PAPER

Privacy Preserving Using Dummy Data for Set Operations in Itemset Mining Implemented with ZDDs

Keisuke OTAKI[†a)], ***Student Member***, **Mahito SUGIYAMA**[††], ***and*** **Akihiro YAMAMOTO**[†], ***Nonmembers***

SUMMARY We present a privacy preserving method based on inserting dummy data into original data on the data structure called *Zero-suppressed BDDs* (ZDDs). Our task is *distributed itemset mining*, which is frequent itemset mining from horizontally partitioned databases stored in distributed places called sites. We focus on the fundamental case in which there are two sites and each site has a database managed by its owner. By dividing the process of distributed itemset mining into the set union and the set intersection, we show how to make the operations secure in the sense of undistinguishability of data, which is our criterion for privacy preserving based on the already proposed criterion, p-indistinguishability. Our method conceals the original data in each operation by inserting *dummy data*, where ZDDs, BDD-based directed acyclic graphs, are adopted to represent sets of itemsets compactly and to implement the set operations in constructing the distributed itemset mining process. As far as we know, this is the first technique which gives a concrete representation of sets of itemsets and an implementation of set operations for privacy preserving in distributed itemset mining. Our experiments show that the proposed method provides undistinguishability of dummy data. Furthermore, we compare our method with *Secure Multiparty Computation* (SMC), which is one of the well-known techniques of secure computation.

key words: *privacy-preserving data mining, distributed itemset mining, dummy data, set operations, zero-suppressed bdds*

1. Introduction

Privacy-Preserving Data Mining (PPDM) is one of the growing fields in Knowledge Discovery where *data anonymization* and *secure computation* are mainly discussed for finding useful knowledge without information leakages. Today, it is becoming more and more general that many users participate a specific mining task. Each user has his/her own data stored in a *site* and try to extract knowledge from the union of all users' data without revealing their information to other users, since some opponents, called *adversaries*, may participate in the mining task.

In this paper, we present a new privacy preserving method to overcome two incompatible requirements, discovering and preserving knowledge, in such a scenario. In particular, we focus on *itemset mining*, which is a fundamental task in Knowledge Discovery. In the task, we want to find sets of items occurring frequently, called *frequent itemsets*, from a given database. Since the first algorithm *Apriori* was proposed by Agrawal *et al.* [1], many algorithms have already been proposed like FP-growth by Pei *et al.* [2] and LCM by Uno *et al.* [3]. When the database is divided into several parts and each part is stored in a distributed site, the task is called *distributed itemset mining*. Most algorithms for the original itemset mining can be extended for the distributed cases like FDM proposed by Cheung *et al.* [4], which is a natural extension of the Apriori.

In distributed itemset mining, we have to take into account two additional issues: communication cost and privacy of data, which do not appear in the original itemset mining. To date, various distributed itemset mining algorithms have been proposed for avoiding the two issues. Lucchese *et al.* [5] proposed to use *closed itemsets* for decreasing communication cost between sites, where the concepts of closed itemsets was introduced by Pasquier *et al.* [6] as a lossless compression of frequent itemsets. On the other hand, for preserving privacy of data, Kantarcioglu *et al.* [7] proposed an extension of the FDM algorithm based on *Secure Multiparty Computation* (SMC), for the disjoint union of horizontally partitioned databases called the *entire database*. Recently, Kuno *et al.* [8] integrated the above methods and adopted encryption during communications of closed itemsets between sites for preserving privacy.

However, the previous works do not provide precise implementations of operations for itemset mining although they are important for estimating information leakages based on actual operations. In this paper, we divide the process of itemset mining into two set operations, the *set union* and the *set intersection*, and use closed itemsets in the same way as the previous work [5] for decreasing communication cost. Moreover, they do not give representation of itemsets, which is also important for practical evaluaitons of information leakages. Here we adopt *Zero-suppressed BDDs* (ZDDs), which are BDD-based directed acyclic graphs. Sets of itemsets can be represented compactly by ZDDs [9], [10] and, therefore, we can reduce the encryption cost using ZDDs, which becomes high for massive databases and is known as one of important problems in privacy preserving [7]. The key strategy of privacy preserving is to insert *dummy data* into the original data to conceal them. Since ZDDs give compact representation of such mixed data including original and dummy, we can effectively and efficiently deal with preserving privacy for ZDDs. As far as we know, this is the first technique which gives a concrete data structure for preserving privacy.

As our criterion of privacy preserving, we define *undis-*

Manuscript received March 27, 2012.
Manuscript revised July 21, 2012.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
[††]The author is with Max Planck Institute for Developmental Biology and the Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany.
a) E-mail: ootaki@iip.ist.i.kyoto-u.ac.jp

DOI: 10.1587/transinf.E95.D.3017

tinguishability for privacy of data in distributed itemset mining. We guarantee to obtain inability to identify whether data treated in the mining actually come from the original data by dummy data.

This paper is constructed as follows: We prepare concepts and definitions of distributed itemset mining and closed itemsets in Sect. 2. In Sect. 3, we explain the method of inserting dummy data and why privacy of data is preserved with such dummy data. We also show how to extract expected results after set operations with dummy data and theoretically prove the correctness of our method for the set union and the set intersection. In Sect. 4, we explain an implementation of our method using ZDDs and introduce a criterion of privacy preserving called undistinguishability. We experimentally show that the method preserves privacy in the sense of undistinguishability in Sect. 5. After showing the comparison between our method and SMC in Sect. 6, we give our conclusion in Sect. 7.

2. Distributed Itemset Mining Using Closed Itemsets

Here we give concepts and notations in distributed itemset mining and discuss the relationship between set operations and itemset mining. We explain how to divide the process into set operations by *closed itemsets* and the operation $\otimes$ with defining them. We also confirm the method to obtain the globally frequent itemsets.

2.1 Overview of Distributed Itemset Mining

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$ be a finite set of *items* and an itemset X be a subset of $\mathcal{I}$. A transaction $t = (id, X)$ is a pair of an identifier *id* and an itemset X. A *transaction database* DB is a finite set of transactions and the number of transactions is denoted by |DB|, that is, $\mathrm{DB} = \{t_1, t_2, \ldots, t_{|\mathrm{DB}|}\}$. We omit *id* when we treat set operations for transactions. For example, we write $X \in \mathrm{DB}$ if $t \in \mathrm{DB}$ for some transaction $t = (id, X)$.

A transaction database DB is said to be *horizontally partitioned* into m sites $S_1, S_2, \ldots,$ and S_m if DB is partitioned into disjoint subsets $\mathrm{DB}_1, \mathrm{DB}_2, \ldots,$ and DB_m and $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2 \cup \cdots \cup \mathrm{DB}_m$, where each DB_k is stored in the site S_k. Each database DB_k is called a *partial database*, and DB is called the *entire database*. Figure 1 illustrates an example of a horizontally partitioned transaction database $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ on $\mathcal{I} = \{a, b, c, d\}$ and its binary table representation. In this paper, we often regard a transaction database DB on the left-hand side in the figure as a relational table of binary data on the right-hand side. The left most column shows the ID of every transaction. Each of the other columns corresponds to an item $i \in \mathcal{I}$, and each row corresponds to a transaction $t = (id, X)$. We put 1 on (id, i) in the table if for the transaction $t = (id, X)$, the itemset X includes the item i. In other cases, we put 0 on the cell. We say that an itemset X *occurs* in a transaction $t = (id, Y)$ if $X \subseteq Y$, and t is an *occurrence* of X.

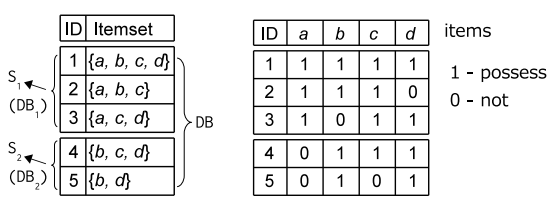

Fig. 1 An example of a horizontally partitioned database $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ on $\mathcal{I} = \{a, b, c, d\}$ (left), and its binary table representation (right).

The number of occurrences of the itemset X in the database DB_k is called the *support* of X in DB_k and it is denoted by $X.\mathrm{sup}_k$. The support of X in the entire database DB is defined as $X.\mathrm{sup} = \sum_k X.\mathrm{sup}_k$ and if $X.\mathrm{sup} \geq \sigma \sum_k |\mathrm{DB}_k|$, the itemset X is said to be *globally frequent* with respect to σ. The parameter σ $(0 \leq \sigma \leq 1)$ is a real number, called *minimum support*, and it is given in an actual scenario of itemset mining. The goal of *distributed itemset mining* is to find all globally frequent itemsets from the entire database DB with a given minimum support σ.

For simplicity, we assume that an *item* is a natural number. We call a set $\{m, m+1, \ldots, n\}$ an *interval* and write it as $[m, n]$ where $m < n$. We say that DB is *on* $[m, n]$ if for all $X \in \mathrm{DB}$, $X \subseteq [m, n]$. In the following, we fix an interval $\mathcal{I} = [1, R]$ and assume that an entire database DB is horizontally partitioned into two databases DB_1 and DB_2 on $\mathcal{I}$, so that $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$. Each partial database DB_1, DB_2 is stored in each site S_1, S_2, respectively. Note that our method can be easily extended to the cases in which there are more than three sites in the task. For example, we can use tournaments to treat many sites with binary operations.

2.2 Closed Itemsets with Set Operations

Let T be a non-empty subset of a transaction database DB and X be a non-empty itemset. *Closed itemsets* are defined using two subsidiary functions, Com and Occ, as follows:

$$\mathrm{Com}(T) = \{i \in \mathcal{I} \mid i \in t \text{ for all } t \in T\},$$
$$\mathrm{Occ}(X) = \{t \in \mathrm{DB} \mid i \in t \text{ for all } i \in X\}.$$

In addition, the *Galois operator* c is defined as $c = \mathrm{Com} \circ \mathrm{Occ}$. We say that the itemset X is *closed* if $c(X) = X$. For $k \in \{1, 2\}$, we denote the set of closed itemsets on DB_k by C_k, and the set of frequent closed itemsets in C_k by FC_k. Moreover, we denote the set of closed itemsets on DB by C_{12}, and the set of frequent closed itemsets on C_{12} by FC_{12}.

It is well-known that the frequent closed itemsets FC_{12} is a lossless compression of the frequent itemsets in the sense that we can generate all frequent itemsets from FC_{12} [6]. Thus, in distributed itemset mining, it is enough to find FC_{12} instead of all frequent itemsets. For example, if $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ is the database illustrated in Fig. 1, we have the followings:

$$C_1 = \{ac, abc, acd, abcd\},\ \ C_2 = \{bc, bcd\},$$
$$C_{12} = \{b, c, d, ac, bc, bd, bcd, abc, acd, abcd\}.$$

Lucchese *et al.* [5] showed that the set C_{12} is equivalent to $C_1 \otimes C_2$, which is defined as $C_1 \otimes C_2 = C_1 \cup C_2 \cup (C_1 \sqcap C_2)$,

where $C_1 \sqcap C_2 = \{X_1 \cap X_2 \mid X_1 \in C_1, X_2 \in C_2\}$. We assume that the site S_1 has C_1 and the site S_2 has C_2. Because the operator $\otimes$ consists of two set operations, the set union $\cup$ and the set intersection $\cap$, all that we have to do is to preserve privacy of these two set operations with dummy data.

2.3 How to Find Frequent Itemsets

Globally frequent itemsets can be obtained from the set of closed itemsets C_{12} in the following manner. As we mentioned earlier, an itemset X is globally frequent if $X.\sup \geq \sigma \sum_k |\mathrm{DB}_k|$. This inequality can be written as $\sum_k v_k \geq 0$, where $v_k = X.\sup_k - \sigma|\mathrm{DB}_k|$. Thus we can judge whether or not the itemset X is globally frequent by taking summation of v_k in all sites. We can easily preserve privacy of this summation using the method *secure sum* [11].

3. Set Operations with Dummy Data

We propose to insert dummy data into original data for privacy preserving in set operations for distributed itemset mining. For the insertion, we prepare three functions, *shift functions*, *mixing functions*, and *screening functions*. We introduce two parameters $\gamma \geq 1$ and $M \geq R + \gamma + 1$ and expand the original interval $\mathcal{I} = [1, R]$ to $\tilde{\mathcal{I}} = [1, M]$, which we call the *expanded interval*. A *dummy database* is a database on the interval $[m, n]$ with $m < \gamma$ and $R + \gamma < n$, where every itemset is generated randomly. Every transaction in dummy databases is called *dummy data* and every transaction in original databases is called *original data*.

We use a *shift function* to make the expanded interval $\tilde{\mathcal{I}}$. The shift function for an item $i \in \mathcal{I}$ is defined as $shift_\gamma(i) = i + \gamma$. The function is extended for an itemset X as $shift_\gamma(X) = \{shift_\gamma(i) \mid i \in X\}$, and for a database DB as $shift_\gamma(\mathrm{DB}) = \{shift_\gamma(X) \mid X \in \mathrm{DB}\}$.

We use a *mixing function* μ to mix a dummy database into the original database. The mixing function μ gets an original database DB and a dummy database D and returns a new database, denoted by $\mu(\mathrm{DB}, \mathrm{D})$, which consists of only original data and dummy data. Since both databases have identifiers for every transaction, we assign new identifiers for every transaction from both databases DB and D randomly, and it forms $\mu(\mathrm{DB}, \mathrm{D})$ so that every transaction in it has a new and unique identifier. By omitting the identifiers for operations, we regard the mixing step as a set operation. In the case that we use the set union $\cup$ as the mixing function, we denote the mixing step by $\mu(\mathrm{DB}, \mathrm{D}) = \mathrm{DB} \cup \mathrm{D}$.

We use a *screening function* e to remove the dummy data from the result of set operations. If the target set operation is the set intersection $\cap$, the screening function e is id, which is an *identity function*. Otherwise if the target set operation is the set union $\cup$, the screening function e is a *choice function*, which gets a set of itemsets on $[1, M]$ and remove the items on $[1, \gamma]$ and $[R + \gamma + 1, M]$ from each itemset. We will show the correctness of the functions in Proposition 1 and Corollary 1 based on some restrictions.

We show the outline of our method to achieve the set operations with dummy data in Fig. 2. We denote the inputs as DB_1 and DB_2, and the method calculates $\mathrm{DB}_1\ op\ \mathrm{DB}_2$. By this method, we calculate C_{12} from two sets C_1 and C_2 of closed itemsets of two sites S_1 and S_2.

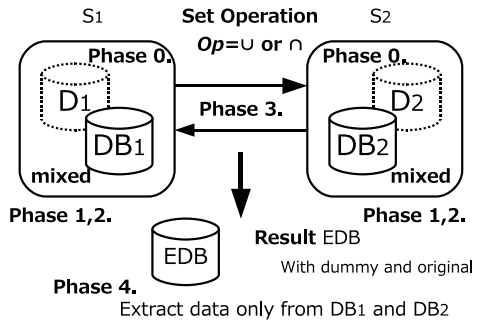

Fig. 2 The outline of inserting dummy data and extraction.

Note. By repeating the following phases, we calculate C_{12} after each site S_1, S_2 prepares C_1, C_2, respectively.

Phase 0. (Preparing) The sites S_1 and S_2 prepare *dummy databases* D_1 and D_2, respectively and independently. They share the parameters γ and M.

Phase 1. (Pre-processing 1) Each site S_k ($k \in \{1, 2\}$) obtains $\mathrm{DB}'_k = shift_\gamma(\mathrm{DB}_k)$ by applying the shift function $shift_\gamma$. Note that no original item remains in the intervals $[1, \gamma]$ and $[R + \gamma + 1, M]$.

Phase 2. (Pre-processing 2) Each site S_k ($k \in \{1, 2\}$) merges a dummy database D_k with DB_k using a mixing function μ and get a new database $\mathrm{EDB}_k = \mu(\mathrm{DB}'_k, \mathrm{D}_k)$.

Phase 3. (Set operation) Both sites S_1 and S_2 execute a set operation $op \in \{\cup, \cap\}$ between EDB_1 and EDB_2. The output $\mathrm{EDB} = \mathrm{EDB}_1\ op\ \mathrm{EDB}_2$ includes both original data and dummy data.

Phase 4. (Post-processing) Each site applies a *screening function* e to the result EDB. To get the answer on the original interval $[1, R]$, it uses the shift function $shift_{(-\gamma)}$ to $e(\mathrm{EDB})$. The result $shift_{(-\gamma)}(e(\mathrm{EDB}))$ becomes the answer on the original interval $[1, R]$.

We show the correctness of our method by proving that we can get $\mathrm{DB}_1 \cup \mathrm{DB}_2$ or $\mathrm{DB}_1 \cap \mathrm{DB}_2$ by our method. Remember that they prepare C_1 and C_2 as their inputs. For simplicity, we divide the expanded interval $\tilde{\mathcal{I}}$ into three intervals $L = [1, \gamma]$, $N = [1 + \gamma, R + \gamma]$, and $H = [R + \gamma + 1, M]$, that is, $\tilde{\mathcal{I}} = L \cup N \cup H$. For the site S_k, dummy databases $\mathrm{D}_k|_L$, $\mathrm{D}_k|_N$, and $\mathrm{D}_k|_H$ on the intervals L, N, and H are prepared so that they satisfy the following conditions.

$$\mathrm{D}_k|_L \subseteq \{t \mid t = (id, X) \text{ for some } id \text{ and } X \subseteq L\},$$
$$\mathrm{D}_k|_N \subseteq \{t \mid t = (id, X) \text{ for some } id \text{ and } X \subseteq N\},$$
$$\mathrm{D}_k|_H \subseteq \{t \mid t = (id, X) \text{ for some } id \text{ and } X \subseteq H\}.$$

We let $\mathrm{D}_k = \mathrm{D}_k|_L \cup \mathrm{D}_k|_H$ for $k \in \{1, 2\}$ and we also let $\mathrm{D}_k|_N = \emptyset$ in this paper. In addition, we assume that $\mathrm{D}_1|_L \cap \mathrm{D}_2|_L = \emptyset$ and $\mathrm{D}_1|_H \cap \mathrm{D}_2|_H = \emptyset$ by setting two parameters γ and M for enlarging two intervals L and H enough. Under those assumptions, the pair of a mixing function $\mu = \cup$ and

a screening function $e = id$ works correctly if we choose dummy databases carefully.

Proposition 1 We adopt $\mu = \cup$ and $e = id$. For $k \in \{1, 2\}$, let $\mathrm{DB}'_k = \mathit{shift}_\gamma(\mathrm{DB}_k)$ and $\mathrm{EDB}_k = \mathrm{DB}'_k \cup (\mathrm{D}_k|_L \cup \mathrm{D}_k|_H)$. For the operation $op = \cap$, it holds that $\mathit{id}\{\mathrm{EDB}_1 \cap \mathrm{EDB}_2\} = \mathit{shift}_\gamma(\mathrm{DB}_1 \cap \mathrm{DB}_2)$.

Proof Proposition 1 is proved as follows:

$$\begin{aligned}
&\mathit{id}\{(\mathrm{DB}'_1 \cup \mathrm{D}_1|_L \cup \mathrm{D}_1|_H) \cap (\mathrm{DB}'_2 \cup \mathrm{D}_2|_L \cup \mathrm{D}_2|_H)\} \\
&\quad = (\mathrm{DB}'_1 \cup \mathrm{D}_1|_L \cup \mathrm{D}_1|_H) \cap (\mathrm{DB}'_2 \cup \mathrm{D}_2|_L \cup \mathrm{D}_2|_H) \\
&\quad = (\mathrm{DB}'_1 \cap \mathrm{DB}'_2) \\
&\qquad \cup \{\mathrm{DB}'_1 \cap (\mathrm{D}_2|_L \cup \mathrm{D}_2|_H)\} \cup \{\mathrm{DB}'_2 \cap (\mathrm{D}_1|_L \cup \mathrm{D}_1|_H)\} \\
&\qquad \cup \{(\mathrm{D}_1|_L \cup \mathrm{D}_1|_H) \cap (\mathrm{D}_2|_L \cup \mathrm{D}_2|_H)\} \\
&\quad = \mathrm{DB}'_1 \cap \mathrm{DB}'_2 = \mathit{shift}_\gamma(\mathrm{DB}_1 \cap \mathrm{DB}_2)
\end{aligned}$$

□

We can easily see that we obtain $\mathrm{DB}_1 \cap \mathrm{DB}_2$ from $\mathit{shift}_\gamma(\mathrm{DB}_1 \cap \mathrm{DB}_2)$ in Phase 4 with the function $\mathit{shift}_{(-\gamma)}$.

For the case $op = \cup$, we need another screening function, called *choice*. It chooses items from the shifted interval $[1 + \gamma, R + \gamma]$ as follows:

$$\mathit{choice}(\mathrm{DB}) = \{t \mid t \in \mathrm{DB} \text{ and } i \in \mathit{shift}_\gamma(\mathcal{I}) \text{ for all } i \in t\}.$$

We can obtain the following corollary.

Corollary 1 For the set operation $\cup$ and the screening function *choice*, we can calculate the set intersection with dummy data as follows:

$$\begin{aligned}
&\mathit{choice}\{(\mathrm{DB}'_1 \cup (\mathrm{D}_1|_L \cup \mathrm{D}_1|_H)) \cup (\mathrm{DB}'_2 \cup (\mathrm{D}_2|_L \cup \mathrm{D}_2|_H))\} \\
&\quad = \mathrm{DB}'_1 \cup \mathrm{DB}'_2 = \mathit{shift}_\gamma(\mathrm{DB}_1 \cup \mathrm{DB}_2).
\end{aligned}$$

Note that we can insert dummy data into the interval $[1 + \gamma, R + \gamma]$. This is why we store sets of itemsets as ZDDs (we give explanation about them in the next section), where itemsets are stored in the form of graphs and transferred to the other site as a list of nodes, where some nodes are merged. We cannot know such merging occurs in mixing databases. Thus when we insert dummy transactions on $[m, n]$ with $m < \gamma$ and $R+\gamma < n$, we cannot find original data from $[m, n]$ as items during set operations. Moreover, such dummy data do not affect the original data because ZDDs do not represent items, but sets of itemsets.

4. Implementation and Privacy Criterion with ZDDs

In this section, we present an implementation of our method using ZDDs, invented by Minato [9], and give a privacy criterion, called undistinguishability.

4.1 Representation of Databases Using ZDDs

First, we briefly introduce ZDDs using Fig. 3. In Fig. 3, we

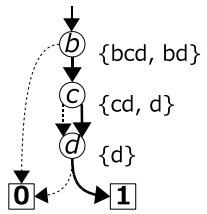

Fig. 3 The database $\mathrm{DB}_2 = \{bcd, bd\}$ in Fig. 1 represented by a ZDD.

illustrate a ZDD representing DB_2 in Fig. 1. The ZDD has one source node labeled with b and two sink nodes represented by square nodes labeled with **0** and **1**. Each circle node represents an item and all items are ordered in each path from the source to the sinks. For example, we have the order $b < c < d$. Each node has two edges. An edge denoted by a dotted line is a 0-edge and an edge denoted by a solid line is a 1-edge. Each 1-edge corresponds to a *possession* of the item assigned to the node. In ZDDs, each path from the source node to the sink node labeled with **1** corresponds to an itemset. Nodes on ZDDs are merged by the rules:

- If the 1-edge is directly connected to the constant **0**, we remove the node, and
- if a tuple (v, p_0, p_1) is isomorphic to an another tuple (v', p'_0, p'_1), then we merge them into one,

where two nodes are *isomorphic* if they have the same item v and connects to the same nodes both on the 0-edges and on the 1-edges. The first rule is the difference between ZDDs and BDDs. By applying it to directed graphs which represent sets of itemsets, we can remove a lot of nodes because usually each transaction consists of a small number of items. In addition, the second rule is also the difference between ZDDs and related structures like FP-trees and prefix trees, that is, the isomorphic nodes are merged on ZDDs. We can remove redundant nodes in the graphs. These are why we can represent sets of itemsets compactly. Refer the literature [9], [10], [12], more precisely. As seen such previous works, ZDDs are widely adopted for itemset mining and related tasks, but there exists no methods for preserving privacy with ZDDs. Because ZDDs could represent a large database, we adopt dummy based methods.

We can also represent a ZDD by a list of tuples (v, p_0, p_1), where v is an item, p_0 is the 0-edge, and p_1 is the 1-edge. The sink nodes labeled with **0** and **1** are represented with $(-1, 0, 0)$ and $(-1, 1, 1)$, respectively. Well-known generic ZDD interpreters have a *node table* [9] to manage the list of tuples representing ZDDs. We show an example of the node table in Fig. 4. Nodes on ZDDs are stored in each row in the node table. Operations of a ZDD are executed by using the list of tuples corresponding to each node of a ZDD [10]. As an example of an algorithm, the set union of two sets represented by two ZDDs $A = (A.top, A_0, A_1)$ and $B = (B.top, B_0, B_1)$ is shown in Fig. 5, where *GetNode* is a function to make a new node. Two tuples A and B show the root nodes of the node tables, and the pointers show the sub-graphs of each node. For ex-

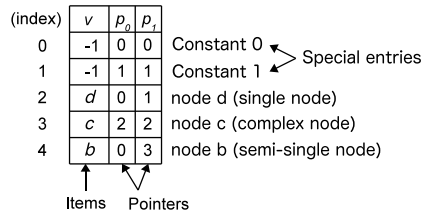

Fig. 4 Example of a node table which represents a ZDD in Fig. 3.

Algorithm: ZDD-UNION

Input: ZDDs $A = (A.top, A_0, A_1)$ and $B = (B.top, B_0, B_1)$
Output: A ZDD $A \cup B$
Procedure:

0. $j = A.top - B.top$
1. **If** $j = 0$ **then** $R = GetNode(A.top, A_0 \cup B_0, A_1 \cup B_1)$
2. **else if** $j < 0$ **then** $R = GetNode(A.top, A_0 \cup B, A_1)$
3. **else** $R = GetNode(B.top, A \cup B_0, B_1)$
4. return R

Fig. 5 The algorithm ZDD-UNION(A, B) which performs the set union where the items are represented by natural numbers.

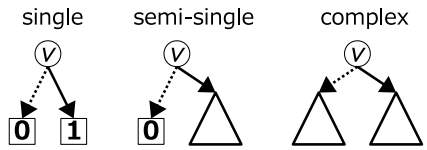

Fig. 6 Three Types of Nodes on ZDDs. The item v is assigned to the root node of those nodes, and the triangles indicate other ZDDs.

ample, $A.top$ is the item of a ZDD in the root node and A_0 and A_1 show sub-graphs.

A node of a ZDD held in one site would be transmitted to the other site, which means that the other site can use the transmitted information for judging whether or not the transmitted node is in the original data or in the dummy data. To analyze privacy preserving in the process precisely, we classify each node into three types as follows:

1. *Single*: Nodes both edges of which are connected to the constant nodes labeled with **0** or **1**.
2. *Semi-single*: Nodes either of edges of which is connected to the constant node.
3. *Complex*: Nodes which are neither single nor semi-single.

These types of nodes are illustrated in Fig. 6, and they are used to define a privacy criterion.

4.2 Privacy Criterion

Vaidya *et al.* [13] proposed that we should address the following two requirements in privacy preserving.

1. We cannot identify an *individual* from revealed information during operations.
2. We cannot construct any attacks to the *individual*.

For the second requirement, we assume *semi-honest* behavior for the participants [14]. In the semi-honest model, the participants obey the given procedures, and our method of inserting dummy data meets the first requirement. In this study, we only consider the case in which there exists only two users for computing closed itemsets. The adversary of a site is the other site. We preserve privacy in the sense of not revealing the structure of ZDDs measured by the number of three types of nodes to the adversary based on undistinguishability. We also assume that we do not want to reveal the ratio of three types of nodes.

For evaluation, we introduce a privacy criterion *undistinguishability* by referring to *p-indistinguishability* introduced by Clifton *et al.* [15] in the contexts of databases and queries to them. *p*-indistinguishability is defined using statistical queries and probabilities based on *computational indistinguishability*. Intuitively, *p*-distinguishability means the property that we cannot identify the individuals from the results of the queries. Based on this criterion, we use *identifiability* of data used in set operations between sites.

The main idea of the criterion is that if there are sufficient dummy data for concealing the original data, we cannot identify original data. In our settings, this means that if we get information during the set operations, we cannot identify whether or not the information is about the original data. More precisely, if we cannot judge it with the probability more than or equal to a half, we regard them as the state of being not identifiable. We formalize this idea as undistinguishability using *well-mixedness* of a pair of databases, which are managed by two sites. In addition, we would like to reduce the size of dummy databases in order to decrease communication costs. For this affair, we basically rely on compact representations of ZDDs.

The number of single nodes (labeled with s), semi-single nodes (labeled with ss), and complex nodes (labeled with c) on a ZDD Z are denoted by $\mathrm{N}(s, Z)$, $\mathrm{N}(ss, Z)$, and $\mathrm{N}(c, Z)$, respectively. For instance, for a ZDD Z represented in the table in Fig. 4, $\mathrm{N}(s, Z) = 1$, $\mathrm{N}(ss, Z) = 1$, and $\mathrm{N}(c, Z) = 1$. Next, we define the *well-mixedness* and the *undistinguishability* as follows:

Definition 1 (Well-mixedness from ZDDs) Let D be a dummy database and DB be an original database, and Z_{D} (resp. Z_{DB}) be a ZDD that represents the database D (resp. DB). The mixed database $\mu(\mathrm{D}, \mathrm{DB})$ with a mixing function μ calculated from D and DB is *well-mixed* if

$$\mathrm{N}(t, Z_{\mathrm{D}}) \geq \mathrm{N}(t, Z_{\mathrm{DB}}) \text{ for all node types } t \in \{s, ss, c\}.$$

We evaluate the mixed database on the viewpoint of representations of its elements.

Definition 2 (Undistinguishability of ZDDs) Let μ, D, and DB be a mixing function, a dummy database, and an original database, respectively. We say that a ZDD Z has *undistinguishability* if Z represents a *well-mixed* database $\mu(\mathrm{D}, \mathrm{DB})$.

This means that a undistinguishable ZDD has enough nodes from both a original database and a dummy database when the site sends nodes to the other site.

5. Evaluation

We evaluate our method using synthetic and real databases. Before evaluating undistinguishability of ZDDs, we check the properties of databases in preliminary experiments to generate the synthetic databases for main experiments. In the main experiments, we used the synthetic databases and the mushroom databases to observe the undistinguishability during the set operations. We performed our experiments using JDD[†] as the ZDD interpreter implemented in Java. All programs are written in Java 6 and executed on Mac OS X 10.6 with Intel Core i5 2.8 GHz and 12 GB memory.

5.1 Preliminary Experiments

We examined the properties and the ratio of three types of nodes of ZDDs by making random databases. Each database consists of 100 itemsets from the interval $[1, 100]$. We generated ZDDs from them, and counted up the number of three types of nodes to examine the ratio among the three types. For generating synthetic databases, we used a parameter α that indicates the average size of itemsets in each transaction because it is a basic future of the databases. For example, if $\alpha = 0.1$ and $\mathcal{I} = [1, 100]$, the mean of the size of itemsets is equal to $100 \times 0.1 = 10$. To make the average size of the synthetic databases similar to that of the mushroom databases, we adjust the parameter α. We let $\alpha = 0.05, 0.1$, and 0.2 to observe changes of the ratio of the three types in a node table representing a ZDD. We measured the ratio of the three node types in the databases and show the results in Fig. 7. The results indicate that the number of complex nodes becomes smaller as α becomes larger. Since the average size of itemsets in mushroom databases is around 20, we prepare the synthetic databases whose average size of itemsets is around 20 by $\alpha = 0.2$. As seen in Fig. 7, the ratio among three types in mushroom databases and dummy databases with letting $\alpha = 0.2$ become similar. Note that the items of them are more randomly located in the interval by randomly generating of databases. Because the types of nodes indicate inner structures of ZDDs, we only consider the ratio among the three types without focusing on inner structures or distributions on ZDDs precisely.

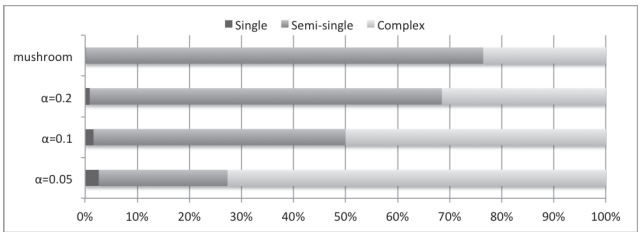

Fig. 7 The ratio of three types of nodes of the mushroom database and the randomly generated databases with $\alpha = 0.05, 0.1$, and 0.2.

5.2 Main Experiments and Results

We chose two types of databases including a real database and a synthetic database for the main experiments, one is *mushroom* from FIMI dataset used in the workshops [16] and [17], and the other is *random-u* which are randomly generated with the parameter $\alpha = 0.2$ in the same manner in the preliminary experiment. The mushroom database is available in the UCI repository [18]. We randomly generated eight databases $ms_1, ms_2, \ldots$, and ms_8 by picking up 100 itemsets for each database from the *mushroom* database and databases $ru_1, ru_2, \ldots$ and ru_8 from *random-u* in the same way. In each ms_k ($k \in \{1, 2, \ldots, 8\}$), we let $L = [1, 100]$, $N = [101, 220]$, and $H = [221, 350]$. Because the interval of original items in ms_k is $[1, 120]$, this setting can be obtained using the shift function with the parameter of shifting $\gamma = 100$. For each ru_k ($k \in \{1, 2, \ldots, 8\}$), we let $L = [1, 1000]$, $N = [1001, 3000]$, and $H = [3001, 4000]$. In both cases, we prepared two dummy databases for both sites S_1 and S_2 from two intervals L and H, respectively. We use the set union $\cup$ for the mixing function μ. For dummy databases, we also required that $\mathrm{D}_1|_L \cap \mathrm{D}_2|_L = \emptyset$ and $\mathrm{D}_1|_H \cap \mathrm{D}_2|_H = \emptyset$. The size of all dummy databases $\mathrm{D}_k|_L$, $\mathrm{D}_k|_L{}'$, $\mathrm{D}_k|_H$, and $\mathrm{D}_k|_H{}'$ is 100.

We tested the well-mixedness of set operations in Phase 0, 1, and 2 (see Sect. 3) because set operations are executed with the sites, the screening phase can be executed simply as seen in Proposition 1.

The results are shown in Tables 1, 2, and 3, where the first row shows the number of nodes from the original database, and the second row shows the number of nodes from the dummy database, and the third row shows the number of nodes created during set operations, that is, intermediate data. For each column, if the value on the second row is more than or equal to that value on the first row, the inequality $\mathrm{N}(t, \mathrm{D}) \geq \mathrm{N}(t, \mathrm{DB})$ holds for all node types $t \in \{s, ss, c\}$. We therefore conclude that the database is well-mixed. Note that, ZDDs used for 2 have more similar items than those used for 1, that is, items of the random-u database are more scattered.

We adopted the set union $\cup$ as a mixing function μ. We used dummy databases $\mathrm{D}_k = \mathrm{D}_k|_L \cup \mathrm{D}_k|_H$ in Table 1.

Table 1 The result for random-u on operation (a) $\cup$ and (b) $\sqcap$ with dummy databases $\mathrm{D}_k = \mathrm{D}_k|_L \cup \mathrm{D}_k|_H$.

(a) for the operation $\cup$

Type	single	semi-single	complex
Original	80	3760	2732
Dummy	173	7916	7504
Intermediate	0	0	362

(b) for the operation $\sqcap$

Type	single	semi-single	complex
Original	80	3760	2732
Dummy	173	7916	7504
Intermediate	1380	210661	10944911

[†]http://javaddlib.sourceforge.net/jdd/

Table 2 The result for mushroom on operation ∪ with dummy databases (a) $D_k = D_k|_L \cup D_k|_H$ and (b) $D_k = D_k|_L \cup D_k|_L' \cup D_k|_H \cup D_k|_H'$.

(a) $D_k = D_k|_L \cup D_k|_H$.

Type	single	semi-single	complex
Original	3	4302	1432
Dummy	59	1706	2963
Intermediate	0	323	354

(b) $D_k = D_k|_L \cup D_k|_L' \cup D_k|_H \cup D_k|_H'$.

Type	single	semi-single	complex
Original	3	4302	1432
Dummy	71	3436	7034
Intermediate	0	323	562

Table 3 The result for mushroom on operation ⊓ with dummy databases (a) $D_k = D_k|_L \cup D_k|_H$ and (b) $D_k = D_k|_L \cup D_k|_L' \cup D_k|_H \cup D_k|_H'$.

(a) $D_k = D_k|_L \cup D_k|_H$.

Type	single	semi-single	complex
Original	3	4302	1432
Dummy	59	1706	2963
Intermediate	193	30282	386765

(b) $D_k = D_k|_L \cup D_k|_L' \cup D_k|_H \cup D_k|_H'$.

Type	single	semi-single	complex
Original	3	4302	1432
Dummy	71	3436	7034
Intermediate	251	60208	1307550

In Tables 2 and 3, we used two types of dummy databases $D_k = D_k|_L \cup D_k|_H$ and $D_k = D_k|_L \cup D_k|_L' \cup D_k|_H \cup D_k|_H'$ to observe the increase of the size of nodes from dummy databases in the node table.

Table 1 shows the results for two picked up databases from ru_k with the set operation ∪ and ⊓. The table shows that two databases are well-mixed.

Tables 2 and 3 show the results for two picked up databases from ms_k with the set operation ∪ and ⊓. In these cases, the undistinguishability is not satisfied since $N(ss, D) \leq N(ss, DB)$ holds for the both cases. However, if we use a dummy database whose size is more than 400 in the mixing process, the undistinguishability could be satisfied as seen in the increment of the number of dummy data from 1706 to 3436. From the experiments, these numbers can be regarded to increase in linear.

5.3 Discussions on Main Experiments

As shown in Table 1, we can satisfy the undistinguishability using dummy data. This result means that we can satisfy it if the size of a dummy database is at least more than or equal to the size of a original database. In contrast, Table 2 shows that we cannot bound the number of nodes from original databases by the number of nodes from dummy databases in terms of semi-single nodes, that is, $N(ss, D) \leq N(ss, DB)$. Thus we need more dummy data to bound the number of semi-single nodes. However, the numbers of single and complex ones were bounded. We can interpret this as follows: For satisfying undistinguishability easily, we need to prepare enough size of dummy data in dummy databases because the number of node from dummy databases increases simultaneously, and it is better that the ratio between three types of dummy data is similar to that of original data.

We conclude that our proposed method can be effectively used for concealing original data to satisfy undistinguishability if we prepare sufficient dummy data.

6. Comparison to SMC

We analyze our method with comparing it to SMC.

6.1 SMC on ZDDs

Secure Multiparty Computation (SMC) [14] is one of subfields of cryptography to provide methods for secure computations among many sites, and it is also called *Multi-Party Computation* (MPC). In this paper, we only consider the case that there are two sites, and this case is already investigated as *Two-Party Computation* (2PC). In SMC, we can use *Secure Function Evaluation* (SFE) as basic tools to construct them, and it gives us fundamental secure operations like *secure sum* and *secure boolean and/or*.

We can use them to calculate set operations, the set union and the set intersection, because they can be regarded as boolean operations, the boolean or, denoted by ∨, and the boolean and, denoted by ∧, on bit sequences, which represent possessions of items in each bit.

A ZDD Z represents a boolean function $f_Z(x_1, \ldots, x_m)$ in which $f_Z(x_1, \ldots, x_m) = 1$ if its inputs $(x_1, x_2, \ldots, x_m)$ is included in Z as an itemset X, where an item x_j belongs to X if $x_j = 1$. An n-arity boolean function f has a truth table T_f whose length is 2^n. We therefore can execute set operations with bit sequences with the truth table T_f and boolean operations ∨ and ∧.

We give examples of bit sequences representing sets of itemsets to provide set operations with SFE. Let DB_1 and DB_2 be databases given in Fig. 1. The databases DB_1 and DB_2 can be regarded as a set of sequences $\{1111, 1110, 1011\}$ and $\{0111, 0101\}$, respectively, as seen in the binary table of the transaction database DB. Sets of closed itemsets C_1 and C_2 can be represented as a set of binary sequences: $C_1 = \{1010, 1110, 1011, 1111\}$ and $C_2 = \{0110, 0111\}$, respectively. Thus a ZDD Z can be regarded as a truth table T of the boolean function of Z. For example, $C_1 = \{1010, 1110, 1011, 1111\}$ shows a boolean function $f_{C_1}(a, b, c, d) = a\bar{b}cd \vee abc\bar{d} \vee a\bar{b}cd \vee abcd$, and $C_2 = \{0110, 0111\}$ shows the function $f_{C_2}(a, b, c, d) = \bar{a}bcd \vee \bar{a}bcd$. Let the truth tables of f_{C_1} and f_{C_2} be $T_{f_{C_1}}$ and $T_{f_{C_2}}$, respectively. We can calculate the set union $C_1 \cup C_2$ by $T_{f_{C_1}} \vee T_{f_{C_2}}$ and the set intersection $C_1 \cap C_2$ by $T_{f_{C_1}} \wedge T_{f_{C_2}}$. Thus, for set operations with SFE, all we have to do is to prepare boolean operations ∨ and ∧.

6.2 Comparison

We compare our methods based on dummy data and SFE for

set operations. In terms of privacy preserving, both methods provide similar results. Our method provides *undistinguishability* defined in Sect. 4.2, which is a criterion based on *identifiability* of dummy data and original data. In contrast, SMC provides the property called *indistinguishability* in the area of cryptography.

By comparison with SMC, the advantage of our method is compact representation of databases, that is, sets of itemsets using ZDDs. Even though our method inserts a lot of dummy data to conceal original data to obtain undistinguishability, it can be treated compactly by adopting ZDDs for representation of a set of itemsets. In contrast, if we use a SMC-based method to use bit operations as shown in the above subsection, vit is difficult to compress itemsets because SFE provides only simple operations. Trivially, if we calculate set operations in the form of bit operations, the resulting bit sequences could become longer exponentially.

7. Conclusion and Future Works

In this paper, we have proposed a new method of inserting dummy data for privacy preserving of set operations used in distributed itemset mining. We also have proposed to use ZDDs for data representation. Our experiments show that our method could provide undistinguishability of data in ZDDs for the worst case. Note that we also consider how to deal with ZDDs in the SMC framework. Introducing more complex mixing functions and screening functions to satisfy more difficult criteria is a our future work. In addition, considering probabilistic properties of dummy data is an another future work. We will investigate and introduce *probability density functions* in order to characterize the dummy data. By introducing them, we could correlate our study based on dummy data with recent studies based on *differential privacy* [19]. We also consider correspondences between distributions on databases and those on ZDDs. Dummy data can be used for other tasks in PPDM for satisfying undistinguishability.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," Proc. International Conference on Very Large Data Bases (VLDB'94), pp.487–499, 1994.
[2] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," SIGMOD Record, vol.29, pp.1–12, 2000.
[3] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "LCM: An efficient algorithm for enumerating frequent closed item sets," Proc. Workshop on Frequent Itemset Mining Implementations (FIMI '03), 2003.
[4] D. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," Proc. 4th International Conference on Parallel and Distributed Information Systems (PDIS'96), pp.31–42, 1996.
[5] C. Lucchese, S. Orlando, and R. Perego, "Distributed mining of frequent closed itemsets: Some preliminary results," Proc. 8th International Workshop on High Performance and Distributed Mining (HPDM'05), 2005.
[6] R.T.N. Pasquier, Y. Bastide, and L. Lakhal, "Discovering frequent closed itemsets for association rules," Proc. 7th International Conference on Database Theory (ICDT'99), pp.398–416, 1999.
[7] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," IEEE Trans. Knowl. Data Eng., vol.16, no.9, pp.1026–1037, 2004.
[8] S. Kuno, K. Doi, and A. Yamamoto, "Frequent closed itemset mining with privacy preserving for distributed databases," Proc. ICDM Workshops on Privacy Aspects of Data Mining (PADM'10), pp.483–490, 2010.
[9] S. Minato, Binary Decision Diagrams and Applications for VLSI CAD, Springer, 1996.
[10] S. Minato, "Zero-suppressed BDDs and their applications," STTT, vol.3, no.2, pp.156–170, 2001.
[11] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M.Y. Zhu, "Tools for privacy preserving data mining," SIGKDD Explorations, vol.4, no.2, pp.28–34, 2002.
[12] S. Minato and H. Arimura, "Frequent closed item set mining based on zero-suppressed BDDs," Information and Media Technologies, vol.2, no.1, pp.309–316, 2007.
[13] J. Vaidya, C.W. Clifton, and M.Y. Zhu, Privacy Preserving Data Mining (Advances in Information Security), Springer-Verlag, 2005.
[14] O. Goldreich, Foundations of Cryptography - Volume 2, Cambridge University Press, 2004.
[15] C. Clifton, M. Kantarcioglu, and J. Vaidya, "Defining privacy for data mining," Proc. National Science Foundation Workshop on Next Generation Data Mining (NGDM '02), pp.126–133, 2002.
[16] B. Goethals and M.J. Zaki, eds., FIMI '03, Frequent Itemset Mining Implementations, Proc. ICDM 2003 Workshop on Frequent Itemset Mining Implementations, CEUR Workshop Proceedings, vol.90, CEUR-WS.org, 2003.
[17] R. Bayardo, B. Goethals, and M.J. Zaki, eds., FIMI '04, Proc. IEEE ICDM Workshop on Frequent Itemset Mining Implementations, CEUR Workshop Proc., vol.126, CEUR-WS.org, 2004.
[18] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.
[19] C. Dwork, "Differential privacy," ICALP, vol.2, pp.1–12, 2006.

Keisuke Otaki received the B.E. degree in Engineering from Kyoto University in 2011. He is now a master course student at Graduate School of Informatics, Kyoto University. His research interests are Knowledge Discovery, in particular, Privacy-Preserving Data Mining, and analyzing generative models of data. He is a student member of IPSJ and JSAI.

Mahito Sugiyama received his B.E., M.E., and Ph.D. degrees from Kyoto University in 2006, 2008, and 2012, respectively. He is currently a researcher at the Max Planck Institute for Developmental Biology and the Max Planck Institute for Intelligent Systems. Since 2010 until 2012 he has been a research fellow of the Japan Society for the Promotion of Science. His research interest is discretization in machine learning and data analysis. He is a member of JSAI.

Akihiro Yamamoto received the B.S. degree from Kyoto University in 1985, and Dr.Sci. degree from Kyushu University in 1990. Currently, he is a Professor of the Department of Intelligence Science and Technology, Graduate School of Informatics at Kyoto University. He has made research contributions to foundations of intelligence science, with a particular focus on application of mathematical logic to machine learning. His recent research interest includes developing machine learning theory with discrete mathematics, computational algebra, and computational calculus. He is a member of JSAI, IPSJ, and JSSST.