A Novel Anonymization Technique For Privacy Preserving Data Publishing

Dr. K. Sagar

Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad, India

Abstract—Privacy preserving data publishing approach provides some methods and tools for publishing useful information while preserving data privacy. Anonymization techniques are used for privacy preserving data publishing such as generalization and bucketization techniques are implemented. Generalization loses considerable amount of information and correlations between different attributes are lost. On the other hand bucketization does not provide membership disclosure protection and will not have a distinct separation between quasi-identifying characteristics and hyper sensitive attributes. In this paper, all of us present any novel technique called Slicing, which partitions the data both horizontally and vertically, within each bucket values in each column are randomly permuted to break the linking between different columns. An extension may be the notion of overlapping chopping, which duplicates a feature in more than one column. This particular releases a more attribute correlations. Here, the time required to provide the privacy is compared to show that slicing is better than generalization and bucketization.

Keywords— Publishing, Privacy, Data Security.

I. INTRODUCTION

Privacy Preserving publishing of micro data has been studied extensively in recent years. Micro data contain records each of which contains information about an individual entity, such as a person, a household, or an organization. Several micro data anonymization techniques have been proposed. The most popular ones are generalization for k-anonymity and bucketization for l-diversity. In both approaches, attributes are partitioned into three categories: 1) some attributes are identifiers that can uniquely identify an individual, such as Name or Social Security Number; 2) some attributes are Quasi Identifiers (QI), which the adversary may already know (possibly from other publicly available databases) and which, when taken together, can potentially identify an individual, e.g., Birthdate, Sex, and Zip code; 3) some attributes are Sensitive Attributes (SAs), which are unknown to the adversary and are considered sensitive, such as Disease and Salary.

In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. The two techniques differ in the next step. Generalization transforms the QI-values in each bucket into “less specific but semantically consistent” values so that tuples in the same bucket cannot be distinguished by their QI values. In bucketization, one separates the SAs from the QIs by randomly permuting the SA values in each bucket. The anonymized data consist of a set of buckets with permuted sensitive attribute values. It has been shown that generalization for k-anonymity losses considerable amount of information, especially for high-dimensional data. This is due to the following three reasons. First, generalization for k-anonymity suffers from the curse of dimensionality. In order for generalization to be effective, records in the same bucket must be close to each other so that generalizing the records would not lose too much information. However, in high dimensional data, most data points have similar distances with each other, forcing a great amount of generalization to satisfy k-anonymity even for relatively small k’s. Second, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution assumption that every value in a generalized interval/set is equally possible, as no other distribution assumption can be justified. This significantly reduces the data utility of the generalized data. Third, because each attribute is generalized separately, correlations between different attributes are lost. In order to study attribute correlations on the generalized table, the data analyst has to assume that every possible combination of attribute values is equally possible. This is an inherent problem of generalization that prevents effective analysis of attribute correlations.

While bucketization has better data utility than generalization, it has several limitations. First, bucketization does not prevent membership disclosure. Because bucketization publishes the QI values in their original forms, an adversary can find out whether an individual has a record in the published data or not.
As shown 87 percent of the individuals in the United States can be uniquely identified using only three attributes (Birthdate, Sex, and Zipcode). A microdata (e.g., census data) usually contains many other attributes besides those three attributes. This means that the membership information of most individuals can be inferred from the bucketized table.

Second, bucketization requires a clear separation between QIs and SAs. However, in many data sets, it is unclear which attributes are QIs and which are SAs. Third, by separating the sensitive attribute from the QI attributes, bucketization breaks the attribute correlations between the QIs and the SAs.

In this paper, we introduce a novel data anonymization technique called slicing to improve the current state of the art. Slicing partitions the data set both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets. Finally, within each bucket, values in each column are randomly permuted (or sorted) to break the linking between different columns.

This document is template. We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace(copy-paste) the content with your own material.

II. PREVIOUS WORK

Preserving privacy in transactional databases has been acknowledged as an important problem in the privacy literature. The data mining community has focused on hiding sensitive rules generated from transactional databases. The authors address this problem by altering the database to hide a given set of sensitive rules. This, however, does not protect individuals’ privacy and requires prior knowledge of the rules and mining model used. The work proposes publishing only rules instead of the underlying data. While this approach will be effective in some situations, in general publishing rules rather than anonymized data precludes the kind of arbitrary analysis one can do in a query session over a data set. [1]

In most previous work, the knowledge of the adversary involves an external table Te such as a voter registration list that maps QIDs to individuals. As in most previous work, we assume that each tuple in Te maps to one individual and no two tuples map to the same individual. The same is also assumed in the table T to be published.

Let us first consider the case when T and Te are mapped to the same set of individuals. Table above is an example of Te. With the notion of minimality in anonymization, the adversary reasons as follows: From the published Table above, there are 2 sensitive tuples in total. From Te, there are 2 tuples with QID=q1 and 5 tuples with QID=q2. Hence, the equivalence class for q2 in the original table must already satisfy 2-diversity, because even if both sensitive tuples have QID=q2, the proportion of sensitive values in the class for q2 is only 2/5. Since generalization has taken place, at least one equivalence class in the original table T must have violated 2-diversity, because otherwise no generalization will take place according to minimality. The adversary concludes that q1 has violated 2-diversity, and that is possible only if both tuples with QID=q1 have a disease value of HIV. The adversary therefore discovers that Andre and Kim are linked to HIV.[2]

A number of privacy models have been proposed for data anonymization, e.g., k-anonymity, l-diversity, t-closeness, and so on. A key limitation of these models is that they cannot guarantee that the sensitive attribute values of individuals are protected when the adversary has additional knowledge (called background knowledge). Background knowledge can come from diverse sources, such as well known facts, demographic information, public records, and information about specific individuals.

The hospital has the original patient table T, which contains three attributes Age, Sex, and Disease. The hospital releases a generalized table T which satisfies 3-diversity. Assume that an adversary knows Bob is a 69-year-old male whose record is in the table, the adversary can only find out that Bob is one of the first three records. However, the adversary may know the correlations between Emphysema and the non-sensitive attributes Age and Sex, e.g., “the prevalence of emphysema was appreciably higher for the 65 and older age group than the 45-64 age groups for each race-sex group” and “the prevalence was higher in males than females and in whites than blacks”.

![Table 1](image)

<table>
<thead>
<tr>
<th>Age</th>
<th>Sex</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>M</td>
<td>Emphysema</td>
</tr>
<tr>
<td>45</td>
<td>F</td>
<td>Cancer</td>
</tr>
<tr>
<td>52</td>
<td>T</td>
<td>Flue</td>
</tr>
<tr>
<td>13</td>
<td>F</td>
<td>Emphysema</td>
</tr>
<tr>
<td>17</td>
<td>F</td>
<td>Emphysema</td>
</tr>
<tr>
<td>50</td>
<td>M</td>
<td>Flue</td>
</tr>
<tr>
<td>36</td>
<td>M</td>
<td>Emphysema</td>
</tr>
<tr>
<td>52</td>
<td>M</td>
<td>Emphysema</td>
</tr>
</tbody>
</table>

(a) Original table T

<table>
<thead>
<tr>
<th>Age</th>
<th>Sex</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>M</td>
<td>Emphysema</td>
</tr>
<tr>
<td>45</td>
<td>F</td>
<td>Cancer</td>
</tr>
<tr>
<td>50</td>
<td>F</td>
<td>Flue</td>
</tr>
<tr>
<td>50</td>
<td>M</td>
<td>Flue</td>
</tr>
<tr>
<td>50</td>
<td>M</td>
<td>Emphysema</td>
</tr>
</tbody>
</table>

(b) Generalized table T

Table 1: Original table and its generalized table

135
Because Bob is a 69-year-old male, then based on the above external knowledge, the adversary can infer that Bob has a much larger probability of having Emphysema than the other two tuples in the first group.\[3\]

In the Table 2.1, the adversary knows the correlations between Emphysema and the attribute Age (and Sex). We call this correlational knowledge. In general, correlational knowledge describes the relationships between the sensitive attribute and the non-sensitive attributes, e.g., male does not have ovarian cancer. Correlational knowledge is one kind of adversarial background knowledge. [4]

III. PROPOSED SYSTEM

Slicing partitions the data set both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets. Finally, within each bucket, values in each column are randomly permuted (or sorted) to break the linking between different columns.

A. Slicing Algorithms:

Our algorithm consists of three phases: attribute partitioning, column generalization, and tuple partitioning. We now describe the three phases.

\textbf{Algorithm tuple-partition} (T, ℓ)
1. Q = \{T\}; SB = ∅.
2. while Q is not empty
3. remove the first bucket B from Q; Q = Q \− \{B\}.
4. split B into two buckets B1 and B2, as in Mondrian.
5. if diversity-check(T, Q ∪ \{B1,B2\} ∪ SB, ℓ)
6. Q = Q ∪ \{B1,B2\}.
7. else SB = SB ∪ \{B\}.
8. return SB.

\textbf{Algorithm diversity-check}(T,T_, ℓ)
1. for each tuple t ∈ T, L[t] = ∅.
2. for each bucket B in T_
3. record f(v) for each column value v in bucket B.
4. for each tuple t ∈ T
5. calculate p(t,B) and find D(t,B).
6. L[t] = L[t] ∪ \{hp(t,B),D(t,B)i}.
7. for each tuple t ∈ T
8. calculate p(t, s) for each s based on L[t].
9. if p(t, s) ≥ 1/ℓ, return false.
10. return true.

B. Data Slicing

In data slicing the attributes of tables are used for slicing and bucketized, in a generalized table each attribute value is replaced with the multiset of values in the bucket. Slicing first partitions attributes into columns. Each column contains a subset of attributes. This vertically partitions the table. Slicing also partition tuples into buckets. Each bucket contains a subset of tuples. This horizontally partitions the table. Within each bucket, values in each column are randomly permuted to break the linking between different columns. The values are randomly permuted so that the linking between the two columns within one bucket is hidden.

C. Diversity Slicing

In diversity, a tuple t can have multiple matching buckets. We now extend the above analysis to the general case and introduce the notion of ‘l-diverse slicing. Consider an adversary who knows all the Q values of t and attempts to infer t’s sensitive value from the sliced table. She or he first needs to determine which buckets t may reside in, i.e., the set of matching buckets of t. Tuple t can be in any one of its matching buckets. Let p be the probability that t is in bucket B. In the second step, the adversary computes the probability that t takes a sensitive value s. p is calculated using the law of total probability. Specifically, let p’ be the probability that t takes sensitive value s given that t is in bucket B, then according to the law of total probability, the probability are calculated. Given a tuple t and a sliced bucket B, the probability that t is in B depends on the fraction of t’s column values that match the column values in B. If some column value of t does not appear in the corresponding column of B, it is certain that t is not in B. In general, bucket B can potentially match j tuples, where Bj is the number of tuples in B. Without additional knowledge, one has to assume that the column values are independent; therefore, each of the Bj tuples is equally likely to be an original tuple. The probability that t is in B depends on the fraction of the Bj tuples that match t.

D. Correlation Measure

In this module we use two widely used measures of association are Pearson correlation coefficient and mean-square contingency coefficient. Pearson correlation coefficient is used for measuring correlations between two continuous attributes while mean-square contingency coefficient is a chi-square measure of correlation between two categorical attributes. We choose to use the mean-square contingency coefficient because most of our attributes are categorical.
Given two attributes A1 and A2 with different domains, their domain sizes are thus d1 and d2, respectively. The mean-square contingency coefficient between A1 and A2 is defined. Here, f_i and f_j are the fraction of occurrences of v1i and v2j in the data, respectively. f_{ij} is the fraction of co-occurrences of v1i and v2j in the data. Therefore, f_i and f_j are the marginal totals. For continuous attributes, we first apply discretization to partition the domain of a continuous attribute into intervals and then treat the collection of interval values as a discrete domain. Discretization has been frequently used for decision tree classification, summarization, and frequent item set mining. We use equal-width discretization, which partitions an attribute domain into some k equal-sized intervals.

E. Attribute Clustering

In this module after having computed the correlations for each pair of attributes, we use clustering to partition attributes into columns. In this algorithm, each attribute is a point in the clustering space. The distance between two attributes in the clustering space is defined which is in between of 0 and 1. Two attributes that are strongly correlated will have a smaller distance between the corresponding data points in our clustering space. We choose the k-medoid method for the following reasons. First, many existing clustering algorithms e.g., k-means requires the calculation of the “centroids.” But there is no notion of “centroids” in our setting where each attribute forms a data point in the clustering space. Second, k-medoid method is very robust to the existence of outliers i.e., data points that are very far away from the rest of data points. Third, the order in which the data points are examined does not affect the clusters computed from the k-medoid method. A well-known k-medoid algorithm PAM (Partition around Medoids) starts by an arbitrary selection of k data points as the initial medoids. In each subsequent step, PAM chooses one medoid point and one nonmedoid point and swaps them as long as the cost of clustering decreases. Here, the clustering cost is measured as the sum of the cost of each cluster, which is in turn measured as the sum of the distance from each data point in the cluster to the medoid point of the cluster. The data points in our clustering space are attributes, rather than tuples in the micro data.

F. Tuple Partitioning

In the tuple partitioning phase, tuples are partitioned into buckets. We modify the Mondrian algorithm for tuple partition.

Unlike Mondrian k-anonymity, no generalization is applied to the tuples; we use Mondrian for the purpose of partitioning tuples into buckets. The algorithm maintains two data structures: a queue of buckets Q and a set of sliced buckets SB. Initially, Q contains only one bucket which includes all tuples and SB is empty. In each iteration, the algorithm removes a bucket from Q and splits the bucket into two buckets. If the sliced table after the split satisfies ‘l-diversity, then the algorithm puts the two buckets at the end of the queue Q. Otherwise, we cannot split the bucket anymore and the algorithm puts the bucket into SB. When Q becomes empty, we have computed the sliced table. The set of sliced buckets is SB. The main part of the tuple-partition algorithm is to check whether a sliced table satisfies ‘l-diversity. For each tuple t, the algorithm maintains a list of statistics about t’s matching buckets. Each element in the list L contains statistics about one matching bucket B: the matching probability p and the distribution of candidate sensitive values. The algorithm first takes one scan of each bucket B to record the frequency f of each column value v in bucket B. Then, the algorithm takes one scan of each tuple t in the table T to find out all tuples that match B and record their matching probability p and the distribution of candidate sensitive values which are added to the list L. At the end we have obtained, for each tuple t, the list of statistics about its matching buckets. A final scan of the tuples in T will compute the p values based on the law of total probability described.

IV. RESULTS

The concept of this paper is implemented and different results are shown below. The proposed paper is implemented in .Net technology on a Pentium-IV PC with minimum 20 GB hard-disk and 1GB RAM. The propose paper’s concepts shows efficient results and has been efficiently tested on different Datasets.

Fig.4.1 Proposed System performing Anonymity on the dataset.
In the Fig 5, it shows that the time consumed for each and every technique such as anonymity, diversity and slicing techniques. y-axis shows the time in milli seconds and x-axis shows the different anonymizations techniques. In the Fig 6, it shows that the time consumed for Slicing and Enhanced Slicing. Y-axis shows the time in milli seconds and X-axis shows the slicing and enhanced slicing.

V. CONCLUSION

This paper presents a new approach called slicing to privacy preserving micro data publishing. Slicing overcomes the limitations of generalization and bucketization and preserves better utility while protecting against privacy threats. We illustrate how to use slicing to prevent attribute disclosure and membership disclosure. Our experiments show that slicing preserves better data utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute. The general methodology proposed by this work is that: before anonymizing the data, one can analyze the data characteristics and use these characteristics in data anonymization. The rationale is that one can design better data anonymization techniques when we know the data better.

REFERENCES


