International Journal of Engineering & Technology, 7 (3.13) (2018) 157-164

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Privacy Preserving Decomposable Mining Association Rules on Distributed Data

Ahmed M. Khedr[1]*, Zaher AL Aghbari[2], and Ibrahim Kamel[3]

[1,2] Department of Computer Science, University of Sharjah, UAE
[1]Mathematics Dept., Zagazig University, Zagazig, Egypt
[3]Department of Electrical and Computer Engineering, University of Sharjah, UAE
**Corresponding author: akhedr@sharjah.ac.ae*

Abstract

In distributed computing, data sharing is inevitable, however, moving local databases from one site to another should be avoided because of the computational overhead and privacy consideration. Most of the data mining algorithms are designed assuming that data repository is stored locally. This paper presents a scheme and algorithms for mining association rules in geographically distributed data. The proposed scheme preserves data privacy of the different geographical site by passing secure messages between them. The algorithms minimize the communication cost by exchanging statistical summaries of the local databases. We provide a privacy and security analysis that shows the privacy preserving aspects of the proposed algorithms. Moreover, the paper presents extensive simulation experiments to evaluate the efficiency of the proposed scheme.

Keywords: *Decomposable algorithm, secure data mining, association rules, vertically and horizontally distributed database.*

1. Introduction

Emerging network-based applications such as counter-terrorism, emergency response, public health awareness, and financial analysis require knowledge and data from a number of geographically distributed databases to make effective decisions. These sites usually belong to independent organizations and possibly using different databases scheme or models. In such application environment, a number of geographically distributed databases together form a global database that contains the relevant data to the computation. Some pattern discovery tasks may require data from different sites, for example querying census databases that is comprised of labor statistics databases and social security databases. Unless the data is integrated and the big picture is formed, it will be impossible to fully answer the pattern discovery tasks. Additionally there is a need to protect the privacy of data within each site (or organization).

The challenge we address in this paper is how to extract the necessary data from a number of geographically distributed databases without relocating databases. Most exiting data mining techniques assume that the data is centralized or the distributed data can efficiently move to one site to become a single model. Moving databases from one site to another results in performance penalty and security concerns. Traditionally, older techniques used brute force integration of the localized databases by moving databases from one site to another. This required translation of format and/or scheme, which are usually done by intermediate gateways. Although XML (extensible Mark-up Language) have facilitated the integration of various databases, yet the efficiency and privacy concerns remain. Organizations may be willing to share only their data scheme, not the data itself. These decentralized techniques have a high risk of privacy breach when full data from local databases is shared with other sites [1]. Organizations are looking for ways to decrease the risk of privacy breaching. It is desirable to have solutions that let the individual databases resides at one site and answers tasks using an implicit image of the global database. In this case, tasks and queries need to be decomposed into subtasks that can be performed locally.

This paper presents a scheme and algorithms for mining distributed data that are vertically distributed using association rule. We demonstrate how mining tasks can be answered in a distributed way. The proposed algorithms have the capability to decompose their computations to fit the nature of data distribution across the sites. Sites exchange summaries of the local databases while preserving privacy.

The distributed algorithms consist of localized computational steps that can be executed at the participating sites. The intermediate results are securely transmitted to the site that initiated the computation. The main contributions of this paper are summarized as follows:

- A scheme and algorithms for mining distributed data that are vertically distributed using association rule by exchanging privacy preserving summaries derived from local databases.
- The paper shows that the proposed distributed computations accurately generate the expected mining results.
- Simulation experiments that show the performance of the proposed solution.
- Theoretical analysis to show the complexity and the privacy preserving aspect of the proposed algorithm.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The rest of the paper is organized as follows: Section 2 presents the system model used in the study. Section 3 presents some required technical background and an overview of the latest techniques for solving the above mentioned problem. The details of the proposed algorithm is presented in Section 4. In section 5, we discuss the complexity and the privacy preserving properties of the proposed algorithm. Section 6 presents the simulation experiments and discuss their results. Finally, conclusion and future works are presented in Section 7.

2. Preliminaries

2.1 System Model

Fig. 1 is a high level system model of electronic health record sharing between autonomous healthcare providers that include hospitals, clinics and pharmacies. The databases of healthcare providers are in different geographic locations. These databases combined are considered one global database. Note that these databases usually cannot be physically moved from one site to the other due to several considerations such as security, size, privacy, or data ownership.
For the doctors to form the big picture about their patients, the data needs to be integrated.

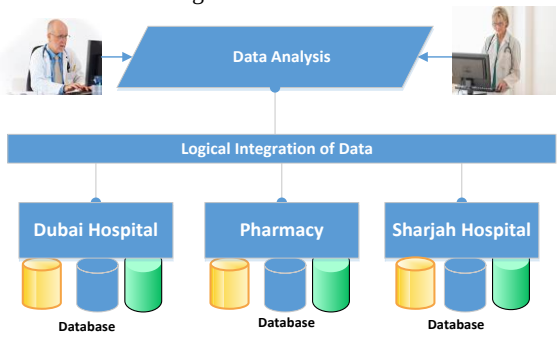


Fig. 1: Application Example (Electronic Health Record)

It is common that different healthcare providers may use different concepts to refer to the same object. For instance, the term influenza may be used in one healthcare provider whereas another healthcare provider may use the term flu. Since the different healthcare providers are willing to share data, they have to follow strict policies to secure the data. In this paper we assume that access can be regulated through appropriate authentication and authorisation mechanisms and this topic is out of the scope of this work.

2.2 Integration of Distributed Data

In a distributed setting, a database D is implicitly defined in n explicit databases D_i located at n different sites. We model a database D_i at the i^{th} site, by a relation that contains patient records (tuples). Each D_i contains a set of attributes. Due to the nature of geographically distributed local databases, data normalization cannot be assumed. A Join operation on all D_is is required to create the implicit database D. However, the tuples of D are distributed on the sites of the local databases, D_i's. This inability to make the tuples of D explicit (physically exist on one site) is the main problem addressed by the proposed privacy preserving mining algorithm.

2.3 Nature of Data Distribution

Vertically Distributed Databases: Here, each component D_i consists of tuples, where each tuple contain a different set of attributes. Some local databases D_j, $j \neq i$ may share some attributes with each other. Each D_i may contain unique attributes which are not shared with other D_j, $j \neq i$.

Vertically distributed databases require computations to be performed on the implicitly Joined, D. The decomposed algorithm utilizes the shared attributes (common attributes among D_is) in the performed computations. This formulation models more general circumstances than the case of a single key and non-overlapping attribute sets for single records distributed at various sites. Our target is to enable those local databases for participation that were designed independently and must have arbitrary overlap of attribute sets with the other local databases they have to collaborate with. We assume that the overlap between the attributes of the local D_i's results in one connected global database. The database for which the association rule is performed is the implicit cross product of the relations stored at the distributed sites.

Horizontally Distributed Databases: The global database D exists as a set of components $D_1, D_2, \ldots, D_n$ such that each D_i contains tuples consisting of an identical set of attributes A; but a distinct set of data tuples resides at each site. Each D_i resides on a different site and the tuples contained in all the D_i's, taken together, constitute the global database D. The decomposition of the proposed algorithm can be applied on horizontally and vertically distributed databases.

2.4 Computation Scenario

A number of vertically or horizontally distributed databases. Joining these databases form an implicit global database that consists of the relevant data to mining or other computational tasks. The main goal is to design a new algorithms that keep the local data as it is without moving and deal with visualized implicit join of the remote databases. Let us say $D_1, D_2, \ldots, D_n$ are n local databases and D is the implicit global database formed by Merging (in case of horizontal distributed) or Joining (in case of vertical distributed) all the participating local D_is. i.e., using D_i, the local results will be sent to be aggregated with other sites at the initiator site.
Our notion of data privacy requires that when the local computations are exchanged over the network, even if they are captured by someone, they should not enable reconstruction of any single tuple residing in any of the participating local databases. It is facilitated, partly, by the absence of knowledge of the aggregation and hash functions by the network intruder.

3. Related Works

In [2], the author introduced the association rule mining problem. It can provide valuable information about buying behavior of the customer. The association rule mining process involves 2 major sub-problems: discovering all frequent itemsets and generating association rules using frequent itemsets. The performance of mining using association rule is decided by the first sub-problem because each association rule can be discovered from the congruent frequent itemsets, for this reason, researchers concentrate on efficiently discovering frequent itemsets. In [3], the authors proposed apriori algorithm to effectively recognize frequent itemsets. Several proposed techniques to handle this problem such as level-wise techniques [1, 3, 13, 14] and pattern-growth technique [15, 16, 17, 18].

Many parallel algorithms are proposed to solve the association rules mining in parallel such [4, 5,6]. In Security Multiparty Computation (SMC), without a leak of knowledge some information about each part can be exchanged among multiparty to get the final results [7, 8]. High polynomial-time complexity is the main disadvantage of SMC 7, 8]. Many algorithms have been proposed using SMC such as finding decision trees [9], and mining of vertically partitioned databases [10].

In distributed environment, each data site of the multiple sites does not have to disclose its individual data, but it is demand from each site to exchange the global candidate itemsets and the corre-

sponding support counts [4,5]. In [11], the authors proposed an approach to reduce the complexity and enhance the security of distributed association rule mining on horizontally partitioned databases. In [12], the authors presented a technique to fasten the global candidate discovering and enhance the privacy preserving the frequent itemsets discovery.

In [23,24], based on the Fast Distributed Mining (FDM), the authors proposed two secure algorithm for mining association rules in horizontally distributed databases. Unsecured distributed version of association rule is given in [25]. The proposed algorithm provides privacy with respect to the protocol in [24]. In addition, it is significantly efficient in terms of exchanged messages and time. Moreover it is simpler.
The above works describe algorithms for the case when *n* records are distributed among a few local databases where each remote database includes a key to recognize some individuals, and some attributes are unique to the local site and are not shared with a local database at any other remote site. The proposed approach can be applied on the most general scenario where existing vertically or horizontally partitioned data would like to cooperate for mining without any constrains that there be only one record at each site for each individual and for any number of shared attributes. Also, the above algorithms are designed for environments where the relevant data are stored horizontally or vertically distributed data where different remote sites includes different attributes for a common set of entities. This is not the case of our distributed databases [29-33].

The proposed decompositions scenario can be seen as regular implemented distributed databases by a number of coordinated agents using message exchanging or visiting remote sites to gather local queries or/and computation results [19, 20, 21, 22]. Several issues related to knowledge distribution and processing capability over a loosely connected communication network have been addressed by Multi-agent systems [26, 27, 28]. In [26, 27, 28], the modeled agents have only a limited view of the global knowledge and resources. In contrast, the proposed technique is directed at systems where cooperative agents honestly access local results from other agents to develop concepts from their collective knowledge while minimizing messages exchanged and disclose data among sites. Also our agent at each site does not need to have any uncertainty about the state of data or knowledge of other sites or computing agents [29-33]. They only need to be requested for its local results and they would truthfully give the needed information look for by other agents. Therefore, the goal of the agents is to minimize the exchange information among themselves for performing the global computations [29-33]. In [19], proposed a technique to perform global tasks with minimum exchanged messages across the network. The global results is then locally performed by the initiating agent after receiving minimal summaries from other agents. However, the algorithm in [19] does not handle the privacy of the information communicated between the different agents. In this paper, we use the distributed computational algorithm different from that of [19], also, we provide a better privacy-preserving solution.

4. Decomposable Association Rule

4.1 Problem Statement

Given a distributed relational database, relational DB for simplicity, between *n* sites, where each site holds a vertical partition of the database $D_1, D_2, \ldots, D_n$. These *n* partitions constitute the global database *D*. Each D_i contains a set of attributes A_i and the union of all attributes of all partitions defines the set of attributes *A* of the global database *D*.

$$A = \bigcup_{i=1}^{n} A_i \quad (1)$$

A subset of attributes s_{ij} that is *shared*, or common, between partitions D_i and D_j.

$$s_{ij} = \bigcup_{x=i,j} A_x \quad (2)$$

The union of all attributes between all partitions constitutes the shared set of attributes *S* of *D*. Thus, *S* contains all attributes that are shared between all sites. We need to perform global computations, such as computing association rules of the implicit database *D* at one of the sites, say the initiator site D_{init}, without moving the local tables of the different partitions to D_{init} due to the security, privacy and size of the local databases. Therefore, the global computation should be decomposed into local computations. Each site should perform the needed local computation keeping in mind the constraints of shared attributes and that the local results should contribute to the global solution at D_{init}.
To support the confidentiality of the data, D_{init} should not know the actual distinct values of the shared attributes nor the number of tuples of each distinct value of the shared attributes at each site.

Table 1: Symbol abbreviation

Symbol	Description
D	Global implicit database that is a union of all local databases
D_i	Local database at site i
D_{init}	Initiator site that starts the process of global computation
A	Set of attributes of the global database
S	Set of shared attributes between all partitions
Shared	The Cartesian product of the received distinct values of S attributes
$C^{D_i}()$	A function that counts the number of transaction in D_i
G_i	The agent, at a site i

4.2 Proposed Solution

To decompose the global computation of the association rules, i.e. finding the association rules in the global database *D*, we represent each site by an agent. Each agent can access any part of its local database. All agents are *mobile* that is they are capable of performing computation locally at their respective sites and at the same time are capable of moving from one site to another to perform computations or collect statistics. These agents cooperate to find the global association rules without moving local databases between sites, however agents may exchange messages and communicate summaries of their local results.
To start the global computation, an agent at D_{init} sends a request to the agents of the participating sites to start the local computations. The proposed algorithm is designed to minimize across site communication and different sets of shared attributes between sites. Furthermore, the presented approach maintains the privacy of the communicated data.

4.3 Decomposition of Global Computation

To illustrate the decomposition process, let's assume that the result *R* of a desired global computation, such as finding the count of a candidate frequent itemset I_j in the global database *D*, is $C^D(I_j)$ when computed on D that is located at one site.

$$R = C^D(I_j) \quad (3)$$

However, in a distributed environment, *D* is partitioned among different sites and local partitions cannot be moved around. The set *S* of shared attributes determines the fraction of the global database *D* that will be generated by aggregating the individual results of the local computations on $D_1, D_2, \ldots, D_n$. Thus, the operation C^D can be decomposed for a given *Shared* to produce equivalent results as follow:

$$R(I_j, Shared) = \sum_{i=1}^{n} \prod_i C^{D_i}(I_j, Shared) \quad (4)$$

$C^{Di}(I_j,Shared)$ is the local computation will be executed by using D_i and the shared attributes S among all sites. The subscript j specifies the condition that is composed of the attribute-value pairs of the itemset I_j. The local computations will be aggregated by the initiated agent at D_{init} using the operation $\sum$ on the cross product (Π operation) of the local numbers of tuples in each site that satisfy the specified condition j.

4.4 Algorithm Description

Given a transaction database, where each transaction is a set of items, an association rule is an expression X ⇒ Y, where X and Y are sets of items. Intuitively, it would mean that transactions in the database, which contains the items in X also contain the items in Y. Traditionally, most of the algorithms to determine association rules are confined to a single large database. The main issues have been the size of the data, how many passes that are to be made over the database and the resultant time taken.

In this research, we focused on finding association rules in a distributed database environment where moving local databases is not possible because of the security, privacy, and other issues. To discover the association rules in a given database [2][3], the algorithm involve iterating over the following 2 steps:

i) Find the itemsets at level L_k from the frequent itemsets (the sets of items that have minimum support) determined at level L_{k-1}.
ii) Compute the support and the confidence of the found itemsets and rules at level L_k.

Similarly, the decomposition of the proposed association rule algorithm can be divided into 2 major tasks: (1) Finding active itemsets and the candidate itemsets for the next level, and (2) Computing the *support* and *confidence*. Below, we show the decomposed the algorithm:

- An agent starts the association rule algorithm at a site.
- This agent computes the frequent itemsets and the candidate itemsets for the next level.
- Agents at different sites consult with each other to compute the support and confidence.

The support of an itemset is defined as the ratio of the number of transactions in which the itemset exists to the number of transactions in D. Thus, the common computational primitive for computing the support is to find the number of all transactions in D as represented by Equation 5.

The local computation $C^{Di}(Shared)$ is performed by an agent using D_i and the shared attributes *Shared* among all sites. The number of transaction in which an itemset exist is an extension of this primitive to count those transactions in D that contain the itemset (see Equation 5). Both of these counts are obtained by aggregating the results of local computations at the participating sites.

$$R(Shared) = \sum_{i=1}^{n} \prod_{i} C^{D_i}(Shared) \quad (5)$$

Below we present the algorithm for the common computational primitive for counting the number of transactions in D.

4.5 TransactionCount Algorithm

When a site needs to compute the number of transactions in D, it invokes its agent G_i to initiate the count process as shown in the *TransactionCount* algorithm. The *TransactionCount* algorithm counts the number of transactions in D by implicitly joining the local databases $D_1, D_2, \ldots, D_n$ at the participating sites without moving these D_is. The algorithm is also preserves the privacy of the data as all communications of data sets between agents of participating sites are either hashed or/and multiplied by some locally determined constant before being communicated. The objective of our algorithm is to perform the global computation in a distributed fashion, minimize communication between sites, hide actual values from other agents, and yet compute the actual results.

Algorithm: *TransactionCount*

Input: request to count number of transactions in D
Output: number of transactions in D

1. Agent G_i requests the attributes names from other agents representing the participating sites.
2. Each requested agent G_j, where j: 1-n, $j \neq i$, replies with a set of attributes $h_A(A_j)$ of its local database D_j. Attributes are hashed using the hash function $h_A()$ before being sent to Agent G_i.
3. Agent G_i determines the set of shared attributes S by computing the intersection of all sets of attributes, $A_1, A_2, \ldots, A_n$.
4. Agent G_i requests the distinct values of the attributes in S from the respective sites through their agents.
5. Each requested agent G_j does the following:
   a. determines the distinct values of the shared attributes in its local database $C^{Dj}(S)$,
   b. hashes them using the hash function $h_V()$, and
   c. sends the results to agent G_i.
6. Agent G_i computes the Cartesian product of the received distinct values and generates a relation called a *shared relation*.
7. Agent G_i initialize $R(Shared) = 0$, which will accumulate the sum of products result of Equation 4.
8. For each tuple t in the shared relation,
   a. Agent G_i requests the number of transactions that contain the values of t from the participating agents.
   b. each requested agent G_j does the following:
      i. Performs the count locally, $C^{Dj}(t, Shared)$.
      ii. Multiplies the count by a constant α_j that is decided locally to hide the actual count. All other agents do not know this constant.
      iii. Sends the resulting value to agent G_i.
   c. Agent G_i multiplies the incoming values from the participating sites and accumulates the results in $R(Shared)$.
9. To remove the multiplied constants from the resulting value $R(Shared)$, agent G_i sends the resulting value to its neighbor agent G_{i+1}, which divides the value by its determined constant α_{i+1} and in turn sends the resulting value to its neighbor agent G_{i+2}, and so in a pre-determined circular fashion till it comes back to the initiating agent G_i which divides the value by its determined constant α_i . The final value represents the actual number of transactions in D.

In the *TransactionCount* algorithm, a site D_i that needs to count the number of transaction in the global database D, initiate the counting process through its agent G_i, which will control the count process. Agent G_i builds the shared relation by collecting the local attributes $A_1, A_2, \ldots, A_n$ from the agents of the participating sites. To preserve the privacy of the data, the agents of the participating sites send the hashed values of their respective local database instead of the actual attribute values. The shared attributes are the intersection of the local attribute sets as in Equation 6.

$$S = \bigcap_{i=1}^{n} A_i \quad (6)$$

Then, the initiating agent G_i requests the distinct values of the shared attributes from the agents of the respective participating sites. Each of these requested agents queries its local database to find the distinct values of the shared attributes that exist locally. To preserve privacy of the local data, the requested agent hashes them using the hash function $h_V()$, and sends them to the initiating agent G_i.

From these collected distinct values, the initiating agent G_i builds the shared relation by performing the Cartesian product between the distinct values. For each tuple of the shared relation,

agent G_i requests the number of transactions, which contain the values of the tuple, from the agents of the participating sites. Each of these agents performs the operation $C^{Di}(t, Shared)$ locally to count the number of tuples in D_i that satisfy the condition *t*. The operation $C^{Di}(t, Shared)$ is translated locally into a simple SQL query "SELECT count(*) FROM D_i WHERE t", where *t* represents the attribute-value condition. Such decomposed operation is desirable due to its simplicity. After computing the count, it is multiplied by a constant α_j that is determined locally by the local agent to hide the actual value and maintain the data privacy. The result is then sent to the initiating agent G_i, which computes the product of all the collect results for the current tuple. The product values of all individual tuple in the shared relation are accumulated in *R(Shared)*.

The accumulated value in *R(Shared)* represents the number of transactions in *D* multiplied by all the constants determined at the participating sites. Therefore, to remove these constants, the initiating agent G_i sends the resulting value to its neighbor agent G_{i+1}, which divides the value by its determined constant α_{i+1} and in turn sends the resulting value to its neighbor agent G_{i+2}, and so. This value is sent from one agent to another in a pre-determined circular fasion that visits every site once till it comes back to the initiating agent G_i which divides the value by its determined constant α_i. The final value represents the actual number of transactions in *D*.

4.6 Computing Support of an Itemset

The support of an itemset is defined as the ratio of the number of transactions in which the itemset exists to the number of transactions in *D* that are represented by Equations 4 and 5 respectively. Therefore, the support *sup* (I_j) of an itemset is:

$$\sup(I_j) = \frac{R(I_j, Shared)}{R(Shared)} \quad (6)$$

The number of transaction in which an itemset exist is an extension of the primitive to count the number of transactions in *D* (see algorithm TransactionCount). Therefore, instead of counting all transactions in D, the algorithm only count those transactions that contain the itemset $(I_j.cond.)$. The subscript *j.cond.* specifies the condition that is composed of the attribute-value pairs of the itemset I_j.cond..

4.7 Complexity Analysis

When a computation is performed in a single computer, or a closely-coupled processor model, the algorithmic complexity is measured in terms of the CPU time and the required memory. On the other hand, in a distributed system where a global computation is performed cooperatively between the participating sites in the network, the total cost is measured in terms the communication cost since the computational is very small as compared to communication cost and thus the computational is neglected.

<u>*One summary per message (un-optimized):*</u> The number of messages is the sum of the following required messages:

- *(n-1)* messages to get the attributes of the participating sites.
- *(n-1)* messages to get the distinct values of the shared attributes.
- *[(n-1)k]* messages to get the number of transactions that contain the values of *shared tuple.* Where *k* is the number of shared tuples in the relation *Shared*. This size might be large depending on the number of different shared values at each site, however, it is possible to send one request with the whole shared relation to all agents of the participating sites. This reduces the exchanged number of messages to *n-1*, which is equal to the number of participating agents. The first approach requires more messages, however each message contains little information and thus it is more privacy aware for transmission over a network. The second approach requires fewer messages, however, each exchanged message includes more information and thus less privacy aware for transmission. The total number of messages exchanged will be *(n-1) (2+k).*

<u>*Exchanging all the summaries in one message (optimized):*</u>

Unlike above method, here the Shared tuple will be sent to all sites simultaneously and then the summaries are received in parallel. In this case the cost is decreased to *(n-1) (2+k)/n*

5. Discussions and Privacy Analysis

In the above, we proposed an algorithm for finding the association rules in a distributed database environment. In such environment, one would be interested in preserving the security and the privacy of the data. Our goal is to protect against eavesdroppers on the communication between the sites. We also would like to preserve the privacy of each site as much as possible.

5.1 Eavesdropping

Malicious attacker can listen to the communication between two different sites to obtain copy of the database. In this section, we show that anonymous listener cannot compromise the privacy of the transmitted information.

To calculate the support and confidence for the association rules, sites exchange information like attributes names, attribute values, and the number of tuples on site D_j that have the same attribute value. To preserve the confidentiality of the information the proposed algorithm uses secure hash function to avoid sending the attribute names on site D_j. In step 2 in the algorithm, when the Agent G_i requests the set of attributes, sites will not send the names of the attributes but rather sends a hash digest for each attribute. Each site D_j calculates the hash digest for the attribute a_j as follow:

Hash Digest = $h_A(K\ a_j)$

where *K* is a secret key that is shared and used by all the sites. This way the attacker or the listener cannot figure out the transmitted attribute names. At the same time, since all sites are using the same key *K* and hash function $h_A()$ the Agent G_i can still match similar hash digest to form the set of shared attributes (step 3 in the algorithm).

Anonymous listener cannot also figure out whether a specific site has a tuple with specific attribute value or not. According to Algorithm *TransctionCount*, each site sends the hash of the vector of distinct values of the common attributes. The use of secret key in the hash makes it impossible for the anonymous listener to know the actual values that are sent.

In step 8 in the algorithm, participating sites send counts of the transactions that contain the received item sets from agent G_i. Sites do not send actual values, but rather they multiply the count by a secret constant. Thus, anonymous attacker who listens to the channel cannot figure out the count that site *i* has for a given value. Note that multiplying the count values by secret constant will not affect the calculation of the support at the agent G_i because this secret value will be factored out in step 9 of the algorithm.

Thus in summary, anonymous listener will not be able to figure out the name of the variables, the distinct values of the shared attributes or the number of tuples in each participating site that correspond to each itemset in the shared relation.

5.2 Site Privacy

This section discuss how much privacy each site can preserve even though the different sites need to collaborate to answer quires. More specifically we will address the attribute name, attribute distinct values and the support for each value.

Recall that all sites are using the same secret key. Site *i* canfigure out whether site j has a specific attribute or not if site *i* has the same attribute that site j sent. By simply comparing the hash di-

gest that site j sent with the list of hash digests that site *i* has. Since the secure hash is a one way function, given a hash digest, site *i* cannot extract the name of the attribute even if the secret key K is known. If site *i* does not have a copy of the database schema, it cannot figure out the name of the attributes that other sites send to agent G_i. Site *i* can still figure out the name of the attribute sent by site j if it has a copy of the global database schema. In this case site *i* will have to calculate the secure hash digest for each attribute in the database schema and compare it with the unknown digest.
The same argument applies for the distinct values and count of itemsets. The participating sites send the hash of these values to agent G_i (step 5 and step 8 in the algorithm) rather than sending the values themselves. Even agent G_i will not be able to figure out the actual values or the counts of the itemsets.

6. Simulation Results

To evaluate the efficiency of the proposed algorithm in computing query results, we carried many simulation experiments. The tests were performed to find out the effect of various parameters on the final result. The three very important variables that affect the result are: the number of tuples in global database, the number of sites, and the average number of shared tuples between local databases. We have performed a number of tests to demonstrate that the queries can be computed in a distributed fashion without moving all the data from one site to another site. We used a network of workstations to test a number of databases residing on different sites. The algorithm was implemented using Java.

In this section, we refer to the proposed algorithm, **P**rivacy Preserving **D**ecomposable **M**ining **A**ssociation **R**ules on Distributed Data, as PDMAR. The first set of experiments was conducted to demonstrate the effect of the number of exchanged messages (as shown in Figure 2), and the effect of the elapsed time (as shown in Figure 3) as the number of local sites varies from 2-14. Fig. 2 demonstrates that the number of messages exchanged increases as the number of local sites increases. Also, Figure 3 shows that the elapsed time increases linearly as the number of local sites increases.

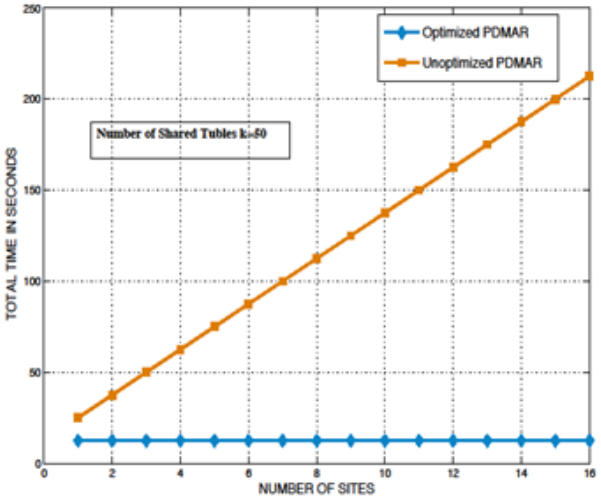


Fig. 2: Exchanged messages to run PDMAR on vertically distributed databases with varying the number of local sites.

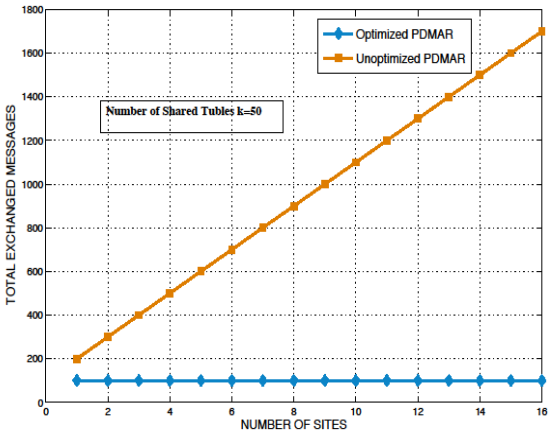


Fig. 3: Elapsed time to run PDMAR on vertically distributed databases with varying the number of local sites.

The second test was done to demonstrate effect on the elapsed time and the number of exchanged messages as the average number of shared tuples between local databases vary. The number of shared tuples varies as 5, 10, 15, 20, and 25. Figure 4 shows that the number of exchanged messages increases as the number of shared values increases.

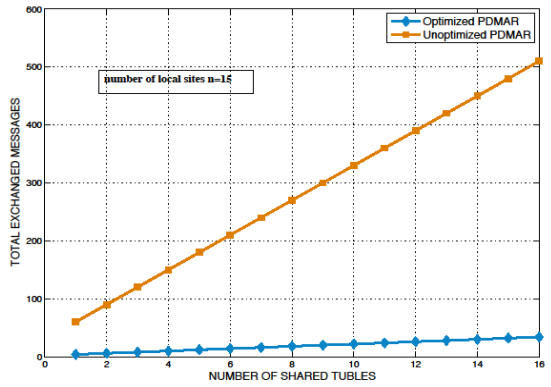


Fig. 4: Exchanged messages to run PDMAR on vertically distributed databases with varying the number of shared tuples.

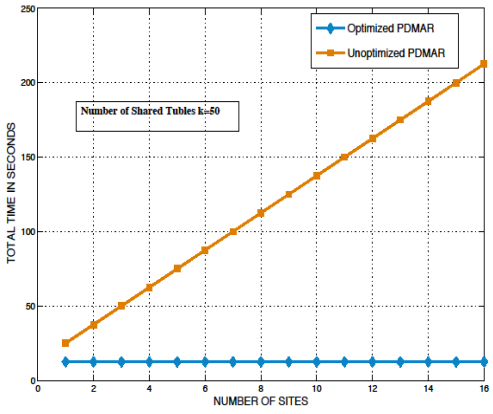


Fig. 5: Elapsed time to run PDMAR on vertically distributed databases with varying the number of shared tuples.

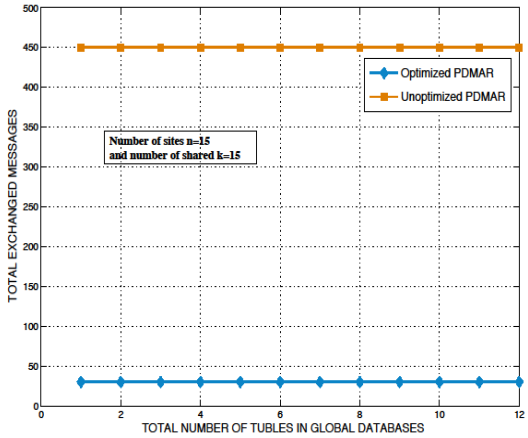


Fig. 6: *Exchanged messages to run PDMAR on vertically distributed databases with varying the number of tuples in global databases.*

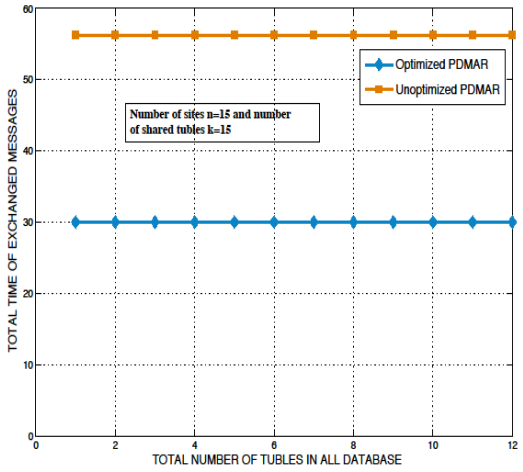


Fig. 7: Elapsed time to run PDMAR on vertically distributed databases with varying the number of tuples in global databases.

Figure 5 shows that the elapsed time to execute PDMAR increases as the number of shared values increases.

The last test was done to demonstrate the effect of the elapsed time and the number of exchanged messages as we vary the number of tuples in the database. Figures 6 and 7 show that the exchanged messages and the elapsed time to run PDMAR in an implicit database *D* is fixed this because the exchanged messages and time of exchanged messages are independent of number of tuples.

7. Conclusion

This paper proposed the integration of arbitrarily distributed data such as performing privacy preserving association rule mining. We employed agent to compute some tasks such as finding the frequent itemsets in implicit database formed by joining distributed local databases. Additionally, agent were shown to compute the support and confidence of discovered itemsets from the implicit database as well as determining the candidate itemsets for the next level from the frequent itemsets of the current level. These tasks are performed in a distributed fashio by the agents on arbitrary overlapped local databases. We have discussed the complexity of performing these computations in terms of messages that need to be exchanged among the sites for performing these computations. One very significant contribution of these results is that PDMAR can compute the association rules of geographically distributed databases. At the same time, the communication cost among the participating sites is minimized while preserving the privacy of the local databases.

References

[1] Atallah M. Bertino E., Elmagarmid A., Ibrahim M., and Verykios V., (1999), Disclosure limitation of sensitive rules," *Proceedings of the Workshop on Knowledge and Data Engineering Exchange*, Chicago, IL, November, pp.45-52.

[2] Agrawal T. Imielinski, and A. Swami, (1993),Mining association rules between sets of items in large databases, In SIGMOD 93 International Csonference on Management of Data, pp. 207-216.

[3] Agrawal, R. and R. Srikant, (1994), Fast algorithms for mining association rules," *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, September, pp.487-499,.

[4] Agrawal R. and Shafer J.C, (1996), Parallel mining of association rules," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp.929-969.

[5] Cheung D. W.-L., Han, J., Ng, V. T. Y., Fu A. W.-C. and Fu, Y. (1996), A fast distributed algorithm for mining association rules," *Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December, pp.21-42.

[6] Savasere, A., Omiecinski, E. and Navathe, S., (1995), An efficient algorithm for mining association rules in large databases," *Proceedings of the 21th International Conference on Very Large Data Bases*, Zurich, Switzerland, September, pp.432-444.

[7] Brickell J. and Shmatikov V., (2005), Privacy-preserving graph algorithm in the semi-honest model," in Roy, B. (Ed.): *Lecture Notes in Compute Science*, Vol. 3788, Springer-Verlag, pp.236-252.

[8] Canetti R., (2000), Security and composition of multiparty cryptographic protocols," *Journal of Cryptology* Vol. 13, No. 1, pp.143-202.

[9] Lindell, Y. and Pinkas, B., (2002), Privacy preserving data mining, *Journal of Cryptology*, Vol. 15, No. 3, pp.177-206.

[10] Dwork, C. and Nissim, K., (2004), "Privacy-preserving data mining on vertically partitioned databases," in Franklin, M.K. (Ed.): *Lecture Notes in Computer Science*, Vol. 3152, Springer-Verlag, pp.528-544.

[11] Kantarcioglu, M. and Clifton, C., (2004), "Privacy- preserving distributed mining of association rules on horizontally partition data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 9, pp.1026-1037.

[12] Veloso, A.A., Meira Jr., W., Parthasarathy, S. and de Carvalho, M.B., (2003), "Efficient, accurate and privacy- preserving data mining for frequent itemsets in distributed databases," *Proceedings of the Brazilian Symposium on Databases*, Manaus, Amazonas, Brazil, October, pp.281- 292.

[13] Chang, C.-C. and Lin, C.-Y. (2005), Perfect hashing schemes for mining association Rules," *The Computer Journal*, Vol. 48, No. 2, pp.168-179.

[14] Park, J.S., Chen, M.-S. and Yu, P.S., (1995), An effective hash-based algorithm for mining association rules," *Proceedings of the 1995 ACM-SIGMOD International Conference on Management of Data*, San Jose, CA, May, pp.175-186.

[15] Agarwal, R.C., Aggarwal,C.C. and Prasad,V., (2001), A tree projection algorithm for generation of frequent itemsets," *Journal of Parallel and Distributed Computing*, Vol. 61, No. 3, pp.350-371.

[16] Grahne, G. and Zhu, J., (2005), Fast algorithms for frequent itemset mining using FP-trees," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 10, pp.1347-1362.

[17] Han, J., Pei, J., Yin, Y. and R. Mao, (2004), Mining frequent pattern without candidate generation: a frequent pattern tree approach, *Data Mining and Knowledge Discovery*, Vol. 8, No. 1, pp.53-87.

[18] Li, Y.-C. and C-C. Chang, (2004), A new FP-tree algorithm for mining frequent itemsets," in Chi, C.-H. and Lam, K.-Y. (Eds.): *Lecture Notes in Computer Science*, Vol. 3309, Springer-V erlag, pp.266-277.

[19] Khedr A. M. and R. Bhatnagar, (2007), Agents for Integrating Distributed Data for Complex Computations, Computing and Informatics Journal , vol. 26, No.2, pp. 149-170.

[20] Khedr A. M., (2011), Nearest Neighbor Clustering over Partitioned Data, Computing and Informatics, Vol. 30, pp. 1001ñ1026.

[21] Khedr A. M. and A. Salim,(2008), Decomposable Algorithms for Finding the Nearest Pair}, J. Parallel Distrib. Comput. vol.68, pp. 902-912.

[22] Khedr A. M., Learning k-Classifier from Distributed Databases}, Computing and Informatics Journal, Vol. (27), pp. 355-376, 2008.
[23] Tamir Tassa, (2011), Secure Mining of Association Rules in Horizontally Distributed Databases, arXiv:1106.5113v1.
[24] Cheung D. W.-L., J. Han, V. Ng, A. W.-C. Fu, and Y. Fu, (1996), A fast distributed algorithm for mining association rules. In PDIS, page 3142.
[25] Kantarcioglu M. and C. Clifton, (2004), Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering,, 16(9):10261037.
[26] Chia M. H., D. E. Neiman, and V. R. Lesser, (1987), Poaching and distraction in asynchronous agent activities}, ICMAS, pp. 88-95.
[27] Durfee E. H., and V. R. Corkill, Coherent cooperation among communicating problem solvers, IEEE Transactions on Computers, 36(11), 1987.
[28] Huhns M. N., Singh M. P., and Ksiezyk T.,(1997) Global Information management via Local Autonomous Agents, Morgan Kaufmann Publishers.
[29] Khedr, A. M. Decomposable Algorithm for Computing k-Nearest Neighbors across PartitionedData, in: International Journal of Parallel, Emergent and Distributed Systems, vol. 31, no. 4, pp. 334-353, 2016.
[30] Khedr, A. M. and R. Bhatnagar, Agents for Integrating distributed databases for Complex Computations, Turk. J. Elec. Eng. & Comp. Sci., 14, pp. 313-327, 2006.
[31] Khedr, A. M. and R. Bhatnagar, New Algorithm for Clustering Distributed Data using k-means, Computing and Informatics, Vol. 33, pp. 1001-1022, 2014.
[32] Khedr, A. M. and R. Mahmoud, Agents for Integrating Distributed Data for Function Computations, Computing and Informatics, Vol. 31, pp. 1101-1125, 2012.
[33] Khedr, A. M. and Ahmed Salim, Decomposable Algorithms for Finding the Nearest Pair, J. Parallel Distrib. Comput., Vol. 68, pp. 902-912, 2008.