Multiparty Protocol for Mining of Association Rules in Horizontally Partitioned Database

Satish R[1], Soundarya R[2], Vidya N[3]
[1-3]Departmet of Computer Science and Engineering,
RajaRajeswari College of Engineering,
Bangalore, India.

Janaki K[4]
[4]Asst. Professor, Department of Computer Science and Engineering,
RajaRajeswari College of Engineering,
Bangalore, India.

***Abstract*—In this paper, propose a protocol for secure mining of association rules in horizontally distributed databases. The current leading protocol is K and C protocol. This protocol is based on the Fast Distributed Mining(FDM) algorithm which is an unsecured version. We want to release aggregate information about the data, without leaking individual's information.The main ingredients in this protocol are two novel secure multi-party algorithms, first one computes the union of private subsets that each of the interacting players hold, and second tests the inclusion of an element held by one player in a subset held by another. This protocol offers enhanced privacy with respect to the other one.**

Keywords— ***Privacy Preserving Data Mining; Frequent Itemsets; Association Rules.***

I. INTRODUCTION

Data mining is a power new technology has emerged as a means of identifying patterns and trends from large quantities of data and it is also a process of digging through and analyzing enormous sets of data and then extracting the meaning of the data. Data mining and data warehousing go hand-in-hand: most tools operate by gathering all data into a central site, then running an algorithm against that data present in the central site. However, for privacy concerns can prevent building a centralized warehouse – data may be distributed among several customers, none of which are allowed to transfer their data to another site.

This concept addresses the problem of computing association rules within such a scenario. We assume homogeneous databases where all sites have the same schema, but each site has information on different entities. The main goal is to produce association rules that hold globally, while reducing the information shared about each site.

The problem is that insurance companies will be concerned about sharing this data. Not only must the privacy of patient records be maintained, but insurers will be unwilling to release rules pertaining only to them. Imagine a rule indicating a high rate of complications with a particular medical procedure. If this rule doesn't hold globally, the insurer would like to know this – they can then try to pinpoint the problem with their policies and improve patient care. If the fact that the insurer's data supports this rule is revealed, the insurer could be exposed to significant public relations or liability problems. This potential risk could exceed their own perception of the benefit of participating in the CDC study.

We study here the problem of secure mining of association rules in horizontally distributed databases. In that setting, there are several players that hold homogeneous databases, i.e., databases that share the same schema but hold information on different entities. The goal is to find all association rules with support at least s and confidence at least c, for some given minimal support size s and confidence level c, that hold in the unified database, while reducing the information disclosed about the private databases held by those players. The information that we would like to protect in this context is not only individual transactions in the different databases, but also information should available globally such as what association rules are supported and used locally in each of those databases. That goal defines a problem of secure multi-party computation. In such problems, there are M players that hold private inputs, $x1, \ldots, xM$, and they wish to securely compute $y = f(x1, \ldots, xM)$ for some public function f. If trusted third party is used, the players could give their inputs and he would perform the function evaluation and send to them the resulting output. If the trusted third party is not used, it is needed to devise a protocol that the players can run on their own in order to get the required output y. Such a protocol is considered perfectly secure if no player can learn from his view of the protocol more than what he would have learnt in the idealized setting where the computation is carried out by a trusted third party. In our problem, the inputs are the partial databases, and the required output is the list of association rules that hold in the unified database with support and confidence no smaller than the given thresholds s and c, respectively. As the above mentioned generic solutions rely upon a description of the function f as a Boolean circuit, they can be applied only to small inputs and functions which are realizable by simple circuits. In more complex settings, such as ours, other methods are required for carrying out this computation. In such cases, some relaxations of the

notion of perfect security might be inevitable when looking for practical protocols, provided that the excess information is deemed benign. Kantarcioglu and Clifton studied that problem in [4] and devised a protocol for its solution. The main part of the protocol is a sub-protocol for the secure computation of the union of private subsets that are held by the different players. That is the most costly part of the protocol and its implementation relies upon cryptographic primitives such as commutative encryption, oblivious transfer, and hash functions. This is also the only part in the protocol in which the players may extract from their view of the protocol information on other databases, beyond what is implied by the final output and their own input. While such leakage of information renders the protocol not perfectly secure, the perimeter of the excess information is explicitly bounded in [4] and it is argued there that such information leakage is innocuous, whence acceptable from a practical point of view. Herein we propose an alternative protocol for the secure computation of the union of private subsets. The proposed protocol improves upon that in [4] in terms of simplicity and efficiency as well as privacy. In particular, our protocol does not depend on commutative encryption and oblivious transfer. While our solution is still not perfectly secure, it leaks excess information only to a small number (three) of possible coalitions, unlike the protocol of [4] that discloses information also to some single players. In addition, we claim that the excess information that our protocol may leak is less sensitive than the excess information leaked by the protocol of [4]. The protocol that we propose here computes a parameterized family of functions, which we call threshold functions, in which the two extreme cases correspond to the problems of computing the union and intersection of private subsets. Those are in fact general-purpose protocols that can be used in other contexts as well. Another problem of secure multiparty computation that we solve here as part of our discussion is the set inclusion problem; namely, the problem where Alice holds a private subset of some ground set, and Bob holds an element in the ground set, and they wish to determine whether Bob's element is within Alice's subset, without revealing to either of them information about the other party's input beyond the above described inclusion.

II. RELATED WORK

In privacy preserving data mining has considered two related settings. One is data owner and other is data miner are the different two entities, and another, in which the data is distributed among several parties who aim to jointly perform data mining on the unified corpus of data that they hold.

The first main goal is to protect the data records from the data miner. Hence, the data owner aims at anonymizing the data prior to its release. The main approach in this context is to apply data perturbation .The idea is that the perturbed data can be used to infer general trends in the data, without revealing original record information.

In the second setting, the goal is to perform data mining while protecting the data records of each of the data owners from the other data owners. This is a problem of secure multiparty computation. The usual approach here is cryptographic rather than probabilistic.

III. FAST DISTRIBUTED MINING ALGORITHM

The protocol, as well as ours, is based on the Fast Distributed Mining (FDM) algorithm. Which is an unsecured distributed version of the Apriori Algorithm. Its main idea is that any *s*-frequent itemset must be also locally *s* frequent in at least one of the sites. Hence, in order to find all globally *s*-frequent itemsets, each player reveals his locally *s*-frequent itemsets and then the players check each of them to see if they are *s*-frequent also globally.

The FDM algorithm proceeds as follows:

(1) Initialization: It is assumed that the players have already jointly calculated $Fk-1\ s$. The goal is to proceed and Calculate Fks .

(2) Candidate Sets Generation: Each player Pm computes the set of all $(k-1)$-itemsets that are locally frequent in his site and also globally frequent; namely, Pm computes the set $Fk-1,ms \cap Fk-1s$. He then applies on that set the Apriori algorithm in order to generate the set Bk, ms of candidate k-itemsets.

(3) Local Pruning: For each $X \in Bk,ms$, Pm computes $suppm(X)$. He then retains only those itemsets that are locally *s*-frequent. We denote this collection of itemsetsby Ck,ms .

(4) Unifying the candidate itemsets: Each player broadcasts his Ck,ms and then all players compute $Ck \cup s := Mm=1\ Ck,ms$.

(5) Computing local supports. All players compute the local supports of all itemsets in $Ck\ s$.

(6) Broadcast Mining Results: Each player broadcasts the local supports that he computed. From that, everyone can compute the global support of every itemset in Cks . Finally,$Fk\ s$ is the subset of Cks that consists of all globally *s*frequent *k*-itemsets.In the first iteration, when $k = 1$, the set $C1,m\ s$ that the *m*thplayer computes (Steps 2-3) is just $F1,ms$, namely, the set of single items that are *s*-frequent in Dm. The complete FDM algorithm starts by finding all single items that are globally *s*-frequent. It then proceeds to find all 2-itemsets that are globally *s*-frequent, and so forth, until it finds the longest globally *s*-frequent itemsets. If the length of such itemsets is K, then in the $(K+1)$th iteration of the FDM it will find no $(K + 1)$-itemsets that are globally *s*-frequent, in which case it terminates.

A running example

Let D be a database of N = 18 itemsets over a set of L = 5 Items, A = {1, 2, 3, 4, 5}. It is partitioned between M = 3 Players and the corresponding partial databases are:
D1 = {12, 12345, 124, 1245, 14, 145, 235, 24, 24}
D2 = {1234, 134, 23, 234, 2345}
D3 = {1234, 124, 134, 23}.
For example, D1 includes N1 = 9 transactions, the third of Which (in lexicographic order) consists of 3 items — 1, 2 and 4. Setting s = 1/3, an itemset is s-frequent in D if it is supported by at least 6 = sN of its transactions. In this case,

F1s = {1, 2, 3, 4}
F2s = {12, 14, 23, 24, 34}
F3s = {124}

F4s = F5s = ∅, and Fs = F1s ∪ F2s ∪ F3s.

For example, the itemset 34 is indeed Globally s-frequent since it is contained in 7 transactions of D. However, it is locally s-frequent only in D2 and D3.

In the first round of the FDM algorithm, the three players Compute the sets C1,ms of all 1-itemsets that are locally Frequent at their partial databases:
C1^1s = {1, 2, 4, 5} , C1^2s = {1, 2, 3, 4} , C1^3s = {1, 2, 3, 4} .
Hence, C1^s = {1, 2, 3, 4, 5}.
Consequently, all 1-itemsets have to be checked for being globally frequent; that check Reveals that the subset of globally s-frequent 1-itemsets is
F1^s = {1, 2, 3, 4}.
In the second round, the candidate itemsets are:
C2^1,s = {12, 14, 24} C2^2s = {13, 14, 23, 24, 34}
C2 ^3,s = {12, 13, 14, 23, 24, 34} .
(Note that 15, 25, 45 are locally s-frequent at D1 but they are not included in C2^1s since 5 was already found to be globally infrequent.)
Hence, C2^s = {12, 13, 14, 23, 24, 34}.
Then, after veryfing global frequency, we are left with
F2^s ={12, 14, 23, 24, 34}.In the third round, the candidate itemsets are:C3^1s = {124} , C3^2s = {234} , C3^3s = {124} .
So, C3^s = {124, 234} and, then, F3^s = {124}. There are no more frequent itemsets.

IV. OVERVIEW AND ORGANIZATION OF PAPER

The FDM algorithm violates privacy in two stages: In Step 4, where the players broadcast the itemsets that are locally frequent in their private databases, and in Step 6, where they broadcast the sizes of the local supports of candidate itemsets. Our improvement is with regard to the secure implementation of Step 4, which is the more costly stage of the protocol, and the one in which the protocol of leaks excess information. In Section 2 we describe secure implementation of Step 4.We then describe our alternative implementation and proceed to analyze the two implementations in terms of privacy and efficiency and compare them. We show that our protocol offers better privacy and that it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost. In Sections 3 and 4 we discuss the implementation of the two remaining steps of the distributed protocol: The identification of those candidate itemsets that are globally *s* frequent, and then the derivation of all (*s*, *c*)-association rules.

V. DISTRIBUTED DATABASE

A distributed database is database in which storage devices are not all attached to a common processing unit such as the CPU, controlled by a distributed database management system (together sometimes called a distributed database system). It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components. System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication.

1. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time- consuming depending on the size and number of the distributed database. This process also requires lot of time and computer resources.
2. Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten. Both replication and duplication can keep the data current in all distributive locations.

VI. ASSOCIATION RULE

In Data mining, association rule is a popular and well researched method for discovering interesting relations between variables in large databases. Piatetsky-shapiro describes analyzing & presenting strong rules discovered in databases using different measures of interestingness. Based on the concept of strong rules, Agrawal et al introduced
association rules for discovering regularities between products in large scale transaction data recorded by point-of sale (POS) systems in supermarkets For example, the rule Found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, he or she is likely to also buy beef. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements. In addition to the above example from market basket analysis

VII. CONCLUSION

We proposed a protocol for secure mining of association rules in horizontally distributed databases that improves significantly upon the current leading protocol [18] in terms of privacy and efficiency. One of the main ingredients in our proposed protocol is a novel secure multi-party protocol for computing the union (or intersection) of private subsets that each of the interacting players hold. Another ingredient is a protocol that tests the inclusion of an element held by one player in a subset held by another. Those protocols exploit the fact that the underlying problem is of interest only when the number of players is greater than two.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.
[3] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In *CCS*, pages 257–266, 2008.
[4] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16:1026–1037, 2004.