A Survey on Efficient Protocol for Privacy Preserving Association Rule Mining

Miss Prajakta Jaswante, Dr. P. R. Deshmukh

***Abstract*- We propose a protocol for secure mining of association rules in horizontally distributed databases. The current leading protocol is that of Kantarcioglu and Clifton [18]. Our protocol, like theirs, is based on the Fast Distributed Mining (FDM) algorithm of Cheung et al. [8], which is an unsecured distributed version of the Apriori algorithm. The main ingredients in our protocol are two novel secure multi-party algorithms — one that computes the union of private subsets that each of the interacting players hold, and another that tests the inclusion of an element held by one player in a subset held by another. Our protocol offers enhanced privacy with respect to the protocol in [18]. In addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.**

***Index Terms*-Privacy Preserving Data Mining; Distributed Computation; Frequent Itemsets; Association Rules.**

I. INTRODUCTION

We study here the problem of secure mining of association rules in horizontally partitioned databases. In that setting, there are several sites (or 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 minimizing 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 more global information such as what association rules are supported 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, $x1, \ldots, x_M$, and they wish to securely compute $y = f(x1, \ldots, x_M)$ for some public function f. If there existed a trusted third party, the players could surrender to him their inputs and he would perform the function evaluation and send to them the resulting output. In the absence of such a trusted third party, it is needed to devise a protocol that the players can run on their own in order to arrive at 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. Yao [1] was the first to propose a generic solution for this problem in
the case of two players. Other generic solutions, for the multi-party case, were later proposed [2, 3, 4].

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 [5] 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. (The private subset of a given player, as we explain below, includes the itemsets that are s-frequent in his partial database.) 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 [5] 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 [5] 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 may leak excess information only to a small number (three) of possible coalitions, unlike the protocol of [5] 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 [5].

The protocol that we propose here may computes a parameterized family of functions, which we call threshold functions, in which the two extreme cases may 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 want solve 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. LITERATURE REVIEW AND RELATED WORK

Previous work in privacy preserving data mining has considered two related settings. One, in which the data owner and the data miner are two different 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.

In the first setting, the 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 [6], [7]. 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. Lindell and Pinkas [8] showed how to securely build an ID3 decision tree when the training set is distributed horizontally. Lin et al. [9] discussed secure clustering using the EM algorithm over horizontally distributed data. The problem of distributed association rule mining was studied in [10], [11], [12] in the vertical setting, where each party holds a different set of attributes, and in [6] in the horizontal setting. Also the work of [13] considered this problem in the horizontal setting, but they considered large-scale systems in which, on top of the parties that hold the data records (resources) there are also managers which are computers that assist the resources to decrypt messages; another assumption made in [13] that distinguishes it from [6] and the present study is that no collusions occur between the different network nodes — resources or managers.

III. PROPOSED OBJECTIVES

- Propose modified protocol for privacy preservation
- Find parameterized family of functions
- Performance evaluation of proposed algorithm

IV. PRELIMINARIES

Let D be a transaction database. As in [5], we view D as a binary matrix of N rows and L columns, where each row is a transaction over some set of items, $A = \{a1, \ldots, a_L\}$, and each column represents one of the items in A. (In other words, the (i, j)th entry of D equals 1 if the ith transaction includes the item aj , and 0 otherwise.) The database D is partitioned horizontally between M players, denoted $P_1, \ldots, P_M$. Player Pm holds the partial database Dm that contains $Nm = |Dm|$ of the transactions in D, $1 \leq m \leq M$. The unified database is $D = D1 \cup \ldots \cup D_M$, and it includes $N := \Sigma^{M}_{m=1} Nm$ transactions.

An itemset X is a subset of A. Its global support, $supp(X)$, is the number of transactions in D that contain it. Its local support, $suppm(X)$, is the number of transactions in Dm that contain it. Clearly, $supp(X) = \Sigma^{M}_{m=1} suppm(X)$. Let s be a real number between 0 and 1 that stands for a required support threshold. An itemset X is called s-frequent if $supp(X) \geq sN$. It is called locally s-frequent at Dm if $suppm(X) \geq sNm$. For each $1 \leq k \leq L$, let F^{k}_{s} denote the set of all k-itemsets (namely, itemsets of size k) that are s-frequent, and $F^{k,m}{}_{s}$ be the set of all k-itemsets that are locally s-frequent at Dm, $1 \leq m \leq M$. Our main computational goal is to find, for a given threshold support $0 < s \leq 1$, the set of all s frequent itemsets, $Fs := \cup L$ $k{=}1$ F^{k}_{s} . We may then continue to find all (s, c)-association rules, i.e., all association rules of support at least sN and confidence at least c. (Recall that if X and Y are two disjoint subsets of A, the support of the corresponding association rule $X \Rightarrow Y$ is $supp(X \cup Y)$ and its confidence is $supp(X \cup Y)/supp(X)$.)

V. ANALYSIS OF PROBLEM

A. The Fast Distributed Mining Algorithm

The protocol is based on the Fast Distributed Mining (FDM) algorithm of Cheung et al. 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 $F^{k-1}{}_{s}$. The goal is to proceed and calculate $F^{k}{}_{s}$.

(2) **Candidate Sets Generation:** Each player P_m computes the set of all $(k - 1)$-itemsets that are locally frequent in his site and also globally frequent; namely, Pm computes the set $F^{k-1,m}{}_{s} \cap F^{k-1}{}_{s}$. He then applies on that set the Apriori algorithm in order to generate the set $B^{k,m}{}_{\mathrm{s}}$ of candidate k-itemsets.

(3) **Local Pruning:** For each $X \in B^{k,m}{}_{s}$, P_m computes $suppm(X)$. He then retains only those itemsets that are locally s-frequent. We denote this collection of itemsets by $C^{km}{}_{\mathrm{s}}$

(4) **Unifying the candidate itemsets:** Each player broadcasts his $C^{km}{}_{\mathrm{s}}$ and then all players compute $C^{k} := \mathrm{U}^{M}{}_{m=1} C^{km}{}_{\mathrm{s}}$

(5) **Computing local supports:** All players compute the local supports of all itemsets in C^{k}_{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 C^{k}_{s} . Finally, $F^{k}{}_{s}$ is the subset of C^{k}_{s} that consists of all globally $s$frequent k-itemsets.

In the first iteration, when $k = 1$, the set $C^{1,m}{}_{s}$ that the mth player computes (Steps 2-3) is just $F^{1,m}{}_{s}$, 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.

B. Disadvantages of Existing System

- Insufficient security
- Leaks more secure information

VI. PROPOSED SYSTEM AND MODULES:

The FDM algorithm violates privacy in two stages: In Step 4, where the players broadcast the itemsets those are locally frequent in their private databases, and in Step 6, where they broadcast the sizes of the local supports of candidate itemsets. Kantarcioglu and Clifton [5] proposed secure implementations of those two steps. 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 this Section we describe Kantarcioglu and Clifton's secure implementation of Step 4. We will be implementing following modules:

A. Registration:

This is module mainly designed to provide the authority to a user in order to access the other modules of the project. Here a user can have the accessibility authority after the registration. After registration each user has given a unique password.

B. Privacy Preserving Data Mining:

Protocol 1: Unifying lists of locally Frequent Itemsets – Kantarcioglu and Clifton (UNIFI-KC)

Protocol UNIFI-KC works as follows: First, each player adds to his private subset $C^{km}{}_s$ fake itemsets, in order to hide its size. Then, the players jointly compute the encryption of their private subsets by applying on those subsets a commutative encryption1, where each player adds, in his turn, his own layer of encryption using his private secret key. At the end of that stage, every itemset in each subset is encrypted by all of the players; the usage of a commutative encryption scheme ensures that all itemsets are, eventually, encrypted in the same manner. Then, they compute the union of those subsets in their encrypted form. Finally, they decrypt the union set and remove from it itemsets which are identified as fake.

Protocol 2: (THRESHOLD) Secure computation of the t threshold function

Input: Each player P_m has an input binary vector $\mathbf{b}_m \in Z^n{}_2$, $1 \le m \le M$.
Output: $\mathbf{b} := T_t(\mathbf{b}_1, \ldots, \mathbf{b}_M)$.

```
1: Each $P_m$ selects $M$ random share vectors $\mathbf{b}_{m,\ell} \in Z^n{}_{M+1}, 1 \le \ell \le M$,
such that $\Sigma^M{}_{\ell=1}\, \mathbf{b}_{m,\ell} = \mathbf{b}_m \bmod (M + 1)$.
2: Each $P_m$ sends $\mathbf{b}_{m,\ell}$ to $P_\ell$ for all $1 \le \ell \ne m \le M$.
3: Each $P_\ell$ computes $\mathbf{s}_\ell = (s_\ell(1), \ldots, s_\ell(n)) := \Sigma^M{}_{m=1}\, \mathbf{b}_{m,\ell} \bmod (M + 1)$.
4: Players $P_\ell$, $2 \le \ell \le M - 1$, send $\mathbf{s}_\ell$ to $P1$.
5: $P_1$ computes $\mathbf{s} = (s(1), \ldots, s(n)) := \sum_{\ell=1}^{M-1} \mathbf{s}_\ell \bmod (M + 1)$.
6: for $i = 1, \ldots, n$ do
7: If $(s(i) + s_M(i)) \bmod (M + 1) < t$ set $b(i) = 0$ otherwise set $b(i) = 1$.
8: end for
9: Output $\mathbf{b} = (b(1), \ldots, b(n))$.
```

Let $P_1, \ldots, P_M$ be M players where P_m has an input binary vector $\mathbf{b}_m \in Z_n{}^2$, $1 \le m \le M$. Protocol 2 (to which we refer as *THRESHOLD* henceforth) computes, in a secure manner, the output vector $\mathbf{b} := T_t(\mathbf{b}_1, \ldots, \mathbf{b}_M)$, for some $1 \le t \le M$. Let $\mathbf{a} = (a(1), \ldots, a(M)) := \Sigma^M{}_{m=1}\, \mathbf{b}_m$ be the sum of the input binary vectors. Since $a(m) \in Z_{M+1} = \{0, 1, \ldots, M\}$, for all $1 \le m \le M$, the sum vector $\boldsymbol{a}$ may be seen as a vector in $Z^n{}_{M+1}$. The main idea behind the protocol is to use the secure summation protocol of [6] in order to compute shares of the sum vector $\boldsymbol{a}$ and then use those shares to securely verify the threshold conditions in each component.

Protocol 3 : (SETINC) Set Inclusion computation

Input: P_1 has a vector $\mathbf{s} = (s(1), \ldots, s(n))$ and P_M has a vector $\Theta = (\Theta(1), \ldots, \Theta(n))$, where for all $1 \le i \le n$, $s(i) \in \Omega$ and $\Theta(i) \subseteq \Omega$ for some ground set Ω.
Output: The vector $\mathbf{b} = (b(1), \ldots, b(n))$ where $b(i) = 0$ if $s(i) \in \Theta(i)$ and $b(i) = 1$
otherwise, $1 \le i \le n$.

```
1: $P_1$ and $P_M$ agree on a keyed-hash function $h_K(\cdot)$ and on a secret key $K$.
2: $P1$ computes $\mathbf{s}' = (s'(1), \ldots, s'(n))$, where $s'(i) = h_K(i, s(i))$, $1 \le i \le n$.
3: $P_M$ computes $\Theta' = (\Theta'(1), \ldots, \Theta'(n))$, where $\Theta'(i) = \{h_K(i, \theta) : \theta \in \Theta(i)\}$, $1 \le i \le n$.
4: $P_1$ sends to $P_2$ the vector $\mathbf{s}'$.
5: $P_M$ sends to $P_2$ the vector $\Theta'$ in which each $\Theta(i)$ is randomly permuted.
6: For all $1 \le i \le n$, $P_2$ sets $b(i) = 0$ if $s'(i) \in \Theta'(i)$ and $b(i) = 1$ otherwise.
7: $P_2$ broadcasts the vector $\mathbf{b} = (b(1), \ldots, b(n))$.
```

The protocol starts with players P_1 and P_M agreeing on a keyed hash function $h_K(\cdot)$ (e.g., HMAC [4]), and a corresponding secret key K (Step 1). Consequently (Steps 2-3), P_1 converts his sequence of elements s = $(s(1), \ldots, s(n))$ into a sequence of corresponding "signatures" s′ = $(s'(1), \ldots, s'(n))$, where $s'(i) = h_K(i, s(i))$ and P_M does a similar conversions to the subsets that he holds. Then, in Steps 4-5, P_1 sends s′ to P_2, and P_M sends to P_2 the subsets $\Theta'(i)$, $1 \le i \le n$, where the elements within each subset are randomly permuted. Finally (Steps 6-7), P_2 performs the relevant inclusion verifications on the signature values. If he finds out that for a given $1 \le i \le n$, $s'(i) \in \Theta'(i)$, he may infer, with high probability, that $s(i) \in \Theta(i)$, whence he sets $b(i) = 0$. If, on the other hand, $s'(i) \notin \Theta'(i)$, then, with certainty, $s(i) \notin \Theta(i)$, and thus he sets $b(i) = 1$.

Protocol 4 (UNIFI) Unifying lists of locally Frequent Itemsets

Input: Each player P_m has an input subset $C_s^{k,m} \subseteq Ap(F_s^{k-1})$, $1 \le m \le M$.
Output: $C_s^{k} = \cup_{m=1}{}^{M}\, C_s^{k,m}$

```
1: Each player $P_m$ encodes his subset $C_s^{k,m}$ as a binary vector $\mathbf{b}_m$ of length $n_k$ =
$|Ap(F^s{}_k-1)|$, in accord with the agreed ordering of $Ap(F^s{}_k-1)$
2: The players invoke Protocol THRESHOLD-C to compute $\mathbf{b} = T_1(\mathbf{b}_1, \ldots, \mathbf{b}_M)$ =
$\vee_{m=1}{}^{M}\, \mathbf{b}_m$.
3: $C_s^{k}$ is the subset of $Ap(F^s{}_k-1)$ that is described by $\mathbf{b}$.
```

C. Distributed Computation:

This Module explains clearly about the Distributed computation, we will execute the unification step using Protocol UNIFI-KC, in the second implementation we will be using our Protocol UNIFI, where the keyed-hash function was HMAC. In both implementations, we will implement FDM algorithm in the secure manner that is described earlier. We will test two implementations with respect to three measures:

i. Total computation time of the complete protocols over all players. That measure includes the Apriori computation time, and the time to identify the globally *s*-frequent item sets

ii. Total computation time of the unification protocols only over all players.

iii. Total message size.

D. *Frequent Item sets:*

In this Module Frequent Item sets the data that are used frequently are combined and the user can directly access the data that are frequently used in the horizontally distributed database without assessing the database the privacy setting are given by using the protocol.

VII. CONCLUSION

We proposed a protocol for secure mining of association rules in horizontally distributed databases that improves significantly upon the current leading protocol 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.

One research problem that this study suggests was; namely, to devise an efficient protocol for inequality verifications that uses the existence of a semihonest third party. Such a protocol might enable to further improve upon the communication and computational costs of the second and third stages of the protocol .Other research problems that
this study suggests is the implementation of the techniques presented here to the problem of distributed association rule mining in the vertical setting, the problem of mining generalized association rules, and the problem of subgroup discovery in horizontally partitioned data.

VIII. REFERENCES

[1] A.C. Yao. Protocols for secure computation. In *FOCS*, pages 160–164, 1982.

[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC*, pages 503–513, 1990.

[3] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In *CCS*, pages 257–266, 2008.

[4] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[5] 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.

[6] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.

[7] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002.

[8] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Crypto*, pages 36–54, 2000.

[9] X. Lin, C. Clifton, and M.Y. Zhu. Privacy-preserving clustering with distributed EM mixture modeling. *Knowl. Inf. Syst.*, 8:68–81, 2005.

[10] M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. In *PAKDD*, pages 515–524, 2009.

[11] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[12] J. Zhan, S. Matwin, and L. Chang. Privacy preserving collaborative association rule mining. In *Data and Applications Security*, pages 153– 165, 2005.

[13] A. Schuster, R. Wolff, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *CCGRID*, pages 411–418, 2004.