Novel Algorithms for Privacy Preserving Utility Mining

Jieh-Shan Yeh, Po-Chiang Hsu and Ming-Hsun Wen
Department of Computer Science and Information Management, Providence University
{jsyeh, g9471055, g9571005}@pu.edu.tw

Abstract
Privacy Preserving Data Mining (PPDM) has become a popular topic in the research community. How to strike a balance between privacy protection and knowledge discovery in the sharing process is an important issue. This study focuses on Privacy Preserving Utility Mining (PPUM) and presents two novel algorithms, HHUIF and MSICF, to achieve the goal of hiding sensitive itemsets so that the adversaries can not mine them from the modified database. In addition, we minimize the impact on the sanitized database in the process of hiding sensitive itemsets. The experimental results show that HHUIF achieves the lower miss costs than MSICF does on two synthetic datasets. On the other hand, MSICF generally has the lower difference between the original and sanitized databases than HHUIF does.

1. Introduction
The traditional association rule mining, one of the most important methodologies in data mining, discovers all itemsets which support values are greater than a given threshold. There are lots of algorithms proposed for discovering the frequent itemsets in literature. The Apriori algorithm [1, 2, 13] is considered as the most famous one. In order to measure how “useful” an itemset is in the database, utility mining is proposed [22]. It overcomes the shortcomings of traditional association rule mining, which ignores the sale quantity and price (or profitability) among items in a transaction.

On the other hand, Privacy Preserving Data Mining (PPDM) [20] becomes a popular research direction in data mining in the past few years. In 1996, Clifton et al. [5] analyzed that data mining can bring about threat against databases and addressed possible solutions to achieve privacy protection of data mining. In 2002, Rizvi et al. discussed the privacy preserving mining of association rules [17, 18]. However, to the best of our knowledge, there is no Privacy Preserving Utility Mining (PPUM) discussed in any related literature. Therefore, this study focuses on PPUM and presents two novel algorithms, HHUIF and MSICF, to achieve the goal of hiding sensitive itemsets so that the adversaries can not mine them from the modified database. In addition, we minimize the impact on the sanitized database in the process of hiding sensitive itemsets.

The rest of this paper is organized as follows. Related works are reviewed in Section 2. Then, Section 3 proposes the HHUIF and MSICF algorithms to improve the balance between privacy protection and knowledge discovery. Section 4 presents the experimental results and evaluates the performance of the proposed algorithms. Finally, we present the conclusions in Section 5.

2. Related Works
2.1. Utility Mining
Utility mining discovers all itemsets whose utility values are equal or greater than a user specified threshold in a transaction database. However, the utility value of a itemset does not satisfy the “downward closure property”. That is, a subset of a high utility itemset may not be a high utility itemset. The challenge of utility mining is in restricting the size of the candidate set and simplifying the computation for calculating the utility. Recently, Li et al. developed some efficient approaches, including the FSM, SuFSM, ShFSM, and DCG methods for share mining [8, 9, 10]. Under appropriate adjustments on item count and external utility of items, share mining is equivalent to utility mining. In the meanwhile, Liu et al. [12] also presented the Two-Phase (TP) algorithm for fast discovering all high utility itemsets. By using isolated items discarding strategy, Li et al. [11] proposed an efficient algorithm for discovering high utility itemsets.

The following set of terms are defined and given in [11] for the utility mining problem.
Table 1. An example of transaction database
(a) Transaction table.

<table>
<thead>
<tr>
<th>TID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>T6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T7</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T8</td>
<td>3</td>
<td>0</td>
<td>25</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>T9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T10</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

(b) The external utility table.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>PROFIT ($) (per unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
</tr>
</tbody>
</table>

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, where $m$ is the total number of items. Let $DB = \{T_1, T_2, \ldots, T_n\}$, the task-relevant database, be a set of transactions where each transaction $T_q$ is a set of items, that is, $T_q \subseteq I$. A set of items is also referred as an itemset. An itemset that contains $k$-items is called a $k$-itemset.

- The item count of item $i_p \subseteq I$ in transaction $T_q$, $c(i_p, T_q)$, is the number of items $i_p$ purchased in transaction $T_q$. For example, $c(A, T_1) = 0$, $c(B, T_1) = 0$, and $c(C, T_1) = 18$, in Table 1 (a).
- Each item $i_p$ has an associated set of transactions $T_p = \{ T_q \in DB | i_p \in T_q \}$.
- A $k$-itemset $X = \{x_1, x_2, \ldots, x_k\}$ is a subset of $I$, where $1 \leq k \leq m$.
- Each $k$-itemset $X$ has an associated set of transactions $T_X = \{ T_q \in DB | X \subseteq T_q \}$.
- The external utility of item $i_p \subseteq I$, $eu(i_p)$, is the value associated with item $i_p$ in the external utility table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1 (b), the external utility of item $A$, $eu(A)$, is 3.

- The utility of item $i_p \subseteq I$ in transaction $T_q$, $u(i_p, T_q)$, is the quantitative measure of utility for item $i_p$ in transaction $T_q$, defined as $eu(i_p) \times c(i_p, T_q)$.
- The utility of itemset $X$ in transaction $T_q$, $u(X, T_q)$, is $\sum_{i_p \in X} u(i_p, T_q)$, where $X \subseteq T_q$.
- The utility of itemset $X$, $u(X)$, is defined as $\sum_{X \subseteq T_q \in DB} u(X, T_q)$.

Utility mining is to find all the itemsets whose utility values are beyond a user specified threshold. An itemset $X$ is a high utility itemset, if $u(X) \geq \varepsilon$, where $\varepsilon$ is the minimum utility threshold. In Table 1, $u(\{A, D\}) = u(\{A, D\}, T_9) + u(\{A, D\}, T_10) = 9 + 27 = 36$, and $u(\{A, D, E\}) = u(\{A, D, E\}, T_9) + u(\{A, D, E\}, T_10) = 14 + 32 = 46$. If $\varepsilon$ is set to be 40, $\{A, D\}$ is a low utility itemset and $\{A, D, E\}$ is a high utility itemset. That is, the “downward closure property” does not hold in the utility mining model.

2.2 Privacy Preserving Mining on Association Rules

The sanitizing algorithms for the privacy preserving mining on association rules can be divided into two categories: (1) Data-Sharing approach and (2) Pattern-Sharing approach.

(1) Data-Sharing approach: the sanitization process acts on the data to remove or hide the group of restrictive association rules that contain sensitive knowledge. Among the algorithms of the Data-Sharing approach, they are classified the following sub-categories [18]: (a) Item Restriction-Based [21], (b) Item Addition-Based [21], and (c) Item Obfuscation-Based [19, 20].

(2) Pattern-Sharing approach: the sanitizing algorithm acts on the rules mined from a database instead of the data itself. Regarding pattern-sharing techniques, the only known approach that falls into this category was introduced in [21].

Rule Restriction-Based: This approach blocks some inference channels to ensure that an adversary cannot reconstruct restrictive rules from the non-restrictive ones. In doing so, we can reduce the inference channels and minimize the side effect. For example, DSA Algorithm proposed by Oliveira et al. [16]. MINSS, MINNS, SIMBLK, and BINFCH by Wang et al. [23, 24].

292
3 Proposed Algorithms

In this section, we present two algorithms for the privacy preserving utility mining: (1) Hiding High Utility Item First Algorithm (HHUIF) and (2) Maximum Sensitive Itemsets Conflict First Algorithm (MSICF). First of all, we give the definitions of some notations which will be used in the rest of the paper.

Definition 1: (Sensitive itemset) Let \( I = \{i_1, i_2, \ldots, i_m\} \) be a set of items, where \( m \) is the total number of items, \( DB = \{T_1, T_2, \ldots, T_n\} \) be a set of transactions, \( \varepsilon \) be the minimum utility threshold, and \( L \) be a set of all high utility itemsets for \( \varepsilon \). Let \( U = \{S_1, S_2, \ldots, S_l\} \) be a subset of \( L \), where \( S_i \), called a sensitive itemset, is an itemset which needs to be hidden according to some security policies.

Definition 2: (Count of conflict) The count of conflict of an item \( i_p \) in the sensitive itemsets \( U \) denotes as \( \text{Icount}_{i_p}(U) \), where \( \text{Icount}_{i_p}(U) = |\{S_i \in U | i_p \in S_i\}| \).

3.1 Hiding High Utility Item First Algorithm (HHUIF)

For each sensitive itemset, the sanitization process decreases the utility value of the sensitive itemset by modifying the quantity value of an item with the highest utility value in some transaction containing the sensitive itemset. The process repeats until the utility values of all sensitive itemsets are below the minimum utility threshold. The pseudo-code of the HHUIF algorithm is as follows:

```
Algorithm HHUIF
Input: the original database \( DB \); the minimum utility threshold \( \varepsilon \); the sensitive itemsets \( U = \{S_1, S_2, \ldots, S_l\} \).
Output: the sanitized database \( DB' \) so that \( S_i \) cannot be mined.
1 For each sensitive itemset \( S_i \in U \)
2 \( \text{diff} = u(S_i) - \varepsilon \)
3 While (\( \text{diff} > 0 \)) {
4 \( (i_p, T_j) = \text{arg max}_{i \in S_i, T_j \subseteq T}(u(i, T)) \)
5 modify \( o(i_p, T_j) \) with
6 \( \begin{cases} 
\text{if } u(i_p, T_j) < \text{diff} \\
\text{if } u(i_p, T_j) > \text{diff}
\end{cases} \)
7 \( \text{diff} = \begin{cases} 
\text{diff} - u(i_p, T_j) & \text{if } u(i_p, T_j) < \text{diff} \\
0 & \text{if } u(i_p, T_j) > \text{diff}
\end{cases} \)
8 End
```

3.2 Maximum Sensitive Itemsets Conflict First Algorithm (MSICF)

To reduce the number of the modified items from the original database, MSICF selects an appropriate item which has the maximum count of conflict among items in the sensitive itemsets. First, MSICF calculates \( \text{Icount}_{i_p}(U) \) and sort \( i_p \) in decreasing order of \( \text{Icount}_{i_p}(U) \). Second, it finds a transaction \( T_j \), such that \( u(i_p, T) \) is maximal. Then, it modifies \( o(i_p, T_j) \).

HSICF sanitizes the transactions containing the sensitive itemsets repeatedly until \( \text{diff} \leq 0 \). Due to the paper page limit, we omit the detail of HSICF here.

4. Experimental Results

To measure the effectiveness of HHUIF and MSICF algorithms, the experiments were conducted on two synthetic datasets. All experiments were performed on a Dell workstation with 3.40 GHz Intel Pentium 4 processor and 2 GB of main memory, running the Windows XP Professional. We first applied ShFSM algorithm [8] to extract all high utility itemsets from the datasets. From the high utility itemsets found in each dataset, we randomly selected two sets of sensitive itemsets with size of five and ten.
For the data deliverer and introduce 
MinUtility=7000 
for the data receiver.

Different items, and the average length of transactions is 9 by default. The second one contains 1,000 transactions with 50 different items, and the average length of transactions is 9 by default.

4.1 Datasets

We used the IBM synthetic data generator [7] to generate two synthetic datasets: data.ntrans1.nitems0.1 and data.ntrans1.nitems0.05. The first one contains 1,000 transactions with 100 distinct items, and the average length of transactions is 9 by default. The second one contains 1,000 transactions with 50 different items, and the average length of transactions is 9 by default.

4.2 Effectiveness Measurement

To measure the effectiveness of our proposed algorithms, we adopt the performance measures introduced by Oliveira and Zaïane [20]. We summarize the performance measures as follows:

(a) Hiding Failure (HF): the ratio of sensitive itemsets that are disclosed before and after the sanitization process. The hiding failure is calculated as follows:

\[ HF = \frac{|U(DB)| \setminus |U(DB')|}{|U(DB)|} \]  

where \(U(DB)\) and \(U(DB')\) denote the sensitive itemsets discovered from the original database \(DB\) and the sanitized database \(DB'\), respectively. The cardinality of a set \(S\) is denoted as \(|S|\).

(b) Miss Cost (MC): the ratio of legitimate itemsets that are hidden by accident after the sanitization. The misses cost is measured as follows:

\[ MC = \frac{|U(DB) \setminus \sim U(DB')|}{|U(DB)|} \]  

where \(\sim U(DB)\) and \(\sim U(DB')\) denote the non-sensitive itemsets discovered from the original database \(DB\) and the sanitized database \(DB'\), respectively.

(c) Difference between the Original and Sanitized Databases: the difference between the original database \(DB\) and the sanitized database \(DB'\), denoted by \(\text{diff}(DB, DB')\), is given by:

\[ \text{diff}(DB, DB') = \frac{|DB - DB'|}{|DB|} \]  

4.3 Comparison of Hiding Failure

| Table 2. Hiding Failure (HF) for data.ntrans1.nitems0.1 with \(|U|=5\) |
|-----------------|----------------|----------------|----------------|
| \(\delta\) \(\text{HHUIF}\) | \(\delta\) \(\text{HHUIF}\) | \(\delta\) \(\text{MSICF}\) | \(\delta\) \(\text{MSICF}\) |
| 2000 100.00% | 2000 100.00% | 3000 100.00% | 3000 100.00% |
| 3000 100.00% | 3000 100.00% | 4000 100.00% | 4000 100.00% |
| 4000 100.00% | 4000 100.00% | 5000 100.00% | 5000 100.00% |

4.4 Comparison of Miss Cost

Table 4 and Table 5 show the performance of HHUIF and MSICF algorithms on the misses cost over various MinUtility and Expecting Minimum Utility Threshold \(\delta\). Applied to data.ntrans1.nitems0.1
and data.ntrans1.nitems0.05 with 5 sensitive itemsets, respectively. We explained three different cases of between MinUtility and δ as follows:

1. If MinUtility is greater than δ, the miss cost must be 0%. That is, all sensitive itemsets will not be mined.
2. If MinUtility is equal to δ, the miss cost may occur. The reason of occurring misses cost is that the sanitized items may be contained in other high utility itemsets. For example, in data.ntrans1.nitems0.1 with MinUtility = 7000, the utility value of high utility itemset {3, 36, 48} is 7037. While we hid the sensitive itemset {29, 36} with deleting the quantity of item 36, it led to the utility value of high utility itemset {3, 36, 48} below 7000.
3. If MinUtility is less than δ, the miss cost usually occur.

### Table 4. Miss costs (MC) for data.ntrans1.nitems0.1 with |U| = 5

<table>
<thead>
<tr>
<th>MinUtility</th>
<th>MC</th>
<th>MinUtility</th>
<th>MC</th>
<th>MinUtility</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>62.04%</td>
<td>5000</td>
<td>50.00%</td>
<td>6000</td>
<td>35.71%</td>
</tr>
<tr>
<td>3000</td>
<td>93.93%</td>
<td>4000</td>
<td>0.00%</td>
<td>5000</td>
<td>0.00%</td>
</tr>
<tr>
<td>2000</td>
<td>0.00%</td>
<td>3000</td>
<td>0.00%</td>
<td>4000</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### Table 5. Miss costs (MC) for data.ntrans1.nitems0.05 with |U| = 5

<table>
<thead>
<tr>
<th>MinUtility</th>
<th>MC</th>
<th>MinUtility</th>
<th>MC</th>
<th>MinUtility</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>70.37%</td>
<td>5000</td>
<td>55.26%</td>
<td>6000</td>
<td>35.71%</td>
</tr>
<tr>
<td>3000</td>
<td>12.96%</td>
<td>4000</td>
<td>13.16%</td>
<td>5000</td>
<td>0.00%</td>
</tr>
<tr>
<td>2000</td>
<td>0.00%</td>
<td>3000</td>
<td>0.00%</td>
<td>4000</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### 4.5 Comparison of Difference between the Original and Sanitized Databases

Table 6 and Table 7 show the performance of differences between DB and DB’ with HHUIF and MSICF algorithms over various MinUtility applied to data.ntrans1.nitems0.1 and data.ntrans1.nitems0.05 with 5 sensitive itemsets, respectively.

### Table 6. Differences between D and D’ for data.ntrans1.nitems0.1 with |U| = 5

<table>
<thead>
<tr>
<th>MinUtility</th>
<th>HHUIF</th>
<th>MSICF</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>1.93%</td>
<td>0.81%</td>
</tr>
<tr>
<td>4000</td>
<td>0.81%</td>
<td>2.54%</td>
</tr>
<tr>
<td>5000</td>
<td>2.95%</td>
<td>0.81%</td>
</tr>
</tbody>
</table>

### Table 7. Difference between DB and DB’ for data.ntrans1.nitems0.05 with |U| = 5

<table>
<thead>
<tr>
<th>MinUtility</th>
<th>HHUIF</th>
<th>MSICF</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000</td>
<td>0.61%</td>
<td>1.32%</td>
</tr>
<tr>
<td>7000</td>
<td>1.32%</td>
<td>1.83%</td>
</tr>
<tr>
<td>8000</td>
<td>1.73%</td>
<td>1.83%</td>
</tr>
</tbody>
</table>

### 5. Conclusions

Data quality plays a quite important role in the mining process. Accurate input data bring meaningful mining results. When users provide false data to protect their privacy will produce other problems at the same time. In strategic alliance cases, companies need to share information with others and protect their own business confidential as well. In this study, we present HHUIF and MSICF algorithms to reduce the impact on the source database for the Privacy Preserving Utility Mining. These algorithms are based on modifying the database transactions containing the sensitive itemsets so that the utility value can be reduced below the given threshold. There is no possible way to reconstruct the original database from the sanitized one. The experimental results show that HHUIF has the lower miss costs than MSICF does in two synthetic datasets. On the other hand, MSICF has the lower difference between the original and sanitized databases than HHUIF has, except in the case where MinUtility = 4000.

### 6. References


