Maintaining Data Privacy in Association Rule Mining

Shariq J. Rizvi

Computer Science & Engineering
Indian Institute of Technology
Mumbai 400076, INDIA
rizvi@cse.iitb.ac.in

Jayant R. Haritsa*

Database Systems Lab, SERC
Indian Institute of Science
Bangalore 560012, INDIA
haritsa@dsl.serc.iisc.ernet.in

Abstract

Data mining services require accurate input data for their results to be meaningful, but privacy concerns may influence users to provide spurious information. We investigate here, with respect to mining association rules, whether users can be encouraged to provide correct information by ensuring that the mining process cannot, with any reasonable degree of certainty, violate their privacy. We present a scheme, based on probabilistic distortion of user data, that can simultaneously provide a high degree of privacy to the user and retain a high level of accuracy in the mining results. The performance of the scheme is validated against representative real and synthetic datasets.

1 Introduction

The knowledge models produced through data mining techniques are only as good as the accuracy of their input data. One source of data inaccuracy is when users deliberately provide wrong information. This is especially common with regard to customers who are asked to provide personal information on Web forms to e-commerce service providers. The compulsion for doing so may be the (perhaps well-founded) worry that the requested information may be misused by the service provider to harass the customer. As a case in point, consider a pharmaceutical company that asks clients to disclose the diseases they have suffered from in order to investigate the correlations in their occurrences – for example, "Adult females with malarial infections are also prone to contract tuberculosis". While the company may be acquiring the data solely for genuine data mining purposes that would eventually reflect itself in better service to the client, at the same time the client might worry that if her medical records are either inadvertently or deliberately disclosed, it may adversely affect her employment opportunities.

We investigate, in this paper, whether customers can be encouraged to provide correct information by ensuring that the mining process cannot, with any reasonable degree of certainty, violate their privacy. At the same time, we would like the mining process to be as accurate as possible in terms of its results. The difficulty lies in the fact that these two metrics: *privacy* and *accuracy*, are typically contradictory in nature, with the consequence that improving one usually incurs a cost in the other [1]. Therefore, we comprise on the ideal and perhaps infeasible goal of having both complete privacy and complete accuracy through *approximate* solutions that provide practically acceptable values for these metrics. Note further that since the purpose of data mining is essentially to identify statistical *trends*, cent-per-cent accuracy in the mining results is perhaps often not even a required feature.

Our study is carried out in the context of extracting *association rules* from large historical databases, a popular mining process [2] that identifies interesting correlations between database attributes, such as the one described in the pharmaceutical example. For this framework, we present a scheme called **MASK** (Mining Associations with Secrecy Konstraints), that attempts to simultaneously provide a high degree of privacy to the user and retain a high degree of accuracy in the mining results. Our scheme is based on a simple probabilistic distortion of user data, employing random numbers generated from a pre-defined distribution function. It is this distorted information that is eventually supplied to the data miner, along with a description of the distortion procedure. We define a privacy metric and present an analytical formula for evaluating the privacy obtained under this metric by the distortion approach.

*Contact Author

Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

A special feature of our scheme is that the distortion process can be easily implemented at the data source itself, that is, at the *user machine*. This increases the confidence of the user in providing accurate information since she does not have to trust a third-party to carry out the distortion process before the data is acquired by the service provider. Note that some of the other privacy techniques suggested in the literature, such as swapping values between records [7], do not support this feature since they require the entire database to be available for their functioning.

As described in detail later in the paper, mining the distorted database can be, apart from being error-prone, significantly *more expensive* in terms of both time and space as compared to mining the true database. We present a variety of optimizations to address these issues.

Finally, the performance of MASK's mining scheme is validated against representative real and synthetic datasets, with respect to both privacy and accuracy. Our results indicate that there are regions of the distortion parameter space that are conducive to satisfactorily meeting the dual objectives.

1.1 Organization

The remainder of this paper is organized as follows: In Section 2, we describe the privacy framework employed in our study and in Section 3, we quantify the privacy attained under this framework by our distortion method. Then, in Section 4, we present our new MASK algorithm for mining the distorted database. Optimizations to improve the space and time complexity of MASK are described in Section 5. The performance model and the experimental results are highlighted in Sections 6 and 7, respectively. Bounds on the reconstruction errors incurred during the mining process are given in Section 8. Related work on privacy-preserving mining is reviewed in Section 9. Finally, in Section 10, we summarize the conclusions of our study and outline future avenues to explore.

2 Problem Framework

In this section, we describe the framework of the privacy mining problem that we consider here.

2.1 Database Model

We assume that each customer contributes a tuple to the database with the tuple being a fixed-length sequence of 1's and 0's. A typical example of such a database is the so-called "market-basket" database [2] wherein the columns represent the items sold by a supermarket, and each row describes, through a sequence of 1's and 0's, the purchases made by a particular customer (1 indicates a purchase and 0 indicates no purchase). We also assume that the overall number of 1's in the database is significantly smaller than the number of 0's – this is especially true for market-baskets since each customer typically buys only a small fraction of all the items available in the store. In short, the database is modeled as a *large disk-resident two-dimensional sparse boolean matrix*.

Note that the boolean representation is only logical and that the database tuples may actually be physically stored as "item-lists", that is, as an ordered list of the identifiers of the items purchased in the transaction. The list representation may appear preferable for the sparse databases that we are considering, since it reduces the space requirement as compared to storing entire bit-vectors. However, because of the fact that we are *distorting* user information, it may be the case that the distorted matrix will not be as sparse as the true database. Therefore, in this paper, we assume that the distorted database is stored as a large collection of bit-vectors.

2.2 Mining Objectives

The goal of the miner is to compute *association rules* on the above database. Denoting the set of transactions in the database by $\mathcal{T}$ and the set of items in the database by $\mathcal{I}$, an association rule is a (statistical) implication of the form $\mathcal{X} \Longrightarrow \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y} \subset \mathcal{I}$ and $\mathcal{X} \cap Y = \phi$. A rule $\mathcal{X} \Longrightarrow \mathcal{Y}$ is said to have a *support* (or frequency) factor s iff at least $s\%$ of the transactions in $\mathcal{T}$ satisfy $\mathcal{X} \cup \mathcal{Y}$. A rule $\mathcal{X} \Longrightarrow \mathcal{Y}$ is satisfied in the set of transactions $\mathcal{T}$ with a *confidence* factor c iff at least $c\%$ of the transactions in $\mathcal{T}$ that satisfy $\mathcal{X}$ also satisfy $\mathcal{Y}$. and confidence are fractions in the interval [0,1]. The support is a measure of statistical significance, whereas confidence is a measure of the strength of the rule.

A rule is said to be "interesting" if its support and confidence are greater than user-defined thresholds sup_{min} and con_{min}, respectively, and the objective of the mining process is to find all such interesting rules. It has been shown in [2] that achieving this goal is effectively equivalent to generating all subsets $\mathcal{X}$ of $\mathcal{I}$ that have support greater than sup_{min} – these subsets are called *frequent* itemsets. Therefore, the mining objective is, in essence, to efficiently discover all frequent itemsets that are present in the database.

2.3 Privacy Metric

As mentioned earlier, the mechanism adopted in this paper for achieving privacy is to *distort* the user data before it is subject to the mining process. Accordingly, we measure privacy with regard to the probability with which the user's distorted entries can be *reconstructed*. While privacy could be measured at the granularity of entire tuples, we consider here the *stronger* requirement of ensuring privacy at the level of *individual* entries in each customer tuple. In short, our privacy

metric is: "With what probability can a given 1 or 0 in the true matrix be reconstructed"?

A related issue here is whether the user would want the *same* level of privacy for both 1's and 0's? For many applications, such as the market-basket database, it appears reasonable to expect that customers would want more privacy for their 1's than for their 0's, since the 1's denote specific actions whereas the 0's are the default options.

3 Quantifying MASK's Privacy

In this section, we present the distortion procedure used by the MASK scheme and quantify the privacy provided by the procedure, as per the above metric.

3.1 Distortion Procedure

A customer tuple can be considered to be a random vector $\mathbf{X} = \{\mathbf{X_i}\}$, such that $X_i = 0$ or 1. We generate the distorted vector from this customer tuple by computing $\mathbf{Y} = \mathbf{distort}(\mathbf{X})$ where $\mathbf{Y_i} = \mathbf{X_i}$ **XOR** $\mathbf{\overline{r}_i}$ and $\overline{r}_i$ is the complement of r_i, a random variable with density function $\mathbf{f(r)} = \mathbf{bernoulli(p)}$ $(\mathbf{0 \leq p \leq 1})$. That is, r_i takes a value **1** with probability p and **0** with probability $1 - p$.

The net effect of the above computation is that the identity of the i^{th} element in X is kept the same with probability p and is *flipped* with probability $(1 - p)$. All the customer tuples are distorted in this fashion and make up the database supplied to the miner – in effect, the miner receives a *probabilistic function* of the true customer database.

Note that, in principle, it is possible to use different settings of p for distorting different items. That is, to have a *vector* of p settings ranging across the columns of the database. For simplicity, we will assume here that a single p is used for all the items – this choice also has useful implementation implications, as described later in Section 5.

We now move on to quantifying the privacy obtained by the above distortion procedure. In the following analysis we first consider the extreme case, where the user wants to have maximum privacy for 1's but is completely unconcerned about the 0's. After that, we derive the general privacy equations where the customer, though more conservative about 1's, requires a degree of privacy for the 0's too.

A caveat: Our privacy estimates do not take into account the fact that there may be a reduction in privacy, as pointed out recently in [1, 8], when the mining output (i.e., the association rules) is used to re-interrogate the distorted database – we plan to investigate this issue in our future work.

3.2 Reconstruction Probability of a 1

Let s_i be the true support of item i, normalized to the number of tuples in the database. This means that the probability that a random customer $\mathcal{C}$ bought this i^{th} item is s_i. We now have to evaluate the probability that given that $\mathcal{C}$ indeed did buy item i, her original '1' can be reconstructed from the distorted entry. Denoting the original entry as X_i and the distorted entry as Y_i, the probability of correct reconstruction is given by:

$\mathcal{R}_1(p, s_i) =$

$$P_r\{Y_i = 1 | X_i = 1\} \times P_r\{X_i = 1 | Y_i = 1\} + P_r\{Y_i = 0 | X_i = 1\} \times P_r\{X_i = 1 | Y_i = 0\}$$

This expression captures the "round-trip" of going from the true database to the distorted database and then returning to guess the contents of the true database. It can be simplified to

$\mathcal{R}_1(p, s_i) =$

$$p \times P_r\{X_i = 1 | Y_i = 1\} + (1 - p) \times P_r\{X_i = 1 | Y_i = 0\}$$

But, we know that

$$P_r\{X_i = 1 | Y_i = 1\} = \frac{P_r\{X_i=1 \cap Y_i=1\}}{P_r\{Y_i=1\}}$$

$$= \frac{P_r\{X_i=1\} \times P_r\{Y_i=1 | X_i=1\}}{P_r\{Y_i=1\}}$$

$$= \frac{s_i \times p}{P_r\{X_i=1\} \times P_r\{Y_i=1|X_i=1\} + P_r\{X_i=0\} \times P_r\{Y_i=1|X_i=0\}}$$

$$= \frac{s_i \times p}{s_i \times p \ + \ (1-s_i) \times (1-p)}$$

Similarly,

$$P_r\{X_i = 1 | Y_i = 0\} = \frac{s_i \times (1-p)}{s_i \times (1-p) \ + \ (1-s_i) \times p}$$

Putting it all together, we obtain

$$\mathcal{R}_1(p, s_i) = \frac{s_i \times p^2}{s_i \times p + (1-s_i) \times (1-p)} + \frac{s_i \times (1-p)^2}{s_i \times (1-p) + (1-s_i) \times p}$$

The above expression reflects the reconstruction probability of a '1' in a random item i. To find a total measure of reconstruction, we range across all items:

$$\mathcal{R}_1(p) = \frac{\Sigma_i \ s_i \mathcal{R}_1(p, s_i)}{\Sigma_i \ s_i} \quad (1)$$

The above expression is *minimized* when all the items in the database have the same support, and increases with the variance in the supports across items. As discussed later in Section 7, with the appropriate choice of p, this increase is marginal for the market-basket type of datasets that we consider here. Therefore, as a first-level approximation, we replace the item-specific supports in the above equation by s_0, the average support of an item in the database. With this, the reconstruction probability simplifies to

$$\mathcal{R}_1(p) = \frac{s_0 \times p^2}{s_0 \times p + (1-s_0) \times (1-p)} + \frac{s_0 \times (1-p)^2}{s_0 \times (1-p) + (1-s_0) \times p} \quad (2)$$

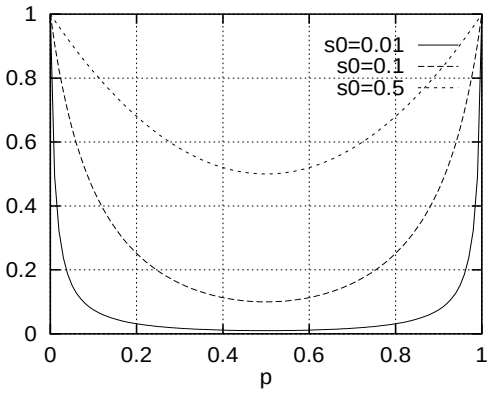

Figure 1: Reconstruction Probability $\mathcal{R}_1$(p)

We can plot $\mathcal{R}_1$(p) as a function of p for different values of s_0, as shown in Figure 1. We observe here that:

1. The reconstruction probability is high at the extremes and lowest at the center (i.e $p = 0.5$). This is to be intuitively expected since setting $p = 0.5$ imparts the maximum randomness to the distorted values.

2. The curves are *symmetric* around $p = 0.5$. The implication of this is that there is no difference, reconstruction-wise, between choosing a value p or its counterpart $1 - p$. This may appear surprising at first glance since a matrix that is distorted with $p = 0.1$ would tend to "look" very different with regard to the true matrix, as compared to the same matrix distorted with $p = 0.9$. However, recall that the data miner is also provided with a description of the distortion procedure, that is, he knows the value of p. In this situation, mere differences in appearance do not result in any additional privacy. A practical use of this feature, however, is in *psychological* terms: Distorting with a low value of p as opposed to its high-valued complement, might be more comforting to the user since at least visually it will appear to be considerably different from the true information that she had supplied.

3. Although the minimum always is at $p = 0.5$, the curves become flatter as the average support of items decreases. For a typical market-basket type database with an average transaction length of 10 and the number of items being 1000, the average support is 0.01, which corresponds to the lowest curve in Figure 1.

In the above derivation, it may appear unintuitive that the reconstruction probability depends on the *support* of items. The reason for this is the following: We are considering the possibility of reconstruction of the true value of an entry given the distorted entry. If the data miner gets a '1' (or a '0') for a particular entry in the distorted database, the probability that it came from a '1' in the true database not only depends on p but also on the distribution of 1's and 0's in the true database.

Yet another issue is that we have used the *true* supports of items in the derivation, but these values are not known to the data miner. Therefore, it may appear that we are overestimating the reconstruction probability. However, the point is that since the ultimate goal is to be able to mine the distorted database correctly, we make the conservative assumption that the miner will be able to derive reasonably accurate item supports, implying that he *does* have access to the s_i values.

3.3 The General Reconstruction Equation

We now move on to deriving the relationship between p and the reconstruction probability for the general case where the customer may wish to protect both her 1's and 0's, but her concern to keep the 1's private is more than that for the 0's.

Analogous to the manner in which we computed $\mathcal{R}_1$(p) above, we can derive the probability with which a '0' can be reconstructed as:

$\mathcal{R}_0(p, s_i) =$

$$P_r\{Y_i = 1|X_i = 0\} \times P_r\{X_i = 0|Y_i = 1\}$$
$$+$$
$$P_r\{Y_i = 0|X_i = 0\} \times P_r\{X_i = 0|Y_i = 0\}$$

leading to

$$\mathcal{R}_0(\mathrm{p}) = \frac{(1-s_0)\times p^2}{(1-s_0)\times p + s_0 \times (1-p)} + \frac{(1-s_0)\times (1-p)^2}{s_0 \times p + (1-s_0)\times (1-p)}$$

Our aim is to minimize a weighted average of $\mathcal{R}_1$(p) and $\mathcal{R}_0$(p). This corresponds to minimizing the probability of reconstruction of both 1's and 0's. The weight denotes the preference which the privacy of 1's has over that of 0's. The total reconstruction probability, $\mathcal{R}$(p), is then given as

$$\mathcal{R}(p) = a\mathcal{R}_1(p) + (1 - a)\mathcal{R}_0(p) \qquad (3)$$

where a is the weight given to 1's over 0's. Note that the a setting must incorporate the fact that the number of 0's in the database is more than that of 1's. So, for example, if we set $a = 0.5$ for a database that has $s_0 = 0.01$, we are indicating that the privacy of 1's is 99 times more critical than that of 0's (as the number of 0's is 99 times more than that of 1's).

3.4 Privacy Measure

Armed with the ability to compute the reconstruction probability, we now simply define user privacy as the following percentage:

$$\mathcal{P}(p) = (1 - \mathcal{R}(p)) * 100. \qquad (4)$$

p	Privacy Attained
0.5	89%
0.7	88%
0.8	87%
0.9	83%
0.95	76%
1	0%

Table 1: Privacy attained with $s_0 = 0.01$ and $a = 0.9$

That is, when the reconstruction probability is 0, the privacy is 100%, whereas it is 0 if the $\mathcal{R}$(p)= 1. In Figure 2, we plot this user privacy as a function of p for $s_0 = 0.01$ with different values of a. Note that for a given value of s_0, the shape of the curve is fixed, and it is only the value of a that decides the absolute value of the attained privacy.

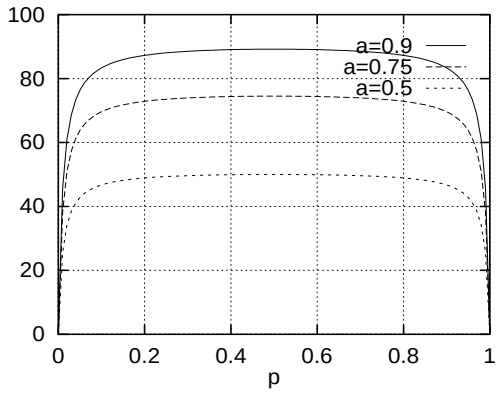

Figure 2: Privacy $\mathcal{P}$(p) attained for $s_0 = 0.01$

Further, note that the curves have the "knee-points" at $p = 0.1$ and $p = 0.9$ and that for $a = 0.9$, the privacy is almost constant at a high value of around 85% in this large range (0.1 to 0.9) of distortion probabilities – the explicit values are shown in Table 1. This result is very encouraging since it means that we now have considerable flexibility in choosing the p value – in particular, we can choose it in a manner that will *minimize the error* in the subsequent mining process.

4 Mining the Distorted Database

Having established the privacy obtained from our distortion procedure, we now move on to presenting MASK's technique for estimating the true (accurate) supports of itemsets from a distorted database. Later, in Section 5, we present a variety of optimizations that help to speed up the estimation process. Finally, in Section 7, we evaluate the quality of these estimations.

In the following discussion, we first show how to estimate the supports of 1-itemsets (i.e. singletons) and then present the general n-itemset support estimation procedure. In this derivation, it is important to keep in mind that the miner is provided with both the distorted matrix as well as the distortion procedure, that is, he *knows* the value of p that was used in distorting the true matrix.

4.1 Estimating Singleton Supports

We denote the original true matrix by T and the distorted matrix, obtained with a distortion probability of p, as D. Now consider a random individual item **i**. Let c_1^T and c_0^T represent the number of 1's and 0's, respectively, in the **i** column of T, while c_1^D and c_0^D represent the number of 1's and 0's, respectively, in the **i** column of D. With this notation, we estimate the support of **i** in T using the following equation:

$$\mathbf{C^T = M^{-1} C^D} \tag{5}$$

where

$$M = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \quad C^D = \begin{bmatrix} c_1^D \\ c_0^D \end{bmatrix} \quad C^T = \begin{bmatrix} c_1^T \\ c_0^T \end{bmatrix}$$

The M matrix in the above equation incorporates the observation that by our method of distortion, if a column had n 1's in T, these 1's will generate approximately pn 1's and $(1-p)n$ 0's for the same column in D. Similarly for the 0's of this column in T. Therefore, given c_1^D and c_0^D, it is possible to estimate the value of c_1^T, that is, the true support of item **i**.

Note also that the equation rules out the possibility of using $\mathbf{p = 0.5}$ because at this value the matrix becomes singular and is not invertible. Intuitively, this happens because at this value of p, the matrix $\mathbf{C^D}$ does not carry sufficient information to be able to reconstruct the values of c_1^T and c_0^T.

4.2 Estimating n-itemset Supports

It is easy to extend Equation 5, which is applicable to individual items, to compute the support for an arbitrary n-itemset. For this general case, we define the matrices as:

$$C^D = \begin{bmatrix} c_{2^n-1}^D \\ \cdot \\ \cdot \\ \cdot \\ c_1^D \\ c_0^D \end{bmatrix} \quad C^T = \begin{bmatrix} c_{2^n-1}^T \\ \cdot \\ \cdot \\ \cdot \\ c_1^T \\ c_0^T \end{bmatrix}$$

Here c_k^T should be interpreted as the count of the tuples in T that have the binary form of k (in n digits) for the given itemset (that is, for a 2-itemset, c_2^T refers to the count of 10's in the columns of T corresponding to that itemset, c_3^T to the count of 11's, and so on). Similarly, c_k^D is defined for the distorted matrix D.

Finally, the matrix $\mathbf{M}$ is defined as:

$m_{i,j}$ = The probability that a tuple of the form corresponding to c_j^T in T goes to a tuple of the form corresponding to c_i^D in D

For example, $m_{1,2}$ for a 2-itemset is the probability that a 10 tuple distorts to a 01 tuple. Accordingly, $m_{1,2} = (1-p)(1-p)$. The basis for this formulation lies in the fact that in our distortion procedure, the component columns of an n-itemset are distorted *independently*. Therefore, we can use the product of the probability terms.

At first glance, it might seem that the above mining process may need an *exponential* number of counters (2^n counters for an n-itemset), making the process infeasible in practice. But, in fact, this is not really so if the same value of p is used for all items. This is because it results in the matrix $\mathbf{M}$ having interesting symmetry properties that are reflected in $\mathbf{M^{-1}}$ also. In particular, we show in Section 5 how a *linear* number of counters (specifically, $n+1$ counters for an n-itemset) are now sufficient to complete the mining process.

4.3 The Full Mining Process

The above equations help us to estimate the value of $c^T_{2^n-1}$ for an n-itemset by using the values of c^D_i, $0 \leq i \leq 2^n - 1$. But, first we need to compute the c^D_i values themselves. For this purpose, in principle we could use, after the modifications described below, any of the numerous association rule mining algorithms proposed in the literature (e.g. [3, 13, 9, 16]). With a view to simplicity, we have currently implemented our system based on the classical *Apriori* algorithm [3]. Apriori is a multi-pass algorithm wherein the i^{th} pass computes the frequent i-itemsets by counting all candidate itemsets associated with the pass and, after each pass, the *AprioriGen* algorithm is used for generating the candidate itemsets for the next pass. In our approach too, the i^{th} pass identifies large i-itemsets, and the AprioriGen algorithm is used for generating the candidate itemsets for the next pass.

A critical difference between our approach and that of Apriori, however, is the following: Consider that we are counting, say, 2-itemsets. Here, Apriori only needs to keep track, for each candidate 2-itemset, of the number of tuples in which there was a '1' for both the items appearing in the itemset. That is, it needs to count only the '11's. But, in our case, we need to keep track of *all* combinations: 00, 01, 10, and 11. This is a direct fallout of the fact that we have distorted the true matrix, and an original '11' can now potentially be any of the four combinations. We describe in Section 5 a simple optimization that can significantly reduce the total amount of counting.

Another important point to note here is that the equation to compute the true supports needs to be evaluated only at the *end of every pass* over the distorted matrix, and not on a tuple-by-tuple basis. Further, if the same p value is used for all columns, the matrix $\mathbf{M}$ is identical for *all* candidate n-itemsets and therefore has to be generated *only once* at the end of each pass. Finally, note that the size of the square matrix is $\mathbf{O(2^n)}$ for candidate n-itemsets.

5 MASK Mining Optimizations

We now describe a set of optimizations to improve the efficiency of the mining process described in the previous section.

5.1 Linear Number of Counters

We first consider how to reduce the number of counters required for each itemset. At the end of the n^{th} pass, we generate a square matrix of size 2^n which depends only on p. We then invert it and multiply the resulting matrix with the counts of the 2^n components of every n-itemset. In effect, the reconstructed support is a *weighted sum* of the counts of all 2^n components in the distorted database. A close observation will reveal, however, that these 2^n weights have *only* $n+1$ *distinct* weights in them.

For example, for a 2-itemset, we have the estimated reconstructed support value to be

$$s_{est} = a_1 C^D_{00} + a_2 C^D_{01} + a_3 C^D_{10} + a_4 C^D_{11}$$

where C^D_{xy} is the count of xy tuples in the distorted database, and the a_i are the associated weights. Here, the weights a_2 and a_3 will be equal because the probability that a '11' distorts to a '10' is equal to the probability that a '11' distorts to a '01' (both are $p(1-p)$). Hence, the *reverse* component weights are also equal. Therefore, overall, we need to maintain only 3 counters – one each for 00's and 11's, and a third which is common for 01's and 10's.

The above observation can be generalized to an n-itemset. For the 2^n counts we have merely $n+1$ distinct weights: One for '00..0', one for '11..1' and one each for components that have the same number of 1's (or equivalently 0's) in them. For example, for a 3-itemset, only 4 counters are required: one each for 000 and 111, one for the triplet (001,010,100) and the fourth for the triplet (011,101,110).

5.2 Reducing Amount of Counting

Apart from reducing the *number* of counters, we can also achieve some reductions in the *amount* of counting by making use of simple algebraic properties.

For example, consider 2-itemsets: The counting of only 11's takes $\mathbf{O}(tlen^2)$ operations, where *tlen* is the transaction length. However, counting all 4 components (00,01,10,11), which we have to do for the distorted matrix, takes $\mathbf{O}(|\mathbf{Candidates}|)$ operations. Since the number of candidate *2*-itemsets is typically very large, the latter will take much more time than the ordinary mining process.

We can optimize the above by using the observation that $c^D_3 + c^D_2 + c^D_1 + c^D_0$ must be equal to the database

cardinality, *dbsize*. This means that we can choose to *not count* one of these components, say '00's, since it is derivable from the other counts. In conjunction with the counting process described below, this optimization can result in considerable savings.

Our counting process examines the (distorted) customer tuples one by one. For the current customer tuple, the purchase vector of 1's and 0's is converted into an *item-list* that contains only the identifiers of the items that have a '1' in the transaction. From this list, the identifiers of all the (previously estimated) infrequent 1-itemsets are removed. The next stage is to create a *complement-list* that consists of all the (previously estimated) frequent 1-itemsets that *do not appear* in this transaction. Let the item-list and the complement-list be of lengths m_1 and m_2, respectively, (with $m_1 + m_2 =\mid F_1 \mid$, the total number of frequent 1-itemsets). If we now restrict our counting to the 11's, 01's and 10's, it will take $\mathbf{O(m_1^2)+O(m_1 m_2)}$ operations. For high settings of the p parameter, the value of m_1 will be rather small and the transaction will have a large number of '00' pairs. The approach of not counting the '00's can therefore result in significant savings in such situations. In fact, we observed in our experiments (described in Section 7) that for $p = 0.9$, the execution time for Pass 2 reduced by a factor of *four* due to this optimization.

6 Performance Framework

The privacy of the MASK scheme was analytically evaluated in Section 3. We now move on to evaluating its accuracy with regard to the mining results that it derives from the distorted database.

Since MASK is a *probabilistic* approach, fundamentally we cannot expect the reconstructed support values to co-incide exactly with the actual supports. This means that we may have errors in the estimated supports of frequent itemsets with the reported values being either larger or smaller than the actual supports.

Errors in support estimation can have an even more pernicious effect than just wrongly reporting the support of a frequent itemset. They can result in errors in the *identities* of the frequent itemsets. This becomes especially an issue when the sup_{min} setting is such that the support of a number of itemsets lie very close to this threshold value. Typically, this happens for lower values of sup_{min} due to the larger number of frequent itemsets in the database. Such "border-line" itemsets may get wrongly reported as either frequent or rare, based on how the probabilistic evaluation estimates their supports. That is, we can encounter both *false positives* and *false negatives*.

Worse, errors in association rule mining *percolate* through the various passes of the mining process – that is, an error in identifying a 1-itemset correctly has a ripple effect in terms of causing errors in the remainder of the frequent itemset lattice.

To assess the above effects, we evaluate the mining process under two conditions: The first with the sup_{min} value provided by the user, and the second with a marginally *lower* value. The lower value is expected to help reduce the number of false negatives at the risk of increasing the number of false positives, but this is in keeping with the view that it is more important to have coverage as compared to precision. For the experiments described here, we evaluate the impact of a 10% reduction, denoted $r = 10\%$, in the sup_{min} value.

To quantify the errors that we are making, we compare our outputs with those derived from Apriori running on the true database with both the user provided sup_{min}, as well as with the lowered value mentioned above.

6.1 Data Sets

Our evaluations were carried out on two representative databases:

1. A synthetic database generated from the IBM Almaden generator [3]. The synthetic database was created with parameters T10.I4.D1M.N1K (as per the naming convention of [3]), resulting in a million customer tuples with each customer purchasing about ten items on average.

2. A real dataset, BMS-WebView-1 [20], placed in the public domain by Blue Martini Software. This database contains click-stream data from the web site of a (now defunct) legwear and legcare retailer. There are about 60,000 tuples with close to 500 items in the schema. In order to ensure that our results were applicable to large disk-resident databases, we scaled this database by a factor of ten, resulting in approximately 0.6 million tuples.

The measured values of s_0 for these two databases turned out to be 0.01 and 0.005, respectively.

6.2 Support and Distortion Settings

We evaluated the mining accuracy of MASK on the above datasets for a variety of sup_{min} and p values. We present here the results for two sup_{min} values, namely 0.25% and 0.5%. The 0.25% sup_{min} value represents, in a sense, the "worst-case" environment for our algorithm due to the presence of a large number of border-line itemsets.

The p values we consider are $p = 0.9$ and $p = 0.7$ (recall that these are equivalent to $p = 0.1$ and $p = 0.3$, respectively, with regard to privacy). For these settings, the associated privacy values for 1's, as computed from Equation 1, are 85% and 96%, respectively, for the synthetic database, and 89% and 97%, respectively, for the real database.

6.3 Error Metrics

We evaluate two kinds of mining errors, Support Error and Identity Error, in our experiments:

Support Error (ρ) :

This metric reflects the (percentage) average relative error in the reconstructed support values for those itemsets that are correctly identified to be frequent. Denoting the reconstructed support by rec_sup and the actual support by act_sup, the support error is computed over all frequent itemsets as

$$\rho = \frac{1}{\mid f \mid} \Sigma_f \frac{\mid rec\_sup_f - act\_sup_f \mid}{act\_sup_f} * 100$$

We compute this metric individually for each level of itemsets, that is, for 1-itemsets, 2-itemsets, etc.

Identity Error (σ) :

This metric reflects the percentage error in identifying frequent itemsets and has two components: σ^+, indicating the percentage of false positives, and σ^- indicating the percentage of false negatives. Denoting the reconstructed set of frequent itemsets with R and the correct set of frequent itemsets with F, these metrics are computed as:

$$\sigma^+ = \frac{|R-F|}{|F|} * 100 \qquad \sigma^- = \frac{|F-R|}{|F|} * 100$$

7 Experimental Results

We now present the results of our experiments conducted under the framework described above. The results for the synthetic database are presented first, followed by those for the real database.

7.1 Synthetic Database

Experiment 1: p=0.9, sup_{min}=0.25%, r=0%

Our first experiment was conducted on the synthetic database with a distortion parameter of p=0.9, sup_{min}=0.25%, and no relaxation. As mentioned earlier, this experiment represents, in a sense, the worst case scenario for MASK due to the extremely low sup_{min} value.

The results for this experiment are shown in Table 2 – in this table, the level indicates the length of the frequent itemset, $\mid F \mid$ indicates the number of frequent itemsets at this level, and the other three columns are the error metrics defined in the previous section.
The results indicate that firstly the support error (ρ) is reasonably small, less than 5% at all levels. Secondly, the negative identity error is also small, not exceeding 6% at the maximum. Finally, the positive identity error is also in the same range. Note that the maximum errors occur for the 2-itemsets, which is as per expectations since the number of itemsets is the maximum at this level.

Level	$\mid F \mid$	ρ	σ^-	σ^+
1	689	3.31	1.16	1.16
2	2648	3.58	4.49	5.14
3	1990	1.71	4.57	2.16
4	1418	1.28	3.67	0.22
5	730	1.27	5.89	0
6	212	1.36	4.25	5.19
7	35	1.40	0	0
8	3	0.99	0	0

Table 2: p=0.9,sup_{min}=0.25%,r=0%,Synthetic

Experiment 2: p=0.9, sup_{min}=0.25%, r=10%

Our second experiment evaluated the effect of marginally relaxing the sup_{min} value by 10% – that is, using a sup_{min} of 0.225% instead of 0.25%, keeping the rest of the parameters the same as that of Experiment 1. The ρ and σ results for this experiment are shown in Table 3.

Level	$\mid F \mid$	ρ	σ^-	σ^+
1	689	3.37	0.73	3.19
2	2648	3.73	0.19	19.68
3	1990	1.76	0	28.09
4	1418	1.29	0	25.81
5	730	1.32	0	16.44
6	212	1.37	0	25.47
7	35	1.40	0	51.43
8	3	0.99	0	66.67

Table 3: p=0.9,sup_{min}=0.25%,r=10%,Synthetic

We see here that while the support error shows little change as compared to the previous experiment, the negative identity error goes down very significantly, becoming less than 1%, achieving the desired goal. The price to pay for this, however, is the substantial increase in the positive identity error. The marked change in the values of the identity errors also highlights the significant presence of "border-line" itemsets in the database.

Experiment 3: p=0.9, sup_{min}=0.5%, r=0%

We now move on to repeating Experiment 1 for an increased sup_{min} value of 0.5, corresponding to a "nicer" environment for MASK. The results of this experiment are shown in Table 4.
The results here show that the errors generally reduce as compared to Experiment 1 due to the sparser distribution of frequent itemsets. Further, 2-itemsets continue to be associated with the maximum error.

Experiment 4: p=0.9, sup_{min}=0.5%, r=10%

Our next experiment evaluated the effect of marginally relaxing the sup_{min} value by 10% – that is, using a

Level	$\|F\|$	ρ	σ^-	σ^+
1	560	2.60	1.25	0.89
2	470	2.13	5.53	4.89
3	326	1.22	3.07	0.31
4	208	1.34	1.44	0.48
5	125	1.81	0	0
6	43	2.62	0	0
7	10	3.44	10	0
8	1	4.50	0	0

Table 4: p=0.9,sup_{min}=0.5%,r=0%,Synthetic

sup_{min} of 0.45% instead of 0.5% – keeping the rest of the parameters the same as that of Experiment 3. The ρ and σ results for this experiment are shown in Table 5.

Level	$\|F\|$	ρ	σ^-	σ^+
1	560	2.66	0.18	4.29
2	470	2.21	0	44.89
3	326	1.26	0	42.64
4	208	1.35	0	51.44
5	125	1.81	0	22.4
6	43	2.62	0	18.60
7	10	3.47	0	10
8	1	4.50	0	0

Table 5: p=0.9,sup_{min}=0.5%,r=10%,Synthetic

Similar to Experiment 2, the results here too indicate that a marginal relaxation can almost completely eliminate the false negative error component, ensuring that none of the true frequent itemsets are missed. At the same time, the false positive error goes up considerably since trying to "catch all the good fish" inevitably also attracts unwanted material.

Experiment 5: p=0.7, sup_{min}=0.25%, r=10%

The previous experiments were all run at a distortion probability of $p = 0.9$ corresponding to a privacy factor of 85%. We now consider the possibility of improving the privacy measure to 96% by changing to $p = 0.7$ and evaluate the impact of this change on the accuracy. The results for this experiment are shown in Table 6.

We see from these results that there is a dramatic increase in all the error metrics. For example, the support error goes up to about 25% on average, while the identity errors are huge. In fact, the errors go up to the extent that the results produced by the mining process are essentially meaningless. The implication is that privacy and accuracy represent an extremely sensitive tradeoff and that there is only a small parameter region wherein we can hope to obtain reasonable values for both metrics.

Level	$\|F\|$	ρ	σ^-	σ^+
1	689	10.16	2.61	7.84
2	2648	25.23	19.52	630.93
3	1990	26.93	42.86	172.71
4	1418	29.14	65.94	0.35
5	730	28.47	79.32	0
6	212	36.25	84.91	0
7	35	51.37	85.71	0
8	3	–	100	0

Table 6: p=0.7,sup_{min}=0.25%,r=10%,Synthetic

7.2 Real Database

We now move on to experiments conducted on the real database which, as mentioned earlier, contains information about the click-stream logs of an online legwear manufacturer. For ease of comparison, these experiments modeled exactly the same environments as those evaluated for the synthetic database – that is, Experiments 6 through 10 presented below correspond one-to-one with Experiments 1 through 5.

Experiment 6: p=0.9, sup_{min}=0.25%, r=0%

Level	$\|F\|$	ρ	σ^-	σ^+
1	249	5.89	4.02	2.81
2	239	3.87	6.69	7.11
3	73	2.60	10.96	9.59
4	4	1.41	0	25.0

Table 7: p=0.9,sup_{min}=0.25%,r=0%,Real

The results of this experiment are shown in Table 7. We observe here that the errors, while still acceptably low, are somewhat higher than the corresponding numbers for the synthetic database. But, in fact, the absolute number of errors is fewer and it is due to the much smaller number of frequent itemsets that the effects of these errors are magnified. For example, the number of 2-itemsets in the real database is an order of magnitude smaller than that in the synthetic database.

Experiment 7: p=0.9, sup_{min}=0.25%, r=10%

Level	$\|F\|$	ρ	σ^-	σ^+
1	249	6.12	1.2	0.40
2	239	4.04	1.26	23.43
3	73	2.93	0	45.21
4	4	1.41	0	75

Table 8: p=0.9,sup_{min}=0.25%,r=10%,Real

The results of this experiment are shown in Table 8. We observe that these results are similar to those of Experiment 2 – the basic errors are low and relaxation

significantly reduces the negative identity error at an attendant increase in the positive identity error.

Experiment 8: p**=0.9,** sup_{min}**=0.5%,** r**=0%**

Level	$\mid F \mid$	ρ	σ^-	σ^+
1	150	4.23	0.67	4.67
2	45	2.42	2.22	4.44
3	6	1.07	0	16.66

Table 9: p=0.9,sup_{min}=0.5%,r=0%,Real

The results of this experiment are shown in Table 9. Here, we see that the stricter support threshold results in a very small set of frequent itemsets and that the errors are acceptably low.

Experiment 9: p**=0.9,** sup_{min}**=0.5%,** r**=10%**

Level	$\mid F \mid$	ρ	σ^-	σ^+
1	150	4.27	0	8
2	45	2.56	0	37.77
3	6	1.07	0	66.66

Table 10: p=0.9,sup_{min}=0.5%,r=10%,Real

The results of this experiment are shown in Table 10. We see that the relaxation completely removes all false negatives, with the expected attendant increase in the false positives.

Experiment 10: p**=0.7,** sup_{min}**=0.25%,** r**=10%**

Level	$\mid F \mid$	ρ	σ^-	σ^+
1	249	18.96	7.23	15.66
2	239	33.59	20.08	1907.53
3	73	32.87	30.14	2308.22
4	4	7.55	50	400

Table 11: p=0.7,sup_{min}=0.25%,r=10%,Real

The results of this experiment are shown in Table 11. These results are similar to those seen in Experiment 5 – the reduction in distortion probability results in a disproportionate increase in the errors for both the support and the identity metrics.

7.3 Summary

Overall, our experiments indicate that by a careful choice of distortion probability, it is possible to simultaneously achieve satisfactory privacy and accuracy. In particular, they show that there is a small "window of opportunity" around the $p = 0.9$ value where these dual goals can be met. Moving away from this window towards lower values of p, however, results in skyrocketing errors, while increasing the value of p will result in significant loss of privacy.

We have conducted several other experiments with different market-basket type databases and the results are consistent with those presented here. One other issue is the running time of our mining algorithm. As mentioned earlier, mining the distorted database is intrinsically significantly more expensive than mining the real database. Currently, we have not yet implemented all the optimizations described in Section 5, and we have therefore refrained from presenting the running times since these numbers will change when all the optimizations are in place. The current situation is that we are able to mine the real database at 0.25% support in about 30 minutes on a low-end Pentium machine.

8 Reconstruction Error Bounds

As shown by our experiments of the previous section, mining the distorted database does result in errors in the reconstructed support. We now provide a loose probabilistic bound on these errors, focusing specifically on 1-itemsets since, as mentioned earlier, their errors percolate through the entire mining process.

Consider a single column in the true matrix. Suppose, it has n 1's and $dbsize - n$ 0's. We expect that the n 1's will distort to np 1's and $n(1-p)$ 0's when distorted with parameter p. Similarly, we expect the 0's to go to $(dbsize - n)p$ 0's and $(dbsize - n)(1-p)$ 1's. However, note that this is assuming that "When we generate a random number, which is distributed as bernoulli(p), then the number of 1's, denoted by P, in n trials is actually np". But, in reality, this will not be so. Actually, P is distributed as *binomial(n,p)*:

$$\text{P} = \text{binomial(n,p)}$$
$$P_r\{P = r, 0 \leq r \leq n\} = {}^nC_r p^r (1-p)^{n-r}$$

As the value of n increases, the sample space of possible outcomes expands. And the probability that $P = np$ decreases. But, we are interested in an error bound around the value np. So, we define a function $P^E(n, p, \epsilon)$ as :

$$P^E(n, p, \epsilon) = P_r\{|Number\ of\ 1's - np| < \epsilon\}$$

That is, $P^E(n, p, \epsilon)$ measures the probability that when the above experiment is conducted, the number of 1's is between $np - \epsilon$ and $np + \epsilon$. Clearly,

$$P^E(n, p, \epsilon) = \Sigma_{r=np-\epsilon}^{np+\epsilon} {}^nC_r p^r (1-p)^{n-r}$$

Now, let the true column have n 1's and m 0's and the distorted column have n' 1's and m' 0's. Then, given the distorted column, MASK reconstructs the support to

$$\overline{n} = \frac{pn'}{2p-1} - \frac{(1-p)m'}{2p-1}$$

$$\Rightarrow \overline{n} = \frac{n'}{2p-1} - \frac{(1-p)dbsize}{2p-1} \quad (6)$$

We now find the possible error in this approximation by using the function P^E. First, with probability $P^E(m,p,\epsilon_1)$, the number of 1's generated from the m 0's in the initial column is between $m(1-p)-\epsilon_1$ and $m(1-p)+\epsilon_1$. Next, with probability $P^E(n,p,\epsilon_2)$, the number of 1's generated from the n 1's in the initial column is between $np-\epsilon_2$ and $np+\epsilon_2$. Finally, with probability $P^E(m,p,\epsilon_1) \times P^E(n,p,\epsilon_2)$ the number of 1's in the distorted database (n') is between $m(1-p)+np-(\epsilon_1+\epsilon_2)$ and $m(1-p)+np+(\epsilon_1+\epsilon_2)$.

Note that the approximation in Equation 6 is based on the expectation that n' is equal to $m(1-p)+np$. Hence, with probability $P^E(m,p,\epsilon_1) \times P^E(n,p,\epsilon_2)$, we can assert that the error in the value of n' is

$$\Delta n' \ \leq \ \epsilon_1 + \epsilon_2$$

Now, from Equation 6, we know that the error in reconstruction is related to the error in the value of n' as follows:

$$\Delta\overline{n} = \frac{\Delta n'}{2p-1}$$

Therefore, we conclude that

$$\Rightarrow \Delta\overline{n} \ \leq \ \frac{\epsilon_1+\epsilon_2}{2p-1}$$

In general, let $\epsilon_1 = \epsilon_2 = \frac{2p-1}{2}\epsilon$. Then we assert "With probability $P^E(m,p,\frac{2p-1}{2}\epsilon) \times P^E(n,p,\frac{2p-1}{2}\epsilon)$, the error in reconstruction is less than ϵ".

Note that in the above derivation, we have considered the worst case when both the errors are in the same direction, that is, deviating n' from the value $np+m(1-p)$ in the same direction. This is the reason for adding the values of ϵ_1 and ϵ_2. In practice, however, we could expect that there is a reasonable likelihood that the errors follow opposite directions and therefore partially or fully negate each other.

9 Related Work

Concurrently with our work, the issue of maintaining privacy in association rule mining has attracted considerable attention over the last year [14, 5, 6, 15, 19, 11, 8]. The problem addressed in [14, 5, 6, 15] is how to prevent *sensitive rules* from being inferred by the data miner – the proposed solutions involve either falsifying some of the entries in the true database or replacing them with NULL values. This work is complementary to ours since it addresses concerns about *output* privacy, whereas our focus is on the privacy of the *input* data. Also note that, by definition, these techniques require a completely materialized true database as the starting point whereas our approach can operate during the data collection process itself.

Maintaining input data privacy is considered in [19, 11] in the context of databases that are *distributed* across a number of sites with each site only willing to share data mining results, but not the source data. While [19] considers data that is vertically partitioned (i.e., each site hosts a disjoint subset of the matrix columns), the complementary situation where the data is horizontally partitioned (i.e., each site hosts a disjoint subset of the matrix rows) is addressed in [11]. The solution technique in [19] requires generating and computing a large set of independent linear equations – in fact, the number of equations and the number of terms in each equation is proportional to the *cardinality* of the database. It may therefore prove to be expensive for market-basket databases which typically contain millions of customer transactions. In [11], on the other hand, the problem is modeled as a secure multi-party computation [10] and an algorithm that minimizes the information shared without incurring much overhead on the mining process is presented. While these techniques are meaningful only in the context of distributed databases, our work is applicable to both centralized and distributed databases. Further, they assume a pre-existing true database at each site, whereas, as mentioned earlier, our approach can provide privacy directly at the user machine. Finally, a set of randomization operators for maintaining data privacy are presented and analyzed in [8].

Privacy-preserving mining in the context of *classification rules* has been investigated recently in [4, 1, 12]. Cryptographic protocols to ensure complete privacy in developing decision tree classifiers across distributed databases were presented in [12]. An alternative value distortion approach wherein a random value is added to each original value was taken in [4] and the privacy attained was quantified by the "fuzziness" provided by the system, that is, for a given level of confidence, the size of the interval that is expected to hold the original true value. Since we assume boolean data values, such an interval-based approach is not applicable in our context.

It was shown in [1] that the privacy estimates of [4] had to be lowered when the additional knowledge that the miner obtains from the reconstructed aggregate distribution was included in the problem formulation. The decrease is primarily due to the fact that values of the distorted data may lie *outside* the domain of the original data, thereby narrowing the interval associated with the true value. This problem cannot occur in our scenario, since both the original data and the distorted data have exactly the same domain, namely, 0 and 1. However, it is still possible, as explained very recently in [8], that the association rules forming the mining output may be used to re-interrogate

the distorted database and thereby reduce the privacy measure.

10 Conclusions

We have investigated the problem of supporting the conflicting goals of privacy and accuracy while mining association rules on large databases. Specifically, we presented a privacy metric and an analytical formula for evaluating the privacy of our MASK scheme, which is based on probabilistic distortion of user data, according to this metric. The formula showed that privacy is a function of the sparseness of the true matrix, as well as of the relative weight given to the privacy of 1's as compared to 0's.

A mining process for generating frequent itemsets from the distorted database was also presented, along with a set of optimizations to address the fact that mining the distorted database is significantly more expensive than mining the true database. The optimizations significantly reduced both the number of counters and the amount of counting that needs to be used in the mining process.

Our experimental results on synthetic and real databases showed that a distortion probability of $p = 0.9$ (equivalently, $p = 0.1$) is ideally suited to provide both privacy and good mining results for the sparse market-basket type of databases that we have considered in this study. Specifically, a privacy of over 80% and an error of less than 10% were simultaneously achieved with this setting.

In our future work, we plan to investigate the extension of our results to generalized [17] and quantitative [18] association rules. We also plan to refine our privacy estimation formulas to include the effects of using the mining output to re-interrogate the distorted database.

Acknowledgements

S. J. Rizvi was supported by a Summer Research Fellowship from the Centre for Theoretical Studies, Indian Institute of Science. J. R. Haritsa was supported by a research grant from the Dept. of Bio-technology, Govt. of India. We thank S. Chakrabarti of IIT-Bombay for his inputs during the early part of this work.

References

[1] D. Agrawal and C. Aggarwal, "On the Design and Quantification of Privacy Preserving Data Mining Algorithms", *Proc. of 20th ACM Symp. on Principles of Database Systems (PODS)*, May 2001.

[2] R. Agrawal, T. Imielinski and A. Swami, "Mining association rules between sets of items in large databases", *Proc. of ACM SIGMOD Intl. Conference on Management of Data (SIGMOD)*, May 1993.

[3] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", *Proc. of 20th Intl. Conf. on Very Large Data Bases (VLDB)*, September 1994.

[4] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 2000.

[5] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim and V. Verykios, "Disclosure Limitation of Sensitive Rules", *Proc. of IEEE Knowledge and Data Engineering Exchange Workshop (KDEX)*, November 1999.

[6] E. Dasseni, V. Verykios, A. Elmagarmid and E. Bertino, "Hiding Association Rules by Using Confidence and Support", *Proc. of 4th Intl. Information Hiding Workshop (IHW)*, April 2001.

[7] D. Denning, *Cryptography and Data Security*, Addison-Wesley, 1982.

[8] A. Evfimievski, R. Srikant, R. Agrawal and J. Gehrke, "Privacy Preserving Mining of Association Rules", *Proc. of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, July 2002.

[9] C. Hidber, "Online association rule mining", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 1999.

[10] O. Goldreich, "Secure Multi-party Computation", *www.wisdom.weizmann.ac.il/~oded/pp.html*, 1998.

[11] M. Kantarcioglu and C. Clifton, "Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data", *Proc. of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, June 2002.

[12] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining", *Advances in Cryptology – CRYPTO 2000*, Springer-Verlag LNCS 1880, 2000.

[13] A. Savasere, E. Omiecinski and S. Navathe, "An efficient algorithm for mining association rules in large databases", *Proc. of 21st Intl. Conf. on Very Large Databases (VLDB)*, September 1995.

[14] Y. Saygin, V. Verykios and C. Clifton, "Using Unknowns to Prevent Discovery of Association Rules", *ACM SIGMOD Record*, vol. 30, no. 4, 2001.

[15] Y. Saygin, V. Verykios and A. Elmagarmid, "Privacy Preserving Association Rule Mining", *Proc. of 12th Intl. Workshop on Research Issues in Data Engineering (RIDE)*, February 2002.

[16] P. Shenoy, J. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa and D. Shah, "Turbo-charging Vertical Mining of Large Databases", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 2000.

[17] R. Srikant and R. Agrawal, "Mining Generalized Association Rules", *Proc. of 21st Intl. Conf. on Very Large Databases (VLDB)*, September 1995.

[18] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables", *Proc. of ACM SIGMOD Intl. Conference on Management of Data*, June 1996.

[19] J. Vaidya and C. Clifton, "Privacy Preserving Association Rule Mining in Vertically Partitioned Data", *Proc. of 8th ACM SIKGDD Intl. Conference on Knowledge Discovery and Data Mining (KDD)*, July 2002.

[20] Z. Zheng, R. Kohavi and L. Mason, "Real World Performance of Association Rule Algorithms", *Proc. of 7th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining (KDD)*, August 2001.