ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (72-77), Month: July - September 2015, Available at: www.researchpublish.com

Mining of Association Rules in Distributed Database

Nayana Marodkar[1], Manoj Chaudhari[2]

[1]M. Tech Student of Computer Science and Engineering, RTMN University, PBCOE, Nagpur, Maharashtra, India
[2]Professor, H.O.D of Computer Science and Engineering, RTMN University, PBCOE, Nagpur, Maharashtra, India

Abstract: **Data mining is the most fast growing area today which is used to extract important knowledge from large data collections but often these collections are divided among several parties. Association rule mining is one of the techniques in data mining. Here , we propose a protocol for mining of association rules in horizontally distributed databases and protocol is based on the Fast Distributed Mining (FDM) algorithm which is an unsecured distributed version of the Apriori algorithm. The main ingredients in 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 addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.**

Keywords: **Apriori Algorithm, Association Rule, Distributed Database; FDM, secure multi-party algorithms.**

I. INTRODUCTION

Data mining can extract important knowledge from large data collections but sometimes these collections are split among various parties. Data mining is defined as the method for extracting hidden predictive information from large distributed databases. It is new technology which has emerged as a means of identifying patterns and trends from large quantities of data. The final product of this process being the knowledge, meaning the significant information provided by the unknown elements.Here we study the problem of mining of association rules in horizontally partitioned databases. There are several sites that hold homogeneous databases, i.e., databases that share the same schema but hold information on different entities [1]. With given minimal support and confidence levels that hold in the unified database the goal is to find all association rules, while minimizing the information disclosed about the private databases held by those players. That goal defines a problem of secure multi-party computation. The information that would like to protect in this proposed work, not only individuals transaction but also more global information such as association rules that are supported locally in each of these database .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. It is needed to devise a protocol that in the absence of such a trusted third party the players can run on their own in order to arrive at the required output y [1]. Then such a devised protocol is considered 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 proposed system is, the inputs are the partial databases, and the required output is the list of association rules with given support and confidence. 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, 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 where more suitable security definitions that allow parties to choose their desired level of security are needed, to allow effective solutions that maintain the desired security and devised a protocol for its solution[2]. 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 crypto-graphic 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 and it is argued that such information leakage is innocuous, whence acceptable from practical point of view.

In this we propose an alternative protocol for the secure computation of the union of private subsets. The proposed protocol improves upon that in terms of simplicity and efficiency as well as privacy. In particular, protocol does not depend on cryptographic primitive i.e. commutative encryption and oblivious transfer. While the solution is still not perfectly secure, it leaks excess information only to a small number of coalitions (three), unlike the protocol of that discloses information also to some single players. In addition, also claim that the excess information that our protocol may leak is less sensitive than the excess information leaked by the protocol. 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.

II. METHODOLOGY

a. **Process Design:** Let consider D be a transaction database. The database is partitioned horizontally between P1, P2 . . . , Pm players, denoted 1 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 U···U DM. 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, sup (X), is the number of transactions in Dm that contain it.

b. **Support :** The rule X ⇒ Y holds with support s if s% of transactions in D contain X ∪ Y. Rules that have a s greater than a user-specified support is said to have minimum support or threshold support. The support of rule is defined as, **sup(X) = no of transactions that contain X / total no of Transactions.**

c. **Confidence:** The rule X ⇒ Y holds with confidence c if c% of the transactions in D that contain X also contain Y. Rules that have a c greater than a user-specified confidence is said to have minimum confidence or threshold Confidence. The confidence of a rule is defined as, **conf(X =>Y) = sup(X U Y)/ supp(X).**

d. FDM 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[20]. In the first iteration of FDM algorithm, when k=1, Cs1,m the set that the mth player computes (Steps 2-3) is just Fs1,m , 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 that case it terminates.

FDM algorithm steps are as follows:

1) Initialization

2) Candidate Sets Generation

3) Local Pruning

4) Unifying the candidate itemsets

5) Computing local supports

6) Broadcast Mining Results

Unifi-KC(FDM-KC):

The input that each player Pm has at the beginning of Protocol UNIFI-KC (Unifying lists of locally Frequent Itemsets — Kantarcioglu and Clifton) [14]. is the collection Csk,m , as defined in Steps 2-3 of the FDM algorithm. Let Ap(Fk−1) denotes the set of all candidate k-itemsets that the Apriori algorithm generates from Fs k−1 s . The output of the protocol is the union .In the first iteration of this computation, and the players compute all -frequent 1- itemsets (here FS 0 = s {∅}). In the next iteration they compute all s-frequent 2-itemsets, and so forth, until the first ≤ in which they find no s-frequent k-itemsets. After computing that union, the players proceed to extract from Csk the subset Fs k that consists of all k-itemsets that are globally s-frequent; Finally, by applying the above described procedure from k=1 until the first value of k≤L for which the resulting set Fsk is empty, the Players may recover the full set of all globally –frequent item sets. Protocol UNIFI-KC works as follows: First, each player adds to his private subset Csk,m fake item sets, in order to hide its size. Then, the players jointly compute the encryption of their private subsets by applying on those subsets a commutative encryption, where each player adds, in his turn, his own layer of encryption using his private secret key. At the end of that stage, every item set in each subset is encrypted by all of the players; the usage of a commutative encryption scheme ensures that all item sets 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 item sets which are identified as fake. Steps For secure computations of all item sets (by K&C):

1. Cryptographic Primitive Selection

- Player selects needed commutative cipher and corresponding private key

- Player selects hash function to apply on all itemsets prior to encryption

- Player compute lookup table with hash values to find preimage of given hash values.

2. All itemsets Encryption

3. Itemset Merging

- Each odd player sends his encrypted set to player 1.

- Each even player sends his encrypted set to player 2.

- Player 1 unifies all sets that were sent by the odd players and removes duplicates.

- Player 2 unifies all sets that were sent by the even players and removes duplicates.

- Player 2 sends his permuted list of itemsets to Player 1.

- Player 1 unifies his list of itemsets and the list received from Player 2 and then remove duplicates from the unified list. Denote the final list by ECSK.

4. Decryption

III. CRYPTANALYSIS AND VALIDATION

SHA algorithm:

A hashing algorithm is a cryptographic algorithm that can be used to provide data integrity and authentication. They are also typically used in password based systems to avoid the need to store plaintext passwords. A hashing algorithm is a deterministic function that takes in an arbitrary length block of data, and returns a fixed-size string, which is called the hash value. A secure hashing algorithm has important properties:

Each hash is unique but always repeatable:

The word 'cat' will hash to something that *no* other word hashes too, but it will *always* hash to the same thing.

The function is 'one way'

If you are given the value of what 'cat' hashes too but you didn't know what made it, you would never be able to find out that 'cat' is original world . There are many different hash functions but the one I will be concentrating on today is called the Secure Hash Algorithm 1 or SHA-1.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (72-77), Month: July - September 2015, Available at: www.researchpublish.com

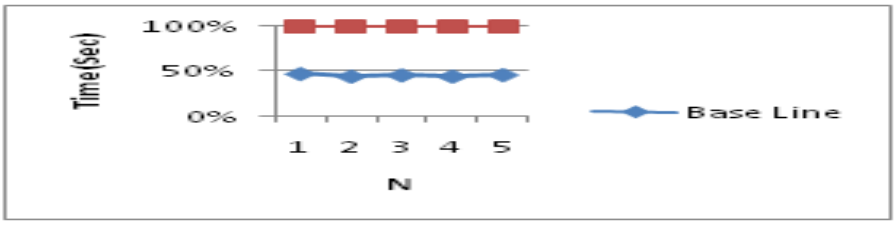
For a hash function for which L is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in approximately 2^L evaluations. This is called a preimage attack and may or may not be practical depending on L and the particular computing environment. The second criterion, finding two different messages that produce the same message digest, namely a *collision*, requires on average only about $1.2 * 2^{L/2}$ evaluations using a birthday attack. For the latter reason the strength of a hash function is usually compared to a symmetric cipher of half the message digest length. Thus SHA-1 was originally thought to have 80-bit strength. Cryptographers have produced collision pairs for SHA-0 and have found algorithms that should produce SHA-1 collisions in far fewer than the originally expected 2^{80} evaluations.

In terms of practical security, a major concern about these new attacks is that they might pave the way to more efficient ones. Whether this is the case is yet to be seen, but a migration to stronger hashes is believed to be prudent. Some of the applications that use cryptographic hashes, like password storage, are only minimally affected by a collision attack. Constructing a password that works for a given account requires a preimage attack, as well as access to the hash of the original password, which may or may not be trivial. Reversing password encryption .e.g. to obtain a password to try against a user's account elsewhere) is not made possible by the attacks. However, even a secure password hash can't prevent brute-force attacks on weak passwords. Due to the block and iterative structure of the algorithms and the absence of additional final steps, all SHA functions are vulnerable to length-extension and partial-message collision attacks. These attacks allow an attacker to forge a message signed only by a keyed hash – $SHA(message||key)$ or $SHA(key||message)$ -by extending the message and recalculating the hash without knowing the key. The simplest improvement to prevent these attacks is to hash twice:

$SHA_d(message) = SHA(SHA(0^b||message))$ (the length of 0^b, zero block, is equal to the block size of hash function)

IV. EXPERIMENTAL RESULT

We built a prototype application that demonstrates the proof of concept. The application has been built in PHP which is the IDE for JAVA environment. The experiments are made in a PC with 1GB RAM, Dual core processing running Windows 7 operating system. The experiments are made in terms of total time taken for mining task.



Inputs: Dataset, support , confidence
Outputs: Association Rules

1 Initialization of security keys
2 Candidate set generation from locally available
3 Local pruning based on support values set locally
4 Unifying the candidate results
5 Computing local supports
6 Merging mining results
7 Returning association rules

Figure 1 – Computational time comparison

As seen in Figure 1, it is evident that the algorithm has been tested with number of rows and columns represented by N which is plotted in horizontal axis while the time taken in seconds is represented in vertical axis. The proposed Secure ARM algorithm is compared with a baseline algorithm. The performance of the proposed algorithm is far better than the baseline algorithm.

User can add entries by specifying number of records number of items and maximum quantity (per item) and tested data mining operation module by specifying the number of rules and minimum confidence. Operation is performed in 5 steps

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (72-77), Month: July - September 2015, Available at: www.researchpublish.com

Data fetched for user...

Step 1 Pre-processing data...

Step 2 converted Agent 1s data to mining format...

Step 3 Processing data...

Step 3 completed mining...

Step 4 will perform distributed data mining...

Step 5 updated results in database...

and the result of above 5 steps shows in fig3: association rule with min suport and min confidence

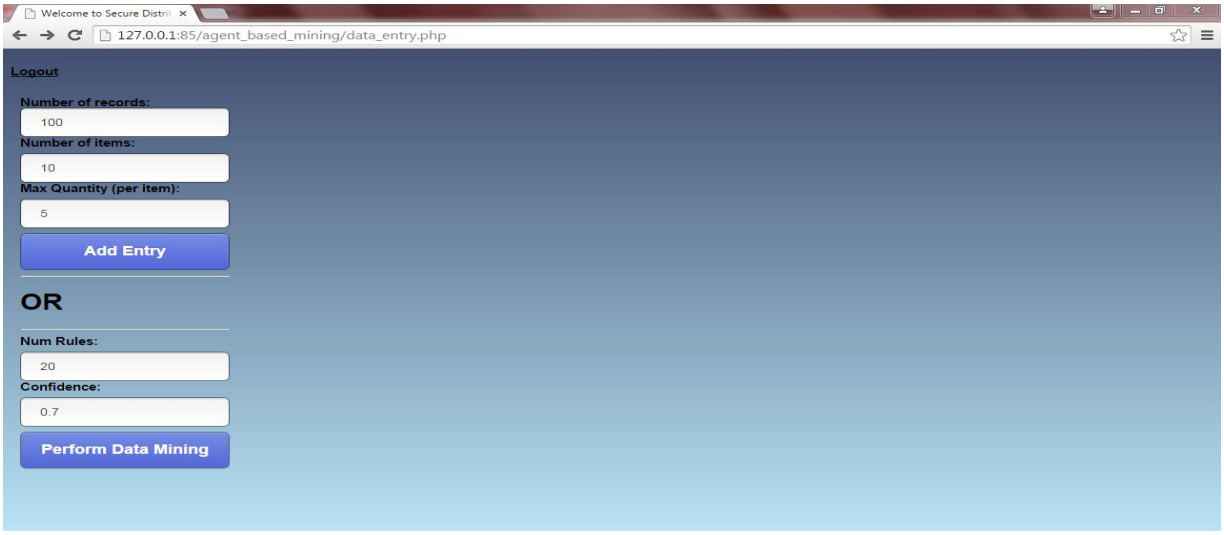


Figure2: adding entries by user or performing frequent item sets

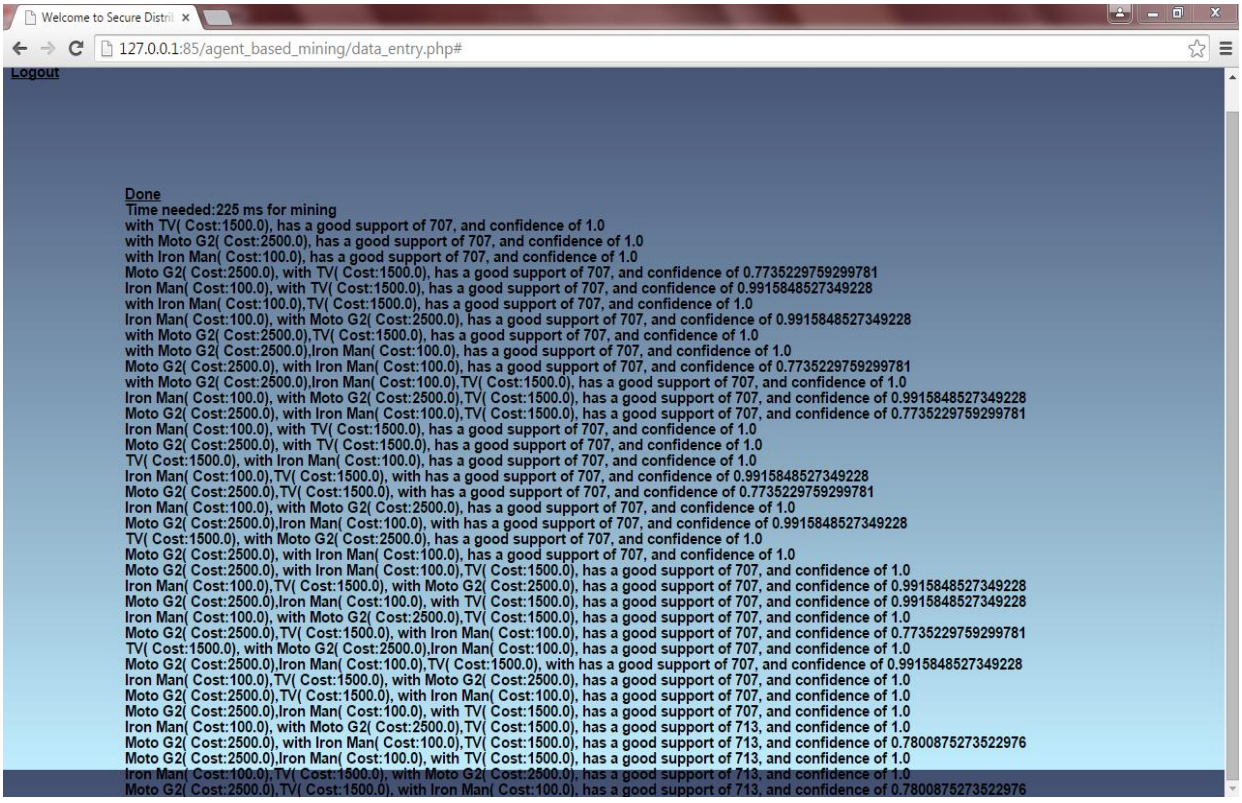


Figure 3:association rule with min suport and min confidence

V. CONCLUSION

Mining association rules is one of the data mining techniques which are very useful for making well informed decisions. In this paper we study secure mining of association rules. Our work is carried out on horizontally distributed database in secure environment. Support and confidence are the statistical measures used for mining association rules. Thus the statistical measures can be used to know how the rules are useful. The more in support and confidence, the more in usefulness of the rules. We used Apriori algorithm along with our algorithm in order to achieve this. Frequent item sets are generated through Apriori and rest of the mechanisms are carried out by the proposed algorithm. We built a prototype application that demonstrates the proof of concept. Later on this architecture is further extended to provide built in auditing mechanism for data consistency. The empirical results are encouraging. In future we improve the prototype and make it a useful tool for mining business intelligence using other data mining algorithms that can be employed to various domains. Thus the tool become useful for acquiring business intelligence for making well informed decisions.

REFERENCES

[1] Tamir Tassa, "Secure Mining of Association Rules in Horizontally Distributed Databases" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 4, APRIL 2014

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," Proc 20th Int'l Conference. Very Large Data Bases (VLDB), pp. 487-499, 1994.

[3] A. Ben-David, N. Nisan and B. Pinkas, "FairplayMP - A System for Secure Multi-Party Computation," Proc. 15th ACM Conference. Computer and Comm. Security (CCS), pp. 257-266, 2008.

[4] M. Kantarcioglu and C. Clifton, "Privacy - Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data ,"IEEE Trans. Knowledge and Data Eng., vol. 16, no. 9, pp. 1026-1037,September 2004

[5] J. Vaidya and C. Clifton, "Privacy Preserving Association Rule Mining in Vertically Partitioned Data," Proc. Eighth ACM SIGKDD Int'l Conference Knowledge Discovery and Data Mining (KDD), pp. 639-644, 2002.

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

[7] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993

[8] M. Houtsma and A. Swami. Set-Oriented Mining of Association Rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.

[9] D.Kerana Hanirex, Dr.K.P.Kaliyamurthie," Mining Frequent Item sets Using Genetic Algorithm", Middle-East Journal of Scientific Research, 19 (6): 807-810, 2014.

[10] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In ASIACRYPT, pages 236–252, 2005.

[11] D.Kerana Hanirex., Dr.K.P.Kaliyamurthie, "Multi-classification Approach for Detecting Thyroid Attacks", IJPBS, 4(3), (B) 1246 – 1251, July (2013).