

Protecting Frequent Item sets Disclosure in Data Sets and Preserving Item Sets Mining

¹, Shaik Mahammad Rafi,², M.Suman ,M.Tech³, Majjari Sudhakar, M.Tech

⁴, P.Ramesh ⁵, P.Venkata Ramanaih, M.Tech

¹PG (M.Tech) Student in CSE Department, Global College of Engineering and Technology, Kadapa, YSR (D.t).

^{2,3,4}Assistant Professors in CSE Department, MRRITS, Udayagiri, SPSR Nellore(D.t).

⁵ Assistant Professor in CSE Department, GCET, Kadapa, YSR(D.t).

ABSTRACT:

Based on the network and data mining techniques, the protection of the confidentiality of sensitive information in a database becomes a critical issue to be resolved. Association analysis is a powerful and popular tool for discovering relationships hidden in large data sets. The relationships can be represented in a form of frequent itemsets or association rules. One rule is categorized as sensitive if its disclosure risk is above some given threshold. Privacy-preserving data mining is an important issue which can be applied to various domains, such as Web commerce, crime reconnoitering, health care, and customer's consumption analysis.

The main approach to hide sensitive frequent itemsets is to reduce the support of each given sensitive itemsets. This is done by modifying transactions or items in the database. However, the modifications will generate side effects, i.e., nonsensitive frequent itemsets falsely hidden (the loss itemsets) and spurious frequent itemsets falsely generated (the new itemsets). There is a trade-off between sensitive frequent itemsets hidden and side effects generated. Furthermore, it should always take huge computing time to solve the problem.

In this study, we propose a novel algorithm, FHSFI, for fast hiding sensitive frequent itemsets (SFI). The FHSFI has achieved the following goals: 1) all SFI can be completely hidden while without generating all frequent itemsets; 2) limited side effects are generated; 3) any minimum support thresholds are allowed, and 4) only one database scan is required.

I. INTRODUCTION

The data mining technologies have been an important technology for discovering previously unknown and potentially useful information from large data sets or databases. They can be applied to various domains, such as Web commerce, crime reconnoitering, health care, and customer's consumption analysis. Although these are useful technologies, there is also a threat to data privacy. For example, the association rule analysis is a powerful and popular tool for discovering relationships hidden in large data sets. Therefore, some private information could be easily discovered by this kind of tools. The protection of the confidentiality of sensitive information in a database becomes a critical issue to be resolved.

The relationships discovered from a database can be represented in a form of frequent itemsets or association rules. One rule is categorized as sensitive if its disclosure risk is above some given threshold. With an association analyzer, if an itemset with support above a given minimal support, we call the itemset as a frequent itemset.

The problem for finding an optimal sanitization of a source database with association rule analysis has been proven to be NP-Hard [1]. In [2,3,4,5] the authors presented different heuristic algorithms that modify transactions via inserting or deleting items for hiding sensitive rules or itemsets.

Vassilios S. Verykios et al. [2] presented algorithms to hide sensitive association rules, but they generate high side effects and require multiple database scans. Instead of hiding sensitive association rules, Shyue-Liang Wang [3] proposed algorithms to hide sensitive items. The algorithm needs less number of database scans but the side effects generated is higher. Ali Amiri [4] also presented heuristic algorithms to hide sensitive items. Finally, Yi-Hung Wu et al. [5] proposed a heuristic method that could hide sensitive association rules with limited side effects. However, it spent a lot of time on comparing and checking if the sensitive rules are hidden and if side effects are produced. Besides, it could fail to hide some sensitive rules in some cases.

In this study, we propose a novel algorithm, FHSFI for fast hiding sensitive frequent itemsets (SFI). The FHSFI has achieved the following goals: 1) all SFI can be completely hidden while without generating all frequent itemsets; 2) limited side effects are generated; 3) any minimum support thresholds are allowed, and 4) only one database scan is required.

The remainder of this paper is organized as follows: Section 2 presents the problem formulation and notations. In Section 3, we introduce the concept of the proposed algorithm for fast hiding sensitive frequent itemsets and giving examples to illustrate the proposed algorithm. Section 4 is the experimental results which present the performance and various side effects of the proposed algorithm. Section 5 is the conclusion and further work.

II. PROBLEM FORMULATION AND NOTATIONS

In Table 1, we summarize the notations used hereafter in this paper. Let I be a set of items in a transaction database D .

And let $I = \{i_1, i_2, \dots, i_m\}$; $D = \{t_1, t_2, \dots, t_n\}$, where every transaction t_i is a subset of I , i.e. $t_i \subseteq I$. An example database is shown in Table 2. Let X be a set of items in I . If $X \subseteq t_i$, we say that the transaction t_i supports X . There are nine items, $|I|=9$, be minimized.

Table 1. Definitions of variables used in this paper

Variable	Definition
D	the original database
D'	the released database which is transformed from D
U	the sets of frequent item sets generated from D
U'	the sets of frequent item sets generated from D'
t_i	a transaction in Database D
$ t_i $	the number of items in t_i
TID	a unique identifier of each transaction
SFI	the set of sensitive frequent itemsets to be hidden
$SFI.t_j$	a sensitive frequent itemset in the SFI
$\ \cdot \ $	the support count of an itemset, i.e., the number of transactions that support the itemset
w_i	prior weight of t_i
PWT	a table for storing TID and w_i for each transaction in an order decreasing by w_i
MIC_i	the maximal number of itemsets in SFI that contain an item i_k , where $i_k \in t_i, SFI.t_j \subseteq t_i$
$SFI.t_i$	transaction to be modified

and five transactions, $|D|=5$, in the database. The support of itemset X can be computed by equation (1). An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I, Y \subset I$ and $X \cap Y = \emptyset$. A rule $X \rightarrow Y$ will be extracted from a database if

- 1) $support(X \cup Y) \geq \min_support$ (a given minimum support threshold) and
- 2) $confidence(X \cup Y) \geq \min_confidence$ (a given minimum confidence threshold),

where $support(X \cup Y)$ and $confidence(X \cup Y)$ are given by equations (2) and (3), .

$$support(X) = \|X\| / |D| \tag{1}$$

$$support(X \cup Y) = \|X \cup Y\| / |D| \tag{2}$$

$$confidence(X \cup Y) = \|X \cup Y\| / |X| \tag{3}$$

Table 2.
Database D

TID	Transaction
1	1,2,4,5,7

2	1,4,5,7
3	1,4,6,7,8
4	1,2,5,9
5	6,7,8

Table 3.Frequent Itemsets

Itemset	Support
1	80%
4	60%
5	60%
7	80%
1,4	60%
1,5	60%
1,7	60%
1,4,7	60%
4,7	60%

In equation (1), $\|X\|$ denotes the number of transactions in the database that contains the itemset X, and $|D|$ denotes the number of the transactions in the database D. If $\text{support}(X) \geq \text{min_support}$, we call X as a frequent itemset. Table 3 shows the frequent itemsets for a given $\text{min_support} = 60\%$.

For the example $X = \{1,4,7\}$, since $X \subseteq t_1, X \subseteq t_2$ and $X \subseteq t_3$, we obtain $\|X\|=3$. Therefore, $\text{support}(1,4,7)=60\%$. Using the form $X \rightarrow Y$ (support, confidence) for association rules, the rules generated from the above itemset $\{1,4,7\}$ can be described as $1 \rightarrow 4,7$ (60%,75%), $4 \rightarrow 1,7$ (60%,100%), $7 \rightarrow 1,4$ (60%,75%), $1,4 \rightarrow 7$ (60%,100%), $1,7 \rightarrow 4$ (60%,100%) and $4,7 \rightarrow 1$ (60%,100%).

Figure 1 shows the relationships among the sets, U, U', and SFI. The study goal is to hide all SFI and to minimize the loss itemsets. That is, $U' \cap \text{SFI} = \emptyset$ and the set $U - U' - \text{SFI}$ should be minimized.

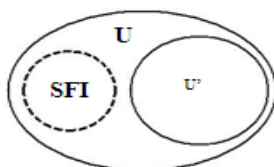


Figure 1. The relationships among the sets, U, U', and SFI

III. THE PROPOSED ALGORITHM

We now demonstrate the algorithm, FHSFI. Given D, SFI, and min_support , the algorithm is to generate a database to be released, D', in which the sensitive frequent itemsets are hidden and the side effects generated are minimized.

The sketch of the FHSFI algorithm is shown in Figure 2, which can be depicted as the following stages.

```

FHSFI();
Input: D, SFI, min_support;
Output: D';

Stage 1
1 For each transaction  $t_i$  in the Database D Do
2 Begin
3   If exist any SFI  $t_i$  (sensitive frequent itemsets) supported by  $t_i$  then
4   Begin
5      $\|SFI_{t_i}\| := \|SFI_{t_i}\| + 1;$ 
6      $MIC_i :=$  the maximum item_count from Heuristic( $t_i$ , SFI);
7     Compute the prior weight  $w_i$  for each  $t_i$  by the function;
8      $w_i = 1 / [2^{(\|t_i\| - 1)} / MIC_i];$ 
9     Store the TID and the  $w_i$  in PWT;
10  End;
11 End;

Stage 2
12 While SFI is not empty ( $\neq \emptyset$ ) do
13 Begin
14   Select a TID from PWT with maximal  $w_i$ ;
15   Determine which item in TID will be modified according to
16   Heuristic(TID, SFI);
17   Modify the item;
18   Modify  $w_{TID}$  of the TID and insert the TID into the PWT;
19 For each SFI  $t_j$  that contains the modified item Do
20 Begin
21    $\|SFI_{t_j}\| := \|SFI_{t_j}\| - 1;$ 

```

Stage 2 repeats to modify transitions one-by-one until all SFI have been hidden. The order of the transaction modifications is according to the prior weight associated with a transition. The following tasks are repeated until SFI is empty.

- Select a transaction t_k from PWT such that w_k is maximal.
- Select the item to be deleted, according to the heuristic shown in Figure 4, and delete it.
- Recompute w_k after modifying each item, and then insert it into the PWT in the maintained order.
- Subtract 1 from $\|SFI.t_j\|$ if $SFI.t_j$ contains the deleted item and is supported by t_k .
- Remove $SFI.t_j$ from SFI, if the $(\|SFI.t_j\| / |D|) < \text{min_support}$.

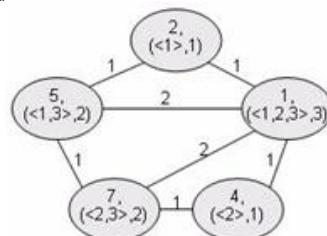


Figure 3. The correlation between t_1 and SFI

```

IF  $\|SFI.t_j\| / |D| < \text{min\_support}$ 
20 then
21   Remove  $SFI.t_j$  from SFI;
22 End;
23 End;
    
```

Figure 2. The pseudo code of the FHSFI algorithm

In stage 1, FHSFI scans database once while collects all useful information about the correlation with SFI for each

Table 4.

An example of sensitive frequent itemsets, SFI

	Itemset
1	1,2,5
2	1,4,7
3	1,5,7
4	6,8

Table 5.

The support count for each itemset in SFI

	Itemset	$\ \cdot\ $
1	1,2,5	2
2	1,4,7	3
3	1,5,7	2
4	6,8	2

transaction, including $\|SFI.t_j\|$ and w_i . The $\|SFI.t_j\|$ is used for checking if $SFI.t_j$ has been hidden. The w_i is a prior weight of a transaction t_i , which provides a heuristic for estimating side effects and can be computed by equation (4).

$$w_i = 1 / [2^{(|t_i|-1)} / MIC_i].$$

Table 4 shows an example of sensitive frequent itemset. Let $t_1 = \{1,2,4,5,7\}$, which supports $SFI.t_1$, $SFI.t_2$ and $SFI.t_3$. As shown in Figure 3 the correlation between t_1 and the SFI can be represented by a graph $G=\langle V,E\rangle$. Each node is for an item i_k in t_1 ; the weight associated with each edge in E denotes the number of the itemsets in SFI that contain the both adjacent nodes connected by the edge. Each node can be represented as $(\{SFI.t_j \mid SFI.t_j \subseteq t_i, i_k \in SFI.t_j\}, \text{item_count}_{SFI.t_i})$. For example, the node $\langle \{1,2,3\}, 3 \rangle$ for item '1' indicates that three itemsets in SFI that contain the item '1', namely the $SFI.t_1$, $SFI.t_2$, and $SFI.t_3$. As shown in Figure 3, item '1' has the maximum $\text{item_count}_{SFI.t_i}$ which is equal to 3. Hence, we obtain $MIC_1 = 3$ and $w_1 = 3/16$.

Figure 4 shows the heuristic procedure for determining which item to be modified and for computing MIC for transaction t_i .

```

Heuristic ();
Input: TID, SFI;
Output: the item to be modified, MICi;
1 Begin
2 For each SFI.t in SFI do
3 Begin
4 If the transaction tTID fully supports SFI.t; then
5 Begin
6 For each item SFI.tj.i in SFI.tj Do
7 item_countSFI.tj.i = item_countSFI.tj.i + 1;
8 End;
9 End;
10 Select the SFI.tj.i with maximum item_count as the item of tTID to be modified;
11 Return(SFI.tj.i, item_count);
12 End;
    
```

Fig:pseudo code of heuristic procedure

Table 6.

The MIC and prior weight for each transaction in D

TID	Transaction	t _i	MIC	w
1	1,2,4,5,7	5	3	3/16
2	1,4,5,7	4	2	2/8
3	1,4,6,7,8	5	1	1/16
4	1,2,5,9	4	1	1/8
5	6,7,8	3	1	1/4

Table 7.The example PWT

	TID	w
1	2	2/8
2	5	1/4
3	1	3/16
4	4	1/8
5	3	1/16

Table 8. Experiment results for |SFI|=5

D	CPU time(ms)	U	U'	#loss itemsets	#modified entries
5000	326.6	439	428.6	5.4	143
10000	454.2	417	406.4	5.6	307.2
15000	701	426	415.6	5.4	513
20000	905	442	431	6	711.6
25000	1183.6	432	421.2	5.8	902.8
30000	1502	443	432.4	5.6	863.8

Now, we use the following example for illustrating the proposed algorithm FHSFI.

Example 1. Given D, SFI, as shown in Tables 2 and 4, and min_support = 40%. As shown in Table 5, the support count for each SFI.t can be obtained from D and SFI. For example, SFI.t₂, {1,4,7}, is supported by t₁, t₂, and t₃, so ||SFI.t₂|| = 3. Table 6 lists the length, MIC, and the prior weight for each transaction in the database. The PWT, as shown in Table 7, can be obtained by sorting Table 6 in the decreasing order by w. Then, the first transaction, i.e., t₂, in PWT is chosen to be modified. According the heuristic shown in Figure 3, the item '1' in t₂ are removed. Hence, ||SFI.t₂|| and ||SFI.t₃|| will be reduced by 1. SFI.t₃ is removed from SFI because the (||SFI.t₃|| / |D|) < min_support. The process is repeated until the SFI is empty. Finally, the FHSFI algorithm removes the item '1' in t₂, the item '6' or '8' in t₅ (select randomly), and the item '1' in t₁. Now all sensitive frequent itemsets in SFI have been hidden. ■

IV. PERFORMANCE EVALUATION

We have performed our experiments on a notebook with 1.5G MHz processor and 512 MB memory, under Windows XP operating system. The IBM data generator [11] is used to synthesize the databases for the experiments. Databases with sizes 5K, 10K, 15K, 20K, 25K, and 30K are generated for the series of experiments. The average length of transactions of each database is 10 and 50 items in the generated database. The minimum support threshold given is 30%. The experimental results are obtained by averaging from 5 independent trials with different SFIs.

The performance of the FHSFI algorithm has been measured according to three criteria: CPU time requirements, side effects produced, and the number of entries modified. Tables 8 and 9 present the experimental results for $|SFI|=5$ and $|SFI|=10$, respectively.

The CPU time requirements, side-effect evaluation, and the number of entries modified for varied $|D|$ and $|SFI|$ are shown in Figures 6, 7, and 8, respectively.

Table 9. Experiment results for $|SFI|=10$

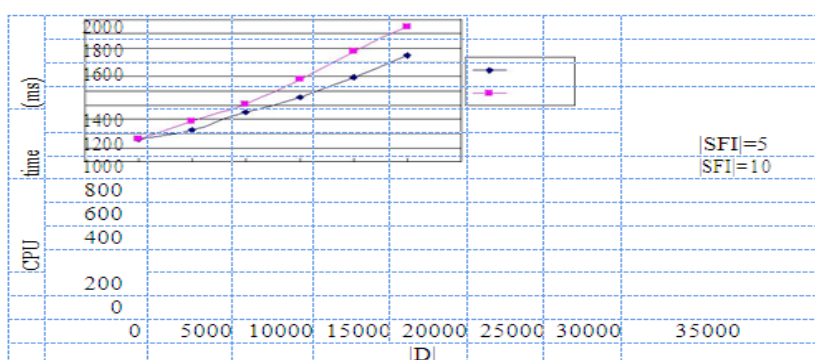


Figure 6. CPU time requirements

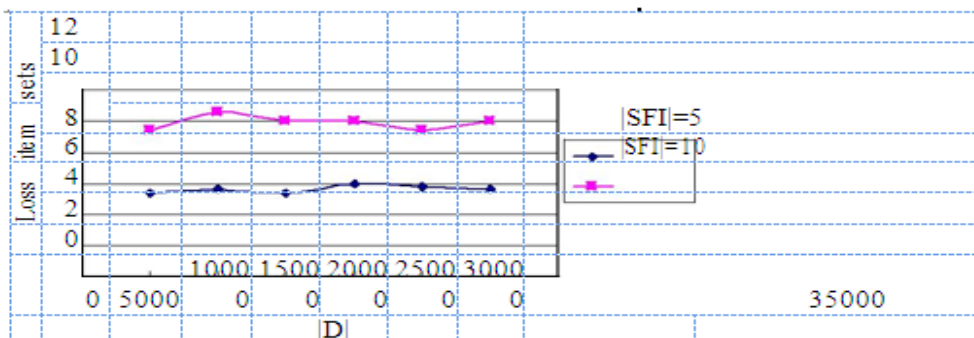
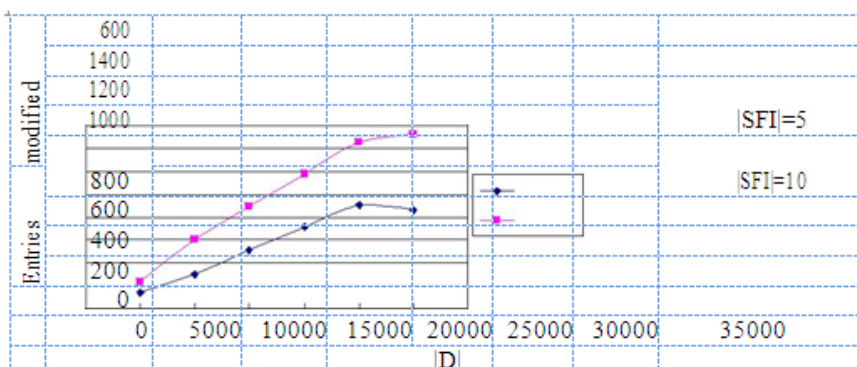


Figure 7. The side-effect evaluation



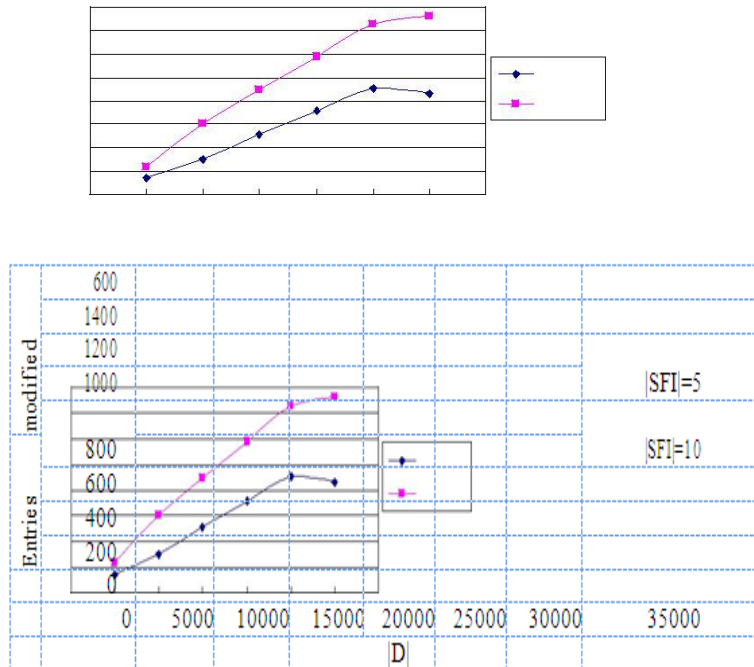


Figure 8. The number of entries modified
The experimental results for FHSFI can be summarized as follows:

- As shown in Figure 6, the CPU time is linear growth with the size of database and is scalable with the size of SFI.
- The number of loss itemsets is independent of the size of database, but linear-related with the size of SFI sets, which can be discovered in Figure 7.
- The number of the modified entries depends on the size of the database and the size of SFI. However, since the heuristic procedures are used to determine the order of modifications, we can observe in Figure 8 that only a small part of transactions in the database are modified. For $|D|=10000$, only 600 transactions are modified for completely hiding the 10 item sets in SFI.

V. CONCLUSIONS AND FURTHER WORK

In this paper, we have presented the FHSFI algorithm in order to fast hide sensitive frequent itemsets with limited side effects. The correlations between the sensitive itemsets and each transaction in the original database are analyzed. A heuristic function to obtain a prior weight for each transaction is given. The order of transactions to be modified can be efficiently decided by the weight for each transaction. This will reduce the time to deal with the transactions whose modification is not helpful for hiding the given sensitive frequent itemsets. In other words, the number of transactions in D that we have to deal with could also be reduced.

Our approach has achieved the following goals: 1) all SFI can be completely hidden while without generating all frequent itemsets; 2) limited side effects are generated; 3) any minimum support thresholds are allowed; and 4) only one database scan is required. In this research, one of our goals is hiding all SFI with limited side effects, but our algorithm still causes some loss rule sets. We are currently considering extensions on the algorithms to solve the problem. Another one is to apply the ideas introduced in this paper to fast hide sensitive association rules. These issues could be studied in the future.

REFERENCES

- [1] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, V. Verykios, "Disclosure limitation of sensitive rules" *Knowledge and Data Engineering Exchange*, pp. 45-52, 1999.
- [2] Vassilios S. Verykios, A.K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association Rule Hiding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434-447, 2004.
- [3] Shyue-Liang Wang, "Hiding sensitive predictive association rules", *Systems, Man and Cybernetics, 2005 IEEE International Conference on Information Reuse and Integration*, vol. 1, pp. 164-169, 2005.
- [4] Ali Amiri, "Dare to share: Protecting sensitive knowledge with data sanitization", *Decision Support Systems archive* vol. 43, issue 1, pp. 181-191, 2007.
- [5] Yi-Hung Wu, Chia-Ming Chiang, and Arbee L.P. Chen, "Hiding Sensitive Association Rules with Limited Side Effects", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, issue 1, pp. 29 - 42, 2007.

Authors Profile



Mr. Shaik.Mahammad Rafi –He was born in Rajampet, Kadapa, A.P, India in 1990.He is studying M.Tech in the department of Computer Science and Engineering at **Global College of Engineering and technology**, Kadapa. He has done Bachelor's of Technology from JNTUA University in the year 2011 in Information Technology.



Mr. M.SUMAN- He was born in Rajampet, Kadapa, A.P, India in 1985.He is Masters of Technology in Computer Science and Engineering from JNTUH University in the year 2013 at **Vathsalya Institute of Technology & Science**, Anantharam Bhongiri, Nalgonda . He has given guidance to many students in their thesis work of M.Tech. He has also contributed in the research work on Image Processing with his papers. He is presently working as Asst. Professor in **Mekapati Rajamohan Reddy Institute of Technology and Science** ,Udayagiri,SPSR Nellore. He has done Bachelor's of Technology from JNTUH University in the year 2006 in Computer Science & Engineering at **Annamacharya Institute of Technology & Science**, Rajampet, Kadapa.



Mr. MAJJARI SUDHAKAR- He was born in Raghunathapuram, Badvel, Kadapa, A.P, India in 1983.He is Masters of Technology in Information Technology from JNTUH University in the year 2010 at **Aurora's Scientific Technological & Research Academy** , Bandlaguda , Hyderabad.He has 3 years of Teaching Experience given guidance to many students in their thesis work of M.Tech. He has also contributed in the research work on Software Engineering with his papers. He is presently working as Asst. Professor in **Mekapati Rajamohan Reddy Institute of Technology and Science** ,Udayagiri,SPSR Nellore. He has done Bachelor's of Technology from JNTUH University in the year 2006 in Computer Science & Engineering at **Sri Venkateswara College of Engineering & Technology**,R.V.S Nagar Chittur.



Mr. P.RAMESH- He was born in Naidupet, S.P.S.R.Nellore, A.P, India in 1989.He is studying M.Tech in the department of Computer Science and Engineering at **SIR C.V.RAMAN INSTITUTE OF TECHNOLOGY & SCIENCE**, Tadipatri, Anantapur. He has done Bachelor's of Technology from JNTUA University in the year 2011 in Computer Science & Engineering



Mr. P.Venkata Ramanaiah –He was born in Khajipet,Kadapa,A.P,India. He is Master of Technology in Computer Science and Engineering at Madanapalle Institute of Technology and Science from JNTUA University in the year 2012. He has given guidance to many students in their thesis work of M.Tech. He has also contributed in the research work on Data Mining with his papers. He was presently working as Assistant. Professor in **Global College of Engineering and technology**,GCET, kadapa.