Privacy Preserving Sequential Pattern Mining in Distributed Databases

Vishal Kapoor, Pascal Poncelet, François Trousset, Maguelonne Teisseire

▶ **To cite this version:**

Vishal Kapoor, Pascal Poncelet, François Trousset, Maguelonne Teisseire. Privacy Preserving Sequential Pattern Mining in Distributed Databases. CIKM: Conference on Information and Knowledge Management, Nov 2006, Arlington, Virginia, United States. pp.758-767, 10.1145/1183614.1183722 . lirmm-00135021

HAL Id: lirmm-00135021

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00135021

Submitted on 20 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Privacy Preserving Sequential Pattern Mining in Distributed Databases

V. Kapoor*
Netaji Subhas Institute of Technology
Azad Hind Fauj Marg
Dwarka, New Delhi-110045, India

P. Poncelet †
F. Trousset
EMA-LGI2P/Site EERIE
Parc Scientifique Georges Besse
30035 Nîmes Cedex, France

M. Teisseire‡
LIRMM UMR CNRS 5506
161 Rue Ada, 34392
Montpellier Cedex 5, France

ABSTRACT

Research in the areas of privacy preserving techniques in databases and subsequently in privacy enhancement technologies have witnessed an explosive growth-spurt in recent years. This escalation has been fueled by the growing mistrust of individuals towards organizations collecting and disbursing their Personally Identifiable Information (PII). Digital repositories have become increasingly susceptible to intentional or unintentional abuse, resulting in organizations to be liable under the privacy legislations that are being adopted by governments the world over. These privacy concerns have necessitated new advancements in the field of distributed data mining wherein, collaborating parties may be legally bound not to reveal the private information of their customers. In this paper, we present a new algorithm PriPSeP (PRI*vacy* P*reserving* SE*quential* P*atterns*) for the mining of sequential patterns from distributed databases while preserving privacy. A salient feature of PriPSeP is that due to its flexibility it is more pertinent to mining operations for real world applications in terms of efficiency and functionality. Under some reasonable assumptions, we prove that our architecture and protocol employed by our algorithm for multi-party computation is secure.

1. INTRODUCTION

The increasing use of multi-database technology, such as computer communication networks and distributed, federated and homogeneous multi-database systems, has led to the development of many large distributed transactional databases. For decision-making, large organizations might need to mine these multiple databases located at disparate branches and locations. Particularly, as the Web is rapidly becoming an information flood, individuals and organizations can take into account low-cost information and knowledge on the Internet while making decisions. Although this large data enables in the improvement of the quality of decisions, it also generates a significant challenge in the form of efficiently identifying quality knowledge from multi-databases [20, 25]. Therefore, large corporations may have to confront the multiple data-source problem. For example, a retail-chain with numerous franchisees might wish to collaboratively mine the union of all the transactional data. Each of the smaller transactional databases could contain information regarding the purchasing history of the same set of common customers transacting through online portals or real stores. However, the greater challenge of these computations can be the additional constraint of adhering to stringent privacy requirements laid down by the formulation of new laws such as HIPAA [15]. These regulatory policies have been the driving force behind the increased consciousness in organizations towards the protection of privacy. Consequently, there has been a paradigm shift towards the creation of privacy-aware infrastructures, which entail all aspects, ranging from data-collection to analysis [3].

Conventionally, data mining has operated on a data-warehousing model of gathering all data into a central site, then running an algorithm against that data. Privacy considerations may prevent this generic approach. Hence, privacy preserving data mining has gained recognition among academia and organizations as an important and unalienable area, especially for highly sensitive data such as health-records. If data mining is to be performed on these sensitive datasets, due attention must be given to the privacy requirements. However, conventional sequential pattern mining methods based on support do not preserve privacy and are ineffective for global pattern mining from multiple data sources.

Traditionally, Secure Multi-Party Protocols have been employed for the secure computation for any generic functions. However, the complexity and overhead of such secure protocols would be prohibitive for complex data mining tasks such as the discovery of sequential patterns. Hence, to alleviate the communication and bandwidth overhead of the Oblivious Transfer required between parties in an SMC, we employ an alternative architecture consisting of semi-honest and non-colluding sites [12]. This tradeoff between security and efficiency is reasonable as none of the participating sites learn the intermediate or the final results of the calculus. Furthermore, due to the uniform random noise in the datasets, the private information of any individual is also guarded from any possible leak.

In this paper, we present an alternative privacy preserving

*email: vkapoor@cse.iitd.ernet.in. This work was performed as part of an internship at the LGI2P Research Center.
†emails: {Pascal.Poncelet,trousset}@ema.fr
‡teisseire@lirmm.fr

data mining approach - PriPSeP, for finding sequential patterns in the distributed databases of a large integrated organization. Our novel algorithm, PriPSeP is useful for mining sequential patterns via collaboration between disparate parties, employing the secure architecture, performing the secure operations via the underlying protocols.

Organization: The remainder of this paper is organized as follows. Section 2 goes deeper into presenting the problem statement and provides an extensive description of the problem at hand. In Section 3, we present an overview of the related work and give our motivation for a new approach. Section 4 describes our proposed solution with the description of the architecture and the algorithms for secure multiparty protocols. Finally, Section 5 concludes the paper with a roadmap for future work.

2. PROBLEM STATEMENT

In this section, we give the formal definition of the problem of privacy preserving collaborative sequential pattern mining. First, we provide a brief overview of the traditional pattern mining problem by summarizing the formal description introduced in [1] and extended in [18]. Subsequently, we extend the problem by considering distributed databases. Finally, we formally define the problem of privacy preserving sequential pattern mining.

2.1 Mining of Sequential Patterns

Let DB be a database containing a set of customer transactions where each transaction T consists of a customer-id(CID), a transaction time(TID) and a set of items involved in the transaction.

Let $I = \{i_1, i_2 ... i_m\}$ be a set of literals called items. An itemset is a non-empty set of items. A sequence S is a set of itemsets ordered according to their timestamp. It is denoted by $< s_1\ s_2\ ...s_n >$, where $s_j, j \in 1...n$, is an itemset. In the rest of the paper we will consider that itemsets are merely reduced to items. Nevertheless all the proposal could be easily extended to deal with itemsets. A k-sequence is a sequence of k items (or of length k). A sequence $S' =< s'_1\ s'_2\ ...\ s'_n >$ is a subsequence of another sequence $S =< s_1\ s_2\ ...\ s_m >$, denoted $S' \prec S$, if there exist integers $i_1 < i_2 < ... i_j ... < i_n$ such that $s'_1 \subseteq s_{i_1}$, $s'_2 \subseteq si_2$, ... $s'_n \subseteq si_n$.

All transactions from the same customer are grouped together and sorted in increasing order and are called a data sequence. A support value (denoted $supp(S)$) for a sequence gives its number of distinct occurrences in DB. Nevertheless, a sequence in a data sequence is taken into account only once to compute the support even if several occurrences are discovered. In other words, the support of a sequence is defined as the fraction of total distinct data sequences that contain S. A data sequence contains a sequence S if S is a subsequence of the data sequence. In order to decide whether a sequence is frequent or not, a minimum support value (denoted *minsupp*) is specified by the user, and the sequence is said to be *frequent* if the condition $supp(S) \geq minsupp$ holds. Given a database of customer transactions, the problem of sequential pattern mining is to find all the sequences whose support is greater than a specified threshold (minimum support). Each of these represents a sequential pattern, also called a frequent sequence.

2.2 From Collaborative to Privacy Preserving Sequential Pattern Mining

Let DB be a database such as $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$. We consider that all databases $DB_1, DB_2 ... DB_D$ share the same number of customers (CIDs), which is N. We also consider that for each customer in the databases, the number of transaction times (TIDs), K, is the same[1]. As we extend the data representation scheme from the SPAM approach [2], we consider that all transactions are depicted in the form of vertical bitmaps, which we denote as vectors for clarity in mathematical formulae.

Definition 1. *Let V_i^j be a vector where j and i correspond respectively to the i^{th} item and the j^{th} database. V_i^j is defined as follows: $V_i^j = [C_1^{i,j} ... C_N^{i,j}]$ where for $u \in \{1..N\}$, $C_u^{i,j} = [T_1^{i,j,u}, ..., T_K^{i,j,u}]$. $T_{v=\{1..K\}}^{i,j,u}$ corresponds to the transaction list of the customer u, from the database DB_j and the item i. It is a K length bit string that has the v^{th} bit as one if the customer u has bought the item i from the database DB_j.*

Given a set of databases $DB_1, DB_2 ... DB_D$ containing customer transactions, the problem of collaborative sequential pattern mining is to find all the sequences whose support is greater than a specified threshold (minimum support). Furthermore, the problem of privacy-preserving collaborative sequential pattern mining is to discover sequential patterns embedded in the union of databases by considering that the parties do not want to share their private datasets with each other.

In order to illustrate this further, let us consider the following example.

Example 1. *Let us assume that three retail franchisees Alice, Bob and Carol wish to securely extract the sequential patterns in the union of their databases without disclosing the identities of any individual customers. Each item is provided with its timestamp (C.f. table 1).*

CID	Alice	Bob	Carol
1	$(1)_1\ (3)_5$	$(2)_2$	$(7)_4$
2	$(2)_4$	$(1)_3$	$(3)_6$
3	$(2)_6\ (3)_7$		$(1)_2\ (7)_3$

Table 1: An example of distributed databases sorted by CID

Let us assume that the minimal support value is set to 50%. From the three distributed databases, we can infer that item (1) is not frequent in any one of the individual databases. However, by considering the union of all databases (C.f. table 2 where the superscript depicts the original database, where the item is derived from), we obtain the sequence of $< (1)(2)(3) >$. By considering the constraints for privacy,

[1]This constraint has been considered purely for readability reasons. All the described algorithms could be easily extended to incorporate customer sequences that do not have the same number of TIDs.

CID	Sequences
1	$(1)^A_1\ (2)^B_2\ (7)^C_4\ (3)^A_5$
2	$(1)^B_3\ (2)^A_4\ (3)^C_6$
3	$(1)^C_2\ (7)^C_3\ (2)^A_6\ (3)^A_7$

Table 2: Sequences for each customer in the union of all databases

this sequence has to be obtained by considering Alice, Bob and Carol are not at liberty to disclose the private transactional history of any of the customers.

3. RELATED WORK

In this section we focus on the various research work closely related to the domain of privacy preserving data mining and sequential patterns.

Sequential Patterns: Since its introduction, more than a decade ago, the sequential pattern mining problem has received a great deal of attention and numerous algorithms have been defined to efficiently find such patterns (e.g. GSP [18], PSP [14], PrefixSpan [16], SPADE [23], FreeSpan[10], SPAM [2]). Our data representation scheme has been extended from the SPAM algorithm [2], wherein for efficient counting, each customer's transactions are represented by a vertical bitmap.

Privacy Preserving Data Mining: Recently, there has been a spate of work addressing privacy preserving data mining [17, 5]. This wide area of research includes classification techniques [7], association rule mining [8], and clustering [11] with privacy constraints. In early work on privacy-preserving data mining, Lindell and Pinkas [13] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by SMC research. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [19], where a solution to the association rule mining problem for the case of two parties was proposed. Recently, a novel secure architecture has been proposed in [12], where the security and accuracy of the data mining results are guaranteed with improved efficiency.

Secure Multi-Party Computation: A Secure Multi-party Computation (SMC) problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participants input and output. Secure two party computation was first investigated by Yao [21, 22] and was later generalized to multi-party computation (e.g. [6, 9, 4]). It has been proved that for any polynomial function, there is a secure multiparty computation solution [9, 4]. The approach used is as follows: the function f to be computed is firstly represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. While this approach is appealing in its generality and simplicity, the protocols it generates depend on the size of the circuit. This size depends on the size of the input (which might be huge as in a data mining application), and on the complexity of expressing f as a circuit (for example, a naive multiplication circuit is quadratic in the size of its inputs). Hence this approach, is highly impractical for large datasets and complicated computations necessary in complex data mining tasks. Our shift away from a traditional SMC approach has been motivated by [12], describing the limitations of highly secure, yet practically unviable protocols.

Previous Work: The research area of privacy preserving sequential pattern mining lies largely unexplored with only one seminal paper [24]. Zhan et al. have proposed a approach, which entails the transformation of the databases of each collaborating party, followed by the execution of a secure protocol, which results in the preservation of privacy, as well as the correct results. Theoretically, the approach is robust and secure, however, it has serious limitations relating to the initial constraints assumed while developing the approach. It has been proposed that each of the collaborating parties carries a unique inventory. For instance, considering our previous example and not taking into account the possibility of items being shared among the distributed parties, we do not arrive at the complete results. An item such as (1), which is not supported by enough customers in one individual database will not appear in the final results. This assumption causes serious limitation for real applications where item sharing between different databases is imperative as well as a fundamental requirement as shown earlier. Moreover, employing their new data representation scheme for sequential data, the same customer buying the same item more than once from the same database but with a different TID is not permissible. One other drawback of mapping each item to a unique code is the additional overhead incurred while sorting the databases, which might be significant for large databases.

4. THE PRIPSEP APPROACH

In this section, we propose our novel approach for privacy preserving sequential pattern mining in distributed and collaborative databases. Firstly we focus only on collaborative sequential pattern mining in order to clearly explain our methodology. This approach is extended in the next section in order to consider privacy requirements and finally we propose a new algorithm and underlying protocols within the secure architecture.

4.1 Collaborative sequential pattern mining

4.1.1 An overview

As previously seen in Section 2, the challenge with collaborative mining lies in the fact that we have to deal with different databases where the order of items is not known beforehand (e.g. item (7) of the CID 1 in Bob's database occurs before item (3) in Alice's database).

For brevity, we consider the Data Miner performing the generating and verifying phases of candidate sequences similar to the Apriori-based algorithms. We assume that the candidate generation is performed conventionally by combining the k-1 frequent sequences in order to generate k-candidate sequences (e.g. C.f. GSP [18] generation phase). We extend the verification phase as follows. As we have to deal with disparate distributed databases, we assume that the Miner

could request information from the D original databases in order to obtain a vector corresponding to the specific item i, i.e. $V_i^{[1..D]}$ for any candidate sequence.

Let us consider that we are provided with two databases, namely DB_1 and DB_2. These databases contain transcations for three customers and each customer has five transaction times or TIDs. The process aims at finding the support value for the sequence $< (1)(2) >$ in the set of all customers of the two databases. First, we extract from DB_1, the vector corresponding to the item (1), i.e. V_1^1, and from DB_2 the vector V_1^2 (left part of figure 1). From the given vectors, two key operations have to be performed: (i) bitwise OR of the two vectors, and (ii) transforming the result in order to check if it could followed by (2). These two vectors are merged together by applying a bitwise operator ($\vee$): $V_1^1 \vee V_1^2$. For the second operation, similar to the S-step process of the SPAM algorithm, we consider a function that transforms the vector(bitmap). For each customer, following the occurrence of the first bit with value one, every subsequent bit in the vector is flagged as one. However, since we have to deal with different databases as well as efficiency issues, we consider that these two operations are performed through the f function defined below to obtain a new vector $Z_1 = f(V_1^1 \vee V_1^2)$.

DEFINITION 2. *Let us consider a vector V_i^j for a database j and an item i. V_i^j is defined as follows: $V_i^j = (C_1^{i,j}...C_N^{i,j})$ where for $u \in \{1..N\}$, $C_u^{i,j} = (T_1^{i,j,u}, ..., T_K^{i,j,u})$. K stands for the number of TIDs and N corresponds to the number of CIDs. For brevity, we denote this vector as V. Let $f : [0,1]^{N\times K} \rightarrow [0,1]^{N\times K}$ be a function such that: $f(V) = f(C_1...C_N) = [f_c(C_1)f_c(C_2)...f_c(C_N)]$. For each $u \in \{1..N\}$, we have: $f_c(C_u) = \left| \begin{array}{l} 0 \\ T_1^u \\ T_1^u \vee T_2^u \\ T_1^u \vee T_2^u \vee T_3^u \\ ... \\ T_1^u \vee ... \vee T_{k-1}^u \end{array} \right|$*

where $\vee$ is a bitwise operator. We can notice that $Card(V) = N \times K$, $Card(C_u) = K$, $Card(f(V)) = N \times K$.

Let $g : [0,1]^{N\times K} \rightarrow [0,1]^N$ be a function such that: $g(V) = g(C_1...C_N) = [g_c(C_1)g_c(C_2)...g_c(C_N)]$. For each $u \in \{1..N\}$, we have: $g_c(C_u) = 1$ if there exists at least one bit with value 1 in the customer transactions. It can be noted that $Card(g(V)) = N$.

In conjunction with the computation of the function f, the vectors corresponding to the item (2) are extracted from DB_1 and DB_2 (V_2^1 and V_2^2 respectively). Similar to the previous step the vector ($Z_2 = V_2^1 \vee V_2^2$) is computed. Following that, the bitwise operator $\wedge$ is used to calculate $Z_1 \wedge Z_2$ and the g function is used to calculate the count for each customer, for the sequence $< (1)(2) >$, i.e. $Z_3 = g(f(V_1^1 \vee V_1^2) \wedge (V_2^1 \vee V_2^2))$. As the resulting vector Z_3 has a cardinality corresponding to the number of customers, i.e. N, the last operation to be performed is a summation of the number of bits with the value 1 in the vector Z_3. This is performed by the $\sum$ operation.

4.1.2 The collaborative support counting algorithm

The COLLABORATIVE FREQUENCY algorithm (see Algorithm 1) has been developed as follows. For each item i of the candidate sequence to be tested, a new vector X_i is generated by applying the $\vee$ bitwise operator on all the corresponding vectors from the original databases. Hence, by considering the result of the previous operation, the f function is applied, followed by the bitwise operator $\wedge$ for each item. At the end of this iteration, a new vector Z of cardinality $N \times K$ is produced. Consequently, the g function is applied to the intermediate result for generating a vector of cardinality N, i.e. Y. Finally, the number of bits which are 1 in Y are summated to compute the final value of support.

Algorithm 1: The COLLABORATIVE FREQUENCY algorithm

Data: $S = < it_1 \ ... \ it_q >$ a sequence to be tested; $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$ a set of databases; N the number of customers shared by all databases; K the number of date shared by all customers of all databases.

Result: The support of the sequence S in DB.

foreach $i \in 1..|S|$ **do**
  $X_i \leftarrow V_{it_i}^1 \bigvee ... \bigvee V_{it_i}^D$;
$Z \leftarrow X_1$;
foreach $i \in 2..|S|$ **do**
  $Z \leftarrow f(Z) \bigwedge X_i$;
$Y \leftarrow g(Z)$;
return $\sum_{i=1}^{N} Y_i$;

Complexity: Let $V_s = N \times K$ be the size of the vectors which are sent and S be the candidate sequence to be verified. The transfers that is performed by the algorithm are: $(V_s \times D \times S)$ for $\bigvee$ and $(V_s \times S)$ for both the f function and $\bigwedge$ operation. There are $(N(K-2))$ $\bigvee$ computations performed by f. If f is already available, i.e. precomputed and stored, we have (N) $\bigvee$ operations, otherwise $(N(K-1))$ $\bigvee$ operations are performed by g.

4.2 From collaborative to privacy-preserving sequential pattern mining

4.2.1 A brief overview of the architecture

In this section we describe an architecture where secure multi-party techniques developed in the cryptographic domain can be easily extended for data mining purposes[12]. Previous work [9] has described that Secure Multi-party protocols can be used directly to solve with total security, any generic data mining task. However, the drawback is the complexity of the protocol and the requirements that all parties need to be online during the entire duration of the lengthy process. Hence, it is potentially unviable for complex data mining tasks, particularly relating to cases with a large number of participants. The communication complexity prohibits efficient scalability and for situations that all parties cannot remain online for the entire process, the SMC protocols are rendered useless.
Hence, as proposed in [12], we deploy a safe architecture for performing the data mining task without leaking any useful or sensitive information to any of the intermediate parties.

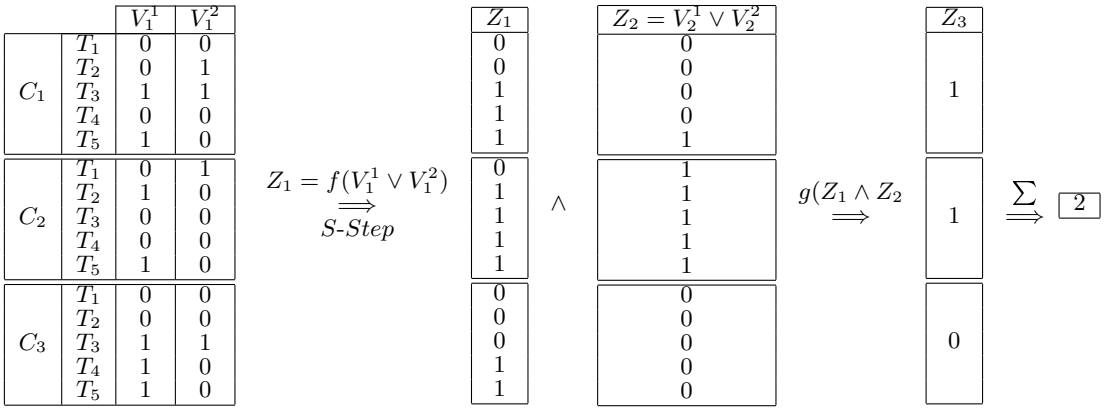

Figure 1: Processing of vectors for collaborative mining

These independent sites collect, store and evaluate information securely. PriPSeP requires three *non colluding* and *semi honest* [9] sites which follow the protocol correctly but can utilize the information collected by them. In effect, all parties correctly follow the protocols, but then are free to use whatever information they see during the execution of the protocols in any way. These are also referred to as *honest but curious* sites.
The detailed functions of each of these sites are described:

- **Data Miner Site** DM: The Data Miner is a randomly chosen collaborator between the original databases. Certainly, the Data Miner could be a third party instead of one of the databases. It's purpose is to perform the mining operations, to interact with NC_1 and NC_2, and to receive the final result of the computation from the PS. It should be noted that the final result is also sent to all the original databases.

- **Non Colluding Sites** NC_1 **and** NC_2: These symmetric sites collect the noisy data from each database including the Data Miner and perform a series of secure operations without inferring any intermediate or final result.

- **Processing Site** PS: This site is utilized by both NC_1 and NC_2 sites for computing securely the various functions and operations underlying PriPSeP. Similar to NC_1 and NC_2, PS learns no actual results.

Let us consider Figure 2 illustrating the sites and, for each operator, the different exchanges performed between the sites. Initially the following preprocessing steps are performed on the databases individually:

1. Each database $DB_1, DB_2 ... DB_D$ adds the same number ε of customers with fake transactions and employ a non-secure counting strategy (this count could be performed by any conventional algorithm since this step is independent of the privacy) to note the number of customers, ε', that have to be pruned from the final result.

2. Let φ be a random number. Each database permutes individually their vector of transactions (V_i^j) according to the value of φ .

3. One of the collaborating parties is randomly elected to perform the data mining steps. This party is termed as the Data Miner (DM).

At the end of the preprocessing we are provided with databases having fake customer transactions and permuted list of vertically aligned vectors. Subsequently, the Data Miner can apply an Apriori-like algorithm for generating candidate sequences as previously mentioned in Section 4.1. This step is immediately followed by the counting phase. For simplicity, let us take the case of counting the value of support for the two-length sequence $< (1)(2) >$. Now, each database DB_j sends its V_1^j vector to NC_1 and NC_2 (dashed arrows numbered 1 in figure 2). In order to minimize the risk of network attacks, we propose a hypothetical function $\text{Send}^S \times DB_d(it)$ which securely transmits the item vector V_{it} from database DB_d to NC_1 and NC_2. Furthermore, in order to make sure that NC_1 and NC_2 receive minimal information, for each database DB_i, we generate a random vector R_{DB_i} having the same size than V_{it} and then calculate a vector: $Z_{DB_i} = V_{it} \bigoplus R_{DB_i}$ and send either Z_{DB_i} to NC_1 and R_{DB_i} to NC_2 or vice versa. It has been proved in [5], that any data mining task (h) defined on a vector $X = [x_1, x_2, ...x_n]$, it suffices to evaluate $h(X \bigoplus R) = h(X)$ since $R = [r_1, r_2, ...r_n]$ and $X \bigoplus R = [x_1 \oplus r_1, x_2 \oplus r_2, ...x_n \oplus r_n]$. In this case, for NC_1 and NC_2 sites we have some R_{DB_i} vectors and since the other vectors are XOR-ed $\bigoplus$ with a random vector, they are indistinguishable from a uniform random distribution.
Similar to Algorithm 4.1, the bitwise operator ($\vee$) has to be applied between every vector. As these vectors are shared by NC_1 and NC_2, we consider a new protocol $\bigvee^S$ (arrows numbered 2 in Figure 2) aiming at computing a bitwise OR between the different vectors. This is performed by sending XOR-ed randomized values from NC_1 and NC_2 to PS. Then PS also garbles the resulting vectors in order to divide the result between NC_1 and NC_2. The calculation continues by computing the f and g functions (subsequently referred to as f^S and g^S) in a similar way and results are also stored between NC_1 and NC_2 (arrows numbered 3 in Figure 2).

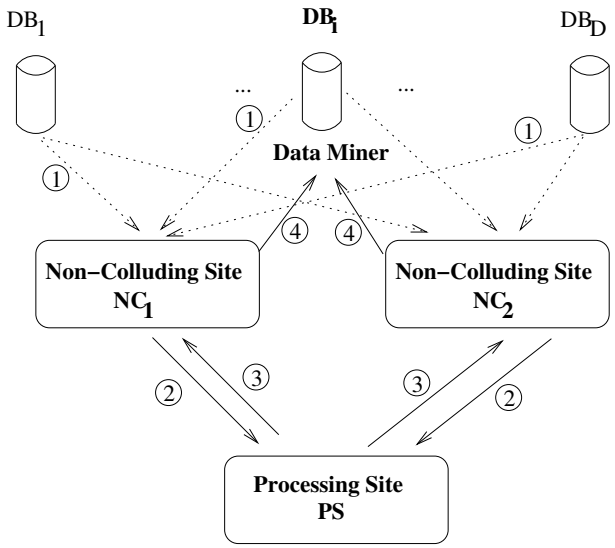

Figure 2: PRIPSEP **Architecture**

Finally, in order to compute the number of bits which are in 1 ($\sum$ function, now termed as $\sum^S$), NC_1 and NC_2 collaborate to append their resultant vector with randomized values and then reorder the new vector. PS then calculates the summation of the number of bits and returns part of the result to NC_1 and NC_2. NC_1 removes their initial random noise and then return those final results to the Data Miner (arrows numbered 4 in Figure 2). At this step, DM only has to combine the result from NC_1 and NC_2 and then remove the ε' value corresponding to random customers added in the preprocessing phase.

In the following sections, we will explain in detail the various protocols, functions and algorithms necessary for PRIPSEP. Firstly, we introduce some notations that are used for describing the algorithms. As our functions employ bitwise operators, we first present new protocols for securely performing bitwise operations. Continuing, we will show how the functions f, g and $\sum$ are extended to f^S, g^S and $\sum^S$ respectively to incorporate security aspects. Finally, we present the SECURE COLLABORATIVE FREQUENCY algorithm. As the main goal of our approach is to preserve privacy of the individual users and do not divulge any information about the final result to any of the sites, we will show that at the end of the process, NC_1, NC_2 and PS will only learn a upper bound on the support count of sequences and will not have any information about the private inputs of any of the individual customers.

4.2.2 Notations

In the next subsections, we will consider the following notations. Let $(\overset{+}{X} \mid \overset{-}{X}) \leftarrow h^S(\overset{+}{Y_1} \ldots \overset{+}{Y_n} \mid \overset{-}{Y_1} \ldots \overset{-}{Y_n})$ be a tripartite calculation of any function h^S between NC_1, NC_2 and PS where NC_1 owns half of the input $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ and gets half of the result $\overset{+}{X}$, and similarly NC_2 owns half of the inputs $\overset{-}{Y_1} \ldots \overset{-}{Y_n}$ and gets half the result $\overset{-}{X}$ at the end of the process. The final result is the logical bitwise XOR ($\bigoplus$) of the $\overset{+}{X}$ and $\overset{-}{X}$. However, this does not imply that NC_1 directly sends $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ to PS and receives the result $\overset{+}{X}$ from PS. Initially, NC_1 transforms its inputs $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ to $\overset{+}{Y'_1} \ldots \overset{+}{Y'_n}$ via the addition of uniform random noise and securely sends these transformed Y' to PS. Symmetrically, NC_2 also sends its garbled inputs to PS. At the end of the computation both the sites receive their share of the noisy result $\overset{+}{X'}$ and $\overset{-}{X'}$ from PS. Henceforth, this intermediate result can be used as the inputs for further computations.

4.2.3 The $\bigwedge^S$ and $\bigvee^S$ protocols

In this section, we define two basic algorithms ($\bigwedge^S$ (see Algorithm 2) and $\bigvee^S$ (see Algorithm 3)) providing the protocol which is used to securely compute the bitwise operators. The underlying fundamental principle that the algorithms operate upon is the addition of uniform random noise to the data which can be removed from the result by the data-owners. The protocol initiates with both NC_1 and NC_2 perturbing their data by XOR-ing it with random values. Subsequently, the randomized data is sent (e.g. for NC_2, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ and $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$) to PS, which can calculate the $\bigwedge$ (resp. the $\bigvee$) securely. It actually operates on the randomized inputs and calculates $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ and $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$. It then also adds random noise to both the intermediate results in order to avoid that either NC_1 or NC_2 have the whole result. At the end of the protocol, non colluding sites can then calculate the final result for their own part by removing the initial noise. For instance, for NC_1, the following operation: $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$ could be done securely since it knows its own part ($\overset{+}{X}$, $\overset{+}{Y}$ and R'_A) and random numbers from NC_2 (R'_B and R_B). Hence, the final results $A^R \bigoplus B^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus B'_{PS}$

Algorithm 2: The $\bigwedge^S$ protocol

Data: $(\overset{+}{X}, \overset{+}{Y} \mid \overset{-}{X}, \overset{-}{Y})$ bits are such as $\overset{+}{X}$ and $\overset{+}{Y}$ owned by NC_1, $\overset{-}{X}$ and $\overset{-}{Y}$ owned by NC_2.

Result: $(A^R|B^R)$ are such that $A^R \bigoplus B^R = (\overset{+}{X} \bigoplus \overset{-}{X}) \bigwedge (\overset{+}{Y} \bigoplus \overset{-}{Y})$.

1. NC_1 and NC_2 mutually generate and exchange four random numbers R_A, R'_A, R_B and R'_B such that $\overset{+}{X}' = \overset{+}{X} \bigoplus R_A$, $\overset{+}{Y}' = \overset{+}{Y} \bigoplus R'_A$, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ and $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$.
2. NC_1 sends $\overset{+}{X}'$ and $\overset{+}{Y}'$ to PS.
3. NC_2 sends $\overset{-}{X}'$ and $\overset{-}{Y}'$ to PS.
4. PS calculates $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ and $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$ and a random number R_{PS}.
5. PS sends $A'_{PS} = \overset{+}{C} \bigoplus R_{PS}$ to NC_1 and $B'_{PS} = \overset{-}{C} \bigoplus R_{PS}$ to NC_2.
6. NC_1 calculates $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$
7. NC_2 calculates $B^R = B'_{PS} \bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A) \bigoplus (\overset{-}{X} \bigwedge \overset{-}{Y}) \bigoplus (R_A \bigwedge R'_B)$.

$\bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A)$. Substituting the value of A'_{PS} and B'_{PS}, the initial and intermediate random numbers are removed due to the boolean property $R_{PS} \bigoplus R_{PS} = 0$. The desired result is $\overset{+}{X} \bigwedge \overset{+}{Y} \bigoplus \overset{+}{X} \bigwedge \overset{-}{Y} \bigoplus \overset{-}{X} \bigwedge \overset{+}{Y} \bigoplus \overset{-}{X} \bigwedge \overset{-}{Y}$. Although, this operation is never performed, the symmetrically divided result lies with both NC_1 and NC_1.

THEOREM 1. *The operand $\bigwedge^S$ (resp. $\bigvee^S$) prohibits NC_1 from learning NC_2's private data and vice versa. Moreover, the third party PS learns none of their private inputs.*

Proof: From the protocol, B'_{PS} is all that NC_2 learns related to the private data of NC_1. Due to the randomness and secrecy of R_{PS}, NC_2 cannot find out the values of $\overset{+}{X}$ or $\overset{+}{Y}$. As the roles of NC_1 and NC_2 are interchangeable, the same argument holds for NC_1 not learning the private inputs $\overset{-}{X}$ or $\overset{-}{Y}$ of NC_2. However, one key security aspect of not leaking any information to PS is achieved by randomizing the inputs before transmitting them to the Processing Site. Due to the randomization performed during the initial step, it just infers a stream of uniformly distributed values, and cannot distinguish between a genuine and a random value.

Complexity: For the $\bigwedge^S$ operator, ten computations have to be performed ($6 \bigotimes$ and $4 \bigwedge$). As, two more XOR operations are performed for the $\bigvee^S$ protocol, we have in total, twelve computations. For each $\bigwedge^S$, NC_1 and NC_2 exchange 2×2 bits. From NC_1 or NC_2, 2×1 bits are sent to PS and one bit returned. Furthermore, both NC_1 and NC_2 calculate 2 random bits while 1 random bit is generated by PS.

Algorithm 3: The $\bigvee^S$ protocol

Data: $(\overset{+}{X}, \overset{+}{Y} \mid \overset{-}{X}, \overset{-}{Y})$ bits are such as $\overset{+}{X}$ and $\overset{+}{Y}$ owned by NC_1, $\overset{-}{X}$ and $\overset{-}{Y}$ owned by NC_2.

Result: $(A^R|B^R)$ are such that $A^R \bigoplus B^R = (\overset{+}{X} \bigoplus \overset{-}{X}) \bigvee (\overset{+}{Y} \bigoplus \overset{-}{Y})$.

1..5. The first 5 steps are the same as $\bigwedge^S$.

6. NC_1 calculates $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus \overset{+}{X} \bigoplus \overset{+}{Y} \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$.
7. NC_2 calculates $B^R = B'_{PS} \bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A) \bigoplus \overset{-}{X} \bigoplus \overset{-}{Y} \bigoplus (\overset{-}{X} \bigwedge \overset{-}{Y}) \bigoplus (R_A \bigwedge R'_B)$.

4.2.4 The f^S, g^S and $\sum^S$ functions

Algorithm 4: The f^S function

Data: Vectors of bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ is coming from NC_1 and $\overset{-}{X}$ is coming from NC_2. K the number of dates shared by each customers of all databases.

Result: Vectors $(\overset{+}{Y} \mid \overset{-}{Y})$ such as $\overset{+}{Y}$ is the share of NC_1 and $\overset{-}{Y}$ the share of NC_2.

foreach $c \in 0..(|\overset{+}{X}|/K) - 1$ **do**
  // For each client c in [0..N-1]
  $(\overset{+}{Y}_{K \times c+1} \mid \overset{-}{Y}_{K \times c+1}) \leftarrow (0|0)$;
  foreach $i \in 2..K$ **do**
    $(\overset{+}{Y}_{K \times c+1} \mid \overset{-}{Y}_{K \times c+1}) \leftarrow$
    $\bigvee^S (\overset{+}{Y}_{K \times c+i-1}, \overset{+}{X}_{K \times c+i-1} \mid \overset{-}{Y}_{K \times c+i-1}, \overset{-}{X}_{K \times c+i-1})$;
return $(\overset{+}{Y} \mid \overset{-}{Y})$;

In this section, we extend the f and g functions in order to incorporate security (see Algorithm 4). As previously mentioned, the SPAM algorithm's S-step Process requires that the vectors corresponding to every customer contain all 1's after the date of the first transaction for that customer. Hence, the f^S function recursively employs the $\bigvee^S$ function to securely compute the resultant vector. The inputs of the function are the randomly distorted customer data and the secure $\bigvee^S$ is used to find the boolean OR between the successive bits residing at the two sites NC_1 and NC_2. Similar to the previous algorithms, the final result of the operation is split into two parts with the Processing Site oblivious of the correct answer.
Similarly, the g^S function (see Algorithm 5) securely computes the existence of at least one bit with value '1' in the vector of each customer transaction. It reduces the vector to a single bit of value either '0' '1' depending on whether the sequence is supported at least once. This function is useful

Algorithm 5: The g^S function

Data: Vectors of bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ is coming from NC_1 and $\overset{-}{X}$ is coming from NC_2. K the number of dates shared by all customers of all databases.

Result: Vectors $(\overset{+}{Y} \mid \overset{-}{Y})$ such as $\overset{+}{Y}$ will be send to NC_1 and $\overset{-}{Y}$ will be send to NC_2.

foreach $c \in 0..(|\overset{+}{X}|/K) - 1$ **do**
  // For each client c in [0..N-1]
  $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \leftarrow (\overset{+}{X_{K\times c+1}} \mid \overset{-}{X_{K\times c+1}})$;
  foreach $i \in 2..K$ **do**
    $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \leftarrow \bigvee^S(\overset{+}{Y_c}, \overset{+}{X_{K\times c+i}} \mid \overset{-}{Y_c}, \overset{-}{X_{K\times c+i}})$;

return $(\overset{+}{Y} \mid \overset{-}{Y})$;

in calculating the support value at the penultimate step of the Algorithm 7.

THEOREM 2. *The functions f^S and g^S are secure and restricts NC_1, NC_2 and PS from inferring each other's private data.*

Proof: From the algorithms, it is apparent that the secure operation $\bigvee^S$ is applied iteratively to arrive at the results. As proved in Theorem 1, no private information is shared while the execution of this operation. Hence, both the functions f^S and g^S are also secure and no site infers any information about any individual customer.

Remark: In fact, calculating $g^S(\overset{+}{X}, \overset{-}{X}) \rightarrow (\overset{+}{Y}, \overset{-}{Y})$ can be returned while calculating $f^S(\overset{+}{X}, \overset{-}{X}) \rightarrow (\overset{+}{Z}, \overset{-}{Z})$ because $\overset{+}{Y_i}, \overset{-}{Y_i}$ can easily be obtained from $(\overset{+}{Z_{i\times K+K}}, \overset{-}{Z_{i\times K+K}})$ by using the following relation:
$(\overset{+}{Y_i} \mid \overset{-}{Y_i}) = \bigvee^S(\overset{+}{Z_{i\times K+K}}, \overset{+}{X_{i\times K+K}} \mid \overset{-}{Z_{i\times K+K}}, \overset{-}{X_{i\times K+K}})$.

THEOREM 3. *The functions $\sum^S$ protocol is fully secure and does not reveal the final value of support for the candidate sequence to either NC_1, NC_2 or PS.*

Proof: Two random vectors R_1 and R_2 are appended to the inputs of NC_1 and NC_2 to prevent PS from distinguishing between genuine and random values. The final computed result is divided by PS between the two sites NC_1 and NC_2. Hence the final computation is performed by the Data Miner, which receives the correct result.

Complexity: In Algorithm 6, the number of bits is increased by a value $\geq 2N$ for security reasons. Let us consider that we set this value as follows $t =\in [2..K]$. For NC_1 and NC_2, $(2N(2t+1))$ operations are performed while $(2N(t+1))$ operations on PS. Furthermore we have $N(t+1)$ operations for randomizing. The number of transfers between NC_1 and NC_2 is $(2tN)$. To exchange the permutation φ between NC_1 and NC_2, we actually need $N(t+1)$ transfers. Nevertheless, if NC_1 and NC_2 share a common set of random values generator, they only need to exchange a number and a seed. So it could be neglected. Finally between NC_1/NC_2 and PS, $N(t+1)$ bits are transferred.

Algorithm 6: The $\sum^S$ protocol

Data: Vectors of bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ is coming from NC_1 and $\overset{-}{X}$ is coming from NC_2.

Result: A number which is shared in two parts: $(\overset{+}{NB} \mid \overset{-}{NB})$, corresponding to the number of bits at 1 in vectors $(\overset{+}{X} \bigoplus \overset{-}{X})$, such as the real result is $NB = \overset{+}{NB} + \overset{-}{NB}$.

1. NC_1 and NC_2 generate and exchange two random vectors R_1 and R_2 of same cardinality such as $(Card(R_1) = Card(R_2) \geq 2N)$. They both calculate $R_1 \bigoplus R_2$ and calculate the number of 1s to be deleted, N_R, at the end of the computation from PS.
2. NC_1 and NC_2 reorder respectively the vector $(\overset{+}{X}, R_1)$ and $(\overset{-}{X}, R_2)$ using a permutation value φ and get respectively $\overset{+}{Y}$ and $\overset{-}{Y}$.
3. NC_1 sends $\overset{+}{Y}$ to PS and NC_2 sends $\overset{-}{Y}$ to PS.
4. PS calculates $\overset{+}{Y} \bigoplus \overset{-}{Y}$ and count the number of bits at 1 and gets NB.
5. PS gets a random number R_{PS} and returns $\overset{+}{N} = NB + R_{PS}$ to NC_1 and $\overset{-}{N} = NB - R_{PS}$ to NC_2.
6. NC_1 computes $\overset{+}{NB} = \overset{+}{N} - N_R$, NC_2 keeps only $\overset{-}{NB} = \overset{-}{N}$.

4.2.5 *The* SECURE COLLABORATIVE FREQUENCY *algorithm*

The SECURE COLLABORATIVE FREQUENCY algorithm (see Algorithm 7) extends the Algorithm 1 in order to perform all operations securely. It is applied after the preprocessing step and thus considers the original database having fake transactions. For each item i of the sequence to be tested, all noisy vectors are sent by SEND^S to NC_1 and NC_2 in order to securely apply an OR between each vector $(\bigvee^S)$. The f^S function followed by the bitwise operator $\bigwedge^S$ is performed. At the end of this loop we are thus provided with a new vector $(\overset{+}{Z} \mid \overset{-}{Z})$ where part of results are shared between NC_1 and NC_2. Then we apply the g^S function for generating $(\overset{+}{Y} \mid \overset{-}{Y})$. Finally, we count the number of bits which are 1 in $(\overset{+}{Y} \mid \overset{-}{Y})$ through the function $\sum^S$. At the end of the process, $\overset{+}{R}$ and $\overset{-}{R}$ are sent respectively by NC_1 and NC_2 to the Data Miner party. To get the real and final result, the miner has just to calculate $\overset{+}{R} + \overset{-}{R}$ (integer summation) and has to remove the initial random noise, i.e. ε', they have added at the beginning of the process.

THEOREM 4. *The randomization, performed at each level (original databases, NC_1, NC_2 and PS), does not affect the accuracy of the result.*

Proof: The first randomization is performed by the original databases while inserting fake transactions, i.e. ε, and permuting the list customers according to the value of φ. As, DM is elected from the original databases, this information

Algorithm 7: The SECURE COLLABORATIVE FREQUENCY algorithm

Data: $S =< it_1...it_q >$ a sequence to be tested; $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$ a set of databases; N the number of customers shared by all databases; K the number of dates shared by all customers of all databases.

Result: The support of the sequence S in DB with random noise.

foreach $i \in 1..|S|$ **do**
  $(\overset{+}{X}_i \mid \overset{-}{X}_i) \leftarrow \text{SEND}^S \times DB_1(i)$;
  foreach $j \in 2..D$ **do**
    $(\overset{+}{V} \mid \overset{-}{V}) \leftarrow \text{SEND}^S \times DB_j(it_i)$;
    $(\overset{+}{X}_i \mid \overset{-}{X}_i) \leftarrow \bigvee^S(\overset{+}{C}_i, \overset{+}{V} \mid \overset{-}{C}_i, \overset{-}{V})$;

$(\overset{+}{Z} \mid \overset{-}{Z}) \leftarrow (\overset{+}{X}_1 \mid \overset{-}{X}_1)$;
foreach $i \in 2..|S|$ **do**
  $(\overset{+}{T} \mid \overset{-}{T}) \leftarrow f^S(\overset{+}{Z} \mid \overset{-}{Z})$;
  $(\overset{+}{Z} \mid \overset{-}{Z}) \leftarrow \bigwedge^S(\overset{+}{T}, \overset{+}{X}_i \mid \overset{-}{T}, \overset{-}{X}_i)$;
$(\overset{+}{Y} \mid \overset{-}{Y}) \leftarrow g^S(\overset{+}{Z} \mid \overset{-}{Z})$;
$(\overset{+}{R} \mid \overset{-}{R}) \leftarrow \sum^S(\overset{+}{Y} \mid \overset{-}{Y})$;
return $(\overset{+}{R} \mid \overset{-}{R})$;

about the noise is available to DM and hence can easily be removed. The second randomization is performed by NC_1 and NC_2 while sending the transaction vectors to PS for the secure computation of $\bigvee^S$, $\bigwedge^S$, f^S and g^S. This added noise is removed at the end of each computation from NC_1 and NC_2 when they receive results from PS by performing an XOR operation with the initial random values. Moreover, we have also proved that no private information about any individual could be learnt by any of the sites (C.f. Theorems 1,2, and 3). Finally, for the computation of the $\sum^S$ function, NC_1 and NC_2 add random noise in their data, i.e. N_R, and also permute their vector according to a φ value. PS also randomizes its integer value and this noise is removed by sending opposite parts to NC_1 and NC_2. The N_R value is removed by NC_1 and NC_2 when returning the result to DM. Finally, when combining results from NC_1 and NC_2, the only operation to be performed by DM to know the real result is to remove the initial ε'.

Complexity: In the secure protocol, each database has to send $2NK$ data bits instead of NK. Each DB has also to calculate NK random bits and perform $(NK) \bigoplus$ operations. According to the previous results on the number of operations performed by the secure operators, the time complexity is $O(12NK)$ for binary operations and $O(7NK)$ for randomizing operations. Hence, it could be bounded by $O(20NK)$. Let us now consider the communication complexity of the protocol. Let $p = D \times S \times N \times K$. The complexity of the Algorithm 1, i.e. without considering security, is linear. Let us consider $C_{or} = O(p)$. As the secure algorithm considers the same structures as well as the same order of the operations, we have the complexity of $20 \times C_{or}$. The number of transfers required is at most four times the complexity of the transfer of the 1.

The secure architecture could be further redefined in order to improve the communication cost between NC_1, NC_2 and PS. Furthermore, all the functions could be parallelized. By considering that operations performed on a new architecture could be done securely by a multiple of 20, we could very easily remove this overhead. The last overhead is the communication cost incurred during the transfer between original databases and NC_1/NC_2. The overhead of the all systems could thus be only 2. We notice, that by considering SMC protocols, no such optimizations are possible, and hence for scalability issues, our alternative approach could be beneficial.

4.2.6 *Security of the protocol*

For analyzing the security, let us examine the information divulged to each site participating in the protocol. Note that during the entire process, the random numbers are securely generated and the communication infrastructure is robust and intrusion free.

- NC_1 **and** NC_2 **View**: During the execution of the protocol, both sites just see a stream of random values with a uniform distribution. By the proposed protocol, they only receive noisy data and noisy shared results. Also NC_1 and NC_2 cannot share information as per the definition of semi-honest non-colluding sites. The value received from the DBs are Xored with random numbers from a uniform distribution and indistinguishable from real values.
- **PS View**: It performs the computation of secure operations ($\bigwedge^S$, $\bigvee^S$, f^S, g^S, $\sum^S$) and provides the results to NC_1 and NC_2. As discussed earlier (C.f. Theorem 1, Theorem 2and Theorem 3) all of these operations and functions reveal no private data of any individual customer from any of the collaborating DBs. Even a succession or sequence of these secure protocols remains secure.
- **Overall Security**: During the entire algorithm, no site gets to know any additional information beyond of what they are already authorized to learn. Hence the security and privacy of every customer is maintained during the computation of support in the architecture. The addition of fake transactions during the preprocessing steps and the permutation of the lists enable that each site is ignorant of the correct intermediate results as well as the final result.

5. CONCLUSION

In this paper, we have addressed the problem of privacy preserving sequential pattern mining in distributed databases. We have presented a novel secure extension of the SPAM algorithm for mining patterns. We also prove, that under reasonable assumptions, our algorithm and the underlying operations, protocols and architecture for multiparty computation is secure. There are various avenues for future work. Firstly, in this paper we have only focused on the S-step process of the SPAM algorithm, i.e. we only considered the problem of discovering sequences reduced to a list of items. As we have proposed a series of secure functions, our approach could be easily improved to also consider I-step process, i.e. a list of itemsets instead of items. In the same

way, the algorithms and underlying protocols could be easily improved by considering that the number of TIDs is different between the customers. In this case, the only constraint to be considered is that databases share the same number of customers or CIDs. Furthermore, in the current version of PriPSep, results are directly returned to the DM party as well as the original databases. In order to improve the whole process, we plan to extend the role of DM wherein, it could store the lexicographic tree and could expand each node in the tree by considering that intermediate results could be stored in shared arrays between NC_1 and NC_2. Hence, incremental mining could be possible and unlike our current approach, previous results need not be recomputed. The storage of results would also be made secure by ensuring that each site has only noisy data or random values.

6. REFERENCES

[1] R. Agrawal and R.Srikant. Mining sequential patterns. In *Proc. of ICDE 95*, pages 3–14, 1995.

[2] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using bitmap representation. In *Proc. of KDD 02*, pages 439–435, 2002.

[3] J. Bhattacharya, S. Gupta, V. Kapoor, and R. Dass. Utilizing network features for privacy violation detection. In *Proc. of Int. Conf. on Communication System Software and Middleware*, January 2006.

[4] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proc. of the 20th Annual Symposium on the Theory of Computing (STOC)*, pages 11–19, 1988.

[5] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, 2003.

[6] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *New Security Paradigms Workshop*, pages 11–20, Cloudcroft, USA, 2001.

[7] W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proc. of the Fourth SIAM Int. Conf. on Data Mining*, pages 222–233, 2004.

[8] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy-preserving mining of association rules. *Information Systems*, 29(4), June 2004.

[9] O. Goldreich. Secure multi-party computation. Working Draft, 2000.

[10] J. Han, J. Pei, B. Mortazavi-asl, Q. Chen, U. Dayal, and M. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. of KDD 00*, pages 355–359, 2000.

[11] G. Jagannathan and R. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proc. of KDD 05*, august 2005.

[12] M. Kantarcioglu and J. Vaidya. An architecture for privacy-preserving mining of client information. In *Proc. of the Workshop on Privacy, Security, and Data Mining in conjunction with the 2002 IEEE ICDM Conf*, 2002.

[13] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(2), 2002.

[14] F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proc. of PKDD 98*, 1998.

[15] U. S. D. of Health & Human Services. Health insurance portability and accountability act of 1996. *http://www.hipaa.org/*, August 1996.

[16] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: mining sequential patterns efficiently by prefix projected pattern growth. In *Proc. of ICDE 01*, 2001.

[17] B. Pinkas. Cryptographic techniques for privacy preserving data mining. *SIGKDD Explorations*, 4(2), 2002.

[18] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of EDBT 96*, pages 3–17, 1996.

[19] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of KDD 02*, July 2002.

[20] X.Wu and S. Zhang. Synthesizing high-frequency rules from different data sources. *IEEE Trans. on Knowledge and Data Engineering*, 15(2):353–367, 2003.

[21] A. C. Yao. Protocols for secure computations. In *Proc. of the 23rd annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[22] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

[23] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1):31–60, February 2001.

[24] J. Zhan, L. Chang, and S. Matwin. Privacy-preserving collaborative sequential pattern mining. In *Workshop on Link Analysis, Counter-terrorism, and Privacy in conjunction with SIAM Int. Conf. on Data Mining*, pages 61–72, Lake Buena Vista, Florida, 2004.

[25] N. Zhong, Y. Yao, , and S. Ohsuga. Peculiarity oriented multi-database mining. In *Proc. of PKDD 99*, pages 136–146, 1999.