AN EFFICIENT ALGORITHM FOR ANONYMIZATION OF SET-VALUED DATA AND REPRESENTATION USING FP-TREE

B.K.Tripathy[1],A.Jayaram Reddy[2],G.V.Manusha[2] and G.S.Mohisin[2]

[1]School of Computing Science and Engineering
VIT University,Vellore-632014,TN,India
[2]School of Information Technology and Engineering
VIT University,Vellore-632014,TN,India
{tripathybk,ajayaramreddy,gshahidmohisin2008}@vit.ac.in,
manushagv@gmail.com

ABSTRACT

Data anonymization techniques enable publication of detailed information, while providing the privacy of sensitive information in the data against a variety of attacks. Anonymized data describes a set of possible worlds that include the original data. Generalization and suppression have been the most commonly used techniques for achieving anonymization. Some algorithms to protect privacy in the publication of set-valued data were developed by Terrovitis et al .,[16]. The concept of k-anonymity was introduced by Samarati and Sweeny [15], so that every tuple has at least (k-1) tuples identical with it. This concept was modified in [16] in order to introduce k^m *-anonymity, to limit the effects of the data dimensionality. This approach depends upon generalisation instead of suppression. To handle this problem two heuristic algorithms; namely the DA-algorithm and the AA-algorithm were developed by them.These alogorithms provide near optimal solutions in many cases.In this paper,we improve DA such that undesirable duplicates are not generated andwe can display the anonymized data even in the FP-Tree way.We illustrate through suitable examples,the efficiency of our proposed algorithm.*

KEYWORDS

Direct anonymization, Apriori anonymization, FP-tree, k-anonymization, k^m*-anonymization*

1. INTRODUCTION

In [10]supermarket transactions were considered as the motivating example to describe the requirement of anonymization of set valued data. Suppose an adversary finds some of the items purchased by a customer. If the supermarket database is published later, even after removing the personal identities, there is a chance that the database contains only one transaction containing the items seen by him. Then the adversary can easily identify the other items purchased by the particular customer and get useful information out of it. Identifying the transaction details in this way is known as re-identification. Inorder to preserve the data from being re-identified data can be k-anonymized. According to Sweeney [16] the data is k-anonymized if the information for each person contained in the release cannot be distinguished from at least k -1 individuals whose information also appears in the release. So, we need to transform the original database D to the anonymized database D'. Even after the data is k-anonymized the data cannot be completely protected from being re-identification. However, in this approach the set of attributes in a database are divided into two broad categories. These are the sensitive attributes and the non-sensitive attributes. But in [15], the attributes are considered to be alike and are not divided into such categories with the view that any subset of the set of attributes can be sensitive attributes and the others as non-sensitive attributes depending upon the application. So, assuming that an adversary has knowledge about at the most m items and we want to prevent him from distinguishing the transaction from a set of k published transactions in the database. Equivalently, for any set of m or less items, there should be at least k transactions, which contain this set, in the published database D'. So, making use of this concept a k^m -anonymization model was developed in [10] and algorithms were developed to deal with such type of set-valued data. A subset of items in a transaction play the role of quasi-identifier.By which the data can be re-identified by linking techniques.The items in the transaction can be anonymized through various anonymization approaches. We describe some of these below. In Suppression information is removed from the data. For example, the gender attribute can be removed from a patient database. In Generalization the information is generalized from more specific to less specific or can be coarsened into sets. For example, in an employee database DOB can be generalized form (dd-mm-yy) format to only year.Mostly Generalization or Suppression is used for the anonymization of data. It can be noted that when suppression is used for anonymization then there is greater loss of information than when Generalization is used. So, the Generalization technique is used for transactional database or "market basket" data analysis.

Three algorithms were introduced in [16]to achieve k^m -anonymization. However, the problem in these algorithms is the generation or redundant transactions while generating the additional tuples to achieve anonymization. In this paper we improve upon the two algorithms (DA and AA algorithms) in [16] by adding several steps so that the number of transactions generated is the exact number required.

2. LITERATURE SURVEY

As mentioned in the introduction, one of the earliest attempts to anonymisation of databases is the introduction of the notion of k-anonymity by Samarati [14] and Sweeny [15]. A table is k-anonymised if each record is indistinguishable from at least k-1 other records with respect to a set of quasi-identifier (QI) attributes. The QIs are than generalised and the records with identical QI values thus form an anonymised group. The process of transforming a database table D into a table D' after anonymisation is called recoding. It has been established in [12] that the problem of optimal k-anonymity for multidimensional QI is NP-hard, under both generalisation and suppression models. Several approximate algorithms those minimise the number of suppressed values have been obtained. These are in [12] with a bound of O(k.logk), in [2] with a bound of O(k), and with a bound of O(logk) in [3]. In [9] an algorithm called Incognito is proposed, which uses dynamic programming approach to find an optimal solution. The problem here is the concept of full-domain recoding, which requires that all values in a dimension must be mapped to the same level of hierarchy. Inspired by Incognito, Terrovitis et al [16] proposed three algorithms, the optimal anonymization (OA) algorithm and two heuristic algorithms called the direct anonymization (DA) algorithm and the apriori anonymization (AA) algorithm. Here, the full-domain recoding is not assumed. Also, in k-anonymity the set of QI attributes are known beforehand. However, in case of [16], since any set of m items (which corresponds to the attributes) can be used by the adversary, no QI set can be predetermined.

The concept of k-anonymity has been extended to the notion of l-diversity by Machanavajjhala et al [11] and further to the notion of t-closeness by Li et al [10]. Several approaches to solve the l-diversity problem efficiently are provided in [5, 21, 22]. Many fast l-diversity algorithms have been developed by Tripathy et al [17, 18, 19, 20]. Also, some of the l-diversity algorithms developed by Tripathy et al take care of uncertainty in databases by using rough set methods to achieve l-diversity. Some of these algorithms deal with hybrid databases.

3. CONCEPTS AND EXISTING ALGORITHMS

In this section we introduce some concepts to be used in the paper and also introduce the three existing algorithms proposed in [16] along with explanations.

3.1 Generalization

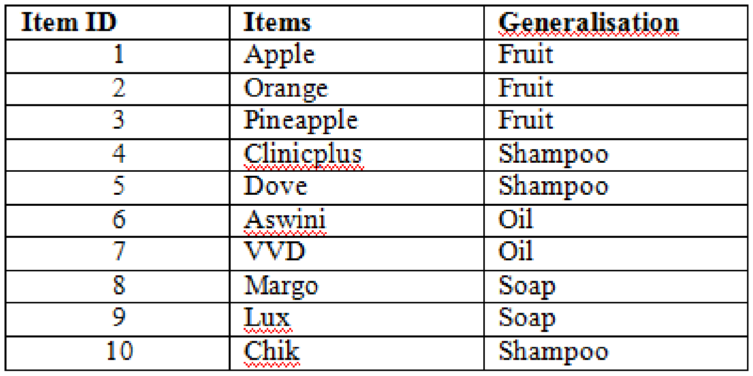
The data generalization concept explored from data mining as a way to hide detailed(more specific) information, rather than discover trends and patterns.Data mining is frequently described as the process of extracting valid, authentic and actionable information from large databases. In other words, data mining derives patterns and trends that exist in data. These patterns and trends can be collected together and defined as a mining model.It ensures that sensitive data is replaced with realistic but not real data.

k^m –anonymization model is proposed for transactional databases, where m is the atmost number of items known to an adversary.For an set of m or less items there should be atleast k transactions which contain m itemset in the published database D', to prevent an adversary from distinguishing the transaction from a set of k transactions in the database.But here there is no fixed,well defined set of quasi-identifier for the sensitive data.A subset of items in a transaction can act as quasi-identifier for the sensitive ones or vice versa.To solve the k^m –anonymization problem for a transactional database, generalization is in use.If original database D does not satisfy the k^m -anonymity then it is transformed to D' by replacing items with their generalized ones.Here in supermarket database while entering the item is provided with its respective generalization. Generalization replaces intial attribute with generalized attribute.

For example consider T={orange,goodday,mango,timepass},in this{orange,mango} can be generalized to Fruits and {goodday,timepass}can be generalized to biscuits and total transaction to {Fruits, biscuits} .

Example 3.1: In the table below we present a set of items along with their generalisations, which form the components of any transaction in a super market.

Table 1. Database of items and their generalizations



Item ID	Items	Generalisation
1	Apple	Fruit
2	Orange	Fruit
3	Pineapple	Fruit
4	Clinicplus	Shampoo
5	Dove	Shampoo
6	Aswini	Oil
7	VVD	Oil
8	Margo	Soap
9	Lux	Soap
10	Chik	Shampoo

Table 2. Database of some transactions and Transformed Transactions

Item ID	Items in Cart	Transformed Transactions
1	Apple,Aswini,ClincPlus	Fruit,Shampoo,Oil,
2	Dove,ClincPlus,Aswini	Oil,Shampoo
3	Apple,Aswini,ClincPlus,Orange,Pineapple	Fruit,Shampoo,Oil
4	ClinicPlus,Apple,Dove,Pineapple	Fruit,Shampoo
5	Aswini,ClincPlus,Orange	Fruit,Shampoo,Oil,
6	Apple,Orange,Pineapple	Fruit
7	Aswini,ClinicPlus,Orange	Fruit,Shampoo,Oil

3.2 Count tree

To find whether generalization applied provides K^m-anonymity,it is to count efficiently the support of all combinations of m-items that appear in the database.To avoid scanning the database each and every time generalization has to be checked.To acheive these two goals a datastructure was constructed which keep track of not only all combinations of m items from the generalized database but also it must know how each generalized value effects the database.The support value of each combination of all items in the transactions is calculated.Inorder to keep track of the support of all the transactions a count tree-data structure was constructed.

To count the support of all these combinations and to store them the count-tree is used, based on the count tree algorithm.The tree assumes an order of items and their generalizations, based on their frequencies(supports)in D.To compute this order, a database scan is required.The support of each itemset with upto m items can be computed by following a corresponding path in the tree and using the support value of the corresponding node[16].Count-tree follows the apriori principle which states that the support of an item set is always less than or equal to the support of its subsets.

Here in the database the items which are present and not present in the transaction are represented 1's and 0's respectively.Based on this the frequent item sets are generated. A frequent item set is an item set whose number of occurrences is above a threshold. For each combination of items in the transaction the support value is calculated and is displayed. The items which are having the support value less than the minimum support value then those items are neglected.

Based on the count tree two anonymization techniques can be performed i.e. Direct Anonymization (DA) and Apriori -based anonymization (AA).

Definition 3.2.1 Support: The support or utility or prevalence for an association rule X=>Y is the percentage of transactions in the database that contains both X and Y.

$$\text{Support}(X \rightarrow Y) = \frac{\text{No.of tuples containing both X and Y}}{\text{Total no.of tuples}} = P(X \cap Y).$$

Table 3. Algorithm for Creation of the tree for k^m anonymity([16])

```
Populate Tree (D, tree, m)
1: For all t in D do for each transaction
2:     expand t with the supported generalized items
3:     For all combination of c ≤m items in the expanded t do
4:         If $\neg\exists$ $i, j \in c$ such that i generalizes j then
5:             insert c in tree
6:             increase the support counter of the final node
```

3.2.2 Output of Count Tree

The following table display the output of the count tree for all the subset of items whose support value is greater than the minimum support

Table 4 The following table provides an example of the output of the above algorithm.

Transaction Table		
Item	**No. of Occurrences**	**Support**
Apple	1	0.5
PineApple	1	0.3571428571428715
Dove	1	0.21428571428571427
Margo	1	0.14285714285714285
Lux	1	0.07142857142857142
VVD	1	0.14285714285714285
Orange	1	0.5
Aswini	1	0.5714285714285714
ClincPlus	1	0.7142857142857143
Chik	1	0.07142857142857142

3.3 Optimal Anonymization(OA)

To find the optimal cut i.e. no generalization that satisfies k^m -anonymity and has the least information loss, we can examine systematically the generalizations in the cut hierarchy, in a bottom-up, breadth first fashion. Initially the cut C_{ng} which corresponds to no generalization is put to queue Q. While Q is not empty, we remove first cut from C from it and examine whether it satisfies k^m -anonymity [16].If it satisfies then it becomes a candidate solution. If it does not satisfy k^m -anonymity, its immediate ancestors in the hierarchy, which do not have a descendant cut that satisfies k^m -anonymity are added to the queue.

Table 5 Optimal Anonymization Algorithm

```
OA(D,I,K,m)
1: $C_{opt}$ :=null; $C_{opt}$.cost := $\infty$        // initialize $C_{opt}$
 2:      add $C_{ng}$ to an initially empty queue Q
3:       While (Q is not empty) do
4:        pop next cut C from Q
5: if  C does not provide $K^m$-anonymity to D then
6: for all immediate ancestors $C_{ans}$ of C do
7: if $C_{ans}$ does not appear in H then
8: push $C_{ans}$ to Q
9: else                          // C provide $K^m$-anonymity to
10: for all immediate ancestors $C_{ans}$ of C do
11: add $C_{ans}$ to H
12: if $C_{ans}$ in Q then
13: delete $C_{ans}$ from Q
14: if $C \cdot cost < C_{opt}.cost$ then
15: $C_{opt}$ := C
16: return $C_{opt}$ .
```

3.4 Direct Anonymization

Direct anonymization is to scan the count tree once and then use the generalized combinations to find a solution that optimizes problem of re-identification. Optimal Anonymization method is based on pre-computation of complete count tree for sets consisting of up to m item sets [16].Direct anonymization scans the tree to detect m-sized paths that have support less than K.For each such paths,it generates all possible generalization.In this direct anonymization, the database is scanned and a count tree is constructed.Once the count tree has been created; direct anonymization initializes the output generalization C_{out} as bottommost cut of the lattice (i.e. no generalization).Then performs preorder traversal of count tree. Based on the initial support count neglect the item sets whose support count is less than the initial. For every node encountered, if the item corresponding to that node has already been generalized in C_{out} , direct anonymization backtracks as all complete m-sized paths passing from there correspond to itemsets that will not appear in the generalized database based on C_{out} (and therefore their supports need not be checked).

Table 6. Direct Anonymization Algorithm

```
DA (D, I, k, m)
1.   Scan D and create count-tree
2.   Initialize $C_{out}$
3.   For each node v in preorder count-tree tranversal do
4. If the item of v has been generalized in $C_{out}$ then 5. backtrack
 6. If v is a leaf node and v.count<k then
 7.  J:= itemset corresponding to v
  8. find generalization of items in J that make J k-anonymous
 9. merge   generalization rules with $C_{out}$
10. backtrack to longest prefix of path J,where no item has been generalized in $C_{out}$
11.Return $C_{out}$ .
```

3.5 Apriori Algorithm

Apriori is a classical algorithm for learning association rules. Association rule mining is finding the frequent patterns, associations, correlations or casual structures among sets of items or objects in transaction databases, relational databases, and other information repositories. Apriori is designed to operate on databases containing transactions (for example, collection of items bought by customers). Apriori uses a “bottom up” approach where frequent subsets are extended one item at a time (a step known as a candidate generation and groups of candidates are tested against the data).The algorithm terminates when no further successful extensions are found.It uses a level wise search, where K-itemsets (an itemsets that contain K items is a K-itemset) are used to explore (K+1) itemsets, to mine frequent itemsets from transactional database for association rules. First, the set of frequent 1-itemsets is found. This set is denoted by L1, which is used to find L2, the set of frequent 2-itemsets, which are used to find L3 and so on until no more frequent K-item sets can be found.

Table 7 . Apriori based Anonymization Algorithm

```
AA (D, I, k, m)
1: Initialize $c_{out}$
2: For i: = 1 to m do                         //for each item set length
3:   initialize a new count-tree
4:   For all t ϵ D do                         //scan D
5:          extend t according to $C_{out}$
6:          add all i-subsets of extended t to count-tree
7:      run DA on count-tree for $m = i$ and update $C_{out}$
```

3.6 FP Growth Algorithm

Algorithm:FP_growth[23]. Mine frequent itemsets using an fp-tree by pattern fragment growth.

Input:

- D,a transaction database;
- Min_sup,the minimum support count threshold.
-

Output: the complete set of frequent threshold.

Method:

1. The FP tree is constructed in the following steps.
a) scan the transaction database D once.Collect F,the set of frequent items,and their support counts.Sort F in support count descending order as L,the llist of frequent items.
b) create the root of an Fp-tree and label it as “null” for each transaction Trans in D do the following.

- Select and sort the frequent items in Trans according to the order of L.let the sorted
- Frequent itemset list in Trans be p[P],where p is the first element and P is the remaining list.
- Call insert_tree(p[P],T)which is performed as follows.If T has a child N such that N.item.name=p.item.name,then increment N’s count by 1;else create a new node N,
- And let its count be 1,its parent link be linked to T,and its node –link to the nodes with
- The same item-name via the node-link structure.if P is non empty,call insert_tree(P,N) Recursively.

2. The fp-tree is mined by calling **FP_ growth**(FP_tree,null)which is implemented as follows.

4 THE PROPOSED ALGORITHMS

As mentioned earlier we improve the two algorithms DA and AA in order to reduce the generation of redundant transactions which makes the further analysis of the output efficient and simpler. First we present the improved DA algorithm below. We have added new steps from 12 to 25 in the existing algorithm.

4.1 Improved Direct Anonymization Algorithm

```
DA(D,I,k,m)
1.scan D and create count-tree
2.initialize $C_{out}$
3.for each node v in preorder count-tree tranversal do
4.    if the item of v has been generalized in $C_{out}$ then
5.            backtrack
6.    if v is a leaf node and v.count<k then
7.            J:= itemset corresponding to v
8.            find generalization of items in J that make J k-anonymous
9.            merge generalization rules with $C_{out}$
10.           backtrack to longest prefix of path J,wherein no item
                  Has been generalized in $C_{out}$
```

```
11.Return $C_{out}$
12. **for** *i* :=1 to $C_{out}$ **do**
13.      initialize count=0
14.      scan each transactions in $C_{out}$
15.     Seperate each item in a transaction and store it in p
16.     Increment count
17. **for** j:=1 to count **do**
18.     **for** all g belongs $C_{out}$ **do**
19.             compare each item of p with that of $C_{out}$
20.              **if** all items of i equal to cout
21.              Increment the r
22.      **if** $k_a$ equal to r then backtrack  to i
23          **else if** r greater than $k_a$ then get the index position of the similar          transactions
24.          make them NULL until $k_a$ equal to r
25         **else** update the transactions in database
```

4.2 Improved Apriori Based Anonymization Algorithm

AA (D, *I*, *k*, m)

```
1: initialize cout
2: **for**  *i* := 1 to m **do**                    for each itemset length
3:       initialize a new count-tree
4:       **for all** t belongs D **do**                    scan D
5:               extend t according to $C_{out}$
6:               add all i-subsets of extended t to count-tree
7: **for** *i* :=1 to $C_{out}$ **do**
8:      initialize count=0
9:      scan each transactions in $C_{out}$
10:     Seperate each item in a transaction and store it in p
11:     Increment count
12: **for** j:=1 to count **do**
13:     **for** all g belongs $C_{out}$ **do**
14:             compare each item of p with that of $C_{out}$
15:              **if** all items of i equal to cout
16:              Increment the r
17:      **if** $k_a$ equal to r then backtrack  to i
18:          **else if** r greater than $k_a$ then get the index position of the similar
transactions
19          make them NULL until $k_a$ equal to r
20.         **else** update the transactions in database
```

4.3 Algorithm:Procedure FP_growth(Tree,α)

Table 8. Improved Apriori Based Anonymization

```
1: if Tree contains a single path P then
2:  for each combination (denoted a β)of the nodes in the path P
3:  generate pattern β∪α with support_count=minimum support count of nodes in β;
4: else for each $a_i$ in the header of Tree{
5: generate pattern β=$a_i$ ∪α with support_count=$a_i$.support_count;
6: construct β's conditional pattern base and then β's conditional FP-tree $tree_β$.
7: if $tree_β$ $\neq^{0}$ then
8: call FP_growth($Tree_β$,β);
10: Call FP_growth($Tree_β$,ß);Where $a_i$=item in the transaction.
```

5 COMPARISON OF THE ALGORITHMS

Here we representing the anonymized data using FP Tree.We are reducing the number of the duplicate transactions that are generated by the previous algorithm ie DA.

5.1 Experimental Analysis

We consider here a Supermarket database where there is a provision to add an item,add an transaction as well as to view the transactions.Here a database is created with limited number of transactions let us consider 10 transactions.

5.1.1 Database

We are considering the following items based on the items transactions are created.

Table 9. Items in the database and their generalizations

Item No	Items	Generalization
1	Apple	Fruit
2	Orange	Fruit
3	Pineapple	Fruit
4	Clinicplus	Shampoo
5	Dove	Shampoo
6	Aswini	Oil
7	VVD	Oil
8	Margo	Soap
9	Lux	Soap
10	Chik	Shampoo

5.1.2 Transactions in Database

We are considering a small database which contains 15 transactions.

Table 10. Transactions in the database

Transaction ID	Transaction Items
1	Apple,Aswini,ClincPlus
2	Dove,ClincPlus,Aswini
3	Apple,Aswini,ClincPlus,Orange,Pineapple
4	ClinicPlus,Apple,Dove,Pineapple
5	Aswini,ClincPlus,Orange
6	Apple,Orange,Pineapple
7	Aswini,ClinicPlus,Orange
8	Apple,Aswini,ClincPlus,Dove,Margo, Orange,PineApple,VVD
9	Aswini,ClincPlus,VVD
10	ClincPlus,Orange,Margo
11	ClinicPlus,Dove
12	Aswini
13	PineApple,Chik,Lux
14	Apple,Dove,Margo
15	ClinicPlus,Aswini

5.1.3 Output Using Earlier Algorithms

We implemented the following algorithms in NETBEANS IDE using JAVA SWINGS with backend technology as SQL SERVER. the algorithms are Outputs of both similar.

5.1.3.1 Direct Anonymization and Apriori Algorithm

The following is the output generated using the earlier Direct Anonymization and Apriori Anonymization algorithms.

Table 11. output of the direct anonymization

Transaction id	Generalized transactions
1	fruit,shampoo,oil,
2	oil,shampoo
3	fruit,shampoo,oil
4	fruit,shampoo
5	fruit,shampoo,oil,
6	Fruit
7	fruit,shampoo,oil
8	fruit,soap,shampoo,oil
9	shampoo,oil
10	fruit,soap,shampoo
11	Oil
12	fruit,soap,shampoo,
13	fruit,soap,shampoo,
14	oil,shampoo,
15	fruit,shampoo,oil
16	fruit,soap,shampoo,oil,
17	fruit,shampoo,oil
18	fruit,soap,shampoo,
19	fruit,

20	oil,
21	shampoo,oil,
22	fruit,shampoo,
23	oil,shampoo,
24	fruit,soap,shampoo,
25	oil,shampoo,
26	fruit,soap,shampoo,

5.1.4 Output Based on New Approach

In this new approach, the number of duplicate transactions is decreased because the algorithm checks for all the conditions inorder to achieve k^m-anonymity completely. As said earlier the outputs are same for both algorithms.

5.1.4.1 Direct Anonymization and Apriori Based Anonymization

The following is the output generated using the Improved algorithms for the Direct and Apriori based anonymization.

Table 12. output of direct anonymization

Transaction id	Generalized transactions
1	fruit,shampoo,oil,
2	oil,shampoo
3	fruit,shampoo
4	Fruit
5	fruit,soap,shampoo,oil,
6	fruit,soap,shampoo,
7	fruit,soap,shampoo,
8	oil,shampoo
9	oil,
10	oil,
11	fruit,shampoo,oil,
12	fruit,soap,shampoo,oil,
13	Fruit
14	fruit,shampoo

5.1.4.2 FP_Growth Algorithm

Let us consider a supermarket database of six transactions and perform the FP_GROWTH.This procedure will be applicable to even large database also.

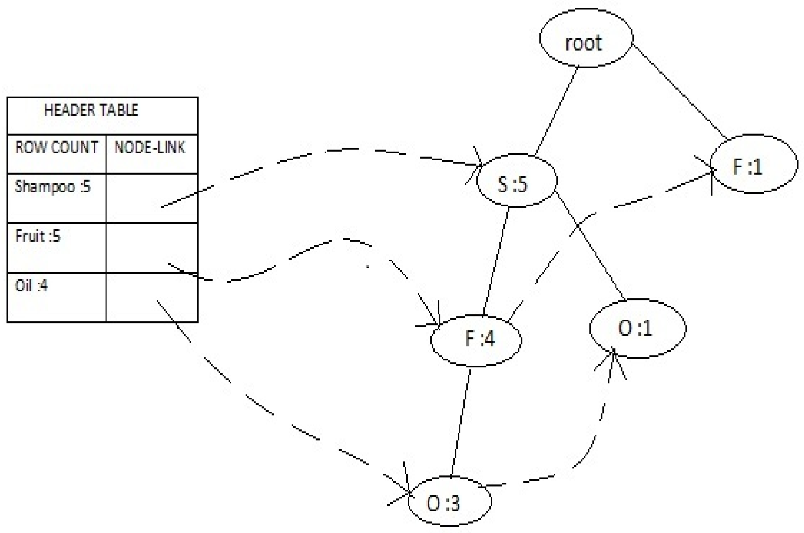
Table 13. Transactions in the database

Transaction ID	Transaction Items
1	Apple,Aswini,ClincPlus
2	Dove,ClincPlus,Aswini
3	Apple,Aswini,ClincPlus,Orange,Pineapple
4	ClinicPlus,Apple,Dove,Pineapple
5	Aswini,ClincPlus,Orange
6	Apple,Orange,Pineapple

Table 14. generalized transactions

Transaction ID	Generalized transactions
1	fruit,shampoo,oil,
2	oil,shampoo
3	fruit,shampoo,oil
4	fruit,shampoo
5	fruit,shampoo,oil,
6	Fruit

5.1.4.2.3 Output of the FP-TREE



Output of the FP-TREE

6 CONCLUSION

In this paper we have improved the k^m-anonymity algorithms developed in [16] for anonymization of set-valued data. The algorithms in [10] generate many redundant transactions and it is very inconvenient for further analysis. The improved algorithms generate the exact number of tuples required for the generalisation.The anonymized data is used to construct the FP_tree which reduces the number of comparisions and provides an easy way to find the count of the generalized items in the transactions. This reduces the size of the output table considerably and makes it simpler for further analysis. We provided an example to illustrate the efficiency of the new algorithms over the existing algorithms. Also, theoretically we have computed the extent of improvement in the results.

7 REFERENCES

[1] Aggarwal, G., Feder, G., Kenthapadi, K., Khuller, S., Panigrahy, R., Thomas, D. and Zhu, A.: Achieving Anonymity via Clustering, In Proc. of ACM PODS, (2006), pp.153-162.

[2] Aggarwal, G., Feder, G., Kenthapadi,R., Motwani, R., Panigrahy, D., Thomas, and Zhu, A.: Approximation Algorithms for k-Anonymity, .Journal of Privacy Technology, (2005).

[3] Atzori, M., Bonchi, F., Giannotti, F., and Pedreschi, D.: Anonymity Preserving Pattern Discovery, VLDB Journal, accepted for publication, (2008).

[4] Bayardo, R. J. and Agrawal, R.: Data Privacy through Optimal k-Anonymization, In Proc. of ICDE, (2005), pp.217-228.

[5] Ghinita, G., Karras, F P., Kalnis, P., and Mamoulis, N.: Fast Data Anonymization with Low Information Loss, In VLDB, (2007), pp.758-769.

[6] Ghinita, G., Tao, Y., and Kalnis, P.: On the Anonymization of Sparse High-Dimensional Data, In Proceedings of ICDE, (2008).

[7] Han, J., Pei, J., and Yin, Y.: Mining frequent patterns without candidate generation, In Proc. of ACM SIGMOD, (2000), pp.1-12.

[8] Iyengar, V.S.: Transforming Data to Satisfy Privacy Constraints, In Proceedings of SIGKDD, (2002), pp.279-288.

[9] LeFevre, K., DeWitt, D.J. and Ramakrishnan, R.: Incognito: Efficient Full-domain k-anonymity, In Proceedings of ACM SIGMOD, (2005), pp. 49-60.

[10] Li, N., Li, T. And Venktasubramanian: t-closeness Privacy Beyond k-anonymity and l-diversity, In Proceedings of ICDE, (2007), pp. 106-115.

[11] Machanavajjhala, A., Gehrke, J., Kifer, D. And Venkitasubramaniam: l-diversity: Privacy Beyond k-Anonymity, In Proceedings of ICDE, (2006).

[12] Meyerson, A. And Williams, R.: On the Complexity of Optimal k-Anonymity, In Proceedings of ACM PODS, (2004), pp. 223-228.

[13] Park, H. And Shim, K.: Approximate algorithms for k-Anonymity, In Proceedings of the ACM SIGMOD, (2007), pp. 67 -78.

[14] Samarati, P.: Protecting Respondents' Identities in Microdata Release, IEEE TKDE, 13(6), (2001), pp. 1010 -1027.

[15] Sweeney, L.: K-Anonymity: A Model for Protecting Privacy, In International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 10(5), (2002), pp.557-570.

[16] Terrovitis, M., Mamoulis, N. and Kalnis, P.: Privacy Preserving Anonymization of Set-Valued Data, PVLDB'08, Auckland, New Zeland, (2008), pp.115-125.

[17] Tripathy, B.K., Devineni, H., Jayasri, K.J. and Bhargava, M.: An Efficient Clustering Algorithm for l-diversity, Proceedings of the International conference on Advances and Emerging Trends in Computing Technologies,ICAET10, SRM university, June 21-24 ,(2010),pp. 76 - 81.

[18] Tripathy, B.K., Panda, G.K. and Kumaran, K.: A Rough Set Approach to develop an efficient l-diversity Algorithm based on Clustering, Proc. of the 2nd IIMA international conference on "Advanced Data Analysis, Business Analytics and Intelligence", 8 -9, Jan (2011), p.34.

[19] Tripathy, B.K., Panda, G.K. and Kumaran, K.: A Fast l - Diversity Anonymisation Algorithm, Proc. Of the third International Conference on Computer Modelling and Simulation (ICCMS 2011), Mumbai, 7-9 January, (2011), pp.V2-648- 652.

[20] Tripathy, B.K., Maity, A., Ranajit, B. and Chowdhuri, D.: A fast p-sensitive l-diversity Anonymisation algorithm, Proceedings of the RAICS IEEE conference, Kerala, Sept.21-23, (2011), pp.741 – 744.

[21] Xiao, X. And Tao, Y.: Anatomy: Simple and Effective Privacy Preservation, In Proceedings of VLDB, (2006), pp.139-150.

[22] Zhang, Q., Koudas, N., Srivastava, D. And Yu, T.: Aggregate Query Answering on Anonymised Tables, In Proceedings of ICDE, (2007), pp.116-125.

[23] Jiawei Han.,Micheline Kamber.,Jian pei :Data mining :Concepts and techniques text book

AUTHORS:

B.K Tripathy is a senior professor in the school of computing sciences and engineering, VIT University,at Vellore, India, has published more than 155 technical papers in international journals/ proceedings of international conferences/ edited book chapters of reputed publications like Springer and guided 12 students for PhD. so far. He is having more than 30 years of teaching experience. Dr.Tripathy is a member of international professional associations like IEEE, ACM, IRSS, CSI, IMS, OITS, OMS, IACSIT, IST and is a reviewer of around 21 international journals which include IEEE, World Scientific, Springer and Science Direct publications. Also, he is in the editorial board of at least 11 international journals. His current research interest includes Fuzzy sets and systems, Rough sets and knowledge engineering, Granular computing, soft computing, Data clustering, Database anonymization techniques, bag theory, list theory and social network analysis.

A.Jayaram Reddy is a Assistant rofessor(senior) in the school of Information Technology and Engineering ,VIT University, at Vellore, India, He is having more than 8 years of teaching experience.His area of interest includes Aritificial Intelligence,Knowledge Engineering,DBMS.

G.V.Manusha is a MS-Software Engineering student in the School of Information Technology and Engineering,VIT University at Vellore,India.She had done project s using various technologies in different domains.Her area of interest are Datamining,DBMS and Software testing.

G. Shahid Mohisin is a MS-Software Engineering student in the School of Information Technology and Engineering, VIT University at vellore,India.He has done various mini projects and a semester project using various technologies of different domains.His area of interest are Objec t Oriented Programming in C++,DBMS,DataMining and Software testing.