Mining Numerical Data using Privacy Enhanced Multiparty Association Rules

N.Phanitha[1], M.Lalitha[2]

[1] Student, Computer science and Engineering, G.Narayanama Institute of Technology and Sciences, Telangana, India
[2] Assistant Professor, Computer science and Engineering, G.Narayanama Institute of Technology and Sciences, Telangana, India

[1] phanithan96@gmail.com [2]lalitha.malla@gmail.com

Abstract

Association rules are used for analysis alluring relationships hidden in datasets. If multiple parties need to accomplish a reckoning supported their non-public inputs, however party is reluctant to disclose its own particluar yield to the other else. Such issue is SMC (Secure multiparty computation) problem. The main ingredients in the protocol are two new secure multi-party algorithms one who reckon the particular abutment connected with non-public subsets that every one of the interacting parties cling on, and another that test the admitance of an element placed by one particular party in a very subset placed by another. Our project, like theirs, is in line with the Fast Dsitributed Mining (FDM) protocol, that is absolutely an unsecured sent out version from the Apriori algorithm.The implementation of the partitioned protocol is identification of these candidate item sets that are globally frequent so the deduction of all association rules from the mined data and also the issue of verifying the set admittance will be apparent as a simplified version of the privacy conversing keyword search. We tend to analyze the performance of secure implementations of FDM algorithm and UNIFI (Unifying List of Locally Frequent Item sets).

1. Introduction

With expansive data, multiple parties hold homogenous data so our objective is to discover revealed connections that are represented in the form of association rules or sets of frequent items.

The work inside security ensuring data mining has regarded two comparative setting. one, in which the information owner along with the data miner are two different entities, as well as another, when the data will be conveyed between several parties who attempt and accordingly conduct data mining around the unified corpus involving data that they hold. From the initial setting, the objective is to shield the data from the miner. The reason, data owner is aimed at anonymzing the data prior to help its unleash. The key methodology with this ambience is to apply data perturbation. The tactic is the perturbed data may be utilized to induce standard patterns inside the data, with out divulge unique data. In the second setting, the objective is to accomplish data mining although ensuring data of each data owner. That is a issue with respect to secure multi-party computation.

This paper recommend an alternative protocol for secure reckoning with the abutment related to private subsets. The offered protocol boosts upon that in relation to simplicity as well as efficiency along with privacy. As an example, our protocol doesn't determined by commutative encryption and oblivious exchange what simplifies this to contributes toward much lessened communication as well as computational costs. While this perfect solution is not flawlessly secure, it leaks extra data merely to a little bit of possible alliances, different from the protocol that discloses information and also to some individual parties.

Also, the undesirable data that our protocol might also perhaps leak may be less sensitive than the excess information leaked through the protocol. The protocol that we propose here computes the parameterized chic of functions, that we call as threshold protocol, when the two intense cases correspond to the complications of calculating the abutment and intersection involving non-public subsets. The above mentioned two instances that general-purpose methods that may be acclimated in various different contexts at the same time. Any hassle of secure multiparty computation that are an integral part of our discussion will be the set admittance problem; that's, the issue where Alice keeps a non-public subset involving some terrain set, and Bob has a factor in the terrain set, and they hope ascertain regardless of whether Bob's part is within Alice's set, without revealing to possibly both ones information around the other party's suggestions beyond the above mentioned admittance.

The algorithm for association rule is usually decomposed into 2 major steps:

1. Find out all large item sets that have support value exceed a minimum support threshold.
2. Find out all the association rules that have value exceed a minimum confidence threshold.

Generally, association rules are taken into consideration if they satisfy both a minimum threshold of support and confidence. The thresholds are set by users or domain specialists. Rules that satisfy each threshold (min sup and min conf) are referred to as strong association rules.

standardise lift: the maximum value that lift can take is n and this happens once a rule is supported by just one transaction. The rules with low support threshold in all the transactions will not typically be of interest and may even be cropped in the rule generation stage of the mining process

Single-level association rules: Associations between items or attributes at the same level of abstraction, i.e., at the same level of hierarchy

Multilevel association rules: Associations between items or attributes at multiple levels of abstraction, i.e, at multiple levels of concept hierarchy. Here, calculate frequent item sets at each concept level, until no more frequent item sets can be found.

2. Background And Related Work

Recent work, describes a awful optimized accomplishing of a two-party protocol that offer security against malicious adversaries of Lindell and Pinkas, but protocols concentrating on the using the same efficiency that never available for the multi-party case. Even that accomplishment introduces a substantial performance penalty, as since it must substantially increase the amount of inputs and the size of the circuit, to reckon multiple copies of the circuit. Given this issue in achieving security against malicious adversaries, the present version of FairplayMP handles solely the semi-honest case. [1]

Our alternative in the semi-honest model follows previous work towards privacy-preserving data processing like Lindell and Pinkas' construction for a privacy-preserving version in the ID3 decision tree learning algorithm for privacy-preserving classification.

The authors in [10], proposed a new algorithm for semi-honest model with negligible collision probability is a modified algorithm of privacy enhacing association rule mining on distributed homogenous dataset. This new set of rules embraces public key cryptosystem which overcomes the overheads involved in employing the algorithm with commutative encryption framework. The new algorithm is fast than old one by considering privacy and accuracy of results. One of the crucial features of this algorithm is scalability which implies the equal set of rules can be extended to any number of parties.

Informally, security of a protocol within the SMC paradigm is outlined as computational indistinguishability from some ideal capabilities, within which a trusty third party accepts the parties inputs and carries out the reckoning. The best capability is therefore secure by definition. The above mentioned protocol is secure if the adversaries view in any protocol execution is simulated so as to provide access solely to the best capability, i.e., the above mentioned protocol doesn't leak any data beyond what is given.

Any polynomial-time multi-party reckoning might be utterly done in a privacy conserving manner using generic techniques of Goldreich and Wigderson. All encompassing constructions, however, are generally impractical to their complexness. Latest analysis has targeted on finding more efficient privacy-preserving algorithms for distinctive troubles which has reckoning of approximations, auctions, set matching and intersection, surveys, reckoning of the k-th ranked element and particularly data processing issues like privacy-preserving reckoning of decision trees, classification of customer data, and mining of vertically partitioned data.[2]

3. Protocol For Unifying List Of Locally Frequent Itemsets

The protocol was instructed for reckoning the unified listing of all locally frequent item sets, at the same time as no longer uncovering the sizes of the subsets neither their substance. The protocol is carried out as soon as the parties already realize F_s^{k-1} —the set of all (k-1) item sets which are globally frequent, and they want to continue and reckon F_s^k. We tendency to test with it right here as Protocol (UNIFI) Unifying list of locally Frequent Item sets as in fig 3. The input that every party P_m has at the begin of Protocol UNIFI is the collection $C_s^{k,m}$, as defined in the FDM set of rules. Let $Ap(F_s^{k-1})$ mean the set of all applicant k-item sets that the Apriori algorithm produces from F_s^{k-1}. At that point, as implicit by the definition of $C_s^{k,m}$, $1 \leq m \leq M$, are all subsets of $Ap(F_s^{k-1})$. The output of the protocol is that the union $C_s^k = \cup_{m=1}^{M} C_s^{k,m}$.

In the first emphasis of this reckoning k=1, and consequently the parties reckon all frequent item sets with support 0.1 (here $F_s^0 = \{\emptyset\}$). In the next emphasis

they reckon all frequent item sets with support of 0.2, and so on, until the k≤ L in which they find no s - frequent k-item sets.

After reckoning the union, the parties proceed to extract from C_s the subset F_s^k that comprises of all k-item sets that are globally frequent. At last, with the aid of applying the above procedure from k=1 till the first value of k≤ L that the ensuring set was void, and also the parties ought to recover the complete set of all globally k=1 frequent item sets.

Protocol UNIFI function as follows: initially, every party provides to his non-public subset $C_s^{k,m}$ fake item sets, as a way to awning its size. Then, the parties collectively reckon the encryption of their non-public subsets by applying on those subsets a homomorphic encryption, wherever each party adds, in his flip, his own layer of encryption using his non-public secret key. At the endure stage, each item set in every subset is encrypted by using all the parties. Then, they reckon the abutment of these subsets in their encrypted shape. At last, they decipher the abutment set and abolish the item sets that are identified as fake.

A homomorphic encryption arrangement is a tuple
(KeyGen, Enc, Dec, Eval)
Generate random public/secret key-pair $c=Enc_{pk}(m)$
Encrypt a message with the public key $m=Dec_{sk}(c)$
Decrypt a cipher textual content with the secret key
$(P, c_1,\ldots,c_n) = c^* = Eval_{pk}(P, c_1,\ldots,c_n)$
$c^* = c_1 \times c_2 \ldots \times c_n = (m_1 \times m_2 \ldots \times m_n)^e \pmod N$

Homomorphic encryption library that executes homomorphic encryption. Our main impelementation is “stuffed”, specifically the complete AES state is organized in barely one ciphertext. As of currently accessible may be a usage of the BGV plan, along with numerous optimizations to accomplish homomorphic assessment runs faster, focusing mostly on able use of the Smart-Vercauteren ciphertext packing tactics and therefore the Gentry-Halevi-Smart optimizations.

Protocol (UNIFI) Unifying lists of locally Frequent Item sets

Input: Every party P_m has an input subset $C_s^{k,m} \subseteq A_p(F_s^{k-1})$, $1 \le m \le M$
Output: $C_s^k = \cup_{m=1}^{M} C_s^{k,m}$

Step 1:Every party P_m encodes his subset $C_s^{k,m}$ as a binary vector r_m of length $n_k = | A_p(F_s^{k-1}) |$ in accord with the agreed ordering of $A_p(F_s^{k-1})$.

Step 2:Those parties invoke protocol THRESHOLD to reckon $r=T_1(r_1\ldots r_M)= V_{m=1}^{M} r_m$
Step 3:C_s^k is the subset of $A_p(F_s^{k-1})$ that is described by r.

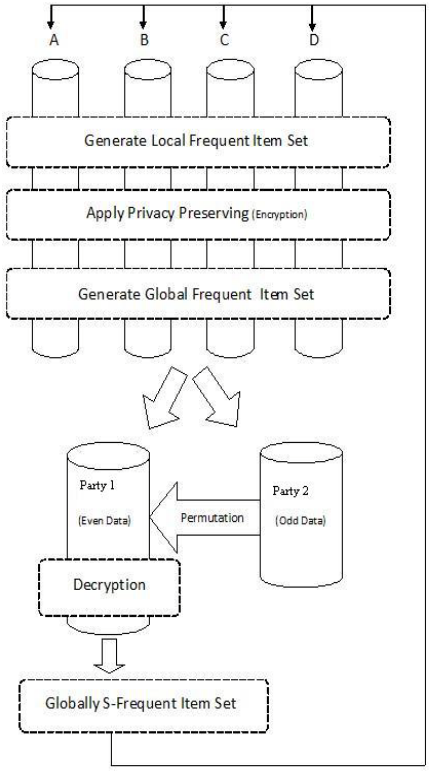

Fig 3:Architecure of UNIFI

Initialization
In this module let A contains set of items and D contains transactional dataset of transactions, every transaction contains a set of items. We have a tendency to denote the support of an itemset S ⊆ A as suppD(S) and also the frequency by freqD(S). For every item i, suppD(i) and freqD(i) denote respectively the individual support and frequency of i. Let D denote the initial transactional dataset that the owner has to guard the identification of individual items, and on that the owner applies an encryption function to D and transforms it. The term item shall mean plain item by default. The notions of plain item sets, plain transactions, plain patterns, transaction over some set of items, and each column represents one of the items in A. In other words, the (i,j) entry of D equals 1 if the ith transaction includes the item aj, and 0 otherwise. The dataset D is partitioned horizontally between M parties, denoted $P_1 \ldots P_M$. Party P_m holds the partial data D_m that contains Nm =

|Dm| of the transactions in D. An item set 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. Every party selects a random private key and hash functions to apply to all item sets.

Candidate and Pruning

Each party P_m computes the set of all k-1 item sets that are locally frequent in his site and also globally frequent; namely, P_m computes the set locally as well as globally frequent sets. Then the Apriori algorithm is applied in order to generate the set B of candidate k-item sets. For each item in B, P_m computes suppm(X). Then retains only those item sets that are locally s-frequent. The collection of these item sets are denoted by C.

4. Threshold Protocol

Protocol UNIFI firmly computes of the union of non-public subsets of some publicly better recognized ground set by reckoning the union of non-public vectors. Certainly, if the ground set Ω, then any subset R of Ω can be delineated by using the feature binary vector $r \in Z_2^n$ where $r_i=1$ if and provided that $w_i \in R$. Let r_m be the binary vector that characterizes the non-public subset held by party P_m , $1 \le m \le M$. Then the union of the non-public subsets is delineated with the aid of the OR of those non-public vectors. It is a lot of easier than Protocol UNIFI and employs less cryptological primitives. This function computes a wider vary of features, that we tend to threshold protocol functions.

Let $P_1 \ldots P_M$ be M parties wherever P_m has an input binary vector $r_m \in Z_2^n$, $1 \le m \le M$. This feature reckons, in an exceedingly secure manner, the output vector, for some $1 \le t \le M$. Let a be the sum of the input m=1 binary vectors. Since $a(m) \in Z_{M+1}=\{0,1,\ldots M\}$, for all $1 \le m \le M$, the sum vector a may be seen as a vector in Z_{M+1}^n. The concept behind the protocol is to use the secure summation protocol of [20] to reckon the shares of the sum vector a, so use those shares to firmly verify the threshold conditions in every part.
Since $a \in Z_{M+1}^n$, each party starts by making random shares in Z_{M+1}^n of his input vector (Step 1); notably, P_m selects M random vectors in Z_{M+1}^n that add up to b_m $1 \le m \le M$. In Step 2, all parties send to any or all parties the corresponding shares in their input vector. Then (Step 3), party P_l, $1 \le l \le M$ adds the shares that he got and arrives at his share, s_l , within the total vector $a := \Sigma_{m=1}^{M} b_m$. particularly, $a = \Sigma_{l=1}^{M} s_l \bmod(M+1)$. and, moreover, any M-1 vectors out of $\{s_1 \ldots s_M\}$ don't reveal any data on the total a. In Steps 4-5, all parties, except the last one, send their shares to P_1 who adds them up to get the share s. Now, parties P_1 and P_M hold additive shares of the sum vector a: P_1 has s, P_M has s_M , and $a=(s+s_M) \bmod (M+1)$. It is now needed to check for each component $1 \le i \le n$ whether

$$(s(i)+s_M(i)) \bmod (M+1) < t \text{---------(2)}$$

Whenever the inequality (2) holds, then the result is set to zero in the binary vector; otherwise, the value is onein the binary vector. We proceed now to discuss the secure verification of inequality (2). The above inequality is equal to the following set admittance:

$$(s(i)+s_M(i)) \bmod (M+1) \in \{j:0 \le j \le t-1\} \text{------(3)}$$

The admittance in (3) is equal to

$$s(i) \in \Theta(i) := \{(j- s_M(i)) \bmod (M+1): 0 \le j \le t-1\} \text{----(4)}$$

THRESHOLD PROTOCOL

Input: Each party P_m has an input binary vector $b_m \in Z_2^n$ $1 \le m \le M$
Output: $r=T_t(r_1 \ldots r_M)$
Step 1: Every P_m selects M random share vectors $r_{m,l} \in Z_{M+1}^n$, $1 \le l \le M$ such that $\Sigma_{l=1}^{M} r_{m,l} = r_m \bmod(M+1)$.
Step 2: Every P_m sends $r_{m,l}$ to P_l for all $1 \le l \ne m \le M$.
Step 3: Every P_l computes $s_l=(s_l(1)\ldots. s_l(n))= \Sigma_{m=1}^{M} r_{m,l} \bmod(M+1)$.
Step 4: Parties P_l $2 \le l \le M-1$, send s_l to P_1
Step 5: P_1 reckon $s=(s_l(1)\ldots. s_l(n))= \Sigma_{l=1}^{M-1} s_l \bmod(M+1)$.
for i=1….n do
Step 6: if $(s(i)+s_M(i)) \bmod (M+1) < t$ set b(i)=0 otherwise set b(i)=1.
end for

5. Set Admittance

Protocol SET ADMITTANCE involves three parties: P_1 incorporates a vector s=(s(1)…s(n)) of parts in some ground set Ω; P_M, however, incorporates a vector Θ=(Θ(1)…Θ(n)) of subsets of that ground set. The predefined yield may be a vector v=(v(1)….v(n)) that portrays the corresponding set admittance within the accompanying way:

v(i)=0 if $s(i) \in \Theta(i)$ and
v(i)=1 if $s(i) \notin \Theta(i)$, $1 \le i \le n$.

The reckoning within the protocol involves a 3rd party P_2. (Whilst SET ADMITTANCE is named to as from THRESHOLD PROTOCOL, the ground set is

$\Omega=Z_{M+1}$ and the inputs s(i) and Θ(i) of the 2 parties are as in Eq. (4), $\leq i \leq n$.) The protocol begins with parties P_1 and P_M concurring on a hash technique, and a its respective secret key K (Step 1). Consequently (Steps 2-3), P_1 changes his sequence of parts s=(s(1)…s(n)) into a chain of corresponding "signatures" s'=(s'(1)…s'(n)), wherever $s'(i)=h_K(i,s(i))$ and P_M will have an analogous 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 $\Theta'(i)$ $\leq i \leq n$, wherever the parts within each subset are capricious permuted. In the end (Steps 6-7), P_2 performs the relevant admittance verifications on the signature values. On the off chance that he finds out that for a given $1 \leq i \leq n$, $s'(i) \in \Theta'(i)$, he may deduce, with high likelihood, that $s(i) \in \Theta(i)$, whence he sets v(i)=0. If, however, $s'(i) \notin \Theta'(i)$, then, with certainty, $s(i) \notin \Theta(i)$, and so therefore he sets v(i)=1.

Encryption of ItemSets

In this module we introduce the encryption scheme, which transforms a TDB D into its encrypted version D. This scheme consists of main steps that is using hashing techniques. All parties compute a composite encryption of the hashed sets C. In first steps each party P_m hashes all item sets in C and then encrypts them using the key Km. Hashing is needed in order to prevent leakage of algebraic relations between item sets. Then, he adds to the resulting set faked item sets in order to hide the number of locally frequent item sets that he has. Then the parties start a loop of M cycles, where in each cycle they perform the following operation: Party P_m sends a permutation of X_m to the next party P_{m+1}; Party P_m receives from P_{m-1} a permutation of the set Xm-1 and then computes a new X_m as X_m. At the end of this loop, P_m holds an encryption of the hashed C using all M keys.

Merging Itemsets

In this module, the parties merge the lists of encrypted item sets. At the completion of this stage P_1 holds the union set C hashed 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 C those fake item sets 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. Not all lists are merged at once since if they were, then the party who did the merging would be able to identify all of his own encrypted item sets as he would get then from P_M and then learn in which of the other sites they are also locally frequent. The merging is carried out according to steps below: Each odd playe sends his encrypted set to parties P_1, Each even party sends his encrypted set to party P_2, P_1 unified all sets that were sent by the odd parties and removes duplicates, P_2 unifies all sets that were sent by the even parties and removes duplicates, P_2 sents its permuted list of itemsets to P_1, P_1 unifies its list of itemsets and list received from P_2 and the removes duplicates from the unified list.

SET ADMITTANCE

Input: P_1 has a vector $s=(s_l(1)\ldots s_l(n))$ and P_M has a vector $\Theta=(\Theta(1)\ldots\Theta(n))$ where for all $1 \leq i \leq n$, $s(i) \in \Omega$ for some ground set Ω.
Output: The binary vector v= (v(1)…v(n)) where v(i)=0 if $s(i) \in \Theta(i)$ and v(i)=1 otherwise $1 \leq i \leq n$.

Step 1: P_1 and P_M agree on a hash function and using a secret key K.
Step 2: P_1 computes s'= (s'(1)…s'(n)) where $s'(i)=h_K(i,s(i))$ $1 \leq i \leq n$.
Step 3: P_M computes $\Theta'=(\Theta'(1)\ldots..\Theta'(n))$ where $\Theta'(i)=\{h_K(i,\theta): \theta \in \Theta(i)\}$, $1 \leq i \leq n$.
Step 4: P_1 sends to P_2 the vector s'
Step 5: P_M sends to P_2 the vector Θ' in which each Θ(i) is capricious permuted.
for all $1 \leq i \leq n$, P_2 sets v(i)=0 if $s'(i) \in \Theta'(i)$ and v(i)=1 otherwise
Step 6: P_2 broadcasts the vector v=(v(1)…..v(n))

Decryption of Itemsets

In this module, a similar round of decryptions is initiated. At the end, the last party who performs the last decryption uses the lookup table T that was constructed in Step 4 in order to identify and remove the fake item sets and then to recover C. Finally, he broadcasts C to all his peers. For all playes, the last party decrypts all itemsets in encrypted C using corresponding K, and sends permuted and decrypted K to other party. It also decrypts all itemsets in EC, it also uses the lookup table to replace hash values with the actual itemsets and to identify and remove faked itemsets. The party finally broadcasts decrypted C.

6. Results

We compared the achievement of secure implementations of the FDM algorithm and UNIFI. Total reckoning time of the function protocols solely (FDM and UNIFI) over all parties.
We ran 3 experiment sets, where every set tested the based on the dependence of the measures on a distinct parameter:
N—the range of transactions within the feduni dataset,

M—the range of parties, and
s—the threshold support size.
The dataset that was used is mushroom dataset from the UCI repository [19] with 8124 instances.
From the intial set of experiments, we will see that N has very little result on the runtime of the unification protocols, FDM and UNIFI.
However, since the time to identify the globally frequent item sets will grow linearly with N, and same process is carried out in the same manner in FDM, the advantage of Protocol UNIFI over FDM in terms of runtime decreases with N. As an example, with N=100 the total computation times for FDM and UNIFI were 3812 and 3843 milli seconds, respectively, which gives an improvement factor of 3.1.)

SNO.	No. Of Transactions	Computational time of Proposed system	Computational time of Existing system
1	100	3812	3843
2	200	5437	9328
3	300	9406	18937
4	400	21187	36594
5	500	24313	46141
6	1000	43875	101000
7	2000	97437	164250
8	3000	269437	330828
9	4000	362093	797188

Fig 6.1 Table for Analysis

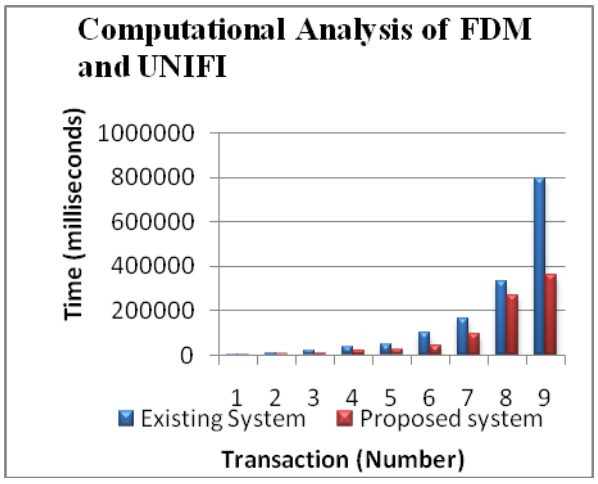

Fig 6.2 Graph to Analyse Computation Time

7. Conclusion

In this paper we tend to devise a protocol for reckoning association rules within a scenario of homogeneous data. The protocol is more efficient than current leading K and C protocol. Multi-party Computation being employed in big datasets that needs to preserve privacy of the private data with respect to various parties. Those practices exploit the truth that the main problem can be of interest only once the amount of parties are more than two. The future work is to devise an most efficient protocol for inequality verifications that uses the existence of semihonest third party and another in the implementation of techniques to the matter of distributed association rule mining in vertical setting.

8. 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] 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, Sept. 2004.*
[3] chin-chen chang, yu-chiang Li An Efficient Algorithm for Incremental Mining of association Rules *15t* (Tassa, 2014)*h international workshop IEEE (2005).*
[6] D.W. Cheung, V.T. Ng, A.W. Fu, Y. Fu, "Efficient mining of association rules in distributed databases", *IEEE Transactions on Knowledge and Data Engineering, vol 8, no 6, pp. 911-922, 1996.*
[7] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007.
[8] J.S. Park, M. Chen, P.S. Yu, "An effective hash-based algorithm for mining association rules", In Proceedings of ACM SIGMOD International Conference on Management of Data, San Jose, California, pp. 175-186, May 1995.
[9] L. Aouad, N. Khac, T. Kechadi, "Performance study of distributed Apriori-like frequent item sets mining", Knowledge Information System, no 23, pp. 55-72, 2010.
[10] lberto Trombetta, Wei Jiang, Elisa Bertino, "Privacy Preserving Updates To Anonymous And Confidential Databases",*IEEE Transactions On Dependable And Secure Computing*, Vol. 8, No. 4, PP. 578-587, 2011.
[11] Murat Kantarcioglu, Chris Clifton, "Privacy-Preserving Distributed Mining Of Association Rules On Horizontally Partitioned Data", *Knowledge And Data Engineering,IEEE Transactions*,Vol. 16, No.9, 2004.
[12] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing, 61(3):350-371, 2001.*
[13] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *In Proc. 20th Int. Conf. on Very Large Databases, pages 487–499, Santiago, Chile, 1994.*

[14] Ratchadaporn Amornchewin Probability-based Incremental Association Rules Discovery Algorithm with Hashing Technique. *[International Journal of Machine Learning and Computing, Vol.1, No. 1, April 2011.*
[15] Ratchadaporn Amornchewin, Worapoj Kreesuradej Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm *[IEEE 2007]*.
[16] W. Ailing, "An Improved Distributed Mining Algorithm of Association Rules", JCIT: *Journal of Convergence Information Technology, vol 6, no 4, pp. 118-122, 2011.*
[17] J.S. Park, M. Chen, P.S. Yu, "An effective hash-based algorithm for mining association rules", In Proceedings of *ACM SIGMOD International Conference on Management of Data, San Jose, California, pp. 175-186, May 1995.*
[18] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, "Tools for privacy preserving distributed data mining", *ACM SIGKDD Explorations, 2003.*
[19] www.ics.uci.edu refers dataset.
[20] Yao,A.C.C.: How to generate and exchange secrets (extended abstract).In Proceedings of the 27th *IEEE Symposium on Foundations of Computer IEEE Pr*