Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.1079 – 1082

RESEARCH ARTICLE

Secure Mining of Association Rules in Horizontally Distributed Databases

Chitteni Siva[1], Selvi[2]

[1]PG Scholar, Department of Computer Science and Engineering, R.M.K Engineering College, Chennai, India

[2]Assistant Professor, Department of Computer Science and Engineering, R.M.K Engineering College, Chennai, India

chittenisiva27@gmail.com [1], yaminianitha@yahoo.co.in [2]

Abstract— The main aim of this project is protocol for secure mining of association rules in horizontally distributed databases. The current leading protocol is that of Kantarcioglu and Clifton. Our protocol, like theirs, is based on the Fast Distributed Mining (FDM) algorithm of Cheung et al., 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. In addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.

Keywords— Privacy Preserving Data Mining; Distributed Computation; Frequent Item sets; 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, . . . , xM, and they wish to securely compute y = f(x1, . . . , xM) 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 [32] was the first to propose a generic solution for this problem in the case of two players.

- 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 most costly part of the protocol and its implementation relies upon cryptographic primitives such as commutative encryption, oblivious transfer, and hash functions.
- In particular, our protocol does not depend on commutative encryption and oblivious transfer (what simplifies it significantly and contributes towards much reduced communication and computational costs). While our solution is still not perfectly secure, it leaks excess information only to a small number (three) of possible coalitions.

II. BACKGROUND

The FDM algorithm violates privacy in two stages: In, where the players broadcast the item sets that are locally frequent in their private databases, and in, where they broadcast the sizes of the local supports of candidate item sets. Kantarcioglu and Clifton [proposed secure implementations of those two steps.

A. Detailed description

In Phase 0 , the players select the needed cryptographic primitives: They jointly select a commutative cipher, and each player selects a corresponding private random key. In addition, they select a hash function h to apply on all itemsets prior to encryption. It is essential that h will not experience collisions on Ap(Fk−1s) in order to make it invertible on Ap(Fk−1s). Hence, if such collusions occur (an event of a very small probability), a different hash function must be selected. At the end, the players compute a lookup table with the hash values of all candidate itemsets in Ap(Fk−1s); that table will be used later on to find the preimage of a given hashvalue.
In Phase 1 (Steps 6-19), all players compute a composite encryption of the hashed sets Ck,m s , 1 ≤ m ≤ M. First (Steps 6-12), each player Pm hashes all itemsets in Ck,ms and then encrypts them using the key Km. (Hashing is needed in order to prevent leakage of algebraic relations between itemsets, see [18, Appendix].) Then, he adds to the resulting set faked itemsets until its size becomes |Ap(Fk−1 s)|, in orderto hide the number of locally frequent itemsets that he has.(Since Ck,ms ⊆ Ap(Fk−1s), the size of Ck,ms is bounded by |Ap(Fk−1s)|, for all 1 ≤ m ≤ M.) We denote the resulting set by Xm. Then (Steps 13-19), the players start a loop of M − 1 cycles, where in each cycle they perform the following operation: Player Pm sends a permutation of Xm to the next player Pm+1; Player Pm receives from Pm−1 a permutation of the set Xm−1 and then computes a new Xm as Xm = EKm(Xm−1). At the end of this loop, Pm holdsan encryption of the hashed Ck,m+1 s using all M keys. Due to the commutative property of the selected cipher, Player Pm holds the set {EM(· · · (E2(E1(h(x)))) · · ·) : x ∈ Ck,m+1s }.
In Phase 2 (Steps 21-26), the players merge the lists of encrypted itemsets. At the completion of this stage P1 holdsthe union set Cks = _M m=1 Ck,m s hashed and then encrypted by all encryption keys, together with some fake itemsets that were used for the sake of hiding the sizes of the sets Ck,m s ; those fake itemsets are not needed anymore and will be removed after decryption in the next phase.

The merging is done in two stages, where in the first stage the odd and even lists are merged separately. As explained in [18, Section 3.2.1], not all lists are merged at once since if they were, then the player who did the merging (say P1)would be able to identify all of his own encrypted itemsets (as he would get them from PM) and then learn in which ofthe other sites they are also locally frequent.

III. 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 [2], [11]. The idea is that Fig. 1. Computation and communication costs versus the number of transactions N 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 [22] showed how to securely build an ID3 decision tree when the training set is distributed horizontally. Lin et al. [21] discussed secure clustering using the EM algorithm over horizontally distributed data. The problem of distributed association rule mining was studied in [19], [31], [33] in the vertical setting, where each party holds a different set of attributes, and in [18] in the horizontal setting. Also the work of [26] 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 [26] that distinguishes it from [18] and the present study is that no collusions occur between the different network nodes — resources or managers.

The problem of secure multiparty computation of the union of private sets was studied in [7], [14], [20], as well as in [18]. Freedman et al. [14] present a privacy-preserving protocol for set intersections. It may be used to compute also set unions through set complements, since $A \cup B = A \cap B$. Kissner and Song [20] present a method for representing sets as polynomials, and give several privacy-preserving protocols for set operations using these representations. They consider the threshold set union problem, which is closely related to the threshold function (Definition 2.1). The communication overhead of the solutions in those two works, as well as in [18]'s and in our solutions, depends linearly on the size of Fig. 3. Computation and communication costs versus the support threshold *s* the ground set. However, as the protocols in [14], [20] use homomorphic encryption, while that of [18] uses commutative encryption, their computational costs are significantly higher than ours. The work of Brickell and Shmatikov [7] is an exception, as their solution entails a communication overhead that is logarithmic in the size of the ground set. However, they considered only the case of two players, and the logarithmic communication overhead occurs only when the size of the intersection of the two sets is bounded by a constant. The problem of set inclusion can be seen as a simplified version of the *privacy-preserving keyword search*. In that problem, the server holds a set of pairs *{*(*xi*, *pi*)*}ni* =1, where *xi* are distinct "keywords", and the client holds a single value *w*. If *w* is one of the server's keywords, i.e., *w* = *xi* for some $1 \leq i \leq n$, the client should get the corresponding *pi*.

In case *w* differs from all *xi*, the client should get notified of that. The privacy requirements are that the server gets no information about *w* and that the client gets no information about other pairs in the server's database. This problem was solved by Freedman et. al. [13]. If we take all *pi* to be the empty string, then the only information the client gets is whether or not *w* is in the set *{x*1*, . . . , xn}*. Hence, in that case the privacy-preserving keyword search problem reduces to the set inclusion problem. Another solution for the set inclusion problem was recently proposed.

IV. PERFORMANCE EVALUATIONS

We describe the synthetic database that we used for our experimentation. In Section VI we explain how the database was split horizontally into partial databases. The results are given in Section VI

A. Synthetic database generation

The databases that we used in our experimental evaluation are synthetic databases that were generated using the same techniques that were introduced in [1] and then used also in subsequent studies such as [8], [18], [23]. Table 1 gives the parameter values that were used in generating the synthetic database. The reader is referred to [8], [18], [23] for a description of the synthetic generation method and the meaning of each of those parameters. The parameter values that we used here are similar to those used in [8], [18], [23]. Parameter Interpretation Value *N* Number of transactions in the whole database 500,000 *L* Number of items 1000 *At* Transaction average size 10 *Af* Average size of maximal potentially large itemsets 4 *Nf* Number of maximal potentially large itemsets 2000 *CS* Clustering size 5 *PS* Pool size 60 *Cor* Correlation level 0.5 *MF* Multiplying factor 1800. Parameters for generating the synthetic database.

B. Distributing the database

Given a generated synthetic database *D* of *N* transactions and a number of players *M*, we create an artificial split of *D* into *M* partial databases, *Dm*, $1 \leq m \leq M$, in the following manner: For each $1 \leq m \leq M$ we draw a random number *wm* from a normal distribution with mean 1 and variance 0.1, where numbers outside the interval [0.1, 1.9] are ignored. Then, we normalize those numbers so that _*M m*=1 *wm* = 1. Finally, we randomly split *D* into *m* partial databases of expected sizes of *wmN*, $1 \leq m \leq M$, as follows: Each transaction $t \in D$ is assigned at random to one of the partial databases, so that Pr($t \in Dm$) = *wm*, $1 \leq m \leq M$.

C. Experimental setup

We compared the performance of two secure implementations of the FDM algorithm (Section 1.1.2). In the first implementation (denoted FDM-KC), we executed the unification step (Step 4 in FDM) using Protocol UNIFI-KC, where the commutative cipher was 1024-bit RSA [25]; in the second implementation (denoted FDM) we used our Protocol UNIFI, where the keyed-hash function was HMAC [4]. In both implementations, we implemented Step 5 of the FDM algorithm in the secure manner that was described in Section 3. We tested the two implementations with respect to three measures:

1) Total computation time of the complete protocols (FDMKC and FDM) over all players. That measure includes the Apriori computation time, and the time to identify the globally *s*-frequent itemsets, as described in Section 3. (The latter two procedures are implemented in the same way in both Protocols FDM-KC and FDM.)
2) Total computation time of the unification protocols only (UNIFI-KC and UNIFI) over all players.
3) Total message size. We ran three experiment sets, where each set tested the dependence of the above measures on a different parameter: *N* — the number of transactions in the unified database.

V. CONCLUSIONS AND FUTURE WORK

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 holds. 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 described in Section 3; 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.

The second and third stages of the protocol of, as described. 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.

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] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In STOC, pages 503–513, 1990.
[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In Crypto, pages 1–15, 1996.
[5] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In CCS, pages 257–266, 2008.
[6] J.C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In Crypto, pages 251–260, 1986.
[7] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In ASIACRYPT, pages 236–252, 2005.
[8] D.W.L. Cheung, J. Han, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In PDIS, pages 31–42, 1996.
[9] D.W.L Cheung, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. IEEE Trans. Knowl. Data Eng., 8(6):911–922, 1996.
[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms.
IEEE Transactions on Information Theoemail address or URL in your paper, you must type out the address or URL fully in Regular font.

ABOUT THE AUTHORS

Mr.CHITTENI SIVA is currently a student of R.M.KEngineering College, Chennai and he is doing his Masters in "NetworkEngineering". He received his bachelor's degree in computer science and engineering in BHARATHIDASAN ENGINEERING COLLEGE in 2012. His area of interest includes Computer Networks, Network Security, Wireless Network, Internet engineering etc…

S SELVI is an assistant professor in Department of Computer Science and Engineering, RMK Engineering College, Kavaraipettai, Chennai.