electronics

Article

Verifiable Privacy-Preserving Outsourced Frequent Itemset Mining on Vertically Partitioned Databases

Zhen Zhao [1,2,*], Lei Lan [1], Baocang Wang [1,*] and Jianchang Lai [3]

1 State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
2 Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China
3 School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China
* Correspondence: zzhen@xidian.edu.cn (Z.Z.); bcwang@xidian.edu.cn (B.W.)

Abstract: In the data era, to simultaneously relieve the heavy computational burden of mining data information from data owners and protecting data privacy, privacy-preserving frequent itemset mining (PPFIM) is presented and has attracted much attention. In PPFIM, data owners and miners outsource the complex task of data mining to the cloud server, which supports strong storage and computing power, and the cloud server cannot extract additional data privacy other than that which is shown by data owners or miners. However, most existing solutions assume that cloud servers will honestly perform the mining process and return the correct results, whereas cloud services are usually provided by a charging third party that may in practice return incorrect results due to computation errors, malicious or criminal activities, etc. To solve this problem, in this paper, we present a verifiable PPFIM protocol on vertically partitioned databases to support the verifiability of the integrity of the mining results, where data owners can authorize the cloud server to perform federated mining on vertically partitioned databases without leaking data information and detect dishonest behaviors in the cloud server from the returned results. We adopt a dual cloud setting to enable data owners to be offline after uploading their encrypted databases to the cloud server, which further relieves the burden on data owners. We implement our protocol and give a detailed analysis in terms of verification accuracy, which shows that the dishonest behaviors of the cloud server can be detected with a probability close to 1 and a sacrifice of only a 1% increase in database size.

Keywords: frequent itemset mining; privacy-preserving; Paillier homomorphic encryption; vertically partitioned databases

check for updates

Citation: Zhao, Z.; Lan, L.; Wang, B.; Lai, J. Verifiable Privacy-Preserving Outsourced Frequent Itemset Mining on Vertically Partitioned Databases. *Electronics* **2023**, *12*, 1952. https://doi.org/10.3390/electronics12081952

Academic Editor: Andrei Kelarev

Received: 20 March 2023
Revised: 18 April 2023
Accepted: 19 April 2023
Published: 21 April 2023

1. Introduction

In the era of big data, data mining technology has attracted much attention since it enables the extraction of valuable information from data. As the core technology of association rule mining, frequent itemset mining (FIM) [1–3] is used to find frequent co-occurrence data items in large transaction databases. It has been widely used in market basket analysis [4], healthcare [5], intrusion detection [6], network traffic management [7], and bioinformatics [8]. Taking the most classic supermarket basket as an example, and given a transaction database, FIM can find the commodity combinations that are more frequently added to the same shopping list. Wal-Mart found that beer and diapers are often bought together because American husbands often buy two bottles of beer to themselves when they buy diapers for their children. Such information can help retailers optimize shelf placement and carry out selective marketing, thus increasing sales.

Due to extensive computation costs, data miners usually choose to outsource the task of mining to cloud servers that possess powerful storage and computation capacity. This greatly relieves the computation burden on the local devices of miners as well as exposes the data information of owners since the data will be uploaded to the cloud for mining. Privacy exposure, especially that of sensitive information, may even lead to huge economic losses or even more terrible consequences, such as the leakage of gene banks. This problem on the FIM has also received a substantial amount of attention, and the concept of privacy-preserving FIM (PPFIM) was proposed correspondingly [9,10]. PPFIM can mine the frequent itemsets among the databases while protecting the privacy security of data and mining results against curious cloud servers which, in practice, are often used by charging third parties.

To be realistically usable in further applications, subsequent research on PPFIM has also paid attention to supporting multiple data owners [11–13], where the mining is performed on a joint database that consists of the transactions collected from multiple users. In the literature, the joint database can be divided into horizontally partitioned databases and vertically partitioned databases. A horizontally partitioned database can be divided into multiple tables belonging to different users, where each table contains the same number of columns and different numbers of rows. Conversely, a vertically partitioned database can be split into multiple tables, each containing the same number of rows but different columns. For example, let us consider two tables with the same service types for two different companies, each consisting of transaction rows that identify the ordered services from different customers. We can create a horizontally partitioned database by simply adding the transactions of one table to another. On the other hand, if the two companies provide different service types for the same group of customers, we can obtain a vertically partitioned database by attaching the transactions of a user in one table with the corresponding user in another table.

In particular, as in our example, for a common scenario where, in reality, different merchants that sell different goods in the same region want to plan a joint sales promotion to stimulate consumption, vertically partitioned databases are more suitable. In this scenario, the merchants jointly mine the frequent itemsets with the help of a third cloud server and accordingly cooperate with the related stores to start the sales promotion. However, most existing PPFIM protocols assume that cloud servers can honestly perform a required mining job and correctly return the mining results. However, typically cloud servers cannot be fully trusted in practice, i.e., they may return incorrect results to the merchants to increase profit, engage in malicious activities, or due to some uncontrolled calculation errors, without any risk of being caught. Incorrect mining results may lead to a failed promotion or even substantial financial loss.

Contributions. In this paper, to solve the aforementioned problem, we propose a verifiable PPFIM protocol on vertically partitioned databases to support the checking of the integrity of returned mining results from the cloud. Our FIM protocol provides both privacy protection and verifies the integrity of the results, which is practically needed for data owners and miners who cooperate with not-fully trusted third parties.

In our protocol, prior to uploading their databases to the cloud, merchants, i.e., data owners, add artificial itemsets that are different from the original itemsets to the transactions, applying the technique proposed in [14]. Data owners can then check the integrity of returned results by detecting the malicious actions of cloud servers on these artificial itemsets. To resist the frequency analysis attack, merchants further adopt the algorithm proposed in [15] to insert fictitious transactions into their database. Merchants encrypt their databases and upload them to the cloud, and the cloud then performs mining on the joint encrypted database, following the improved FIM protocol [11] by applying the Paillier [16] algorithm in our solution. Compared with [11], we utilize a dual-cloud setting, instead of a single-cloud setting, such that the merchants do not need to stay online after outsourcing the data. We finally implement our solution and depict the computation costs, which shows that the verification accuracy increases along with the increase in the number of dishonest behaviors in the cloud and the proportion of fictitious itemsets. Particularly, as both the proportion of itemsets affected by the CSP's dishonest behaviors reaches 0.6% and the artificial itemsets proportion in the mining results reaches 1%, the data owner can capture dishonest behaviors from the CSP with a probability close to 100%.

Organization

The rest of this paper is organized as follows. In Section 2, we introduce the literature on PPFIM and verifiable PPFIM. In Section 3, we describe the preliminaries of our solution. Then, we introduce the system model, security model, and design goals of this paper in Section 4. In Section 5, we present our PPFIM scheme based on vertically partitioned databases. The security and experimental analysis of our protocol is shown in Section 6.

2. Related Work

Privacy-preserving frequent itemset mining (PPFIM). Due to the traditional involvement of FIM, the raw data of data owners may cause direct privacy leakage problems in practice; thus, PPFIM is presented [9,10,12] to protect data privacy in the mining process. PPFIM can be generally classified into randomization-based solutions and cryptography-based solutions, among which the former [15,17] hides the original data information through data perturbation techniques so that data mining can be performed without exposing original data values, and the latter protects data by encrypting them prior to uploading them to clouds. In cryptographic-based solutions, as we explained above, data are usually partitioned with two approaches: vertical [10,11,18] and horizontal [9,19]. In the literature on cryptographic-based PPARM algorithms, many studies [11–13,20] have been based on homomorphic encryption schemes [16] to support computations on encrypted data.

Verifiable PPFIM. In most PPFIM protocols, the cloud servers are supposed to honestly execute the mining process and will return correct results. Wong et al. [14] designed an audit environment in which the data owners add artificial data items that originally did not exist in the outsourced database, where data owners can verify the correctness and integrity of the mining results by checking the integrity of artificial data items. However, Ref. [14] does not change the frequency of data items in the original database, which allows the cloud server with some background knowledge to distinguish the real data items from the artificial data items by frequency analysis. Chen et al. [21] applied the BLS signature [22] to verify data integrity and ensure data reliability, which requires more computation costs in terms of checking the integrity than that in [14]. Furthermore, in their scheme, data are encrypted with the homomorphic encryption for higher security, which leads to a larger computation cost and an additional cloud server for auxiliary decryption. In this paper, we improve the technique proposed by [14] to realize an efficient PPFIM protocol, which provides the integrity verification and resists frequency analysis attacks. As shown in Table 1, we compare the contribution of our protocol with the verifiable FIM protocols presented in [14,21].

Table 1. Comparison of contributions.

Scheme	**[14]**	**[21]**	**Ours**
Verifiability	√	√	√
Data Privacy	×	√	√
Against frequency analysis attack	×	√	√
Number of third-party cloud servers	2	3	2
Data encryption	–	Homomorphic encryption	Substitution cipher Homomorphic encryption

3. Preliminaries

In this section, we introduce the building blocks that are applied to our protocol, including the Paillier homomorphic encryption [16], an improved secure comparison protocol (SCP) based on [13,16], the substitution cipher, and some definitions of FIM [14].

3.1. Paillier Homomorphic Encryption

We recall the Paillier homomorphic encryption [16] as below. Note that the Paillier encryption is an additively homomorphic encryption scheme.

Paillier.KeyGen: Taking two large prime numbers p, q as input, it computes $N = pq$ and $\lambda = lcm(p-1, q-1)$, where $lcm(\cdot)$ denotes the least common multiple functions. It selects a random number $g \in \mathbb{Z}^*_{N^2}$ that satisfies $gcd(L(g^\lambda \mod N^2), N) = 1$, where $gcd(\cdot)$ denotes the greatest common divisor function and $L(x) = (x-1)/N$ with $x \in \mathbb{Z}_{N^2}$ and $x \equiv 1 \mod N$. It outputs the public key $pk = \{N, g\}$ and the private key $sk = \lambda$.

Paillier.Enc: Taking as input a message $m \in \mathbb{Z}_N$, it selects a random number $r \in \mathbb{Z}^*_{N^2}$ and computes the ciphertext as $c = [m]_{pk} = g^m r^N \mod N^2$.

Paillier.Dec: Taking as input a ciphertext c, it computes the message with the private key λ as $m = \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N$.

Paillier.Add: Taking as input two ciphertexts $[m_1]_{pk}$ and $[m_2]_{pk}$, we have

$$[m_1]_{pk} \cdot [m_2]_{pk} = [m_1 + m_2]_{pk}.$$

This is because

$$\begin{aligned}&[m_1]_{pk} \cdot [m_2]_{pk} = g^{m_1} r_1^N \mod N^2 \cdot g^{m_2} r_2^N \mod N^2\\ =&g^{m_1+m_2} r_1 + r_2{}^N \mod N^2\\ =&[m_1 + m_2]_{pk}.\end{aligned}$$

Paillier.Mult: With the additive homomorphic property, we can easily have that given a ciphertext $[m_1]_{pk}$ and a message $m_2 \in \mathbb{Z}_N$,

$$([m_1]_{pk})^{m_2} = [m_1 \cdot m_2]_{pk}.$$

In our protocol, we use Paillier encryption to encrypt 0 and 1, and the corresponding ciphertexts are $r^N \mod N^2$ and $gr^N \mod N^2$, where r is the randomly chosen number, $N = pq$, and g is one of the public key pk of the user.

3.2. Substitution Cipher

Substitution cipher protects the privacy of a single unit but it does not change the frequency of items. This means that every ciphertext and the corresponding item appear at the same frequency. As such, the substitution cipher is vulnerable to frequency analysis attacks. If attackers have some knowledge of the frequencies of the items in raw transactions, they can recover the plaintext by frequency analysis. For example, an attacker knows that bread is the most sold commodity and milk is the second. Once they acquire the transactions database encrypted by substitution cipher, they can find the ciphertexts for bread and milk by counting the frequencies of ciphertexts. We counter frequency analysis by adding fictitious transactions to hide the accuracy frequency of items.

3.3. FIM

We first recall the definition of FIM [23]. Let $I = \{i_1, i_2, \cdots, i_n\}$ be the item domain and a transaction database be $D = \{t_1, t_2, \cdots, t_n\}$, where a transaction t_i is a subset of I. We say that a transaction t_i contains an itemset x if and only if $x \subseteq t_i$. Given a transaction database D, the support of itemset x, denoted by $supp(x)$, is the number of transactions that the itemset x contains in D. Given a support threshold $s\%$, the itemset x is a frequent itemset if and only if $supp(x) > |D| \times s\%$, where $|D|$ is the total number of transactions in D. We also refer to $|D| \times s\%$ as $supp_{min}$. FIM serves to find all frequent itemsets in a given database D.

In our protocol, artificial transactions are inserted into data owners' transaction datasets to verify the integrity of the mining results [14]. To better understand this, we recall the related definitions and theorems below. First of all, a correct mining result must be a valid return.

Definition 1. *(Valid Return) Given a returned FIM result R from the miner, we said that R is valid if* $\forall y \in R, \forall x \subset y$ *and* $x \neq \emptyset \Rightarrow x \in R$ *and* $supp(x) \geq supp(y)$*. We refer to this property as the "monotonicity property" which states that any subset of a frequent itemset must be frequent.*

Then, we introduce the concepts of a positive border and a negative border. Given an item domain I, let L be a set of frequent itemsets that satisfy the monotonicity property.

Definition 2. *(Positive Border) The positive border of L, denoted by* $B^{+}(L)$*, is the set of all frequent itemsets with a maximal length in L, i.e.,* $B^{+}(L) = \{x \mid \forall x \in L \text{ and } \forall x \subset y, y \notin L\}$*.*

Definition 3. *(Negative Border) The negative border of L, denoted by* $B^{-}(L)$*, is the set of all infrequent itemsets with minimal length with respect to L, i.e.,* $B^{-}(L) = \{x \mid \forall x \subset I \text{ and } x \notin L \text{ and } \forall y \subset x, \text{ where } y \neq \emptyset,\ y \in L\}$*.*

Theorem 1. *Let a set of frequent itemsets* L' *be a valid return from the miner while the real set of frequent itemsets is L. No infrequent itemset is inserted into L if and only if all itemsets in* $B^{+}(L')$ *are frequent. No frequent itemset is deleted from L if and only if all itemsets in* $B^{-}(L')$ *are infrequent. Proof of this theorem can be found in [14].*

The inserted artificial itemsets should satisfy the following requirements.

Definition 4. *(Valid Pattern) Let I be a set of unique items in the transaction database D and* I_A *be a set of artificial items. Assume that* $I_A \cap I = \emptyset$*. Given the support threshold s%, we generate a set of artificial frequent itemsets named* AFI *and a set of artificial infrequent itemsets named* AII*. We say that the itemset pattern* (AFI, AII) *is a s-valid pattern if there exists a transaction database* T_A *in which all itemsets in* AFI *are frequent and all itemsets in* AII *are infrequent with the support threshold s%.*

When generating artificial transactions, we insert itemsets belonging to AFI into transactions and try to avoid inserting itemsets belonging to AII. However, if a transaction contains multiple itemsets of AFI, it may contain one or more itemsets of AII. Such a situation is called a "conflict" between the itemsets in AFI. If such "conflicts" never occur in the same transaction, the itemsets of AII will not be included in any transaction.

Definition 5. *(Conflict in* AFI*) Let* x_i, x_j *be two itemsets belonging to* AFI*,* $i \neq j$*. We say that* x_i *conflicts with* x_j *if and only if there exists* $z \in AII$ *satisfying* $(z - x_i) \cap x_j \neq \emptyset$ *and* $(z - x_j) \cap x_i \neq \emptyset$*. If a transaction t of the artificial database* T_A *contains both* x_i *and* x_j *while* x_i *conflicts with* x_j*, the corresponding itemset z may be included in transaction t.*

Definition 6. *(Conflict Index) Let graph* $G = (V, E)$ *represent the conflict relationships of itemsets in* AFI*. Each node in graph G represents an itemset in* AFI*. An edge* (n_1, n_2) *in graph G represents the conflict between* n_1 *and* n_2*. The conflict index of node n, denoted by* c_n*, is the number of neighbor nodes of n, i.e., the number of itemsets in* AFI *in conflict with n. The conflict index of graph G, denoted by* $CI(G)$*, is equal to the maximum value of the conflict index of nodes in G, i.e.,* $CI(G) = max_{n \in V} c_n$*.*

Theorem 2. *Given an artificial itemset pattern (AFI, AII) while AFI and AII satisfy the monotonicity, (AFI, AII) is an $s-valid$ pattern if $CI(G) \leq \frac{1}{s\%} - 1$. Proof of this theorem can be found in [14].*

4. Models and Design Goals

In this section, we depict the system model, security model, and design goals of our protocol.

4.1. System Model

As shown in Figure 1, our system model consists of two cloud servers (CSP and Evaluator) and multiple data owners.

- *CSP* integrates the vertically partitioned databases from different data owners to generate a joint database. With the help of the Evaluator, CSP mines frequent itemsets in the joint database and returns the mining results to the relevant data owners.
- *Evaluator* is responsible for generating and distributing public/private key pairs (pk, sk) of Paillier encryption for data owners. Additionally, Evaluator assists the CSP in performing FIM.
- *Data owners* sell different goods. They take commodities as items and transform customers' purchase records in the store into digital transactions. By generating a unique identifier TID for each transaction according to the user's bank card, contact information, and other related information, each data owner finally obtains a private transaction database. After inserting artificial data items and fictitious transactions into their own transaction database, data owners encrypt it and then send it to the CSP for PPFIM. It is worth noting that there is no intersection between itemsets among different data owners' databases.

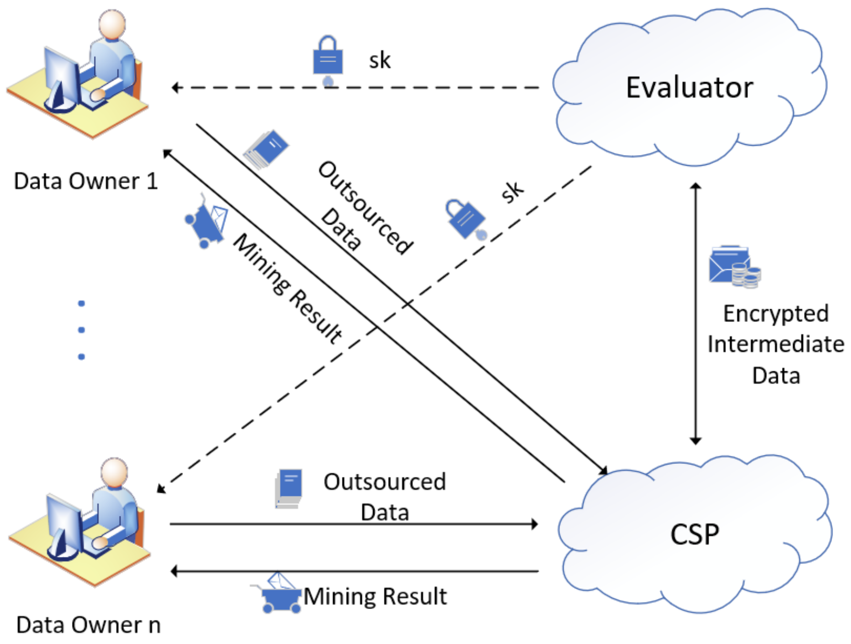

Figure 1. System model.

4.2. Security Model

In our system, we assume the following.

- The CSP is dishonest and curious. Due to saving costs in terms of computation, making illegal profits, or unexpected errors, the CSP may behave dishonestly or erroneously by returning inaccurate mining results to data owners. In addition to that, the CSP is curious about the original transaction information and mining results.

- The Evaluator is honest but curious. If the Evaluator is dishonest, the user can still detect errors in the mining results and confirm whether the CSP or Evaluator performs dishonestly, which is the same for data owners since cloud servers are dishonest. We therefore assume that the Evaluator is honest and curious. The CSP and Evaluator are set to not collude, or else the cloud servers can extract all the underlying messages of encrypted data.
- Data owners are set to not collude with the CSP. In a vertically partitioned data mining scenario, data owners cooperate to send an encrypted transaction database to the CSP to mine frequency itemsets. They ultimately benefit from the correct mining results. Therefore, data owners are not supposed to collude with the CSP.

Note that each data owner should know the items in other owners' transaction databases but have limited knowledge of the TIDs therein in practical applications. Our protocol is more meaningful when there are many common TIDs in data owners' databases, such as those for stores in the same community.

4.3. Design Goals

We design a verifiable privacy-preserving outsourced FIM protocol on vertically partitioned databases to ensure the following:

- *Security of private information.* The original transactions of the encrypted database should be kept secret from the CSP and Evaluator. Data owners should also learn as little information as possible about other owners' databases. The support and threshold should be concealed as they may be used to infer information about raw databases [24]. The CSP and Evaluator should not obtain accurate mining results. Each data owner can only obtain frequent itemsets that contain items in their own transaction database.
- *Verifiability of the integrity of mining results*. Data owners should be able to verify the integrity of mining results, which means that data owners can detect inaccurate results if the CSP performs dishonestly in the mining.

5. PPFIM Protocol

Our protocol can be divided into three phases. First, data owners insert artificial transactions into their own private transaction databases to achieve verifiable integrity in their mining results. Then, data owners process and encrypt their transactions and send the encrypted databases to the CSP. Then, CSP integrates the databases from multiple data owners to generate a joint database and performs FIM on it. Data owners can finally obtain encryption-related mining results from the CSP.

5.1. Artificial Transaction Insertion

For a data owner, given its private transaction database T and support threshold $s\%$, these generate a set of artificial frequent itemsets AFI and a set of artificial infrequent itemsets AII. Upon the corresponding artificial database T_A, the data owner merges T_A and T into a new database D. We divide this process into two main steps: artificial itemset generation and artificial transaction generation.

5.1.1. Artificial Itemset Generation

We adopt the scheme proposed in [14] to generate the artificial frequent itemsets AFI and artificial infrequent itemsets AII that satisfy the requirement of an s-valid pattern. The execution process of artificial itemset generation can be seen in Figure 2 and the details are as follows.

Note that the itemsets belonging to AFI and AII should be satisfied with the demand for the monotonicity property and the conflicting itemsets in AFI jeopardize correctness. We describe the conflicting relationship among AFI by a conflict graph G. (AFI, AII) should be an s-valid pattern (see Theorem 2) such that the data owner could generate the corresponding artificial database $T_A = \{t_1, t_2, \cdots, t_n\}$ in which each itemset in AFI is frequent and each itemset in AII is infrequent.

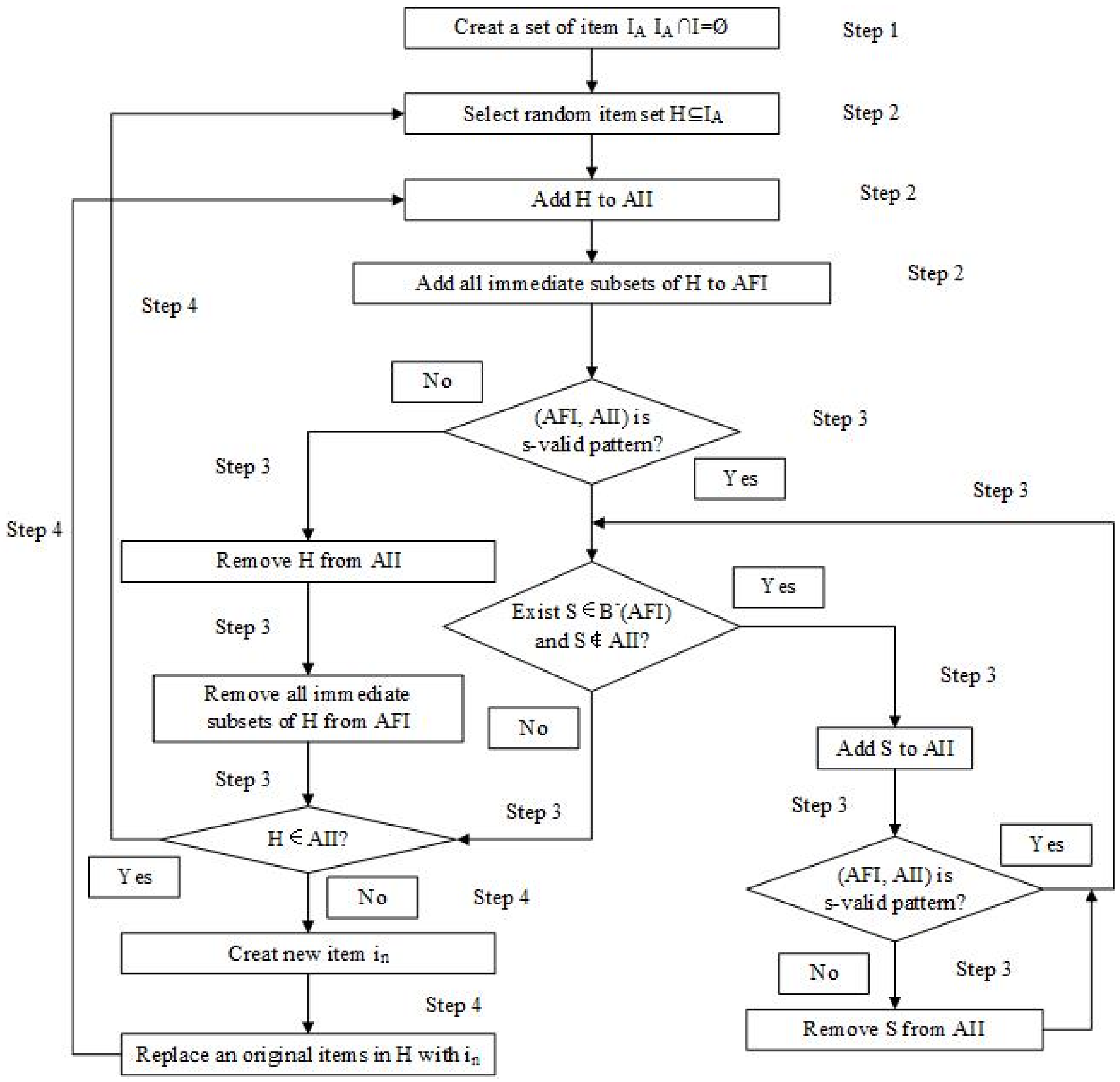
Creat a set of item I_A I_A ∩I=∅
Step 1
Select random itemset H⊆I_A
Step 2
Add H to AII
Step 2
Add all immediate subsets of H to AFI
Step 2
Step 4
No
(AFI, AII) is s-valid pattern?
Step 3
Yes
Step 3
Step 3
Step 4
Remove H from AII
Exist S ∈B⁻(AFI) and S ∉ AII?
Yes
Step 3
Remove all immediate subsets of H from AFI
No
Step 3
Add S to AII
Step 3
Step 3
Step 3
H ∈ AII?
Yes
(AFI, AII) is s-valid pattern?
Yes
No
Step 4
Creat new item i_n
No
Step 3
Step 4
Remove S from AII
Replace an original items in H with i_n

Figure 2. Artificial Itemset Generation.

Step 1: The data owner creates a set of artificial items I_A satisfying $I_A \cap I = \emptyset$, where I represents the item domain of T. The size of AII is denoted as the initial size of I_A.

Step 2: The data owner randomly selects an itemset $H \subseteq I_A$ satisfying $H \notin AII$. It then adds H to AII and all immediate subsets of H to AFI. For example, if $H = ABC$, the data owner adds $\{ABC\}$ to AII and $\{AB, BC, AC\}$ to AFI.

Step 3: The data owner updates the conflict graph G of AFI. If the obtained (AFI, AII) is not an s-valid pattern, it rolls back the addition operation performed in step 2 and goes to step 4. If (AFI, AII) is an s-valid pattern, the data owner calculates $B^-(AFI)$. If there exists an itemset S satisfying $S \in B^-(AFI), S \notin AII$, it adds S to AII. Otherwise, go to step 4. After the addition, the data owner updates G and checks whether (AFI, AII) is an s-valid pattern. If not, it rolls back the insertion.

Step 4: If H is successfully added to AII, repeat steps 2 and 3 until the sizes of AFI and AII meet the requirements. Otherwise, $H \notin AII$, the data owner creates a new artificial item i_n, adds i_n to I_A, replaces an original item in H with i_n, and tries to add H to AII again (step 2).

Suppose the size of H is k, then after a maximum of k attempts, H will be replaced with a completely new itemset which must be successfully inserted into AII. Thus, the algorithm can be successfully executed in finite time. The procedure tries to add itemsets to AFI and AII until they are both big enough. We refer to Section 6 for the relationship between the size of (AFI, AII) and verification accuracy.

5.1.2. Artificial Transaction Generation

In this phase, the data owner generates the artificial transaction database T_A corresponding to (AFI, AII). Recalling the requirements of the artificial database that each itemset of AFI is frequent in T_A and each itemset of AII is infrequent in T_A, we tend to add the AFI itemsets into the transactions without inserting any AII itemsets to avoid jeopardizing correctness. In other words, conflicting AFI itemsets should not be added to the same transaction. We depict the artificial transaction generation process in Figure 3 and the details are as follows.

Step 1: The data owner generates an array f_{AFI}, the size of which is set to be equal with that of AFI. $f_{AFI}[i]$ records the number of times that the i-th itemsets in AFI is added to the artificial transactions.

Step 2: The data owner generates artificial transactions by adding non-conflicting AFI itemsets. It maintains two initially empty sets S^+, S^-, randomly selects an itemset $U \in AFI$, adds U to S^+, and then adds all neighbor nodes of U in the conflict graph G to S^-. Recall that the neighbor nodes of U in G represent itemsets that are in conflict with U.

Step 3: Repeat step 2 until $AFI = S^+ \cup S^-$. Itemsets in S^+ are conflict-free and each itemset in S^- conflicts with at least one itemset in S^+. The first transaction to be generated is the union of all itemsets in S^+. Update f_{AFI}.

Step 4: Remove all itemsets existing in S^+ from AFI and remove all corresponding nodes and related edges in graph G to obtain the conflict graph G' corresponding to S^-. Since each itemset in S^- conflicts with at least one itemset in S^+, $CI(G') \leq CI(G) - 1$.

Step 5: Set $G = G'$, repeat steps 2–4 until $CI(G') = 0$. Take the itemsets left in AFI as an artificial transaction since they are conflict-free. A maximum of $CI(G) + 1$ transactions can be generated in this process.

Step 6: If all values in f_{AFI} are greater than the support threshold, i.e., $f_{\mathsf{AFI}}[i] > \mathsf{Supp}_{min}$, return all artificial transactions as the artificial database T_A. Otherwise, reset the set U and graph G and return to step 2. The data owner merges the original database T with T_A to obtain a new database D. For the following processing and mining, we give an example of database D in Table 2.

Table 2. Data Owners' Transaction Databases.

Alice's Database		**Bob's Database**	
TID	Transaction	TID	Transaction
1	A B C	2	E F
2	A B	3	D E
4	B C	6	F

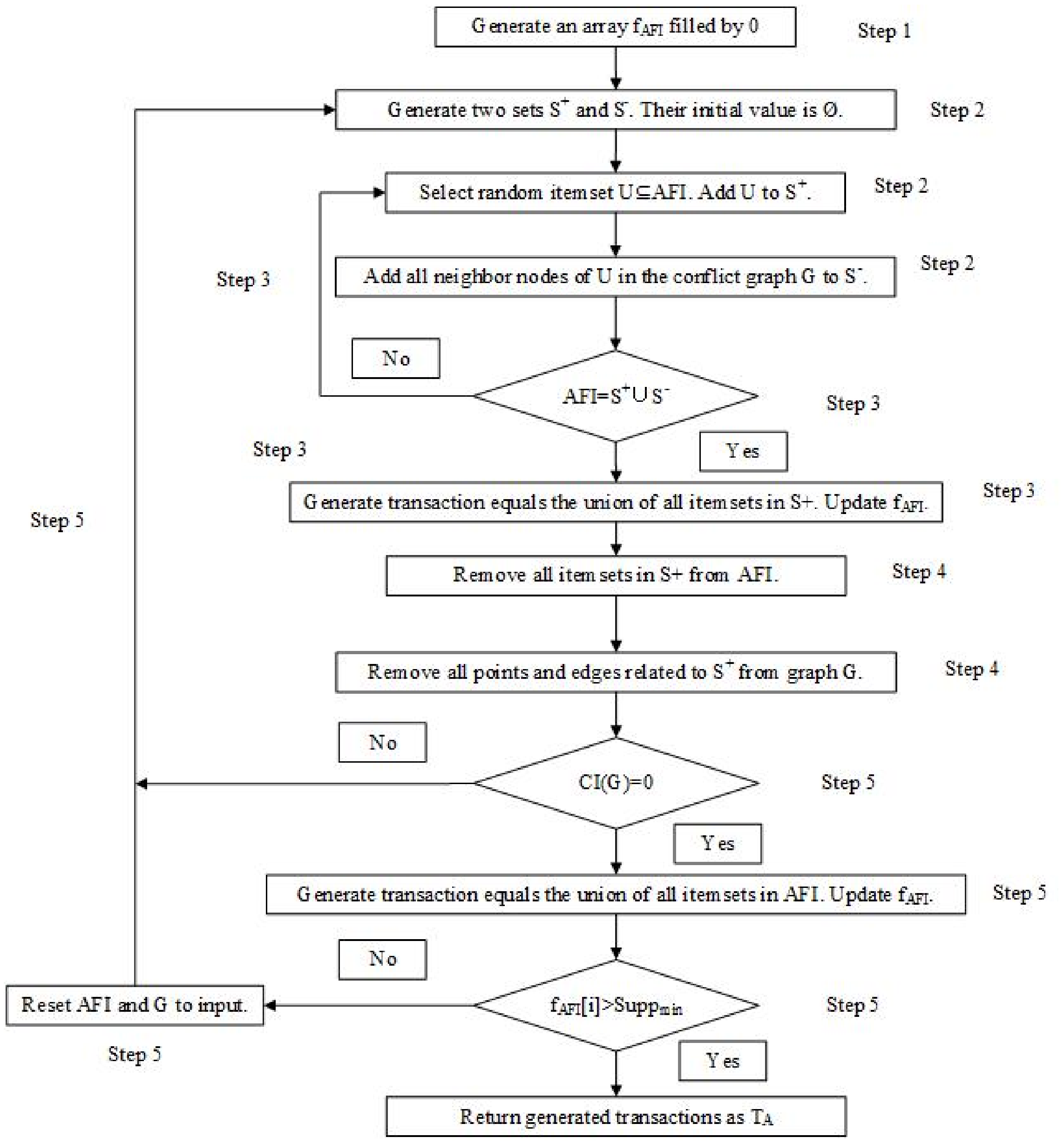
Generate an array f_AFI filled by 0
Step 1
Generate two sets S+ and S-. Their initial value is Ø.
Step 2
Select random itemset U⊆AFI. Add U to S+.
Step 2
Add all neighbor nodes of U in the conflict graph G to S-.
Step 2
Step 3
No
AFI=S+∪S-
Step 3
Step 3
Yes
Generate transaction equals the union of all itemsets in S+. Update f_AFI.
Step 3
Step 5
Remove all itemsets in S+ from AFI.
Step 4
Remove all points and edges related to S+ from graph G.
Step 4
No
CI(G)=0
Step 5
Yes
Generate transaction equals the union of all itemsets in AFI. Update f_AFI.
Step 5
No
Reset AFI and G to input.
f_AFI[i]>Supp_min
Step 5
Step 5
Yes
Return generated transactions as T_A

Figure 3. Artificial Transaction Generation.

5.2. Encryption of Databases

In this phase, each data owner encrypts their own transaction database.

Step 1: The Evaluator generates a public/private key pair (pk, sk) of Paillier encryption and sends sk to data owners.

Step 2: In order to counter frequency analysis attacks, we adopt the algorithm proposed in [15] to insert fictitious transactions into database D to obtain a database Z. Note that fictitious transactions are different from artificial transactions, where fictitious transactions are inserted to resist frequency analysis attacks and artificial transactions are inserted to realize the verifiability of the integrity of mining results. In Z, each item has the same frequency of occurrence as at least $k-1$ other items, where k is a number negotiated by data owners.

Step 3: In order to eliminate the influences of fictitious transactions on the mining results, each data owner tags each transaction in Z with an encrypted realness value ERV. The underlying realness values RV of ERV for a transaction in D is 1 and that of a fictitious transaction is 0, which means that $ERV \in \{[1]_{pk}, [0]_{pk}\}$. Since Paillier encryption is a probabilistic encryption cryptosystem, the CSP cannot distinguish the original and fictitious transactions from ERV.

Step 4: With a cryptographic hash function H, data owners replace the transaction's unique identifier TID with $H(TID)$. Each data owner then encrypts the items in database Z according to a substitution alphabet. Note that data owners use different private substitution alphabets, which do not influence the mining since the items belonging to different owners are different. An example of the encrypted databases is shown in Table 3.

Step 5: Each data owner outsources the encrypted database to the CSP.

Table 3. Data Owners' Encrypted Databases.

	Alice's Database			**Bob's Database**	
TID	Transaction	ERV	TID	Transaction	ERV
H(1)	S_A(A) S_A(B) S_A(C)	[1]	H(2)	S_B(E) S_B(F)	[1]
H(2)	S_A(A) S_A(B)	[1]	H(3)	S_B(D) S_B(E)	[1]
H(3)	S_A(A)	[0]	H(4)	S_B(D)	[0]
H(4)	S_A(B) S_A(C)	[1]	H(6)	S_B(F)	[1]
H(5)	S_A(C)	[0]			

$H(\cdot)$: Hash value of original TID. $S(\cdot)$: The item encrypted by substitution cipher $[\cdot]$: RV encrypted by Paillier.

5.3. Frequent Itemset Mining

The frequent itemset mining is shown below.

Step 1: Data owners negotiate a shared support threshold $supp_{min}$ and send the encrypted support threshold $T_s = [supp_{min}]_{pk}$ to the CSP.

Step 2: The CSP aggregates all data owners' databases based on $H(TID)$ to generate a joint database, where the transaction corresponding to a certain $H(TID)$ consists of that related to $H(TID)$ in all owners' databases. Table 4 gives the joint database of databases listed in Table 3.

Table 4. Joint Database.

TID	**Alice's Partition**	**Bob's Partition**	**Alice's ERV**	**Bob's ERV**
H(1)	S_A(A) S_A(B) S_A(C)	NULL	[1]	NULL
H(2)	S_A(A) S_A(B)	S_B(E) S_B(F)	[1]	[1]
H(3)	S_A(A)	S_B(D) S_B(E)	[0]	[1]
H(4)	S_A(B) S_A(C)	S_B(D)	[1]	[0]
H(5)	S_A(C)	NULL	[0]	NULL
H(6)	NULL	S_B(F)	NULL	[1]

Step 3: The CSP runs the classical FIM algorithm Eclat to find all frequent itemsets in the joint database. The Eclat algorithm outputs a set of $H(TID)$ that is related to the frequent itemsets in the mining result. It is worth noting that since we add artificial transactions to databases, some itemsets with less support than the support threshold are mistakenly returned as frequent itemsets. In other words, the frequent itemsets returned by the Eclat algorithm are only "seemingly frequent", which obviously contains all real frequent itemsets.

Step 4: For a "seemingly frequent" itemset S, suppose that its actual support is $supp(S)$. As mentioned before, the Eclat algorithm will return the $H(TID)$ set of S-related transactions $T(S)$. The items contained in S may come from multiple data owners, and we denote the set of these users as $D(S)$. In other words, the items contained in S all come from the data owner in $D(S)$, while each data owner has at least one item in itemset S. The i-th transaction i in $T(S)$ consists of transactions from one or more data owners, and we denote the set of these owners as $D(S,i)$. Let $ERV_{i,j}$ represent the ERV of the j-th data owner's partition in the i-th transaction. The CSP then computes the encrypted support of S as

$$[supp(S)]_{pk} = \prod_{i \in T(S)} \left[SC\left(\prod_{j \in D(S,i)} ERV_{i,j}, |D(S,i)| \right) \right],$$

where the secure comparison protocol $SC\left(\sum_{j \in D(S,i)} ERV_{i,j}, |D(S,i)| \right)$ will output a ciphertext of 1 if and only if every data owner in $|D(S,i)|$ has a real partition in the i-th transaction and it outputs 0 otherwise.

Step 5: For a "seemingly frequent" itemset S, the CSP computes the encrypted frequent realness value of S as

$$EFRV(S) = SC\left([supp(S)]_{pk}, T_s \right), \tag{1}$$

where $T_s = [supp_{min}]$ is the encrypted support threshold. If S is a real frequent itemset, $EFRV(S)$ is the ciphertext of 1. Otherwise, it is a ciphertext of 0. CSP sends every "seemingly frequent" itemset S and $EFRV(S)$ to all the data owners in $D(S)$.

Step 6: Data owners decrypt the received $EFRV$ of the "seemingly frequent" itemsets with sk to determine the real frequent itemsets. Each data owner can extract the related itemsets in its partition associated with the real frequent itemsets using the substitution cipher.

Step 7: Data owners verify the integrity of the mining results by checking: (1) whether the CSP returned any itemsets in AII; and (2) whether the CSP returned all the itemsets in AFI and their subsets. If (1) happens or (2) does not happen, owners can determine whether dishonest behaviors have occurred.

Step 8: Although data owners cannot directly obtain the related frequent itemsets of other owners, they can know which owners share the same frequent itemset. Data owners can interact with each other to line up willing partnerships and then share the real related itemsets with each other for the next commercial cooperation.

6. Performance Evaluation

6.1. Security Analysis

We analyze the security of this protocol against the CSP and Evaluator below. In our model, the CSP is set to be dishonest and curious, such that it will return incorrect mining results and try to extract the information of data owners' original transactions and the underlying mining results. For the misbehaviors of the CSP, the correctness and security of adding artificial itemsets protect the successfully tracing of the CSP.

In our protocol, the items of the transactions in the database are protected by a substitution cipher that is more secure but vulnerable to frequency analysis attacks. Furthermore, to further resist this type of attack, we apply the technique proposed in [15] to hide an item into at least a set of k items with the same frequency. The security of the technique was proven in [15]. In this protocol, data owners attach an ERV that is a Paillier ciphertext of 1 or 0 to each transaction, which obviously does not help the CSP distinguish the items through the indistinguishability of Paillier encryption. In conclusion, the security of adding artificial itemsets, hiding itemset techniques, and Paillier encryption prevent the CSP from extracting data owners' original transactions.

Next, regarding the security of mining results, we first prove the security of the improved secure comparison protocol below. Suppose that x_i is the input of party P_i, y_i is the input of party P_i, and $\Pi_i(\pi)$ is P_i's execution image of protocol π. We say that the protocol π is secure if $\Pi_i(\pi)$ can be simulated from x_i and y_i such that the distribution of the simulated image $\Pi_i^s(\pi)$ is computationally indistinguishable from $\Pi_i(\pi)$. The Evaluator's execution image of the SC protocol is denoted as $\Pi_{Evaluator}(SC) = \{(w, W), d\}$ where w is the decryption of W. If $t = 0$, then $w = r_1 \cdot (x - y) + r_2$, and if $t = 1$, then $w = r_1 \cdot (y - x) + r_2$. Recall that t is a random number from $(0, 1)$ and r_1, r_2 are randomly selected from $\{1, \cdots, 2^l\}$. In addition, d is the comparison result from w. We assume that the simulated image of the Evaluator is $\Pi^s_{Evaluator}(SC) = \{(w', W'), d'\}$ where (w', W') are randomly selected from $\mathbb{Z}_N$ and d' is a random number from $(0, 1)$. As Paillier is proven to be semantically secure, (w, W) is computationally indistinguishable from (w', W'). Furthermore, t is randomly selected from $(0, 1)$, so d and d' are either 0 or 1 with equal probability. Thus, d is computationally indistinguishable from d'. In summary, the $\Pi_{Evaluator}(SC)$ is computationally indistinguishable from $\Pi^s_{Evaluator}(SC)$. Similarly, $\Pi_{CSP}(SC) = \{X, Y, Z, W, [d]_{pk}, [e]_{pk}\}$ is the execution image of CSP. The simulated image of CSP is $\Pi^s_{CSP}(SC) = \{X', Y', Z', W', \alpha, \beta\}$, where $\{X', Y', Z', W', \alpha, \beta\}$ are random numbers from $\mathbb{Z}_N$. Since Paillier is semantically secure, $\{X', Y', Z', W', \alpha, \beta\}$ are computationally indistinguishable from $\{X, Y, Z, W, [d]_{pk}, [e]_{pk}\}$. Thus, we can claim that $\Pi^s_{CSP}(SC)$ is computationally indistinguishable from $\Pi_{CSP}(SC)$. As a result, we can conclude that the secure comparison protocol is secure.

The CSP can obtain the mining results of the Eclat mining algorithm, the encrypted real support of itemsets, and the encrypted frequent realness value of itemsets. However, since the additional fictitious transactions, the mining results of the Eclat algorithm are actually "seemingly frequent" itemsets that contain infrequent itemsets. Furthermore, the itemsets are also encrypted with a substitution cipher. Due to the security of the secure comparison protocol and Paillier encryption, the CSP can only obtain a set of Paillier ciphertexts without knowing the underlying messages such that it also cannot extract the mining results.

6.2. Experimental Analysis

In this section, we implement our protocol and give the analysis in terms of verification accuracy and computational complexity. Our simulation experiments are performed on a computer with a 64-bit Windows 10 system, an Intel(R) Core(TM) i3-8100 CPU @ 3.60 GHz, and 4.00 GB of RAM. The Paillier algorithm is implemented in Java with a large integer N a length of 1024 bits and a safety parameter λ of 80 bits.

Verification Accuracy. We generate 10 random databases with the same set of data items [25]. The number of data items in each database is 1000, represented by integers from 1 to 1000. The average length of transactions is 10, and the number of transactions in the databases varies from $10,000$ to $100,000$. For simplicity, the frequent itemsets in these transaction databases are identical, but their support is different. This means that the support thresholds of these databases are different, while the frequent itemsets in mining results are identical. The exact mining result is denoted as L, the number of artificial frequent itemsets is $|AFI| = v \cdot |L|$, and the number of artificial infrequent itemsets is $|AII| = v \cdot |B^-(L)|$. We assumed that the cloud performs $e \cdot (|L| + |B^-(L)|)$ dishonest behavior to the mining results, and the probability that the data owner captures the dishonest behavior of the cloud is p. We randomly delete, add, or replace the frequent itemsets in the mining results and repeat the experiment 1000 times for each database, counting the average probability that these modifications to the experimental results are captured. The experimental results are shown in Figure 4.

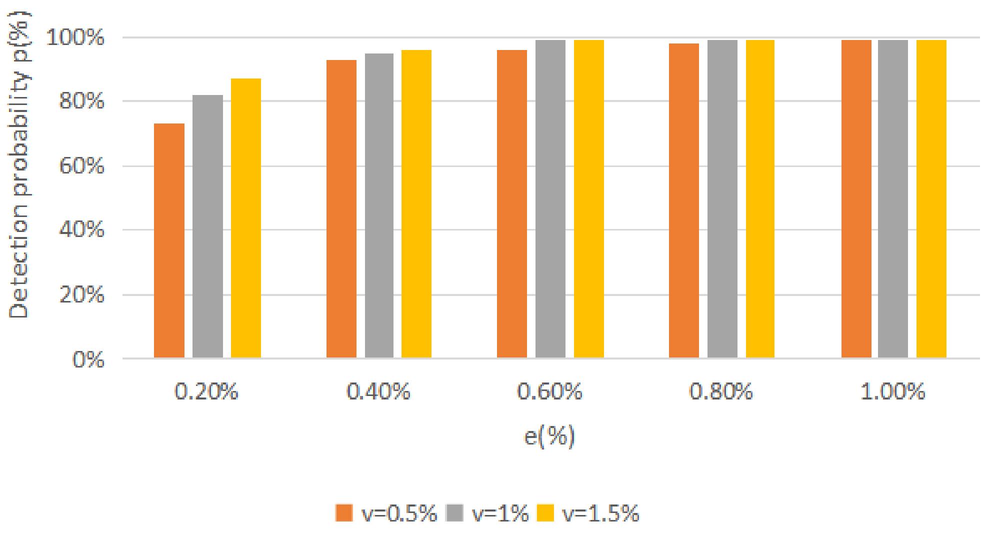

Figure 4. Probability of detecting the dishonest CSP.

As shown in Figure 4, the verification accuracy increases along with the increase in the number of dishonest behaviors in the cloud and the proportion of artificial itemsets. As e represents the proportion of itemsets affected by the CSP's dishonest behaviors and v represents the proportion of artificial itemsets in the mining results which reaches 1%, the data owner can capture dishonest behaviors from the CSP with a probability close to 100%.

Computational complexity. We evaluate the computational complexity of our protocol by computation costs. To be realistic, we test the performance of our solution using the retail database from [25] for mining. To simulate t data owners, we partition the retail database into t vertically partitioned databases. In this experiment, we fix t to 2 and the proportion of artificial itemsets to the mining results to 1%. In our experiment, we assume that the data owners and the two cloud servers are run with identical hardware conditions. We set different support thresholds for PPFIM with our scheme for the retail database. The running time of the Eclat algorithm and the number of frequent itemsets in mining results are shown in Table 5. We also measure the running time of the data owner, CSP, and Evaluator during the FIM of the retail dataset under different thresholds, as shown in Figure 5.

Table 5. Mining result and running time of Eclat.

Support Threshold	200	500	1000	2000	4000
Number of frequent itemsets	2191	468	135	45	16
Running time of Eclat	2319 ms	1139 ms	857 ms	970 ms	834 ms

As shown in Table 5, the number of frequent itemsets significantly decreases as the threshold value increases. The computation cost of the CSP and Evaluator also decreases along with the increase in support threshold, i.e., the reduction in frequent itemsets in the mining results, as depicted in Figure 5. This can be naturally deduced since more frequent itemsets also imply more operations in mining. Taking Table 5 and Figure 5 together, we have that the vast majority of the computation costs are spent on ciphertext-based operations for the CSP and Evaluator and on encryption and decryption operations for both the data owner and the Evaluator. A secure and more efficient homomorphic encryption algorithm can effectively improve the execution efficiency of our protocol.

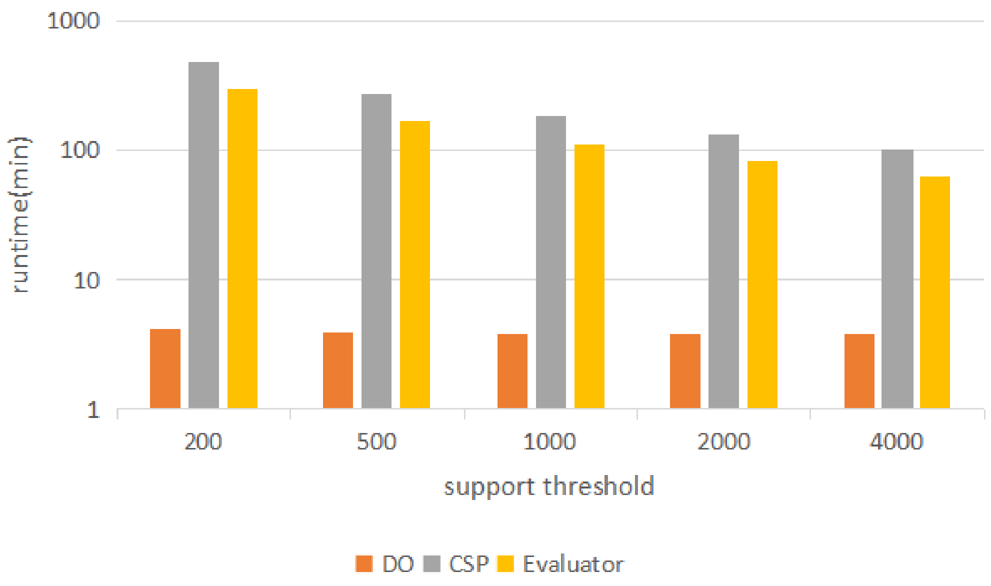

Figure 5. Computational costs of DO, CSP, and Evaluator.

We compare our protocol with the privacy-preserving frequent mining protocols [12,13], which also support multi-user settings [11] in terms of functions, as shown in Table 6. Note that the protocol of [11] that we based ours upon has been proven to be insecure since the security of the applied symmetric homomorphic encryption scheme therein is completely broken [26]—we omit the related comparison in this table. As shown, only our protocol can resist the malicious cloud server, i.e., realizing the verifiability of the integrity of the mining results. In [12], the Evaluator who also owns the private key as our protocol can finally know the final result of the secure comparison protocol and whether a mining itemset is frequent, which obviously leaks the privacy of mining results. This problem was solved in [13] and in our scheme. In addition, the protocols in [12,13] performed mining on horizontally partitioned databases and our protocol is performed on vertically partitioned databases.

Table 6. Comparison of functions.

Scheme	**[11]**	**[12]**	**[13]**	**Ours**
FIM	$\surd$	$\surd$	$\surd$	$\surd$
Data privacy	Partial	$\surd$	$\surd$	$\surd$
Verifiability of mining result integrity	$\times$	$\times$	$\times$	$\surd$
Mining result privacy protection	$\surd$	$\times$	$\surd$	$\surd$
Offline	$\times$	$\surd$	$\surd$	$\surd$

7. Conclusions

Based on the Paillier encryption, we propose a verifiable PPFIM protocol on vertically partitioned databases. Our protocol can provide privacy-preserving outsourced mining across multiple data owners with vertically partitioned databases and can further allow data owners to check the integrity of results that are returned from the cloud. The verifiability in the field of privacy-preserving frequent itemset mining is quite significant since the outsourced clouds are usually not fully trusted in practice. However, particular data owners can be offline after they upload their encrypted database to the cloud, which is convenient for data owners and is practical in real applications. As a trade-off, we add artificial transactions in the proposed protocol, which increases the size of the databases and the cost of computation. Experimental analysis shows that a 100% probability of verification accuracy in our protocol in addition to a 1% increased artificial itemset proportion is an acceptable trade-off for small databases. However, when dealing with a database with big data, such as tens of millions of data, the additional 1% increased itemsets will cause a large computational load. Therefore, to efficiently support the verification of the integrity of the mining results without increasing the database size and computation cost can be listed as our future work.

Author Contributions: Conceptualization, Z.Z. and L.L.; methodology, B.W.; software, L.L.; validation, Z.Z., B.W. and J.L.; writing—original draft preparation, Z.Z. and L.L.; writing—review and editing, B.W. and J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant Nos. 62102299, 62272362, and U19B2021, the Henan Key Laboratory of Network Cryptography Technology under Grant No. LNCT2022-A05, the Fundamental Research Funds for the Central Universities under Grant No. ZYTS23169, and the Youth Innovation Team of Shaanxi Universities.

Data Availability Statement: The data presented in this study are available on request from the corresponding authors. The data are not publicly available due to privacy requirements of the project.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Agrawal, R.; Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the VLDB'94, Santiago de Chile, Chile, 12–15 September 1994; pp. 487–499.
2. Han, J.; Pei, J.; Yin, Y.; Mao, R. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Min. Knowl. Discov.* **2004**, *8*, 53–87. [CrossRef]
3. Zaki, M.J. Scalable Algorithms for Association Mining. *IEEE Trans. Knowl. Data Eng.* **2000**, *12*, 372–390. [CrossRef]
4. Brijs, T.; Swinnen, G.; Vanhoof, K.; Wets, G. Using Association Rules for Product Assortment Decisions: A Case Study. In Proceedings of the SIGKDD, San Diego, CA, USA, 15–18 August 1999; pp. 254–260.
5. Brossette, S.E.; Sprague, A.P.; Hardin, J.M.; Waites, K.B.; Jones, W.T.; Moser, S.A. Research Paper: Association Rules and Data Mining in Hospital Infection Control and Public Health Surveillance. *J. Am. Med. Inform. Assoc.* **1998**, *5*, 373–381. [CrossRef] [PubMed]
6. Lee, W.; Stolfo, S.J. Data Mining Approaches for Intrusion Detection. In Proceedings of the USENIX, San Antonio, TX, USA, 26–29 January 1998.
7. Estan, C.; Varghese, G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.* **2003**, *21*, 270–313. [CrossRef]
8. Creighton, C.; Hanash, S. Mining gene expression databases for association rules. *Bioinformatics* **2003**, *19*, 79–86. [CrossRef] [PubMed]
9. Kantarcioglu, M.; Clifton, C. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1026–1037. [CrossRef]
10. Vaidya, J.; Clifton, C. Privacy preserving association rule mining in vertically partitioned data. In Proceedings of the SIGKDD, Edmonton, AB, Canada, 23–26 July 2002; pp. 639–644.
11. Li, L.; Lu, R.; Choo, K.R.; Datta, A.; Shao, J. Privacy-Preserving-Outsourced Association Rule Mining on Vertically Partitioned Databases. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1847–1861. [CrossRef]
12. 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.* **2020**, *8*, 312–323. [CrossRef]
13. Liu, L.; Su, J.; Chen, R.; Liu, X.; Wang, X.; Chen, S.; Leung, H. Privacy-Preserving Mining of Association Rule on Outsourced Cloud Data from Multiple Parties. In Proceedings of the ACISP, Wollongong, NSW, Australia, 11–13 July 2018; Volume 10946, pp. 431–451.
14. Wong, W.K.; Cheung, D.W.; Hung, E.; Kao, B.; Mamoulis, N. An Audit Environment for Outsourcing of Frequent Itemset Mining. *Proc. VLDB Endow.* **2009**, *2*, 1162–1172. [CrossRef]
15. Giannotti, F.; Lakshmanan, L.V.S.; Monreale, A.; Pedreschi, D.; Wang, W.H. Privacy-Preserving Mining of Association Rules From Outsourced Transaction Databases. *IEEE Syst. J.* **2013**, *7*, 385–395. [CrossRef]
16. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of the EUROCRYPT '99, Prague, Czech Republic, 2–6 May 1999; Volume 1592, pp. 223–238.
17. 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, Vienna, Austria, 23–27 September 2007; pp. 111–122.

18. Lamba, J.; Venkaiah, V.C. Privacy-preserving frequent itemset mining in vertically partitioned database using symmetric homomorphic encryption scheme. *Int. J. Inf. Priv. Secur. Integr.* **2020**, *4*, 203–225. [CrossRef]
19. Domadiya, N.H.; Rao, U.P. Privacy Preserving Approach for Association Rule Mining in Horizontally Partitioned Data using MFI and Shamir's Secret Sharing. In Proceedings of the ICIISIEEE, Piscataway, NJ, USA, 1–2 December 2018; pp. 217–222.
20. Ma, C.; Wang, B.; Jooste, K.; Zhang, Z.; Ping, Y. Practical Privacy-Preserving Frequent Itemset Mining on Supermarket Transactions. *IEEE Syst. J.* **2020**, *14*, 1992–2002. [CrossRef]
21. Chen, Y.; Zhao, Q.; Duan, P.; Zhang, B.; Hong, Z.; Wang, B. Verifiable privacy-preserving association rule mining using distributed decryption mechanism on the cloud. *Expert Syst. Appl.* **2022**, *201*, 117086. [CrossRef]
22. Boneh, D.; Lynn, B.; Shacham, H. Short Signatures from the Weil Pairing. In Proceedings of the ASIACRYPT, Gold Coast, Australia, 9–13 December 2001; pp. 514–532.
23. Agrawal, R.; Imielinski, T.; Swami, A.N. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the SIGMOD, Washington, DC, USA, 26–28 May 1993; pp. 207–216.
24. Rozenberg, B.; Gudes, E. Association rules mining in vertically partitioned databases. *Data Knowl. Eng.* **2006**, *59*, 378–396. [CrossRef]
25. Fournier-Viger, P.; Gomariz, A.; Gueniche, T.; Soltani, A.; Wu, C.; Tseng, V.S. SPMF: A Java open-source pattern mining library. *J. Mach. Learn. Res.* **2014**, *15*, 3389–3393.
26. Wang, B.; Zhan, Y.; Zhang, Z. Cryptanalysis of a Symmetric Fully Homomorphic Encryption Scheme. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1460–1467. [CrossRef]