1^{\lambda})$$

- $\mathsf{AddEncTransRecord}(\mathsf{SK}, T)$: An algorithm run by Owner which, given the system's secret key SK and transaction T, outputs an encrypted transaction c_T.

$$c_T \leftarrow \mathsf{AddEncTransRecord}(\mathsf{SK}, T)$$

- $\mathsf{RetrieveTransRecord}(\mathsf{SK}, c_T)$: An algorithm run by Owner which, given the system's secret key SK and an encrypted transaction c_T, outputs the original transaction T.

$$T \leftarrow \mathsf{RetrieveTransRecord}(\mathsf{SK}, c_T)$$

- $\mathsf{GetFreqItemSets}(\mathsf{Owner}, \mathsf{Server})$: A protocol executed between Owner and Server for Owner to obtain frequent itemsets after Server performs association rule mining on all encrypted transactions.

We consider the following security properties for outsourcing association rule mining:

1. *Data privacy*: Server cannot obtain any information about the original transaction set.
2. *Mining privacy*: At the end of GetFreqItemSets protocol, Server cannot obtain any information about the frequent itemsets.
3. *Soundness*: During the execution of GetFreqItemSets protocol, Server cannot convince Owner that an itemset is a frequent itemset if it is not.

4.1. Overview of our construction

We have *first attempt* and *second attempt* before the *final solution*. In our *first attempt*, we employ fully homomorphic encryption [12], which enables arbitrary functions to be computed on encrypted data. However, we find this solution, based on the semantically secure fully homomorphic encryption schemes is extremely inefficient. Then, we turn to our *second attempt*. We consider symmetric-key predicate-only encryption scheme for inner products [40] this time. In the existing predicate encryption schemes, they do not provide two kinds of algorithms for data Owner to retrieve frequent itemsets from

hidden frequent itemsets and plain transactions from ciphertexts of transactions, respectively, although the latter one might not be necessary in outsourcing. Besides, the existing efficient symmetric-key predicate-only encryption scheme for inner products [40] is only proved to be selectively secure (it means an adversary must commit to part of the challenge strings before the setup phase of attacks). Unfortunately, this tentative construction does not meet our objectives of obtaining a fully secure symmetric-key predicate-only encryption scheme for inner products and achieving soundness. Finally, based on a predicate encryption scheme for inner products proposed by Lewko et al. [27], we obtain a symmetric-key predicate-only encryption scheme for inner products, and we apply the dual system encryption methodology introduced by Waters [49] to achieve full security. This final construction reaches our objectives and requirements aforementioned. We present the details of our *first attempt*, *second attempt* and the *final solution* below.

4.1.1. First attempt

To achieve data privacy and mining privacy, Owner needs a semantically secure encryption scheme to encrypt each transaction before it outsources data to Server. It is required that Server be able to perform association rule mining on the encrypted data and obtain encrypted frequent itemsets from which Owner is able to retrieve all frequent itemsets. Since fully homomorphic encryption [12] enables arbitrary functions to be computed on encrypted data, our first attempt is to apply fully homomorphic encryption in our construction.

According to [12], a fully homomorphic public key encryption scheme consists of four algorithms: KeyGen, Encrypt, Decrypt, and Evaluate. Given any valid public key pk, any circuit C, and any ciphertexts $c_i \leftarrow \mathsf{Encrypt}(pk, m_i)$, such a scheme outputs

$$\bar{c} \leftarrow \mathsf{Evaluate}(pk, C, c_1, \ldots, c_t),$$

where $\bar{c}$ is a valid encryption of $C(m_1, \ldots, m_t)$, i.e., one can obtain $C(m_1, \ldots, m_t)$ after a decryption on $\bar{c}$. Obviously, one can arbitrarily compute on encrypted data without the decryption key by means of fully homomorphic public key encryption schemes.

Let $\mathcal{E}$ be a fully homomorphic public key encryption scheme. The basic idea of our construction with $\mathcal{E}$ is as follows. To setup the system, Owner first runs $(pk, sk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^{\lambda})$. Then Owner publishes pk and keeps sk secret. To encrypt a transaction T, Owner runs

$$c_T \leftarrow \mathcal{E}.\mathsf{Encrypt}(pk, T).$$

Owner sends c_T as well as the transaction identifier TID (e.g., an obfuscated serial number) of T to Server. If $\mathcal{E}$ is semantically secure, Server cannot obtain any information about the transaction T from c_T (i.e., this construction achieves data privacy). To perform association rule mining, for each 1-itemset X, Owner runs $c_X \leftarrow \mathcal{E}.\mathsf{Encrypt}(pk, X)$ and sends the ciphertexts of all 1-itemsets to Server. Then, for each encrypted 1-itemset X, Server runs $c_{T,X} \leftarrow \mathcal{E}.\mathsf{Evaluate}(pk, C, (c_T, c_X))$ for all encrypted transactions T, and returns to Owner with $c_{T,X}$ as well as the transaction identifier (TID) associated with the encrypted transaction T, where C is a circuit and

$$C(T,X) = \begin{cases} 1, & \text{if } T \text{ contains } X; \\ 0, & \text{otherwise.} \end{cases}$$

If $\mathcal{E}$ is semantically secure, Server cannot obtain any information about X, T, and $C(T,X)$ during the mining process without knowing Owner's secret key (i.e., this construction achieves mining privacy). Finally, Owner runs $\mathsf{Decrypt}(sk, c_{T,X})$ to obtain $C(T,X)$, and thus get all frequent 1-itemsets together with their TID's. The frequent k-itemsets can be obtained from frequent 1-itemsets by using Apriori algorithm [2] or FP-Growth algorithm [18,19].

Unfortunately, the above solution, based on existing semantically secure fully homomorphic encryption schemes, is extremely inefficient. We shall see in Section 4 that the computational costs of applying fully homomorphic encryption are at least several orders higher than those of our proposed solution at both Owner's side and Server's side. Even if more efficient fully homomorphic encryption schemes may exist for association rule mining (i.e., computing $c_{T,X}$ from c_T and c_X rather than computing arbitrary functions on encrypted data), it will incur high communication cost between Owner and Server and high computational cost on Owner's side. This is because $C(T,X)$ is hidden from Server; for *each* encrypted 1-itemset c_X, Server needs to return to Owner $c_{T,X}$ (as well as TID's) for *all* encrypted transactions c_T. Not only does the Owner need to decrypt $c(T,X)$ for all combinations of X and T, but also discover all frequent 1-itemsets, and all other frequent itemsets. The overhead incurred on Owner's side is thus even higher than running traditional association rule mining itself, not to mention the extremely high cost incurred on Owner–Server communication and on Server's computation.

4.1.2. Second attempt

We use the following notation for the rest of this paper. Let $\mathcal{I} = \{it_1, \ldots, it_n\}$ be a set of items. A transaction T is represented as a binary vector $(t_1, \ldots, t_n)$, where $t_i = 1$ if T contains item it_i, and $t_i = 0$ otherwise. An itemset X is also represented as a binary vector $(x_1, \ldots, x_n)$, where $x_i \in \{0, 1\}$. Itemset X is called a k-itemset if $\sum_{i=1}^{n} x_i = k$. Note that transaction T contains k-itemset X if the inner product of T and X is k.

If a transaction $T = (t_1, \ldots, t_n)$ contains an itemset $X = (x_1, \ldots, x_n)$, then we have $T' \cdot X' = 0$, where $T' = (t_1, \ldots, t_n, t_{n+1} = -1)$, $X' = (x_1, \ldots, x_n, x_{n+1} = \sum_{i=1}^{n} x_i)$. Based on this observation, we consider symmetric-key predicate-only encryption scheme for inner products[1] [40], which consists of the following four algorithms:

- $\mathsf{Setup}(1^\lambda)$: An randomized algorithm which, given a security parameter λ, outputs a public key PK and a master secret key MSK.
- $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \vec{v})$: An randomized algorithm which, given public key PK, master secret key MSK and predicate vector $\vec{v}$, outputs a token key $\mathsf{SK}_{\vec{v}}$.
- $\mathsf{Encrypt}(\mathsf{PK}, \mathsf{MSK}, \vec{x})$: An randomized algorithm which, given public key PK, master secret key MSK and attribute vector $\vec{x}$, outputs a ciphertext c.
- $\mathsf{Query}(\mathsf{PK}, \mathsf{SK}_{\vec{v}}, c)$: An algorithm which, given public key PK, token key $\mathsf{SK}_{\vec{v}}$ and ciphertext c, outputs either 0 or 1.

The output of $\mathsf{Query}(\mathsf{PK}, \mathsf{SK}_{\vec{v}}, c)$ indicates the value of the predicate $\vec{v}$ evaluated on the underlying attribute vector $\vec{x}$ of $c = \mathsf{Encrypt}(\mathsf{PK}, \mathsf{MSK}, \vec{x})$. It is required that with overwhelming probability, if $\vec{x} \cdot \vec{v} = 0$, then $1 \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{SK}_{\vec{v}}, \mathsf{Encrypt}(\mathsf{PK}, \mathsf{MSK}, \vec{x}))$; otherwise, $0 \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{SK}_{\vec{v}}, \mathsf{Encrypt}(\mathsf{PK}, \mathsf{MSK}, \vec{x}))$.

Let Π be a symmetric-key predicate-only encryption scheme for inner products. We construct an association rule mining solution with Π as follows. To setup the system, Owner runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, publishes PK and keeps MSK secret. To encrypt a transaction $T = (t_1, \ldots, t_n)$, Owner runs

$$c_{T'} \leftarrow \Pi.\mathsf{Encrypt}(\mathsf{PK}, \mathsf{MSK}, T'),$$

where $T' = (t_1, \ldots, t_n, t_{n+1} = -1)$. Owner sends $c_{T'}$ and the transaction identifier TID of T to Server. If Π is attribute-hiding, then Server cannot obtain any information about the transaction T from $c_{T'}$ (i.e., this construction achieves data privacy).

To perform the data mining task, for each 1-itemset $X = (x_1, \ldots, x_n)$, Owner runs $\mathsf{SK}_{X'} \leftarrow \Pi \cdot \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, X')$, where $X' = (x_1, \ldots, x_n, x_{n+1} = \sum_{i=1}^{n} x_i)$. Owner sends all token keys $\mathsf{SK}_{X'}$ to Server. If Π has predicate privacy, Server cannot obtain any information about the itemset X from $SK_{X'}$ (i.e., this construction achieves mining privacy). Upon receiving the token keys, Server runs algorithm $\Pi \cdot \mathsf{Query}$ on each combination of token keys $\mathsf{SK}_{X'}$ and encrypted transactions $c_{T'}$. If and only if $\Pi.\mathsf{Query}$ yields 1, the original transaction T contains 1-itemset X. Server can thus calculate the support of each 1-itemset X represented by token key $\mathsf{SK}_{X'}$ and record the token keys whose supports are greater than a predetermined threshold. We call such token keys as hidden frequent 1-itemsets. For each hidden frequent 1-itemset, Server also records a set of TID's which are the identifiers of the (encrypted) transactions that contain the corresponding 1-itemset.

The task of finding frequent k-itemsets can be performed on either Owner side or Server side. If it is performed on Owner side, Server sends all hidden frequent 1-itemsets together with their TID sets to Owner, who can retrieve all frequent 1-itemsets from the hidden sets using its key and discover all other frequent itemsets from frequent 1-itemsets (by using Apriori algorithm [2] or FP-Growth algorithm [18,19]). If it is performed on the Server side, Server can discover all hidden frequent itemsets from hidden frequent 1-itemsets (by using Apriori algorithm [2] or FP-Growth algorithm [18,19]) and send all hidden frequent itemsets back to Owner, who can retrieve all of the frequent itemsets with its key. Without loss of generality, we take the later option in presenting our solution in the following.

Note that, Server knows that certain itemsets (represented by the token keys $\mathsf{SK}_{X'}$) are frequent. However, If Π has predicate privacy, Server cannot obtain any information about the itemset X from $SK_{X'}$ and cannot know which specific itemset (i.e., X) is frequent. On the other hand, since the algorithm $\Pi \cdot \mathsf{KeyGen}$ is randomized, with overwhelming probability, the token keys corresponding to an itemset in any two data mining tasks are different; then, the information about one token key in a data mining task is not useful in another data mining task.

Clearly, we need to construct an algorithm for Owner to retrieve frequent itemsets from hidden frequent itemsets (i.e., retrieve X from $\mathsf{SK}_{X'}$). In addition, for data recovery purpose, we also need an algorithm to retrieve plaintext transactions T from $c_{T'}$ even though this may not be necessary in outsourcing. These algorithms are not provided in existing predicate encryption schemes in the literature.

Another concern is that even though there exists an efficient symmetric-key predicate-only encryption scheme for inner products [40], it is only proved to be selectively secure, which means an adversary must commit to part of challenge strings before the setup phase of attacks. To remove this constraint, we need to construct a fully secure symmetric-key predicate-only encryption scheme for inner products. In addition, we also need further study on how to achieve soundness in our final solution.

4.1.3. Final solution

Our final solution is based on a predicate encryption scheme for inner products proposed by Lewko et al. [27] using DPVS. We modify it to obtain symmetric-key predicate-only encryption scheme for inner products and apply the dual system encryption concept recently introduced by Waters [49] to achieve full security.

[1] In order to achieve predicate privacy, which is inherently impossible to achieve in the public-key setting, Shen and Shi [40] introduced symmetric-key predicate encryption. They also considered the predicate-only variant, in which evaluating a token on a ciphertext outputs a bit indicating whether the encrypted plaintext satisfies the predicate corresponding to the token.

In our final solution, Owner generates two token keys for each 1-itemset X. The first token key is used as before, while the second token key is used to achieve soundness. In particular, Owner assigns an identifier TID for each transaction T and an identifier SID for each 1-itemset X. Owner embeds a session key $\hat{e}(g,g)^{\eta\zeta}$ to the ciphertext of T and the second token key of X, where $\eta = F_K(\mathsf{SID})$, $\zeta = F_K(\mathsf{TID}), F_K$ is a pseudorandom function, and K is a secret key. If transaction T does not contain itemset X, Server cannot obtain any information about the session key. However, if T contains X, Server can derive the session key $\hat{e}(g,g)^{\eta\zeta}$ from the ciphertext of T and the second token key of X, and send the session key to Owner as a soundness proof.

To retrieve plaintext T from $c_{T'}$, Owner performs a pairing operation on his secret key and the ciphertext $c_{T'}$ in DPVS, which has the desired properties to ensure correctness of decryption. To retrieve frequent itemsets from hidden frequent itemsets, we use an index system. In this system, Owner maintains a list of (X,SID) for all 1-itemset X, where SID is the identifier of X. The SIDs are also used to index the token keys of X when the token keys are sent to Server for mining hidden frequent itemsets. Each hidden frequent 1-itemset is represented by a 1-itemset identifier SID and associated with a set of transaction identifiers (TIDs) and session keys. After verifying the soundness of data mining results, Owner retrieves the frequent itemsets from hidden frequent itemsets according to the maintained list of (X,SID). The detail of our construction is given below.

4.2. Detail of our construction

The predicate encryption scheme for inner products proposed by Lewko et al. [27] has been proven to be attribute-hiding. We modify it to also achieve predicate privacy, which enables us to construct an outsourcing association rule mining scheme with data privacy, mining privacy, and soundness. Our solution consists of the following algorithms:

- $\mathsf{SysSetup}(1^\lambda)$. Given a security parameter λ, Owner first runs

$$(q,\mathbb{G},\mathbb{G}_T,\hat{e},g,\mathbb{V},\mathbb{B},\mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^\lambda, 2n+5),$$

where n is the number of items in $\mathcal{I}$. Let $F:\{0,1\}^{\ell_k}\times\{0,1\}^{\ell_t}\to\mathbb{F}_q$ be a family of functions. Then, Owner chooses $K_1,K_2\in\{0,1\}^{\ell_k}$ uniformly at random. Finally, Owner publishes the system's public parameter PK and keeps the system's secret key SK, where

$$\mathsf{PK} = (q,\mathbb{G},\mathbb{G}_T,\hat{e},g,\mathbb{V}),\ \mathsf{SK} = (\mathbb{B},\mathbb{B}^*,F_{K_1},F_{K_2}).$$

- $\mathsf{AddEncTransRecord}(\mathsf{SK},T)$. Given the system's secret key SK and a transaction $T=(t_1,\ldots,t_n)$, Owner first sets $T'=(t_1,\ldots,t_n,t_{n+1})$, where $t_{n+1}=-1$. Then, Owner chooses a unique transaction identifier $\mathsf{TID}\in\{0,1\}^{\ell_t}$. Owner also chooses $\delta_1,\delta_2\in\mathbb{F}_q$ uniformly at random. Next, Owner sets $\zeta = F_{K_1}(\mathsf{TID})$ and computes

$$\mathbf{c}_T = \delta_1\left(\sum_{i=1}^{n+1} t_i\mathbf{b}_i\right) + \zeta\mathbf{b}_{2n+3} + \delta_2\mathbf{b}_{2n+5}.$$

Finally, Owner sends $c_T=(\mathsf{TID},\mathbf{c}_T)$ to Server, who adds it to its database $\mathcal{D}$.

- $\mathsf{RetrieveTransRecord}(\mathsf{SK},c_T)$. Given the system's secret key SK and an encrypted transaction $c_T=(\mathsf{TID},\mathbf{c}_T)$, for $1\leqslant i\leqslant n$, Owner sets

$$t_i = \begin{cases} 0, & \text{if } \hat{e}(\mathbf{c}_T,\mathbf{b}_i^*) = 1\in\mathbb{G}_T;\\ 1, & \text{otherwise.}\end{cases}$$

Then, Owner outputs the original transaction $T=(t_1,\ldots,t_n)$. The correctness of this decryption is due to the fact that,

$$\begin{aligned}\hat{e}(\mathbf{c}_T,\mathbf{b}_i^*) &= \hat{e}\left(\delta_1\left(\sum_{i=1}^{n+1}t_i\mathbf{b}_i\right) + \zeta\mathbf{b}_{2n+3} + \delta_2\mathbf{b}_{2n+5},\mathbf{b}_i^*\right)\\ &= \hat{e}(\delta_1 t_i\mathbf{b}_i,\mathbf{b}_i^*) = \hat{e}(g,g)^{\delta_1 t_i}.\end{aligned}$$

If $t_i=0, \hat{e}(\mathbf{c}_T,\mathbf{b}_i^*) = \hat{e}(g,g)^{\delta_1 t_i} = 1$; otherwise, with overwhelming probability, $\hat{e}(\mathbf{c}_T,\mathbf{b}_i^*)\neq 1$.

- $\mathsf{GetFreqItemSets}(\mathsf{Owner},\mathsf{Server})$. In order to obtain frequent itemsets, firstly, for each 1-itemset $X=(x_1,\ldots,x_n)$, Owner proceeds as follows.
  1. Set $X'=(x_1,\ldots,x_n,x_{n+1})$, where $x_{n+1}=\sum_{i=1}^n x_i$, choose a unique 1-itemset identifier $\mathsf{SID}\in\{0,1\}^{\ell_t}$, and add the tuple (X,SID) to the list $\mathcal{ISD}$, which is empty initially.
  2. Choose random $\gamma_1,\gamma_2,\theta_1,\theta_2\in\mathbb{F}_q$, and set $\eta = F_{K_2}(\mathsf{SID})$;
  3. Compute $\mathbf{k}_1^* = \gamma_1(\sum_{i=1}^{n+1}x_i\mathbf{b}_i^*) + \gamma_2\mathbf{b}_{2n+4}^*$ and $\mathbf{k}_2^* = \theta_1(\sum_{i=1}^{n+1}x_i\mathbf{b}_i^*) + \eta\mathbf{b}_{2n+3}^* + \theta_2\mathbf{b}_{2n+4}^*$;
  4. Set the token key $\mathsf{TK}_X = (\mathsf{SID},(\mathbf{k}_1^*,\mathbf{k}_2^*))$.

Then, Owner sends all token keys to Server and a support threshold. For each token key $\mathsf{TK}_X=(\mathsf{SID},(\mathbf{k}_1^*,\mathbf{k}_2^*))$ and each encrypted transaction $c_T=(\mathsf{TID},\mathbf{c}_T)$, Server performs the following to decide whether the transaction contains the 1-itemset or not.

1. Check whether the equality $\hat{e}(\mathbf{c}_T, \mathbf{k}_1^*) = 1$ holds or not. If yes, the transaction represented by TID contains the 1-itemset represented by SID (for simplicity, we say that TID contains SID) due to the fact that $\hat{e}(\mathbf{c}_T, \mathbf{k}_1^*) = \hat{e}(\delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i) + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5}, \gamma_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*) + \gamma_2 \mathbf{b}_{2n+4}^*) = \hat{e}(\delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i),\ \gamma_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*)) = \hat{e}(g,g)^{\delta_1\gamma_1(T'\cdot X')}$ and $T' \cdot X' = 0$ iff TID contains SID.
2. To convince Owner that TID contains SID, generate a proof $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^*)$.

Given a support threshold, Server discovers all frequent SID using the above method. For each frequent SID, Server sends $(\mathsf{SID}, (\mathsf{TID}_1, \mathsf{Proof}_1), (\mathsf{TID}_2, \mathsf{Proof}_2), \ldots)$ to Owner, where Proof_i is the proof that TID_i contains SID.

Given $(\mathsf{SID}, \mathsf{TID}, \mathsf{Proof})$, if the equality $\mathsf{Proof} = \hat{e}(g,g)^{\eta\zeta}$ holds, where $\eta = F_{K_2}(\mathsf{SID})$ and $\zeta = F_{K_1}(\mathsf{TID})$, Owner is convinced that TID contains SID. This is correct due to the fact that $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^*) = \hat{e}(\delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i) + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5}, \theta_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*) + \eta \mathbf{b}_{2n+3}^* + \theta_2 \mathbf{b}_{2n+4}^*) = \hat{e}(g,g)^{\delta_1\theta_1(T'\cdot X') + \eta\zeta}$; if TID contains SID, then $T' \cdot X' = 0$ and $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^*) = \hat{e}(g,g)^{\delta_1\theta_1(T'\cdot X') + \eta\zeta} = \hat{e}(g,g)^{\eta\zeta}$.

Finally, for each frequent SID, Owner can retrieve the corresponding frequent 1-itemset by searching the list $\mathcal{ISD}$. On the other hand, Server can obtain hidden frequent k-itemsets from hidden frequent 1-itemsets by using Apriori algorithm [2] or FP-Growth algorithm [18,19], and Owner can retrieve frequent k-itemsets from hidden frequent k-itemsets.

We note that the dominant task of association rule mining in our solution is to discover the hidden frequent itemsets, which involves expensive pairing operations, while the task of discovering all frequent itemsets from hidden frequent itemsets can be performed within negligible time in comparison as it does not involve any cryptographic operations. The dominant task is performed on the Server's side in our solution.

Since in GetFreqItemSets algorithm, the token key and identifier corresponding to any particular 1-itemset are generated randomly, with overwhelming probability, they are different in different data mining tasks; therefore, an adversary cannot derive any information about frequent itemsets by comparing token keys and identifiers in different data mining tasks. In the next subsection, we will formalize the security requirements of outsourcing association rule mining and prove that our scheme satisfies the security requirements.

4.3. Security

Now we define the desired properties of outsourcing association rule mining, including *data privacy*, *mining privacy*, and *soundness*. These properties are defined based on security games between a challenger and an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ at cloud servers has the capability of adaptively obtaining plaintext–ciphertext pairs in attacks as required by semantic security. The adversary may also insert some false data into the data mining results. Our adversary model is stronger than the models in previous works on outsourcing association rule mining [14,33,41,50], which assume that the adversary is honest but curious and it does not have the capability of obtaining plaintext–ciphertext pairs in attacks. Given formal definitions on the security games, we provide security proofs of our construction with respect to desired security properties.

Firstly, we give the *data privacy* model for outsourcing association rule mining. We need to ensure that an $\mathsf{AddEncTransRecord}(\mathsf{SK}, T)$ does not reveal any information about T unless the corresponding SK is available. We define *data privacy* against an active attacker who is able to obtain ciphertexts c_T for any transaction T of his choice and token keys TK_X for any itemset X he wants. Even under such attack, the attacker should not be able to distinguish an encryption of a transaction T_0^* from an encryption of a transaction T_1^*, without SK. We define *data privacy* against an active attack $\mathcal{A}$ using the following game between a challenger and the attacker:

Setup: The challenger runs $\mathsf{SysSetup}(1^\lambda)$ to obtain a system's public parameters PK and secret key SK. It gives PK to the adversary $\mathcal{A}$ and keeps SK to itself.

Query phase 1: The adversary $\mathcal{A}$ adaptively issues queries, where each query is one of two types:

1. Ciphertext query. On the jth ciphertext query, $\mathcal{A}$ outputs a transaction T_j. In response, the challenger runs

$$c_{T_j} \leftarrow \mathsf{AddEncTransRecord}(\mathsf{SK}, T_j),$$

and gives the encrypted transaction c_{T_j} to $\mathcal{A}$.

2. Token query. On the jth token query, $\mathcal{A}$ outputs an itemset X_j. In response, the challenger generates the corresponding token key TK_{X_j} and gives it to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ submits two transactions T_0^*, T_1^*, subject to the restriction that, T_0^* and T_1^* cannot contain any of queried itemsets. The challenger selects a random bit $\beta \in \{0,1\}$, sets

$$c^* \leftarrow \mathsf{AddEncTransRecord}(\mathsf{SK}, T_\beta^*),$$

and sends c^* to the adversary as its challenge ciphertext.

Query phase 2: The adversary continues to adaptively issue additional queries as in Query phase 1, subject to the same restriction with respect to the challenge as above.

Guess: The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 3. An outsourcing association rule mining scheme achieves data privacy if all probabilistic polynomial time adversaries have at most a negligible advantage in the data privacy game.

Secondly, we give the *mining privacy* model for outsourcing association rule mining. We need to ensure that a token key TK_X does not reveal any information about the itemset X unless the corresponding encrypted transaction c_T is available. We define *mining privacy* against an active attacker who is able to obtain ciphertexts c_T for any transaction T of his choice and token keys TK_X for any itemset X he wants. Even under such attack, the attacker should not be able to distinguish a token key corresponding to a itemset X_0^* from the one corresponding to a itemset X_1^*, without encrypted transaction c_{T^*} containing X_0^* or X_1^*. We define *mining privacy* against an active attack $\mathcal{A}$ as the following game between a challenger and the attacker:

Setup: The same as it does in *data privacy* game.
Query phase 1: The same as it does in *data privacy* game.
Challenge: The adversary $\mathcal{A}$ submits two itemsets X_0^*, X_1^*, subject to the restriction that, all queried transactions cannot contain X_0^* and X_1^*. The challenger selects a random bit $\beta \in \{0,1\}$, generates the corresponding token key $\mathsf{TK}_{X_\beta^*}$ and sends it to the adversary as its challenge token key.
Query phase 2: The adversary continues to adaptively issue additional queries as in Query phase 1, subject to the same restriction with respect to the challenge as above.
Guess: The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 4. An outsourcing association rule mining scheme achieves mining privacy if all probabilistic polynomial time adversaries have at most a negligible advantage in the mining privacy game.

Finally, we give the *soundness* model for outsourcing association rule mining. We need to ensure that a malicious cloud server is not able to generate a valid Proof to convince the data owner that a transaction contains an itemset which it actually does not contain. We define *soundness* against an active attacker who is able to obtain ciphertexts c_T for any transaction T of his choice and token keys TK_X for any itemset X he wants. Even under such attack, the attacker should not be able to generate a proof Proof deceive the data owner into believing that a transaction T contains an itemset X, if T actually contains nothing about X. We define *soundness* against an active attack $\mathcal{A}$ as the following game between a challenger and the attacker:

Setup: The same as it does in *data privacy* game.
Query phase 1: The same as it does in *data privacy* game.
Output: The adversary $\mathcal{A}$ outputs a transaction identifier TID, an itemset identifier SID and a proof Proof, and succeeds if the following conditions hold.
1. The transaction T does not contain the itemset X, where T is associated with the transaction identifier TID and X is associated with the itemset identifer SID.
2. The proof Proof can convince the challenger that T contains X.

Definition 5. An outsourcing association rule mining scheme achieves soundness, if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds is negligible in the soundness game.

Note that, in our outsourcing association rule mining scheme, in order to obtain frequent itemsets, Owner needs to generate the token keys for all 1-itemsets and gives them to Server. So, in the definition of data privacy, mining privacy and soundness, we allow the adversary $\mathcal{A}$ to issue Token query. With these definitions, we have the following:

Theorem 1. *If Assumptions 1 and 2 hold and the function family F is pseudorandom, then our outsourcing association rule mining scheme achieves data privacy.*

Theorem 2. *If Assumptions 3 and 4 hold and the function family F is pseudorandom, then our outsourcing association rule mining scheme achieves mining privacy.*

Theorem 3. *If the function family F is pseudorandom, then our outsourcing association rule mining scheme achieves soundness.*

The proofs of the above theorems are provided in Appendix A.

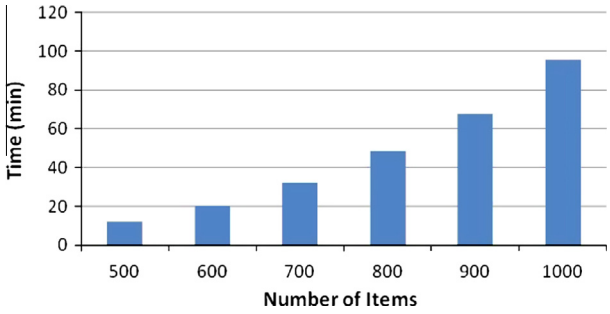

Fig. 2. The computational cost of key-pair generation.

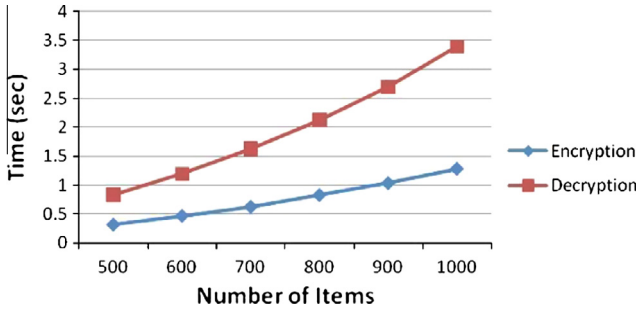

Fig. 3. The computational cost of encryption and decryption per transaction.

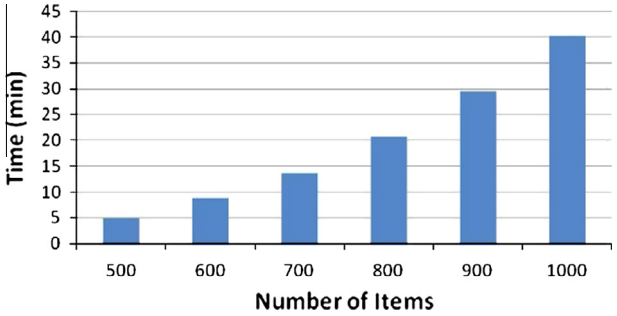

Fig. 4. The computational cost of generating token keys for all 1-itemsets.

5. Performance

We implement our scheme in C++ on Ubuntu Linux 11.10. Our cryptographic algorithms are built based on MIRACL library [39], which provides efficient implementations for modular arithmetic of big numbers. We use 160 bits for the order of group $\mathbb{G}$ and the size of secret key in pseudorandom functions F_{K_1} and F_{K_2}. The larger the order of group is, the better security it achieves at the cost of efficiency. Because of the fact that 160-bit group order strikes a balance between security and efficiency very well, it is widely employed. So is the reason why we use 160 bits for the order of group $\mathbb{G}$. Our experimental machine consists of a 2.53 GHz Intel Core 2 Duo E7200 processor and 4 GB DDR2 800 Hz RAM. All experiments are performed on synthetic data, which is generated by IBM quest market-basket synthetic data generator with similar settings as in [50]. The number of different items varies from 500 to 1000. The number of transactions ranges from 5 K to 10 K.

To illustrate how much more efficient our solution is compared to the solution with fully homomorphic encryption, we refer to [13] for the implementation of a variant of Gentry's fully homomorphic encryption scheme [12] on a more powerful platform than ours.[2] In the "medium" security setting of the implementation in [13], the time to encrypt *one* bit is 19 s, the ciphertexts size to encrypt *one* bit is 380 bits, the time to decrypt *one* bit is 0.13 s, and the time to run *one* bootstrapping operation (which is an essential operation in evaluating whether an encrypted transaction contains an encrypted itemset) is

[2] Gentry and Halevi [13] ran their implementation of fully homomorphic encryption on an IBM System x3500 server, featuring a 64-bit quad-core Intel Xeon E5450 processor, running at 3 GHz, with 12 MB L2 cache and 24 GB of RAM.

2.8 min. For lower bound estimation, only one bootstrapping operation is assumed to be performed on evaluating whether an encrypted transaction contains an encrypted itemset.

In our solution, the major computational tasks on the Owner side include generating public/private key pair, encrypting/decrypting transactions, and generating token keys for all 1-itemsets. The major computational task of Server is on discovering hidden frequent itemsets.

Fig. 2 shows the time for setting up a public/private key pair. The time is dominated by computing matrix inversion for $\mathbb{B}^*$. Note that, Owner can generate a public/private key pair off line and this can be done just once before the data mining task.

Fig. 3 shows the time for encrypting and decrypting one transaction with different number of items. Note that the encryption operation is used to protect each transaction before outsourcing, while the decryption operation is needed for data recovery purpose. In comparison, fully homomorphic encryption would take at least 5000 times more time in our experimental setting (i.e., $18 * 1000$ s for 1000 items).

Fig. 4 shows the time for generating token keys for all 1-itemsets, which is the major task performed on Owner side for outsourcing an association rule mining task. This computational cost is at least 7500 times lower than generating all encrypted 1-itemsets (i.e., "token keys") in fully homomorphic encryption, which would take about $18 * 1000 * 1000$ s for 1000 items.

Fig. 5 shows the time for discovering hidden frequent itemsets, which is the dominant task performed on Server side in our solution. In comparison, fully homomorphic encryption would take at least 60,000 times more time even if only one bootstrapping operation is performed for evaluating whether an encrypted transaction contains an encrypted itemset (it would take $2.8 * 1000 * 10{,}000$ min for 1000 items and 10,000 transactions).

Fig. 6 shows the time for retrieving frequent itemsets from hidden frequent itemsets, which is the task performed on Owner side. In comparison, when applying fully homomorphic encryption, Owner needs to perform $|\mathcal{D}|$ times of decryption operations, and it would take at least 500,000 times more time even if only one bit is decrypted (it would take $0.13 * 10{,}000 * 500$ s for 10,000 transactions and 500 items).

The communication cost of sending hidden frequent items is showed in Fig. 7. In comparison, when applying fully homomorphic encryption, Server needs to send $|\mathcal{D}| \times n$ ciphertexts to Owner and it would correspond to a 80-fold increase in communication cost even if each ciphertext is an encryption of one bit (it would take $380 * 10{,}000 * 500$ bits for 10,000 transactions and $n = 500$ items).

As expected, our solution is much more efficient compared to the solution with fully homomorphic encryption. For 1000 items and 10,000 transactions, our solution takes about 450 min to finding hidden frequent itemsets, while the fully homomorphic encryption solution takes 28,000,000 min to do that. Even though the implementation [13] of a variant of Gentry's fully homomorphic encryption scheme [12] is built on a more powerful platform than ours, the time for the fully

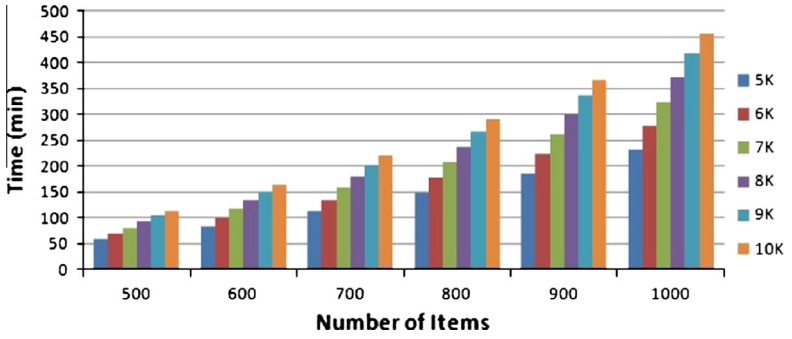

Fig. 5. The computational cost of finding hidden frequent itemsets.

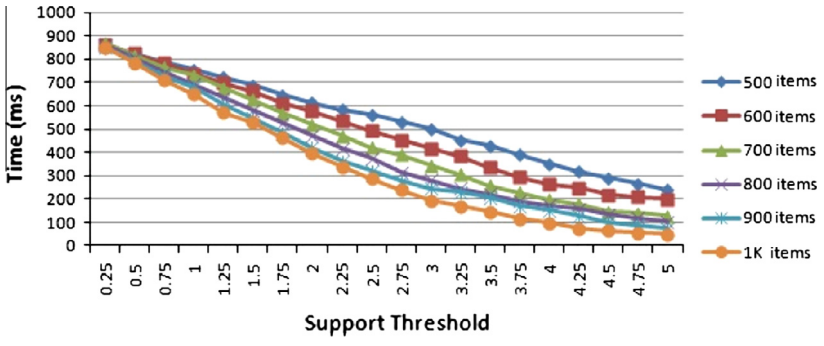

Fig. 6. The computational cost of retrieving frequent itemsets from hidden frequent itemsets with 10,000 transactions.

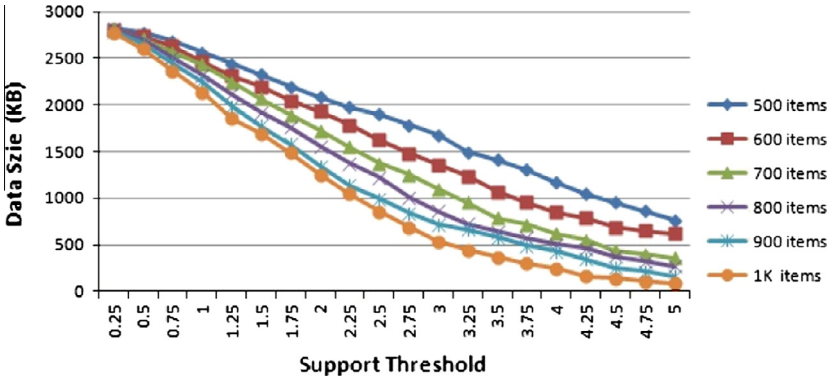

Fig. 7. The communication cost of sending hidden frequent itemsets with 10,000 transactions.

homomorphic encryption solution to discover hidden frequent itemsets is at least 60,000 times more than the time that our solution takes to perform the same task. That also means the solution with fully homomorphic encryption scheme [13] is extremely inefficient and impractical.

6. Conclusion

In this paper, we proposed the first semantically secure solution for outsourcing association rule mining with data privacy, mining privacy and soundness. Our solution is non-deterministic and secure against an adversary at cloud servers which has the capability of adaptively obtaining plaintext–ciphertext pairs as required by semantic security. The adversary may also insert false data into the data mining results. In comparison, adversary models used in previous works on outsourcing association rule mining assume that the adversary/server is honest but curious and it does not have the capability of obtaining any plaintext–ciphertext pairs in attacks. Consequently, the previous solutions, which are usually based on substitution mappings, are not semantically secure, nor can they ensure soundness for the data mining results.

One future direction is to pursue more efficient solutions that are semantically secure. It is also important to investigate how to ensure completeness of the data mining results; that is, Owner can effectively and efficiently check whether any frequent itemset is missing from the data mining results returned by Server. For completeness, perhaps the non-zero inner-product encryption in [7] can be explored towards this goal.

Acknowledgement

We are grateful to the anonymous reviewers for their helpful comments. The work of Junzuo Lai was supported by the National Natural Science Foundation of China (Nos. 61300226, 61272534, 61272453), the Research Fund for the Doctoral Program of Higher Education of China (No. 20134401120017), the Guangdong Provincial Natural Science Foundation (No. S2013040014826), and the Fundamental Research Funds for the Central Universities. The work of Yingjiu Liu was supported by the research grant 12-C220-SMU-001 from MOEs AcRF Tier 1 funding support through Singapore Management University. The work of Robert H. Deng was supported by the research grant 13-C220-SMU-05 from the Office of Research, Singapore Management University. The work of Jian Weng was supported by the National Science Foundation of China (Nos. 61272413, 61373158, 61133014, 61272415), the Fok Ying Tung Education Foundation (No. 131066), the Program for New Century Excellent Talents in University (No. NCET-12-0680) and the Research Fund for the Doctoral Program of Higher Education of China (No. 20134401110011).

Appendix A. Security proofs of our construction

To prove the security of our construction, we apply the dual system encryption concept recently introduced by Waters [49]. We first describe the correct forms of *normal* ciphertexts and *normal* token keys in our outsourcing association rule mining scheme. Then, we define two additional structures: *semi-functional* ciphertexts and *semi-functional* token keys. These will not be used in the real system, but will be used in our proof.

A *normal* ciphertext for a transaction $T = (t_1, \ldots, t_n)$ is created as follows:

1. Set $T' = (t_1, \ldots, t_n, t_{n+1})$, where $t_{n+1} = -1$.
2. Choose a unique transaction identifier $\mathsf{TID} \in \{0,1\}^{\ell_t}$, and choose $\delta_1, \delta_2 \in \mathbb{F}_q$ uniformly at random.
3. Compute $\zeta = F_{K_1}(\mathsf{TID})$ and $\mathbf{c}_T^{(norm)} = \delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i) + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5}$.
4. The normal ciphertext $c_T^{(norm)}$ is set to be $c_T^{(norm)} = (\mathsf{TID}, \mathbf{c}_T^{(norm)})$.

A *semi-functional* ciphertext for T is set to be $c_T^{(semi)} = (\mathsf{TID}, \mathbf{c}_T^{(semi)})$, where

$$\mathbf{c}_T^{(semi)} = \delta_1\left(\sum_{i=1}^{n+1} t_i \mathbf{b}_i\right) + \sum_{i=1}^{n+1} w_i \mathbf{b}_{n+1+i} + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5},$$

and $w_i \in \mathbb{F}_q$ is chosen uniformly at random.

A *normal* token key for an itemset $X = (x_1, \ldots, x_n)$ is created as follows:

1. Set $X' = (x_1, \ldots, x_n, x_{n+1})$, where $x_{n+1} = \sum_{i=1}^{n} x_i$.
2. Choose a unique itemset identifier $\mathsf{SID} \in \{0,1\}^{\ell_t}$, and choose $\gamma_1, \gamma_2, \theta_1, \theta_2 \in \mathbb{F}_q$ uniformly at random.
3. Compute $\mathbf{k}_1^{*(norm)} = \gamma_1\left(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*\right) + \gamma_2 \mathbf{b}_{2n+4}^*$.
4. Compute $\eta = F_{K_2}(\mathsf{SID})$ and $\mathbf{k}_2^{*(norm)} = \theta_1\left(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*\right) + \eta \mathbf{b}_{2n+3}^* + \theta_2 \mathbf{b}_{2n+4}^*$.
5. The normal token key $\mathsf{TK}_X^{(norm)}$ is set to be $\mathsf{TK}_X^{(norm)} = (\mathsf{SID}, \mathbf{k}^{*(norm)} = (\mathbf{k}_1^{*(norm)}, \mathbf{k}_2^{*(norm)}))$.

A *semi-functional* token key for X is set to be $\mathsf{TK}_X^{(semi)} = (\mathsf{SID}, \mathbf{k}^{*(semi)} = (\mathbf{k}_1^{*(semi)}, \mathbf{k}_2^{*(semi)}))$, where $\mathbf{k}_1^{*(semi)} = \gamma_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*) + \sum_{i=1}^{n+1} \xi_i \mathbf{b}_{n+1+i}^* + \gamma_2 \mathbf{b}_{2n+4}^*$, $\mathbf{k}_2^{*(semi)} = \theta_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^*) + \sum_{i=1}^{n+1} \varphi_i \mathbf{b}_{n+1+i}^* + \eta \mathbf{b}_{2n+3}^* + \theta_2 \mathbf{b}_{2n+4}^*$, and $\xi_i, \varphi_i \in \mathbb{F}_q$ are chosen uniformly at random.

Note that, if a transaction T contains an itemset X, then $\hat{e}(\mathbf{c}_T^{(norm)}, \mathbf{k}_1^{*(norm)}) = \hat{e}(\mathbf{c}_T^{(norm)}, \mathbf{k}_1^{*(semi)}) = \hat{e}(\mathbf{c}_T^{(semi)}, \mathbf{k}_1^{*(norm)}) = 1 \in \mathbb{G}_T$; $\hat{e}(\mathbf{c}_T^{(norm)}, \mathbf{k}_2^{*(norm)}) = \hat{e}(\mathbf{c}_T^{(norm)}, \mathbf{k}_2^{*(semi)}) = \hat{e}(\mathbf{c}_T^{(semi)}, \mathbf{k}_2^{*(norm)}) = \hat{e}(g,g)^{\eta\zeta}$; $\hat{e}(\mathbf{c}_T^{(semi)}, \mathbf{k}_1^{*(semi)})$ and $\hat{e}(\mathbf{c}_T^{(semi)}, \mathbf{k}_2^{*(semi)})$ are uniformly and independently distributed over $\mathbb{G}_T$.

A.1. Proofs of Theorem 1

Proof. We will prove *data privacy* by a hybrid argument using a sequence of games. The first game, Game 0, will be the original or real data privacy game. In Game 1, the function F_{k_1} is replaced by a random function with the same domain and range. In Game 2, the function F_{k_2} is replaced by a random function with the same domain and range. In Game 3, the challenge ciphertext given to the adversary is semi-functional. In Game 4-k, the challenge ciphertext is semi-functional, the first k token keys are semi-functional and the rest of the keys are normal. In Game 5, all token keys are semi-functional and the challenger ciphertext is a semi-functional encryption of a random transaction, not one of the transactions provided by the adversary.

Then we will prove these games are indistinguishable in the following five lemmas. Note that, in Game 5, the value of β is information-theoretically hidden from the adversary. Hence the adversary can attain no advantage in Game 5. Therefore, we conclude that the advantage of the adversary in Game 0 (i.e., the real data privacy game) is negligible. This completes the proof of theorem. □

Lemma 1. *Suppose that the function family F is pseudorandom. Then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ distinguishes between Game 0 and Game 1 with non-negligible probability. We are going to construct another probabilistic polynomial time (PPT) $\mathcal{B}$ that makes use of $\mathcal{A}$ to break the pseudorandomness of the function family F with non-negligible probability.

$\mathcal{B}$ is provided oracle access to a function f and tries to decide if f is drawn at random from F, namely $f = F_{K_1}$, where $K_1 \xleftarrow{\$} \{0,1\}^{\ell_k}$, or is drawn at random from $\mathsf{Rand}^{\{0,1\}^{\ell_t} \to \mathbb{F}_q}$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

Setup: $\mathcal{B}$ first runs $(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{ob}(1^\lambda, 2n+5)$. Then, $\mathcal{B}$ chooses $K_2 \in \{0,1\}^{\ell_k}$ uniformly at random. Finally, $\mathcal{B}$ sets the system's public information $\mathsf{PK} = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V})$, and sends PK to the adversary $\mathcal{A}$ and keeps $\mathbb{B}, \mathbb{B}^*, F_{K_2}$ to itself.

Query phase 1: The adversary $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:

– *Ciphertext query.* When the adversary $\mathcal{A}$ issues a ciphertext query on a transaction $T = (t_1, \ldots, t_n)$, $\mathcal{B}$ proceeds as follows:

1. Set $T' = (t_1, \ldots, t_n, t_{n+1})$, where $t_{n+1} = -1$.
2. Choose a unique transaction identifier $\mathsf{TID} \in \{0,1\}^{\ell_t}$, and $\delta_1, \delta_2 \in \mathbb{F}_q$ uniformly at random.
3. Query its oracle on TID and set the result as ζ. Note that, if $f = F_{K_1}$, then $\zeta = F_{K_1}(\mathsf{TID})$; otherwise, ζ is a random element in $\mathbb{F}_q$.
4. Compute $\mathbf{c}_T = \delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i) + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5}$, and send $c_T = (\mathsf{TID}, \mathbf{c}_T)$ to $\mathcal{A}$.

– *Token query.* When the adversary $\mathcal{A}$ issues a token key query on an itemset $X = (x_1, \ldots, x_n)$, $\mathcal{B}$ generates a normal token key for X using $\mathbb{B}^*$ and F_{K_2}, and sends it to $\mathcal{A}$.

Challenge: At this point, $\mathcal{A}$ submits two transactions T_0^*, T_1^*. $\mathcal{B}$ selects a random bit $\beta \in \{0,1\}$. Then, $\mathcal{B}$ creates a ciphertext c_T^* for T_β^* as the response of Ciphertext query in Query phase 1, and sends c_T^* to $\mathcal{A}$.

Query phase 2: The adversary continues to adaptively issue additional queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.

Note that, if $f = F_{K_1}$, then $\mathcal{B}$ has properly simulated Game 0. If f is a random function, namely $f \xleftarrow{\$} \mathsf{Rand}^{\{0,1\}^{\ell_t} \to \mathbb{F}_q}$, then $\mathcal{B}$ has properly simulated Game 1. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for f. □

Lemma 2. *Suppose that the function family F is pseudorandom. Then Game 1 and Game 2 are computationally indistinguishable.*

Proof. Since the proof of this lemma is essentially the same as that for Lemma 1, we omit the proof. □

Lemma 3. *Suppose that $\mathcal{G}_{ob}$ satisfies Assumption 1. Then Game 2 and Game 3 (or Game 4–0) are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ distinguishes between Game 2 and Game 3 (or Game 4–0) with non-negligible probability. Then we can build an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to break Assumption 1.

$\mathcal{B}$ receives $(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \{\mathbf{e}_{b,i}\}_{i=1,\ldots,n+1})$ and tries to decide if $b = 0$ or 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

Setup: $\mathcal{B}$ sets the system's public information $\mathsf{PK} = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V})$, and sends PK to the adversary $\mathcal{A}$.

Query phase 1: The adversary $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:

- *Ciphertext query.* When the adversary $\mathcal{A}$ issues a ciphertext query on a transaction $T = (t_1, \ldots, t_n)$, $\mathcal{B}$ generates a normal ciphertext for T using $\widehat{\mathbb{B}}$, and sends it to $\mathcal{A}$.
- *Token query.* When the adversary $\mathcal{A}$ issues a token key query on an itemset $X = (x_1, \ldots, x_n)$, $\mathcal{B}$ generates a normal token key for X using $\widehat{\mathbb{B}}^*$, and sends it to $\mathcal{A}$.

Challenge: At this point, $\mathcal{A}$ submits two transactions T_0^*, T_1^*. $\mathcal{B}$ proceeds as follows:

1. Select a random bit $\beta \in \{0,1\}$, parse T_β^* as $(t_1^*, \ldots, t_n^*)$ and set $t_{n+1}^* = -1$.
2. Choose a unique transaction identifier $\mathsf{TID}^* \in \{0,1\}^{\ell_t}$, and $\zeta^* \in \mathbb{F}_q$ uniformly at random.
3. Compute $\mathbf{c}_T^* = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_{b,i} + \zeta^* \mathbf{b}_{2n+3}$, and send $c_T^* = (\mathsf{TID}^*, \mathbf{c}_T^*)$ to $\mathcal{A}$.

Query phase 2: The adversary continues to adaptively issue additional queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.

Note that, if $b = 0$, then

$$\mathbf{c}_T^* = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_{b,i} + \zeta^* \mathbf{b}_{2n+3} = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_{0,i} + \zeta^* \mathbf{b}_{2n+3} = \delta_1 \left(\sum_{i=1}^{n+1} t_i^* \mathbf{b}_i \right) + \zeta^* \mathbf{b}_{2n+3} + \left(\sum_{i=1}^{n+1} t_i^* \delta_{2,i} \right) \mathbf{b}_{2n+5},$$

and $\mathcal{B}$ has properly simulated Game 2. If $b = 1$, then

$$\mathbf{c}_T^* = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_{b,i} + \zeta^* \mathbf{b}_{2n+3} = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_{1,i} + \zeta^* \mathbf{b}_{2n+3} = \delta_1 \left(\sum_{i=1}^{n+1} t_i^* \mathbf{b}_i \right) + \sum_{i=1}^{n+1} \left(\sum_{j=1}^{n+1} \rho t_j^* u_{j,i} \right) \mathbf{b}_{n+1+i} + \zeta^* \mathbf{b}_{2n+3} + \left(\sum_{i=1}^{n+1} t_i^* \delta_{2,i} \right) \mathbf{b}_{2n+5},$$

and $\mathcal{B}$ has properly simulated Game 3 (or Game 4–0). Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for b. □

Lemma 4. *Suppose that $\mathcal{G}_{ob}$ satisfies Assumption 2. Then Game 4-(k-1) and Game 4-k are computationally indistinguishable for $1 \leqslant k \leqslant v$, where v denotes the number of token key queries the adversary makes.*

Proof. Suppose there exists an adversary $\mathcal{A}$ distinguishes between Game 4-(k-1) and Game 4-k with non-negligible probability. Then we can build an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to break Assumption 2.

$\mathcal{B}$ receives $(q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \{\mathbf{h}_{b,i}^*, \mathbf{e}_i\}_{i=1,\ldots,n+1})$ and tries to decide if $b = 0$ or 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

Setup: $\mathcal{B}$ sets the system's public information $\mathsf{PK} = (q, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, \mathbb{V})$, and sends PK to the adversary $\mathcal{A}$.

Query phase 1: The adversary $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:

- Ciphertext query. When the adversary $\mathcal{A}$ issues a ciphertext query on a transaction $T = (t_1, \ldots, t_n)$, $\mathcal{B}$ generates a normal ciphertext for T using $\widehat{\mathbb{B}}$, and sends it to $\mathcal{A}$.

– Token query. When the adversary $\mathcal{A}$ issues the j-th token key query on an itemset $X = (x_1, \ldots, x_n)$, $\mathcal{B}$ proceeds as follows. If $j < k$, $\mathcal{B}$ creates a semi-functional token key for X using $\widehat{\mathbb{B}}^*$, and sends it to $\mathcal{A}$. If $j > k$, $\mathcal{B}$ creates a normal token key for X using $\widehat{\mathbb{B}}^*$, and sends it to $\mathcal{A}$. If $j = k$, $\mathcal{B}$ performs as follows:

1. Set $X' = (x_1, \ldots, x_n, x_{n+1})$, where $x_{n+1} = \sum_{i=1}^{n} x_i$.
2. Choose a unique itemset identifier $\mathsf{SID} \in \{0,1\}^{\ell_t}$, and choose $\gamma, \gamma', \theta, \theta', \eta \in \mathbb{F}_q$ uniformly at random.
3. Compute $\mathbf{k}_1^* = \gamma(\sum_{i=1}^{n+1} x_i \mathbf{h}_{b,i}^*) + \gamma' \mathbf{b}_{2n+4}^*$ and $\mathbf{k}_2^* = \theta(\sum_{i=1}^{n+1} x_i \mathbf{h}_{b,i}^*) + \eta \mathbf{b}_{2n+3}^* + \theta' \mathbf{b}_{2n+4}^*$.
4. Send $\mathsf{TK}_X = (\mathsf{SID}, \mathbf{k}^* = (\mathbf{k}_1^*, \mathbf{k}_2^*))$ to $\mathcal{A}$.

Note that, if $b = 0$, then TK_X is a normal token key for X. If $b = 1$, then TK_X is a semi-functional token key for X.

Challenge: At this point, $\mathcal{A}$ submits two transactions T_0^*, T_1^*. $\mathcal{B}$ proceeds as follows:

1. Select a random bit $\beta \in \{0,1\}$, parse T_β^* as $(t_1^*, \ldots, t_n^*)$ and set $t_{n+1}^* = -1$.
2. Choose a unique transaction identifier $\mathsf{TID}^* \in \{0,1\}^{\ell_t}$, and $\zeta^*, \delta_2^* \in \mathbb{F}_q$ uniformly at random.
3. Compute $\mathbf{c}_T^* = \sum_{i=1}^{n+1} t_i^* \mathbf{e}_i + \zeta^* \mathbf{b}_{2n+3} + \delta_2^* \mathbf{b}_{2n+5}$, and send $c_T^* = (\mathsf{TID}^*, \mathbf{c}_T^*)$ to $\mathcal{A}$.

Note that, c_T^* is a semi-functional encryption of T_β^*.

Query phase 2: The adversary continues to adaptively issue additional queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.

Obviously, if $b = 0$, then $\mathcal{B}$ has properly simulated Game 4-(k-1). If $b = 1$, then $\mathcal{B}$ has properly simulated Game 4-k. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for b. □

Lemma 5. *Game 4-v and Game 5 are equivalent.*

Proof. To prove this lemma, we will show distribution $(\{\mathbf{k}^{(j)*}\}_{j=1,\ldots,v}, \{\mathbf{c}_T^{(j)}\}_{j=1,\ldots,\varrho}, \mathbf{c}_T^*)$ in Game 4-v and that in Game 5 are equivalent, where v, ϱ denote the number of token key and ciphertext queries the adversary makes, respectively.

Now, we define new bases $\mathbb{D}$ and $\mathbb{D}^*$ of $\mathbb{V}$ as follows:

1. Generate random $Z = (\xi_{i,s}) \xleftarrow{\$} \mathbb{F}_q^{(n+1)\times(n+1)}$, $\theta_i \xleftarrow{\$} \mathbb{F}_q$, for $i = 1, \ldots, n+1$ and $s = 1, \ldots, n+1$.
2. Set $\mathbf{d}_i = \mathbf{b}_i$ and $\mathbf{d}_{n+1+i} = \mathbf{b}_{n+1+i} - \sum_{s=1}^{n+1} \xi_{i,s} \mathbf{b}_s - \theta_i \mathbf{b}_{2n+3}$ for $i = 1, \ldots, n+1$. Then, set $\mathbf{d}_{2n+3} = \mathbf{b}_{2n+3}, \mathbf{d}_{2n+4} = \mathbf{b}_{2n+4}$, $\mathbf{d}_{2n+5} = \mathbf{b}_{2n+5}$.
3. Set $\mathbf{d}_i^* = \mathbf{b}_i^* + \sum_{s=1}^{n+1} \xi_{s,i} \mathbf{b}_{n+1+s}^*$ and $\mathbf{d}_{n+1+i}^* = \mathbf{b}_{n+1+i}^*$ for $i = 1, \ldots, n+1$. Then, set $\mathbf{d}_{2n+3}^* = \mathbf{b}_{2n+3}^* + \sum_{s=1}^{n+1} \theta_s \mathbf{b}_{n+1+s}^*$, $\mathbf{d}_{2n+4}^* = \mathbf{b}_{2n+4}^*, \mathbf{d}_{2n+5}^* = \mathbf{b}_{2n+5}^*$.
4. Set $\mathbb{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_{2n+5})$ and $\mathbb{D}^* = (\mathbf{d}_1^*, \ldots, \mathbf{d}_{2n+5}^*)$.

It is easy to verify that $\mathbb{D}$ and $\mathbb{D}^*$ are dual orthonormal, and are distributed the same as the original bases, $\mathbb{B}$ and $\mathbb{B}^*$. Token keys and ciphertexts $(\{\mathbf{k}^{(j)*}\}_{j=1,\ldots,v}, \{\mathbf{c}_T^{(j)}\}_{j=1,\ldots,\varrho}, \mathbf{c}_T^*)$ in Game 4-v are expressed over bases $\mathbb{B}$ and $\mathbb{B}^*$ as

$$\mathbf{k}^{(j)*} = (\mathbf{k}_1^{(j)*}, \mathbf{k}_2^{(j)*}),$$

$$\mathbf{k}_1^{(j)*} = \gamma_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{b}_i^* \right) + \sum_{i=1}^{n+1} \xi_i^{(j)} \mathbf{b}_{n+1+i}^* + \gamma_2^{(j)} \mathbf{b}_{2n+4}^*,$$

$$\mathbf{k}_2^{(j)*} = \theta_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{b}_i^* \right) + \sum_{i=1}^{n+1} \varphi_i^{(j)} \mathbf{b}_{n+1+i}^* + \eta^{(j)} \mathbf{b}_{2n+3}^* + \theta_2^{(j)} \mathbf{b}_{2n+4}^*,$$

$$\mathbf{c}_T^{(j)} = \delta_1^{(j)} \left(\sum_{i=1}^{n+1} t_i^{(j)} \mathbf{b}_i \right) + \zeta^{(j)} \mathbf{b}_{2n+3} + \delta_2^{(j)} \mathbf{b}_{2n+5},$$

$$\mathbf{c}_T^* = \delta_1^* \left(\sum_{i=1}^{n+1} t_i^* \mathbf{b}_i \right) + \sum_{i=1}^{n+1} w_i \mathbf{b}_{n+1+i} + \zeta^* \mathbf{b}_{2n+3} + \delta_2^* \mathbf{b}_{2n+5}.$$

Observe that,

$$\mathbf{k}_1^{(j)*} = \gamma_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{b}_i^* \right) + \sum_{i=1}^{n+1} \xi_i^{(j)} \mathbf{b}_{n+1+i}^* + \gamma_2^{(j)} \mathbf{b}_{2n+4}^* = \gamma_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{d}_i^* \right) + \sum_{s=1}^{n+1} \xi_s^{(j)\prime} \mathbf{d}_{n+1+s}^* + \gamma_2^{(j)} \mathbf{d}_{2n+4}^*,$$

where $\xi_s^{(j)\prime} = \xi_s^{(j)} - \gamma_1^{(j)} \sum_{i=1}^{n+1} x_i^{(j)} \xi_{s,i}$, which are uniformly, independently distributed.

$$\mathbf{k}_2^{(j)*} = \theta_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{b}_i^* \right) + \sum_{i=1}^{n+1} \varphi_i^{(j)} \mathbf{b}_{n+1+i}^* + \eta^{(j)} \mathbf{b}_{2n+3}^* + \theta_2^{(j)} \mathbf{b}_{2n+4}^* = \theta_1^{(j)} \left(\sum_{i=1}^{n+1} x_i^{(j)} \mathbf{d}_i^* \right) + \sum_{s=1}^{n+1} \varphi_s^{(j)\prime} \mathbf{d}_{n+1+s}^* + \eta^{(j)} \mathbf{d}_{2n+3}^* + \theta_2^{(j)} \mathbf{b}_{2n+4}^*,$$

where $\varphi_s^{(j)\prime} = \varphi_s^{(j)} - \theta_1^{(j)} \sum_{i=1}^{n+1} x_i^{(j)} \xi_{s,i} - \eta^{(j)} \theta_s$, which are uniformly, independently distributed.

$$\mathbf{c}_T^{(j)} = \delta_1^{(j)} \left(\sum_{i=1}^{n+1} t_i^{(j)} \mathbf{b}_i \right) + \zeta^{(j)} \mathbf{b}_{2n+3} + \delta_2^{(j)} \mathbf{b}_{2n+5} = \delta_1^{(j)} \left(\sum_{i=1}^{n+1} t_i^{(j)} \mathbf{d}_i \right) + \zeta^{(j)} \mathbf{d}_{2n+3} + \delta_2^{(j)} \mathbf{d}_{2n+5},$$

$$\mathbf{c}_T^{*} = \delta_1^{*} \left(\sum_{i=1}^{n+1} t_i^{*} \mathbf{b}_i \right) + \sum_{i=1}^{n+1} w_i \mathbf{b}_{n+1+i} + \zeta^{*} \mathbf{b}_{2n+3} + \delta_2^{*} \mathbf{b}_{2n+5} = \sum_{s=1}^{n+1} t_s' \mathbf{d}_s + \sum_{i=1}^{n+1} w_i \mathbf{d}_{n+1+i} + \zeta^{*\prime} \mathbf{d}_{2n+3} + \delta_2^{*} \mathbf{d}_{2n+5},$$

where $t_s' = \delta_1^{*} t_s^{*} + \sum_{i=1}^{n+1} w_i \xi_{i,s}$, $\zeta^{*\prime} = \zeta^{*} + \sum_{i=1}^{n+1} w_i \theta_i$, which are uniformly, independently distributed.

Therefore, $(\{\mathbf{k}^{(j)*}\}_{j=1,\ldots,v}, \{\mathbf{c}_T^{(j)}\}_{j=1,\ldots,\varrho}, \mathbf{c}_T^{*})$ can be expressed as token keys and ciphertexts in two ways, in Game 4-v over bases $(\mathbb{B}, \mathbb{B}^{*})$ and in Game 5 over bases $(\mathbb{D}, \mathbb{D}^{*})$. Thus, Game 4-$v$ can be conceptually changed to Game 5. □

A.2. Proofs of Theorem 2

Proof. We will prove *mining privacy* by a hybrid argument using a sequence of games. The first game, Game 0, will be the original or real mining privacy game. In Game 1, the function F_{k_1} is replaced by a random function with the same domain and range. In Game 2, the function F_{k_2} is replaced by a random function with the same domain and range. In Game 3, the challenge token key given to the adversary is semi-functional. In Game 4-k, the challenge token key is semi-functional, the first k ciphertexts are semi-functional and the rest of the ciphertexts are normal. In Game 5, all ciphertexts are semi-functional and the challenger token key is a semi-functional token key of a random itemset, not one of the itemsets provided by the adversary.

Then we will prove these games are indistinguishable in the following five lemmas. Note that, in Game 5, the value of β is information-theoretically hidden from the adversary. Hence the adversary can attain no advantage in Game 5. Therefore, we conclude that the advantage of the adversary in Game 0 (i.e., the real mining privacy game) is negligible. This completes the proof of theorem.

Lemma 6. *Suppose that the function family F is pseudorandom. Then Game 0 and Game 1 are computationally indistinguishable.*

Lemma 7. *Suppose that the function family F is pseudorandom. Then Game 1 and Game 2 are computationally indistinguishable.*

Lemma 8. *Suppose that $\mathcal{G}_{ob}$ satisfies Assumption 3. Then Game 2 and Game 3 (or Game 4–0) are computationally indistinguishable.*

Lemma 9. *Suppose that $\mathcal{G}_{ob}$ satisfies Assumption 4. Then Game 4-(k-1) and Game 4-k are computationally indistinguishable for $1 \leqslant k \leqslant \varrho$, where ϱ denotes the number of ciphertext queries the adversary makes.*

Lemma 10. *Game 4-ϱ and Game 5 are equivalent.*

The proofs of Lemmas 6–10 are similar to those of Lemmas 1–5, respectively. We omit the proofs in order to keep the paper compact.

A.3. Proofs of Theorem 3

Proof. Let $c_T = (\mathsf{TID}, \mathbf{c}_T)$ be a ciphertext for a transaction T and $\mathsf{TK}_X = (\mathsf{SID}, (\mathbf{k}_1^{*}, \mathbf{k}_2^{*}))$ be a token key for an itemset X. If T contains X, in our construction, Server computes $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^{*})$ as a proof to convince Owner that T contains X. Note that, $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^{*}) = \hat{e}(\delta_1(\sum_{i=1}^{n+1} t_i \mathbf{b}_i) + \zeta \mathbf{b}_{2n+3} + \delta_2 \mathbf{b}_{2n+5},\ \theta_1(\sum_{i=1}^{n+1} x_i \mathbf{b}_i^{*}) + \eta \mathbf{b}_{2n+3}^{*} + \theta_2 \mathbf{b}_{2n+4}^{*}) = \hat{e}(g,g)^{\delta_1 \theta_1 (T\prime \cdot X\prime) + \eta\zeta}$. If T contains X, then $T\prime \cdot X\prime = 0$ and $\mathsf{Proof} = \hat{e}(\mathbf{c}_T, \mathbf{k}_2^{*}) = \hat{e}(g,g)^{\delta_1 \theta_1 (T\prime \cdot X\prime) + \eta\zeta} = \hat{e}(g,g)^{\eta\zeta}$, where $\eta = F_{K_2}(\mathsf{SID})$ and $\zeta = F_{K_1}(\mathsf{TID})$. Otherwise, Proof is uniformly distributed in $\mathbb{G}_T$ due to the assumption that the function family F is pseudorandom. So, if T does not contain X, the probability that an adversary outputs $\mathsf{Proof} = \hat{e}(g,g)^{\eta\zeta}$ is negligible. Therefore, we conclude that our outsourcing association rule mining scheme achieves soundness. □

References

[1] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, in: SIGMOD Conference, pp. 207–216.
[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: VLDB, pp. 487–499.
[3] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD'00, pp. 439–450.

[4] B. Alatas, E. Akin, An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules, Soft Comput. 10 (2006) 230–237.
[5] J. Alcalá-Fdez, R. Alcalá, F. Herrera, A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning, IEEE Trans. Fuzzy Syst. 19 (2011) 857–872.
[6] J. Alcalá-Fdez, N.F. Papè, A. Bonarini, F. Herrera, Analysis of the effectiveness of the genetic algorithms based on extraction of association rules, Fundam. Inform. 98 (2010) 1–14.
[7] N. Attrapadung, B. Libert, Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation, in: Public Key Cryptography, pp. 384–402.
[8] C.H. Chen, T.P. Hong, V.S. Tseng, Finding pareto-front membership functions in fuzzy data mining, Int. J. Comput. Intell. Syst. 5 (2012) 343–354.
[9] A.V. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke, Privacy preserving mining of association rules, in: KDD, pp. 217–228.
[10] B.C.M. Fung, K. Wang, R. Chen, P.S. Yu, Privacy-preserving data publishing: a survey of recent developments, ACM Comput. Surv. 42 (2010).
[11] J. Gehrke, D. Kifer, A. Machanavajjhala, Privacy in Data Publishing, in: ICDE, p. 1213.
[12] C. Gentry, Fully Homomorphic Encryption Using Ideal Lattices, in: STOC, pp. 169–178.
[13] C. Gentry, S. Halevi, Implementing Gentry's Fully-Homomorphic Encryption Scheme, Cryptology ePrint Archive, Report 2010/520, 2010 <http://eprint.iacr.org/>.
[14] F. Giannotti, L.V. Lakshmanan, A. Monreale, D. Pedreschi, H. Wang, Privacy-preserving mining of association rules from outsourced transaction databases, in: SPCC Workshop.
[15] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, in: STOC, pp. 218–229.
[16] S. Goldwasser, S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information, in: STOC, pp. 365–377.
[17] G. Grahne, J. Zhu, Efficiently using prefix-trees in mining frequent itemsets, in: FIMI.
[18] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: SIGMOD Conference, pp. 1–12.
[19] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Min. Knowl. Discov. 8 (2004) 53–87.
[20] J. Hipp, U. Güntzer, G. Nakhaeizadeh, Algorithms for association rule mining – a general survey and comparison, SIGKDD Explor. 2 (2000) 58–64.
[21] T.P. Hong, C.H. Chen, Y.C. Lee, Y.L. Wu, Genetic-fuzzy data mining with divide-and-conquer strategy, IEEE Trans. Evolut. Comput. 12 (2008) 252–265.
[22] M. Kantarcioglu, C. Clifton, Privacy-preserving distributed mining of association rules on horizontally partitioned data, in: DMKD.
[23] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, Cryptology ePrint Archive, Report 2007/404, 2007 <http://eprint.iacr.org/>.
[24] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in: EUROCRYPT, pp. 146–162.
[25] M. Kaya, R. Alhajj, Genetic algorithm based framework for mining fuzzy association rules, Fuzzy Sets Syst. 152 (2005) 587–601.
[26] S. Kotsiantis, D. Kanellopoulos, Association rules mining: a recent overview, GESTS Int. Trans. Comput. Sci. Eng. 32 (2006) 71–82.
[27] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in: EUROCRYPT, pp. 62–91.
[28] J.L. Lin, J.Y.C. Liu, Privacy preserving itemset mining through fake transactions, in: SAC, pp. 375–379.
[29] Y. Lindell, B. Pinkas, Privacy preserving data mining, in: CRYPTO, pp. 36–54.
[30] J.M. Luna, J.R. Romero, S. Ventura, Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules, Knowl. Inform. Syst. 32 (2012) 53–76.
[31] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, l-diversity: privacy beyond k-anonymity, in: ICDE, p. 24.
[32] A. Mohaisen, N.S. Jho, D. Hong, D. Nyang, Privacy preserving association rule mining revisited: privacy enhancement and resources efficiency, IEICE Trans. 93-D (2010) 315–325.
[33] I. Molloy, N. Li, T. Li, On the (in)security and (im)practicality of outsourcing precise association rule mining, in: ICDM, pp. 872–877.
[34] B. Pinkas, Cryptographic techniques for privacy-preserving data mining, SIGKDD Explorat. 4 (2002) 12–19.
[35] A. Salleb-Aouissi, C. Vrain, C. Nortet, Quantminer: a genetic algorithm for mining quantitative association rules, in: IJCAI, pp. 1035–1040.
[36] A. Savasere, E. Omiecinski, S.B. Navathe, An efficient algorithm for mining association rules in large databases, in: VLDB, pp. 432–444.
[37] Y. Saygin, V.S. Verykios, C. Clifton, Using unknowns to prevent discovery of association rules, SIGMOD Rec. 30 (2001) 45–54.
[38] Y. Saygin, V.S. Verykios, A.K. Elmagarmid, Privacy preserving association rule mining, in: RIDE, pp. 151–158.
[39] M. Scott, Miracl Library, 2007 <http://www.shamus.ie>.
[40] E. Shen, E. Shi, B. Waters, Predicate privacy in encryption systems, in: TCC, pp. 457–473.
[41] C.H. Tai, P.S. Yu, M.S. Chen, k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining, in: KDD, pp. 473–482.
[42] J. Vaidya, C. Clifton, Privacy preserving association rule mining in vertically partitioned data, in: KDD, pp. 639–644.
[43] J.M. Vázquez, J.L.Á. Macías, J.C.R. Santos, Mining numeric association rules via evolutionary algorithm, in: 5th International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 264–267.
[44] J.M. Vázquez, J.L.Á. Macías, J.C.R. Santos, Discovering numeric association rules via evolutionary algorithm, in: PAKDD, pp. 40–51.
[45] J.M. Vázquez, J.L.Á. Macías, J.C.R. Santos, An evolutionary algorithm to discover numeric association rules, in: SAC, pp. 590–594.
[46] V.S. Verykios, A.K. Elmagarmid, E. Bertino, Y. Saygin, E. Dasseni, Association rule hiding, IEEE Trans. Knowl. Data Eng. 16 (2004) 434–447.
[47] V.S. Verykios, A. Gkoulalas-Divanis, A survey of association rule hiding methods for privacy, in: Privacy-Preserving Data Mining, 2008, pp. 267–289.
[48] C. Wang, C. Tjortjis, Prices: an efficient algorithm for mining association rules, in: IDEAL, pp. 352–358.
[49] B. Waters, Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions, in: CRYPTO, pp. 619–636.
[50] W.K. Wong, D.W. Cheung, E. Hung, B. Kao, N. Mamoulis, Security in outsourcing of association rule mining, in: VLDB, pp. 111–122.
[51] X. Xiao, Y. Tao, Anatomy: simple and effective privacy preservation, in: VLDB, pp. 139–150.
[52] Y. Xu, K. Wang, A.W.C. Fu, P.S. Yu, Anonymizing transaction databases for publication, in: KDD, pp. 767–775.
[53] Z. Yang, S. Zhong, R.N. Wright, Anonymity-preserving data collection, in: KDD, pp. 334–343.
[54] A.C.C. Yao, How to generate and exchange secrets (extended abstract), in: FOCS, pp. 162–167.
[55] Z. Zhu, W. Du, K-anonymous association rule hiding, in: ASIACCS, pp. 305–309.