Secure Outsourced Frequent Pattern Mining by Fully Homomorphic Encryption

Junqiang Liu[1(✉)], Jiuyong Li[2], Shijian Xu[1], and Benjamin C.M. Fung[3]

[1] School of Information and Electronic Engineering,
Zhejiang Gongshang University, Hangzhou 310018, China
jjliu@alumni.sfu.ca
[2] School of Information Technology and Mathematical Sciences,
University of South Australia, Adelaide, SA 5001, Australia
[3] School of Information Studies, McGill University,
Montreal, QC H3A 1X1, Canada

Abstract. With the advent of the big data era, outsourcing data storage together with data mining tasks to cloud service providers is becoming a trend, which however incurs security and privacy issues. To address the issues, this paper proposes two protocols for mining frequent patterns securely on the cloud by employing fully homomorphic encryption. One protocol requires little communication between the client and the cloud service provider, the other incurs less computation cost. Moreover, a new privacy notion, namely α-pattern uncertainty, is proposed to reinforce the second protocol. Our scenario has two advantages: one is stronger privacy protection, and the other is that the outsourced data can be used in different mining tasks. Experimental evaluation demonstrates that the proposed protocols provide a feasible solution to the issues.

Keywords: Privacy and security · Big data · Data mining · Frequent patterns · Cloud computing

1 Introduction

The big data era is coming with exponentially growing data and increasingly complicated technologies. On one hand, big data are becoming important assets, and people are more and more interested in applying data mining technology to better utilize such assets. On the other hand, the huge volume of big data and the great complexity of technologies make it hard for an average user or business to manage and analyze their data. Therefore, outsourcing both data storage and data mining to cloud service providers is becoming a trend.

While outsourcing data storage and data mining benefits from the scale of economy and greatly reduces the complexity of deploying information technology, it comes with the privacy and security issues [10], one of which is the risk of disclosing sensitive information in the mining process.

In the literature, there are many works on privacy preserving data mining. Some works [8,15,17,21] retain the mining models at the aggregate level and

S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 70–81, 2015.
DOI: 10.1007/978-3-319-22729-0_6

preserve the privacy at the data record level. Some [5,20] preserve the privacy of data records when publishing the mining results. Others [12,19,22] retain the precise supports of patterns while observing k-support anonymity. However, no prior work preserves both the privacy of data and the privacy of mining results while retaining the exactness of the mining results in outsourcing frequent pattern mining. Moreover, the randomization and group based approaches [2] employed by the prior works cannot deal with such a scenario.

Fortunately, fully homomorphic encryption emerged as the most promising method to address security and privacy issues in cloud computing. The concept of full homomorphism was first introduced by Rivest et al. [16] in 1978, and not realized until Gentry [9] proposed an ideal lattice based scheme in 2009. Since then, there were efforts applying homomorphic encryption in data mining, for example, in answering private queries [13] and in similarity ranking [6], but to the best of our knowledge no work in frequent pattern mining.

In this paper, we consider a secured outsourcing scenario, that is, to keep encrypted data in the storage provided by a cloud service provider and to outsource data mining tasks to the service provider by employing algorithms that can work on the encrypted data directly. Such a scenario has two advantages: One is that the encrypted data can be used in various mining tasks while the randomized data by the priori works can only be used in a specific task. The other advantage is that by employing fully homomorphic encryption, it releases much less information in the mining process and thus provides more protection.

Our scenario is nontrivial as the fully homomorphic encryption supports no comparison operation, i.e., no predicate for a total order. To deal with such a challenge, we propose two secured protocols. The main contributions are as follows: First, we propose one protocol, for outsourcing both the encrypted data and the frequent pattern mining task to a cloud service provider, that requires little communication between the cloud service provider and the client. Second, we propose a new privacy notion, namely α-pattern uncertainty, and implement this privacy notion by shadow mappings. By employing both α-pattern uncertainty and fully homomorphic encryption, we propose another secured protocol for mining frequent patterns, which incurs less computation cost. Experimental evaluation shows that our protocols provide a feasible solution to addressing security and privacy issues when outsourcing frequent pattern mining.

The rest of this paper is organized as follows: Sect. 2 reviews the related works. Section 3 presents our scenario for securing outsourced frequent pattern mining. Section 4 proposes our first secured protocol based on fully homomorphic encryption. Section 5 proposes the notion of α-pattern uncertainty and our second protocol. Section 6 evaluates our protocols. Section 7 concludes the paper.

2 Related Works

Our work relates to prior works on privacy preserving frequent pattern mining and association rule mining. We can only review the most relevant and representative works due to space limit.

The works [8,15,17,21] retain the data mining models at the aggregate level and preserve the privacy at the level of individual data records by random perturbation or approximation [2]. Concretely, [8,17] proposed to randomize the data for limiting privacy breaches and to send the randomized data to the server for data mining. [15] presented a Bloom filter based solution for outsourcing the task of mining association rules. [21] proposed to generate synthetic data in such a way that the frequent patterns discovered in the original data can be mined in the synthetic data and the privacy leakage can be limited.

The works [5,20] preserve the privacy of individual records when publishing the mining results. [20] hides a subset of sensitive association rules by adding items to the original data or removing items to tune the supports and confidences of sensitive rules. [5] deals with privacy threats embedded in mining results by distorting the supports of patterns to eliminate inference channels.

The works [12,19,22] retain precise patterns with precise supports and observe the k-support anonymity notion, i.e., each (transformed) item is indistinguishable from at least $k-1$ other items w.r.t. their supports, under the assumption that the attacker knows the exact supports of certain items. Their approach is to map items to meaningless symbols and to add noise to prevent re-identification. In [12], the noise is a set of fake transactions consisting of the original items. In [22], the noise is introduced by adding fake items into the original transactions. In [19], a pseudo taxonomy tree is employed to facilitate hiding of the original items and to limit the fake items.

Our work is the first that employs fully homomorphic encryption in privacy preserving frequent pattern mining although there are works employing fully homomorphic encryption in other tasks. For example, Hu et al. [13] answered private queries over a data cloud, and Chu et al. [6] proposed a privacy preserving similarity ranking protocol based on fully homomorphic encryption.

3 Preliminaries

We consider a secured scenario for outsourcing frequent pattern mining to a cloud service provider with a guarantee of the privacy and the exactness of both the data and the mining results, faciliatated by fully homomorphic encryption.

3.1 Frequent Pattern Mining Problem

Frequent pattern mining is one of the fundamental data mining problems [3,11] with a variety of applications, for example, consumer behavior analysis, design of goods shelves, inventory control, product promotion, and so on. Given a database consisting of a group of transactions, frequent patterns are sets of items that are present in more than a given number of transactions as defined as follows.

Definition 1 (Frequent Patterns). *Let $I = \{i_1, i_2, \cdots, i_m\}$ be a set of items, a database $D = \{(tid_1, t_1), (tid_2, t_2), \cdots, (tid_n, t_n)\}$ is a set of transactions where tid_k is the identifier of transaction $t_k \subseteq I$. A pattern $p \subseteq I$ is a set of items*

Table 1. Transaction database D in different representations

(a) D as sets

tid	items
1	{a, b, c, d, f}
2	{a, b, c, e}
3	{b, f}
4	{a, b, c, d }
5	{a, c, e}

(b) D as bit vectors

tid	a	b	c	d	e	f
1	1	1	1	1	0	1
2	1	1	1	0	1	0
3	0	1	0	0	0	1
4	1	1	1	1	0	0
5	1	0	1	0	1	0

(c) $E(D)$: the encrypted

tid	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	$\tilde{d}$	$\tilde{e}$	$\tilde{f}$
1	$1'$	$1'$	$1'$	$1'$	$0'$	$1'$
2	$1'$	$1'$	$1'$	$0'$	$1'$	$0'$
3	$0'$	$1'$	$0'$	$0'$	$0'$	$1'$
4	$1'$	$1'$	$1'$	$1'$	$0'$	$0'$
5	$1'$	$0'$	$1'$	$0'$	$1'$	$0'$

from I, and p is contained by a transaction (tid, t) if $p \subseteq t$. The number of transactions in D containing p is the absolute support of p, which is denoted as $p.supp$. Any pattern is a frequent pattern if its support is equal to or greater than a user-defined minimum threshold, $minSup$.

Running Example: Given the database D in Table 1(a) and $minSup = 3$, all frequent patterns are {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, and {a,b,c}.

While some mining algorithms [4,14] represent transactions as sets of items as in Table 1(a), some other algorithms [1,18] employ the bit vector representation where there is a bit vector for each transaction with each bit indicating the corresponding item's presence in or absence from the transaction. For example, Table 1(b) is the bit vector representation of D in Table 1(a).

3.2 Secure Outsourced Mining by Fully Homomorphic Encryption

With our secured scenario, the database is represented by bit vectors [1,18], and encrypted and kept in the cloud in advance, from which various types of knowledge can be discovered by outsourcing the corresponding mining tasks.

The encrypted database, denoted as $E(D) := \mathbf{Encrypt}(pk, \mathbf{Mask}(D))$, is derived as follows. First, the **Mask** function is called to map the items in I to meaningless symbols in a set $\hat{I}$ and to rearrange rows and columns in the bit-matrix, i.e., the bit vector representation. Second, the bits in $\mathbf{Mask}(D)$ are encrypted by calling the **Encrypt** function provided by the fully homomorphic encryption scheme [7,9].

The scheme [7,9] supports any computation that can be expressed by a boolean circuit **C** that consists of AND-gates and XOR-gates and takes as input the plaintext bits. Concretely, this scheme is defined by four functions: a key generation function, **KeyGen**, an encryption function, **Encrypt**, a decryption function, **Decrypt**, and an evaluation function, **Evaluate**.

- $\mathbf{KeyGen}(\lambda)$ generates a secret key sk and a public key pk given a security parameter λ. The sk is an odd η-bit integer, and the pk is defined as (pk^*, y) where pk^* is a set of $\tau + 1$ random γ-bit integers with $pk_i^* = sk \cdot q_i + 2r_i$ and q_i and r_i randomly selected for $i \in \{0, \cdots, \tau\}$, and y is a vector of Θ-components with κ bits of precision after the binary point that will be used with the encrypted secret key to evaluate a circuit homomorphically.

- **Encrypt**$(pk, m_1, \cdots, m_s)$ encrypts any s plaintext bits by the public key pk. The ciphertext c_i of $m_i \in \{0, 1\}$ for $i \in \{1, \cdots, s\}$ is the sum of a subset of pk^* plus m_i and random noise $2r$.
- **Decrypt**$(sk, c_1, \cdots, c_t)$ decrypts any t ciphertexts by the secret key sk. Each plaintext bit m_i of c_i for $i \in \{1, \cdots, t\}$ is recovered as $(c_i \bmod sk) \bmod 2$.
- **Evaluate**$(pk, \mathbf{C}, c_1, \cdots, c_t)$ takes as input a public key pk, a boolean circuit $\mathbf{C}$, and t ciphertexts c_i, and correctly evaluates the circuit $\mathbf{C}$ on ciphertexts in that if $c_i =$ **Encrypt**(pk, m_i) for $i \in \{1, \cdots, t\}$, then **Decrypt**$(sk,$ **Evaluate**$(pk, \mathbf{C}, c_1, \cdots, c_t)) = \mathbf{C}(m_1, \cdots, m_t)$.

For the running example, Table 1(c) shows $E(D)$ where $I = \{$a, b, c, d, e, f$\}$, $\hat{I} = \{\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f}\}$, **Mask**(a) $= \hat{a}$, **Mask**$^{-1}(\hat{a}) =$ a, and so on. Each entry in Table 1(c) is the encryption of the corresponding bit in Table 1(b).

An important observation is that the encryption of any bit b contains random noise, and thus two invocation of **Encrypt**(pk, b) will yield different results. Therefore, no comparison is supported on encrypted bits, which has profound impact on our design of protocols.

3.3 A Variant of the Apriori Algorithm

The Apriori algorithm [4,14] is the most influential algorithm for mining frequent patterns, which represents a transaction as a set of items as in Table 1(a). The **BinaryApriori** algorithm in Algorithm 1 is a variant of Apriori for mining data in the bit vector representation [1,18] as in Table 1(b), which may be converted into a homomorphic counterpart that can mine the encrypted data.

Algorithm 1. **BinaryApriori**$(D, minSup)$

```
1: for each candidate c ∈ I do c.supp := countSupport(c, D);
2: L_1 := {c ∈ I | c.supp ≥ minSup};
3: for ( i := 1; L_i ≠ ∅; i++ ) do
4:   C_{i+1} := genCandidates(L_i);
5:   for each candidate c ∈ C_{i+1} do c.supp := countSupport(c, D);
6:   L_{i+1} := {c ∈ C_{i+1} | c.supp ≥ minSup};
7: end for
```

BinaryApriori computes the set L_1 of frequent patterns of length 1 by counting every item's support (at lines 1 to 2), and then finds frequent patterns of length $i + 1$ for $i \geq 1$ iteratively (at lines 3 to 7) as follows.

First, **BinaryApriori** generates the set C_{i+1} of candidates of length $i + 1$ by calling the **genCandidates** function (at line 4). This function employs the monotone property: the subsets of a frequent pattern must be frequent.

Second, **BinaryApriori** counts the support of every candidate $c \in C_{i+1}$ (at line 5), and identifies those candidates whose supports are no less than $minSup$ to get L_{i+1} (at line 6). The **countSupport** function computes the support of c in D by first performing bitwise AND-operation on the columns corresponding to the constituent items of c and then summing up the resulting bits.

For the running example, to count the support of {a, b}, **countSupport** first performs the bitwise AND-operation on columns a and b in Table 1(b), i.e. $(1,1,0,1,1)^T$ and $(1,1,1,1,0)^T$, to produce $(1,1,0,1,0)^T$, and then sums up the bits to get the result 3. In short, **countSupport** can be expressed by a boolean circuit of AND-gates and OR-gates.

4 Secured Protocol Based on FHE for Pattern Mining

It is nontrivial to protect the security and privacy of outsourced data and frequent pattern mining based on fully homomorphic encryption (FHE), since no comparison can be performed on encrypted data and hence the server, i.e., the cloud service provider, cannot run a frequent pattern mining algorithm on encrypted data without the help of the client. Therefore, we propose a protocol, namely **Sphene** - **S**ecured **p**rotocol by **h**omomorphic **en**um**e**ration, that avoids the comparison operation on the encrypted data at the server side.

In a preparation step of the Sphene protocol (Step 0 in Fig. 1), the client's encrypted database, $E(D)$, is outsourced to the server in advance, and the client keeps a secret key sk to himself and shares a public key pk with the server.

In interaction steps (Steps 1 - 4 in Fig. 1), the server enumerates patterns of a given length and counts the number of frequent patterns in the encrypted data homomorphically, and the client controls the iterations over the pattern length, which is detailed in the following.

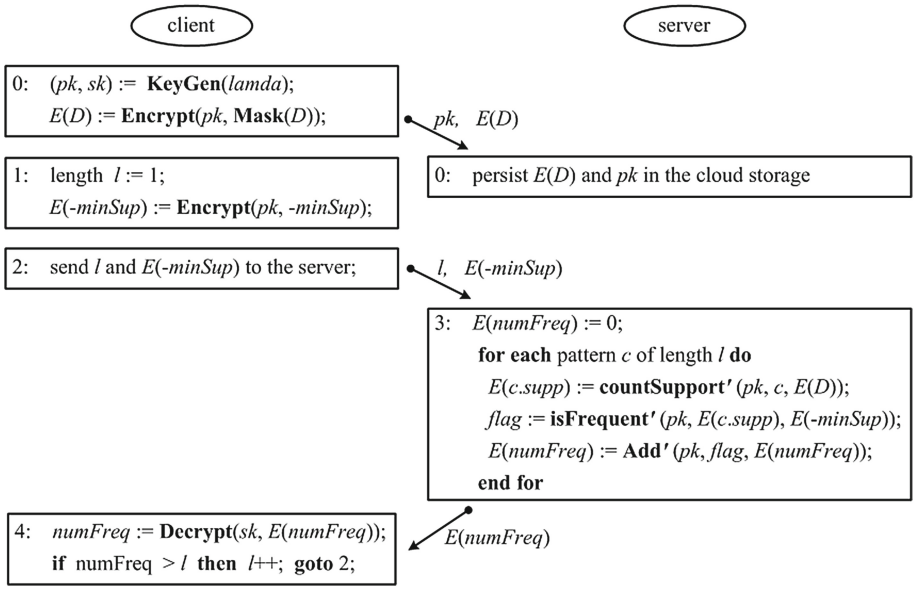

Fig. 1. Sphene - our first protocol for mining frequent patterns based on FHE

Step 1: The client encrypts the complement of $minSup$ in $\lceil log_2|D| \rceil + 2$ bits, denoted as $E(-minSup)$, and starts the iterations with length l of 1.

Step 2: The client sends a pattern length l to the server together with the encrypted complement of $minSup$, $E(-minSup)$, which will be used by the server to evaluate if a pattern is frequent homomorphically.

Step 3: The server enumerates every pattern c of length l, computes the encrypted support of c and an encrypted bit $flag$ that is an encryption of 1 if c is frequent or that of 0, and adds up such bits to get $E(numFreq)$, i.e., the encrypted number of frequent patterns of length l, by employing the following.

- $\mathbf{countSupport}'(pk, c, E(D))) = \mathbf{Evaluate}(pk, \mathbf{countSupport}, c, E(D))$ to compute the encrypted support of c homomorphically as in Sect. 3.
- $\mathbf{isFrequent}'(pk, E(c.supp), E(-minSup))$ to evaluate 1 XOR-ed with the most significant "bit" of $\mathbf{Add}'(pk, E(c.supp), E(-minSup))$.
- $\mathbf{Add}'(pk, \cdot, \cdot)$ to evaluate the sum of encrypted integers homomorphically.

Step 4: The client gets $numFreq$ by decrypting the encrypted number of frequent patterns sent back from the server. If $numFreq$ is greater than l, the client starts the next iteration, otherwise it stops since a frequent pattern of length $l + 1$ should have $l + 1$ frequent subsets of length l.

Theorem 1. *The Sphene protocol is correct, and is secured in the sense that no one except the client can know the minimum support threshold and the support of a pattern, and can infer if a pattern is frequent. An attacker can only infer that the length of a frequent pattern is no more the number of iterations.*

The advantage of this protocol is that the client and the server only need to exchange little information. However, it needs to count the supports of many unnecessary candidate patterns, and hence its efficiency might be low when running on datasets of many items.

5 Privacy Preserving Protocol for Pattern Mining

An alternative to the first protocol is to reduce the workload at the server side through close coordination between the two sides. That is to let the client lead the execution of the **BinaryApriori** algorithm, in particular, to let the client generate the candidates to be counted by the server. However, such an alternative may leak more information.

5.1 The Notion of α-Pattern Uncertainty

The privacy risk is that an attacker or the server may infer the total number of frequent patterns by making use of his knowledge about the **BinaryApriori** algorithm, that is, any subset of a candidate is frequent, although he cannot infer the minimum support threshold $minSup$, the support of any pattern, and whether a particular pattern is frequent as all items are masked.

We propose a new privacy notion, namely α-pattern uncertainty, to limit the server's certainty in inferring the total number of frequent patterns.

Definition 2 (α-pattern Uncertainty). *A set M of (masked) patterns is α-uncertain, if an attacker's certainty in inferring any pattern in M to be a true candidate pattern is no more than a privacy parameter α set by the client.*

For example, if the client sends the masked version of the true candidate pattern set, $\hat{C}_2 = \{\{\hat{a}, \hat{b}\}, \{\hat{a}, \hat{c}\}, \{\hat{b}, \hat{c}\}\}$ to the server, the server will infer that there are 3 frequent items although he does not know what $\hat{a}$, $\hat{b}$, and $\hat{c}$ stand for. But, if the client sends an expanded set $E\hat{C}_2 = \hat{C}_2 \cup \{\{\hat{d}, \hat{e}\}, \{\hat{d}, \hat{f}\}, \{\hat{e}, \hat{f}\}\}$, the attacker's certainty of inference may drop to 50 %.

Definition 3 (shadow mappings and shadow patterns). *The shadow mappings, denoted as $sMaps$, are $(1/\alpha - 1)$ one-to-one and onto functions, that have the same domain and disjoint ranges, from frequent items to infrequent items. For each pattern c, shadow patterns of c are generated by applying on c the functions from $sMaps$, denoted as* ***shadows****$(c, sMaps)$.*

To investigate the feasibility of applying FHE in outsourcing frequent patten mining, a simple technique, shadow mappings (patterns), is proposed, which however may be subject to further attacks. We leave as a future work an improved technique that prevents any further inference.

5.2 Privacy Preserving Protocol for Counting Candidates

Our second protocol, namely **P3CC** - **P**rivacy **P**reserving **P**rotocol for **C**ounting **C**andidates homomorphically in outsourcing frequent pattern mining as shown in Fig. 2, employs both α-pattern uncertainty and fully homomorphic encryption as privacy and security measures.

The preparation step of the P3CC protocol (Step 0 in Fig. 2) is the same as the Sphene protocol. In the interaction steps, the client generates the set of candidate patterns and the set of shadow patterns, and sends the masked version EC of their union to the server. The server homomorphically computes the support of each pattern in EC, and sends the result to the client. The client recovers and decrypts the result and gets the frequent patterns. The interaction is further explained with the running example in the following.

Step 1: The client sends the masked candidate set $EC = \mathbf{Mask}(C_1)) = \mathbf{Mask}(I)) = \{\{\hat{a}\}, \{\hat{b}\}, \{\hat{c}\}, \{\hat{d}\}, \{\hat{e}\}, \{\hat{f}\}\}$ to the server.

Step 2: The server counts the support of each masked candidate pattern homomorphically, and sends the result back to client in the encrypted form.

Step 3: The client recovers the candidate set from the masked one, decrypts the support of each candidate in $C_1 = EC'$ and determines the set of frequent patterns of length 1, $L_1 = \{\{\text{a}\}, \{\text{b}\}, \{\text{c}\}\}$. Suppose $\alpha = 50\,\%$, then only one function is in the shadow mappings, e.g., $sMaps = \{\{(\text{a}, \text{d}), (\text{b}, \text{e}), (\text{c}, \text{f})\}\}$.

Step 4: The client gets the candidate set $C_2 = \{\{\text{a}, \text{b}\}, \{\text{a}, \text{c}\}, \{\text{b}, \text{c}\}\}$, and sends the masked and expanded candidate set $EC_2 = \{\{\hat{a}, \hat{b}\}, \{\hat{a}, \hat{c}\}, \{\hat{b}, \hat{c}\}, \{\hat{d}, \hat{e}\}, \{\hat{d}, \hat{f}\}, \{\hat{e}, \hat{f}\}\}$ to the server.

Step 5: The server counts the supports and sends the encrypted result back.

Step 6: The client recovers and decrypts the support of each pattern in EC_2 and gets $L_2 = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. As the size of L_2 is greater than 2, the protocol enters a new round by jumping to Step 4, and so on.

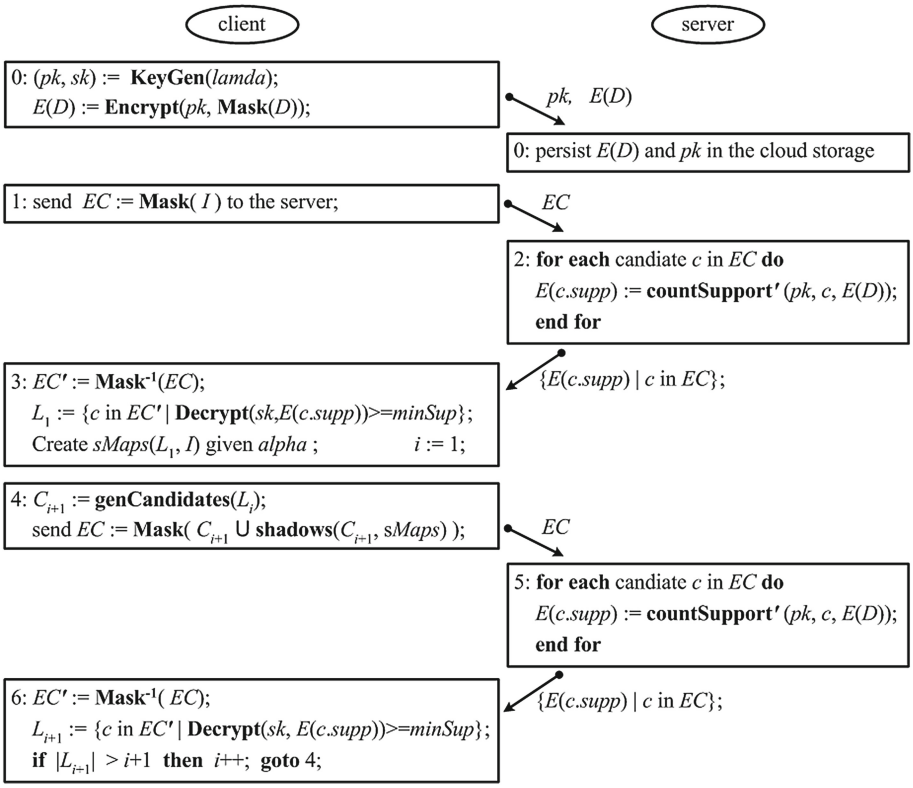

Fig. 2. P3CC - our second protocol for mining frequent patterns

Theorem 2. *The P3CC protocol is correct, and is secured in the sense that no one except the client can know the minimum support threshold and the support of a pattern, and can infer if a pattern is frequent. An attacker can only infer that the length of a frequent pattern is no more the number of iterations, and infer the total number of frequent patterns with a certainty no more than* α.

6 Experimental Evaluation

This section evaluates our protocols experimentally. To the best of our knowledge, there is no prior work on FHE-based frequent pattern mining, and the related works [5,6,8,12,13,15,17,19–22] consider scenarios that are not comparable to ours. We can only evaluate our protocols, Sphene and P3CC.

Two datasets are used in experiments. The first is the artificial dataset, T10I6N50D5kL1k, generated by the generator from the IBM Almaden Quest research group[1]. The second is the Chess dataset from FIMI[2].

The protocols were implemented upon a general-purposed FHE library that supports the scheme [7] and uses the GNU multiple precision arithmetic library[3]. The implementation renders parallelism by multi-threading. The experiments were performed on an HP Pavilion dm4 laptop running Ubuntu 12.04.4.

We first evaluate the Sphene and P3CC protocols on the T10I6N50D5kL1k and chess datasets with varying the minimum support threshold, *minSup*. For T10I6N50D5kL1k as shown in Fig. 3(a), P3CC is up to 3 orders of magnitude more efficient than Sphene. For Chess as shown in Fig. 3(b), P3CC is up to 5 orders of magnitude more efficient than Sphene.

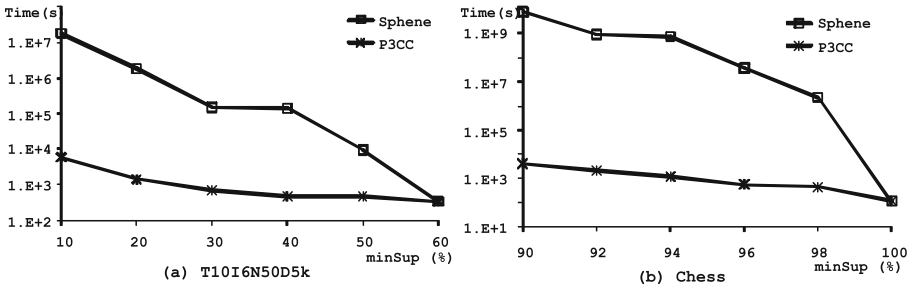

Fig. 3. Running time of two protocols with varying *minSup* on two datasets

We then evaluate the protocols with varying data features and privacy requirements. First, Fig. 4(a) shows the result for the artificial dataset with the number (D) of transactions ranging from 1,000 (D1k) to 10,000 (D10k), which depicts that both P3CC and Sphene scale almost linearly in database size.

Second, Fig. 4(b) shows the result with the number (N) of distinct items ranging from 10 (N10) to 100 (N100). While the performance of P3CC is not highly correlated to this feature, the performance of Sphene fluctuates depending on which of the two opposite effects with the increase of the number of items will dominate: one is that the number of candidates in a protocol iteration increases, the other effect is the the number of iterations decreases (as data get sparser).

Third, Fig. 4(c) shows the result with the number (L) of possible frequent patterns ranging from 1,000 (L1k) to 6,000 (L6k), which indicates this feature has little impact on the performance of P3CC and Sphene.

Finally, Fig. 4(d) shows the result with the varying privacy parameter α with $k = 1/\alpha$ ranging from 1 to 6. While Sphene does not employ the privacy preserving measure, P3CC performs linearly in the privacy parameter.

[1] http://miles.cnuce.cnr.it/~palmeri/datam/DCI/datasets.php/.
[2] http://fimi.cs.helsinki.fi/data/.
[3] https://gmplib.org/.

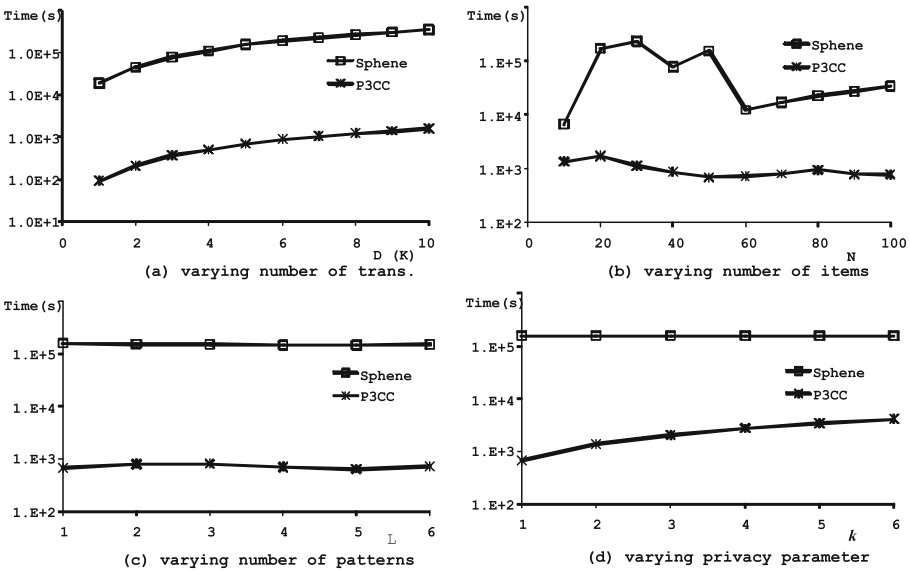

Fig. 4. Evaluation with varying data features on the artificial dataset

In summary, the experimental evaluation demonstrates that while the Sphene protocol may be not efficient enough for real applications, the P3CC protocol provides a solution for outsourcing both data storage and frequent pattern mining tasks, although P3CC has higher communication cost than Sphene.

7 Conclusions and Future Work

This paper proposes two secured protocols for mining frequent patterns on the cloud, which secures the data and the mining process by employing fully homomorphic encryption (FHE), and addresses privacy issues by introducing the α-pattern uncertainty and reinforcing the second protocol by shadow mappings. Preliminary experimental evaluation shows the feasibility of the protocols.

The future work is to address the limitations of this preliminary work, which includes: (1) new techniques other than shadow mappings that prevent further inference attacks when observing the α-pattern uncertainty; (2) handling possible attacks on the α-pattern uncertainty and on FHE by formal analysis; (3) distributed mining approaches based on the MapReduce and Spark framework to improve the efficiency; (4) new homomorphic encryption schemes customized for frequent pattern mining with lower time complexity.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China (61272306), and the Zhejiang Provincial Natural Science Foundation of China (LY12F02024).

References

1. Agarwal, R., Aggarwal, C., Prasad, V.V.V.: Depth first generation of long patterns. In: 6th SIGKDD, pp. 108–118 (2000)
2. Aggarwal, C.C., Yu, P.S.: Privacy-preserving Data Mining: Models and Algorithms. Springer-Verlag, Boston (2008)
3. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: 1993 SIGMOD, pp. 207–216 (1993)
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Research Report RJ 9839. IBM Almaden Research Center, San Jose, CA (1994)
5. Atzori, M., Bonchi, F., Giannotti, F., Pedreschi, D.: Anonymity preserving pattern discovery. VLDB J. **17**(4), 703–727 (2008)
6. Chu, Y.-W., Tai, C.-H., Chen, M.-S., Yu, P.S.: Privacy-preserving simrank over distributed information network. In: 12th ICDM, pp. 840–845 (2012)
7. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
8. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: 22nd PODS, pp. 211–222 (2003)
9. Gentry, C.: Fully Homomorphic encryption using ideal lattices. In: 41st ACM Symposium on Theory of Computing, pp. 169–178 (2009)
10. Gellman, R.: Privacy in the clouds: risks to privacy and confidentiality from cloud computing. In: World Privacy Forum, 23 February (2009)
11. Goethals, B.: Survey on frequent pattern mining. Technical report, University of Helsinki (2003)
12. Giannotti, F., Lakshmanan, L.V.S., Monreale, A., Pedreschi, D., Wang, H.: Privacy-preserving mining of association rules from outsourced transaction databases. IEEE Syst. J. **7**(3), 385–395 (2012)
13. Hu, H., Xu, J., Ren, C., Choi, B.: Processing private queries over untrusted data cloud through privacy homomorphism. In: 27th ICDE, pp. 601–612 (2011)
14. Mannila, H., Toivonen, H., Verkamo, A.I.: Efficient algorithms for discovering association rules. In: AAAI Workshop Knowledge Discovery in Databases, KDD 1994, pp. 181–192 (1994)
15. Qiu, L., Li, Y., Wu, X.: Protecting business intelligence and customer privacy while outsourcing data mining tasks. Knowl. Inf. Syst. **17**(1), 99–120 (2008)
16. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: DeMillo, R.A., et al. (eds.) Foundations of Secure Computation, pp. 169–179. Academic Press, New York (1978)
17. Rizvi, S., Haritsa, J.R.: Maintaining data privacy in association rule mining. In: 28th VLDB, pp. 682–693 (2002)
18. Shenoy, P., Haritsa, J.R., Sudarshan, S., Bhalotia, G., Bawa, M., Shah, D.: Turbocharging vertical mining of large databases. In: 2000 SIGMOD, pp. 22–33 (2000)
19. Tai, C. H., Yu, P.S., Chen, M.S.: k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In: 16th SIGKDD, pp. 473–482 (2010)
20. Verykios, V.S., Elmagarmid, A.K., Bertino, E., Saygin, Y., Dasseni, E.: Association rule hiding. IEEE Trans. Knowl. Data Eng. **16**(4), 434–447 (2004)
21. Wang, Y., Wu, X.: Approximate inverse frequent itemset mining: privacy, complexity, approximation. In: 5th ICDM, pp. 482–489 (2005)
22. Wong, W.K., Cheung, D.W., Hung, E., Kao, B., Mamoulis, N.: Security in outsourcing of association rule mining. In: 33rd VLDB, pp. 111–122 (2007)