Personalized Privacy Preserving Publication of Transactional Datasets Using Concept Learning

S. Ram Prasad Reddy, Kvsvn Raju, and V. Valli Kumari

Abstract—**In this paper we study the problem of protecting privacy in the publication of transactional data. Consider a collection of transactional data that contains detailed information about items bought together by individuals. Even after removing all personal characteristics of the buyer, which can serve as links to his identity, the publication of such data is still subject to privacy attacks from adversaries who have partial knowledge about the set. Unlike previous works, we do not distinguish data as sensitive and non-sensitive, but we consider them both as potential quasi-identifiers and potential sensitive data, depending on the point of view of the adversary. We define a new version of the anonymity guarantee using concept learning. Our anonymization model relies on generalization using concept hierarchy and concept learning. The proposed algorithms are experimentally evaluated using real world datasets.**

Index Terms—**Privacy preserving, hashing, anonymity, concept learning, transactional datasets.**

I. Introduction

Recently, mining of databases has attracted a growing amount of attention in database communities due to its wide applicability in retail industries and for improving marketing strategy. As pointed out in [1], the progress in barcode technology has made it possible for retail organizations to collect and store massive amounts of sales data. A record in such a data typically consists of the transaction date, the items bought in the transaction, and possibly also customer-id if such a transaction is made via the use of a credit card or any kind of customer card. It is noted that analysis of past transaction data can provide valuable information on customer buying behavior, and thus improve the quality of business decisions. It is essential to collect a sufficient amount of sales data before we can draw any meaningful conclusion from them. As a result, the amount of these sales data tends to be huge.

We consider the problem of publishing transactional data, while preserving the privacy of individuals associated to them. Consider a transactional database D, which stores information about items purchased at a drug store by various customers (patients). We observe that the direct publication of D may result in unveiling the identity of the person associated with a particular transaction, if the adversary has some partial knowledge about a subset of items purchased by that person. Besides this, the major concern is that the adversary may also identify the customer's illness from the identified transaction. For example, assume that Thomas went to the drug store on a particular day and purchased medicines {riboflavin, beplex forte, tiniba and roxid}. Assume that some of the medicines (e.g., riboflavin, beplex forte) purchased by Thomas were observed by another customer, James, who has come to buy medicines at that point of time. Unfortunately, James happens to be neighbor of Thomas. Thomas would not like James to find out other medicines that he bought because if the remaining medicines are known there is a likelihood of James identifying the illness he is suffering from. However, if the drug store decides to publish its transactions and there is only one transaction containing {riboflavin, beplex forte}, James can infer that this transaction corresponds to Thomas and he can find out his complete list of medicines and thereby identify the disease as glossitis. This scenario not only reveals other items in the transaction but also gives a scope to the adversary to identify the illness.

This motivating example stresses the need to transform the original transactional database D to a database D' before publication. In practice, we expect the adversary to have only partial knowledge about the transactions otherwise there would be little sensitive information to hide.

In summary, this paper proposes a solution using concept learning for publishing transactional data sets. It also ensures that publishing transactional data does not provide any scope for the adversary to gain the identity of the individual. Experiments are conducted on adult dataset to verify our technique. The results show that our solution is very promising in real world applications.

The rest of this paper is organized as follows. Section 2 surveys the previous works. Section 3 explains the system architecture. Section 4 comprises of proposed model. In Section 5, we present an analysis of our approach. Empirical study is given in section 6. Section 7 concludes our work and points out some possible future directions.

II. Related Work

Anonymity for relational data has received considerable attention due to the need of several organizations to publish microdata without revealing the identity of individual records. Even if the identifying attributes like name is removed, an attacker may be able to associate records with specific persons using combinations of other attributes (e.g., Postal Code; Gender; birth-date), called quasi-identifiers (QID). Two techniques to preserve privacy are generalization and

Manuscript received August 9, revised September 7, 2012

S. R. P. Reddy is with Computer Science Engineering Department, Vignan's Institute of Engineering for Women, Visakhpatnam-530049, Andhra Pradesh, India (e-mail: reddysadi@ gmail.com).

K. Raju was with Andhra University, Visakhapatnam-530003, Andhra Pradesh, India. He is now with the R&D, ANITS as Director, Visakhapatnam, India (e-mail: kvsvn.raju@gmail.com).

V. V. Kumari is with Computer Science and Systems Engineering Department, Andhra University Visakhapatnam-530003, Andhra Pradesh, India (e-mail: vallikumari@gmail.com).

10.7763/IJMLC.2012.V2.225

suppression [5]. Generalization replaces their actual QID values with more general ones. Suppression excludes some QID attributes or entire records (known as outliers) from the microdata. *k*-anonymity is a technique that prevents joining attacks by generalizing/suppressing portions of the released microdata so that no individual can be uniquely distinguished from a group of size k [4], [7].

The privacy preserving transformation of the microdata is referred to as recoding. Two models exist: in global recoding [2], a particular detailed value must be mapped to the same generalized value in all records. Local recoding, on the other hand, allows the same detailed value to be mapped to different generalized values in each anonymized group.

For any anonymity mechanism, it is desirable to define some notion of minimality. As to our model, the notion of minimal full-domain generalization was defined [3], [4] using the distance vector of the domain generalization. Informally, this definition says that a full-domain generalized private table (PT) is minimal if PT is *k*-anonymous, and the height of the resulting generalization is less than or equal to that of any other *k*-anonymous full-domain generalization.

Yabo Xu et al [6] model the power of attackers by the maximum size of public itemsets that may be acquired as prior knowledge, and proposed a novel privacy notion called “coherence” suitable for transactional databases.

Manolis Terrovitis et al [8] proposed the *k*-anonymization problem of set-valued data. They defined the novel concept of k^m-anonymity for such data and analyzed the space of possible solutions.

Our problem is different, since any combination of *m* items (which correspond to attributes) can be used by the attacker as a quasi-identifier. In this paper, we focus on specific local-recoding model of *k*-anonymity. Our objective is to find the minimal *k*-anonymous generalization (table) under the definition of minimality defined by Samarati [4]. By introducing concept learning, we provide a new approach to generate minimal *k*-anonymous tables for transactional datasets.

Definition 1 (k-anonymity): Given a set of QID attributes $Q_1,\ldots\ldots,Q_d$, released data D' is said to be *k*-anonymous with respect to $Q_1,\ldots\ldots,Q_d$ if each unique tuple in the projection of *D*' on $Q_1,\ldots\ldots,Q_d$ occurs at least k times.

Definition 2 (Quasi-Identifier): A set of non-sensitive attributes $\{Q_1,\ldots,Q_w\}$ of a table is called a quasi-identifier if these attributes can be linked with external data to uniquely identify at least one individual in the general population.

Definition 3 (Sensitive Attribute): The attributes that should not be disclosed directly to the public or may be disclosed after disassociating its value with an individual’s other information.

III. System Architecture

The transactional database D consists of set of transactions and would be updated frequently. We consider the users’ consent to publish the transactions. To satisfy this constraint an extra attribute is appended to the existing database specifying users’ consent (UC). The UC is a binary value. Binary value ‘0’ indicates that the user is not interested in revealing the information. If the user is not interested in getting the transaction released, the transaction is totally suppressed. If the user gives willingness to publish the transaction i.e. *UC*=1, the transaction is anonymized and then published. The transactions are distributed into various equivalence classes using hashing. After this, each equivalence class is checked for the minimality threshold (*k*). If *k* is satisfied for all the equivalence classes then each bucket of transactions is anonymized using concept learning. If *k* is not satisfied for any of the equivalence class then the total transaction dataset is rehashed and the procedure continues.

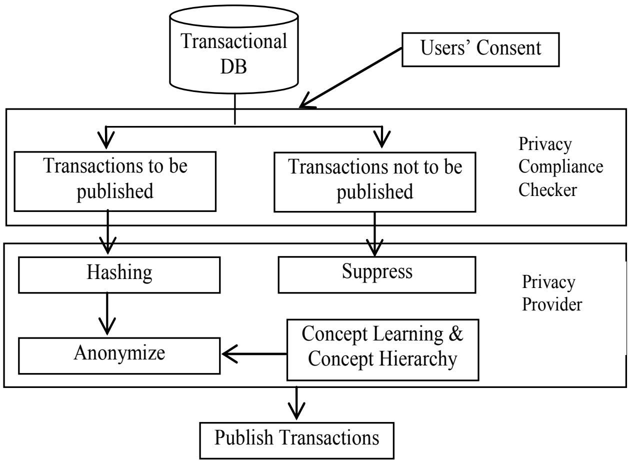
Transactional DB
Users’ Consent
Transactions to be published
Transactions not to be published
Privacy Compliance Checker
Hashing
Suppress
Privacy Provider
Anonymize
Concept Learning & Concept Hierarchy
Publish Transactions

Fig. 1. System architecture

IV. Proposed Model

Let *D* be a transactional database containing |*D*| transactions. Let $I = \{I_1, \ldots, I_{|m|}\}$ be a set of items. Each transaction $T \in D$ is a set of items such that $T \subseteq I$. I is the domain of possible items that can appear in a transaction. Each transaction is associated with an identifier TID. A set of items is referred to as itemset. An itemset that contains m items is an m-itemset. We assume that the database provides answers to subset queries, i.e. queries of the form $\{T \mid (Q_s \subseteq T) \Lambda (T \in D)\}$, where Q_s is a set of items from ‘I’ provided by the user. The number of query items provided by the user in Q_s defines the size of the query. We define a database as *k*-anonymous as follows:

Problem Definition: Given a database *D*, no adversary that has a subset of background knowledge of items of a transaction $T \in D$ can use these items to identify less than k tuples from *D*.

In other words, any subset query of size *m* or less, issued by the adversary should return either nothing or more than *k* answers. Note that queries with zero answers are also secure, since they correspond to background information that cannot be associated to any transaction.

In general the structure of the transactional database may be in two different ways: (*i*) Horizontal data format and (*ii*) Vertical data format. In this paper, transactions of database are stored in the horizontal format. In horizontal data format, the data is represented as tid-itemset format, where tid is the transaction identifier and itemset is the set of items the transaction contains.

TABLE I: THE HORIZONTAL DATA FORMAT OF THE TRANSACTIONAL DATABASE D

Tid	Itemset
T_1	$\{I_1, J_1, J_2\}$
T_2	$\{I_2, J_1\}$
T_3	$\{I_2, J_1, J_2\}$
T_4	$\{I_1, I_2, J_2\}$

The items in the transactions are hashed based on the value produced by the hash function as $h(t) = \sum_{i=1}^{m} \sigma(I_i)$ mod n.

So, the transaction T_1 is stored in 2nd location (bucket). Similarly, the process is repeated for other transactions in the transactional database. Table II indicates the support counts of each item. The process of hashing is illustrated in Table III. From the Table III, we understand that transactions T_1 and T_4 are stored in bucket 2 and transactions T_2 and T_3 are stored in bucket 0.

Definition 4 (Support Count): The number of transactions in which a particular itemset exists, gives the support count, denoted by σ.

TABLE II: SUPPORT COUNTS

Item	Support count(σ)
I_1	2
I_2	3
J_1	3
J_2	3

TABLE III: APPLYING HASH FUNCTION

Tid	Itemset	Sum of the support counts(SSC)	SSC mod 3
T_1	$\{I_1, J_1, J_2\}$	2+3+3=8	2
T_2	$\{I_2, J_1\}$	3+3=6	0
T_3	$\{I_2, J_1, J_2\}$	3+3+3=9	0
T_4	$\{I_1, I_2, J_2\}$	2+3+3=8	2

A. Concept Hierarchy

Much research effort has been found on using concept hierarchy towards databases management, text categorizations, natural language processing and so on. Algorithms on discover associations between different items from levels of taxonomy (which is represented in hierarchies) was introduced in 1995 as the mining approach for generalized association rules [11]. As an example, a keyword suggestion approach based on concept hierarchy has been proposed [12] to facilitate user's web search. A data mining system has been proposed to induce the classification rules using concept hierarchy [13]. Concept Hierarchy can reflect the concepts and relationships of a given knowledge domain. Such hierarchies are useful towards generalization and specialization. Fig. 2 shows sample hierarchy for the above set of items in Table I.

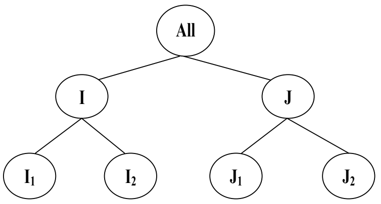

Fig. 2. Concept hierarchy

B. Anonymization Using Concept Learning

Concept learning is a machine learning concept to generalize the items optimally. Each bucket of transactions is considered independently and generalized. Initially transaction1 is considered as it is and this would be the current resultant transaction. Next transaction is considered and is matched with the resultant transaction. If it exactly matches the resultant transaction remains as it is. If it does not match then the set of items that are present in the current transaction and not present in the previous transaction are added to the previous transaction. This is nothing but simply performing a union (U) operation on the two transactions. We obtain the resultant transaction. The procedure is repeated until a single transaction is obtained such that each transaction in the bucket is contained within the resultant transaction. The resultant transaction is now generalized using the concept hierarchy. The anonymized version for data in Table I is given in Table IV.

```
Algorithm 1 – Anonymizer
Input:   Bucket of transactions $B_j$
         Concept hierarchy
Output: Anonymized transactions
Initialize the result to the most specific transaction
      $R = \{T_i\}$
for each transaction $T_{i+1}$ in $B_j$
for each item $I_k$ in $T_{i+1}$
    if item $I_k$ in $T_{i+1}$ is contained in R
        do nothing
            else if item $I_k$ is a sibling to one of the
                  items in R generalize both items
                  using the concept hierarchy
            else
                add item $I_k$ to R
        end if
    end for
end for
Publish R for each transaction in $B_j$
```

TABLE IV: ANONYMIZED TRANSACTIONS

Tid	Itemset
T_1	$\{I, J\}$
T_2	$\{I_2, J\}$
T_3	{I_2, J}
T_4	$\{I, J\}$

```
Algorithm 2 – Personalized Privacy Preserving
Input:   Transaction database D
            Number of buckets (Equivalence Classes)N
            Hash Function H
            Minimal Anonymity k
Output: Buckets $B_0, B_1, B_2, \ldots, B_N$ of anonymized transactions
1: for each distinct items $I_x$ in T
2: compute $\sigma(I_x)$
3: for each transaction $T_i$ in D
4: $j \leftarrow H(T_i)$ { j is the index of the bucket}
5: $B_j \leftarrow T_i$
6: end for
7: for each bucket $B_j$
8: if $|B_j| < k$
9:     $N = N + 1$
10:        go to step 3
11:    end if
12: end for
13: for each bucket $B_j$ in the hash table
14:    call Anonymizer($B_j$)
15: end for
```

C. Algorithm for Personalized Privacy Publishing

The algorithm starts by considering the transaction set, number of buckets, hash function and minimal anonymity as input parameters and distributes the data uniformly into all buckets satisfying the minimal anonymity 'k'. The support count of each distinct item in the transaction set is computed. This support count is used as the weight of the item. The hash function is applied to each transaction independently by computing the residue of the sum of the support counts of all the items in the transaction (SSC) divided by the number of buckets (N). The residue generated is the bucket address. Each bucket of data is called a hash equivalence class. Depending on the residue produced, the transaction is mapped to the respective bucket. The process is repeated for all the transactions. Now, each bucket is checked for minimal anonymity (*k*). If all the buckets satisfy minimal anonymity then the transactions in each equivalence class is anonymized using the concept hierarchy and is published.

V. Analysis

A. Privacy

The privacy metric depends on the level of anonymization. The strength of privacy is given by the distribution of transactions into equivalence classes and the anonymization of transactions in each equivalence class. The privacy gain of an item x in transaction T is defined in (1).

$$PG_{(x)} = \begin{Bmatrix} 0, \left|N_{(x)}\right| = 1 \\ \left|N_{(x)}\right| / \left|CH\right|, Otherwise \end{Bmatrix} \tag{1}$$

where N_x is the node of the item generalization hierarchy where x is generalized. $|N_x|$ is the number of leaves under N_x and |CH| is the number of leaves in the concept hierarchy. If the item being published is at the leaf level it implies that privacy is not being provided and the privacy gain is zero. On the other hand, if the published item is a non-leaf node then it denotes that the item is being secured and some amount of privacy is being provided as specified in (1).

The privacy gain for the whole database is measured by computing the privacy gain of generalized item in each transaction. The usage of local recoding prevents the generalization of an item to be non-uniform and varies for each equivalence class. The privacy gain of each transaction *T* ($PG_{(T)}$), of an item x in the entire database ($PG_{(x\in D)}$) and of the entire database *D* ($PG_{(D)}$) is given in (2), (3) and (4) respectively.

$$PG_{(T)} = \frac{\sum_{x\in T} PG_{(x)}}{|T|} \tag{2}$$

$$PG_{(x\in D)} = \frac{\sum_{x\in D} PG_{(x)}}{|\sigma(x)|} \tag{3}$$

$$PG_{(D)} = \frac{\sum_{T\in D} PG_{(T)}}{|D|} \tag{4}$$

where $|T|$ is the size of the transaction, $\sigma(x)$ is the total number of occurrences of item x in the database *D* and $|D|$ is the size of the transactional database to be published. The value lies in between 0 and 1 where '0' implies no privacy gain and '1' implies complete privacy gain.

B. Information Loss

All privacy-preserving transformations cause information loss, which must be minimized in order to maintain the ability to extract meaningful information from the published data. A variety of information loss metrics have been proposed. The Classification Metric (CM) [10] is suitable when the purpose of the anonymized data is to train a classifier, whereas the Discernibility Metric (DM) [9] measures the cardinality of the anonymized groups. More accurate is the Generalized Loss Metric [10] and the similar Normalized Certainty Penalty (NCP) [5]. In the case of categorical attributes NCP is defined with respect to the hierarchy. On the similar guidelines, the information loss is defined in (5). Let x be an item in T. Then:

$$IL_{(x)} = \begin{Bmatrix} 0, \left|N_{(x)}\right| = 1 \\ \left|N_{(x)}\right| / \left|CH\right|, Otherwise \end{Bmatrix} \tag{5}$$

where N_x is the node of the item generalization hierarchy where I is generalized. $|N_x|$ and $|CH|$ are the number of leaves under N_x and in the entire hierarchy, respectively. Intuitively, the IL tries to capture the degree of generalization of each item, by considering the ratio of the total items in the domain that are indistinguishable from it. For example, in the hierarchy of Fig. 2, if I_1 is generalized to I in a transaction *T*, the information loss $IL_{(I1)}$ is 2/4.

The information loss for the whole database weighs the information loss of each generalized item in each transaction. Since local recoding is used, the generalization of an item will not be common for the total database. It varies from equivalence class to equivalence class. The information loss of a particular generalization ranges from 0 to 1 and can be easily measured. '0' implies no information loss '1' implies complete information loss. The information loss of transaction T ($IL_{(T)}$), of an item in the entire database ($IL_{(x\in D)}$) and of the entire database ($IL_{(D)}$) are given in (6), (7) and (8) respectively.

$$IL_{(T)} = \frac{\sum_{x\in T} IL_{(x)}}{|T|} \tag{6}$$

$$IL_{(x\in D)} = \frac{\sum_{x\in D} IL_{(x)}}{|\sigma(x)|} \tag{7}$$

$$IL_{(D)} = \frac{\sum_{T\in D} IL_{(T)}}{|D|} \tag{8}$$

where |T| is the size of the transaction, σ(x) is the total number of occurrences of item x in the database D and |D| is the size of the transactional database to be published.

VI. Empirical Results

The method is implemented in Java. For our experiments, we worked on adult dataset for medical transactions. Graph in Fig. 3 is developed by considering the number of buckets on x-axis and number of transactions on y-axis. The graph in Fig. 3 depicts the distribution of transactions into various buckets. Fig. 4 reveals the information loss of each tuple after anonymizing using the concept hierarchy. X-axis represents the individual transaction and y-axis represents the relevant information loss. Fig. 5 illustrates the information loss vs. privacy gain for transactional databases of different sizes. Database sizes are given on the x-axis and information loss is given on the y-axis. As the privacy keeps on increasing the information loss also keeps on creeping up.

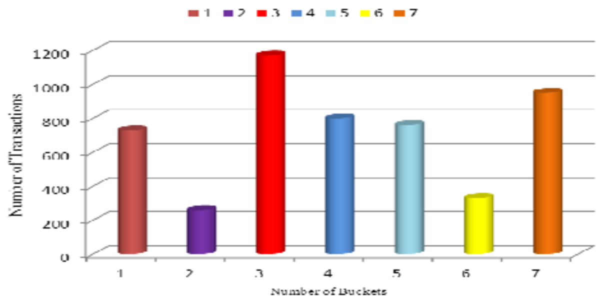

Fig. 3. Distribution of transactions into various buckets

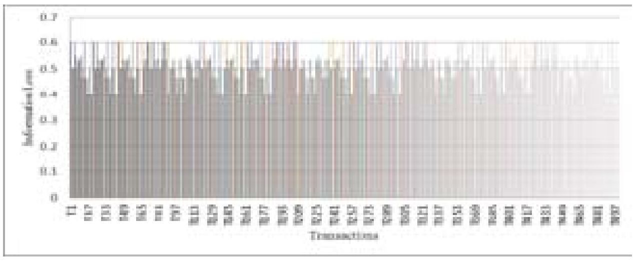

Fig. 4. Information loss for each transaction (|*D*| = 500)

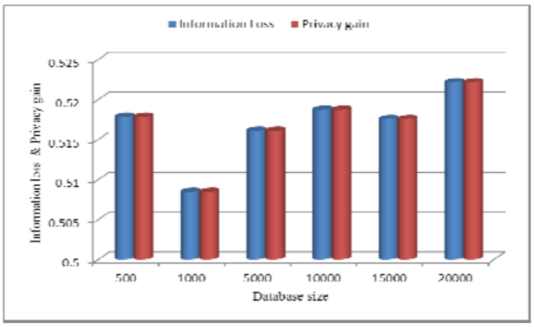

Fig. 5. Information loss vs. privacy gain for databases of different sizes

VII. Conclusions and Future Work

This model brings out a practical problem of maintaining anonymity against transactional data and proposes a simple, yet effective solution. We have provided a simple solution using concept learning for maintaining anonymity for transactional datasets.

In this paper, we studied the anonymization problem of transactional data. We considered the idea of support count as a weight assigned to each item of the transaction and the minimal anonymity property. Hashing is applied to the items of the transactions by considering the sum of the support counts of each item in the transaction. Based on our analysis, we developed an efficient algorithm which is practical for large databases.

We emphasize that our technique is also directly applicable for databases, where transactions may contain both non-sensitive attribute and sensitive attributes. In this case, *k*-anonymity principle can be used to help avoiding associating the sensitive value to less than *k* tuples. In future, we aim at extending our model to *l*-diversity considering sensitive values associated to non-sensitive values.

References

[1] R. Agarwak and R. Srikanth, "Fast Algorithms for Mining association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Databases*, pp. 478 - 499, 1994.

[2] L. Willenborg and T. DeWaal, "Elements of Statistical Disclosure Control," *Springer-Verlag Lecture Notes in Statistics*, 2001.

[3] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," *Technical Report SRI-CSL-98-04*, 1998.

[4] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1010 - 1027, 2001.

[5] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu, "Utility-Based Anonymization Using Local Recoding," in *Proceedings of SIGKDD*, pp. 785 - 790, 2006.

[6] Yabo Xu, Ke Wang, Ada Wai-Chee Fu, and Philip. S. Yu, "Anonymizing Transaction Databases for Publication," in *Proceedings of ACM SIGKDD*, pp. 767 – 775, 2008.

[7] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, pp. 557 - 570, 2002.

[8] Manolis Terrovitis, Nikos Mamoulis, Panos Kalnis, "Privacy-Preserving Anonymization of Set-valued Data," *International Conference on Very Large Data Bases*, pp. 115 – 125, 2008.

[9] R. J. Bayardo and R. Agrawal, "Data Privacy through Optimal k-Anonymization," in *Proceedings of ICDE*, pp. 217 - 228, 2005.

[10] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient Full-domain k-Anonymity," *In Proceedings of ACM SIGMOD*, pp. 49 - 60, 2005.

[11] R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proceedings of the 21th International Conference on Very Large Data Bases*, pp. 407 - 419, 1995.

[12] Y. Chen, G.-R. Xue, and Y. Yu, "Advertising keyword suggestion based on concept hierarchy," in *Proceedings of the international conference on Web search and web data mining, ACM*, pp. 251 - 260, 2008.

[13] M. E. M. D. Beneditto and L. N. de Barros, "Using concept hierarchies in knowledge discovery," *Lecture Notes in Computer Science, Springer*, pp. 255 - 265, 2004.

S. Ram Prasad Reddy holds an MTech in Computer Science and Technology from Andhra University, Visakhapatnam.

He is currently working as Associate Professor in the Department of Computer Science and Engineering, Vignan's Institute of Engineering for Women, Visakhapatnam. Mr. Reddy research interests include Data Mining and Data Security.

Dr. Kvsvn Raju holds a PhD degree in Computer Science and Technology from IIT, Kharagpur.

He is presently working as Professor in the Department of Computer Science and Engineering and is the Director of R&D, at ANITS, Visakhapatnam. Dr. Raju research interests include Software Engineering Data Engineering and Data Security. He is a member of IEEE, ISTE, senior member of CSI and Fellow of IETE.

Dr. V. Valli Kumari holds a PhD degree in Computer Science and Systems Engineering from Andhra University Engineering College, Visakhapatnam.

She is presently working as Professor in the Department of Computer Science and Systems Engineering, Andhra University, Visakhapatnam.

Dr. Valli Kumari research interests include Security and privacy issues in Data Engineering, Network Security and E-Commerce. She is a member of IEEE and ACM and is a fellow of IETE.