Protecting Sensitive Frequent Itemsets in Database Transaction Using Unknown Symbol

Dedi Gunawan[#], Jan Wantoro[#]

[#]*Informatics Department, Department of Information Technology Education*
Universitas Muhammadiyah Surakarta

Surakarta, Indonesia

[1]dedi.gunawan@ums.ac.id

[2]jan.wantoro@ums.ac.id

Abstract—**Data mining algorithms give advantages on data analytics, thus the information which are hidden from database can be revealed maximally as a result the data owner may use it effectively. Besides the benefits, it also brings some challenges like some information which is considered as sensitive can be revealed under some algorithms. Sensitive information can be considered as the information of people or organization that should be kept under certain rule before it is published. Therefore, in this research we propose an efficient approach to deal with privacy preserving data mining (PPDM) for avoiding privacy breach in frequent itemsets mining. The size of database is also be considered, therefore we conduct data segregation in order to separate between transactions with sensitive itemsets and transactions without sensitive itemsets. This step is followed by deriving which item from transactions that is going to be replaced using unknown symbol to perform data sanitization. A set of experiment is conducted to show the benefit of our approach. Based on the experimental results, the proposed approach has good performance for hiding sensitive itemsets and also it results less changes in the original database.**

Keywords—Frequent itemsets, Sensitive itemsets hiding, Data Mining, Unknown Symbol.

I. INTRODUCTION

Data mining has a great contribution in information industry, it plays an essence role for data analytic in many areas ranging from marketing and business analysis to scientific aspect [12]. However, we have to admit that some mining tools may cause sensitive information leakage and there is no guarantee that it can be abused by some people. Privacy preserving data mining tries to give solutions about the mentioned problem. It investigates the side effects of data mining methods that emanate from the penetration into the data privacy[2] and information privacy[13]. Data privacy more likely related to raw data such as name and address from individual or organization while information privacy is about the knowledge behind the data such as personal buying pattern or patient disease. In this work we focus on frequent itemsets hiding which has close relation with association rules mining. Apriori-based algorithm and FP-Growth based algorithm are most common types of association rule mining algorithms. The first mentioned algorithm [6] firstly scans the dataset to count the 1-itemset occurrences then by using subsequent pass *k* to generate k-itemsets until it generates the maximum k-itemsets. On the other hand, FP-Growth algorithm [7] is using pattern tree structure (FP-tree) in order to find frequent pattern in transactional database. there are several approaches that have been proposed in the research literature to deal with privacy in data mining, furthermore after the pioneer works of [3] and [4] were introduced to solve the problem in PPDM. Generally, there are three tasks to solve in privacy preserving data mining. As the first and the most important requirement of all, the entire sensitive itemsets should not be mined in the sanitized database. The second requirement, non-sensitive itemsets in the original database should also be mined from the sanitized database. The last, the difference between original database and sanitized database should be minimized. Finding an excellent sanitized database which achieves the three requirements above is a very difficult problem moreover, it has been proved that the problem is NP-Hard [5]. As a result, there are many heuristic approaches were proposed to solve the problem. Based on the transaction modification strategies, PPDM can be categorized into three classes namely heuristic-approach, border-based approach and constraint-satisfaction problem approach. Each single approach has its advantages as well as drawbacks.

In this research we use our methodology from previous research finding [18] and borrow the idea from [2]. However, asterix symbol is devised rather than question mark The proposed approach can be roughly divided into two phases. The first phase is data preprocessing phase. We omit non-sensitive transaction in database such that only sensitive transactions reside in the database. The second phase is our main algorithm which is selecting victim item and replacing the victim item using asterix symbol.

The rest of paper is organized as follows. Section 2 explains the previous works. The advised approach and example are described in section 3. Section 4 states the experimental results of our approach by comparing with a naïve approach. Finally, section 5 concludes this work as well as direction for further research.

II. RELATED WORK

The work of [2] proposed a framework to solve the problem to hide sensitive frequent items. The researchers used blocking methodology by replacing selected sensitive item

with question mark '?'. Meanwhile, the proposed method from [8] was using distributed systems to deal with hiding sensitive frequent itemsets in horizontally partitioned database. There are two main steps in this approach. The first step is using commutative encryption protocol to protect sensitive information for each node. Following that, the next phase is removing selected item in order to reduce support and confidence of the rules. However, the performance of the algorithm reduce slightly since the encrypt and decrypt for every process in distributed system takes longer time to complete. In contrary, proposed solution from [9] was able to work in horizontally partitioned and vertically partitioned dataset to hide frequent itemsets. The main idea of the algorithm is using non sensitive items to replace frequent sensitive items as a result it is no longer frequent in database. However, this algorithm has drawback in terms of modifying transaction since it has to select and count the number of item to be removed and do the same thing for non frequent items as the substitute. Another research proposed the insertion of fake transactions into original database [10]. this approach is successfully hide the sensitive itemsets due to the number of artificial transaction makes the support count decreases under the minimum support threshold. Since it added artificial transaction, consequently, the released database changes significantly. In [11] the technique proposed to deal with privacy preserving problem is by inserting noisy items in certain transactions. The noisy items are selected based on queue and random number generator. However, if a transaction has a lot of items then a lot of noisy items are inserted in to the transaction. another renown technique is border-based approach. This approach focuses on preserving the border of non-sensitive frequent itemsets and it does not considering all non-sensitive itemsets during the sanitization process. There are several finding which is discussing this approach like in [14][15][16]. Another approach was also proposed in [1] by multiplying the original database with sanitization matrix.

III. PROPOSED ALGORITHM

In this section, a detail description of our approach is presented. In the approach, database is firstly split into two parts. The first part will be a database that has sensitive itemsets and the other part without any sensitive itemsets.

A. Pre-processing database

The aim of splitting database algorithm in figure 1, is to eliminate unnecessary transactions before hiding process. Steps 1 to 4 are the preparation steps where we create temporary list to save sensitive itemsets from user. In step 5, for each transaction we check whether it contains sensitive itemsets. If the transaction contains sensitive itemsets, then this transaction will be put into sensitive dataset otherwise it will be saved into non-sensitive dataset. Following that, a heuristic algorithm is performed on each server to hide the sensitive itemsets.

B. Heuristic Algorithm using Unknown Symbol

Heuristic algorithm using unknown symbol (HAUS) is proposed in this paper. In this technique items are selected based on its occurrences in sensitive itemsets. An item is selected as the candidate of victim item if its occurrence is higher than minimum degree of sensitivity. Minimum degree of sensitivity is the average value of sensitive items occurrences. In other words, the higher the occurrence of the item in the sensitive itemsets, the higher the priority the item to be selected as the victim item. Moreover, to avoid the situation of transaction loss, we sort the transaction based on the length in descending order. After that, we use equation 1 to compute M_i, where M_i is the minimum number of modifications for victim item *i*.

$$M_i = V_i - [minsup*N] + 1 \quad (1)$$

In the equation, V_i is the number of occurrences of victim item *i* , *minsup* is minimum support given by user and N is the number of total transaction in database. The last step is replacing victim item in a transaction by using unknown symbol.

Let us consider *si* as the minimum degree of sensitivity, *d* is the number of distinct items in sensitive itemsets and *r* is the occurrences of an item in the sensitive itemsets. Equation 2 explains how the minimum degree of sensitivity can be computed.

$$si = \frac{\sum_{k=1}^{d} r}{d}. \quad (2)$$

input: dataset, sensitive itemsets
output: sensitive dataset, non-sensitive dataset

1. Load dataset
2. Read the sensitive itemsets from user input
3. Create list to keep sensitive itemsets
4. Add the sensitive itemsets into list
5. For each transaction
   a. if transaction contains sensitive itemsets
   b. then, add to sensitive dataset
   c. else, add to non-sensitive dataset
6. Save into files

Fig. 1.Splitting database algorithm

Step 1 to 4 are the initial steps, in which database is loaded and followed by creating list of sensitive itemsets from user's input. Step 6 is to select victim item. However, if there are some sensitive itemsets where each of item occurrences is not higher than the minimum degree of sensitivity, we will pick one item from it randomly then add the item into victim item list. After the victim items have been determined, the following step is sorting the transaction based on its length. The aim of sorting is to avoid the situation of transaction loss due to all the items in certain transaction are replaced. The next step is counting the number of transaction to be modified. After that, transaction will be modified by replacing victim items with asterix symbol. The process of modifying transaction only need one scan of database hence it will reduce the time computation. The last step is saving the modified transaction into sanitized database.

Input: sensitive data chunk, minimum support, sensitive itemsets
Output: sanitized database

1. Load database
2. While (transaction != null)
   a. number transaction ++
   b. add transaction to the list of transaction
3. Create map to count item occurrences
   a. while (element != null)
      i. number of item ++
      ii. support item = number item / number transaction
   b. sorting the items in ascending order
4. for i = 0, i< length of sensitive itemsets
   a. number of sensitive itemset +=1
   b. add sensitive itemset to the list of sensitive itemset
   c. sorting list in ascending order
5. Sort transaction based on the length in descending order
6. victim items selection
   a. create hash map
   b. count the sensitivity of each sensitive item
      i. counting the total occurrences of distinct item in sensitive itemset
      ii. sensitivity = number of occurrences of item in list / distinct sensitive item
      iii. if occurrences of sensitive item > value of sensitivity
      iv. add item into list of victim item
   c. if the item occurrences in sensitive itemsets is below the minimum degree of sensitivity
      i. select one item from the sensitive itemsets
      ii. add the selected item to the victim item list
7. count the number of modified transactions
   a. creating hash map
   b. for each victim item
      i. count the number of modified item
      ii. replaced item = number of occurrences of each victim item – minsup*number of transaction + 1
      iii. add the number of replaced item into hash map
8. transaction modification
   a. for each victim item
   b. replacing victim item with asterix symbol for y < number of victim item
9. save the result into sanitized database

Fig. 2. Heuristic Algorithm Using Unknown Symbol

C. Example

We use the following example to illustrate the proposed algorithm. Refer to table 1 and assume the sensitive itemsets S is {cfh, af, cd} and user specified minimum support is 40%. Thus, the minimum count is 4. Based on the equation 2, firstly we count the number of distinct item in S, and then followed by counting the item occurrences in sensitive itemsets. Since there are five distinct items in S, so that d=5 and the occurrence of each item is r_a = 1, r_c=2, r_d=1, r_f=2, and r_h=1 such that $si = \frac{1+2+1+2+1}{5} = 1.4$. Thus, the victim items are c and f due to those occurrences is higher than the si value. Based on the equation 1, items c and f will be replaced from Tid.1, Tid.3, Tid.4, Tid.8 and Tid.10. As can be seen in table 2, the sensitive itemsets are no longer frequent in the sanitized database. Moreover, the sanitized dataset has less modification and the number of transactions remain the same.

TABLE I. SAMPLE DATABASE TRANSACTIONS

Tid	Item
1	a b c d f g h
2	a d e
3	b c d f g h
4	a b c f h
5	c d e i
6	a c f i
7	b c f g
8	c d f h i
9	a f i
10	a c e f h

TABLE II. SANITIZED DATABASE

Tid	Item
1	a b d g h * *
2	a d e
3	b d g h * *
4	a b h * *
5	c d e i
6	a c f i
7	b c f g
8	d h i * *
9	a f i
10	a e h * *

IV. Experimental Results

In this section, a set of experiment is performed to show the efficiency of the proposed approach. In the experiment, testing database is generated from IBM Synthetic data generator [19]. The testing is conducted in local computer with windows 8 operating system, 6 GB memory and 500 GB hard drive. The setting parameters are listed in table 3.

We use the idea of measurements from [17] such as, hiding failure, misses cost and dissimilarity, in order to know the performance of the proposed approach. In the following discussion, we use D and D' to indicate the original and sanitized databases, respectively.

Hiding failure is measured by HF = $\frac{|Ph(D')|}{|Ph(D)|}$ where |Ph(x)| denoted as the number of sensitive itemsets in dataset X. Since the goal of our approach is to hide the sensitive itemsets, thus modified itemsets resulting zero hiding failure. Therefore, we do not show the experimental result here.

The value of misses cost is denoted by MC = $\frac{|\sim Ph(D)|-|\sim Ph(D')|}{|\sim Ph(D)|}$ where |~Ph(x)| is the number of sensitive itemsets in database X. The misses cost of our approach and the naïve approach is shown in figure 4. As shown in the result, the misses cost of our approach is much smaller than that of naïve approach.

The dissimilarity value between original and sanitized datasets is measured by Dis = $\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(Dij)}{\sum_{i=1}^{n}\sum_{j=1}^{m}}$. Dissimilarity value from proposed approach as shown in figure 5 is considerably low compared with the result from naïve approach since the proposed algorithm performs victim item selection. As a result, it will reduce the number of transaction modification. Furthermore, the number of transaction in sanitized database remains the same with the number of transaction in the original database. Figure 6 shows the computation time from proposed approach and naïve approach respectively. As showed in the results, the computation time of proposed approach is lower than that of naïve approach. This is because the proposed approach only modifies selected transactions even though it needs to sort transactions and perform selection on victim items, meanwhile in naïve approach there is no victim item selection therefore, all the items in sensitive itemsets are considered as the victim items and all the transaction that contains sensitive itemsets are subjected to be modified.

TABLE III. TESTING PARAMETERS

Parameter	Default value	Range
Min Sup	3 %	1 – 5%
Number sensitive itemset	0.3 %	0.1 – 0.3%
Transaction length	-	1-10
Number of Transaction	100000	-

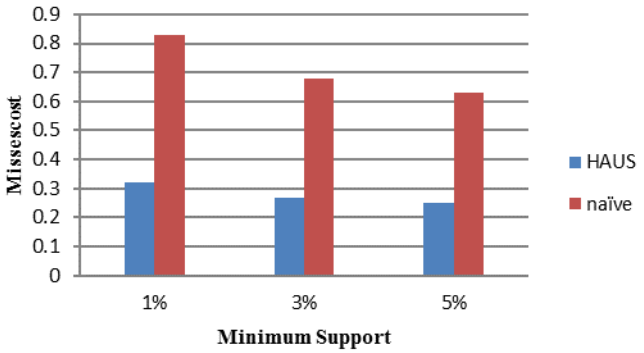

Fig.4. Misses cost comparison

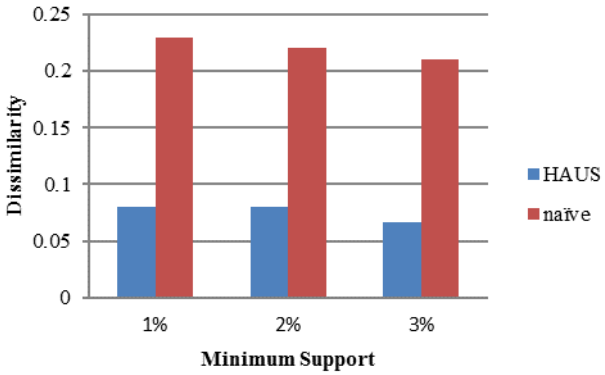

Fig.5. Dissimilarity comparison

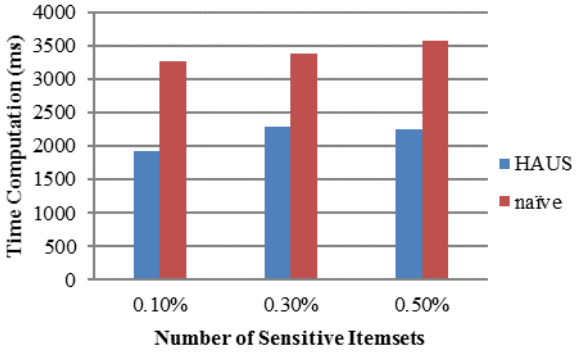

Fig.6.Comparison of time computation

V. Conclusion

This research is focusing on hiding sensitive itemsets in transactional database. Therefore, we propose heuristic approach that performs splitting algorithm to reduce the size of data and boost the hiding process while keep the transaction in order to minimize side effect; such as missing transaction and lowering item lose. Result indicates that the proposed approach has better performance compared with naïve approach in terms of misses cost and dissimilarity which is about more than twice times lower.

VI. References

[1]. Wang, E.T and Lee Guanling. An efficient sanitization algorithm for balancing information privacy and knowledge discovery in association patterns mining. Data & Knowledge Engineering. pp. 463–484, 2008.

[2]. Aris Gkoulalas-Divanis., Vassilios S. Verykios.Association Rule Hiding for Data Mining. Springer, 2010.

[3]. R. Agrawal and R. Srikant. Privacy preserving data mining. SIGMOD Record, pp. 439–450, 2000.

[4]. Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal* of Cryptology, pp. 36–54, 2000.

[5]. M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure limitation of sensitive rules. In Proceedings of the 1999 IEEE Knowledge

and Data EngineeringExchange Workshop (KDEX). pp. 45–52, 1999.

[6]. Agrawal, R., Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499, 1994.

[7]. Han, J., Pei, J., Yin, Y. Mining Frequent Patterns without Candidate Generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, , pp. 1-12, 2000.

[8]. Alaa Kh. *Juma'a*, Sufyan T. F Al-janabi, Nazar A. Ali. Hiding sensitive frequent itemsets over privacy preserving distributed data. raf. j. of comp & math's., vol. 10, no. 1, pp. 91-101, 2013.

[9]. Al-Khafaji, K, H. and Al-saidi A. N. A new algorithm to preserve sensitive frequent itemsets in horizontal and vertical database. eng&tech. journal vol.31, part (B). No.6. , pp. 755-769, 2013

[10]. Lin, C. W et al. A greedy-based approach for hiding *sensitive* itemsets by transaction insertion. Journal of Information hiding and multimedia signal processing. Vol 4, Number 4, pp. 201-214, 2013.

[11]. Lin Jun-lin., Cheng yung-wei. Privacy preserving itemset mining through noisy items. Expert system with Applications. pp. 5711-5717, 2008.

[12]. Fayyad, Usama et al. From Data Mining to Knowledge Discovery in Database. American Association for Artificial Intelligence. , pp. 37-54, 1996.

[13]. Hughes, Dominic and Shmatikov, Vitaly. Information Hiding, Anonymity and Privacy: A Modular Approach. Journal of computer security. pp. 3 – 36. 2004.

[14]. Sun X, Yu PS. A border-based approach for hidingsensitivefrequent itemsets. In: Proceedings of IEEEInternational Conference on Data Mining. Houston,TX; pp. 426–433, 2005.

[15]. Sun X, Yu PS. Hiding sensitive frequent itemsets by aborder-based approach.Comput Sci Eng, vol.1, 74–94, 2007.

[16]. Moustakidesa GV, Verykios VS. A Maxmin approach for hiding frequent itemsets. Data Knowledge Engineering, Vol. 65, 75–89, 2008.

[17]. S.R.M. Oliveira, O.R. Zaıane. Privacy preserving frequent itemset mining, in: Proceedings of the IEEE ICDM Workshop on Privacy,Security, and Data Mining, Maebashi City, Japan, pp. 43–54, 2002.

[18]. Gunawan Dedi and Lee Guanling. Heuristic Approach on Protecting Frequent Sensitive Itemset in Parallel Computing Environment. Proceeding of International Conference on Pure and Applied Research. Malang, Indonesia. 2015

[19]. IBM Synthetics data generator. http://synthdatagen.codeplex.com/wikipage?title=Generating%20Transactions&referringTitle=Using%20SyntheticDataGenerator.