Threshold Cryptosystem Mining of Association Rules using Horizontal Distributed Database

T.Satish Vijay Rajeev[#1], Gousiya Begum [*2]

[#1] Assistant Professor CSE MGIT Hyderabad T.S. India

[*2]Assistant Professor CSE MGIT Hyderabad T.S. India

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.*

Keywords — *threshold-c, mergekc, privacy, association rules, FDM*

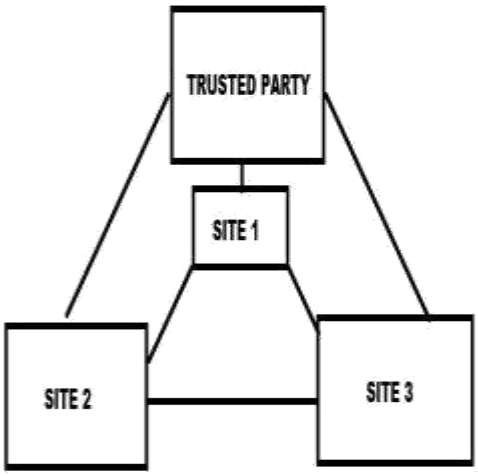
I. INTRODUCTION

Data mining techniques find hidden information from large database while secret data is preserved safely when data is allowed to access by single person. Data mining has been viewed as a threat to privacy because of the widespread proliferation of electronic data maintained by corporations. This has lead to increased concerns about the privacy of the underlying data. . In large applications the whole data may be in single place called centralized or multiple sites called distributed database, i.e., database that share the same schema but hold information on different entities. The goal is to minimizing the information of different databases with in their own transaction using association rules while the global information should be supported by all supporters in every databases. Now a days many people want to access data or hidden information using data mining technique even they are not fully authorized to access. For getting mutual benefits, many organizations wish to share their data to many

Processing Our proposed protocol based on two novel secure multiparty algorithm using these algorithms the protocol provides enhanced privacy, security and efficiency as it uses commutative encryption. privacy liability can prevent building a centralized warehouse data may be distributed among several custodians none of which are allowed to transfer their data to another site. Data mining can extract important knowledge from large data collections but sometimes these collections are split among various parties. Privacy liability may prevent the parties from directly sharing the data, and some types of information about the data.

F → public functions

F(y1,y2,y3…) → private inputs



In case of its abscence, it is needed to devise a protocol that the players can run on their own in order to arrive at the required output x. 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 previous year various techniques are applied for secure mining of association rules in horizontally partitioned database. These approaches use various techniques such as data perturbation, homo-morphic encryption, keyword search and oblivious pseudorandom functions etc.. Yao [32] 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 in[3],[5],[15]. 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.

These privacy preserving approaches are inefficient due to

- Homo-morphic encryption
- Higher computational cost

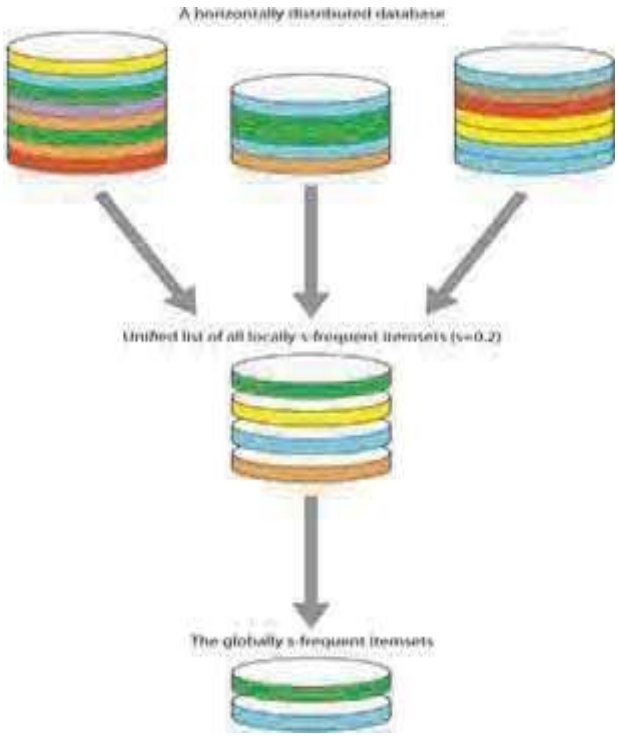
In some of the techniques data owner tries to hide data from data miner. Our proposed protocol based on two novel secure multiparty algorithm using these algorithms the protocol provides enhanced privacy, security and efficiency as it uses commutative encryption. In this project we propose a protocol for secure mining of association rules in horizontally distributed database. This protocol is based on: FDM Algorithm which is an unsecured distributed version of the Apriori algorithm. In our protocol two secure multiparty algorithms are involved:

1. Computes the union of private subsets that each interacting players hold.
2. Tests the inclusion of an element held by one player in subset held by another. In Horizontally partitioned database there are several players that hld homogeneous database. Our protocol offers enhanced privacy with respect to the currentleading K and C protocol simplicity, more efficient in terms of communication rounds, communication cost and computational cost. 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.

now we propose an another alternative protocol for the very high secured computation of the combination of private subsets. The proposed protocol improves upon that in [18] in terms of simplicity and efficiency as well as privacy. 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, unlike the protocol of [18] 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 [18].

The protocol that we propose here computes a threshold functions which is use of parametric family of 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. Our



2. LITERATURE SURVEY

1. Keying hash functions for message authentication

AUTHORS: M. Bellare, R. Canetti, and H. Krawczyk

The use of cryptographic hash functions like MD5 or SHA-1 for message authentication has become a standard approach in many applications, particularly Internet security protocols. Though very easy to implement, these mechanisms are usually based on ad hoc techniques that lack a sound security analysis. We present new, simple, and practical constructions of message authentication schemes based on a

cryptographic hash function. Our schemes, NMAC and HMAC, are proven to be secure as long as the underlying hash function has some reasonable cryptographic strengths. Moreover we show, in a quantitative way, that the schemes retain almost all the security of the underlying hash function. The performance of our schemes is essentially that of the underlying hash function. Moreover they use the hash function (or its compression function) as a black box, so that widely available library code or hardware can be used to implement them in a simple way, and replaceability of the underlying hash function is easily supported.

2. FairplayMP - A system for secure multi-party computation

AUTHORS: A. Ben-David, N. Nisan, and B. Pinkas

We present FairplayMP (for "Fairplay Multi-Party"), a system for secure multi-party computation. Secure computation is one of the great achievements of modern cryptography, enabling a set of untrusting parties to compute any function of their private inputs while revealing nothing but the result of the function. In a sense, FairplayMP lets the parties run a joint computation that emulates a trusted party which receives the inputs from the parties, computes the function, and privately informs the parties of their outputs. FairplayMP operates by receiving a high-level language description of a function and a configuration file describing the participating parties. The system compiles the function into a description as a Boolean circuit, and perform a disributed

evaluation of the circuit while revealing nothing else. FairplayMP supplements the Fairplay system [16], which supported secure computation between two parties. The underlying protocol of FairplayMP is the Beaver-Micali-Rogaway (BMR) protocol which runs in a constant number of communication rounds (eight rounds in our implementation). We modified the BMR protocol in a novel way and considerably improved its performance by using the Ben-Or-Goldwasser-Wigderson (BGW) protocol for the purpose of constructing gate tables. We chose to use this protocol since we believe that the number of communication rounds is a major factor on the overall performance of the protocol. We conducted different experiments which measure the effect of different parameters on the performance of the system and demonstrate its scalability. (We can now tell, for example, that running a second-price auction between four bidders, using five computation players, takes about 8 seconds.)

3. Secret sharing homomorphisms keeping shares of a secret secret.

AUTHORS: J.C. Benaloh

lackley and Shamir independently proposed schemes by which a secret can be divided into many shares which can be distributed to mutually suspicious agents. This paper describes a homomorphism property attained by these and several other secret sharing schemes which allows multiple secrets to be combined by direct computation on shares. This property reduces the need for trust among agents and allows secret sharing to be applied to many new problems. One application described gives a method of verifiable secret sharing which is much simpler and more efficient than previous schemes. A second application is described which gives a fault-tolerant method of holding verifiable secret-ballot elections.

4. Privacy-preserving graph algorithms in the semi-honest model

AUTHORS: J. Brickell and V. Shmatikov

We consider scenarios in which two parties, each in possession of a graph, wish to compute some algorithm on their joint graph in a privacy-preserving manner, that is, without leaking any information about their inputs except that revealed by the algorithm's output.

Working in the standard secure multi-party computation paradigm, we present new algorithms

Approach & Implementation:

Annotations and Denotations:

In horizontal distributed database their will be row and coloums.

D → Transactional database

R → rows (used for transactions)

L→ coloumns (used for itemsets)

D = D1 ∪ ・ ・ ・ ∪ DM,

Rm= |Dm| (1≤m≤M)

Supp(Y) → global support (number of transations in D)

Fk,m

Supp(Y)= ∑Mm=1 Suppm(Y)

Supp(Y) ≥ sR. It is called locally s-frequent at Dm if supp(Y) ≥ sRm For each 1 ≤ k ≤ L,

Fks→ set of all k-itemsets (namely, itemsets of size k) that are s-frequent,

s → set of all k-itemsets that are locally s-

frequent at Dm,1 ≤ m ≤ M.

Our main computational goal is to find, for a given threshold support 0 < s ≤ 1, the set of all s-frequent itemsets,

L= Fs

k=1 Fks . We may then continue to find all (s, c)-association rules, i.e., all association rules of support at least sR 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 ⇒ Y is supp(X ∪ Y) and its confidence is supp(X ∪ Y)/supp(X).)

Modules:

1. Privacy Preserving Data Mining
2. Distributed Computation
3. Frequent Itemsets
4. Association Rules

MODULES DESCRIPTION:

1. Privacy Preserving Data Mining:

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 anonym zing the data prior to its release. The main approach in this context is to apply data perturbation. The idea is that. 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

Whether 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.

2. Distributed Computation:

We compared the performance of two secure implementations of the FDM algorithm Section In the first implementation (denoted FDM-KC), we executed the unification step using Protocol UNIFI-KC, where the commutative cipher was 1024-bit RSA in the second implementation (denoted FDM) we used our Protocol UNIFI, where the keyed-hash function was HMAC. In both implementations, we implemented Step 5 of the FDM algorithm in the secure manner that was described in later. 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 item sets, as described in later.

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.

3. Frequent Itemsets:

We describe here the solution that was proposed by Kantarcioglu and Clifton. They onsidered two possible settings. If the required output includes all globally s-frequent item sets, as well as the sizes of their supports, then the values of Δ(x) can be revealed for all. In such a case, those values may be computed using a secure summation protocol, where the private addend of Pm is suppm(x) − sNm. The more interesting setting, however, is the one where the support sizes are not part of the required output. We proceed to discuss it.

4. Association Rules:

Once the set Fs of all s-frequent itemsets is found, we may proceed to look for all (s, c)-association rules (rules with support at least sN and confidence at least c). In order to derive from Fs all (s, c)-association rules in an efficient manner we rely upon the straightforward lemma.

The Fast Distributed Mining algorithm:

Fast Distributed Mining (FDM) algorithm 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 -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.

Fsk,m = Fsk

should be satisfy for atleast one of the sites.

STEP1: Goal is to calculate FSK-1

STEP2: Each player Pm contains set of all K-1 itemsets whish are both local and global.

Pm→ Fsk-1,m ∩ Fsk-1

By making of all intersections we then applies Apriori algorithm in order to generate the set BSK,M where BSK,M → candidate itemsets – k.

STEP3: each Y ∈ Bk,ms Pm computes suppm(X). He then retains only those itemsets that are locally s-frequent. We denote this collection of itemsets by Ck,m.

STEP4: each pm broadcasts his Csk,m and

STEP5: All players compute the local supports of all itemsets in CKS

Disadvantages of FDM algorithm:

The FDM algorithm violates privacy in two stages:

1. In Step 4, where the players broadcast the itemsets that are locally frequent in their private databases
2. in Step 6, where they broadcast the sizes of the local supports of candidate itemsets.

Kantarcioglu and Clifton [18] proposed secure implementations of those two steps. Our improvement is with regard to the secure implementation of Step 4, which is the morecostly
trends stage of the protocol, and the one in which the protocol of [18] leaks excess information.

Providing Security to all Locally Frequent Itemset Using Kantarcioglu and Clifton protocol : As the security will be provided by using an apriori algorithm to all local itemsets.
$Ap(F^{K-1}{}_S)$ → set of all candidate K-itemsets that apriori algorithm generates from $F^{K-1}{}_S$.
Fs → FULLSET contains all global s-frequent sets

$$F_s = U_{K=1}{}^{L} F^{K}{}_S$$

This figure and expression shows that everything was included in the apriori function with all local and global itemsets.

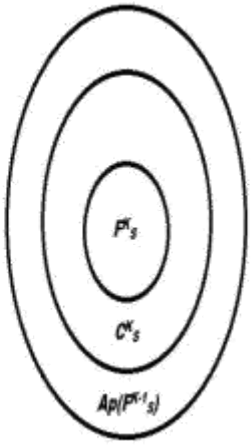
Merging lists of local frequent itemsets (MERGE-KC) :

Protocol MERGE-KC securely computes of the union of private subsets of some publicly known ground set ($Ap(F^{k-1}{}_s)$). Such a problem is equivalent to the problem of computing the OR of private vectors. Indeed, if the ground set is $\Omega = \{\omega 1, \ldots, \omega n\}$,
then any subset B of Ω may be described by the characteristic binary vector
$\mathbf{b} = (b1, \ldots, bn) \in Zn2$ where $bi = 1$ if and only if $\omega i \in B$.

Let $\mathbf{b}m$ be the binary vector that characterizes the private subset held by player Pm,
$1 \le m \le M$. Then the merge of the private subsets is described by the OR of those
private vectors,

$$b := V^{M}{}_{m=1}$$

Such a simple function can be evaluated securely by the generic solutions suggested in [3], [5], [15]. We present here a protocol for computing that function which is much simpler to understand and program and much more efficient than those generic solutions. It is also much simpler than Protocol MERGE-KC and employs less cryptographic primitives. Our protocol computes a wider range of functions, which wecall threshold functions.



PROTOCOL: MERGE-KC:

Input: Each player Pm has an input set Ck, ms⊆Ap(Fk−1s), 1 ≤ m ≤ M. Output: Cks =UM m=1 Ck,ms .

1: STAGE 0: starting
2: The players decide on a commutative cipher and each player Pm, 1 ≤ m ≤ M, selects a random secret encryption key Km.
3: The players select a hash function h and compute h(y) for all y ∈ Ap(Fk−1s).
4: Build a lookup table T = {(y, h(y)) : y ∈ Ap(Fk−1s)}.
5: STAGE 1: Encrypting all itemsets by fake identity
6: for all Player Pm, 1 ≤ m ≤ M, do
7: Set Ym = ∅.
8: for all y ∈ Ck,ms do
9: Player Pm computes EKm(h(y)) and adds it to Ym.
10: end for
11: Player Pm adds to Ym faked itemsets until its size becomes |Ap(Fk−1s)|.
12: end for
13: for i = 2 to M do
14: for all 1 ≤ m ≤ M do
15: Pm sends a permutation of Ym to Pm+1.
16: Pm receives from Pm−1 the permuted Ym−1.
17: Pm computes a new Ym as the encryption of the permuted Ym−1 using the key Km.
18: end for
19: end for
20: STAGE 2: Merging of all itemsets
21: Each odd player sends his encrypted set to player P1.
22: Each even player sends his encrypted set to player P2.
23: P1 unifies all sets that were sent by the odd players and removes duplicates.
24: P2 unifies all sets that were sent by the even players and removes duplicates.
25: P2 sends his permuted list of itemsets to P1.

26: P1 unifies his list of itemsets and the list received from P2 and then removes duplicates from the unified list. Denote the final list by ECks .
27: STAGE 3: Decrypting buy an key
28: for m = 1 to M − 1 do
29: Pm decrypts all itemsets in ECks using Km.
30: Pm sends the permuted (and Km-decrypted) ECks to Pm+1.
31: end for
32: PM decrypts all itemsets in ECks using KM; denote the resulting set by Cks .
33: PM uses the lookup table T to replace hashed values with the actual itemsets, and to identify and remove faked itemsets.
34: PM broadcasts Cks .
Explanation of MERGE-KC:

STAGE 1: The hashed sets Ck,ms ,1 ≤ m ≤ M, all players will perform encryption. each player Pm hashes all itemsets in Ck,ms and then encrypts them using the key Km.. In order to hide the number of locally frequent itemsets that he has, he adds faked itemsets to the resulting set until its size becomes |Ap(Fk−1s)|. the players start a loop of M − 1 cycles, where in each cycle they perform the following operation:

Player Pm sends a permutation of Ym to the next player Pm+1;Challenge. Player Pm receives from Pm−1 a permutation of the set Xm−1 and then computes a new Ym as Y m = EKm(Ym−1).

STAGE 2: the players merge the lists of encrypted itemsets. P1 holds the union set Cks = UMm=1 C k,ms hashed At the completion of this stage and then encrypted by

all encryption keys, together with some fake item sets that were used for the sake of hiding the sizes of the sets Ck,ms ; those fake itemsets are not needed anymore and will be removed after decryption in the next phase.

STAGE 3: a similar round of decryptions is initiated. At the end, the last player who performs the last decryption uses the lookup table T that was constructed in order to identify and remove the fake itemsets and then to recover Cks . Finally, he broadcasts Cks to all his peers.

p1 → private subset + Ck,ms (to hide size) p2 → private subset + Ck,ms(to hide size)
:
:
:
pm → private subset + Ck,ms
Pm + ck,ms

ENCRTPTION(EACH PLAYER)
→ADD PRIVATE KEY TO EACH PLAYER
MERGING ALL THE ENCRYPTED SUBSETS

Secure Multi-party Computation:

Substantial work has been done on secure multi-party computation. The key result is that a wide class of computations can be computed securely under reasonable assumptions. We give a brief overview of this work, concentrating on material that is used later in the paper. The definitions given here are from Goldreich. For simplicity, we concentrate on the two-party case. Extending the definitions to the multi-party case is straightforward.

Security in semi-honest model: A semi-honest party follows the rules of the protocol using its correct input, but is free to later use what it sees during execution of the protocol to compromise security. This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. Also, a protocol that is buried in large, complex software can not be easily altered. A formal definition of private two-party computation in the semi-honest model is given below. Computing a function privately is equivalent to computing it securely. The formal proof of this can be found in Gold Reich.

THRESHOLD-C:
Threshold + SETINC = Threshold-c

1. Threshold:

We show that given keys for any sufficiently efficient system of this type, general multiparty computation protocol protocols for n parties can be devised which are secure against an active adversary that corrupts any minority of the parties. The total number of bits broadcast is O(nk|C|), where k is the security parameter and |C| is the size of a (Boolean) circuit computing the function to be securely evaluated. An earlier proposal by Franklin and Haber with the same complexity was only secure for passive adversaries, while all earlier protocols with active security had complexity at least quadratic in n.

protocol for computing that function which is much simpler to understand and program and much more efficient than those generic solutions. It is also much simpler than Protocol MERGE-KC and employs less cryptographic primitives. Our protocol (STAGE 2) computes a wider range of functions, which we call threshold functions

Relation between Threshold Cryptosystem and Secure Multiparty

1. A threshold cryptosystem means the decryption key can be split into n shares such that only t≤n are required to recover it.
2. That property in isolation is not useful for multiparty computation.
3. However when you combine a threshold cryptosystem with one that is at least partially homomorphic (meaning you can do some operation,

like addition or multiplication, under encryption), then the two properties combined can make a useful basis for multiparty computation.

4. The model is that each party encrypts their inputs, you use the homomorphic property to do some computation on the cipher texts while they are still encrypted, and the

threshold property enforces that only the final value is decrypted assuming less than t dishonest key shareholders.

5. Without the threshold (or at least a distributed n-out-of-n key), the key holder could simply decrypt the inputs and learn everyone's value. Protocol SETINC involves three players: P1 has a vector s = (s(1), . . . , s(n)) of elements in some ground set Ω; PM, on the other hand, has a vector Θ = (Θ(1), . . . , Θ(n)) of subsets of that ground set. The required output is a vector b = (b(1), . . . , b(n)) that describes the corresponding set inclusions in the following manner: b(i) = 0 if s(i) ∈ Θ(i).

Results and discussion:

DISCUSSION:

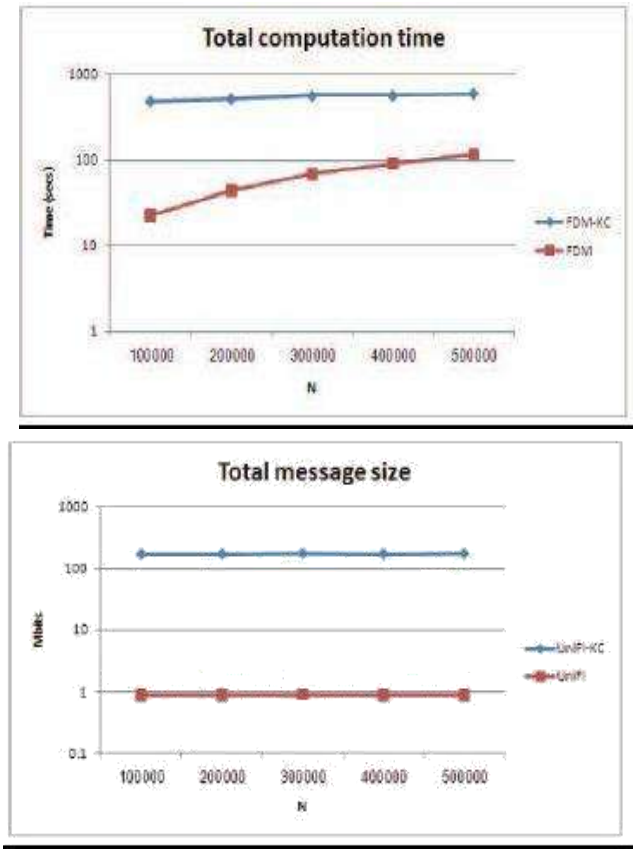


Figure 1: Computation and communication costs versus the number of transactions N

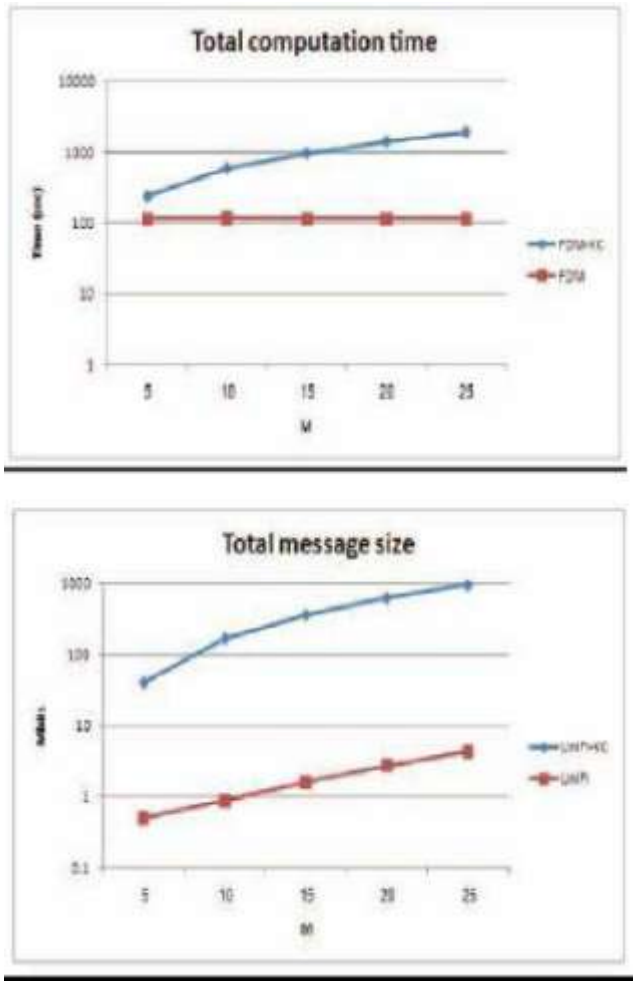


Figure 2: Computation and communication costs versus the number of players M

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] 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 Theory*, 31:469–472, 1985.

[11] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002. [12] R. Fagin, M. Naor, and P. Winkler. Comparing Information Without Leaking It. *Communications of the ACM*, 39:77–85, 1996.

[13] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.

[14] M.J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.

[15] 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.

[16] H. Grosskreutz, B. Lemmen, and S. R¨uping. Secure distributed subgroup discovery in horizontally partitioned data. *Transactions on Data Privacy*, 4:147– 165, 2011.

[17] W. Jiang and C. Clifton. A secure distributed framework for achieving *k*-anonymity. *The VLDB Journal*, 15:316–333, 2006.

[18] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontallypartitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16:1026–1037, 2004.

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

[20] L. Kissner and D.X. Song. Privacy-preserving set operations. In *CRYPTO*, pages 241–257, 2005.

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

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

[23] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. In *SIGMOD Conference*, pages 175–186, 1995.

[24] S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logarithms over *gf*(*p*) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[25] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

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

[27] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.

[28] T. Tassa and D. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[29] T. Tassa and E. Gudes. Secure distributed computation of Anonymized views of shared databases.
Transactions on Database Systems, 37, Article 11, 2012.

[30] T. Tassa, A. Jarrous, and J. Ben-Ya'akov. Oblivious evaluation of multivariate polynomials. *Submitted*.

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

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

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

[34] S. Zhong, Z. Yang, and R.N. Wright. Privacy-enhancing *k*anonymization of customer data. In *PODS*, pages 139–147, 2005.