Privacy Preserving Association Rule Mining in Vertically Partitioned Data *

Jaideep Vaidya
Department of Computer Sciences
Purdue University
West Lafayette, Indiana
jsvaidya@cs.purdue.edu

Chris Clifton
Department of Computer Sciences
Purdue University
West Lafayette, Indiana
clifton@cs.purdue.edu

ABSTRACT

Privacy considerations often constrain data mining projects. This paper addresses the problem of association rule mining where transactions are distributed across sources. Each site holds some attributes of each transaction, and the sites wish to collaborate to identify globally valid association rules. However, the sites must not reveal individual transaction data. We present a two-party algorithm for efficiently discovering frequent itemsets with minimum support levels, without either site revealing individual transaction values.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.2.4 [**Database Management**]: Systems—*Distributed databases*; H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*

1. INTRODUCTION

Data mining technology has emerged as a means for identifying patterns and trends from large quantities of data. Mining encompasses various algorithms such as clustering, classification, association rule mining and sequence detection. Traditionally, all these algorithms have been developed within a centralized model, operating on the principle of gathering all data into a central site, then running algorithms against that data.

Many applications are not feasible using this methodology, leading to a need for distributed data mining. The obvious solution of a "virtual" data warehouse – heterogeneous access to all the data – is not always possible. The problem is not just that the data is distributed, but that it *must remain* distributed. There are several situations where this arises:

1. Connectivity. Transmitting large quantities of data to a central site may not be feasible.

2. Heterogeneity of sources. Is it easier to combine results than combine sources?

3. Privacy of sources. Organizations may be willing to share data mining results, but not data.

To address the above problems, the Distributed Data Mining (DDM) model has been developed wherein the data sources are assumed to be distributed across multiple sites. However, to date, algorithms in DDM have primarily been developed from the perspective of efficiency, not security.

This paper concentrates on the third issue: obtaining data mining results that are valid across a distributed data set, while limiting the sharing of data between sites. Specifically, we present a privacy preserving algorithm to mine association rules from vertically partitioned data. By *vertically partitioned*, we mean that each site contains some elements of a transaction. Using the traditional "market basket" example, one site may contain grocery purchases, while another has clothing purchases. Using a key such as credit card number and date, we can join these to identify relationships between purchases of clothing and groceries. However, this discloses the individual purchases at each site, possibly violating consumer privacy agreements.

There are more realistic examples. The transportation industry provides one. Manufacturers build many vehicles (aircraft, trucks, etc.). These are sold to fleet operators (airlines, trucking companies) that buy from multiple manufacturers. Manufacturers recommend specific maintenance schedules, loading factors, usage etc., but do not have control over how their product is used. The fleet operators do keep maintenance and operational data. Manufacturers have cross-fleet data on replacement parts and manufacturing differences. Combining these can reveal bad (or best) practices, product suitability for particular use, and other information that improves safety and reliability. Association rules, such as "short cut" engine removal techniques leading to stress cracks in engine mountings (an actual example from the airline industry), are known to be important. However, the fleet operators and manufacturers are justifiably concerned about revealing too much information, for both competitive and liability reasons. The ability to gen-

*Submitted to The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002).

erate association rules *without* revealing private data could lead to improved safety and reliability, benefiting all parties.

A similar situation exists in the sub-assembly manufacturing process, where different manufacturers provide components of the finished product. Cars incorporate several subcomponents; tires, electrical equipment etc.; made by independent producers. Again, we have proprietary data collected by several parties, with a single key joining all the data sets, where mining would help detect/predict malfunctions. The recent trouble between Ford Motor and Firestone Tire provide a real-life example. Ford Explorers with Firestone tires from a *specific* factory had tread separation problems in certain situations, resulting in 800 injuries. Since the tires did not have problems on other vehicles, and other tires on Ford Explorers did not pose a problem, neither side felt responsible. The delay in identifying the real problem led to a public relations nightmare and the eventual replacement of 14.1 million tires[17]. Many of these were probably fine – Ford Explorers accounted for only 6.5 million of the replaced tires [11]. Both manufacturers had their own data – early generation of association rules based on *all* of the data may have enabled Ford and Firestone to resolve the safety problem before it became a public relations nightmare.

Informally, the problem is to mine association rules across two heterogeneous data sets. One database is designated the *primary*, and is the initiator of the protocol. The other database is the *responder*. There is a *join key* present in both databases. The remaining attributes are present in one database or the other, but not both. The goal is to find association rules involving attributes other than the join key.

Although actual data may appear more complex, it can generally be flattened into the above model. The Ford/Firestone example could be viewed as having a unique key in the Ford database (VIN) corresponding to multiple keys in the Firestone database (each tire on a given vehicle). However, by treating each row as a vehicle, with the tire at each position on the vehicle corresponding to a different set of attributes, we can represent the data in the above model. Alternatively, each tire can be represented individually with vehicles repeated; the serial number of the tire serving as the join key.

Such modeling issues, as well as data cleansing and heterogeneity, exist independent of privacy concerns. We leave these issues to the data warehousing community.

In addition to stating our view of data and our mining goal, we must lay out the privacy constraints. Ideally we would achieve complete zero knowledge, but for a practical solution controlled information disclosure may be acceptable. Thus we foresee several possible security policies:

- True zero knowledge. Each party knows only what can be inferred from its own database and the global association rules,
- Partial knowledge of data values, related only to the association rule, or
- Partial results that do not disclose protected data by themselves, but when combined with outside information may lead to loss of security. In this case, exactly what externally obtained information leads to loss of security should be explicit.

Finally, we need to quantify the accuracy and the efficiency of the algorithm, in view of the security restrictions.

We first outline related work and the problem background. In Section 3, we formally define the problem and give the overall solution. Section 4 presents the component scalar product protocol, the key *privacy-preserving* part of the solution. We discuss what the method discloses in Section 5, and present extensions that further limit disclosure. Section 6 analyzes the security provided and communication costs of the algorithm. We conclude by summarizing the contributions of this paper and giving directions for future work.

2. BACKGROUND AND RELATED WORK

The centralized data mining model assumes that all the data required by any data mining algorithm is either available at or can be sent to a central site. A simple approach to data mining over multiple sources that will not share data is to run existing data mining tools at each site independently and combine the results[5, 6, 18]. However, this will often fail to give globally valid results. Issues that cause a disparity between local and global results include:

- Values for a single entity may be split across sources. Data mining at individual sites will be unable to detect cross-site correlations.
- The same item may be duplicated at different sites, and will be over-weighted in the results.
- Data at a single site is likely to be from a homogeneous population. Important geographic or demographic distinctions between that population and others cannot be seen on a single site.

Data mining algorithms that partition the data into subsets have been developed[20]. In particular, work in parallel data mining may be relevant [24, 15]. Parallel data mining algorithms may also serve as a starting point.

Algorithms have been proposed for distributed data mining. Cheung et al. proposed a method for horizontally partitioned data[8], and more recent work has addressed privacy in this model[14]. Distributed classification has also been addressed. A meta-learning approach has been developed that uses classifiers trained at different sites to develop a global classifier [5, 6, 18]. This *could* protect the individual entities, but it remains to be shown that the individual classifiers do not disclose private information. Recent work has addressed classification using Bayesian Networks in vertically partitioned data [7], and situations where the distribution is itself interesting with respect to what is learned [22]. However, none of this work addresses *privacy* concerns.

There has been research considering how much information can be inferred, calculated or revealed from the data made available through data mining algorithms, and how to minimize the leakage of information [16, 4]. However, this has been restricted to classification, and the problem has been

treated with an "all or nothing" approach. We desire quantification of the security of the process. Corporations may not require absolute zero knowledge protocols (that leak no information at all) as long as they can keep the information shared within strict (though possibly adjustable) bounds.

In [4], data perturbation techniques are used to protect individual privacy for classification, by adding random values from a normal/Gaussian distribution of mean 0 to the actual data values). One problem with this approach is a tradeoff between privacy and the accuracy of the results[1]. More recently, data perturbation has been applied to boolean association rules [19]. Here, the idea is to "flip" 0's and 1's so that reconstruction of the values for any individual transaction is difficult, but the rules learned on the distorted data approximately match the "true" rules. One interesting feature of this work is a flexible definition of privacy; e.g., the ability to correctly guess a value of '1' from the perturbed data can be considered a greater threat to privacy than correctly learning a '0'. [16] use cryptographic protocols to achieve complete zero knowledge leakage for the ID3 classification algorithm for two parties with horizontally partitioned data.

There has been work in cooperative computation between entities that mutually distrust one another. Secure two party computation was first investigated by Yao [23], and later generalized to multiparty computation. The seminal paper by Goldreich proves existence of a secure solution for *any* functionality[12]. The idea is as follows: the function F to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol to securely compute every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, means that the size of the protocol depends on the size of the circuit, which depends on the size of the input. This is inefficient for large inputs, as in data mining. In [9], relationships have been drawn between several problems in Data Mining and Secure Multiparty Computation. Although this shows that secure solutions exist, achieving *efficient* secure solutions for privacy preserving distributed data mining is still open.

3. PROBLEM DEFINITION

We consider the heterogeneous database scenario considered in [7], a vertical partitioning of the database between two parties A and B. The association rule mining problem can be formally stated as follows[2]: Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. Associated with each transaction is a unique identifier, called its *TID*. We say that a transaction T *contains* X, a set of some items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form, $X \Rightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \cap Y = \phi$. The rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* c if c% of transactions in $\mathcal{D}$ that contain X also contain Y. The rule $X \Rightarrow Y$ has *support* s in the transaction set $\mathcal{D}$ if s% of transactions in $\mathcal{D}$ contain $X \cup Y$.

Within this framework, we consider mining boolean association rules. The absence or presence of an attribute is represented as a 0 or 1. Transactions are strings of 0 and 1; the database can be represented as a matrix of {0,1}.

To find out if a particular itemset is frequent, we count the number of records where the values for all the attributes in the itemset are 1. This translates into a simple mathematical problem, given the following definitions:

Let the total number of attributes be $l + m$, where A has l attributes A_1 through A_l, and B has the remaining m attributes B_1 through B_m. Transactions/records are a sequence of $l + m$ 1s or 0s. Let k be the support threshold required, and n be the total number of transaction/records.

Let $\vec{X}$ and $\vec{Y}$ represent columns in the database, i.e., $x_i = 1$ iff row i has value 1 for attribute X. The scalar (or dot) product of two cardinality n vectors $\vec{X}$ and $\vec{Y}$ is defined as

$$\vec{X} \cdot \vec{Y} = \sum_{i=1}^{n} x_i * y_i$$

Determining if the two-itemset $\langle XY \rangle$ is frequent thus reduces to testing if $\vec{X} \cdot \vec{Y} \geq k$.

In Section 4 we present an efficient way to compute scalar product $\vec{X} \cdot \vec{Y}$ without either side disclosing its vector. First we will show how to generalize the above protocol from two-itemsets to general association rules without sharing information other than through scalar product computation.

The generalization of this protocol to a w-itemset is straightforward. Assume A has p attributes $a_1 \ldots a_p$ and B has q attributes $b_1 \ldots b_q$, and we want to compute the frequency of the $w = p + q$-itemset $\langle a_1, \ldots, a_p, b_1, \ldots, b_q \rangle$. Each item in $\vec{X}$ ($\vec{Y}$) is composed of the product of the corresponding individual elements, i.e., $x_i = \prod_{j=1}^{p} a_j$ and $y_i = \prod_{j=1}^{q} b_j$. This computes $\vec{X}$ and $\vec{Y}$ without sharing information between A and B. The scalar product protocol then securely computes the frequency of the entire w-itemset.

For example, suppose we want to compute if a particular 5-itemset is frequent, with A having 2 of the attributes, and B having the remaining 3 attributes. I.e., A and B want to know if the itemset $l = \langle A_a, A_b, B_a, B_b, B_c \rangle$ is frequent. A creates a new vector $\vec{X}$ of cardinality n where $\vec{X} = \vec{A_a} * \vec{A_b}$ (component multiplication) and B creates a new vector $\vec{Y}$ of cardinality n where $\vec{Y} = \vec{B_a} * \vec{B_b} * \vec{B_c}$. Now the scalar product of $\vec{X}$ and $\vec{Y}$ provides the (in)frequency of the itemset.

The complete algorithm to find frequent itemsets is:

1. L_1 = {large 1-itemsets}
2. for (k=2; $L_{k-1} \neq \phi$; k++) do begin
3. C_k = apriori-gen(L_{k-1});
4. for all candidates $c \in C_k$ do begin
5. if all the attributes in c are entirely at A or B
6. that party independently calculates $c.count$
7. else
8. let A have l of the attributes and B have the remaining m attributes
9. construct $\vec{X}$ on A's side and $\vec{Y}$ on B's side where $\vec{X} = \prod_{i=1}^{l} \vec{A_i}$ and $\vec{Y} = \prod_{i=1}^{m} \vec{B_i}$
10. compute $c.count = \vec{X} \cdot \vec{Y} = \sum_{i=1}^{n} x_i * y_i$
11. endif
12. $L_k = L_k \cup c | c.count \geq minsup$
13. end

14. end
15. Answer $= \cup_k L_k$

In step 3, the function apriori-gen takes the set of large itemsets L_{k-1} found in the $(k-1)th$ pass as an argument and generates the set of candidate itemsets C_k. This is done by generating a superset of possible candidate itemsets and pruning this set. [3] discusses the function in detail.

Given the counts and frequent itemsets, we can compute all association rules with $support \geq minsup$.

Only the steps 1, 3, 10 and 12 require sharing information. Since the final result $\cup_k L_k$ is known to both parties, steps 1, 3 and 12 reveal no extra information to either party. We now show how to compute step 10 without revealing information.

4. THE COMPONENT ALGORITHM

Secure computation of scalar product is the key to our protocol. Scalar product protocols have been proposed in the Secure Multiparty Computation literature[10], however these cryptographic solutions do not scale well to this data mining problem. We present an algebraic solution that masks the true values by placing them in equations masked with random values. The knowledge disclosed by these equations only allows computation of private values if one side learns a substantial number of the private values from an outside source. (A different algebraic technique has recently been proposed [13], however it requires at least twice the bitwise communication cost of the method presented here.)

We assume that n is even for simplifying the protocol. The protocol can easily be generalized to the case where n is odd as well. We first present a simple version of the protocol where if one side learns half of the other sides values, the equations exchanged by the protocol can be used to obtain the rest. In Section 4.2 we show the general case, allowing the two sides to trade off their relative security.

4.1 Protocol I

Protocol I consists of 3 steps. In step 1, A generates $n/2$ random numbers $R_1 \ldots R_{n/2}$. A uses $\vec{X}$, the randoms, and a matrix A of coefficients $a_{i,j}$ to compute a vector $\vec{X}'$ consisting of the following n values:

$$\begin{array}{l}\langle x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n/2} * R_{n/2} \rangle \\ \langle x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n/2} * R_{n/2} \rangle \\ \vdots \\ \langle x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n/2} * R_{n/2} \rangle\end{array}$$

The matrix A is known to both A and B. The only constraint on A is that the $a_{i,j}$ form coefficients for a set of n linear equations, out of which any $n/2$ are *independent*. Section 4.3 presents a method to compute such a matrix.

A sends all n values to B. B computes $S = \vec{X}' \cdot \vec{Y}$ by multiplying each value received with the corresponding y value and summing the results.

In step 2, B sends S and the following $n/2$ values to A:

$$\begin{array}{l}\langle a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n \rangle \\ \langle a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n \rangle \\ \vdots \\ \langle a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n \rangle\end{array}$$

Note that the coefficient matrix of the equations sent by B is just the transpose of the coefficient matrix of the equations sent by A. Since the equations are linearly independent, the set of equations sent by B will also be linearly independent.

S can be written as follows:

$$\begin{array}{l}S = \\ \quad y_1 * (x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n/2} * R_{n/2}) \\ \quad +y_2 * (x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n/2} * R_{n/2}) \\ \quad \vdots \\ \quad +y_n * (x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n/2} * R_{n/2})\end{array}$$

Simplifying the equation further, and grouping the $x_i * y_i$ terms, we get:

$$\begin{array}{l}S = \\ \quad (x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n) \\ \quad +(y_1 * a_{1,1} * R_1 + y_1 * a_{1,2} * R_2 + \\ \quad\quad \cdots + y_1 * a_{1,n/2} * R_{n/2}) \\ \quad +(y_2 * a_{2,1} * R_1 + y_2 * a_{2,2} * R_2 + \\ \quad\quad \cdots + y_2 * a_{2,n/2} * R_{n/2}) \\ \quad \vdots \\ \quad +(y_n * a_{n,1} * R_1 + y_n * a_{n,2} * R_2 + \\ \quad\quad \cdots + y_n * a_{n,n/2} * R_{n/2})\end{array}$$

The first line of the R.H.S. can be written as $\sum_{i=1}^{n} x_i * y_i$, the desired final result. In the remaining portion, we group the multiplicative components vertically and rearrange the equation to factor out all the R_i values, giving:

$$\begin{array}{l}S = \\ \quad \sum_{i=1}^{n} x_i * y_i \\ \quad +R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n) \\ \quad +R_2 * (a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n) \\ \quad \vdots \\ \quad +R_{n/2} * (a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n)\end{array}$$

A already knows the $n/2$ R_i values. The $n/2$ other values sent by B are the same as the coefficients on the R.H.S. of the above equation. A multiplies the $n/2$ values received from B with the corresponding R_i and subtracts the sum from S, giving the desired result.

In step 3, A reports the result to B.

Therefore A and B are able to cooperatively get the total number of transactions where both attributes are present. Since the number of equations is less than the number of unknowns, neither side can solve the equations to reveal details of any particular transaction. Section 6.1 covers this in more detail.

Protocol I does have a problem if run with different parties, e.g., to compute the frequency of A_1B_1 and A_1C_1. If B and C collude, they have more equations than unknowns. One solution is for A to use the same $a_{i,j}$ values and randoms R_i for every invocation on A_1.

4.2 Protocol II

We now generalize the protocol to allow trading risk of disclosure between A and B. Depending on the tradeoff chosen, this also allows A to run the protocol with multiple parties without having to reuse the randoms and coefficients.

Step 1: A generates n randoms $R_1 \ldots R_n$. From these, $\vec{X}$, and a matrix A forming coefficients for a set of linear independent equations, A generates the following vector $\vec{X}'$:

$$\begin{aligned}
&\langle x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n} * R_n \rangle \\
&\langle x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n} * R_n \rangle \\
&\vdots \\
&\langle x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n} * R_n \rangle
\end{aligned}$$

A sends all $\vec{X}'$ to B. This is step 1 of the protocol. B computes $S = \vec{X}' \cdot \vec{Y}$. B also calculates the following n values:

$$\begin{aligned}
&\langle a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n \rangle \\
&\langle a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n \rangle \\
&\vdots \\
&\langle a_{1,n} * y_1 + a_{2,n} * y_2 + \cdots + a_{n,n} * y_n \rangle
\end{aligned}$$

But B can't send these values, since A would then have n independent equations in n unknowns $(y_1 \ldots y_n)$, revealing the y values. Instead, B generates r random values, $R'_1 \ldots R'_r$. The number of values A would need to know to obtain full disclosure of B's values is governed by r.

B partitions the n values created earlier into r sets, and the R' values are used to hide the equations as follows:

$$\begin{aligned}
&\langle a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n + R'_1 \rangle \\
&\vdots \\
&\langle a_{1,n/r} * y_1 + a_{2,n/r} * y_2 + \cdots + a_{n,n/r} * y_n + R'_1 \rangle \\
&\langle a_{1,(n/r+1)} * y_1 + a_{2,(n/r+1)} * y_2 + \\
&\quad \cdots + a_{n,(n/r+1)} * y_n + R'_2 \rangle \\
&\vdots \\
&\langle a_{1,2n/r} * y_1 + a_{2,2n/r} * y_2 + \cdots + a_{n,2n/r} * y_n + R'_2 \rangle \\
&\vdots \\
&\langle a_{1,((r-1)n/r+1)} * y_1 + a_{2,((r-1)n/r+1)} * y_2 + \\
&\quad \cdots + a_{n,((r-1)n/r+1)} * y_n + R'_r \rangle \\
&\vdots \\
&\langle a_{1,n} * y_1 + a_{2,n} * y_2 + \cdots + a_{n,n} * y_n + R'_r \rangle
\end{aligned}$$

In step 2, B sends S and the n above values to A, who writes:

$$\begin{aligned}
S = \; & \\
& (x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n} * R_n) * y_1 \\
& + (x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n} * R_n) * y_2 \\
& \vdots \\
& + (x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n} * R_n) * y_n
\end{aligned}$$

Simplifying further and grouping the $x_i * y_i$ terms gives:

$$\begin{aligned}
S = \; & (x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n) \\
& + (y_1 * a_{1,1} * R_1 + y_1 * a_{1,2} * R_2 + \cdots + y_1 * a_{1,n} * R_n) \\
& + (y_2 * a_{2,1} * R_1 + y_2 * a_{2,2} * R_2 + \cdots + y_2 * a_{2,n} * R_n) \\
& \vdots \\
& + (y_n * a_{n,1} * R_1 + y_n * a_{n,2} * R_2 + \cdots + y_n * a_{n,n} * R_n)
\end{aligned}$$

The first line of the R.H.S. can be succinctly written as $\sum_{i=1}^{n} x_i * y_i$, the desired final result. In the remaining portion, we group all multiplicative components vertically, and rearrange the equation to factor out all the R_i values, giving:

$$\begin{aligned}
S = \; & \sum_{i=1}^{n} x_i * y_i \\
& + R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n) \\
& + R_2 * (a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n) \\
& \vdots \\
& + R_n * (a_{1,n} * y_1 + a_{2,n} * y_2 + \cdots + a_{n,n} * y_n)
\end{aligned}$$

Adding and subtracting the same quantity from one side of the equation does not change the equation in any way. Hence, the above equation can be rewritten as follows:

$$
\begin{aligned}
S = \quad & \sum_{i=1}^{n} x_i * y_i \\
& + \{R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n) \\
& \quad + R_1 * R'_1 - R_1 * R'_1\} \\
& \vdots \\
& + \{R_{n/r} * (a_{1,n/r} * y_{n/r} + a_{2,n/r} * y_2 + \cdots + a_{n,n/r} * y_n) \\
& \quad + R_{n/r} * R'_1 - R_{n/r} * R'_1\} \\
& + \{R_{n/r+1} * (a_{1,n/r+1} * y_{n/r+1} + a_{2,n/r+1} * y_2 + \\
& \qquad \cdots + a_{n,n/r+1} * y_n) \\
& \quad + R_{n/r+1} * R'_2 - R_{n/r+1} * R'_2\} \\
& \vdots \\
& + \{R_{2n/r} * (a_{1,2n/r} * y_{2n/r} + a_{2,2n/r} * y_2 + \\
& \qquad \cdots + a_{n,2n/r} * y_n) \\
& \quad + R_{2n/r} * R'_2 - R_{2n/r} * R'_2\} \\
& \vdots \\
& \vdots \\
& + \{R_{(r-1)n/r+1} * (a_{1,(r-1)n/r+1} * y_{(r-1)n/r+1} + \\
& \qquad a_{2,(r-1)n/r+1} * y_2 + \cdots + a_{n,(r-1)n/r+1} * y_n) \\
& \quad + R_{(r-1)n/r+1} * R'_r - R_{(r-1)n/r+1} * R'_r\} \\
& \vdots \\
& + \{R_n * (a_{1,n} * y_1 + a_{2,n} * y_2 + \cdots + a_{n,n} * y_n) \\
& \quad + R_n * R'_r - R_n * R'_r\}
\end{aligned}
$$

Now A factors out the R_i from the first two components and groups the rest vertically, giving:

$$
\begin{aligned}
S = \quad & \sum_{i=1}^{n} x_i * y_i \\
& + R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n + R'_1) \\
& \vdots \\
& + R_{n/r} * (a_{1,n/r} * y_{n/r} + a_{2,n/r} * y_2 + \\
& \qquad \cdots + a_{n,n/r} * y_n + R'_1) \\
& + R_{n/r+1} * (a_{1,n/r+1} * y_{n/r+1} + a_{2,n/r+1} * y_2 + \\
& \qquad \cdots + a_{n,n/r+1} * y_n + R'_2) \\
& \vdots \\
& + R_{2n/r} * (a_{1,2n/r} * y_{2n/r} + a_{2,2n/r} * y_2 + \\
& \qquad \cdots + a_{n,2n/r} * y_n + R'_2) \\
& \vdots \\
& + R_{(r-1)n/r+1} * (a_{1,(r-1)n/r+1} * y_{(r-1)n/r+1} + \\
& \qquad a_{2,(r-1)n/r+1} * y_2 + \cdots + a_{n,(r-1)n/r+1} * y_n + R'_r) \\
& \vdots \\
& + R_n * (a_{1,n} * y_1 + a_{2,n} * y_2 + \cdots + a_{n,n} * y_n + R'_r) \\
& - R_1 * R'_1 - \cdots - R_{n/r} * R'_1 \\
& - R_{n/r+1} * R'_2 - \cdots - R_{2n/r} * R'_2 \\
& \vdots \\
& - R_{(r-1)n/r+1} * R'_r - \cdots - R_n * R'_r
\end{aligned}
$$

A already knows the n R_i values. B also sent n other values, these are the coefficients of the n R_i values above.

A multiplies the n values received from B with the corresponding R_i and subtracts the sum from S to get:

$$
\begin{aligned}
Temp = \quad & \sum_{i=1}^{n} x_i * y_i \\
& - R_1 * R'_1 - \cdots - R_{n/r} * R'_1 \\
& - R_{n/r+1} * R'_2 - \cdots - R_{2n/r} * R'_2 \\
& \vdots \\
& - R_{(r-1)n/r+1} * R'_r - \cdots - R_n * R'_r
\end{aligned}
$$

Factoring out the R'_i gives:

$$
\begin{aligned}
Temp \quad = \quad & \sum_{i=1}^{n} x_i * y_i \\
& - (R_1 + R_2 + \cdots + R_{n/r}) * R'_1 \\
& - (R_{n/r+1} + R_{n/r+2} + \cdots + R_{2n/r}) * R'_2 \\
& \vdots \\
& - (R_{((r-1)n/r)+1} + R_{((r-1)n/r)+2} + \cdots + R_n) * R'_r
\end{aligned}
$$

To get the desired final result (viz. $\sum_{i=1}^{n} x_i * y_i$), A needs

to add the sum of the r multiplicative terms to Temp.

In step 3, A sends the r values to B, and B (knowing R') computes the final result. This gives B additional equations in A's unknowns, so that if B knows $n - r$ of A's items, it can compute the rest. Finally B replies with the result.

4.3 Selection of $a_{i,j}$

The above protocols require a matrix A of values that form coefficients of linear independent equations. The necessity of this is obvious from the fact that the equations are used to hide the data values. If any equation can be eliminated using less than half of the other equations, a linkage between less than $n/2$ of the unknowns is created. This is undesirable. In Protocol I, A needs to form n equations in $n/2$ unknowns. Obviously, only $n/2$ of these can be linearly independent while the other $n/2$ equations are linear combinations of the independent equations.

With high probability, a coefficient matrix generated using a pseudo-random function will form linearly independent equations. This enables construction of the $a_{i,j}$ matrix by sharing only a seed and a pseudo-random number generating function. However, the following method can be used to ensure linearly independent equations.

Note that the following conditions are equivalent:
Linearly independent equations $\Leftrightarrow$ Matrix is non-singular $\Leftrightarrow$ Matrix determinant non-zero $\Leftrightarrow$ Matrix is invertible

The value of the determinant of the transpose of a matrix is the same as the value of the determinant of the matrix. Hence, if the equations used by the first party are independent, the equations sent by the second party will also be independent, since the coefficient matrix for B is the transpose of the coefficient matrix for A.

One way of generating a coefficient matrix with equations linearly independent is as follows:

Create a matrix A by selecting the $\{a_{i,j}\}$ randomly. We now generate the matrix A' from A such that A' is nonsingular.

A matrix is diagonally dominant if the absolute value of each diagonal element is greater than the sum of the absolute values of the other elements in its row (or column). Any diagonally dominant matrix is nonsingular[21]. A' is calculated as follows:

For every row i, replace the diagonal element with the sum of the absolute values of elements of the corresponding row of the matrix A. The other elements are the same as the corresponding elements in A. I.e., A' is defined as $\{a'_{i,j}\}$ where:

$$a'_{i,j} = \begin{array}{ll} \sum_{l=1}^{k} |a_{i,l}| & , i = j \\ a_{i,j} & , i \neq j \end{array}$$

This forces the matrix A' to be diagonally dominant, since the diagonal elements are now larger than the sum of the absolute values of the other elements.

To summarize, the vector $\vec{X}$ is computed as follows:

Procedure generate_vector($\vec{X}$,S_1,S_2): $\vec{X}'$

1. Initialize random number generators RG_1, RG_2 with seeds S_1, S_2 respectively (possibly date+time)
2. Initialize the variable *curr_diag_elem* to 0
3. for (i=1; $i \leq n$; i++) do begin
4. $X'_i = X_i$
5. Initialize variable *sum* to 0
6. for (j=1; $j \leq n$; j++) do begin
7. Generate new coefficient $a_{i,j}$ from RG_1
8. $sum = sum + |a_{i,j}|$
9. Generate new random R_j from RG_2
10. if $i \neq j$ then $X'_i = X'_i + a_{i,j} * R_j$
11. else $curr\_diag\_elem = R_j$
12. end for
13. $X'_i = X'_i + curr\_diag\_elem * sum$
14. Reinitialize random number generator RG_2 with S_2
15. end for
16. Result=$\vec{X}'$

Although we describe the process in terms of an $n \times n$ matrix, the procedure only requires $O(n)$ space: The vector $\vec{X}'$, and the scalars *sum* and *curr_diag_elem*. Recomputing the matrix only requires the seed S_1, and recomputing the R_j requires only S_2.

For Protocol I, the coefficient matrix is order $n \times n/2$. We generate this in two phases. Choose arbitrary $l > 0$. Generate first $n/2$ rows using the above procedure, with randoms from the range $[ln/2, \infty]$. Then generate the remaining $n/2$ rows choosing randoms from the range $[1, l]$. This ensures that the diagonal elements are dominant over both the row and column.

Note that neither the $a_{i,j}$ values nor the R values need to be stored, even if we wish to keep the same values for multiple runs. By storing the seeds for the random number generator, we can generate the values whenever required. Hence, only two seeds have to be stored.

5. WHAT THIS METHOD DISCLOSES

The goal of this work is to create a practical, efficient method to compute association rules without disclosing entity values. This does *not* require a complete zero-knowledge solution. We will now discuss what is disclosed, and ways to limit unwanted potential disclosure.

5.1 What Must be Disclosed

The nature of the problem – each party knows its own data, and learns the resulting global association rules – results in some disclosure. For example, if we have a support threshold of 5%, a rule $A_1 \rightarrow B_1$ holds, and exactly 5% of the items on A have item A_1, A knows that at least those same items on B have property B_1. It is acceptable for the protocol to disclose knowledge that could be obtained from the global rules and one's own database.

This method discloses more than just the presence or absence of a rule with support above a threshold. Side A learns the exact support for an itemset. This increases the probability that A will learn that a set of items on B have a given property. This occurs when the global support equals A's support, not just when A's support is at the threshold. In any case, A learns the probability that an item in the set

supported by A has a property in B, computed as the ratio of the actual support to A's support.

It is unlikely that specific individual data values will be disclosed with certainty by this method. This possibility can be reduced further by allowing approximate answers, as described in 5.3.

First we look at a more troubling issue: A difficulty posed by boolean attributes.

5.2 The Trouble with $\{0, 1\}$

When we mine boolean association rules, the input values (x_i and y_i values) are restricted to 0 or 1. This creates a disclosure risk in other scalar product protocols [10, 13]. Input restriction causes serious problems with security in our protocols as well. In Protocol I, restricting the x_i values to 0 or 1 does not decrease A's security, as the random values overshadow the x_i values.

Unfortunately, the same cannot be said for B. Recall that B provides $n/2$ equations in the n unknowns (y_i). If the equations have a unique solution, A could try all possible values of 0 and 1 for all the y_i to obtain the correct solution. This 2^n brute force approach enables A to *know* the correct values for the y_i. The matrix generation procedure in Section 4.3 makes the problem even worse – a 0 or 1 in the diagonal entry contributes more to the sum than all of the remaining entries, allowing us to partition equations into those with a 0 on the diagonal and those with a 1. (For Protocol II, the partition is only within equations masked by the same R'_i.)

To solve this, B masks each of the n equations sent in step 2 with a new random R''_i. A is now unable to determine if a guess is correct. However, this gives A an additional "leftover" component in the sum of $R_i \cdot R''_i$. A needs to subtract $\sum_{i=1}^{n} R_i * R''_i$. This is simply a secure dot product. Protocol I can be used for this (or Protocol II, if repeated executions of the protocol are expected.) Since the values are arbitrary real numbers, the boolean problem vanishes.

For Protocol I, B sends back:

$$\begin{aligned}
&\langle a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n + R''_1 \rangle \\
&\langle a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n + R''_2 \rangle \\
&\vdots \\
&\langle a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n + R''_{n/2} \rangle
\end{aligned}$$

Recollect that S can be written as:

$$\begin{aligned}
S = &\sum_{i=1}^{n} x_i * y_i \\
&+ R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n) \\
&+ R_2 * (a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n) \\
&\vdots \\
&+ R_{n/2} * (a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n)
\end{aligned}$$

Rewriting the equation:

$$\begin{aligned}
S = &\sum_{i=1}^{n} x_i * y_i \\
&+ R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n + R''_1) \\
&+ R_2 * (a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n + R''_2) \\
&\vdots \\
&+ R_{n/2} * (a_{1,n/2} * y_1 + \cdots + a_{n,n/2} * y_n + R''_{n/2}) \\
&- \sum_{i=1}^{n/2} R_i * R''_i
\end{aligned}$$

The middle rows are the values B sent back. All that remains is to securely compute $\sum_{i=1}^{n/2} R_i * R''_i$ using the original Protocol I. An analogous construction works for Protocol II.

5.3 Handling an Inquisitive Cheater

We have presumed a semi-honest model: Each party behaves correctly during execution of the protocol, but may use information learned to discover private values. If a party "cheats" by changing its input, privacy can be compromised. Suppose A is dishonest, and wants to know the value of attribute B_1 for item x. If A runs the protocol, but submits x as the only item for which A_1 holds, the support for A_1B_1 (0 or 1) tells if B_1 holds for x.

Using Secure Multiparty Computation for a single comparison, Yao's Millionaires' problem[23], prevents either side from learning the exact support. In step 3 of Protocol II, A sends the r sums of randoms, but not $Temp$. A and B then securely compare $Temp \geq Threshold + \sum_{i=1}^{r} r_i * R'_i$. The dishonest party is unable to distinguish between 0 and 1.

This still isn't sufficient, as A could substitute $Threshold$ for item x. This attack works with *any* protocol using scalar product to compute if support exceeds a threshold, no matter how secure. The good news is, bounding the $a_{i,j}$ and R_i values allows this attempt to be detected.

Referring to Protocol II, in step 1 A sends x_k masked by addition of $\sum_{i=1}^{n} a_{k,i} * R_i$. If the $a_{i,j}$ and R_i are bounded so the sum must be in the range $[0, Threshold - 1]$, an x_k set to $Threshold$ instead of 0 or 1 can be detected. If B sets y_k to $Threshold$, then the k^{th} value returned to A contains $a_{k,k} * y_k$. The construction of the coefficient matrix means that the value of row $k \leq \sum_{i=1}^{n} a_{i,k} * 1 + R' \leq 2 * a_{k,k} + R'$. If the R' are in the range $[0, a_{k,k}(Threshold - 2)]$, the offending y_k will exceed the maximum, and A can detect B's deception.

If B tries to hide y_k with a negative R', the other rows with the same R' will be too small. B can detect if A tries a negative R_k through bounds on the step 1 values, calculated with the values sent in step 3 and the known $a_{i,j}$. The bounds on the random values enable A and B to improve their probability of guessing the other's values, but the improvement is insignificant (50% correct vs. roughly 50%+1/$Threshold$).

It remains to be determined if falsifying the $a_{i,j}$ values would

enable A or B to hide such an attack. Any deviation from the agreed on coefficients will distort the result, so the dishonest party is unlikely to learn anything from the result.

5.4 Presense or Absence of an Item

This method requires a globally agreed upon key to identify items across databases. This enables side B to learn the items that are present in A's database. In the first step, B is given a set of (randomized) values, identified by the keys of every item present in A's database.

To avoid this problem, we can allow A to send fictitious entries. These are legal keys, but with all values set to 0 before adding the random masking values. The fictitious entries will not affect the support of any rule, but since B is unable to learn the true values for any item, B will not know which items are fictitious and which are actually present in A's data. This does increase the communication cost, but the tradeoff between privacy and communication cost is known only to A – so even the knowledge that this *can* be done prevents B from knowing what items are in A's database with any certainty.

This still gives B a *superset* of the items present in A's database. We can partly address this by allowing A to drop entities that do not support the itemset being computed. This limits B to knowing a superset of the items in A's database that support A's portion of the itemset. However, there is no guarantee that any particular item in this set is in A's database – only that every item that does support the rule is included. Again, by allowing bounded approximations (A dropping a limited number of items that do support the rule), we can reduce even this to a guess.

5.5 Disclosure Summary

In the worst case, learning global association rules, along with knowledge of one's own data, *can* reveal the other party's attribute values for some entities. The method presented reveals no more than this worst case. However, due to revealing the actual support and not just if support exceeds a threshold, in some cases we reveal extra information. This disclosure is generally probabilistic – we can estimate the probability that a given item has a specific value. The actual values for a specific item aren't revealed unless knowledge of the global rule discloses them.

6. SECURITY / COMMUNICATION ANALYSIS

We now provide a security and communication analyses of protocols I and II.

6.1 Security Analysis

For a complete security analysis of the process, we must first analyze the security of the component scalar product protocol, and then analyze the security of the entire association rule mining algorithm.

The security of the scalar product protocol is based on the inability of either side to solve k equations in more than k unknowns. Some of the unknowns are randomly chosen, and can safely be assumed as private. However, what if some of the actual data values are known to the other party? This reduces the number of unknowns. When the number of unknowns is less than the number of equations, the equations can be solved and all data values are revealed.

Table 1: Security Analysis of Protocols

	Protected values	*Number of randoms generated*	*Total number of unknowns*	*Number of equations revealed*
Protocol I				
A	$x_1 \cdots x_n$	$n/2$	$3n/2$	$n+1$
	boolean:	$+n/4$	$+n/4$	$+n/2$
B	$y_1 \cdots y_n$	0	n	$n/2$
	boolean:	$+n/2$	$+n/2$	$+n/2$
Protocol II				
A	$x_1 \cdots x_n$	n	$2n$	$n+r$
	boolean:	$+n$	$+n$	$+n+r'$
B	$y_1 \cdots y_n$	r	$n+r$	n
	boolean:	$+n+r'$	$+n+r'$	$+n$

Therefore, the disclosure risk in this method is based on the number of data values that the other party might know from some external source. Table 1 presents the number of unknowns and equations generated for protocols I and II. Two rows are given for each protocol/party: One for the basic protocol, and one for the additional step needed to protect boolean values as described in Section 5.2. This shows the number of data values the other party must have knowledge of to obtain full disclosure.

The security of the component protocol hinges on having fewer equations than unknowns. The scalar product protocol is used once for every candidate item set. Possible problems with this are:

- Multiple w-itemsets in the candidate set may be split as 1, $w-1$ on each side. Consider two possible candidate sets A_1, B_1, B_2, B_5 and A_1, B_2, B_3. If A uses new/different equations for each candidate set, it imperils the security of A_1. However, B can reuse the values sent the first time. The equations sent by B can be reused for the same combinations of B_i, only a new sum must be sent. This reveals an additional equation, limiting the number of times B can run the protocol. In Protocol II, this limit is adjusted by r, and is unlikely to be a reached if the number of entities is high relative to the number of attributes. This also lowers communication cost, as every itemset from each side is sent at most once.
- Combining itemsets on one side and providing equations in the combined itemset. There is no question of linear equations in this case, so it does not perceptibly weaken the privacy.

6.2 Communication Analysis

For a complete communication analysis of the process, we first analyze the communication cost of the component scalar product protocol, then analyze the communication cost of the entire association rule mining algorithm.

In Protocol I, A sends one message with n values. B replies with $n/2+1$ values. A then sends the result. This gives three

Table 2: Communication Cost

	Rounds	*Bitwise cost*	
Protocol I	3	$1.5 * n * MaxValSz^*$	$O(n)$
boolean:	+2	$+.75 * n * MaxValSz$	
Protocol II	4	$2 * n * MaxValSz$	$O(n)$
boolean:	+3	$+2 * n * MaxValSz$	

*MaxValSz = Maximum bits to represent any input value

communication rounds. The bitwise communication cost is O(n) with the constant being 3/2. The masking process for boolean data adds two rounds, with messages half the size of the first two rounds.

In Protocol II, A sends one message with n values. B replies with a message consisting of $n + 1$ values. A then sends a message consisting of r values. Finally B sends the result, for a total of four communication rounds. The bitwise communication cost is O(n) with constant approximately 2 (assuming r is constant).Masking boolean data adds three rounds, and doubles the bitwise cost. The comparative communication cost of both protocols is presented in table 2.

Both protocols face the additional cost of communicating the $a_{i,j}$ values, which is quadratic. However, this cost can made constant by agreeing on a function and a seed value to generate the $a_{i,j}$ values.

The communication analysis for the entire association rule mining algorithm follows: There is no communication cost for any itemset wholly contained on either side. For every itemset split between the 2 parties, we engage in the scalar product protocol once. As noted earlier, values that have already been sent can be reused. The entire algorithm extends the apriori algorithm. Essentially we provide a means of determining if a candidate itemset is frequent. The communication cost can be expressed in terms of the i/o cost of the apriori algorithm, in fact a constant multiple of the i/o cost of the apriori algorithm.

7. CONCLUSION AND FUTURE WORK

The major contributions of this paper are a privacy preserving association rule mining algorithm given a privacy preserving scalar product protocol, and an efficient protocol for computing scalar product while preserving privacy of the individual values. We show that it is possible to achieve good individual security with reasonable communication cost.

There are several directions for future research. First is handling multiple parties. This algorithm is limited to two parties. Extending it to multiple parties is non-trivial, especially if we consider collusion between parties. A second area for future work is quantitative association rule mining and non-categorical attributes. This work is limited to boolean association rule mining. Non-categorical attributes significantly increase the complexity of the problem.

The same privacy issues face other types of data mining, such as Clustering, Classification, and Sequence Detection. Together with different data partitioning strategies (horizontal and vertical, or an interleaving of both), there is much scope for future research. The challenge is to formulate problems addressing interesting applications, and develop solutions to those problems. Our grand goal is to develop methods enabling *any* data mining that can be done at a single site to be done across various sources, while respecting the privacy policies of those sources.

8. ACKNOWLEDGMENTS

The authors thank Prof. John Rice for help in formulating an efficient way to generate the coefficient matrix, and Murat Kantarcioglu for suggestions on addressing boolean attributes. Portions of this work were supported by Grant EIA-9903545 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security.

9. REFERENCES

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Santa Barbara, California, USA, May 21-23 2001. ACM.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., May 26–28 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, Sept. 12-15 1994. VLDB.

[4] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, Dallas, TX, May 14-19 2000. ACM.

[5] P. Chan. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, 1996. (Technical Report CUCS-044-96).

[6] P. Chan. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems*, 8:5–28, 1997.

[7] R. Chen, K. Sivakumar, and H. Kargupta. Distributed web mining using bayesian networks from multiple data streams. In *The 2001 IEEE International Conference on Data Mining*. IEEE, Nov. 29 - Dec. 2 2001.

[8] D. W.-L. Cheung, V. Ng, A. W.-C. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *Transactions on Knowledge and Data Engineering*, 8(6):911–922, Dec. 1996.

[9] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 New Security Paradigms Workshop*, Cloudcroft, New Mexico, Sept. 11-13 2001.

[10] W. Du and M. J. Atallah. Secure multi-party computational geometry. In *Proceedings of the Seventh International Workshop on Algorithms and Data Structures*, Providence, Rhode Island, Aug. 8-10 2001.

[11] Ford Motor Corporation. Corporate citizenship report. http://www.ford.com/en/ourCompany/communityAndCulture/buildingRelationships/strategicIssues/firestoneTireRecall.htm, May 2001.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[13] I. Ioannidis, A. Grama, and M. Atallah. A secure protocol for computing dot products in clustered and distributed environments. In *Submitted to the International Conference on Parallel Processing*, Vancouver, Canada, Aug. 18-21 2002.

[14] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, June 2 2002.

[15] H. Kargupta and P. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT Press, 2000.

[16] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, Aug. 20-24 2000.

[17] National Highway Traffic Safety Administration. Firestone tire recall. http://www.nhtsa.dot.gov/hot/Firestone/Index.html, May 2001.

[18] A. Prodromidis, P. Chan, and S. Stolfo. *Meta-learning in distributed data mining systems: Issues and approaches*, chapter 3. AAAI/MIT Press, 2000.

[19] S. J. Rizvi and J. R. Haritsa. Privacy-preserving association rule mining. In *Proceedings of 28th International Conference on Very Large Data Bases*. VLDB, Aug. 20-23 2002.

[20] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of 21th International Conference on Very Large Data Bases*, pages 432–444. VLDB, Sept. 11-15 1995.

[21] G. Strang. *Linear Algebra And Its Applications*, chapter 7, pages 387,492. Harcourt Brace Jovanovich, Publishers, San Diego, CA, USA, 1988.

[22] R. Wirth, M. Borth, and J. Hipp. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Ubiquitous Data Mining for Mobile and Distributed Environments workshop associated with the Joint 12th European Conference on Machine Learning (ECML'01) and 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, Freiburg, Germany, Sept.3-7 2001.

[23] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

[24] M. J. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency, special issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, Dec. 1999.