From Collaborative to Privacy Preserving Sequential Pattern Mining

V. Kapoor[1], P. Poncelet[2], F. Trousset[3], and M. Teisseire[4]

[1] Microsoft, One Microsoft Way, Redmond, WA - 98052
vishal.kapoor@microsoft.com
[2] LIRMM UMR CNR5 5506, 161 Rue Ada, 34392, Montpellier Cedex 5, France
poncelet@lirmm.fr
[3] EMA-LGI2P/EERIE, Parc Scientifique Georges Besse, 30035 Nîmes Cedex, France
Francois.Trousset@mines-ales.fr
[4] UMR TETIS, Maison de la Télédetection, 500, rue J.F.Breton 34093, Montpellier Cedex 5, France
teisseire@teledetection.fr

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 primarily 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 increasingly 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 chapter, firstly we present the sequential pattern discovery problem in a collaborative framework and subsequently enhance the architecture by introducing the context of privacy. Thus we propose to extract sequential patterns from distributed databases while preserving privacy. A salient feature of the proposal is its flexibility and as a result is more pertinent to mining operations for real world applications in terms of efficiency and functionality. Furthermore, under some reasonable assumptions, we prove that the architecture and protocol employed by our algorithm for multi-party computation is secure. Finally we conclude with some trends of current research being conducted in the field.

1 Introduction

The increasing popularity of multi-database technology, such as communication networks and distributed, federated and homogeneous multi-database systems, has led to the development of many large distributed transaction databases for real world applications. However, for the purposes of decision-making, large organizations would need to mine these distributed databases located at disparate locations. Moreover, the Web has rapidly transformed into an information flood, where individuals and organizations can access free and accurate information and knowledge on the Internet while making decisions. Although this large data assists in improving the quality of decisions, it also results into a significant challenge of efficiently identifying quality knowledge from multi-databases [1, 2].

Therefore large corporations might 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. The individual transactional databases contain information regarding the purchasing history of the same set of common customers transacting through e-commerce portals or brick and mortar stores. However, the bigger challenge of such computations is compliance to stringent privacy requirements laid down by the formulation of laws such as HIPAA [3]. These regulatory policies are the driving force behind the growing consciousness towards the protection of privacy of individuals and their data. Consequently, there has been a paradigm shift towards the creation of a privacy-aware infrastructure, which entails aspects ranging from data-collection to analysis [4].

Conventionally, data mining has been applied to the traditional data warehouse model of a central data repository, and conducting analysis on it. However, privacy considerations prevents this generic approach and 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. Indeed, if data mining is to be performed on these sensitive datasets, due attention must be given to the privacy requirements. Recently there has been a spate of work addressing privacy preserving data mining [5, 6]. This wide area of research includes classification techniques [7], association rule mining [8], and clustering [9] with privacy constraints. In early work on privacy-preserving data mining, Lindell and Pinkas [10] 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 [11], where a solution to the association rule mining problem for the case of two parties was proposed. In [12], a novel secure architecture has been proposed where the security and accuracy of the data mining results are guaranteed with improved efficiency.

Traditionally, Secure Multi-Party Protocols have been used for the secure computation for generic functions. 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 [13, 14] and was later generalized to multi-party computation (e.g. [15–17]). It has been proved that for any polynomial function, there is a secure multiparty computation solution [16, 17]. 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). However, the complexity of such a secure protocol is prohibitive for complex data mining tasks such as the discovery of sequential patterns.

In this paper, we present an alternative privacy preserving data mining approach - PriPSeP, for discovering sequential patterns in the local databases of a large integrated organization. PriPSeP is useful for mining sequential patterns via collaboration between disparate parties, employing secure architecture, operations and underlying protocols. Hence, to counter the communication and bandwidth overhead of the Oblivious Transfer required between two-parties in an SMC, this paper proposes an alternate architecture consisting of "semi-honest" and "non-colluding" sites. This tradeoff between security and efficiency is reasonable as none of the participating sites are privy to the intermediate or the final results of the calculus. Furthermore, due to the uniform random noise in the datasets, private information of any individual is also guarded from any possible leak.

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. Section 3 describes our proposed solution with the description of the architecture and the algorithms for secure multi-party protocols. Finally, Section 4 concludes the paper with a roadmap for future work as well as new trends on privacy preseving sequential pattern mining approaches.

2 Problem Statement

In this section we give the formal definition of the problem of privacy preserving collaborative sequential pattern mining. First, we give a brief overview of the traditional pattern mining problem by summarizing the formal description introduced in [18] and extended in [19]. 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, a transaction time 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 actual 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. The anti-monotonic Apriori property [20] holds for sequential patterns [21].

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 [19], PSP [22], PrefixSpan [23], SPADE [24], FreeSpan[25], SPAM [26], CLOSPAN [27], PRISM [28], SAMPLING [29], etc.).

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. Our data representation scheme considers 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_j^i is defined as follows: $V_j^i = [C_1^{i,j}...C_N^{i,j}]$ 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 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 find all the sequential patterns embedded in the set of databases by considering parties do not want to share their private data sets with each other. In order to illustrate this further, let us consider the following example.

Example 1 *Let us consider an example of three retail franchisees Alice, Bob and Carol wishing to extract securely the sequential patterns 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

CID	**Sequences**
1	$(1)_1^A$ $(2)_2^B$ $(7)_4^C$ $(3)_5^A$
2	$(1)_3^B$ $(2)_4^A$ $(3)_6^C$
3	$(1)_2^C$ $(7)_3^C$ $(2)_6^A$ $(3)_7^A$

Table 2. The union of all databases

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), we obtain the sequence of $< (1)(2)(3) >$*. By considering privacy, this sequence has to be obtained by considering Alice, Bob and Carol do not want to share their private data sets with each other.*

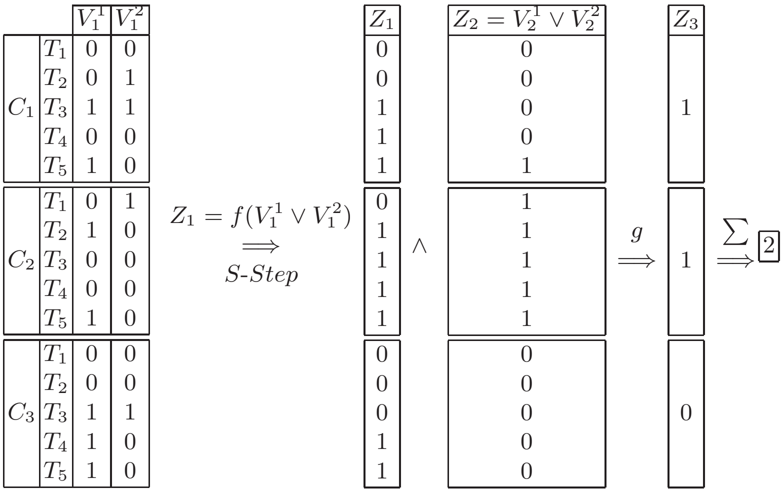
In, [30], Zhan et al. have proposed a novel approach, which entails the transformation of the databases of each collaborating party, followed by a protocol, which results in the preservation of privacy, as well as the correct results. Theoretically, the approach is robust and secure, however, it has serious limitations relating to the initial constraints considered while developing the protocol. It has been assumed that each of the collaborating party carries a unique inventory. For instance, considering our previous example and our problem statement, and following the previous approach and not taking into account the possibility of items being shared among the distributed parties, we come up with erroneous results. An item such as (1) which is not supported by enough customers in one individual database might not appear in the final results. This assumption causes serious limitation for different real applications where items sharing between different databases is imperative and a fundamental requirement as proved earlier. Moreover, the same customer buying the same item twice from the same database but on different times is not permissible, employing their new data representation scheme for sequential data. The other drawback of mapping each item to a unique code is the additional overhead incurred while sorting the databases.

3 The PriPSeP approach

In this section, we propose our novel approach for privacy preserving sequential pattern mining in distributed and collaborative databases. A preliminary version of this proposal has been published in [31]. 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.

3.1 Collaborative sequential pattern mining

An overview As previously seen in Section 2, the main difficulty with collaborative mining is that we have to deal with different databases where the order of items is not known beforehand (e.g. consider the item (7) of the CID 1 in the Bob's database is before the item (3) of the Alice's database).
For brevity, we consider that we are provided with a Data Miner performing the generating and verifying phases of candidate sequences as Apriori-like algorithms. We assume that the candidate generation is performed in conventionally by combining the k-1 frequent sequences in order to generate k-candidate sequences (e.g. C.f. GSP [19] generation phase). We extend the verification phase as follows. First, we consider that our data representation scheme has been extended from the SPAM algorithm [26], wherein for efficient counting, each customer's transactions are represented by a vertical bitmap. These bitmap or vectors are vertically aligned for various computations to calculate the support value for any sequence. As we have to deal with disparate distributed databases, we assume that the Miner could request the N 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 three customers and each customer has five transaction times or CIDs. Let us consider that we are in the candidate step counting of an Apriori-like algorithm. Let us assume that we are currently finding how many times the sequence $<(1)(2)>$ appears in the set of all customers of the two databases. First, we extract from DB_1, the vector corresponding to the item



		V_1^1	V_1^2	Z_1	$Z_2 = V_2^1 \vee V_2^2$	Z_3
C_1	T_1	0	0	0	0	1
	T_2	0	1	0	0	
	T_3	1	1	1	0	
	T_4	0	0	1	0	
	T_5	1	0	1	1	
C_2	T_1	0	1	0	1	1
	T_2	1	0	1	1	
	T_3	0	0	1	1	
	T_4	0	0	1	1	
	T_5	1	0	1	1	
C_3	T_1	0	0	0	0	0
	T_2	0	0	0	0	
	T_3	1	1	0	0	
	T_4	1	0	1	0	
	T_5	1	0	1	0	

Fig. 1. Processing of vectors for collaborative mining

(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) merge the two vectors, and then (ii) transform the result in order to check if they 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 or 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 and due to efficiency considerations, we consider that these two operations are performed through the f function defined below and thus we 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 $\vee$ are bitwise operators. 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 it exists at least one 1 in the customer transactions, i.e. customer dates, or 0 otherwise. We can notice that* $Card(g(V)) = N$.

In conjunction to 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). Subsequently, 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 count for each customer, for the sequence $<(1)(2)>$ has to be calculated. This is performed by the g function, 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, the last operation to be performed is a summation of the number of 1's in the vector Z_3. This is performed by the $\sum$ operation.

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 vectors from the original databases. Then 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. Subsequently, 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 main transfers that are 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.

3.2 From collaborative to privacy-preserving sequential pattern mining

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.

Previous work [16] has described that Secure Multi-Party protocols can be used directly to solve with total security, any generic data mining task. However, the tradeoff 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 for 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.

Evidently traditional approaches do not fulfill the requirements of a complex sequential mining algorithm. 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. These independent sites collect, store and evaluate information securely. PriPSeP requires three *non colluding* and *semi honest* [16] sites which follow the protocol correctly but are free to utilize the information collected by them. They 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 original databases. Its purpose is to interact with NC_1 and NC_2, and it receives the final result of the computation from the PS.
- **Non Colluding Sites** NC_1 **and** NC_2: These symmetric sites collect the noisy data from each database including the Data Miner and perform 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. The operations are described as follows. Initially the following preprocessing steps are performed on the databases individually:

1. Each database $DB_1, DB_2 ... DB_D$ adds ε 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.

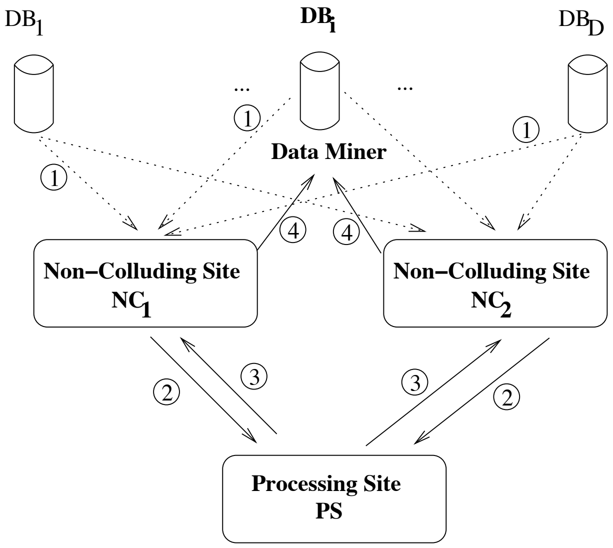


Fig. 2. PriPSep Architecture

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 as previously mentioned in Section 3.1. This step is immediately followed by the counting phase. For simplicity, let us consider that we are 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 transfers, we propose a hypothetical function 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 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 [6], 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 3.1, 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). 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 the initial random noise and then return this final result 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.

Notations In the next subsections, we will consider the following notations. Let $(\overset{+}{X} \mid \overset{-}{X}) \leftarrow h^S(\overset{+}{Y_1} ... \overset{+}{Y_n} \mid \overset{-}{Y_1} ... \overset{-}{Y_n})$ be a tripartite calculation of any function h^S between NC_1, NC_2 and PS where NC_1 owns half of the input $\overset{+}{Y_1} ... \overset{+}{Y_n}$ and gets half of the result $\overset{+}{X}$, and similarly NC_2 owns half of the inputs $\overset{-}{Y_1} ... \overset{-}{Y_n}$ 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 $\overset{+}{Y_1} ... \overset{+}{Y_n}$ to PS and receives the result $\overset{+}{X}$ from PS. Initially, NC_1 transforms its inputs $\overset{+}{Y_1} ... \overset{+}{Y_n}$ to $\overset{+}{Y'_1} ... \overset{+}{Y'_n}$ 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.

```
Algorithm 2: The algorithm $\bigwedge^S$

    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 $A^R \oplus B^R = (\overset{+}{X} \oplus \overset{-}{X}) \bigwedge (\overset{+}{Y} \oplus \overset{-}{Y})$
     1. $NC_1$ and $NC_2$ mutually generate and exchange four random bits $R_1$, $R_2$, $S_1$ and
        $S_2$ such that: $X_1 = \overset{+}{X} \oplus\ R_1$, $Y_1 = \overset{+}{Y} \oplus\ S_1$, $X_2 = \overset{-}{X} \oplus\ R_2$, $Y_2 = \overset{-}{Y} \oplus\ S_2$,
        $R = R_1 \oplus\ R_2$ and $S = S_1 \oplus\ S_2$.
     2. $NC_1$ sends $X_1$ and $Y_1$ to $PS$.
     3. $NC_2$ sends $X_2$ and $Y_2$ to $PS$.
     4. $PS$ calculates $C = (X_1 \oplus X_2) \bigwedge (Y_1 \oplus Y_2)$ and a random bit $R_{PS}$.
     5. $PS$ sends $A_{PS} = C \oplus R_{PS}$ to $NC_1$ and $B_{PS} = R_{PS}$ to $NC_2$ (or vice versa).
     6. $NC_1$ calculates $A^R = A_{PS} \oplus (\overset{+}{X} \bigwedge S) \oplus (\overset{+}{Y} \bigwedge R) \oplus (R \bigwedge S)$
     7. $NC_2$ calculates $B^R = B_{PS} \oplus (\overset{-}{X} \bigwedge S) \oplus (\overset{-}{Y} \bigwedge R)$
```

The $\bigwedge^S$ and $\bigvee^S$ protocols In this section, we define two basic algorithms $\bigwedge^S$ (see Algorithm 2) and $\neg^S$ (see Algorithm 3) providing the protocol which is used to compute securely the bitwise operators from two bits. The $\bigvee^S$ is obtained from the logical equation $A \bigvee B = \neg(\neg A \bigwedge \neg B)$ calculated by using the secure operators $\bigwedge^S$ and $\neg^S$. The 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, $X_2 = \overset{-}{X} \oplus R_2$ and $Y_2 = \overset{-}{Y} \oplus S_2$) to PS, which can calculate the $\bigwedge$ securely. It actually operates on the randomized inputs and calculates $C = (X_1 \oplus X_2) \bigwedge (Y_1 \oplus Y_2)$. It then also adds random noise to the intermediate results in order to avoid NC_1 or NC_2 having the complete 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: $A^R = A_{PS} \oplus (\overset{+}{X} \bigwedge S) \oplus (\overset{+}{Y} \bigwedge R) \oplus (R \bigwedge S)$ could be done securely since it knows its own inputs ($\overset{+}{X}$, $\overset{+}{Y}$, R_1 and S_1) and random numbers from NC_2 (R_2 and S_2). Hence, the final results $A^R \oplus B^R = A_{PS} \oplus (\overset{+}{X} \bigwedge S) \oplus (\overset{+}{Y} \bigwedge R) \oplus (R \bigwedge S) \oplus B_{PS} \oplus (\overset{-}{X} \bigwedge S) \oplus (\overset{-}{Y} \bigwedge R)$. Substituting the value of A_{PS} and B_{PS}, the initial and intermediate random numbers are removed due to the boolean property $R_{PS} \oplus R_{PS} = 0$. The desired result is $\overset{+}{X} \bigwedge \overset{+}{Y} \oplus \overset{+}{X} \bigwedge \overset{-}{Y} \oplus \overset{-}{X} \bigwedge \overset{+}{Y} \oplus \overset{-}{X} \bigwedge \overset{-}{Y}$. Although, this operation is never performed, the symmetrically divided result lies with both NC_1 and NC_1. More importantly, the Processing Site receives no information regarding to the private inputs of any individual. Due to the randomization performed during the initial step, it just sees a stream of uniformly distributed values, and cannot distinguish between a genuine and a random value.

Algorithm 3: The $\neg^S$ protocol

```
Data: $(\overset{+}{X} \mid \overset{-}{X})$ bits are such as $\overset{+}{X}$ owned by $NC_1$, $\overset{-}{X}$ owned by $NC_2$
Result: $(A^R|B^R)$ are such that $A^R \oplus B^R = \neg(\overset{-}{X} \oplus \overset{+}{X}))$.
    1. $NC_1$ calculates $A^R = \neg\, \overset{+}{X}$
    2. $NC_2$ calculates $B^R =\overset{-}{X}$
    Remark: roles of $NC_1$ and $NC_2$ may be exchanged.
```

Theorem 1 *The operand $\bigwedge^S$ 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.

Remark: The privacy theorem is obvious for the $\neg^S$ operator as no data is exchanged.

Complexity: For the $\bigwedge^S$ operator, nine computations have to be performed (6 $\bigotimes$ and 3 $\bigwedge$). As, two more $\neg^S$ operations are needed by the $\bigvee^S$ protocol, we have in total, eleven computations. For each $\bigwedge$, NC_1 and NC_2 exchange 2×2 bits among each other. 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.

The f^S, g^S and $\sum^S$ functions In this section, we extend the f and g functions in order 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

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 $c \in 0..(\mid \overset{+}{X} \mid /K) - 1$ do
    // For each client c
    $(\overset{+}{Y_{K\times c+1}} \mid \overset{-}{Y_{K\times c+1}}) \longleftarrow (0|0)$;
    foreach $i \in 2..K$ do
        $(\overset{+}{Y_{K\times c+1}} \mid \overset{-}{Y_{K\times c+1}}) \longleftarrow \bigvee^S(\overset{+}{Y_{K\times c+i-1}}, \overset{+}{X_{K\times c+i-1}} \mid \overset{-}{Y_{K\times c+i-1}}, \overset{-}{X_{K\times c+i-1}})$;
return $(\overset{+}{Y} \mid \overset{-}{Y})$;
```

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 $c \in 0..(\mid \overset{+}{X} \mid /K) - 1$ do
    // For each client c
    $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \longleftarrow (\overset{+}{X_{K\times c+1}} \mid \overset{-}{X_{K\times c+1}})$;
    foreach $i \in 2..K$ do
        $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \longleftarrow \bigvee^S(\overset{+}{Y_c}, \overset{+}{X_{K\times c+i}} \mid \overset{-}{Y_c}, \overset{-}{X_{K\times c+i}})$;
return $(\overset{+}{Y} \mid \overset{-}{Y})$;
```

Processing Site oblivious of the correct answer.
Similarly, the g^S function (see Algorithm 5) securely computes the existence of at least '1' in the vector of each customer transaction. It replaces the customer vector to either a '0' or a '1' depending on whether the sequence is supported at least once. This function is useful in calculating the support value at the penultimate step of the Algorithm 7.

Remarks: 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: $(\overset{+}{Y_i} \mid \overset{-}{Y_i}) = \bigvee^S(\overset{+}{Z}_{i\times K+K}, \overset{+}{X}_{i\times K+K} \mid \overset{-}{Z}_{i\times K+K}, \overset{-}{X}_{i\times K+K})$.

Algorithm 6: The $\sum^S$ protocol

Data: Vectors of bits $(X_1|X_2)$. X_1 is coming from NC_1 and X_2 is coming from NC_2.

Result: A number which is shared in two parts: $(NB_1|NB_2)$ corresponding to the number of bits at 1 in vectors $(X_1 \bigoplus X_2)$.

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 (X_1, R_1) and (X_2, R_2) using a permutation value φ and get respectively Y_1 and Y_2.
3. NC_1 sends Y_1 to PS and NC_2 sends Y_2 to PS.
4. PS calculates $Y_1 \bigoplus Y2$ and count the number of bits at 1 and gets NB.
5. PS gets a random number R_{PS} and returns $N_1 = NB + R_{PS}$ to NC_1 and $N_2 = NB - R_{PS}$ to NC_2.
6. NC_1 computes $NB_1 = N_1 - N_R$, NC_2 keeps only $NB_2 = N_2$.

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 $(2tN)$. The $N(t+1)$ number of permutations could be neglected if NC_1 and NC_2 have their own generators. Finally between NC_1/NC_2 and PS, $N(t + 1)$ bits are transferred.

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

```
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 $(\overset{+}{R} \mid \overset{-}{R})$;
```

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 2 *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 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. Theorem 1). 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 permutate 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 ε' previously inserted.
Complexity: In the secure protocol, each database has to send $2NK$ data bits instead of NK. Subsequently, each DB has 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 $O(7N)$ for randomizing operations. Hence, it could be bounded by $O(20N)$. Let us now consider the communication complexity of the protocol. Let $p = D \times S \times N \times K$. If the number of operations is at most $O(12p)$, then the number of transfers required is at most $O(4p)$ and for random values it is $O(3p)$. Hence, the whole algorithm has to send at most $O(20p)$.
Remarks: 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 except f^S(it operates on individual customers), could be parallelized. The total overhead incurred by our secure protocol could be easily reduced by a factor of two. We notice, that by considering SMC protocols, no such optimizations are possible, and hence for scalability issues, our alternative approach could be beneficial.

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 only 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 all of these operations reveal no private data of any individual customer from any of the collaborating DBs. Even a succession or sequence of the secure operations remains secure.
- **Overall Security**: During the entire algorithm, no site obtains any additional information beyond of what they are already authorized to learn.

Hence 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 permutation of the lists enable that each site is ignorant of the correct intermediate results as well as the final result.

3.3 Improving the robustness of the system

As described in the previous algorithms, all the data are stored in the two non colluding sites NC_1 and NC_2. If a malicious party gains access to both sites, it will be trivial to obtain all the information and hence violate the tenets of privacy. Thus, in order to improve the robustness of the system, it would be interesting to have more than 2 NC_i (in fact an arbitrary number w) such that the knowledge of the data may only be obtained if one gains access to all the w NC_i and otherwise get no more than random numbers. Furthermore in order to be useful, the complexity must stay linear with w In this section, we describe the secure operators generalized to w non colluding sites NC_i and we focus on the most important protocols: $SEND^S$, $\bigwedge^S$, $\neg^S$ and $\sum^S$.

Sending data to the w NC_i: $SEND^S$ In the original case, for sending a data D to NC_1, NC_2, sites must generate a random number R and send $D \oplus R$ to NC_1 and R to NC_2 (or vice versa). This method could be generalized to w sites NC_i by generating $w-1$ random numbers, by calculating $V_1 = D \oplus R_1$, $V_2 = R_1 \oplus R_2$, ... $V_i = R_{i-1} \oplus R_i$, ... $V_w = R_w$ and then by sending one V_1 to each site NC_j in any order.
As in the original case, each NC_j obtains only random numbers such as $D = V_1 \oplus ... \oplus V_w$. The only way to obtain information on D is by gaining access to all the NC_j. If one has access to all but one, it is analogous to the NC_1 and NC_2 scenario, which has already been proven to be secure.

The $\neg^S$ operation With w sites NC_i, this operation is still analogous and simple as the one when $w = 2$. In order to implement it, it is sufficient that an odd number of the sites (for example only one : NC_1) negates their part of the value, and the other ones does nothing. There is still nothing exchanged and hence still no issues pertaining to privacy.

The $\bigwedge^S$ operation Similar to the case $w = 2$ each site garbles its own part of the data (X_i and Y_i such that the real data is $X = X_1 \oplus ... \oplus X_w$ and $Y = Y_1 \oplus ... \oplus Y_w$) before sending it to PS. To do that they generate two random numbers R_i to encode X_i and S_i to encode Y_i and gets $X'_i = X_i \oplus R_i$ and $Y'_i = Y_i \oplus R_i$ which are sent to PS. They also exchange the value $R = R_1 \oplus ... \oplus R_w$ and $S = R_1 \oplus ... \oplus S_w$ between all NC_i. Then PS will calculate $P = (X'_1 \oplus ... \oplus X'_w) \bigwedge (Y'_1 \oplus ... \oplus y'_w)$ which could be written as $(X \oplus R) \bigwedge (Y \oplus S)$. PS sends its result P to all NC_i by using the $SEND^S$ protocol. Now all NC_i get a value P_i and they only need to remove garbled terms $(X \wedge S)$, $(Y \wedge R)$ and

$(R \wedge S)$. To do that, it is sufficient that an odd number of NC_i (for example : NC_1) perform $Z_i = P_i \oplus (X_i \wedge S) \oplus (Y_i \wedge R) \oplus (S \wedge R)$ and all other ones perform $Z_i = P_i \oplus (X_i \wedge S) \oplus (Y_i \wedge R)$. We then obtain the expected results such that $Z = Z_1 \oplus ... \oplus Z_w = X \wedge Y$.
The number of operation performed by each site (real operations / random numbers generation / data sending and receiving) will increase linearly with the number of NC_i (w) and thus the full secure processing still remains linear compared to the unsecured one.

The $\sum^S$ operation The generalization of the Sum^S algorithm is described in Algorithm 8. Its complexity increases linearly with the number w of NC_i and remains linear compared to the same unsecured process.

Algorithm 8: The $\sum^S$ protocol

```
Data: Vectors of bits $(X_1|...|X_w)$. $X_i$ are coming from $NC_i$ such that $X = X_1 \oplus ... \oplus X_w$
Result: A number which is shared in $w$ parts: $(NB_1|...|NB_w)$ such that $NB = NB_1 + ... + NB_w$ corresponds to the number of bits at 1 in vectors $X$.
let $N = card(X) = card(X_i)$ be the number of bits in vector $X$.

1a One of the sites (for example $NC_1$) will generate $w$ random vectors of bits
   $(R_1...R_n)$ of same size such that $card(R_i) \geq 2 * N$.
1b It calculates $R = R_1 \oplus ... \oplus R_w$ and $Nr$ the number of bits equal to 1 in $R$.
1c R and Nr are sent to all $NC_i$.
1. A permutation $\varphi$ is chosen to permute $card(R) + card(X)$ bits.
2. Each $NC_i$ reorder its vector $(X_i, R_i)$ using the permutation $\varphi$ and gets $Y_i$.
3. Each $NC_i$ sends its $Y_i$ to $PS$.
4. $PS$ calculates $Y = Y_1 \oplus ... \oplus Y_w$ and counts the number of bits at 1 and gets $NB$.
5. $PS$ generates $w - 1$ random numbers $RP_i$ and calculates $N_1 = NB + RP_1$,
   $N_2 = NB + RP_2 - RP_1$ ... $N_{w-1} = NB + RP_{w-1} - RP_{w-2}$ and
   $N_w = NB - RP_{w-1}$.
6. $PS$ sends one of the $N_i$ to each $NC_j$ in any order.
7. $NC_1$ computes $NB_1 = N_1 - N_R$, all other $NC_i$ keeps only $NB_i = N_i$.

Remark: all additions and substraction are done modulo
$card(Y) = card(R) + card(X)$.
```

4 Conclusion

In this paper we have addressed the problem of privacy preserving sequential pattern mining in the context of 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. The proposed secure functions can also be extended to the I-step process, i.e. a list of itemsets instead of items. Furthermore, in the current version of PriPSep, results are directly returned to the DM party. 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 do not have to be recomputed. The storage of results would also be made secure by ensuring that each site has only noisy data or random values.
In addition, as the volume of data increases to a deluge, it becomes increasingly expensive (sometimes impossible) to store all available data before processing them and hence it is necessary to process it ”on the fly” as streams of data. Several new applications directly generate streams of data produced by a large number of sensors (e.g., supermarket transactions, medical data). In order to address this increase of available data, for which the privacy issue could also be very important, new research work is being done to apply data mining methods such as sequential patterns mining directly on the streams without storing them [32].Lastly, as network traffic data becomes more relevant in the context of detection of Internet worms and intrusions by discovering abnormal traffic patterns, recent research is trying to solve the problem whilst preserving privacy of customers [33].
In sum, research in privacy preserving data mining, especially sequential patterns is at an exciting stage, with new papers laying shaping the future for the field.

References

1. X.Wu, Zhang, S.: Synthesizing high-frequency rules from different data sources. IEEE Trans. on Knowledge and Data Engineering **15**(2) (2003) 353–367
2. Zhong, N., Yao, Y., , Ohsuga, S.: Peculiarity oriented multi-database mining. In: Proc. of PKDD 99. (1999) 136–146
3. of Health & Human Services, U.S.D.: Health insurance portability and accountability act of 1996. http://www.hipaa.org/ (August 1996)
4. Bhattacharya, J., Gupta, S., Kapoor, V., Dass, R.: Utilizing network features for privacy violation detection. In: Proc. of Int. Conf. on Communication System Software and Middleware. (January 2006)
5. Pinkas, B.: Cryptographic techniques for privacy preserving data mining. SIGKDD Explorations **4**(2) (2002)
6. Clifton, C., Kantarcioglu, M., Lin, X., Vaidya, J., Zhu, M.: Tools for privacy preserving distributed data mining. SIGKDD Explorations **4**(2) (2003) 28–34
7. Du, W., Han, Y., Chen, S.: Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: Proc. of the Fourth SIAM Int. Conf. on Data Mining. (2004) 222–233
8. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy-preserving mining of association rules. Information Systems **29**(4) (June 2004)

9. Jagannathan, G., Wright, R.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proc. of KDD 05. (august 2005)
10. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology **15**(2) (2002)
11. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proc. of KDD 02. (July 2002)
12. Kantarcioglu, M., Vaidya, J.: 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. Yao, A.C.: Protocols for secure computations. In: Proc. of the 23rd annual IEEE Symposium on Foundations of Computer Science. (1982) 160–164
14. Yao, A.C.: How to generate and exchange secrets. In: Proc. of the 27th Symposium on Foundations of Computer Science (FOCS). (1986) 162–167
15. Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: A review and open problems. In: New Security Paradigms Workshop, Cloudcroft, USA (2001) 11–20
16. Goldreich, O.: Secure multi-party computation. Working Draft (2000)
17. Chaum, D., Crepeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proc. of the 20th Annual Symposium on the Theory of Computing (STOC). (1988) 11–19
18. Agrawal, R., R.Srikant: Mining sequential patterns. In: Proc. of ICDE 95. (1995) 3–14
19. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Proc. of EDBT 96. (1996) 3–17
20. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large database. In: Proc. of ACM SIGMOD 93. (1993) 207–216
21. Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: Proc. of CIKM 02, MCLean, USA (2002) 18–25
22. Masseglia, F., Cathala, F., Poncelet, P.: The PSP approach for mining sequential patterns. In: Proc. of PKDD 98. (1998)
23. Pei, J., Han, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns efficiently by prefix projected pattern growth. In: Proc. of ICDE 01. (2001)
24. Zaki, M.: SPADE: An efficient algorithm for mining frequent sequences. Machine Learning Journal **42**(1) (February 2001) 31–60
25. Han, J., Pei, J., Mortazavi-asl, B., Chen, Q., Dayal, U., Hsu, M.: Freespan: Frequent pattern-projected sequential pattern mining. In: Proc. of KDD 00. (2000) 355–359
26. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using bitmap representation. In: Proc. of KDD 02. (2002) 439–435
27. Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large datasets. In: In SDM. (2003) 166–177
28. Gouda, K., Hassaan, M., Zaki, M.J.: Prism: A primal-encoding approach for frequent sequence mining. In: ICDM'07. (2007) 487–492
29. Raissi, C., Poncelet, P.: Sampling for sequential pattern mining: From static databases to data streams. In: ICDM'07. (2007) 631–636
30. Zhan, J., Chang, L., Matwin, S.: Privacy-preserving collaborative sequential pattern mining. In: Workshop on Link Analysis, Counter-terrorism, and Privacy in conjunction with SIAM Int. Conf. on Data Mining, Lake Buena Vista, Florida (2004) 61–72

31. Kapoor, V., Poncelet, P., Trousset, F., Teisseire, M.: Privacy preserving sequential pattern mining in distributed databases. In: 15th ACM International Conference on Information and Knowledge Management, CIKM 06, Arlington, USA (2008)
32. Huang, Q.H.: Privacy preserving sequential pattern mining in data stream. In: 4th International Conference on Intelligent Computing, ICIC 2008, Shanghai, China (2008)
33. Seung-Woo, K., Sanghyun, P., Jung-Im, W., Sang-Wook, K.: Privacy preserving data mining of sequential patterns for network tra?c data. Information Sciences: an International Journal **178**(3) (2008) 694–713