Available at: www.researchpublications.org

A Technique for Improving Security on Horizontally Distributed Databases by Association Rule Mining

Rohinee K. Deshmukh[1], Sandeep Khanna[2]

[1] *Post Graduate Student, Department of CE, Padm. Dr. V.B.K.C.O.E., Malkapur, S.G.B.A. University, Maharashtra, India*

[2] *Asst. Professor, Department of CSE, Padm. Dr. V.B.K.C.O.E., Malkapur, S.G.B.A. University, Maharashtra, India*

[1]rohinee910d@gmail.com

[2]sandeeppietse@gmail.com

Abstract— **Data mining is the fast growing area today which is the process of Semi-automatically analyzing large databases to find useful patterns. The association rule technique is used in data mining for revealing some interesting relationship between locally and globally large item sets. The current leading protocol is Kantarcioglu and Clifton which is also known as K&C protocol. This paper introduces a protocol which is based on unsecured distributed version of Apriori algorithm which is known as Fast Distributed Mining (FDM) algorithm, that generate small number of candidate sets. The main ingredients in this 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 an element held by one player is included in a subset held by another. This protocol offers enhanced security with respect to the earlier protocol.**

Keywords— Distributed computation, Secure mining, Association rules, Frequent item sets, Multi-party

I. INTRODUCTION

Data miningis the process of semi-automatically analyzing large databases to find useful patterns. i.e. to extract important knowledge from large data collections but sometimes these collections are split among various parties. Privacy liability may prevent the parties from directly data sharing, and some types of information about the data. This paper studies the problem of association rules mining in horizontally distributed databases & their solution. In the distributed databases, there are several players that hold homogeneous databases which share the same schema but hold information on distinct entities. The goal is to find association rules with support '**s**' and confidence '**c**' to minimize the information disclosed about the private databases held by those players [1].

In distributed databases, there are N numbers of different sites containing database $D_1, D_2, \ldots, D_n$ respectively. In horizontally partitioned databases, the database D separating into different parts $D_1, D_2, \ldots, D_n$ such that each part D_i contains same attribute set X_i but distinct set of data tuples. For each database the set of shared attributes S which is the same as Xi. In vertically distributed databases, the database D separating into different parts $D_1, D_2, \ldots, D_n$, such that each part D_i contains same may share some attributes with another database D_j where i is not same as j. Vertically partitioned datasets to share knowledge across the different participating nodes.

Association rules are if-then statements that help uncover relationships between seemingly unrelated data in a relational databases or other information storage [2]. An example of association rule is "If a customer who purchases a computer, also tend to buy antivirus software". In association rules an antecedent which is if part is on left side and a consequent which is then part is on right side. An antecedent is an item found in the data. A consequent is an item that is founded which combined with the antecedent. Association rules are generated by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is measure of what fraction of population satisfies both the antecedent and the consequent of the rule. Confidence is measure of how often consequent is true when antecedent is true.

This paper defines a secure multi-party computation problem, there are various sites (or players) which hold homogeneous databases that share same schema but hold information on different entities. There are M players that hold private inputs, $x_1, \ldots, x_M$, and they want to compute securely $y = f(x_1, \ldots, x_M)$ for some public function f. If there is trusted third party is exist, the players can give to him their inputs and he perform the function evaluation and send to them the resulting output. If such a trusted third party is absent, it is needed to create a protocol that the players can run on their own in order to arrive at the required output y. If no player can learn from his view of the protocol more than what he would have learn in the idealized setting where the computation is carried out by a trusted third party then that protocol is considered as perfectly secure. In our problem, the inputs are partial databases and generate list of association rules that hold in unified databases with support and confidence not smaller than given threshold s and c respectively.

Kantarcioglu and Clifton studied that problem in [3] and develop a protocol for its solution. The main part of protocol is sub-protocol for the secure computation of union of private subsets that are held by different players. The improved protocol is based on two novel secure multi-party algorithms using these algorithms the protocol provides enhanced privacy, security and efficiency as it uses commutative encryption.

In this paper the improved UniFI protocol is involved which include two secure multiparty algorithms:

1. Computing the intersection and union of own subsets that each interacting players hold.
2. Tests an element held by one player included in subset held by another.

The protocols of [3], as well as herein, are based on the Fast Distributed Mining (FDM) algorithm of Cheung et al. [4], which is an unsecured distributed Apriori algorithm and anonymous ID assignment [5]. In that player finds their locally s-frequent item sets then the players check each of them to find out globally s-frequent item set. The FDM algorithm is as follows:

1. Firstly Players must be initialized
2. Secondly each player must compute the set of (k-1) item sets.
3. Divide the item sets and retains only those which are locally s-frequent
4. Broadcasting of item sets by individual players.
5. Next local support is computed.
6. Finally mining result is broadcasted.

The FDM algorithm is start by finding all globally s-frequent single items. Then it finds all globally s-frequent 2-item sets, till it finds the longest globally s-frequent item sets. At the $(K + 1)^{th}$ iteration of the FDM it will find no (K + 1)-globally s-frequent item sets, where K is the length of that item sets, and then it terminates.

II. LITERATURE REVIEW

The before old work in secure data mining has considered two related settings. first the data owner and the data miner are two different entities, and then, 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 main goal is to protect the data records from the data miner. The main approach in this context is to apply data perturbation [6], [7]. The idea is that the perturbed data can be used to infer general trends in the data, without exposing 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 multi-party computation. The typical method is cryptographic instead of probabilistic. The problem of distributed association rule mining in the vertical partitioning was studied in [8], where each party holds a different set of attributes, and the horizontal partitioning in [1], [9], but large-scale systems considered in [9], on top of the parties that hold the data records there are also managers which are computers that assist the resources to decrypt messages; another assumption made in [9] that distinguishes it from [3] 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 [10], [11], as well as in [3]. Freedman et al. [6] proposed a privacy-preserving protocol for set intersections. It used to compute also set unions through set complements. Kissner and Song [11] present a method for representing sets for polynomials, and give several privacy-preserving protocols for set operations. They also consider the threshold set union problem that related to the threshold function. The communication overhead of the solutions in those two works, as well as in [3]'s and in our solutions, depends linearly on the size of the ground set. However, as the protocols in [10], [11] use homomorphic encryption, while that of [3] uses commutative encryption. The problem of set inclusion can be seen as a simplified version of the privacy-preserving keyword search. If $w = x_i$ for some $1 \leq i \leq n$ which is one of the server's keywords, the client should get the consistent p_i. The client should get notified thatin case w differs from all x_i. Freedman et al. [6] solved the privacy requirements are that the client gets no information about other pairs in the server's database and the server gets no information about w. Suppose we take the empty string, only information the client gets is whether or not w is in the set $\{x_1, . . . , x_n\}$. However, the privacy-preserving keyword search problem reduces to the set inclusion problem.

III. RELATED WORK

In this topic we discuss the secure implementation of step 4 in the FDM algorithm, i.e., the secure computation of the union of .

A. *All locally Frequent Item set Secure Computation*

In this section, we explain three protocols, first is used for unifying list of locally frequent item sets. Second is for computing the OR of private binary vectors. Third is computing the set inclusion. Commutative encryption as [3] is an important tool that can be used in many Privacy-preserving protocols. And the fourth protocol is used for the secure computation of locally frequent item sets.

1) UniFI-KC Protocol:

UniFI-KC Protocol of [3] is proposed by Kantarcioglu and Clifton for calculating unified list of all locally frequent item sets and they wish to continue and calculate from which is set of all (k-1) globally s-frequent item sets.UniFI-KC Protocol is used for, to secure the locally frequent item sets using private key and hash function and also remove the faked item sets Xm which are generated at time of encryption.

Let, Ap ( ) be the set of all applicant k-item sets are generated by Apriori algorithm from set of globally s-frequent item sets . The input is collection of item sets , which is defined in Steps 2-3 of the FDM algorithm. The is subset of Ap( ). The outputof the protocol is the union of which is . In the first iteration the players compute all s-frequent 1-item sets at k = 1. In the next time they calculate all s-frequent 2-item sets, and so forth, till the k< L in which they not find s-frequent k-item sets.

Steps of UniFI-KC protocol is as proceed:

1. Select the cryptographic primal

- Player chooses commutative cipher and its consistent random secret keyK_m to each player P_m.
- For encryption, players choose a hash function h to apply on all item sets.
- The players construct a lookup table with hash values of all candidate item sets which is from Ap(F_s^{k-1}).

2. Encrypts the all candidate item sets

- Players hashed all item sets in $C_s^{k,m}$, and encrypts that using a secret random key K_m.
- Players adds to consequent set X_m which is faked item sets till its size becomes | Ap(F_s^{k-1})|
- Player transmits the arrangement of X_m to next Player and takes the arrangement of X_{m-1} from previous playerfor M-1 times.
- Player calculates a newconsequent set X_m by encoding the previous player's consequent set X_{m-1} using random secret key K_m.
- Player hold an encryption of hashed candidate set $C_s^{k,m+1}$using all M players.

3. Combining the Item sets

- Player combines the list of encrypted item sets and computes the union of private subset.
- For combining item sets, firstly combine the each odd and even players and that are sends his encrypted set to player P_1 and P_2 respectively.
- P_1 combines the item sets list which is sent by odd and even players and removes duplicates from that list. The final list denoted by EC_s^k.

4. *Decrypts the candidate item sets*

- Player decrypts all item sets inEC_s^k, using secret random key K_m, the consequent set byC_s^k.
- For replacing hashed values with actual item sets and identifying and removing the fake item sets, player operates the lookup table T. Then retrievesC_s^k.
- P_m transmits C_s^k to all his peers.

2) t-Threshold Protocol:

Protocol Threshold is a secure multi party protocol for computing the OR of private binary vectors. The UniFI-KC protocol safely calculates the union of private subsets of publicly known ground set Ap(F_s^{k-1}).That problem is similar to problem of calculating OR of private vectors. Actually, if the ground set is, Ω={$\omega_1,\ldots,\omega_n$}then any subset B of Ω may be described band employs less cryptographic primitives. The Protocol t-Threshold computes a larger range of functions, is known as threshold functions. That protocol is use the secure summation protocol of [12] in order to compute shares of the sum vector and then use those shares to securely verify the threshold conditions in each component.

Let b=($b_1,\ldots,b_n$) be the characteristic binary vector where b_i = 1 if and only if ω_i □ B otherwise it is zero. The own subset's union set is illustrated by the OR of those own vectors b =$V_{m=1}^{M} b_m$. Protocol t-Threshold is used for calculating function which can be evaluated and protected by generic solutions mentioned in [13], [14]. The threshold function defined in [1] is more efficient than those generic solutions and simple to understand the program. It is also much simple than Protocol UNIFI-KC.

The OR function of b_m is the 1- threshold function which is same as$T_1(b_1,..,b_m)$, and the AND function of b_m is M-threshold function which is same as$T_m(b_1,..,b_m)$. Those cases may be used for secure computation of locally frequent item sets, to compute in a privacy-preserving manner unions and intersections of private subsets.

Steps of Threshold protocol are as follows:

1. Firstly player chooses the random shares in input binary vector and sends the consequent share to all other players.
2. Each player calculates the s_l by adding the shares and sends to P_1.
3. P_1calculates s by adding the all $s_l$of M-1 players.
4. Players P_1 and P_M hold additive shares of the sum vector a: P_1 has s, P_M has s_M.
5. The set b(i)=0, 1≤ i ≤ n, if $s(i) + s_M(i) \bmod (M+1) < t$ otherwise set b (i) =1.

The only P_1 knows the value of s(i) while only P_M knows the set θ(i). In the OR function, t = 1, which is appropriate for us, the set (i) is of size 1, and therefore it is the problem of insensitive string comparison, which is solved in, e.g., [15]. Then M >2, invoke secure protocols of [6] or [15] no need to invoke. Actually, as M > 2, the existence of other semi-honest players can be used to verify the inclusion much more easily. This is done in Protocol 3 (SetInc) which we proceed to describe at next.

3) SetInc Protocol:

Protocol SetInc included three players: P_1 has a vectorsof elements in some ground set Ω, $P_M$has a vector $\theta$of subsets of that ground set and the output which isrequired as a vector bthat describes the consequent set inclusions in the following manner:b(i)=0 if s(i)$\epsilon\theta(i)$andotherwiseb(i)=1, where 1 ≤ i ≤ n. The calculation in the protocol includes a third player P_2.

Steps for SetInc Protocol as proceed:

1. Players P_1 and P_M agree on a keyed hash function (e.g., HMAC [16]), and a corresponding secret key K.
2. Player P_1 converts his sequence of elementssinto corresponding signatures,s', where s' is the keyed hash function of s and P_M does a similar conversions to the subsets which that he holds.
3. Player P_1 sends s' to P_2, and P_M sendsθ' to P_2 the subsets$\theta(i)$, 1 ≤ i ≤ n, the elements within each subset are randomly arranged.

4. Player P_2 performs the significant inclusion verifications on the signature values. If he discover that for a given $1 \leq i \leq n$, P2 sets,b(i)=0, if , s'(i) □ θ'(i) ,otherwiseb(i)=1.
5. Player P_2transmits vector b.

The liability of Protocol THRESHOLD-C to association is not important because of two reasons:

i. The sum vector entries do not show information about particular input vectors.
ii. The players P_1, P_M and P_2areconspire together to study information else where the intention of the protocol.

4) An Improved Protocol:

An improved protocol is used for the secure calculation of all locally frequent item sets. The set of all globally frequent (k - 1)-item sets denoted byAp(F_s^{k-1}).Apriori algorithm applied on F_s^{k-1} and generates the set of k-item sets.The sets of locally frequent k-item sets, $C_s^{k,m}$, $1 \leq m \leq$ Mare subsets ofAp(F_s^{k-1}).They may be encoded as binary vectors of length|Ap(F_s^{k-1})|.The binary vector which is encoded in the union of $C_s^{k,m}$which is the OR of the vectors that encoded form of the locally frequent item sets $C_s^{k.m}$. By invoking Threshold-C Protocol on binary input vectors, the players can calculate the union. That is summarized in Protocol 4 (UniFI).

Steps of UniFI protocol:Securely unifying lists of all locally frequent item sets:

1. Encodethe subset $C_s^{k.m}$by each player $P_m$as binary vector b_m of length|Ap (F_s^{k-1})|.
2. The players invoke protocol Threshold-C for calculating bwhich is same as OR of b_m.
3. Ap(F_s^{k-1}) is the superset of C_s^k which is asserted by b.

5) Privacy:

We start by analyzing the privacy which tendered by Protocol UniFI-KC which does not respect perfect privacy since it discover to the player information that is not implied by their own input and the final output. By using the fake item sets each player increases the Xm set in step 2 of Uni-KC protocol. Those fake items sets are random string chosen commutative cipher are not necessary for avoiding hash and encryption calculations. At the end of step 1, the possibility of two players selecting a random string is negligible. Therefore, each encrypted item set exists in two separate lists that indicate the high probability a true item set that means locally s-frequent in both of the consistent sites.

The Protocol UniFI-KC exposes the following excess information:

1. P_1 and P_2 may speculate for any subset of the odd players and even players respectively, the number of item sets that are locally supported by all of them.
2. At least one odd player and at least one even player support the number of item sets which may be speculated by P_1.
3. If P_1 and P_2 conspire together, and they expose for any subset of the players the number of item sets that are locally supported by all of them.

The privacy tendered by Protocol UniFI, there two cases are considered:

i. Without collusions - In that, the UniFI protocol tender computational privacy with respect to P_2 and exact privacy with respect to all players P_m, $m \neq 2$. The latter protocol exposes information to P_1 and P_2 if they not joinwith any other player therefore that privacy better than that offered by Protocol UniFI-KC.
ii. Withcollusions - There are both UniFI-KC and UniFI Protocols allow the joining parties to study prohibited information.

The excess information of Uni-FI protocol may be extracted by P_1 and $P_2$are commonly the number of frequent item sets between any subset of the players which might be benefited from collusion. While only P_1, P_2 and $P_M$in UniFI protocol can extract additional information if any two of them conspire together as in [1], they can study the sum of all private vectors. The number of sites in Ap(F_s^{k-1}) is frequent, when the sum make known for each specific item set in that.

To review without collusions, the UniFI protocol exposes no excess information, and, with collusions, the UniFI protocol exposes excess information leaves the partial databases totally identical; therefore it offers enhanced privacy preservation in comparison to Protocol UniFI-KC.

B. *Identifying the Globally s- frequent item set*

The UniFI-KC and UniFI Protocols produce the setC_s^k that consists of all locally s-frequent item sets in at least one site. Those k-item sets have potential to be also globally s-frequent. In order to expose those item sets is globally s-frequent there is a need to securely calculate the support of each of those item sets. That computation must not expose the local support in any of the sites.

Here we describe the solution that considered two possible settings which was proposed by Kantarcioglu and Clifton.

1. If the required output includes all globally s-frequent item sets, and also the sizes of their supports, then the values of $\Delta(x)$ can be exposed for all$x \in C_s^k$ In such a case, those values may be calculated using a secure summation protocol (e.g., [12]), where the private summand of P_m is$supp_m(x) - sN_m$.
2. The part of the required output not contains support sizes.

As $|\Delta(x)| \leq N$,an item set $x \in C_s^k$is s-frequent if and only if$\Delta(x) mod\ q \leq N$ for q = 2N + 1 and then to check the inequality by starting an implementation of the secure summation protocol of [12] on the private inputs$supp_m(x) - sN_m$ mod q. In such protocol, all players jointly calculate random additive shares of the required sum $\Delta(x)$ and then, by sending all shares to P_1, he may add them and expose the sum. The P_1 will have one random share, s_1(x), of, $\Delta(x)$,and P_M will have a consistent share,$s_M(x)$ that means,$s_1(x) + s_M(x)$ is same as $\Delta(x) \mod q$ if P_M with holds his share of the sum.

The Yao proposed the generic secure circuit evaluation. Yao's protocol is designed for the two-party case. Here, the setting, as M>2, there exist additional semi-honest players.

C. *All (s, c) Association rules*

The set F_s is of all s-frequent item sets is found, we start to look for all (s, c)-association rules that means support is at least sN and confidence is at least c, as described in [3].The X ⇒ Y holds with support s if s% of transactions in D contain X ∪ Y and with confidence c if c% of the transactions in D that contain X also contain Y. Rules that have s greater than a user-specified support is known as minimum support or threshold support and a c greater than a user-specified confidence is known as minimum confidence or threshold confidence.

The support of a rule is defined as,
supp(X) = number of transactions that contain X / total number of Transactions.

The confidence of a rule is defined as,
conf(X =>Y) = sup(X U Y)/ supp(X).

For X, Y is present in$F_s$where, $X \cap Y = \emptyset$ the compatible association rule X => Y has confidence at least c if and only if,$supp(X \cup Y)/supp(X) \geq c$. If $|C_{x,y}| \leq N$, then by taking q = 2N + 1, the players can verify in parallel, for all candidate association rules.

By induction, assume that we found all (s, c) -rules with j-consequents for all $1 \leq j \leq l$-1. To find all (s, c) -rules with l-consequents depends on X => Y is an (s, c) -rule only if X=>Y' were found to be (s, c) -rules for all $Y' \subset Y$if Z is belongs to $F_s$and Z=X ∪ Y where,$X \cap Y = \emptyset$ and |Y| = l. Hence, we create all candidate rules with l-consequents and test them in parallel.

It usually aims at finding association rules of the form X => Y where |Y| = 1, or at least |Y| = l for some small constant$l$. The above procedure may proceed till all candidate association rules, with no upper bounds of the resulting size, are found.

D. *Fully Secure Protocol*

In the step 2-4 in FDM algorithm,the players distribute the local pruning and union calculation and, test all candidate item sets in Ap(F_s^{k-1}) are globally s-frequent. That protocol is fully secure, as it exposes only the set of globally s-frequent item sets but no further information about the partial databases. As discussed in [3], such a protocol would be much more costly since it requires each player to compute the local support of |Ap(F_s^{k-1})| item sets in the k^th round instead of only $|C_s^k|$ item sets which is union set of $C_s^{k,m}$. The players will execute the secure comparison protocol to verify inequality (6) for |Ap(F_s^{k-1})| rather than only $|C_s^k|$, item sets. Both types of added operations are very costly: the time to calculate the support size relies linearly on the size of the database, while the secure comparison protocol entails a costly oblivious transfer sub-protocol. |Ap(F_s^{k-1})| is much larger than $|C_s^k|$, the added calculating time in such a protocol is expected to dominate the cost of the secure computation of the union of all locally s-frequent item sets as shown in [7]. Therefore, the enhanced security offered by such a protocol is accompanied by increased implementation costs.

IV. CONCLUSION

Data mining describes applications that look for hidden knowledge or patterns in large amounts of data. In this paper improved protocol for secure mining of association rules in horizontally distributed databases. The improved protocol get better the current leading protocol in terms of privacy. The one ingredients of two novel secure multi-party protocol is for calculating the union of private subsets that each of the interacting players hold. And another ingredient is a protocol that tests an element held by one player included in a subset held by another. That data mining has a very important role in our life, so we use and handle it regularly. Therefore privacy and security should be provided to database.

REFERENCES

[1] TamirTassa,"Secure mining of association rule in horizontally distributed databases" ,IEEE trans. Knowledge and Data Engg.,Vol. 26, no.2, April 2014.

[2] Ahmed M. Khedr and Raj Bhatnagar, "Agents for Integrating Distributed Data for ComplexComputations",Turk J ElecEngin, Vol. 14, No.2, 2006.

[3] 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.

[4] 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," Proc. Fourth Int'l Conf. Parallel and Distributed Information Systems (PDIS), pp. 31-42, 1996.

[5] Ms. R. Kalaivani, Ms. R. Kiruthika, "Automated Anonymous ID Assignment For Maintaining Data privacy", Proceedings of 2nd International Conference on Science, Engineering and Management, Srinivasan Engineering college, TamilNadu, India, March 28-29,2014.

[6] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword Search and Oblivious Pseudorandom Functions," Proc. Second Int'l Conf. Theory of Cryptography (TCC), pp. 303-324, 2005.

[7] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy Preserving Mining of Association Rules," Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 217-228, 2002.

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

[9] A. Schuster, R. Wolff, and B. Gilburd, "Privacy-Preserving Association Rule Mining in Large-Scale Distributed Systems," Proc. IEEE Int'l Symp.Cluster Computing and the Grid (CCGRID), pp. 411-418, 2004.

[10] M.J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), pp. 1-19, 2004.

[11]L. Kissner and D.X. Song, "Privacy-Preserving Set Operations," Proc. 25th Ann. Int'l Cryptology Conf. (CRYPTO), pp. 241-257, 2005.

[12] J.C. Benaloh, “Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret,” Proc. Advances in Cryptology (Crypto), pp. 251-260, 1986.

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

[14] D. Beaver, S. Micali, and P. Rogaway, “The Round Complexity of Secure Protocols,” Proc. 22nd Ann. ACM Symp. Theory of Computing (STOC), pp. 503-513, 1990.

[15] R. Fagin, M. Naor, and P. Winkler, “Comparing Information without Leaking It,” Comm. ACM, vol. 39, pp. 77-85, 1996.

[16] M. Bellare, R. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication,” Proc. 16th Ann. Int’l Cryptology Conf. Advances in Cryptology (Crypto), pp. 1-15, 1996.