Association for Information Systems

AIS Electronic Library (AISeL)

ICEB 2005 Proceedings

International Conference on Electronic Business (ICEB)

Winter 12-5-2005

# Privacy-Preserving Sequential Pattern Mining Over Vertically Partitioned Data

Justin Zhan

Stan Matwin

LiWu Chang

# Privacy-Preserving Sequential Pattern Mining Over Vertically Partitioned Data

Justin Zhan[1], Stan Matwin[2], LiWu Chang[3]
[1,2]School of Information Technology & Engineering, University of Ottawa, Canada
[3]Center for High Assurance Computer Systems, Naval Research Laboratory, USA
{zhizhan, stan}@site.uottawa.ca, lchang@itd.nrl.navy.mil

**Abstract:** Privacy-preserving data mining in distributed environments is an important issue in the field of data mining. In this paper, we study how to conduct sequential patterns mining, which is one of the data mining computations, on private data in the following scenario: Multiple parties, each having a private data set, want to jointly conduct sequential pattern mining. Since no party wants to disclose its private data to other parties, a secure method needs to be provided to make such a computation feasible. We develop a practical solution to the above problem in this paper.

**Keywords:** Privacy, security, sequential pattern mining.

## I. Introduction

Data mining and knowledge discovery in databases is an important research area that investigates the automatic extraction of previously unknown patterns from large amounts of data. They connect the three worlds of databases, artificial intelligence and statistics. The information age has enabled many organizations to gather large volumes of data. However, the usefulness of this data is negligible if meaningful information or knowledge cannot be extracted from it. Data mining and knowledge discovery, attempts to answer this need. In contrast to standard statistical methods, data mining techniques search for interesting information without demanding a priori hypotheses. As a field, it has introduced new concepts and are becoming more and more popular with time.

One of important computations is sequential pattern mining [1, 8, 2, 7, 3], which is concerned of inducing rules from a set of sequences of ordered items. The main computation in sequential pattern mining is to calculate the support measures of sequences by iteratively joining those sub-sequences whose supports exceed a given threshold. In each of above works, an algorithm is provided to conduct such a computation assume that the original data are available. However, conducting such a mining without knowing the original data is challenging.

Generic solutions for any kind of secure collaborative computing exist in the literature [4, 5, 6]. These solutions are the results of the studies of the Secure Multi-party computation problem [9, 5, 6, 4], which is a more general form of secure collaborative computing. However, the proposed generic solutions are usually impractical. They are not scalable and cannot handle large-scale data sets because of the prohibitive extra cost in protecting data secrecy. Therefore, practical solutions need to be developed. This need underlies the rationale for our research.

The paper is organized as follows: Section 2 discusses the related work. We then formally defines the mining sequential patterns on private data problem in Section 3. In Section 4, we describe our secure protocols. Section 5 analyzes the security and communication cost of our solution. We give our conclusion in Section 6.

## II. Mining Sequential Patterns on Private Data

### II.1 Background

Data mining includes a number of different tasks. This paper studies the sequential pattern mining problem. Since its introduction in 1995 [1], the sequential pattern mining has received a great deal of attention. It is still one of the most popular pattern-discovery methods in the field of Knowledge Discovery. Sequential pattern mining provides a means for discovering meaningful sequential patterns among a large quantity of data. For example, let us consider the sales database of a bookstore. The discovered sequential pattern could be like "70% of people who bought Harry Porter also bought Lord of Ring at a later time". The bookstore can use this information for shelf placement, promotions, etc.

In the sequential pattern mining, we are given a database D of customer transactions. Each transaction consists of the following fields: customer-ID, transaction-time, and the items purchased in the transaction. No

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 489 - 495.

customer has more than one transaction with the same transaction-time. We do not consider quantities of items bought in a transaction: each item is a binary variable representing whether an item was bought or not. An itemset is a non-empty set of items. A sequence is an ordered list of itemsets. A customer support is a sequence $s$ if $s$ is contained in the customer-sequence for this customer. The support for a sequence is defined as the fraction of total customers who support this sequence. Given a database D of customer transactions, the problem of mining sequential patterns is to find the maximal sequences among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a sequential pattern.

**Mining Sequential Patterns On Private Data problem:** Party 1 has a private data set $D_1$, party 2 has a private data set $D_2$, $\cdots$, and party n has a private data set $D_n$, data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ is the union of $D_1$, $D_2$, $\cdots$, and $D_n$ (by vertically putting $D_1$, $D_2$, $\cdots$, and $D_n$ together.)[*] Let $N$ be a set of transactions with $N_k$ representing the $k$th transaction. These n parties want to conduct the sequential pattern mining on $[D_1 \cup D_2 \cup D_3 \cdots \cup D_n]$ and to find the sequential patterns with support greater than the given threshold, but they do not want to share their private data sets with each other. We say that a sequential pattern of $x_i \leq y_j$, where $x_i$ occurs before or at the same time as $y_j$, has support $s$ in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if $s$% of the transactions in $[D_1 \cup D_2 \cdots \cup D_n]$ contain both $x_i$ and $y_j$ with $x_i$ happening before or at the same time as $y_j$ (namely, $s\% = Pr(x_i \leq y_j)$).

### II.2 Sequential Pattern Mining Procedure

The procedure of mining sequential patterns contains the following steps:

**Step I:** Sorting

The database $[D_1 \cup D_2 \cdots \cup D_n]$ is sorted, with customer ID as the major key and transaction time as the minor key. This step implicitly converts the original transaction database into a database of customer sequences. As a result, transactions of a customer may appear in more than one row which contains information of a customer ID, a particular transaction time and items bought at this transaction time. For example, suppose that datasets after being sorted by their customer-ID numbers are shown in Fig. 1. Then after being sorted by the transaction time, data tables of Fig. 1 will become those of Fig. 2.

**Step II:** Mapping

Each item of a row is considered as an attribute. We map each item of a row (i.e., an attribute) to an integer in an increasing order and repeat for all rows. Re-occurrence of an item will be mapped to the same integer. As a result, each item becomes an attribute and all attributes are binary-valued. For instance, the sequence $< B, (A, C) >$, indicating that the transaction B occurs prior to the transaction (A,C) with A and C occurring together, will be mapped to integers in the order $B \rightarrow 1$, $A \rightarrow 2$, $C \rightarrow 3$, $(A, C) \rightarrow 4$. During the mapping, the corresponding transaction time will be kept. For instance, based on the sorted dataset of Fig. 2, we may construct the mapping table as shown in Fig. 3. After the mapping, the mapped datasets are shown in Fig. 4.

**Step III:** Mining

Our mining procedure will be based on mapped dataset. The general sequential pattern mining procedure contains multiple passes over the data. In each pass, we start with a seed set of large sequences, where a large sequence refers to a sequence whose itemsets all satisfy the minimum support. We utilize the seed set for generating new potentially large sequences, called candidate sequences. We find the support for these candidate sequences during the pass over the data. At the end of each pass, we determine which of the candidate sequences are actually large. These large candidates become the seed for the next pass.

The following is the procedure for mining sequential patterns on $[D_1 \cup D_2 \cdots \cup D_n]$.

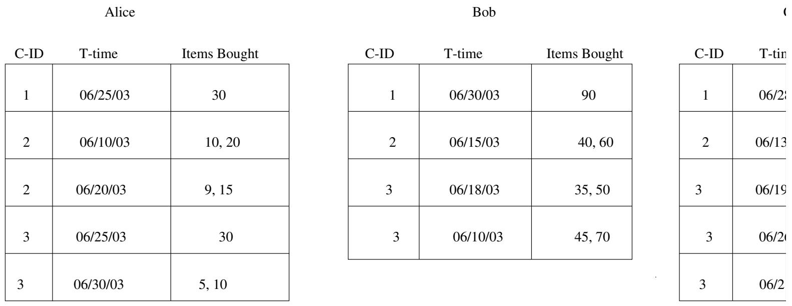
```
$L_1$ = large 1-sequence
for (k = 2; $L_{k-1} \neq \phi$; k++) **do**{
    $C_k$ = **apriori-generate**($L_{k-1}$)
    for all candidates $c \in C_k$ **do** {
      **Compute** $c.count$
   (Section  will show how to compute this count on
   private data)
      $L_k = L_k \cup$ c | $c.count \geq minsup$
   end
end
Return $U_k L_k$
```

where $L_k$ stands for a sequence with $k$ itemsets and $C_k$ stands for the collection of candidate k-sequences. The procedure **apriori-generate** is described as follows:

First: join $L_{k-1}$ with $L_{k-1}$:

```
insert into $C_k$
select $p.litemset_1$, $\cdots$, $p.litemset_{k-1}$,
   $q.litemset_{k-1}$, where $p.litemset_1 = q.litemset_1$,
   $\cdots$,
   $p.litemset_{k-2} = q.litemset_{k-2}$
from $L_{k-1}$ p, $L_{k-1}$ q.
```

[*] Vertically partitioned datasets are also called heterogeneous partitioned datasets where different datasets contain different types of items with customer IDs are identical for each transaction.



Alice

| C-ID | T-time | Items Bought |
|---|---|---|
| 1 | 06/25/03 | 30 |
| 2 | 06/10/03 | 10, 20 |
| 2 | 06/20/03 | 9, 15 |
| 3 | 06/25/03 | 30 |
| 3 | 06/30/03 | 5, 10 |

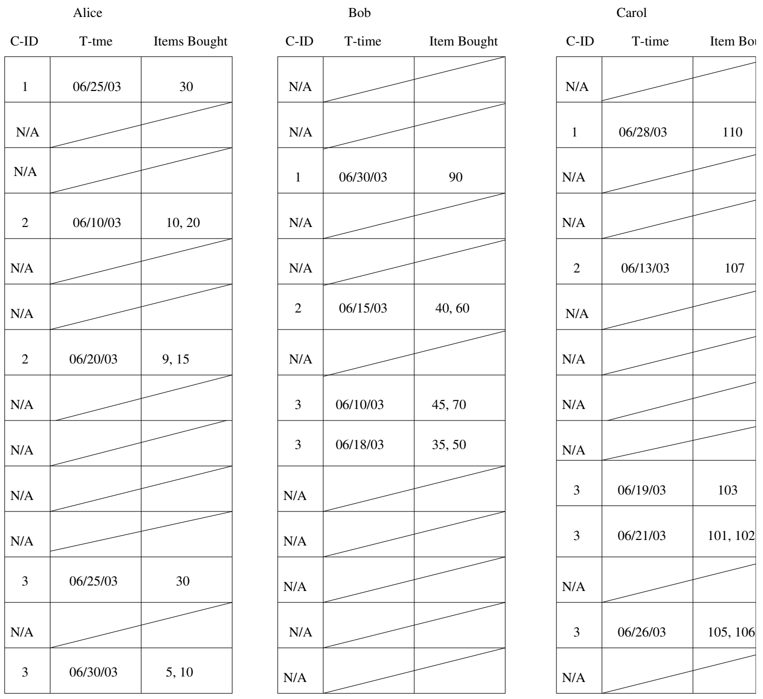
Bob

| C-ID | T-time | Items Bought |
|---|---|---|
| 1 | 06/30/03 | 90 |
| 2 | 06/15/03 | 40, 60 |
| 3 | 06/18/03 | 35, 50 |
| 3 | 06/10/03 | 45, 70 |

(

| C-ID | T-tir |
|---|---|
| 1 | 06/2: |
| 2 | 06/13 |
| 3 | 06/19 |
| 3 | 06/2( |
| 3 | 06/2 |

Figure 1: Raw Data Sorted By Customer ID



Alice

| C-ID | T-tme | Items Bought |
|---|---|---|
| 1 | 06/25/03 | 30 |
| N/A | | |
| N/A | | |
| 2 | 06/10/03 | 10, 20 |
| N/A | | |
| N/A | | |
| 2 | 06/20/03 | 9, 15 |
| N/A | | |
| N/A | | |
| N/A | | |
| N/A | | |
| 3 | 06/25/03 | 30 |
| N/A | | |
| 3 | 06/30/03 | 5, 10 |

Bob

| C-ID | T-time | Item Bought |
|---|---|---|
| N/A | | |
| N/A | | |
| 1 | 06/30/03 | 90 |
| N/A | | |
| N/A | | |
| 2 | 06/15/03 | 40, 60 |
| N/A | | |
| 3 | 06/10/03 | 45, 70 |
| 3 | 06/18/03 | 35, 50 |
| N/A | | |
| N/A | | |
| N/A | | |
| N/A | | |
| N/A | | |

Carol

| C-ID | T-time | Item Bo |
|---|---|---|
| N/A | | |
| 1 | 06/28/03 | 110 |
| N/A | | |
| N/A | | |
| 2 | 06/13/03 | 107 |
| N/A | | |
| N/A | | |
| N/A | | |
| N/A | | |
| 3 | 06/19/03 | 103 |
| 3 | 06/21/03 | 101, 102 |
| N/A | | |
| 3 | 06/26/03 | 105, 106 |
| N/A | | |

N/A: The information is not available.

Figure 2: Raw Data Sorted By Customer ID and Transaction Time

Second: delete all sequences $c \in C_k$ such that some (k-1)-subsequence of c is not in $L_{k-1}$.

**Step IV:** Maximization

Having found the set of all large sequences S, we provide the following procedure to find the maximal sequences.

```
1. for (k = m; $k \leq 1$; k- -) do
2.     for each k-sequence $s_k$ do
3.         Delete all subsequences of $s_k$ from S
```

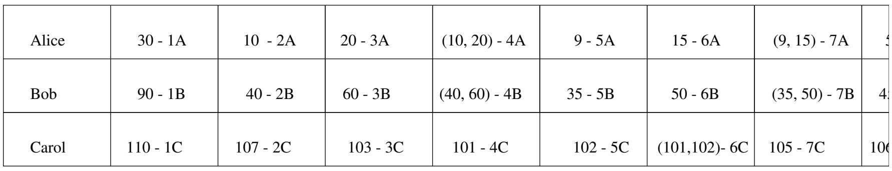
**Step V:** Converting

The items in the final large sequences are converted back to the original item representations used before the mapping step. For example, if 1A belongs to some large sequential pattern, then 1A will be converted to item 30, according to the mapping table, in the final large sequential patterns.

### II.3 How to compute *c.count*

To compute *c.count*, in other words, to compute the support for some candidate pattern (e.g., $P(x_i \cap y_i \cap z_i | x_i \geq y_i \geq z_i)$), we need to conduct two steps: one is to deal with the condition part where $z_i$ occurs before $y_i$ and both of them occur before $x_i$; the other is to compute the actual counts for this sequential pattern.

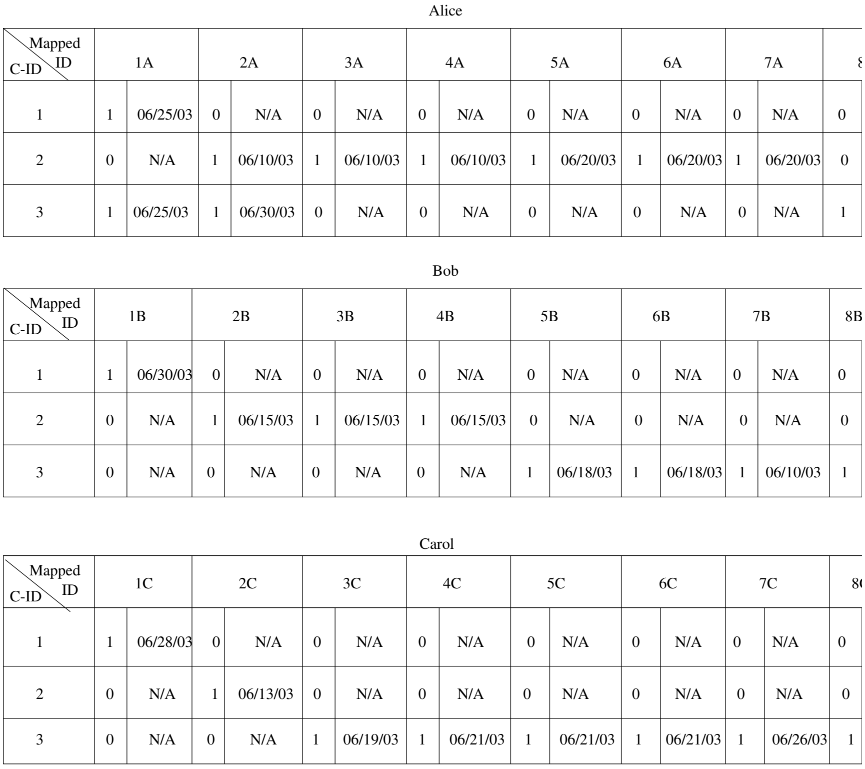
If all the candidates belong to one party, then *c.count*, which refers to the frequency counts for candidates, can be computed by this party since this party has all the information needed to compute it. However, if the can-



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Alice | 30 - 1A | 10 - 2A | 20 - 3A | (10, 20) - 4A | 9 - 5A | 15 - 6A | (9, 15) - 7A | 5 |
| Bob | 90 - 1B | 40 - 2B | 60 - 3B | (40, 60) - 4B | 35 - 5B | 50 - 6B | (35, 50) - 7B | 4 |
| Carol | 110 - 1C | 107 - 2C | 103 - 3C | 101 - 4C | 102 - 5C | (101,102)- 6C | 105 - 7C | 10 |

Note that, in Alice's dataset, item 30 and 10 are reoccurred, so we map them to the same mapped-ID.

Figure 3: Mapping Table



Alice

| Mapped ID / C-ID | 1A | | 2A | | 3A | | 4A | | 5A | | 6A | | 7A | | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 06/25/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 |
| 2 | 0 | N/A | 1 | 06/10/03 | 1 | 06/10/03 | 1 | 06/10/03 | 1 | 06/20/03 | 1 | 06/20/03 | 1 | 06/20/03 | 0 |
| 3 | 1 | 06/25/03 | 1 | 06/30/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 1 |

Bob

| Mapped ID / C-ID | 1B | | 2B | | 3B | | 4B | | 5B | | 6B | | 7B | | 8B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 06/30/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 |
| 2 | 0 | N/A | 1 | 06/15/03 | 1 | 06/15/03 | 1 | 06/15/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 |
| 3 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 1 | 06/18/03 | 1 | 06/18/03 | 1 | 06/10/03 | 1 |

Carol

| Mapped ID / C-ID | 1C | | 2C | | 3C | | 4C | | 5C | | 6C | | 7C | | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 06/28/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 |
| 2 | 0 | N/A | 1 | 06/13/03 | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 | N/A | 0 |
| 3 | 0 | N/A | 0 | N/A | 1 | 06/19/03 | 1 | 06/21/03 | 1 | 06/21/03 | 1 | 06/21/03 | 1 | 06/26/03 | 1 |

N/A : The information is not available.

Figure 4: Data After Being Mapped

didates belong to different parties, it is a non-trivial task to conduct the joint frequency counts while protecting the security of data. We provide the following steps to conduct this cross-parties' computation.

### II.3.1 Vector Construction

The parties construct vectors for their own attributes (mapped-ID). In each vector constructed from the mapped dataset, there are two components: one consists of the binary values (called the value vector); the other consists of the transaction time (called the transaction time vector). Suppose we want to compute the *c.count* for $2A \geq 2B \geq 6C$ in Fig. 4. We construct three vectors: 2A, 2B and 6C depicted in Fig. 5.

### II.3.2 Transaction time comparison

To compare the transaction time, each time vector should have a value. We let all the parties randomly generate a set of transaction time for entries in the vector where their values are 0's. They then transform their values in time vector into real numbers so that if transaction $tr_1$ happens earlier than the transaction $tr_2$, then the real number to denote $tr_1$ should smaller than the number that denotes $tr_2$. For instance, "06/30/2003" and "06/18/2003" can be transform to 363 and 361.8 respectively. The purpose of transformation is that we will securely compare them based their real number denotation. The comparison can be conducted pairwise. Next, we will present a secure protocol that allows n parties to compare their transaction time.

The goal of our privacy-preserving classification sys-

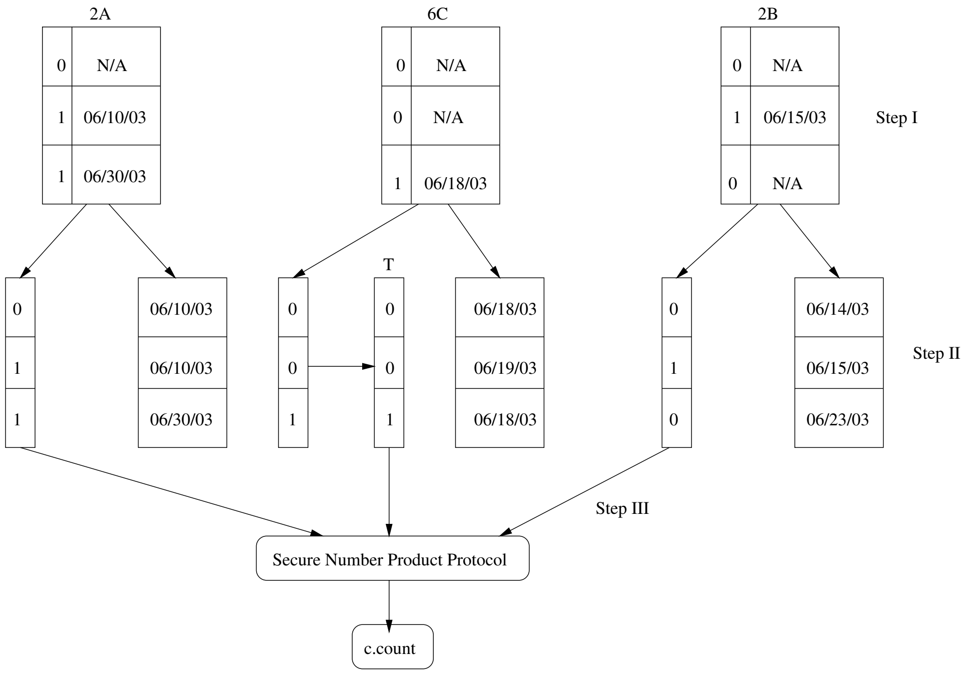


Figure 5: An Protocol To Compute c.count

tem is to disclose no private data in every step. We firstly select a key generator who produces the encryption and decryption key pairs. The computation of the whole system is under encryption. For the purpose of illustration, let's assume that $P_n$ is the key generator who generates a homomorphic encryption key pair (e, d). Next, we will show how to conduct each step.

### II.3.3 The Comparison of Transaction Time

Without loss of generality, assuming there are k transaction time: $e(g_1)$, $e(g_2)$, $\cdots$, and $e(g_k)$, with each corresponding to a transaction of a particular party.

**Protocol 1** .

1. *$P_{n-1}$ computes $e(g_i) \times e(g_j)^{-1} = e(g_i - g_j)$ for all $i, j \in [1, k], i > j$, and sends the sequence denoted by $\varphi$ to $P_n$ in a random order.*

2. *$P_n$ decrypts each element in the sequence $\varphi$. He assigns the element $+1$ if the result of decryption is not less than $0$, and $-1$, otherwise. Finally, he obtains a $+1/-1$ sequence denoted by $\varphi'$.*

3. *$P_n$ sends $+1/-1$ sequence $\varphi'$.*

4. *$P_{n-1}$ compares the transaction time of each entry of vectors such as $2A$, $2B$, and $6C$ in our example. She makes a temporary vector $T$. If the transaction time does not satisfy the requirement of $2A \geq 2B \geq 6C$, she sets the corresponding entries of $T$ to 0's; otherwise, she copies the original values in 6C to T (Fig. 5).*

**Theorem 1** *(Correctness). Protocol 1 correctly sort the transaction time.*

*Proof* $P_{n-1}$ is able to remove permutation effects from $\varphi'$ (the resultant sequence is denoted by $\varphi''$) since she has the permutation function that she used to permute $\varphi$, so that the elements in $\varphi$ and $\varphi''$ have the same order. It means that if the $q$th position in sequence $\varphi$ denotes $e(g_i - g_j)$, then the $q$th position in sequence $\varphi''$ denotes the evaluation results of $g_i - g_j$. We encode it as $+1$ if $g_i \geq g_j$, and as -1 otherwise. $P_{n-1}$ has two sequences: one is the $\varphi$, the sequence of $e(g_i - g_j)$, for $i, j \in [1, k](i > j)$, and the other is $\varphi''$, the sequence of $+1/-1$. The two sequences have the same number of elements. $P_{n-1}$ knows whether or not $g_i$ is larger than $g_j$ by checking the corresponding value in the $\varphi''$ sequence. For example, if the first element $\varphi''$ is $-1$, $P_{n-1}$ concludes $g_i < g_j$. $P_{n-1}$ examines the two sequences and constructs the index table (Table 1) to compute the largest element.

In Table 1, $+1$ in entry $ij$ indicates that the value of the row (e.g., $g_i$ of the $i$th row) is not less than the value of a column (e.g., $g_j$ of the $j$th column); -1, otherwise. $P_{n-1}$ sums the index values of each row and uses this number as the weight of the information gain in that row. She then sorts the sequence according the weight.

To make it clearer, let's illustrate it by an example. Assume that: (1) there are 4 elements with $g_1 < g_4 <$

| | $g_1$ | $g_2$ | $g_3$ | $\cdots$ | $g_k$ |
|---|---|---|---|---|---|
| $g_1$ | +1 | +1 | -1 | $\cdots$ | -1 |
| $g_2$ | -1 | +1 | -1 | $\cdots$ | -1 |
| $g_3$ | +1 | +1 | +1 | $\cdots$ | +1 |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| $g_k$ | +1 | +1 | -1 | $\cdots$ | +1 |

Table 1:

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ | Weight |
|---|---|---|---|---|---|
| $S_1$ | +1 | -1 | -1 | -1 | -2 |
| $S_2$ | +1 | +1 | -1 | +1 | +2 |
| $S_3$ | +1 | +1 | +1 | +1 | +4 |
| $S_4$ | +1 | -1 | -1 | +1 | 0 |

Table 2:

$g_2 < g_3$; (2) the sequence $\varphi$ is $[e(g_1-g_2), e(g_1-g_3), e(g_1-g_4), e(g_2-g_3), e(g_2-g_4), e(g_3-g_4)]$. The sequence $\varphi''$ will be $[-1,-1,-1,-1,+1,+1]$. According to $\varphi$ and $\varphi''$, $P_{n-1}$ builds the Table 2. From the table, $P_{n-1}$ knows $g_3 > g_2 > g_4 > g_1$ since $g_3$ has the largest weight, $g_2$ has the second largest weight, $g_4$ is third largest weight, $g_1$ has the smallest weight.

∎

**Theorem 2** *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

*Proof* We need prove it from two aspects: (1) $P_{n-1}$ doesn't get transaction time (e.g.,$g_i$) for each vector. What $P_{n-1}$ gets are $e(g_i-g_j)$ for all $i, j \in [1, k], i > j$ and $+1/-1$ sequence. By $e(g_i - g_j)$, $P_{n-1}$ cannot know each transaction time since it is encrypted. By $+1/-1$ sequence, $P_{n-1}$ can only know whether or not $g_i$ is greater than $P_j$. (2) $P_n$ doesn't obtain transaction time for each vector either. Since the sequence of $e(g_i - g_j)$ is randomized before being send to $P_n$ who can only know the sequence of $g_i - g_j$, he can't get each individual transaction time. Thus private data are not revealed. ∎

**Theorem 3** *(Efficiency). The computation of protocol 1 is efficient from both computation and communication point of view.*

*Proof* The total communication cost is upper bounded by $\alpha m^2$. The total computation cost is upper bounded by $m^2 + m + 1$. Therefore, the protocols are very fast. ∎

After the above step, they need to compute *c.count* based their value vector. For example, to obtain *c.count* for $2A \geq 2B \geq 6C$ in Fig. 5, they need to compute $\sum_{i=1}^{N} 2A[i] \cdot 2B[i] \cdot T[i] = \sum_{i=1}^{N} 2A[i] \cdot 2B[i] \cdot T[i] = \sum_{i=1}^{3} 2A[i] \cdot 2B[i] \cdot T[i] = 0$, where N is the total number of values in each vector. In general, let's assume the value vectors for $P_1, \cdots, P_n$ are $x_1, \cdots, x_n$ respectively. Note that $P_1$'s vector is $T$. For the purpose of illustration, we denote $T$ by $x_{n-1}$. Next, we will show how n parties compute this count. without revealing their private data to each other.

### II.3.4 The Computation of *c.count*

**Protocol 2** *Privacy-Preserving Number Product Protocol*

1. *$P_n$ sends $e(x_{n1})$ to $P_1$.*

2. *$P_1$ computes $e(x_{n1})^{x_{11}} = e(x_{n1}x_{11})$, then sends it to $P_2$.*

3. *$P_2$ computes $e(x_{n1}x_{11})^{x_{21}} = e(x_{n1}x_{11}x_{21})$.*

4. *Continue until $P_{n-1}$ obtains $e(x_{11}x_{21}\cdots x_{n1})$.*

5. *Repeat all the above steps for $x_{1i}$, $x_{2i}$, $\cdots$, and $x_{ni}$ until $P_{n-1}$ gets $e(x_{1i}x_{2i}\cdots x_{ni})$ for all $i \in [1, N]$.*

6. *$P_{n-1}$ computes $e(x_{11}x_{21}\cdots x_{n1}) \times e(x_{12}x_{22}\cdots x_{n2}) \times \cdots \times e(x_{1N}x_{2N}\cdots x_{nN}) = e(x_{11}x_{21}\cdots x_{n1} + x_{12}x_{22}\cdots x_{n2} + \cdots + x_{1N}x_{2N}\cdots x_{nN}) = c.count$.*

**Theorem 4** *(Correctness). Protocol 2 correctly compute* c.count.

*Proof* In step 2, $P_1$ obtains $e(x_{n1})$. He then computes $e(x_{n1}x_{11})$. In step 3, $P_2$ computes $e(x_{n1}x_{11}x_{21})$. Finally, in step 5, $P_{n-1}$ gets $e(x_{1i}x_{2i}\cdots x_{ni})$. He then computes $e(x_{11}x_{21}\cdots x_{n1}) \times e(x_{12}x_{22}\cdots x_{n2}) \times \cdots \times e(x_{1N}x_{2N}\cdots x_{nN}) = e(x_{11}x_{21}\cdots x_{n1} + x_{12}x_{22}\cdots x_{n2} + \cdots + x_{1N}x_{2N}\cdots x_{nN})$ which is equal to *c.count.*

∎

**Theorem 5** *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

*Proof* In protocol 2, all the data transmission are hidden under encryption. The parties who are not the key generator can't see other parties' private data. On the other hand, the key generator doesn't obtain the encryption of other parties's private data. Therefore, protocol 2 discloses no private data.

∎

**Theorem 6** *(Efficiency). The computation of c.count is efficient from both computation and communication point of view.*

*Proof* To prove the efficiency, we need conduct complexity analysis of the protocol. The bit-wise communication cost of protocol 2 is $\alpha(n-1)N$. The computation cost of protocol 2 is $nN$, of protocol 2 is $5t+3$. The total computation cost is upper bounded by $nN-1$. Therefore, the protocols are sufficient fast.

∎

## III. Overall Discussion

Our privacy-preserving classification system contains several components. In Section , we show how to correctly compare the transaction time. In Section , we present protocols to compute *c.count*. We discussed the correctness of the computation in each section.

As for the privacy protection, all the communications between the parties are encrypted, therefore, the parties who has no decryption key cannot gain anything out of the communication. On the other hand, there are some communication between the key generator and other parties. Although the communications are still encrypted, the key generator may gain some useful information. However, we guarantee that the key generator cannot gain the private data by adding random numbers in the original encrypted data so that even if the key generator get the intermediate results, there is little possibility that he can know the intermediate results. Therefore, the private data are securely protected with overwhelming probability.

In summary, we provide a novel solution for sequential pattern mining over vertically partitioned private data. Instead of using data transformation, we define a protocol using homomorphic encryption to exchange the data while keeping it private. Our mining system is quite efficient that can be envisioned by the communication and computation complexity. The total communication complexity is upper bounded by $\alpha(nN+m^2-N)$. The computation complexity is upper bounded by $m^2+m+5t+4$.

## References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[2] Jay Ayres, Johannes Gehrke, Tomi Yiu, and Jason Flannick. Sequential pattern mining using a bitmap representation.

[3] G. Chirn. *Pattern discovery in sequence databases: Algorithms and applications to DNA/protein classification.* PhD thesis, Department of Computer and Information Science, New Jersey Institute of Technology, 1996.

[4] O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[6] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, Santa Barbara, CA USA, August 21-24 1997.

[7] H. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP: Approximate mining of consensus sequential patterns. Technical Report TR02-031, UNC-CH, 2002.

[8] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[9] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.