World Wide Web (2021) 24:607–629
https://doi.org/10.1007/s11280-021-00863-w

Efficient privacy-preserving frequent itemset query over semantically secure encrypted cloud database

Wei Wu[1] · Ming Xian[2] · Udaya Parampalli[3] · Bin Lu[1]

Received: 22 September 2020 / Revised: 26 December 2020 / Accepted: 11 January 2021 /
Published online: 2 February 2021

Abstract

In recent years, more users tend to use data mining as a service (DMaaS) provided by cloud service providers. However, while enjoying the convenient pay-per-use mode and powerful capacity of cloud computing, users are also threatened by the potential risk of privacy leakage. In this paper, we aim to efficiently perform privacy-preserving DMaaS, and focus on frequent itemset mining over encrypted database in outsourced cloud environment. Existing work apply different encryption methods to design various privacy-preserving mining solutions. Nevertheless, these approaches either cannot provide sufficient security requirements, or introduce heavy computation costs. Some of them also need users staying on-line to execute computations, which are not practical in real-world applications. In this paper, we propose a novel efficient privacy-preserving frequent itemset query (PPFIQ) scheme using two homomorphic encryptions and ciphertext packing technique. The proposed scheme protects transaction database with semantic security, preserves mining privacy and resists frequency analysis attacks. Meanwhile, efficiency is guaranteed by inherent parallel computations for packed plaintexts and users could stay off-line during the mining process. We provide formal security analysis and evaluate the performance of our scheme with extensive experiments. The experiment results demonstrate that the proposed scheme can be efficiently implemented on large databases.

Keywords Privacy-preserving · Frequent itemset mining · Cloud computing · Fully homomorphic encryption · Ciphertext packing

1 Introduction

Recently, with the increasing popularity of cloud computing, data mining as a service (DMaaS) becomes attractive for its flexibility and cost-saving properties. Using DMaaS, users can outsource their data and further data mining operations to cloud servers in a pay-as-you-go mode. However, for users from domains such as medical institutions, governmental organizations and financial establishments, both their raw data and the mining

✉ Wei Wu
goodwuwei18@163.com

Extended author information available on the last page of the article.

results may contain sensitive information [27, 39, 42, 47]. Concerning the leakage of privacy [18, 24, 40, 41, 44, 45], a user might be reluctant or forbidden to use DMaaS in cloud environment.

In the data mining field [10, 16, 19, 25, 33], frequent itemset mining (FIM) is a widely used method in many applications [5, 6, 8], which are utilized to find hidden frequent itemsets in large databases. Specifically, for a given transaction database, FIM aims to find itemsets included by the transactions which have frequencies larger than a threshold value. To preserve privacy during the outsourced frequent itemset mining processes, some perturbation-based solutions [11, 26, 30, 52] have been proposed. Unfortunately, these approaches can only provide limited security protection and greatly reduce the accuracy of mining results. To solve these problems, later work [13, 23, 37, 48] apply substitution ciphers to encrypt data before uploading them to cloud servers. Nevertheless, since substitution ciphers are not semantically secure, all these schemes cannot resist chosen plaintext attacks, making them vulnerable against strong adversaries. To achieve better security, Lai et al. [21] propose the first semantically secure solution performing outsourced FIM. However, this solution is insecure against frequency analysis attacks (FAA) and introduces heavy computation cost. Later works [17, 28, 34, 51] utilize several homomorphic encryption schemes to design various mining solutions with different privacy levels. However, none of the existing solutions could efficiently perform frequent itemset mining on semantically secure encrypted data without on-line users. For better comparison, we present some existing work [17, 21, 23, 28, 34, 51] for outsourced FIM in Table 1.

As shown in the table, we list five required properties for a privacy-preserving frequent itemset mining scheme. The first one, *D. Semantic Security* denotes the encrypted transaction database has semantic security, which is essential for data sharing and outsourcing. For the second property *Mining Privacy*, we define the preservation of the following private information: (1) frequent itemset, (2) support of itemset and (3) threshold value (i.e., minimum support). The next property *Resist FAA* indicates that a scheme is able to resist frequency analysis attacks. The fourth one *Off-line Users* is a practical property, which means users do not need to stay on-line during the mining process. The last one *Computation Cost* shows the efficiency of the mining process, implying whether a scheme can be efficiently applied on large databases.

In this paper, focusing on the aforementioned five properties, we propose a novel efficient privacy-preserving frequent itemset query (PPFIQ) scheme. We consider that multiple data

Table 1 Comparison of our work and some existing works

	D. Semantic security	Mining privacy	Resist FAA	Off-line users	Computation cost
Li et al [23]	No	No	Yes	No	Low
Lai et al [21]	Yes	No	No	Yes	High
Yi et al [51]*	No	Yes	Yes	Yes	High
Imabayashi et al [17]	Yes	Yes	Yes	No	Low
Qiu et al [34]*	Yes	Yes	Yes	Yes	High
Liu et al [28]	Yes	Yes	Yes	Yes	High
Ours	Yes	Yes	Yes	Yes	Low

*Three solutions with different privacy levels are proposed in [51] and [34], we only show the one with the highest level here owners (DOs) outsource their encrypted transaction databases and further mining tasks to cloud servers. Then, to check the frequency, a query user (QU) sends an encrypted itemset query to cloud servers for frequent itemset mining based on the encrypted cloud databases.

Example Suppose several retail corporations (act as DOs) want to share their transaction databases for better frequent itemset mining results. However, they cannot disclose the plain transactions which contain precious value and customer privacy. Therefore, they encrypt the private databases and outsource them as well as relevant data mining tasks to cloud servers. When one of these corporations, or another retailer (acts as a QU) wants to determine the frequency of its private itemset, it also performs encryption and sends the ciphertext of query itemset to cloud servers. In this way, cloud servers cloud securely execute frequent itemset mining for the query itemset based on the aggregated encrypted transaction databases.

We emphasize that after outsourcing, neither DOs nor QU need to stay on-line, and all computations are efficiently executed by cloud servers. To the best of our knowledge, our work is the first approach efficiently achieving frequent itemset query on semantically secure encrypted data without on-line DOs and QU. The main contributions of the proposed scheme are summarized as follows.

- We propose a new privacy-preserving frequent itemset query (PPFIQ) scheme over encrypted cloud database using two homomorphic encryptions YASHE (Yet Another Somewhat Homomorphic Encryption) [3] and Paillier [32]. The proposed scheme protects transaction databases with semantic security, preserves mining privacy and resists frequency analysis attacks.
- The proposed scheme is highly efficient by using the ciphertext packing technique, which achieves parallel computation without extra costs. Besides, both the data owners and query user do not need to participate during the mining process, who can stay off-line after outsourcing the databases and query itemset.
- We provide formal security analysis for the proposed scheme under semi-honest model. We also evaluate the performance with extensive experiments on both real and synthetic databases. The experiment results demonstrate that our PPFIQ scheme can securely and efficiently perform frequent itemset mining over encrypted large databases.

The rest of the paper is organized as follows. The related works are discussed in Section 2. We provide the system architecture, threat model, background knowledge and notations in Section 3. In Section 4, several new secure primitives are presented, which are used as sub-routines in our scheme. In Section 5, we propose the novel PPFIQ scheme and provide formal security analysis. The experiment results are given in Section 6. Lastly, we conclude the paper along with the future work in Section 7.

2 Related work

The problem of privacy-preserving frequent itemset mining is first addressed by Vaidya and Clifton [38] and Kantarcioglu and Clifton [20], which considers the circumstances of vertically and horizontally partitioned databases respectively. However, both of the two solutions cannot be applied on large databases for their low efficiency. Later, a large number of perturbation-based approaches [11, 26, 30, 52] have been proposed to protect private information during the outsourced frequent itemset mining (association rule mining) processes.

Nevertheless, these methods consider the confidentiality of data records and ignore the privacy of mining results. Moreover, they have low security levels and return approximate results with poor mining precision.

To achieve better privacy preservation, other works introduce substitution ciphers for data encryption [13, 23, 37, 48]. Wong et al. [48] present a scheme encrypting transactions using a mapping function and insert random dummy items into the encrypted transactions. Unfortunately, Molloy et al. [31] demonstrate that it is easy to remove the fake items by computing the low correlations between items, leading to re-identifications for the top frequent items. Tai et al. [37] propose k-support anonymity in which the support of each item is similar with at least $k-1$ other items. Giannotti et al. [13] use k-privacy approach requiring that each itemset is distinguishable from at least $k-1$ other itemsets which have the same sizes. Li et al. [23] propose an efficient solution by applying a new designed symmetric homomorphic encryption scheme. In their solution, data owners stay on-line and cooperate with a cloud to the execute frequent itemset mining. However, Wang et al. [43] present an attack method for the homomorphic encryption scheme in [23], showing an adversary can obtain the secret key based on some known plaintext-ciphertext pairs. Li et al. [22] proposes a database reconstruction-based algorithm which preserves data privacy with reasonable data utility. They combine an IFIM based on FP-tree and an extension-based method to reconstruct a new database, in which the same frequent itemsets could be mined according to a given threshold. Although these works [13, 22, 23, 37, 48] achieve privacy preservation, all of them cannot provide semantic security, making them vulnerable against chosen plaintext attacks.

To provide semantic security, by leveraging predicate encryption and dual system encryption methodology, Lai et al. [21] first propose a semantically secure scheme for outsourced association rule mining. Nevertheless, this scheme cannot resist frequency analysis attacks since they use fixed ciphertext for each 1-itemset. Later, Yi et al. [51] present three solutions with different privacy levels using distributed ElGamal cryptosystem. Nevertheless, their work requires more than two cloud servers to collaboratively perform the mining processes on encrypted data, therefore introducing large amounts of communication overhead. Imabayashi et al. [17] propose a secure frequent itemset mining approach using fully homomorphic encryption with ciphertext packing technique. However, this work also requires DO on-line to perform decryption and comparison, which brings extra costs for DO. By using two homomorphic encryptions Paillier [32] and BGN [2], Qiu et al. [34] design three secure frequent itemset mining protocols for a given query itemset with different privacy requirements. Nevertheless, utilizing these public-key encryptions introduces high computation costs and results in low efficiency. Recently, Liu et al. [28] also propose a scheme for frequent itemset query considering the situation that data owners and query users have different encryption keys. Their solution is based on the BCP cryptosystem [4], which is additive homomorphic and allows double decryption mechanism. However, in the mining process, the master key (a secret key can decrypt any given ciphertext) is shared with all query users, which greatly increases the risk of key leakage. Ma et al. [29] proposes a protocol to perform encrypted frequent itemset mining query on supermarket transactions. They divide the encrypted transactions into different blocks and only conduct bilinear pairings on part of the ciphertext blocks, which partly reduces the mining cost. Although these works [21, 28, 29, 34, 51] achieve better security, they are all very inefficient because of the high computation costs introduced by the underlying encryption methods, making them impractical to be applied on large databases.

3 Problem statement

In this section, we introduce the system architecture and threat model, provide the background knowledge and notations used in this paper.

3.1 System architecture

The system architecture of our scheme is illustrated in Figure 1, which supports the frequent itemset query for a given itemset over encrypted cloud databases. As shown in the figure, we have multiple data owners DO_l $(1 \leq l \leq L)$, a query user (QU) and two cloud servers CSP (cloud service provider) and Evaluator in our system model. Here, Evaluator generates the public and secret key pairs (pk_Y, sk_Y) and (pk_P, sk_P) for the YASHE and Paillier encryptions respectively. To enable encryptions, the public keys are shared with DOs, QU and CSP, while the secret keys are kept private by Evaluator. The data owners have private horizontally or vertically partitioned transaction databases D_l, which are encrypted as $[D_l]$ using YASHE and outsourced to CSP for frequent itemset mining. The QU owns a private itemset x and wants to identify whether x is frequent or not based on the combined encrypted database $[D]$. To preserve privacy, the QU also encrypts x into $[x]$ using YASHE and sends it to CSP as a query. The CSP has the aggregated encrypted database $[D]$ and cooperates with Evaluator to perform the mining process. To emphasize, we do not share the secret keys with DOs and QU to reduce the risk of key leakage. Therefore, each DO should keep

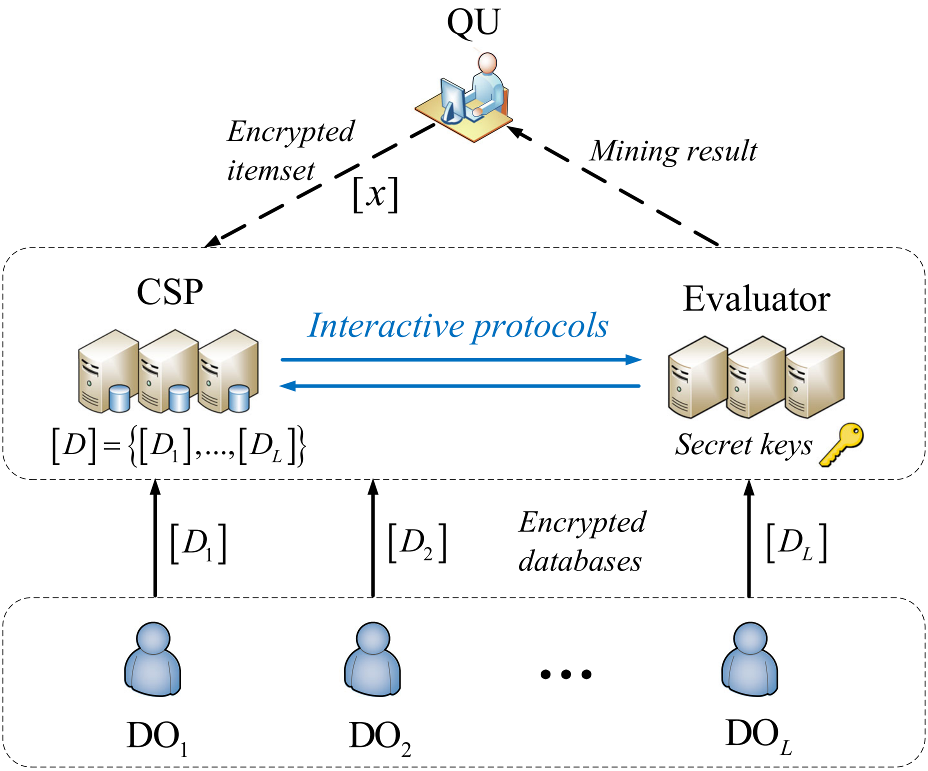


Figure 1 System architecture

a copy of its plain transaction database instead of retrieving it from CSP. This is acceptable since the transactions might have great value, and we mainly focus on the outsourced mining process.

3.2 Threat model

In our model, DOs and QU do not fully trust CSP for the privacy of transaction databases and query itemset. Therefore, the transaction databases and the query itemset as well as the threshold of minimum support are encrypted before uploaded to CSP. We assume that both CSP and Evaluator are semi-honest, which means they will strictly preform the designed protocols but try to figure out private information during the mining processes. Besides, we assume that CSP and Evaluator cannot collude with each other in our scheme. This non-colluding model between two clouds has been widely applied in many previous works [35, 46, 49] to support interactive protocols over encrypted data. We also assume that the two clouds cannot corrupt DOs or QU to obtain extra information. To define the security of a secure protocol, we adopt the simulation-based approach and provide the description of security under the semi-honest model in the following definition [14, 15].

Definition 1 Assume $\Pi_P(\pi)$ is the execution image of a protocol π for a party P, with input a and output b. Then, π is secure if $\Pi_P(\pi)$ can be simulated by a and b satisfying that the distribution of the simulated image $\Pi_P^S(\pi)$ is computationally indistinguishable from that of $\Pi_P(\pi)$.

As shown in the definition, an execution image usually consists of the input, the output and the information exchanged during the interactive processes. To prove the security of a protocol under the semi-honest model, we generally need to demonstrate no private information is disclosed from the execution image respecting the inputs of participating parties [15].

3.3 Background knowledge

In this section, we provide the background knowledge used in our PPFIQ scheme.

- **Frequent itemset mining (FIM)** As one of the most important data mining methods, frequent itemset mining (FIM) is used to discover relations of different items in large transaction databases, which was first proposed by Agrawal et al. [1]. Specifically, let $Q = \{q_1, \ldots, q_m\}$ be the set of all items q_j, and $D = \{t_1, \ldots, t_n\}$ be the database containing n transactions $t_i \subseteq Q$. Then, the aim of FIM is to find out itemsets with frequency (i.e., support) larger than a threshold value. Here, the support of an itemset x ($x \subseteq Q$) is defined as S_x, which is the number of transactions that include x in the given database D. An itemset x is frequent if $S_x \geq ms$ where ms is the threshold of minimum support required in FIM.

In this paper, we present the transactions t_i and itemset x as binary vectors $t_i = (t_{i1}, \ldots, t_{im})$ and $x = (x_1, \ldots, x_m)$, in which $t_{ij} = 1$ (resp., $x_j = 1$) indicates t_i (resp., x) contains the jth item and $t_{ij} = 0$ (resp., $x_j = 0$) otherwise. An example of transaction database is given in Table 2. Assume an itemset $x = \{q_3, q_5\}$ (the corresponding binary vector is $x = (0, 0, 1, 0, 1, 0)$) which has an itemset length $k = 2$ (the number of items in x, i.e., the number of ones in the binary vector). Then, as given in Table 2, the transactions

Table 2 Transaction database ($n = 5, m = 6$)

Transaction	Itemset	Binary vector
t_1	$\{q_1, q_2, q_3, q_5\}$	(1, 1, 1, 0, 1, 0)
t_2	$\{q_2, q_5, q_6\}$	(0, 1, 0, 0, 1, 1)
t_3	$\{q_3, q_4\}$	(0, 0, 1, 1, 0, 0)
t_4	$\{q_1, q_3, q_6\}$	(1, 0, 1, 0, 0, 1)
t_5	$\{q_3, q_4, q_5\}$	(0, 0, 1, 1, 1, 0)

t_1 and t_5 contain x, therefore the support $S_x = 2$. Considering the binary vectors in the example, if t_i $(1 \le i \le 5)$ contains x, then we have $t_i \circ x = k$, or we get $t_i \circ x < k$ otherwise ($\circ$ denotes the inner product between two binary vectors). We will use this property to determine the inclusion of an itemset by a transaction in our proposed scheme.

As discussed in Section 3.1, our scheme supports the circumstances of multiple data owners with horizontally or vertically partitioned databases. For horizontally partitioned databases, each DO has some transactions with all attributes in the binary vectors. For vertically partitioned databases, each DO has all transactions with partial attributes in the binary vectors. Using the example database in Table 2, we horizontally and vertically partition it with three data owners, which are shown in Figure 2. In the figure, when horizontally partitioned, DO_1 owns t_1 and t_2, DO_2 owns t_3 and t_4, while DO_3 has the last transaction t_5. For the situation of vertically partitioned databases, each DO owns two unique attributes of all the five transactions.

- **YASHE encryption** In this work, we use the fully homomorphic encryption (FHE) YASHE (semantically secure) [3] to encrypt the transactions and query itemset. Compared with the first FHE work by Gentry [12], YASHE is extremely efficient on the evaluation of arithmetic circuit with fixed multiplicative depth. Moreover, using the Single Instruction Multiple Data (SIMD) operations [36], YASHE supports ciphertext packing technique which packs M plaintexts into one ciphertext. Here, M denotes the

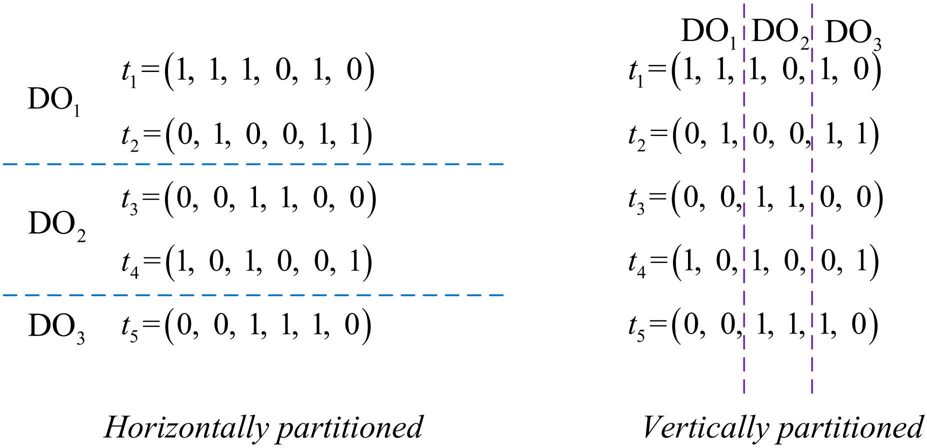


Figure 2 Partitioned databases

degree of polynomial modulus of YASHE (e.g., $M = 4096$). In this way, parallel computations are executed inherently without extra costs, which significantly improves the efficiency. When utilizing the packing technique, we need to choose an appropriate plaintext modulus P as a prime congruent to 1 modulo $2M$. As a result, the plaintext space of YASHE is $\mathbb{Z}_P$, indicating all underlying plaintexts are values modulo P. Besides, YASHE also supports directly arithmetic operations between plaintext and ciphertext, which are more efficient when we add additive and multiplicative noises to certain ciphertexts. To better understand other properties and practical considerations of YASHE, we recommend readers to references [3, 7, 9, 50] for details.

- **Paillier encryption** In this paper, we use the additive homomorphic Paillier encryption (semantically secure) [32] to achieve secure comparison of Paillier ciphertexts. Given two Paillier ciphertexts $\langle m_1 \rangle$ and $\langle m_2 \rangle$ of plaintexts $m_1, m_2 \in \mathbb{Z}_N$ (N is a product of two large primes and $\mathbb{Z}_N$ is the plaintext space), Paillier has the following properties:(1) $\langle m_1 + m_2 \rangle = \langle m_1 \rangle \cdot \langle m_2 \rangle \ mod\ N^2$ and (2) $\langle \alpha \cdot m_1 \rangle = \langle m_1 \rangle^{\alpha} \ mod\ N^2$, in which α is a plaintext from $\mathbb{Z}_N$. In this paper, we use these properties to design protocols. More details of Paillier can be found in [32].

3.4 Notations

The notations used in this paper are given in Table 3.

As shown in the table, n and m are the numbers of transactions and items in a database, and k is the itemset length. For YASHE encryption, M is the degree of polynomial modulus and P is the plaintext modulus, which is a prime congruent to 1 modulo $2M$. We use $\{\cdot\}_M$ to denote a pack of M plain values for the ciphertext packing technique. The plaintext spaces of YASHE and Paillier are denoted as $\mathbb{Z}_P$ and $\mathbb{Z}_N$ respectively. Since we introduce two underlying encryptions, we use $E_Y(\cdot)/D_Y(\cdot)$ and $E_P(\cdot)/D_P(\cdot)$ to represent encryption and decryption operations for YASHE and Paillier separately. Similarly, $[\cdot]$ is used to denote the YASHE ciphertext with M-packed plaintexts, and $\langle \cdot \rangle$ denotes the Paillier ciphertext. For the operations of YASHE, we use $\oplus$, $\ominus$ and $\otimes$ to denote addition, subtraction and multiplication operations between ciphertexts, while use $+$, $-$ and $\times$ to represent arithmetic operations between plaintext and ciphertext. We simply use $\cdot$ to denote multiplication of Paillier ciphertexts.

Table 3 Notations

n	The number of transactions in a database
m	The number of items in a database
k	The length of an itemset
M	Degree of polynomial modulus of YASHE
P	Plaintext modulus of YASHE
$\{\cdot\}_M$	A pack of M plain values
$\mathbb{Z}_P, \mathbb{Z}_N$	Plaintext spaces of YASHE and Paillier
$E_Y(\cdot)$, $D_Y(\cdot)$	Encryption and decryption of YASHE
$E_P(\cdot)$, $D_P(\cdot)$	Encryption and decryption of Paillier
$[\cdot]$, $\langle \cdot \rangle$	Ciphertexts of YASHE and Paillier
$\oplus, \ominus, \otimes$	Add, Sub and Mul on ciphertexts of YASHE
$+, -, \times$	Plain-Cipher arithmetic operations of YASHE
$\cdot$	Multiplication of Paillier ciphertexts

4 Secure primitives

In this section, we present a set of new secure primitives which will be used as sub-routines in the proposed PPFIQ scheme.

4.1 Transaction database encryption (TDE)

In this protocol, we consider a data owner (DO) with a private transaction database D as input, and outputs the encrypted database $\left[D'\right]$ using the YASHE encryption with the ciphertext packing technique. After encryption, $\left[D'\right]$ will be outsourced to CSP for frequent itemset mining. Here, we assume that D contains n transactions and m items (attributes). A transaction $t_i = (t_{i1}, \ldots, t_{im})$ is represented as a binary vector, in which $t_{ij} = 1$ indicates t_i contains the jth item and $t_{ij} = 0$ otherwise. The proposed transaction database encryption (TDE) protocol is given in Algorithm 1.

Algorithm 1 TDE(D) → $\left\{\left[D'\right], \left[S^1\right], \left[S^0\right]\right\}$.

```
Performed by: DO
Generate $f$ all-zeros transactions and insert them into $D$
Randomly permute $D' = \left\{t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+f}\right\}$
Get masks $S^1 = \left(s_1^1, \ldots, s_{n+f}^1\right)$ and $S^0 = \left(s_1^0, \ldots, s_{n+f}^0\right)$
$z \leftarrow (n + f)/M$
for $a = 1$ to $z$
    $i \leftarrow (a - 1)M + 1$
    for $j = 1$ to $m$
        $Pack \leftarrow \left\{t_{ij}, t_{i+1,j}, \ldots, t_{i+M-1,j}\right\}_M$
        $\left[D'_{aj}\right] \leftarrow E_Y(Pack)$
    $Pack \leftarrow \left\{s_i^1, s_{i+1}^1, \ldots, s_{i+M-1}^1\right\}_M$
    $\left[S_a^1\right] \leftarrow E_Y(Pack)$
    $Pack \leftarrow \left\{s_i^0, s_{i+1}^0, \ldots, s_{i+M-1}^0\right\}_M$
    $\left[S_a^0\right] \leftarrow E_Y(Pack)$
$\left[D'\right] = \{\left[D'_{11}\right], \ldots, \left[D'_{1m}\right], \ldots, \left[D'_{z1}\right], \ldots, \left[D'_{zm}\right]\}$
$\left[S^1\right] = \{\left[S_1^1\right], \ldots, \left[S_z^1\right]\}$, $\left[S^0\right] = \{\left[S_1^0\right], \ldots, \left[S_z^0\right]\}$
Send $\left[D'\right]$, $\left[S^1\right]$ and $\left[S^0\right]$ to CSP
```

As shown in the algorithm, except for the encrypted database $\left[D'\right]$, we also output two encrypted "masks" (vectors) $\left[S^1\right]$ and $\left[S^0\right]$, which will be used in the SFIM protocol (see Section 4.3) to insert fake transactions and remove dummy support counts. To start, DO generates f all-zeros transactions with all entries set to zeros. Then, DO adds them into D and randomly permutes all the transactions to obtain the new database D'. Here, these all-zeros vectors are "places" to insert random fake transactions and the details are provided in Algorithm 3. Next, DO computes two $(n + f)$-length binary vectors S^1 and S^0, which will be used as masks in the process of frequent itemset mining. In mask S^1, the entries s_i^1 corresponding to the all-zeros transactions t_i in D' are set to ones and the other entries are set to zeros for $1 \leq i \leq n + f$. The values of entries in mask S^0 are opposite with s_i^0 set to be zeros for the all-zeros transactions while the remaining entries are set to ones.

Now, DO can perform encryption for the database D' and the masks S^1, S^0 using YASHE with ciphertext packing technique in Step 4-13. First, DO calculates the number of data blocks $z = (n + f)/M$, in which M is the degree of polynomial modulus of YASHE (e.g., $M = 4096$). To correctly pack M plain values into one YASHE ciphertext, we require that the size of D' (i.e., $n + f$) is divisible by M. Then, for each data block ($1 \le a \le z$) and each attribute ($1 \le j \le m$), DO puts the same attributes of M transactions into one pack $\left\{t_{ij}, t_{i+1,j}, \ldots, t_{i+M-1,j}\right\}_M$ and encrypts it as $\left[D'_{aj}\right]$. Finally, the encrypted transaction database $\left[D'\right]$ consists of $z \times m$ ciphertexts. The process of database encryption is shown in Figure 3. Similarly, DO computes the encryption of the masks S^1 and S^0 as $\left[S^1\right]$ and $\left[S^0\right]$, which both contain z ciphertexts. After encryption, DO outsources the encrypted database $\left[D'\right]$ and encrypted masks $\left[S^1\right]$, $\left[S^0\right]$ to CSP for further mining operations.

We claim that the computation cost for DO in our TDE protocol is very low (shown in the experiment results) since both z and m have small values in general. Besides, since we pack the same attributes of multiple transactions into one ciphertext, our TDE protocol can be easily executed to support the circumstances of multiple data owners with horizontally or vertically partitioned databases.

- For horizontally partitioned databases D_l ($1 \le l \le L$, each DO has some transactions with all attributes), DO_l chooses its f_l and runs the TDE protocol to compute its encrypted database $[D'_l]$ as well as the encrypted masks $[S^1_l]$ and $[S^0_l]$. After receiving all ciphertexts from different data owners, CSP could run frequent itemset mining over the combined encrypted database $\left[D'\right] = \left\{\left[D'_1\right], \ldots, \left[D'_L\right]\right\}$ with all the encrypted masks $\left\{\left[S^1_1\right], \ldots, \left[S^1_L\right]\right\}$ and $\left\{\left[S^0_1\right], \ldots, \left[S^0_L\right]\right\}$.
- For vertically partitioned databases D_l ($1 \le l \le L$, each DO has all transactions with m_l attributes), all DOs use the same f and random permutation function (therefore generating the same masks S^1 and S^0), change m to m_l in Step 7 and execute the TDE protocol to obtain the encrypted values. In this situation, CSP could perform the mining task on the aggregated encrypted database $\left[D'\right] = \left\{[D'_1], \ldots, [D'_L]\right\}$ by just using the two encrypted masks $\left[S^1\right]$ and $\left[S^0\right]$.

4.2 Itemset encryption (IE)

In this protocol, we consider a query user (QU) with a private binary itemset $x = (x_1, \ldots, x_m)$ as input, and outputs the encrypted itemset $[x]$ and encrypted itemset length $[k]$ using the YASHE encryption with ciphertext packing technique. Here, k is the number of items included in x, i.e., the number of attributes x_j with value of one for $1 \le j \le m$. After encryption, $[x]$ and $[k]$ will be uploaded to CSP for frequent itemset query. The proposed itemset encryption (IE) protocol is given in Algorithm 2.

Algorithm 2 IE$(x, k) \rightarrow \{[x], [k]\}$.

```
Performed by: QU
for $j = 1$ to $m$
    $Pack \leftarrow \left\{x_j, \ldots, x_j\right\}_M$
    $\left[x_j\right] \leftarrow E_Y(Pack)$
$Pack \leftarrow \{k, \ldots, k\}_M$
$[k] \leftarrow E_Y(Pack)$
Send $[x] = \{[x_1], \ldots, [x_m]\}$ and $[k]$ to CSP
```

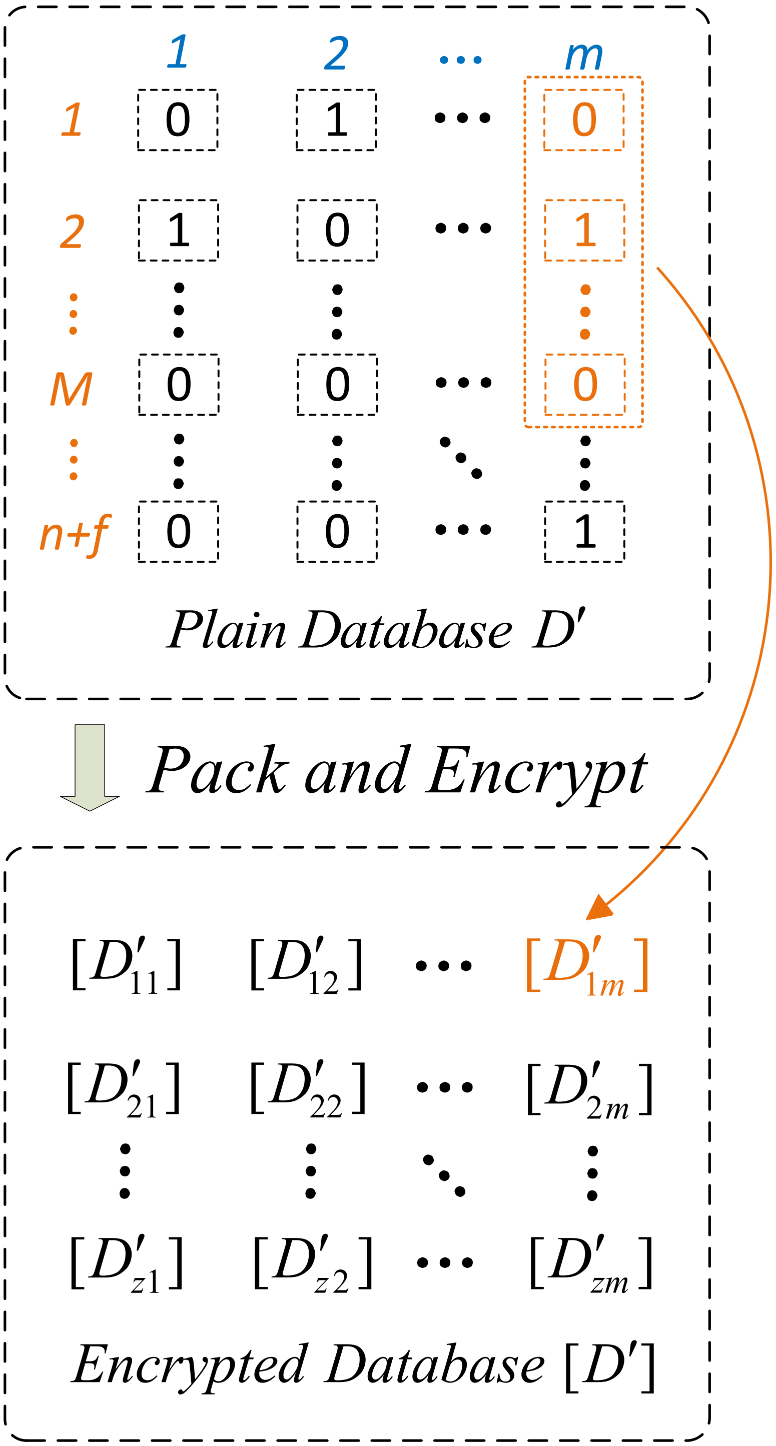

1
2
…
m
1
0
1
…
0
2
1
0
…
1
M
0
0
…
0
n+f
0
0
…
1
Plain Database D'
Pack and Encrypt
$[D'_{11}]$ $[D'_{12}]$ $\cdots$ $[D'_{1m}]$
$[D'_{21}]$ $[D'_{22}]$ $\cdots$ $[D'_{2m}]$
$[D'_{z1}]$ $[D'_{z2}]$ $\cdots$ $[D'_{zm}]$
Encrypted Database $[D']$

Figure 3 Transaction database encryption

This algorithm is very simple by just packing and encrypting M same attributes x_j into one ciphertext $[x_j]$ for $1 \leq j \leq m$. Similarly, QU also encrypts the itemset length k into $[k]$ using ciphertext packing technique. In this way, parallel computation with M transactions are enabled in the following secure frequent itemset mining (SFIM) protocol.

4.3 Secure frequent itemset mining (SFIM)

In this protocol, we consider that CSP and Evaluator cooperate to perform frequent itemset mining for QU's encrypted query itemset $[x]$ based on the encrypted transaction database $\left[D'\right]$ outsourced by DO. During the execution, the two clouds securely compute the Paillier encrypted support $\langle S_x \rangle$ for the itemset $[x]$, compare it with the Paillier encrypted threshold of minimum support $\langle ms \rangle$ and output the comparison result θ ($\theta = 1$ indicates x is frequent and $\theta = 0$ otherwise). The proposed secure frequent itemset mining (SFIM) protocol is given in Algorithm 3.

As shown in the algorithm, the SFIM protocol is complicated with interactive processes between CSP and Evaluator. The protocol can be partitioned into four main parts. In the first part, CSP executes Step 1-11 to compute the shuffled YASHE ciphertexts $[w]$, which contain packed plain values of zeros or random numbers. Here, the number of zeros represents the dummy support of $[x]$ since we insert fake transactions in Step 5. In the second part (Step 12-17), Evaluator decrypts $[w]$ and checks each plain value w_{ab} for $1 \leq a \leq z$ and $1 \leq b \leq M$. The YASHE ciphertexts $[u]$ are computed according to the values of w_{ab}, which are then sent to CSP. Next, in the third part, CSP performs Step 18-25 to remove the influence of fake transactions and computes the YASHE ciphertext $[\beta]$, which contains perturbed partial supports of $[x]$. In the last part (Step 26-33), CSP and Evaluator first compute the Paillier encrypted support $\langle S_x \rangle$ for the query itemset $[x]$. Then, they securely compare $\langle S_x \rangle$ with the Paillier encrypted minimum support $\langle ms \rangle$ to check whether $[x]$ is frequent or not.

Specifically, in Part 1 of SFIM (Step 1-11), for $1 \leq a \leq z$ and $1 \leq j \leq m$, CSP first randomly generates the attributes of fake transactions (Step 4), then it uses the encrypted mask $\left[S_a^1\right]$ to filter out the entries of the real transactions and remain the random values for the positions of all-zeros transactions added in the TDE protocol $\left(\left[\gamma_a^j\right] = \left[S_a^1\right] \times \sigma_a^j \text{ in Step 5}\right)$. Next, CSP inserts these fake transactions into the encrypted database $\left[D'\right]$ by computing $\left[D''_{aj}\right] = \left[D'_{aj}\right] \oplus \left[\gamma_a^j\right]$. Note that since the entries of mask S^1 for the real transactions are all zeros, adding $\left[\gamma_a^j\right]$ to $\left[D'_{aj}\right]$ will not change the values of these true transactions. The new encrypted database $\left[D''\right] = \left\{\left[D''_{11}\right], \ldots, \left[D''_{1m}\right], \ldots, \left[D''_{z1}\right], \ldots, \left[D''_{zm}\right]\right\}$ is now inserted with f random fake transactions. After that, CSP multiplies $\left[D''_{aj}\right]$ with the encrypted attribute $[x_j]$ of the query itemset to obtain $\left[p_j\right]$ $(1 \leq j \leq m)$, and sums up them to get the encrypted inner product $[IP_a]$ $(1 \leq a \leq z)$. Then, CSP generates a pack of M positive random numbers $r_a = \{r_{a1}, \ldots, r_{aM}\}_M$ from $\mathbb{Z}_P^*$ (P is the plaintext modulus of YASHE), and computes ciphertexts $[w_a] = ([IP_a] \ominus [k]) \times r_a$ by subtracting the encrypted query itemset length $[k]$ $(1 \leq a \leq z)$. Here, each $[w_a]$ consists of packed plain values of zeros or random numbers, and the entries of zeros indicate that the corresponding transactions contain the query itemset x. This is because that if x is included in a transaction, the inner product of them should be k, which is the itemset length of x (i.e., the number of ones in x). To note that, since we pack M plain values in each ciphertext, parallel computations are inherently executed in these aforementioned operations. Finally, CSP generates a random permutation function π to shuffle $[w] = \{[w_1], \ldots, [w_z]\}$ and sends the permuted result to Evaluator.

Algorithm 3 SFIM$\left(\left[D'\right], \left[S^0\right], \left[S^1\right], [x], [k], \langle ms \rangle\right) \rightarrow \{\theta, \langle S_x \rangle\}$.

```
Performed by: CSP and Evaluator
 1: CSP performs:
 2: for $a = 1$ to $z$
 3:     for $j = 1$ to $m$
 4:         $\sigma_{ab}^j \xleftarrow{R} \{0, 1\}, 1 \le b \le M, \sigma_a^j \leftarrow \left\{\sigma_{a1}^j, \ldots, \sigma_{aM}^j\right\}_M$
 5:         $\left[\gamma_a^j\right] \leftarrow \left[S_a^1\right] \times \sigma_a^j, \left[D''_{aj}\right] \leftarrow \left[D'_{aj}\right] \oplus \left[\gamma_a^j\right]$
 6:         $\left[p_j\right] \leftarrow \left[D''_{aj}\right] \otimes [x_j]$
 7:     $[IP_a] \leftarrow \sum_{j=1}^{m} \left[p_j\right]$
 8:     $r_{ab} \xleftarrow{R} \mathbb{Z}_P^*, 1 \le b \le M, r_a \leftarrow \{r_{a1}, \ldots, r_{aM}\}_M$
 9:     $[w_a] \leftarrow ([IP_a] \ominus [k]) \times r_a$
10: Generate $\pi$ to randomly shuffle $[w] = \left\{[w_1], \ldots, \left[w_z\right]\right\}$
11: Send $[w]$ to Evaluator
12: Evaluator performs:
13: for $a = 1$ to $z$
14:     $\{w_{a1}, \ldots, w_{aM}\}_M \leftarrow D_Y([w_a])$
15:     if $w_{ab} = 0, u_{ab} \leftarrow 1$, else $u_{ab} \leftarrow 0, 1 \le b \le M$
16:     $Pack \leftarrow \{u_{a1}, \ldots, u_{aM}\}_M, [u_a] \leftarrow E_Y(Pack)$
17: Send $[u] = \{[u_1], \ldots, [u_z]\}$ to CSP
18: CSP performs:
19: Use $inv(\pi)$ to permute $[u]$
20: for $a = 1$ to $z$
21:     $\left[u'_a\right] \leftarrow [u_a] \otimes \left[S_a^0\right]$
22:     $r_{ab} \xleftarrow{R} \mathbb{Z}_P, 1 \le b \le M, r_a \leftarrow \{r_{a1}, \ldots, r_{aM}\}_M$
23:     $\left[u''_a\right] \leftarrow \left[u'_a\right] + r_a, \delta_a \leftarrow \sum_{b=1}^{M} r_{ab}$
24: $[\beta] \leftarrow \sum_{a=1}^{z} \left[u''_a\right], \delta \leftarrow \sum_{a=1}^{z} \delta_a \; mod \; P, \langle \delta \rangle \leftarrow E_P(\delta)$
25: Send $[\beta]$ to Evaluator
26: Evaluator performs:
27: $\{\beta_1, \ldots, \beta_M\}_M \leftarrow D_Y([\beta]), \eta \leftarrow \sum_{b=1}^{M} \beta_b \; mod \; P$
28: $\langle \eta \rangle \leftarrow E_P(\eta)$, send $\langle \eta \rangle$ to CSP
29: CSP and Evaluator perform $\theta \leftarrow SC(\langle \eta \rangle, \langle \delta \rangle)$
30: if $\theta = 0$ then CSP computes $\langle \eta \rangle \leftarrow \langle \eta \rangle \cdot \langle P \rangle \; mod \; N^2$
31: CSP computes $\langle S_x \rangle \leftarrow \langle \eta \rangle \cdot \langle \delta \rangle^{-1} \; mod \; N^2$
32: CSP and Evaluator perform $\theta \leftarrow SC(\langle S_x \rangle, \langle ms \rangle)$
33: Send $\theta$ to QU
```

After receiving $[w]$, Evaluator performs the second part (Step 12-17) to compute the ciphertexts $[u]$. First, Evaluator decrypts each $[w_a]$ to obtain the packed plain values w_{ab} for $1 \le a \le z$ and $1 \le b \le M$. Then, Evaluator sets $u_{ab} = 1$ or $u_{ab} = 0$ depending on whether $w_{ab} = 0$ or not. Here, the value of $\sum_{a=1}^{z} \sum_{b=1}^{M} u_{ab}$ is the dummy support of $[x]$ based on the encrypted database $\left[D''\right]$ which contains inserted fake transactions. To compute the real support while preserving privacy, for $1 \le a \le z$, Evaluator computes the ciphertexts $[u_a]$ and sends them to CSP.

Next, in Part 3 (Step 18-25), CSP first computes the inverse permutation function $inv(\pi)$ and uses it to permute the ciphertext vector $[u]$. Then, for $1 \le a \le z$, CSP removes the support counts (entries of ones inside $[u_a]$) for the inserted fake transactions by multiplying $[u_a]$ with the mask $\left[S_a^0\right]$, in which the corresponding entries are all zeros. The obtained ciphertexts $\left[u'_a\right]$ contain the correct support counts for the real transactions. Next, CSP generates random numbers r_{ab} from $\mathbb{Z}_P$, uses them to perturb $\left[u'_a\right]$ and obtains $\left[u''_a\right]$. After that, CSP computes the sum of ciphertexts $\left[u''_a\right]$ to get the ciphertext $[\beta]$, which is packed with M perturbed partial supports of $[x]$. Besides, CSP also calculates $\delta = \sum_{a=1}^{z} \delta_a \; mod \; P$ and encrypts it as $\langle\delta\rangle$ using Paillier encryption, which will be used to remove perturbation. At last, CSP sends the ciphertext $[\beta]$ to Evaluator.

In the last part of SFIM (Step 26-33), Evaluator first decrypts $[\beta]$ into $\{\beta_1, \ldots, \beta_M\}_M$ and computes $\eta = \sum_{b=1}^{M} \beta_b \; mod \; P$. Here, η is the perturbed real support of the query itemset $[x]$. To preserve privacy, Evaluator encrypts η into $\langle\eta\rangle$ using Paillier encryption and sends it to CSP. In Step 29-31, by applying the SC (Secure Compare) protocol [34], CSP and Evaluator cooperate to compute the Paillier encrypted real support $\langle S_x\rangle$. Here, the SC protocol securely compares $\langle\eta\rangle$ and $\langle\delta\rangle$, outputs $\theta = 1$ if $\eta > \delta$ or $\theta = 0$ otherwise. Then, the two clouds run SC to compare $\langle S_x\rangle$ with the Paillier encrypted minimum support $\langle ms\rangle$ to check whether $[x]$ is frequent ($\theta = 1$) or not ($\theta = 0$). At last, θ is sent to QU as the query result.

To note that, when we compute η in Step 27, we modulo the plaintext modulus P of YASHE encryption. Here, to obtain correct support S_x, we require that $S_x < P$ to guarantee that $S_x \; mod \; P = S_x$, which is generally satisfied by using large P (e.g., $P = 114689$). Now, we analyze the reason for performing Step 29 and 30. Let $u'_a = \left\{u'_{a1}, \ldots, u'_{aM}\right\}_M$, so we have $S_x = \sum_{a=1}^{z} \sum_{b=1}^{M} u'_{ab}$. Also, let $r = \sum_{a=1}^{z} \sum_{b=1}^{M} r_{ab}$ to be the sum of all perturbations. Then, we obtain that $\eta = (S_x + r) \; mod \; P$ and $\delta = r \; mod \; P$. Here, if $S_x \; mod \; P + r \; mod \; P < P$, we get $\eta = S_x \; mod \; P + r \; mod \; P = S_x + \delta$, otherwise we get $\eta = (S_x \; mod \; P + r \; mod \; P) - P = S_x - P + \delta$. In the first circumstance, we have $\eta - \delta = S_x > 0$ ($\theta = 1$ in Step 29), while for the second situation, we have $\eta - \delta = S_x - P < 0$ ($\theta = 0$ in Step 29). Therefore, if we get $\theta = 0$, it implies that $\eta < \delta$, then we have to compute $\eta + P = S_x + \delta$ (in Step 30, $\langle\eta\rangle = \langle\eta\rangle \cdot \langle P\rangle = \langle\eta + P\rangle$) to ensure obtaining correct support in Step 31 ($\langle S_x\rangle = \langle\eta\rangle \cdot \langle\delta\rangle^{-1} = \langle\eta - \delta\rangle$).

4.4 Secure support decryption (SSD)

In this protocol, we securely decrypt the Paillier encrypted support $\langle S_x\rangle$ to obtain the plain value S_x, which is only known by QU. The proposed secure support decryption (SSD) protocol is given in Algorithm 4.

```
**Algorithm 4** SSD($\langle S_x\rangle$) $\rightarrow S_x$.
**Performed by:** Clouds and QU
**CSP performs:**
$r \xleftarrow{R} \mathbb{Z}_N, \langle r\rangle \leftarrow E_P(r), \left\langle S'_x\right\rangle \leftarrow \langle S_x\rangle \cdot \langle r\rangle \; mod \; N^2$
Send $\left\langle S'_x\right\rangle$ to Evaluator, send $r$ to QU
**Evaluator performs:**
$S'_x \leftarrow D_P\left(\left\langle S'_x\right\rangle\right)$, send $S'_x$ to QU
QU performs $S_x \leftarrow \left(S'_x - r\right) \; mod \; N$
```

As shown in the algorithm, CSP first adds noise r to the Paillier encrypted support $\langle S_x \rangle$. Here, the noise r is randomly generated from the plaintext space $\mathbb{Z}_N$ of Paillier cryptosystem. Then, CSP sends the perturbed ciphertext $\left\langle S'_x \right\rangle$ to Evaluator and sends the perturbation r to QU. Next, Evaluator performs decryption and sends the plain perturbed support S'_x to QU. At last, QU simply removes the noise to obtain the plain support S_x.

5 The proposed scheme

In this section, we propose our new privacy-preserving frequent itemset query (PPFIQ) scheme over semantically secure encrypted database. The proposed scheme is designed for the situation that a query user (QU) wants to securely identify whether its private itemset is frequent or not based on the encrypted transaction database on cloud server. To the best of our knowledge, our PPFIQ scheme is the first approach supporting efficient frequent itemset mining on semantically secure encrypted data without on-line DO and QU.

5.1 Privacy-preserving frequent itemset query (PPFIQ)

In this section, using the proposed protocols in Section 4 as sub-routines, we provide our new PPFIQ scheme which efficiently achieves frequent itemset mining for a given query itemset over encrypted cloud database. The details of the scheme are given in Algorithm 5 as follows.

Algorithm 5 PPFIQ$(D, x, k, ms) \rightarrow \{\theta, S_x\}$.

```
$\{[D'], [S^1], [S^0]\} \leftarrow$ TDE$(D)$ (DO)
$\langle ms \rangle \leftarrow E_P(ms)$, send $\langle ms \rangle$ to CSP (DO)
$\{[x], [k]\} \leftarrow$ IE$(x, k)$ (QU)
$\{\theta, \langle S_x \rangle\} \leftarrow$ SFIM$\big([D'], [S^0], [S^1], [x], [k], \langle ms \rangle\big)$ (Clouds)
$S_x \leftarrow$ SSD$(\langle S_x \rangle)$ (Clouds and QU)
```

As shown in the algorithm, the data owner (DO) first runs the TDE protocol to encrypt its private transaction database D using the YASHE encryption with ciphertext packing technique. Except for the encrypted database $[D']$, TDE also outputs two encrypted masks $[S^1]$ and $[S^0]$, which are used to insert fake transactions and remove dummy support counts in the SFIM protocol. Next, DO encrypts the threshold of minimum support ms using the Paillier encryption to get the ciphertext $\langle ms \rangle$, which will be used in the SFIM protocol. Then, DO outsources the ciphertexts $[D']$, $[S^1]$, $[S^0]$ and $\langle ms \rangle$ to CSP, and stays off-line after that. To note that, the proposed PPFIQ scheme supports the situations of multiple data owners with horizontally or vertically partitioned databases. For conciseness, we simply give the situation for one DO, while a set of TDE protocols are required for multiple DOs (see detailed discussion in Section 4.1). To preserve privacy and enable parallel computation, QU executes the IE protocol to encrypt its private query itemset x and itemset length k using YASHE with ciphertext packing technique. Then, QU uploads the ciphertexts $[x]$ and $[k]$ to CSP for frequent itemset mining, and stays off-line after that. Next, the two clouds CSP and Evaluator cooperate to perform the SFIM protocol for QU's encrypted query itemset $[x]$ based on DO's encrypted database $[D']$. During the execution, the Paillier encrypted support $\langle S_x \rangle$ is computed and compared with the encrypted minimum support $\langle ms \rangle$, returning the mining result θ to QU ($\theta = 1$ indicates x is frequent and $\theta = 0$ otherwise). At last, the clouds and QU together run the SSD protocol to securely decrypt $\langle S_x \rangle$ and obtain the plain support S_x, which is only known by QU.

5.2 Security analysis

In this section, based on the standard simulation argument [15], we assume CSP and Evaluator to be the potential adversaries and provide security proof for the proposed PPFIQ scheme. We first prove the security of the protocols (used as sub-routines in PPFIQ) given in Section 4 under the semi-honest model. Then, according to the Composition Theorem [15], we prove that PPFIQ is also secure under the semi-honest model. Briefly, we preserve privacy from CSP since all data and messages it has are semantically secure ciphertexts, while the information obtained by Evaluator are all random values that also leak no privacy. As discussed in Threat Model (Section 3.2), we assume that CSP and Evaluator cannot collude with each other, and they cannot corrupt DO or QU to obtain extra information. Now we provide the security analysis as follows.

- **Security of TDE and IE**
  The security of the proposed TDE and IE protocols can be directly obtained from the security of the YASHE encryption. Since the encrypted transactions and itemsets are only submitted to CSP which cannot collude with Evaluator, both of these private data are protected from the two clouds with semantic security.
- **Security of SFIM**

Theorem 1 *The proposed SFIM protocol is secure against the semi-honest CSP as long as the YASHE and Paillier encryptions are semantically secure, and also against a semi-honest Evaluator to disclose privacy from the intermediate computation results as long as the non-collusion assumptions are satisfied.*

Proof To formally prove the security of SFIM under the semi-honest model, we need to demonstrate that the simulated image of SFIM is computationally indistinguishable from its actual execution image. An execution image generally contains the exchanged messages and the results computed from these messages. □

According to Algorithm 3, we have the execution image of CSP denoted as $\Pi_C(\text{SFIM}) = \{[u], \langle\eta\rangle\}$, where $[u]$ and $\langle\eta\rangle$ are YASHE and Paillier ciphertexts respectively. Let the simulated image of CSP be $\Pi_C^S(\text{SFIM}) = \left\{[u]', \langle\eta\rangle'\right\}$, where $[u]'$ and $\langle\eta\rangle'$ are randomly generated from the ciphertext spaces of YASHE and Paillier cryptosystems separately. Since the underlying encryptions are semantically secure, it implies that $[u]$ and $\langle\eta\rangle$ are computationally indistinguishable from $[u]'$ and $\langle\eta\rangle'$ respectively. Therefore, $\Pi_C(\text{SFIM})$ is computationally indistinguishable from $\Pi_C^S(\text{SFIM})$ based on Definition 1. Besides, the security proof of the SC (Secure Compare) protocol is given in [34]. Therefore, according to the Composition Theorem [15], CSP cannot learn any private information during the execution of SFIM.

For Evaluator, the execution image is denoted as $\Pi_E(\text{SFIM}) = \{[w], w_{ab}, u_{ab}, [\beta], \beta_b, \eta\}$ for $1 \le a \le z$ and $1 \le b \le M$. Here, $[w]$ and $[\beta]$ are YASHE ciphertexts, w_{ab} and β_b are corresponding decrypted packed plaintexts, u_{ab} and η are plain values computed using w_{ab} and β_b respectively. Without loss of generality, we assume the simulated image of Evaluator to be $\Pi_E^S(\text{SFIM}) = \left\{[w]', w'_{ab}, u'_{ab}, [\beta]', \beta'_b, \eta'\right\}$ for $1 \le a \le z$ and $1 \le b \le M$. Here, $[w]'$ is set as $[w]' = \{[p_1 \cdot q_1], \ldots, [p_z \cdot q_z]\}$, in which $p_a = \{p_{a1}, \ldots, p_{aM}\}_M$ and

$q_a = \{q_{a1}, \ldots, q_{aM}\}_M$ with $p_{ab} \overset{R}{\leftarrow} \{0, 1\}$ and $q_{ab} \overset{R}{\leftarrow} \mathbb{Z}_P^*$ for $1 \le a \le z$ and $1 \le b \le M$. We define $p_a \cdot q_a = \{p_{a1}q_{a1}, \ldots, p_{aM}q_{aM}\}_M$ as an element-wise product, and $[p_a \cdot q_a]$ is a YASHE ciphertext with M packed plain values of $p_{ab}q_{ab}$ for $1 \le b \le M$. Similarly, $[\beta]'$ is set as $[\beta]' = [e]$, where $e = \{e_1, \ldots, e_M\}_M$ with $e_b \overset{R}{\leftarrow} \mathbb{Z}_P$ for $1 \le b \le M$. For the other simulated values, w'_{ab} and β'_b are decrypted plaintexts of $[w]'$ and $[\beta]'$, u'_{ab} and η' are computed using w'_{ab} and β'_b as in Step 15 and 27 of SFIM respectively.

Obviously, Evaluator cannot distinguish $[w]'$ and $[w]$, $[\beta]'$ and $[\beta]$ due to the semantic security of YASHE. Although Evaluator can decrypt $[w]'$ and $[\beta]'$ with the secret key, all the $w'_{ab} = p_{ab}q_{ab}$ and $\beta'_b = e_b$ are perturbed and randomly distributed as the decrypted w_{ab} and β_b respectively. For u'_{ab} and u_{ab}, it is apparent that u'_{ab} is randomly distributed because of the randomness of w'_{ab}, while u_{ab} is also randomly distributed since random fake transactions are inserted into the database. Similarly, η' is a random number due to the random choices of β'_b, and η is also randomly perturbed with additive noise. Therefore, Evaluator also cannot distinguish w'_{ab} and w_{ab}, β'_b and β_b, as well as u'_{ab} and u_{ab}, η' and η. Based on these analyses, it implies that Π_E(SFIM) is computationally indistinguishable from Π_E^S(SFIM) according to Definition 1. Considering the security of SC protocol [34] and the Composition Theorem [15], Evaluator cannot figure out any privacy during the process of SFIM. Putting everything together, we claim that the proposed SFIM protocol is secure against CSP and Evaluator under the semi-honest model according to Definition 1.

- **Security of SSD** Similar as in SFIM, in the SSD protocol, CSP only views semantically secure ciphertexts, while Evaluator only obtains random numbers. Therefore, by using the same method of security analysis for SFIM, we omit detailed discussion here and claim that the proposed SSD protocol is secure against CSP and Evaluator under the semi-honest model according to Definition 1.
- **Security of PPFIQ** As shown in Algorithm 5, the proposed PPFIQ scheme is a sequential composition of the secure protocols given in Section 4 (except Step 2 performed by DO). Since we have already proved the security for all the sub-routines, according to Definition 1 and the Composition Theorem [15], we claim that our PPFIQ scheme is secure against semi-honest CSP and Evaluator. Besides, since the real support of the query itemset is hidden from both clouds, our scheme also resists frequency analysis attacks.

6 Experiment results

In this section, we execute experiments to demonstrate the performance of the proposed scheme. The experiment results show that our solution requires low computation costs, making it suitable to be used on large databases. All experiments were implemented on Windows 10 with Intel Core i7-7700HQ 2.80 GHz CPU and 16 GB RAM. We use the SEAL library [7] to perform the YASHE encryption with 128-bits security. We set the degree of polynomial modulus $M = 4096$ and the plaintext modulus $P = 114689$ to guarantee efficiency and correctness. Another YASHE parameter decomposition bit count (dbc) is set to be 16 for small consumptions of noise budgets. For Paillier encryption, we use the Crypto++ 5.6.3 library to implement it with 1024-bits security. Besides, we set the number of all-zeros transactions $f = n/2$ and add some extra ones to ensure $M \mid (n + f)$. To better show the performance, we run experiments on several real databases[1] as well as synthetic ones with

[1] http://fimi.ua.ac.be/data/

different parameter settings. To compare the performance, we also run two state-of-the-art schemes [28, 34] using the security parameters given in the two papers. According to these two works, $n/2$ dummy transactions are also inserted in the original transaction database to preserve the real support of the query itemset. Next, we provide the experiment results as follows.

- **Experiments on real databases**

  We use four real databases: chess, mushroom, connect and pumsb to implement the experiments. The parameters of these databases and the computation costs (average values for 1000 times) for the proposed protocols are given in Table 4.

As shown in the table, for the smallest database chess, running TDE to encrypt all the transactions only needs 0.7 second, and a QU only takes 0.4 second to perform itemset encryption (IE), while 1.5 second is required to securely perform frequent itemset mining by the two clouds. For the next two larger databases mushroom and connect, the computation costs of the three protocols increase with the database size and attribute, but still remain quite low. Here, the running time of TDE and SFIM are determined by both n and m, while the execution cost of IE is only affected by m. The last database pumsb contains more transactions with very large attribute value ($m = 2113$), which leads to more computation costs. Here, a DO needs 141.8 second to perform database encryption for all the 49046 transactions. Nevertheless, TDE is a one-time work and the computation time is still reasonable. For a QU, it takes 7.8 second to encrypt an itemset with 2113 attributes, which remains efficient even for the users with limited computation resources. Clearly, performing SFIM also consumes more cost for CSP and Evaluator, which takes 326.6 second. However, considering cloud servers with much powerful computation ability, the running time of SFIM for the pumsb database is still quite acceptable. We do not provide the computation cost for the SSD protocol since it is negligible.

For comparison, using the chess database, we provide the experiment results of frequent itemset mining on ciphertexts in Table 5. We do not implement experiments for the two compared schemes [28, 34] on other larger databases since they have very heavy computation costs. As shown in the table, the schemes in [34] and [28] require 668.12 and 734.84 minute respectively to perform frequent itemset mining for a query itemset, while our scheme only needs 1.5 second by using the proposed SFIM protocol, which is much more efficient (26724 times faster than [34] and 29393 times faster than [28]). Here, there are two main reasons of the remarkable efficiency improvement of our PPFIQ scheme. First, we introduce the ciphertext packing technique in our new designed secure protocols, which achieves efficient parallel computations without extra cost. Second, since we utilize fully homomorphic encryption YASHE as our main underlying cryptosystem, our proposed scheme requires less interactive steps, which further improves the efficiency.

Table 4 Computation costs on real databases (in second)

Database	Size (n)	Attribute (m)	TDE cost	IE cost	SFIM cost
Chess	3196	75	0.7	0.4	1.5
Mushroom	8124	119	1.5	0.5	3.3
Connect	18078	123	3.4	0.6	7.6
Pumsb	49046	2113	141.8	7.8	326.6

Table 5 Mining costs on chess database

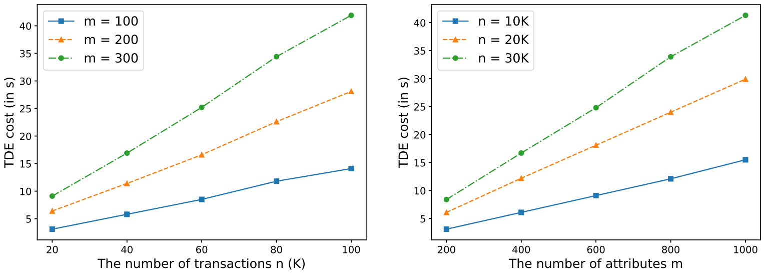
Performance	Our scheme	Scheme in [34]	Scheme in [28]
Mining cost	1.5 s	668.12 min	734.84 min

- **Experiments on synthetic databases** To better demonstrate the performance of our PPFIQ scheme, we generate multiple synthetic databases by varying the database size n and attribute m. The computation costs are shown in Figures 4–6, which are the average values for 1000 times.

As shown in Figure 4, the computation costs of the TDE protocol increase linearly along with both n and m. In Figure 4a, we vary n from 20K to 100K, and run experiments with three different values of m. When the attribute m is small ($m = 100$), the running time of TDE grows slowly, which only needs about 3 second for $n = 20$K and 14 second for $n = 100$K. The computation cost increases quickly when the database has large attribute. For $m = 200$, encrypting the database requires about 7 second with $n = 20$K, while the cost increases to 28 second with $n = 100$K. When a larger m is used ($m = 300$), the cost of TDE is about 9 second for a small database ($n = 20$K) and about 42 second for a large database ($n = 100$K). In Figure 4b, we vary m from 200 to 1000, and test the running time of TDE with three different values of n. When the size of the database is small ($n = 10$K), encrypting the database is very efficient with costs of 3 second and 15 second for $m = 200$ and $m = 1000$. More running time is consumed with larger database size n. When varying m from 200 to 1000, the cost of TDE increases from 6 second to 30 second for $n = 20$K, and increases form 8 second to 41 second for $n = 30$K. Considering the TDE protocol is a one-time work, the computation costs shown in Figure 4 are quite acceptable for a data owner who can efficiently perform encryption for large databases.

The computation costs of the IE protocol are shown in Figure 5, which are only affected by the values of attribute m. As shown in the figure, we vary m from 200 to 1000, and the cost of IE increases from 0.8 second to 3.7 second. This is very efficient for a query user who might have restrained computation resources and enables it to launch multiple queries simultaneously.

The computation costs of the SFIM protocol are shown in Figure 6, which grow linearly along with the database size n and attribute m. In Figure 6a, we vary n from 20K to 100K and run SFIM with different values of m. When m is small ($m = 100$), the execution of SFIM is very efficient (32 second) even for a large database with $n = 100$K. The running time increases quickly when m is large, which grows from 14 to 64 second for $m = 200$, and



(a) (b)

Figure 4 Computation costs of TDE (in second)

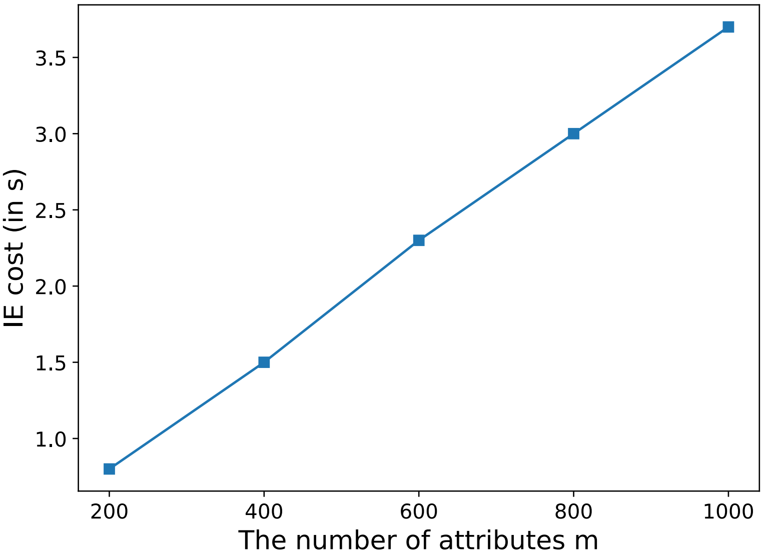
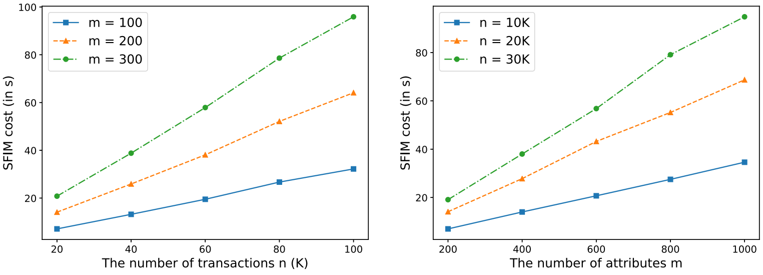


Figure 5 Computation costs of IE (in second)

21 to 96 second for $m = 300$. In Figure 6b, we test SFIM by setting different values of n and varying m from 200 to 1000. As shown in the figure, when m is small ($m = 200$), executing frequent itemset mining requires less than 20 second (7, 14, 19 second for n =10K, 20K, 30K respectively). As for a large $m = 1000$, running SFIM is still efficient with different database sizes n, which takes 35 second for $n = $ 10K, 69 second for $n = $ 20K and 95 second for $n = $ 30K. Since the process of frequent itemset mining is performed by cloud servers which might have "unlimited" computation power, the SFIM protocol is efficient and suitable to be applied on large databases.

By analyzing the experiment results, we find that our proposed PPFIQ scheme introduces low computation costs for both DO and QU, and efficiently performs frequent itemset mining on encrypted databases. From the computation costs for a set of real and synthetic databases, it is clear that PPFIQ could be applied on databases with various sizes and attributes, showing its practicability in real-world situations. Besides, parallel operations



(a) (b)

Figure 6 Computation costs of SFIM (in second)

can be easily executed by partitioning the encrypted database into several parts and running SFIM on multiple clouds. In this way, we could further improve the efficiency of the outsourced frequent itemset mining process.

7 Conclusion and future work

This paper proposed a novel efficient scheme PPFIQ addressing the problem of privacy-preserving frequent itemset mining on outsourced encrypted cloud database. The proposed scheme preserves privacy for the original transaction database as well as the mining results. The whole mining process is performed by cloud servers without on-line users. We provided formal security analysis of our scheme and evaluated its performance with extensive experiments. The experiment results show that the proposed scheme can be efficiently implemented over large databases. For future work, we plan to extend our research on other privacy-preserving data mining tasks under more practical scenarios.

Acknowledgements This work has been supported by a grant from the National Natural Science Foundation of China (Grant No. 61801489).

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM Sigmod Record, ACM, vol 22, pp 207–216 (1993)
2. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Theory of Cryptography Conference, Springer, pp 325–341 (2005)
3. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: IMA International Conference on Cryptography and Coding. Springer, pp 45–64 (2013)
4. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp 37–54 (2003)
5. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. ACM Sigmod Rec. **26**(2), 255–264 (1997)
6. Brossette, S.E., Sprague, A.P., Michael. H.J., Waites, K.B., Jones, W.T., Moser, S.A.: Association rules and data mining in hospital infection control and public health surveillance. J. Am. Med. Inform. Assoc. **5**(4), 373–381 (1998)
7. Chen, H., Laine, K., Player, R.: Simple encrypted arithmetic library-seal v2.3.0–4 (2017)
8. Creighton, C., Hanash, S.: Mining gene expression databases for association rules. Bioinformatics **19**(1), 79–86 (2003)
9. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International Conference on Machine Learning, pp 201–210 (2016)
10. Du, J., Michalska, S., Subramani, S., et al.: Neural attention with character embeddings for hay fever detection from twitter. Health Inf Sci Syst **7**, 21 (2019)
11. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. Inf. Syst. **29**(4), 343–364 (2004)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. Stoc **9**(4), 169–178 (2009)
13. Giannotti, F., Lakshmanan, L.V., Monreale, A., Pedreschi, D., Wang, H.: Privacy-preserving mining of association rules from outsourced transaction databases. IEEE Syst. J. **7**(3), 385–395 (2013)
14. Goldreich, O.: General cryptographic protocols. Found. Crypt. **2**, 599–764 (2004)
15. Goldreich, O.: Encryption schemes. Found. Crypt. **2**, 373–470 (2004)
16. Huang, J., Peng, M., Wang, H., Cao, J., Gao, W., Zhang, X.: A probabilistic method for emerging topic tracking in microblog stream. World Wide Web **20**(2), 325–350 (2017)

17. Imabayashi, H., Ishimaki, Y., Umayabara, A., Sato, H., Yamana, H.: Secure frequent pattern mining by fully homomorphic encryption with ciphertext packing. In: Data Privacy Management and Security Assurance. Springer, pp 181–195 (2016)
18. Ji, Z., Li, H., Liu, X., Luo, Y., Chen, F., Wang, H., Chang, L.: On efficient and robust anonymization for privacy protection on massive streaming categorical information. IEEE Trans. Dependable Secure Comput. **14**(5), 507–520 (2015)
19. Ji, Z., Tao, X., Wang, H.: Outlier detection from large distributed databases. World Wide Web **17**(4), 539–568 (2014)
20. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans. Knowl. Data Eng. **16**(9), 1026–1037 (2004)
21. Lai, J., Li, Y., Deng, R.H., Weng, J., Guan, C., Yan, Q.: Towards semantically secure outsourcing of association rule mining on categorical data. Inform. Sci. **267**, 267–286 (2014)
22. Li, S., Nankun, M.U., Le, J., Liao, X.: Privacy preserving frequent itemset mining: Maximizing data utility based on database reconstruction. Comput. Secur. **84**, 17–34 (2019)
23. Li, L., Rongxing, L.U., Choo, K.K.R., Datta, A., Shao, J.: Privacy-preserving-outsourced association rule mining on vertically partitioned databases. IEEE Trans. Inf. Forensic. Secur **11**(8), 1847–1861 (2016)
24. Li, M., Sun, X., Wang, H., Zhang, Y., Ji, Z.: Privacy-aware access control with trust management in web service. World Wide Web **14**(4), 407–430 (2011)
25. Li, H., Ye, W., Wang, H., Zhou, B.: Multi-window based ensemble learning for classification of imbalanced streaming data. World Wide Web **20**, 1507–1525 (2017)
26. Lin, J.L., Liu, J.Y.C.: Privacy preserving itemset mining through fake transactions. In: Proceedings of the 2007 ACM Symposium on Applied Computing. ACM, pp 375–379 (2007)
27. Liu, L., Chen, R., Liu, X., Jinshu, S., Qiao, L.: Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Trans. Inf. Forensic. Secur. **15**, 2914–2929 (2020)
28. Liu, L., Jinshu, S., Chen, R., Liu, X., Wang, X., Chen, S., Leung, H.: Privacy-preserving mining of association rule on outsourced cloud data from multiple parties. In: Australasian Conference on Information Security and Privacy. Springer, pp 431–451 (2018)
29. Ma, C., Wang, B., Jooste, K., Zhang, Z., Ping, Y.: Practical privacy-preserving frequent itemset mining on supermarket transactions. IEEE Syst. J. **14**(2), 1992–2002 (2020)
30. Mohaisen, A., Jho, N.S., Hong, D., Nyang, D.: Privacy preserving association rule mining revisited: Privacy enhancement and resources efficiency. IEICE Trans. Inf. Syst. **93**(2), 315–325 (2010)
31. Molloy, I., Li, N., Li, T.: On the (in) security and (im) practicality of outsourcing precise association rule mining. In: 2009 Ninth IEEE International Conference on Data Mining. IEEE, pp 872–877, p. 2009 (2009)
32. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp 223–238 (1999)
33. Peng, M., Zeng, G., Sun, Z., Huang, J., Wang, H., Tian, G.: Personalized app recommendation based on app permissions. World Wide Web **21**, 89–104 (2018)
34. Qiu, S., Wang, B., Li, M., Liu, J., Shi, Y.: Toward practical privacy-preserving frequent itemset mining on encrypted cloud data. IEEE Trans. Cloud Comput. **8**(1), 312–323 (2020)
35. Rong, H., Wang, H., Liu, J., Xian, M.: Privacy-preserving k-nearest neighbor computation in multiple cloud environments. IEEE Access **4**, 9589–9603 (2016)
36. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**(1), 57–81 (2014)
37. Tai, C.H., Yu, P.S., Chen, M.S.: k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In: Proceedings of the 16th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining. ACM, pp 473–482 (2010)
38. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the Eighth ACM SIGKDD International Conference On Knowledge Discovery And Data Mining. ACM, pp 639–644 (2002)
39. Vimalachandran, P., Liu, H., Lin, Y., et al.: Improving accessibility of the Australian My Health Records while preserving privacy and security of the system. Health Inf Sci Syst **8**, 31 (2020)
40. Wang, H., Cao, J., Zhang, Y.: A flexible payment scheme and its role-based access control. IEEE Trans. Knowl. Data Eng. **17**(3), 425–436 (2005)
41. Wang, H., Wang, Y., Taleb, T., Jiang, X.: Special issue on security and privacy in network computing. World Wide Web **23**, 951–957 (2020)
42. Wang, H., Yi, X., Bertino, E., Sun, L.: Protecting outsourced data in cloud computing through access management. Concurr. Comput. Pract. Experience **28**(3), 600–615 (2016)
43. Wang, B., Zhan, Y.U., Zhang, Z.: Cryptanalysis of a symmetric fully homomorphic encryption scheme. IEEE Trans Inf. Forensic Secur. **13**(6), 1460–1467 (2018)

44. Wang, H., Zhang, Y., Cao, J.: Effective collaboration with information sharing in virtual universities. IEEE Trans. Knowl. Data Eng. **21**(6), 840–853 (2009)
45. Wang, H., Zhang, Z., Taleb, T.: Special issue on security and privacy of iot. World Wide Web **21**, 1–6 (2018)
46. Wei, W., Liu, J., Rong, H., Wang, H., Xian, M.: Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database. IEEE Access **6**, 41771–41784 (2018)
47. Wei, W., Parampalli, U., Liu, J., Xian, M.: Privacy preserving k-nearest neighbor classification over encrypted database in outsourced cloud environments. World Wide Web **22**(1), 101–123 (2019)
48. Wong, W.K., Cheung, D.W., Hung, E., Kao, B., Mamoulis, N.: Security in outsourcing of association rule mining. In: Proceedings of the 33rd International Conference On Very Large Data Bases. VLDB Endowment, pp 111–122 (2007)
49. Wu, W., Liu, J., Wang, H., Hao, J., Xian, M.: Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique. IEEE Trans. Knowl. Data Eng. (2020)
50. Wu, W., Liu, J., Wang, H., Tang, F., Xian, M.: Ppolynets: Achieving high prediction accuracy and efficiency with parametric polynomial activations. IEEE Access **6**, 72814–72823 (2018)
51. Yi, X., Rao, F.Y., Bertino, E., Bouguettaya, A.: Privacy-preserving association rule mining in cloud computing. In: Proceedings of the 10th ACM Symposium On Information, Computer And Communications Security. ACM, pp 439–450 (2015)
52. Yücel, S., Vassilios, S.V., Ahmed, K.E.: Privacy preserving association rule mining. In: Proceedings Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems RIDE-2EC 2002. IEEE, pp 151–158 (2002)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Wei Wu[1] · Ming Xian[2] · Udaya Parampalli[3] · Bin Lu[1]

Ming Xian
qwertmingx@sina.com

Udaya Parampalli
udaya@unimelb.edu.au

Bin Lu
stoneclever@126.com

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing,
Zhengzhou 450001, China

[2] College of Electronic Science and Technology, National University of Defense Technology,
Changsha 410073, China

[3] School of Computing and Information Systems, Melbourne School of Engineering,
University of Melbourne, Melbourne, Australia