4.1 Introduction

Experts say that data mining in the wrong hands will end up in destruction. The main threat of data mining is to privacy and security of data residing in large data stores [32, 33, 44, 48, 50, 75-78, 83, 84]. Some of the information considered as private and secret can be brought out with advanced data mining tools. It is a real concern of people working in the field of database technology. Different research efforts are under way to deal with this problem of preserving security and privacy.

Sensitive information contained in a database can be extracted with the help of non-sensitive information. This is called the inference problem [50, 75, 85]. Different concepts have been proposed to handle the inference problem. The process of modifying the transactional database to hide some sensitive information is called sanitization. By sanitizing the original transactional database, the sensitive information can be hidden. In the sanitization process, selective transactions are retrieved and modified before handing over the database to a third party.

Modification of transaction involves removing an item from a transaction or adding an element to the transaction. In some cases, transactions are either added to or removed from the database as suggested in [75]. The modified database is called sanitized database or released database.

The technique proposed and discussed in this chapter has been published as a paper [a].
Several approaches have been proposed to hide sensitive data with good accuracy. The efficiency of a privacy-preserving algorithm is measured based on 1) the time taken to hide the data, 2) the number of new rules introduced as a result of the hiding process, and 3) the number of legitimate rules lost or cannot to be extracted from the released database.

The task of locating a transaction for sanitization from a massive amount of data is not a trivial process and it is certainly a time consuming one. In many research efforts, the highly time consuming process of retrieving the transactional database is not taken into account efficiently. This work proposes a method to hide sensitive association rule in a faster manner. It takes the advantages of Frequent Pattern Growth Tree to identify and retrieve the generating transactions directly from the transactional database without exhaustive search. An array is used to keep track of the identifiers of the required transactions for sanitization.

This chapter is organized as follows: Section 4.2 discusses the existing related works. The proposed approach is discussed in Section 4.3. The performance of the algorithms is discussed in Section 4.4 and Section 4.5 presents a summary.

4.2. Related Work

Typically, the hiding process involves two steps: a) generation of association rules and b) hiding of association rules. Association Rule Mining is one of the functionalities of data mining. The process of producing association rules consists of i) the frequent itemsets generation and ii) the rule generation. Frequent itemsets generation is a tedious process because it performs the time consuming task of the generation of candidates and pruning of unnecessary itemsets.
A number of Apriori derivative algorithms are available to improve the efficiency of association rule mining [1, 8, 9, 20, 62, 82]. The algorithms for mining association rule differ in the approaches they use. Bottom-up [8, 9], top-down or combination of both the approaches [20] is used for generating frequent itemsets. The approaches differ in terms of how the transactions of database are scanned. A few algorithms generate frequent itemsets without the costly candidate generation. The Frequent Pattern growth Tree (FPT) generates the frequent itemsets without candidate generation[1, 65]. The tree is constructed based on the occurrence of the frequent items. Each transaction in the database is reordered before adding the items of the transaction to the FPT. It proposes a novel frequent pattern structure. The entire database is compressed into a small data structure and eliminates the unnecessary scans over the database. It reduces the number of scans over the database to only two passes irrespective of the size of the database. Fig. 4.1 depicts the structure of FPT for transactional database of Table 4.1 [1].

<table>
<thead>
<tr>
<th>TID</th>
<th>List of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>11,12,15</td>
</tr>
<tr>
<td>T200</td>
<td>12,14</td>
</tr>
<tr>
<td>T300</td>
<td>12,13</td>
</tr>
<tr>
<td>T400</td>
<td>11,12,14</td>
</tr>
<tr>
<td>T500</td>
<td>11,13</td>
</tr>
<tr>
<td>T600</td>
<td>12,13</td>
</tr>
<tr>
<td>T700</td>
<td>11,13</td>
</tr>
<tr>
<td>T800</td>
<td>11,12,13,15</td>
</tr>
<tr>
<td>T900</td>
<td>11,12,13</td>
</tr>
</tbody>
</table>

Many techniques have been suggested to hide sensitive data in large databases. Clifton et al. [75] suggest different possible solutions to protect the sensitive data such as limiting access to the data, fuzzing the data and augmenting the data. In [78], a sanitization matrix is proposed according to the sensitive patterns and non-sensitive patterns.
In [77], two algorithms, namely, 1) Round-robin algorithm and 2) Random algorithm are proposed. Oliveira et al. [76] propose an algorithm called Sliding-window algorithm which is introduced to hide data. It applies a look-ahead procedure to verify whether that transaction has been selected as sensitive transaction for other rules to be hidden. In [33], the authors discuss an Expectation Maximization (EM) algorithm for distribution-reconstruction, which is more effective than the currently available methods in terms of level of information loss. Specifically, it proposes that the EM algorithm converges to the maximum likelihood estimate of the original distribution based on the perturbed data. In [50], various assumptions are followed to hide association rules. Five algorithms have been proposed. They hide rules by considering the antecedent part of the rule or consequent part of the rule or the entire rule to sanitize the generating transactions of the transactional database. While the above mentioned approaches are moderately successful in hiding sensitive data, they are computationally intensive and time consuming. Hence, for practical applications, there is a great need for faster algorithms and approaches for sensitive information hiding.
4.2.1 Problem formulation

Let $D$ be the database of transactions.

Let $R$ be the set of association rules mined from $D$.

Let $R_H$ be the subset of $R$, the set of rules to be hidden.

The problem is to transform $D$ into $D'$ in such a way that the rules in $R_H$ cannot be mined from $D'$ which is the released database.

4.3. The Proposed Approach

This thesis work proposes a method to hide sensitive association rule in a faster manner. It takes advantage of the Frequent Pattern Growth Tree to construct an array called Generating-Transaction Identifiers, which helps to retrieve the necessary transactions directly. Since the proposed approach attempts to hide rules quickly, any sanitization method which directly deals at the transaction level for hiding rule can be used with the proposed approach to reduce the time complexity.

Each node of the FPT contains the name of the element and its corresponding number of occurrences. In the modified FPT, besides those two values, a list is maintained for keeping the transaction identifiers. The list is called Transaction Locator. The Transaction Locators provide the identifiers of the generating transactions to facilitate direct retrieval of the respective transactions from the database without an exhaustive search. It will be kept in the secondary storage when the size is large. Hence, the time to retrieve the generating transactions is only about the total number of identifiers available in the respective transaction locators. In order to reduce the time for traversing the tree, an array called the Generating Transaction Count (GTC) is created. The GTC array is of maximum size $|L_i| \times |L_j|$, where $|L_i|$ is the number of frequent items. The rows are assigned according to the frequency of items. The item with
maximum occurrence in the database occupies the first row and so on and so forth. The columns in the array specify the different levels and the positions of a particular item in the FPT and the transactions of the database respectively. Using the array, either the number of partially or fully generating transactions can be found without traversing the tree.

In the proposed approach, the sorting of generating transactions is not done since the array is already constructed based on the occurrence of these items. It is easier to refer to the respective transactions with the help of the GTC array.

For example, the Modified FPT (MFPT) is shown in Fig. 4.2 and the equivalent Generating Transaction Count array is shown in Table 4.2 for the data in Table 4.1.

![Fig. 4.2 Modified FPT (MFPT).](image)
Table 4.2 Generating Transaction Count array (GTC)

<table>
<thead>
<tr>
<th>Items</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I3</td>
<td></td>
<td></td>
<td>T300, T500, T600, T700</td>
<td></td>
<td>T900</td>
</tr>
<tr>
<td>I4</td>
<td></td>
<td>T200</td>
<td></td>
<td></td>
<td>T400</td>
</tr>
<tr>
<td>I5</td>
<td></td>
<td></td>
<td></td>
<td>T100</td>
<td>T800</td>
</tr>
</tbody>
</table>

For example, the transaction T100 in the Table 4.1 contains 3 frequent items. The item with minimum occurrence among those three items is I5. Hence, the transaction identifier T100 is stored in the GTC array in the fifth row and third column position. The remaining generating transactions identifiers are stored onto the array in the same manner. The following sequence of steps gives an overview of the approach.

**General algorithm:**

*Input:* Minimum support, Minimum confidence, Database D,

Database size |D|, a set of rules to be hidden R_H

*Output:* Released Database D'

1. Construct a GTC array for the database D.
2. Using the GTC array,
   i) for each rule to hide, find out which transactions are to be retrieved for sanitization.
   ii) using the transaction identifiers available in GTC, retrieve the respective generating transactions from the database.
3. Apply the sanitization process
The process of hiding association rules rapidly is depicted in Fig. 4.3. The proposed method can be applied for any sanitization algorithm, which openly deals with items in a transaction. For comparison, we have considered the algorithms 1.a and 2.c in [50]. The algorithm 1.a sanitizes a set of transactions that partially supports the antecedent of the rule to be hidden. The algorithm 2.c sanitizes the transactions that fully support the large itemsets to be hidden in a round-robin fashion. A rule can have many generating transactions; finding all those transactions requires $|D|$ scans. The proposed approach deals with this problem and minimizes the amount of time required.

The GTC array is referred to based on the concept of the algorithm used. The algorithm 1.a in [50] requires $O(|R_H| \times |A_D|)$ time to hide the rules in $R_H$. For each rule to be hidden, first it scans the database to collect the generating transactions of the antecedent. The generating transactions are selected for sanitization by the number of antecedent items they contain. The transaction with a larger number of items of antecedent is considered for sanitization first. As a second step, it counts and sorts the generating transactions depending on the number of items of antecedents they contain. Then it chooses the transaction with the largest number of antecedent items for sanitization. The algorithm 1.a hides the sensitive rule by increasing the support of the antecedent of the rule. Thereby it reduces the confidence of the rule below the minimum confidence threshold.
The algorithm 2.6 requires $O(|R_H| \cdot |A_D|)$ time for hiding a set of $L_H$ large itemsets. In the proposed algorithm, the generating transaction can be located directly without exhaustive search with the help of GTC array. This requires a time of only $O(2|D|)$.

In this approach, an attempt is made to improve the processing time of the hiding algorithms. The sanitization methods recently prescribed by Verykios et al. are followed without any modification [50]. Thus the other two factors, rules lost and new rules introduced, which measure the performance of any hiding algorithms, are the same as in those algorithms. These two factors are not changed.

1. In the 1st scan, identify frequent items and their supports.
2. In the 2nd scan, construct the GTC array

   In a cell of the GTC array, we assign transaction identifiers having the original level as row and different occurring levels as columns. (Size of the GTC array is $|L_I| \cdot |L_I|$)

3. For each rule $r$ in $R_H$
   3.1 Determine $K$, the number of generating transactions to be modified.
   3.2 Retrieve the items of the antecedent one by one from the last.
   3.3 For each item extracted from the antecedent, retrieve the cell value from the GTC array with the row as the original value of the item and column as $(lr-i)$, $i$ varies from positional $i$ to $(lr-1)$. It gives the list of transaction identifiers.
   3.4 Retrieve and sanitize the transactions from the dataset directly using the transactions identifier.

   $K=K-$ (number of transaction modified)

3.5 Repeat 3.3 until $K \leq 0$.

Fig. 4.4 Proposed algorithm.
The algorithm in Fig. 4.4 explains our approach. Here,

\[ K \] - the number of generating transactions to be modified
\[ R \] - the set of association rules mined from \( D \)
\[ R_H \] - the set of rules to be hidden
\[ l_r \] - antecedent of the rule \( r \), to be hidden

After constructing the GTC array, the array is checked for getting the identifiers of generating transactions with maximum number of antecedent elements of the rule to be hidden. For instance, to hide a rule \( \text{antecedent} \rightarrow \text{consequent} \), according to algorithm 1.a, transactions with \( n-1 \) elements of the antecedent are processed for hiding where \( n \) is the total number of elements in the antecedent. Then transactions with \( n-2 \) elements of the antecedent are processed and if some more transactions are still required, it will consider the transactions with \( n-3 \) elements and so on.

The GTC array in the proposed approach directly gives the transactions with the corresponding number of supporting elements. If an identifier is at the \( n^{th} \) row and an entry at the \( n^{th} \) level in the array, it means that the MFPT is having a path containing all the first \( n \) frequent items. Suppose the item is at some other lower level in the tree it means that the particular path does not have one or more frequent items whose support is greater than the current item under consideration. For example, if an item is at the fifth row as well as at the fifth column, it means that the path containing all the first five frequent items are in the sequence. Suppose the item is at level 1, it means that the first four frequent items are not in that path.
4.4. Performance Evaluation

This Section gives details of the experimental evaluation of the performance of the proposed algorithm.

4.4.1 Experimental setup

As in previous chapters, the experiment was conducted on a Pentium IV computer with a CPU clock rate of 2.8 GHz, 256 MB of main memory running Windows Operating System. The datasets for the experiment were generated synthetically as suggested by [9]. Three databases T5.I3.D5K, T5.I3.D10K, and T5.I3.D50K were generated. In each dataset the average transaction length (ATL) and the maximal frequent itemset size were kept at 5 and 3 respectively. The sizes of the three datasets, |D|, are 5K, 20K, and 50K respectively.

Input for the association rule hiding algorithms were either a set of frequent itemsets or a set of association rules. Hence to generate the required inputs, the Apriori and Pincer algorithms were used. Other effective association rule mining algorithms can also be used.

4.4.2 Time complexity of the proposed algorithm

The process of hiding an association rule requires exhaustive scan over the transactional database to identify the necessary generating transactions of a rule. The time taken for this process will be more when the number of rules to be hidden is large.

Due to the hiding process, there is a chance to lose some of the valuable rules already existing in the transactional database. New rules may also be introduced. These are the side effects of any hiding process. Thus, these three points, namely time taken for hiding a set of association rules, rules lost and new
rules introduced are considered as parameters to measure the performance of any hiding algorithm. This work attempts to reduce the time complexity alone. As this work uses the hiding technique given by Verykios et al., the side effects of our approach on the database are also the same as those algorithms of [50].

In an association rule hiding algorithm, the time required to identify and sanitize the necessary generating transactions of a rule is central to the hiding process. The time for revisiting the database for each and every rule to be hidden is a major time consuming process. The proposed approach suggests an effective method suitable for real-time applications. It tries to minimize the time complexity substantially with the help of the GTC array.

The time complexity of our approach is compared with all the five algorithms given in [50]. The comparison is depicted as graphs in Fig. 4.5.

![Graph for Time Requirement](image)

(a) Algorithm 1a
for ATL=5, ||I||=10

(b) Algorithm 1.b

for ATL=5, ||I||=10

(c) Algorithm 2.a

(d) Algorithm 2.b
A theoretical expression for the time complexity is derived below. The time required to hide a set of rules $R_{II}$ by the proposed approach is

$$2 * |D| + \sum_{i=1}^{R_{II}} K_{ij}.$$ 

Since the approach requires two scans of the database, the time required to construct the GTC array is $O(2 * |D|)$ which includes the generation of frequent items, L1 and filling of cell values of the array. The variable $K_{ij}$ is the number of generating transactions that needs to be retrieved and sanitized to hide a rule $r_i$.
of \( R_H \). It may be noted that \( \sum_{i=1}^{k_H} K_y \) values will always be very much smaller than \( |D| \). For example \( \sum_{i=1}^{k_H} K_y \) may be a few tens while \( |D| \) may be a few tens of thousands. The GTC array simplifies this process by providing the respective transaction identifiers, TID to enable direct access from the transactional database. Therefore the exhaustive scan over the transactional database for locating the generating transactions of each rule is prevented. In other approaches the database is repeatedly and completely scanned for the same purpose.

Of the five algorithms 1.a, 1.b, 2.a, 2.b, and 2.c of Vassilios S Verykios et al., the algorithm 1.a takes \( O(|R_H| A_D + C) \), the algorithms 1.b and 2.a each take \( O(|R_H| A_D) \), and the algorithms 2.b and 2.c each take \( O(|D| |L_H|) \) to perform the hiding process[50].

Compare the time complexity of the proposed approach with the time complexity of algorithm 1.a of Verykios et al [50, 86]. The time complexity of the proposed approach is

\[
= O(2|D| + \sum_{i=1}^{k_H} K_y )
\]

where, \( j \) is the number of transaction to be sanitized for hiding a rule, \( r_i \).

\[
\approx O(|D|)
\]

since, \( \sum_{i=1}^{k_H} K_y \) is very much negligible compared to \( 2|D| \).
The first term of Equation (4.1), $2|D|$ is for constructing the GTC array.

The second term $\sum_{i=1}^{g_{ui}} K_i$ is the number of generating transactions to be retrieved from the transactional database for sanitization, which varies from 1 to $|R_h|$.

The time complexity of algorithm 1.a of Verykios et al. [50, 86] is

$$= O(|R_h| \cdot A_D + C)$$

$$\approx O(|R_h| \cdot A_D)$$

$$\approx O(|R_h| \cdot |D| \cdot ATL)$$

(4.4)

since $C$ is negligible compared to the other terms.

Thus the ratio of complexity of this work compared to the time complexity of algorithm 1.a of Verykios et al. [50, 86] is

$$2|D| : |R_h| \cdot |D| \cdot ATL$$

i.e.,

$$1 : \frac{|R_h| \cdot ATL}{2}$$

(4.5)

Proceeding along the similar lines the ratio for the other algorithms are

$$1 : \frac{|L_H|}{2}$$

(4.6)

for algorithms 1.b and 2.a and

$$1 : \frac{|L_H|}{2}$$

(4.7)

for algorithms 2.b and 2.c.

From perusal of Equations (4.5), (4.6), and (4.7), it is seen that the time complexity of the proposed technique is less - ranging from $\frac{|L_H|}{2}$ times to $\frac{|R_h| \cdot ATL}{2}$. For the experimentation conducted here,
\(|R_H| = 3\)
\(|L_{\cdot H}| = 5\)
\(ATL = 5\)
\(|L_{\cdot H}|/2 = 2.5\)
\(\left(\frac{|R_H| \cdot ATL}{2}\right) = 7.5\)

Therefore the reduction in time is 2.5 times to 7.5 times.

4.4.3 Discussion and limitations

Here some of the limitations of the proposed approach are discussed. When a database is very large, the GTC array may not fit in the available main memory. To accommodate into the main memory, the GTC array may be segmented either vertically or horizontally into different sub-tables. Based on the need, the necessary segments or sub-tables can be brought to the main memory from the secondary storage. To further minimize the space complexity the GTC array can be replaced with some other data structure.

The proposed approach is suitable for real time environment where time is a major criterion. It reduces the time required to hide a set of association rules by a factor of 2.5 times compared to other hiding algorithms.

The space complexity

In case the sizes of transaction locators are big, they are placed in the secondary storage as buckets. On demand, they are brought into the main memory for sanitization. For instance, to hide a rule \(r\) of \(R_H\), the bucket which contains the identifiers of the generating transactions of \(r\) is identified by traversing the MFPT. Then that particular bucket is transferred from secondary memory to main memory to locate the generating transactions for sanitization.
The space complexity is much lower because each bucket contains selective transactions identifiers only. Moreover if a database contains \( n \) attributes, storing the identifier of each transactional record is not going to increase the space complexity. The buckets are chosen based on the rule to be hidden and at a single point of time there will be either only one or part of one bucket residing in the main memory. Therefore the size of a particular bucket is normally much smaller than the size of the transactional database i.e., \(| \text{Bucket}_i | < |D|, \ i=1,2,\ldots,m. \) Where \( m \) is the number of buckets.

To hide any information on a transactional database it is obvious that the selective transactions should be subjected to some alterations. There are many works proposed to handle the side effects arising out of the hiding process.

4.5 Summary

In this work, a novel approach is presented to hide association rules on a large transactional database in a faster manner. The proposed approach reduces the time for hiding association rules by a factor of 2.5 times to about 7.5 times. It is found that the time to revisit the transactional database has been reduced with the help of the data structure, Generating Transaction Count (GTC). This work can be incorporated to any hiding algorithms to enhance their response time. The space complexity of the algorithm has also been drastically reduced. The effectiveness of the proposed approach is demonstrated using synthetic data. As future work, the technique given in this chapter can be suitably extended for hiding association rules in multiple data sources. This chapter has discussed a technique for hiding rules rapidly while the next chapter discusses another better technique for the same problem.