# Incremental Privacy-Preserving Association Rule Mining Using Negative Border

Duc H. Tran[1,2(✉)], Wee Keong Ng[1], Y.D. Wong[2,3], and Vinh V. Thai[4]

[1] School of Computer Engineering, Nanyang Technological University, Singapore, Singapore
ductran@ntu.edu.sg
[2] Maritime Institute, Nanyang Technological University, Singapore, Singapore
[3] School of Civil and Environmental Engineering, Nanyang Technological University, Singapore, Singapore
[4] School of Business IT and Logistics, RMIT University, Melbourne, Australia

**Abstract.** Privacy preserving association rule mining can extract important rules from distributed data with limited privacy breaches. Protecting privacy in incremental maintenance for distributed association rule mining is necessary since data are frequently updated. In privacy preserving data mining, scanning all the distributed data is very costly. This paper proposes a new incremental protocol for privacy preserving association rule mining using negative border concept. The protocol scans old databases at most once, and therefore reducing the I/O time. We also conduct experiments to show the efficiency of our protocol over existing ones.

**Keywords:** Privacy preserving · Secure protocol · Association rule mining · Negative border · Incremental

## 1 Introduction

Since its inception, preserving privacy has become one of the major tasks of data mining area and has attracted tremendous interest among researchers. A variety of algorithms and techniques has been introduced to perform mining in secure manners. Traditionally, all these algorithms are designed with assumption that data is persistent. In case that there are updates, deletion of records, data mining process needs to run again. This certainly is impractical since mining on distributed data are so costly that it cannot be performed frequently.

Let see a scenario: $n$ parties hold their horizontally partitioned data. They together perform a association rule mining as proposed by Kantarcioglu and Clifton [11]. After all parties have completed the algorithm and got results, another party with his own data wishes to join the task. The first $n$ parties certainly want him to do the mining task for the accuracy of the mining results (the more data involved, the more accurate the result is). However, the first $n$ parties have finished the mining task (often time-consuming) and are not really willing to start over again.

M. Chau et al. (Eds.): PAISI 2016, LNCS 9650, pp. 87–100, 2016.
DOI: 10.1007/978-3-319-31863-9_7

Another scenario is that, after a day or a week, all the parties collect more transaction data. They would like to update the mining result with less effort. The problem of updating association rules is first studied by Cheung with insertion operation in [4] and then updated with deletion and modification in [5]. However, the issue of maintaining association rules in privacy-preserving context is a challenge and to our best knowledge, there is no related work trying to solve this problem.

In this paper, we tackle this challenge using the original incremental techniques proposed by Cheung [4] and secure multi-party computation (SMC) in [8]. We propose a novel incremental association rule mining protocol that can be used when data of parties are updated or some new parties join the mining tasks. Our new protocol will update final results considering the old mining results and new data. In most cases, the protocol does not require to read the old data.

The rest of this paper is organized as follows. Section 2 demonstrates the problem. Section 3 reviews related work to our solution. Section 4 presents the background that is important to build our protocol. Section 5 introduces our novel protocol to incrementally perform association rule mining. We conduct experiments in Sect. 6. Finally, Sect. 7 gives the summary of this paper.

## 2 Problem Definition

In this section, we define the association rule mining problem. Here are some notations.

- $I = \{i_1..i_m\}$: a set of items
- $DB$: a set of transactions
- $T$: a transaction in $DB$. A transaction is a set of itemsets in $I$. We have $T \subseteq I$.
- $L^{DB}$: all frequent itemsets in $DB$.
- $L_k^{DB}$: all $k$-large itemsets in $DB$.

Given an itemset $X \subseteq I$. An association rule can be written as $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \bigcap Y = \emptyset$. The rule $X \Rightarrow Y$ is said to have support $s$ if $s\,\%$ of transactions in $DB$ contain $X \bigcup Y$. The rule holds is said to have confidence $c\,\%$ if $c\,\%$ transactions in $DB$ that contain $X$ also contains $Y$. The mining association rule problem is to discover all rules that have satisfied support and confidence thresholds.

**Distributed Data.** A database $DB$ is horizontally partitioned in $n$ sites $(S_1, ..., S_n)$. Their database are $DB_1, ..., DB_n$ respectively.

Assume that itemset $X$ has count of $X.sup_i$ at each site $S_i$ (i.e., $X.sup_i$ of the transactions contains $X$). Then we can compute global count of $X$ as $X.sup = \sum_{i=1}^{n} X.sup_i$.

The goal of privacy preserving association rule mining is to discover aassociation rules satisfying thresholds, i.e., the sets $L_k$ for all $k > 1$ without any privacy breach. A protocol is privacy preserving if no site should be able to learn extra information of any transaction at any other site other than the final results of mining tasks.

**Incremental Mining.** Assume that $n$ parties have found the large itemsets for their data, presented in $\bigcup_{k=1}^{n} DB_k$. Now $r$ new sites $S_{n+1}, S_{n+2}, ..., S_{n+r}$ want to do mining tasks with current $n$ sites. The new sites have database $DB_i$, for sites $i = n+1, ..., n+r$ respectively. It is simple that we can do this by asking all $n+r$ to do mining tasks again. However, this method takes long time and old sites may not be willing to run again. The purpose of incremental mining is to discover the new updated results without running algorithms all over old and new datasets. To protect the privacy of the old sites, the $r$ new sites should not know the large itemsets $L$ of the $n$ old sites. On the other hand, the $n$ old sites should also not know any other information of the $r$ new sites except one stated as follows. However, due to the nature of this problem, privacy breach is unavoidable no matter how secure the protocol is. For instance, if an itemset $X$ is large (small) in the $n$ old sites but after updating it becomes small (large) in $n+r$ sites then all the old sites know that $X$ is small (large) in the $r$ new sites. Even if we run the protocol in [11] again, this conclusion is still correct.

**Definition 1.** *Let $X.sup_i$ be the support count of $X$ in $S_i$. An itemset $X$ is said to be globally large if $\sum_{k=1}^{n+r} X.sup_k \geq \sum_{k=1}^{n+r} DB_k \times s\%$. $X$ is said to be group large in new sites if $\sum_{k=n+1}^{n+r} X.sup_k \geq \sum_{k=n+1}^{n+r} DB_k \times s\%$. $X$ is said to be group large in old sites if $\sum_{k=1}^{n} X.sup_k \geq \sum_{k=1}^{n} DB_k \times s\%$.*

## 3 Related Works

### 3.1 Incremental Association Rule Mining

Association rule mining algorithms have been divided into two categories: Apriori-based and FP-tree-based. The problem of maintenance of association rules in large databases was first presented in 1996 by Cheung et al. with the FUP algorithm [4]. They then upgraded to the FUP2 algorithm [5] including not only addition but also deletion and modification of data. FUP2 is more efficient than FUP. In 1997, Thomas *et al.* [16] introduced a new method to boost the progress of the incremental mining using the negative border. Ayan *et al.* [2] presented a new method called UWEP (Update With Early Pruning), which exploits a dynamic look-ahead strategy. The list of large itemsets could be updated by checking itemsets if they are frequent in new database. In 2001, SWF method [12] was first demonstrated by Lee *et al.*. The method splits databases into partitions, and using a filtering threshold in each partition to create a list of candidate itemsets. Veloso *et al.* proposed a new method called ZigZag algorithm that makes use of *tidlist* and generates maximal frequent itemsets in new database to prevent from creating too many unnecessary candidates [18].

### 3.2 Privacy Preserving Association Rule Mining

While there has been a lot of related work in privacy-preserving data mining, due to space constraints, we only focus on the tightly related efforts. The method presented by Kantarcioglu *et al.* [11] is the first cryptography-based solutions for private distributed association rules mining, it assumes three or more parties, and they jointly do the distributed Apriori algorithm with the data encrypted. In the recent research papers [7,9,15,17,19], some privacy-preserving association rules schemes are proposed. These papers are similar and developed a secure multi-party protocol based on homomorphic encryption.

While the two above related issues have been well studied, there is a very limited work has been dedicated to simultaneously solve both incremental maintenance and privacy issue of association rule mining. Wong et al. proposed an incremental method to protect privacy for distributed association rule mining [19]. Their algorithm is based on the original FUP/FUP2 algorithm that may scan the old databases many times. In this paper, we present a new algorithm that requires the old database at most once. Our algorithm makes use of the negative border and cryptography techniques.

## 4 Preliminaries

### 4.1 Negative Border

In this subsection, we review the concept of *negative border* which was presented by Mannila and Toivonen [13].

**Definition 2.** *The negative border $Bd^-(L)$, of a collection of itemsets $L$ is defined as follows: Given a collection $L \subseteq P(R)$ of sets, closed with respect to the set inclusion relation, the negative border $Bd^-(L)$ of $L$ consists of the minimal itemsets $X \subseteq R$ not in $L$.*

*Example 1.* Let $R = \{A, B, ..., F\}$ and assume the collection $L$ of frequent itemsets is L={{A},{B},{C},{F}, {A,B},{A,C}, {A,F},{C,F},{A,C,F}}. The negative border of the collection $L$ contains the itemset $\{B, C\}$ since it is not in $L$ but all its subsets are. The whole negative border is

$$Bd^-(L) = \{\{B, C\}, \{B, F\}, \{D\}, \{E\}\}.$$

The intuition behind the concept is that, given a collection $L$ of itemsets that are frequent, the negative border contains the "closet" itemsets that could be frequent, too.

**Definition 3.** *The closed set $CS(L)$ of a collection of itemsets $L$ is defined as follows. $CS(L) \equiv L \cup Bd^-(L)$.*

### 4.2 Secure Building Blocks

In this subsection, we review some secure protocols that will be used later in our algorithms.

**Protocol 1.** Homomorphic Secure Sum Protocol

```
1: Party 1 generates public key pair with a homomorphic encryption. It then shares
   the public key to all parties.
2: Party 1 calculates: $s_1 = E(v_1)$ where E(.) is the encryption operation. Then site 1
   sends $s_1$ to site 2.
3: Each party $i$, where $2 \leq i \leq m$, gets $s_{i-1}$ from party $i-1$ and calculates: $s_i =$
   $s_{i-1} \cdot E(v_i)$
4: Party $m$ send $s_m$ to party 1
5: Party 1 decrypts $s_m$ using his private key and shares the result to all parties.
```

## 4.3 Secure Sum

Secure Sum allows parties to securely compute the sum of data values from many parties. Assume that party $i$ holds a value $v_i$. They want to calculate $v = \sum_{l=1}^{m} v_l$, where $0 \leq v \leq n$. In SMC method, there are two basic protocols that are allow to calculate a sum of many values from different sites in a secure manner.

Homomorphic encryption can be used to compute secure sum as shown in Protocol 1. This protocol is secure unless there is a collusion between the first party, who is private key keeper, and any other party.

Another method to securely compute sum of many values was proposed by Clifton *et al.* [6].

## 4.4 Secure Comparison

Let take the famous problem about two millionaires, named Alice and Bob. They want to know who is richer without revealing their money. The problem is originally presented by A. Yao [20]. He used secure comparison protocols to solve the problem. Yao then improved his method [21] to obtain a protocol that takes a linear complexity. Ioannidis then introduced a protocol that could perform secure comparison in logarithm time [10]. For the details of protocol, please refer to [10,20,21].

## 4.5 The Incremental Large Itemset Mining

In this section, we review a fast algorithm for incremental update of association rule mining proposed by Thomas *et al.* in [16]. The algorithm is done with two support algorithms: Apriori Generator and Negative Border Generator. Then the main algorithm Update-Large-Itemset is reviewed.

**Apriori Generator.** Protocol 2 demonstrates the appriori generation to produce $k$-large candidate itemsets when it has the set of $(k-1)$-large frequent itemsets. It takes an argument $L_{k-1}$, set of $(k-1)$-large itemsets and returns candidate $k$-large itemset. The correctness of protocol is proved in [1]. As we can see from the protocol, it can be done by any party without compromising

**Protocol 2.** Apriori-Gen

```
Require: $L_{k-1}$
Ensure: $C_k$
 1: for each $p$ and $q \in L_{k-1}$ do
 2:     if $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}$ and $p.item_{k-1} < q.item_{k-1}$
        then
 3:         Insert itemset $p.item_1, ..., p.item_{k-1}, q.item_{k-1}$ into $C_k$
 4:     end if
 5: end for
 6: for each $c \in C_k$ do
 7:     if some $(k-1)$-subset of $c \notin L_{k-1}$ then
 8:         Delete $c$ from $C_k$
 9:     end if
10: end for
```

**Protocol 3.** Neg-Border-Gen

```
Require: $L$
Ensure: $L \cup Bd^{-}L$
 1: Split $L$ into $L_1, L_2, ..., L_n$ where $n$ is the size of the largest itemset in $L$
 2: for each $k = 1, 2, ..., n$ do do
 3:     Compute $C_{k+1}$ using $apriori - gen(L_K)$
 4: end for
 5: $L \cup Bd^{-}L = \bigcup_{i=2,3,...,n+1} C_k \cup I_1$ where $I_1$ is the set of 1-itemsets
```

data privacy. The reason is that by default, all parties hold $L = \bigcup_{k=1..n} L_k$ at the end of the protocol. Hence, there is no need to apply privacy preserving techniques for this protocol.

**Negative Border Generator.** Protocol 3 shows the negative border generation. The input is $L$, a collection of all large itemset. The output is the negative border of $L$ along with $L$ itself. The correctness of this protocol can be found in [16]. Since the protocol can be performed by any party, there is no need to consider privacy issues. Hence, it can directly be applied into privacy preserving data mining versions. Figure 1 show the relationship between $L^{DB}$, $L^{db}$ and $Bd^{-}(L^{DB})$.

**Update Large Itemsets.** Thomas *et al.* proposed an efficient algorithm to incremental update large itemset in [16]. The algorithm is presented in Protocol 4. The correctness proof of this algorithm can be found in [16]. The algorithm includes the following parts:

- **Part 1:** Compute large itemsets for new data at all sites (Step 1). All new parties or parties with data changed can used FUP algorithm to compute large itemset on new data.

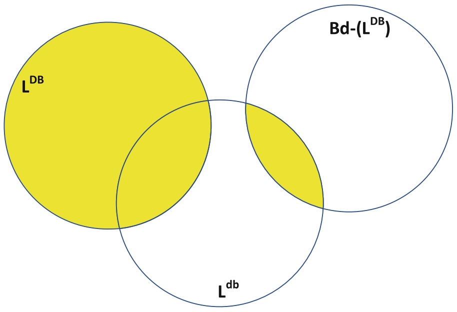

**Fig. 1.** The relationship of $L^{DB}$, $L^{db}$ and $Bd^{-}(L^{DB})$.

- **Part 2:** Update large itemsets for all sites (Step 2–15). These steps are used to update new large itemsets or delete ones that are no longer large in new data.
- **Part 3:** Compute negative border of large itemsets and if there are some new itemsets, scan old datasets to update final large itemsets (Step 16–28).

**Security Analysis.** In this part, we analyze the privacy-preserving issues of the Prtocol 4. Steps 2–4 clearly breach privacy information: new parties know all itemsets in $L^{DB} \cup Bd^{-}(L^{DB})$ and old parties know the support count in new parties for each itemset $X \in L^{DB} \cup Bd^{-}(L^{DB})$.

## 5 The Proposed Incremental Privacy-Preserving Association Rule Mining

In this section, we use the tools described above to construct an incremental protocol that preserves the privacy of every site. Our algorithm follows the general method of the Update-Large-Itemset algorithm as reviewed in Sect. 4.5.

The protocol is presented in Protocol 5. It includes the following parts:

- Compute large itemsets in new sites only
- Compute Negative Border for old and new sites
- Update large itemsets for all sites.

In the next sections, we sequentially discuss in detail for each step.

### 5.1 Computing Large Itemsets in New Sites

In Step 1, all new sites simply apply the algorithm in [11]. The first site keeps a list of random numbers $x_p$, where $p = 1, ..., |L^{db}|$. The last site holds: $S = \sum_{i=1}^{r}(X.sup_i - |DB_i| \times s\,\%) + x_p$. To check if itemset $X$ is large, the first and the last site only need to compare $S$ with $x_p$ using Yao's protocol [21]. After this step, every new site holds the large itemset $L^{db}$.

**Protocol 4.** Update-Large-Itemset

```
Require: $L^{DB}, Bd^{-}(L^{DB}), db.$
Ensure: $L^{DB+}$
 1: Compute $L^{db}$
 2: for each itemset $X \in L^{DB} \cup Bd^{-}(L^{DB})$ do
 3:     $t_{db}(X)$ = number of transactions in $db$ containing $X$
 4: end for
 5: $L^{DB+} = \emptyset$
 6: for each itemset $X \in L^{DB}$ do
 7:     if $t_{DB}(X) + t_{db}(X) \geq (DB + db) \times s\,\%$ then
 8:         $L^{DB+} = L^{DB+} \cup X$
 9:     end if
10: end for
11: for each itemset $X \in L^{db}$ do
12:     if $X \notin L^{DB}$ and $X \in Bd^{-}(L^{DB})$ and $t_{DB}(X)+t_{db}(X) \geq (DB+db)\times s\,\%$ then
13:         $L^{DB+} = L^{DB+} \cup X$
14:     end if
15: end for
16: if $L^{DB} \neq L^{DB+}$ then
17:     $Bd^{-}(L^{DB+})$=negative-border-gen($L^{DB+}$))
18: else
19:     $Bd^{-}(L^{DB+}) = Bd^{-}(L^{DB})$
20: end if
21: if $L^{DB} \cup Bd^{-}(L^{DB}) \neq L^{DB+} \cup Bd^{-}(L^{DB+})$ then
22:     $S = L^{DB+}$
23:     repeat
24:         Compute $S = S \cup Bd^{-}(S)$
25:     until $S$ does not grow
26: end if
27: $L^{DB+} = \{X \in S | sup(X) \geq s\}$
28: $Bd^{-}(L^{DB+})$ = negative-border-gen($L^{DB+}$))
```

## 5.2 Compute Negative Border

After Step 1, all new sites hold $L^{db}$ while all old sites hold $Bd^{-}(L^{db}) \cup L^{db}$. Applying Secure Union Set proposed by Tassa in [15], all sites can securely compute union set $L = L^{db} \cup Bd^{-}(L^{db}) \cup L^{db}$.

## 5.3 Updating Large Itemsets for All Sites

Steps 3–10, all sites compute large itemsets. Details are as follows.

- Step 4: New sites compute support count of itemsets using Secure Sum building blocks. There is no privacy leakage here as only new sites working together.
- Step 5: Old sites compute support count of itemsets using Secure Sum building blocks. Again, this step is privacy preserving as it relates to old sites only.
- Step 6: Using Secure Sum and Secure Comparison building blocks, one of the old sites and one of the new sites can easily check if an itemset is large.

**Protocol 5.** Incremental Privacy Preserving Large Itemset Mining

```
Require: $L^{DB}, Bd^-(L^{DB})$ from $n$ old sites. $DB_1, DB_2, ..DB_r$ from $r$ new sites.
Ensure: $L^{DB+}$: The frequent itemset of $(n+r)$ sites.
 1: All $r$ new sites compute $L^{db}$ using protocol in [11]
 2: Compute $L = L^{db} \cup Bd^-(L^{db}) \cup L^{db}$ using Secure Union Set
 3: for each itemset $s \in L$ do
 4:     All new sites compute $t_{db}(s)$, the support count of the itemset $s$ in new sites
        using Secure Sum
 5:     All old site compute $t_{DB}(s)$, the support count of the itemset $s$ in old sites using
        Secure Sum
 6:     One of the new sites and one of the old sites together use Secure Sum and Secure
        Comparison to check if $s$ is a large itemset.
 7:     if $s$ is large then
 8:         $L^{DB+} = L^{DB+} \cup s$
 9:     end if
10: end for
11: One of the sites compute Negative Border: $L^{DB+} \cup Bd^-(L^{DB+})$
12: if $L^{DB} \cup Bd^-(L^{DB}) \neq L^{DB+} \cup Bd^-(L^{DB+})$ then
13:     $L^{DB+} = \{X \in S | sup(X) \geq s\}$
14: end if
```

- Steps 7–10: The itemset is put into $L^{DB+}$, the set of all large itemset in both new and old data.
- Step 11: Compute Negative Border for $L^{DB+}$ using Protocol 3.
- Step 12: One of the old site checks if the Negative Border of $L^{DB}$ and $L^{DB+}$ are the same. If they are the same, it means that all candidate new large itemset are in the Negative Border of $L^{DB}$, where their support counts have been computed before. Hence, there is no need to scan old datasets again. On the other hand, if the Negative Border of $L^{DB}$ and $L^{DB+}$ are different, there is a need to scan old datasets on the old sites once more time to compute support counts for new candidate itemsets. The correctness of this statement is similar to that of Protocol 4, which is proved by Thomas *et al.* in [16].

**Theorem 1.** *Protocol 5 is correct, i.e., it correctly returns all itemsets that are frequent in combined data from old and new sites.*

*Proof.* Since Protocol 5 is derived based on Protocol 4, which is a non-secure version, the protocol is correct if Protocol 4 is correct. We have known that Protocol 4 is correct as it is proved in Theorem 1 in [16].

Thus we can safely conclude that Protocol 5 is correct, i.e., it generates all large itemsets. □

**Theorem 2.** *Protocol 5 is secure in semi-honest model.*

*Proof.* To prove that the protocol is secure, we need to prove that each step is secure.

- Step 1: This step is secure as it uses a secure protocol proposed by Kantarcioglu [11].
- Step 2: This step can be done by one of the new site and one of the old site using an efficient secure set union proposed by Tassa in [15]. Thus we can say that this step is secure.
- Step 4: Computing sum between new site is secure by using Secure Sum.
- Step 5: Similar to Step 4, this step is also secure with Secure Sum building blocks.
- Step 6: Since this step is to check if an itemset is frequent using Secure Sum and Secure Comparison building blocks, this step is secure too.
- Step 8: This step can securely done using Secure Set Union as in Step 2.
- Steps 11–12: Same as Step 8.
- Step 13: When it needs to scan old datasets again, we can apply the protocol by Kantarcioglu [11] to securely compute support count of itemsets and check if they are frequent. This step is thus secure too.

All steps in the protocol are secure. Then we can conclude that Protocol 5 is secure. □

## 6 Experiments

Section 6.1 describes parameters to generate synthesis datasets for experiments. Section 6.2 demonstrates the two experiments set to be conducted. And in Sect. 6.3, we discuss about the experiments' results.

### 6.1 Generating Synthetic Data

We have generated synthetic data using the same techniques as in [1,15]. Table 1 presents a list of parameters to generate synthesis data. Except parameter $m$ for number of sites, other parameters were used in previous work such as [1,3,11,14–16].

**Table 1.** Parameters to generate synthetic data.

| Parameter | Description | Value |
|---|---|---|
| m | Number of sites | 10 |
| N | Number of transactions per site | 500,000 |
| L | Number of items | 1,000 |
| T | Mean size of a transaction | 10 |
| I | Mean size of maximal potentially large itemsets | 4 |

## 6.2 Experimental Setup

We assume that $k$ sites are old sites, i.e., they have completed running association rule mining using the protocol presented in [15]. Now the other $10 - k$ sites are new and want to join the mining task to get final results. We will compare the running time of two approaches: (i) All 10 sites have to run again protocol in [15], called **TASSA14** method. (ii) All 10 sites apply our incremental method as in Protocol 5, called **INCRE** method.

We have conducted two experiment sets as follows.

- We fix $k = 5$, i.e., 5 old sites and 5 new sites. The support threshold varies from 0.5 % to 2 %. The running time of two approaches have been computed.
- We fix the support threshold at 2 %. The number of old sites varies from 2 to 8. Hence, the number of new sites changes from 8 to 2. We also measure the running time to compare. Note that running time of **TASSA14** remains unchanged although the number of new sites is changed. The reason is that the protocol always runs on data of all sites.

We have implemented the protocol in [15] and our proposed protocol in Java. Each site is running on a Windows 7 64-bit OS with features: 8 GB of RAM, Intel Xeon E5-1620 3.6 GHZ of CPU. All sites are connected to each other via Gigabit ethernet cable.

## 6.3 Experimental Results

Figure 2 presents the total running time of two approaches: **TASSA14** method and our **INCRE** method when support threshold changes. From the results, we can see that the running time of two methods are decreasing when we increase support threshold. This is a known result as when support thresholds are bigger, the number of candidate itemsets are smaller. Thus both protocols will have to run less iterations. The interesting result in this experiment is that **INCRE** takes less time than **TASSA14** to complete the large itemset mining. This can be explained as follows. While **TASSA14** has to run on datasets of all 10 sites, **INCRE** runs on datasets of 5 new sites. Our protocol then makes use of results from old 5 sites (which is generated before) along with results from 5 new sites to compute the final large itemsets. Experimental results show that our method can reduce total running time about 50 % comparing with thaose of **TASSA14**.

Figure 3 shows the total running time of the two methods when the number of old site changes (total number of sites is still 10). If there are 2 old sites, then there are 8 new sites and so on. We can see that total running time of **TASSA14** remains unchanged. The total running time of our protocol is decreasing when the number of old sites increase (or number of new sites decrease). This can be explained as follows. When there are many old sites, our protocols makes use of old results from those sites and hence cut down the time to run on them again. The more old sites there are, the more time our protocol can save. If there are less new sites, then the protocols takes less time to access new datasets and compute new large itemsets. In the real world, there is normally less new

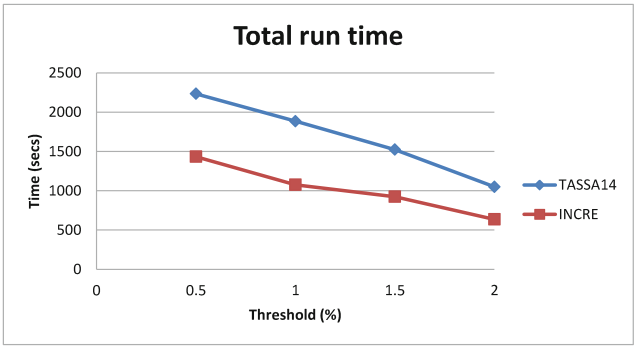

**Fig. 2.** Total run time of **TASSA14** versus **INCRE** when changing support threshold. Fixed 5 old sites and 5 new sites.

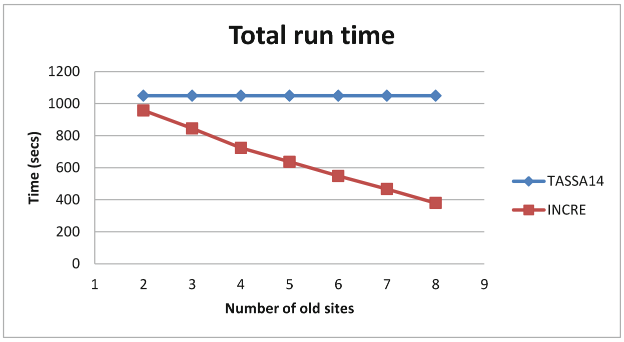

**Fig. 3.** Total run time of **TASSA14** versus **INCRE** when changing the number of old sites. Threshold fixed at 2 %.

sites than old sites. Our protocol thus expects to much outperform **TASSA14**. In this experiment, when there is 8 old sites and 2 new sites, our protocol can reduce total running time at 70 %.

## 7 Summary

In this paper, we have proposed a novel incremental protocol for secure mining of association rules in horizontally distribution. The protocol improves the fast incremental algorithm in [16] in term of privacy. One of the main improvements is making use of Negative Border in secure way to boost the performance of protocol. Thanks to this feature, our protocol outperforms the latest privacy preserving association rule mining proposed by Tassa in [15].

As we know that in real world, datasets can be deleted or some parties want to leave the mining process, a protocol to adapt with this changes is expected. In the future work, we will apply the concept of our protocol to deal with "deletion" of data.

**Acknowledgments.** This research is supported by the Singapore Maritime Institute.

## References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of 20th International Conference on Very Large Databases, VLDB, pp. 487–499. Morgan Kaufmann (1994)
2. Ayan, N., Tansel, A., Arkun, E.: An efficient algorithm to update large itemsets with early pruning. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 287–291. ACM (1999)
3. Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y.: A fast distributed algorithm for mining association rules. In: Fourth International Conference on Parallel and Distributed Information Systems, pp. 31–42. IEEE (1996)
4. Cheung, D., Han, J., Ng, V., Wong, C.: Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 106–114. IEEE (1996)
5. Cheung, D.W., Lee, S.D., Kao, B.: A general incremental technique for maintaining discovered association rules. In: Proceedings of the Fifth International Conference On Database Systems For Advanced Applications, pp. 185–194 (1997)
6. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. SIGKDD Explor. Newsl. **4**(2), 28–34 (2002)
7. Duan, Y., Canny, J.F., Zhan, J.Z.: Efficient privacy-preserving association rule mining: p4p style. In: CIDM, pp. 654–660. IEEE (2007)
8. Goldreich, O.: Secure multi-party computation. Manuscript (2002)
9. Han, S., Ng, W.-K.: Privacy-preserving genetic algorithms for rule discovery. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 407–417. Springer, Heidelberg (2007)
10. Ioannidis, I., Grama, A.: An efficient protocol for yao's millionaires' problem. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, p. 6, January 2003
11. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans. Knowl. Data Eng. **16**(9), 1026–1037 (2004)
12. Lee, C., Lin, C., Chen, M.: Sliding-window filtering: an efficient algorithm for incremental mining. In: Proceedings of the tenth international conference on Information and knowledge management, pp. 263–270. ACM (2001)
13. Mannila, H., Toivonen, H.: On an algorithm for finding all interesting sentences. In: Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research. Citeseer (1996)
14. Park, J.S., Chen, M.-S., Yu, P.S.: An effective hash-based algorithm for mining association rules. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 1995, pp. 175–186. ACM, New York (1995)
15. Tassa, T.: Secure mining of association rules in horizontally distributed databases. IEEE Trans. Knowl. Data Eng. **26**(4), 970–983 (2014)
16. Thomas, S., Bodagala, S., Alsabti, K., Ranka, S.: An efficient algorithm for the incremental updation of association rules in large databases. In: Knowledge Discovery and Data Mining, pp. 263–266 (1997)
17. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. J. Comput. Secur. **13**(4), 593–622 (2005)

18. Veloso, A., Meira Jr., W., De Carvalho, M., Pôssas, B., Parthasarathy, S., Zaki, M.: Mining frequent itemsets in evolving databases. In: SIAM International Conference on Data Mining (2002)
19. Wong, W.K., Cheung, D.W., Hung, E., Liu, H.: Protecting privacy in incremental maintenance for distributed association rule mining. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 381–392. Springer, Heidelberg (2008)
20. Yao, A.C.: Protocols for secure computations. In: SFCS 1982: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE Computer Society, Washington (1982)
21. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science, pp. 162–167 (1986)