A High Collusion-Resistant Approach to Distributed Privacy-preserving Data Mining

Shintaro Urabe,† Jiahong Wang,† Eiichiro Kodama† and Toyoo Takata†

Data mining across different companies, organizations, online shops, or the likes, called sites, is necessary so as to discover valuable shared patterns, associations, trends, or dependencies in their shared data. Privacy, however, is a concern. In many situations it is required that data mining should be conducted without any privacy being violated. In response to this requirement, this paper proposes an effective distributed privacy-preserving data mining approach called CRDM (Collusion-Resistant Data Mining). CRDM is characterized by its ability to resist the collusion. Unless all sites but the victim collude, privacy of a site cannot be violated. Considering that for such applications that need not so high a level of security, excess security assurance would incur extra costs in communication, an extension scheme is also presented so that communication cost can be restrained while still maintaining a user-specified level of security. Results of both analytical and experimental performance study demonstrate the effectiveness of CRDM.

1. Introduction

The association mining algorithm is a data mining algorithm for analyzing relationship among data items in a database. It is widely used to discover potentially valuable patterns, associations, trends, and dependencies in data. For example, in a supermarket, transactions, which are sets of items such as {milk, cheese, bread, meal} that customers purchase at a time, are processed, and transaction data are stored in a transaction database. By mining associations in the transaction database, it may be found that customers who purchased milk frequently purchased bread too. Such association rules will help purchase items more economically, arrange showrooms more reasonably, and schedule advertisements more effectively.

Most of the association mining algorithms are based on the Apriori property. FDM (Fast Distributed Mining of association rules) is an influential Apriori-based algorithm for distributed systems. It assumes a system of M sites, each of which has a partition DBi (1 ≤ i ≤ M) of a transaction database DB. The goal of FDM is to find all globally frequent k-itemsets for all k ≥ 1 and support counts for these itemsets. Here k-itemset denotes such an item set that contains k items, support count (resp. support) of an itemset X denotes the number (resp. the rate) of transactions that contain X, frequent itemset denotes the itemset whose support satisfies a user-specified support threshold. FDM is summarized below.

Step1 (Candidate sets generation): Generate candidate sets CGi(k) on the basis of the set of globally frequent (k−1)-itemsets locally supported at site Si. (This is done by using the classic Apriori candidate generation algorithm.)

Step2 (Local pruning): For each X ∈ CGi(k), scan database DBi to compute X’s local support count. If X is locally frequent at Si, it is included in the LLi(k) set, which is the set of all locally frequent k-itemsets at site Si. (It is clear that if X is globally frequent, it must be supported at some site.)

Step3 (Support count exchange): Candidate sets in LLi(k) are broadcasted, and each site computes the local support count for the itemsets in ∪i LLi(k).

Step4 (Broadcast mining results): Each site broadcasts the local support counts for itemsets in ∪i LLi(k). From these, each site computes the set of globally frequent k-itemsets.

In steps 3 and 4, candidate sets and their support counts have to be sent to other sites. Thus for FDM, privacy of participating sites may be violated. Here privacy of a site is defined as follows:

Definition 1 (Privacy). Privacy of a site is
defined as a three-tuple: (1) contents of transactions processed, (2) frequent (or infrequent) itemsets occurred, and (3) values of supports for itemsets mined, at that site.

Most of the existing association mining algorithms do not take privacy into consideration. However, if the privacy cannot be preserved, association mining cannot be conducted in many applications. As examples, (1) several credit card agencies such as banks would cooperate to mine the union of their databases to detect patterns matching credit card frauds, and (2) hospitals, insurance companies, and public health agencies would cooperate to mine the union of their databases to identify public health problems. In both examples extensive data sharing is necessary, however, factors such as trade secrets and personal data of customers stand in the way of the necessary sharing, and prevent building a centralized data warehouse.

Generally it is required that we can accomplish the desired results while still preserving privacy of individual site. More concretely, it is required that no site should be able to learn contents of a transaction at any other site, what itemsets are supported by any other site, or the specific value of support for any itemset at any other site, unless that information is revealed by knowledge of one’s own data and the final result. For example, if an itemset is supported globally but not at one’s own site, it can be deduced that at least one other site supports the itemset.

In some application environments, the collusion may pose a threat to privacy. Imagine that association mining is conducted among sites that belong to two different parties with conflict of interests. Let \( M \) be the total number of sites. In the worst case, there may have a collusion of \( M - 1 \) sites. Assume that two of those sites belong to Party A, and the remainder belongs to Party B. Party A would require that, even if all sites of Party B collude, its privacy could not be violated. To meet this requirement, we adopt the above privacy definition, and address the subject of privacy-preserving distributed association mining from the viewpoint of the collusion resistance, as defined below:

**Definition 2 (Privacy-preserving data mining).** A distributed data mining algorithm is said to be a privacy-preserving data mining algorithm if the privacy of any participating site cannot be violated unless some sites collude.

**Problem Statement 1.** Given a set of sites connected with a network, and given a user-specified support threshold denoted as MinSup, mine the complete set of such globally frequent itemsets whose supports are not less than MinSup, under the condition that the privacy of any site cannot be violated unless some sites collude.

The contribution is a collusion-resistant privacy-preserving distributed association mining algorithm, called CRDM (Collusion-Resistant Data Mining). CRDM is characterized by (1) its ability to preserve privacy; (2) its ability to resist collusion (Unless all sites but the victim collude, privacy of a site cannot be violated); (3) its distributed processing ability without needing any third-party sites.

For applications with numerous sites, CRDM’s degree of collusion resistance may be too high to be necessary, while the communication cost may become a problem in this case. A scheme for extending CRDM is suggested so that communication cost can be restrained while still maintaining a user-specified degree of collusion resistance.

Performance of CRDM has been studied analytically and experimentally. Results of performance studies are reported for demonstrating the effectiveness of CRDM. The results of performance study also show that CRDM has better performance than the existing approaches.

The remainder of this paper is organized as follows. A review of related work is presented in Section 2. On the basis of definitions and system model given in Section 3, CRDM is presented in Section 4. Performance study is described in Section 5. A discussion of extending CRDM is given in Section 6. Conclusion appears in Section 7.

## 2. Related Work

A comprehensive survey of privacy-preserving data mining can be found in Ref. 24). Most previous work \(^{2}-^{4},^{9},^{17},^{19},^{22}\) has basically addressed one issue: preserving privacy of sites that participate in data mining by distorting the data values. The idea is that the distorted data does not reveal any private information and thus is safe to use for data mining. The assumption with these approaches is that the values must be kept private from any other sites, which is different from our work. We assume that a site can see the union of private data of all sites that participate in data mining, but does not know which data belongs to which
site. In return, for our proposed approach, exact rather than approximate mining results can be obtained.

Another kind of approaches is the cryptography-based approach\(^\text{16,17}\). Take the algorithm proposed in Ref. 16), called SFDM, as an example. SFDM follows the FDM, with special protocols replacing the broadcasts of \(LL_{i(k)}\) and support counts of items in \(LL_{i(k)}\), where the commutative encryption such as RSA is used. In addition to the consideration in security, the commutative encryption is used mainly because of its special properties that can be helpful in checking whether two encrypted items are equal without revealing them. SFDM is a highly influential privacy-preserving data mining algorithm, but it has some weaknesses: (1) It cannot function normally if the collusion exists. (2) It is very heavy since the commutative encryption such as RSA is adopted, which tends to be very time-consuming and resource-consuming. For example, for RSA on an Intel PentiumIV 1.8 GHz processor, one round of encryption and decryption would take about 25.1 ms for 1,024-bit key, and 180.1 ms for 2,048-bit key\(^\text{18}\). By our experiments, on an Intel Core Duo 1.83 GHz processor, using Bouncy Castle Crypto APIs for JAVA, one round of encryption and decryption would take about 13.9 ms for 1,024-bit key, and 88.1 ms for 2,048-bit key. Considering that a large number of candidate itemsets tend to be generated in data mining, encrypting and decrypting each of them would cost too much time, resources, and network bandwidth, and make SFDM unsuitable for mining association rules in the large database.

A random number-based approach called SDDM ( Secure Distributed Data Mining) was proposed in Ref. 10). Compared with SFDM, SDDM has higher degree of collusion resistance and makes lighter use of system resources. If more than three sites collude, however, privacy may be violated. Similar to SDDM, CRDM uses mainly the technique of sharing a secret among sites instead of the commutative encryption to preserve privacy, therefore CRDM is expected to be very light. Different from SDDM, CRDM’s degree of collusion resistance can be variable, and in the case of specifying the highest level of security, it is proportional to the number of sites participating in data mining.

In Ref. 23) an approach that can also get the exact mining results was proposed. This approach is characterized by its low communication cost. This is mainly because a specified combiner is adopted, which plays the role of a trusted third party. Similarly, an architecture was proposed in Ref.15) where secure multiparty techniques developed in the cryptography domain can be easily used for data mining purposes. In this architecture, some kind of intermediate parties are also used where the collected information can be stored securely. For CRDM, the third-party is unnecessary, and therefore it can be used more widely.

Crowds-based approaches\(^\text{21}\) are used for protecting users’ anonymity, and therefore can be used for privacy-preserving data mining. When used in data mining, however, a large number of sites are needed to achieve a high degree of collusion resistance. MIX-based approaches\(^\text{5}\) can be used to hide the relationship between incoming and outgoing messages, and therefore can also be used for preserving the privacy. When used in data mining, however, additional sites called mixes are necessary, and for environment where there are no mixes available, MIX-based approaches cannot be used.

3. Definitions and System Model

A homogeneous distributed system of \(M\) sites, denoted as \(\{S_1, S_2, \cdots, S_M\}\), is considered. Note that a homogeneous distributed system is such a system in which sites have similar frequent itemset distributions. A transaction database is a set of transactions denoted as \(DB = \{T_1, T_2, \cdots, T_n\}\). \(T_i (1 \leq i \leq n)\) is a set of items such that \(T_i \subseteq I\), where \(I = \{i_1, i_2, \cdots, i_m\}\) denotes the set of items that constitute the schema of \(DB\) (i.e., the item domain). \(DB\) is partitioned horizontally into \(\{DB_1, DB_2, \cdots, DB_M\}\), with \(DB_i\) being located at \(S_i (1 \leq i \leq M)\). It is assumed that every site knows the item domain \(I\).

Let \(X_{sup}\) (called the local support count) and \(X_{sup}\) (called the local support count at \(S_i\)) denote the number of transactions containing itemset \(X\) in \(DB\) and in \(DB_i\), respectively. The global support count of \(X\) is given as \(X_{sup} = \sum_{i=1}^{M} X_{sup,i}\). Itemset \(X\) is said to be frequent if \(X\)’s support is not less than a pre-specified support threshold \(s (s \leq 1)\). Especially, \(X\) is said to be locally frequent at \(S_i\) if \(X_{sup} \geq s \times |DB_i|\), and \(X\) is said to be globally frequent if \(X_{sup} \geq s \times |DB|\).

Let \(L_k\) denote the set of all globally frequent \(k\)-itemsets. Given a \(DB\), the goal of CRDM is to discover all itemsets that their supports are
not less than a user-specified support threshold, i.e., to discover the set of $L_k$ for $k \geq 1$ and the support counts for these itemsets. Particularly, it is required that the privacy must be preserved.

An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \phi$. $X \rightarrow Y$ holds in $DB$ with support $S$ if $S\%$ of the transactions in $DB$ contains $X \cup Y$, and with confidence $C$ if $C\%$ of transactions in $DB$ that contain $X$ also contain $Y$. We do not consider the association rules further in this paper. In fact, using the globally frequent itemsets, association rules can be computed easily.

As in Refs. 10) and 16), we also adopt the semi-honest model\(^{11)}\): A site is assumed to follow the algorithm, but is free to later use what it sees during execution of the algorithm to compromise security. As explained in Ref. 16): “This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. Also, a protocol that is buried in large, complex software can not be easily altered”.

4. Mining Patterns Securely

In this section we present CRDM — a Collusion-Resistant privacy-preserving Data Mining algorithm. CRDM will be presented in two steps: (1) to give an anonymous communication method called $SecureSum$, and (2) on the basis of $SecureSum$ to present the algorithm CRDM.

We will focus on the approach adopted by CRDM instead of its implementation, so that it is flexible. In fact, CRDM can be customized differently. For example, a light version is possible by specifying a smaller value for the dividing number of object $V$ in $SecureSum$; however, in case that there are few parties, this may risk privacy with high probability. We may also divide sites into groups and apply CRDM approach over groups to reduce communication cost; however, grouping may be costly, and has to be controlled by human managers.

4.1 Totaling Values with $SecureSum$

In CRDM, anonymous communication is required for totaling up the instances of a specified object (itemset or itemset’s support count) at every site that participates in data mining.

Problem Statement 2 (Total up secretly every instance of an object) Given a system of $M$ sites, and an object denoted by $V$. $V_i$ is $V$’s instance at site $S_i$ ($0 \leq i < M$). Calculate $\sum_{i=0}^{M-1} V_i$ in such a way that $V_i$ cannot be known by any others unless some sites collude with each other.

The anonymous communication method is given in procedure $SecureSum$ shown in Fig. 1. A technique of so-called “sharing and masking” is used to preserve $V_i$’s anonymity, and effort is
made to reduce the communication cost.

Example 1. In Fig. 2, a system of 6 sites is used to illustrate SecureSum. There are two phases. In Phase 1, site $S_i$ $(1 \leq i \leq 5)$ divides $V$’s instance $V_i$ randomly into $(6-i)$ parts: $\{V_{i,j}, V_{i,j+1}, \cdots, V_{i,5}\}$, and sends $V_{i,j}$ to $S_j$ $(i+1 \leq j \leq 5)$. In Phase 2, site $S_i$ $(1 \leq i \leq 5)$ sends $(V_{i,1} + \sum_{j=1}^{i-1} V_{j,2})$ to $S_0$. Finally, site $S_0$ obtains the secure sum $\sum_{i=0}^{5} V_i$.

Next we give an analysis on degree of collusion resistance and communication cost of procedure SecureSum. They are given as its two properties: Property 1 and Property 2.

Property 1 (Degree of collusion-resistance) Given a system of $M$ sites. With using the procedure SecureSum, site $S_i$ $(0 \leq i < M)$ cannot know $V_j$ $(0 \leq j < M$, and $j \neq i)$ unless it colludes with all other sites except $S_i$. That is, the degree of collusion-resistance is $M - 2$.

Proof sketch. We prove this property in two cases:
Case 1 (For site $S_0$): $S_0$ does not send any data concerning $V_0$ to any other site as it is. Before sent to other sites, $V_0$ is protected with all values from all the other sites. Therefore, $V_0$ cannot be known unless all the other sites collude.

Case 2 (For site $S_i$, $1 \leq i < M$): Assume $S_x$ $(x \neq i)$ has known $V_i$. It means that $S_x$ must have known $\{V_{i,1}, V_{i,1+1}, \cdots, V_{i,M-1}\}$. To know $\{V_{i,1+1}, V_{i,1+2}, \cdots, V_{i,M-1}\}$, it is required for $S_{i+1}, S_{i+2}, \cdots, S_{M-1}$ to collude with each other. To know $V_{i,1}$, it is required for $S_1, S_2, \cdots, S_{i-1}$ and $S_0$ to collude. Therefore, $V_i$ cannot be known unless all the other sites collude.

Since $V_i$ is uniformly randomly divided over $(-\infty, +\infty)$ (Note that in fact, computers have a limitation to single integer numbers), given that some sites have collected some dividing results of $V_i$, they cannot make any correct guess at $V_i$ unless they also know the remainder, including the $V_{i,1}$.

By this we have shown that $S_x$ cannot know $V_i$ or make guess at $V_i$ unless it colludes with all other sites except $S_i$. Thus we have the property.

Property 2 (Communication Cost) Given a system of $M$ sites. With using the procedure SecureSum, communication cost in terms of

1. the message number is $M(M-1)/2$
2. the time requirement is $(M-1)/2$, where $T$ is the average time for sending a message from one site to another site.

Proof sketch. We prove this property in two steps.

(1) For message number: For site $S_i$ $(1 \leq i < M)$, $M-i-1$ messages are sent in Phase 1, and one message in Phase 2. Since we have $M$ sites, in total, the number of messages can be estimated as $M(M-1)/2$.

(2) For time requirement: In Phase 1, $S_i$ $(1 \leq i < M)$ sends messages for $V_{i,j}$ $(i+1 \leq j < M)$ sequentially. Among all sites, $S_1$ sends the maximum number of messages, and the number is $M - 2$. Since for any sites $S_i$ and $S_j$ $(i \neq j)$, messages for $V_i$ and $V_j$ are sent simultaneously, if we take all the simultaneously-occurring messages as one message, the number of messages
Fig. 3 An example to illustrate CRDM.

should be \( M - 2 \). Then, assume we have enough network bandwidth, the time requirement for sending the messages should be \((M - 2)T\). Similarly, in Phase 2 one more message is sent. Therefore the total time requirement can be estimated as \((M - 1)T\).

Thus we have the property. \(\Box\)

For being unguessable \( V_i \) from \( V_{i,j} \), \( V_i \) must be uniformly randomly divided over \((-\infty, +\infty)\), and the dividing results must be kept in secret. For the former, considering that nowadays real random number generators can be obtained in little cost, with using a real random number generator and according to its results, \( V_i \) can be randomly divided. For example, \( V_i = 10 \) can be randomly divided into 2 and 8, \(-36 \) and 46, or any other pair of numbers. For the latter, it is up to the site to keep the dividing results in secret, so as to preserve its own privacy. In fact, no site would tell others the dividing results of its own, and thus, other sites cannot know what the dividing results are, unless they collude.

Note that wiretapping communication may pose a threat to privacy. In Fig. 2, for example, \( V_{1,1} - V_{1,5} \) may be collected, and thereby \( V_i \) may be known. As stated in Section 3, in this paper we adopt the semi-honest model, and thus we do not take the wiretapping problem into consideration. In fact, this problem can be easily solved using tools such as OpenSSL or cryptographic algorithms such as AES (Advanced Encryption Standard).

4.2 Mining Patterns with CRDM

CRDM is given as Algorithm 1 shown in Fig. 4. It consists of two phases. In Phase 1, as the candidates of globally frequent itemsets, all such itemsets are found that are locally frequent at at least one site. In Phase 2, using results of Phase 1, the globally frequent itemsets are obtained. We use Example 2 and Fig. 3 to illustrate CRDM.

Example 2 (Refer to Fig. 3) Assume that we have 6 sites, have finished the first two iterations of CRDM, and thus have known the globally frequent 1-itemsets and 2-itemsets, represented by \( L_1 \) and \( L_2 \), respectively. We are to generate the \( L_3 \).

To start the 3rd iteration, \( ItemsetLen \) is increased by 1, so as to generate the \( L_3 \). With joining \( L_2 \), sites \((S_i, 0 \leq i < 6)\) share the set of excess frequent 3-itemset candidates, denoted by \( CG_3 \).
Algorithm 1: CRDM — Collusion-Resistant Data Mining

input: (1) \( \{ S_i \}_{0 \leq i < M} \): A set of sites participating in distributed data mining, \( M \geq 3 \)
(2) \( I \): the set of items
(3) \( DB_i \): database located at site \( i \) \( (0 \leq i < M) \); \( DB = \bigcup_{i=0}^{M-1} DB_i \)
(4) \( Min_{sup} \): support threshold

output: A set of itemsets with supports not less than \(( \geq ) Min_{sup}\)

CRDM begin

\( ItemsetLen \leftarrow 0; \)

repeat

/* Frequent \( ItemsetLen \)-itemsets are to be generated for \( ItemsetLen = 1, 2, 3, \ldots \) */

\( ItemsetLen \leftarrow ItemsetLen + 1; \)

PHASE1(get the candidates of frequent itemsets) begin

/* Every site prepares locally frequent itemsets in an array denoted by \( A_i \) */

foreach site \( S_i \) do

if \( ItemsetLen = 1 \) then \( CG'_{ItemsetLen} \leftarrow I \) sorted lexicographically;
else \( CG'_{ItemsetLen} \leftarrow \) results joining \( L_{ItemsetLen-1} \) with itself, sorted lexicographically;

\( ItemsetNum \leftarrow |CG'_{ItemsetLen}| \), and initialize an array \( A_i[1..ItemsetNum] \);

for \( j = 1 \) to \( ItemsetNum \) do

if the \( j \)th element of \( CG'_{ItemsetLen} \) is locally frequent then \( A_i[j] \leftarrow 1; \)
else \( A_i[j] \leftarrow 0 \);

/* Securely total up \( A_i \) for \( i = 0 \) to \( M - 1 \), and return the result to site 0 */

call SecureSum(\( \{ S_i \}_{0 \leq i < M} \), \( A_i[j] \) \( (0 \leq i < M, 1 \leq j \leq ItemsetNum) \), and

return the results in \( B = < \sum_{i=0}^{M-1} A_i[1], \sum_{i=0}^{M-1} A_i[2], \ldots, \sum_{i=0}^{M-1} A_i[ItemsetNum] >; \)

/* Announce the itemsets that are locally frequent at at least one site */

for site \( S_0 \) do

for \( j = 1 \) to \( ItemsetNum \) do

if \( B[j] > 0 \) then \( CG_{ItemsetLen} \leftarrow \) the \( j \)th itemset of \( CG'_{ItemsetLen} \);

\( ItemsetNum \leftarrow |CG_{ItemsetLen}|; \)

sort \( CG_{ItemsetLen} \) in lexicographical order;

broadcast \( CG_{ItemsetLen} \) and \( ItemsetNum \) to all sites;

end PHASE2(get the frequent itemsets and their support counts) begin

/* Every site prepares local support counts in an array denoted by \( A_i \) */

foreach site \( S_i \) do

find the local support of each element in \( CG_{ItemsetLen}; \)

Initialize an array \( A_i[1..ItemsetNum]; \)

for \( j = 1 \) to \( ItemsetNum \) do \( A_i[j] \leftarrow \) local support count of the \( j \)th itemset of \( CG_{ItemsetLen}; \)

/* Securely total up \( A_i \) for \( i = 0 \) to \( M - 1 \), and return the result to site 0 */

call SecureSum(\( \{ S_i \}_{0 \leq i < M} \), \( A_i[j] \) \( (0 \leq i < M, 1 \leq j \leq ItemsetNum) \), and

return the results in \( B = < \sum_{i=0}^{M-1} A_i[1], \sum_{i=0}^{M-1} A_i[2], \ldots, \sum_{i=0}^{M-1} A_i[ItemsetNum] >; \)

/* Announce the globally frequent \( ItemsetLen \)-itemsets */

for site \( S_0 \) do

for \( j = 1 \) to \( ItemsetNum \) do

if \( B[j] \geq Min_{sup} \ast |DB| \) then \( L_{ItemsetLen} \leftarrow B[j] \) and the \( j \)th element of \( CG_{ItemsetLen} \)

\( ItemsetNum \leftarrow |L_{ItemsetLen}|; \)

broadcast \( L_{ItemsetLen} \) to all sites;

end

until no new globally frequent itemsets can be generated;

end

Fig. 4 CRDM — Collusion-Resistant Data Mining.
In PHASE1, as the candidate set of globally frequent 3-itemsets (denoted as CG₃), the itemsets that are frequent at at least one site are collected in four steps: (1) For each itemset in CG’₃ that is locally frequent at Sᵢ, the corresponding element of an array called Aᵢ is set to 1, and all Aᵢ’s other elements are set to 0. (2) \( \sum_{i=0}^{5} A_i \) is calculated using SecureSum, and the result is given to S₀ in an array called B. (3) CG₃ is derived from B: the elements whose values are larger than 0 correspond to the candidates. (4) S₀ broadcasts CG₃ and the total number of candidates (denoted by ItemsetNum).

In PHASE2, the globally frequent 3-itemsets and their support counts are calculated as follows. First, for each candidate in CG₃, the sum of its support counts at individual sites is calculated in two steps: (1) each element of Aᵢ is initialized to a local support count of the corresponding candidate; (2) \( \sum_{i=0}^{5} A_i \) is calculated, as in PHASE1. Then, using the result, S₀ obtains \( L_3 \) and the corresponding support counts, and broadcasts them to other sites. \( L_3 \) will be used in the next iteration to get \( L_4 \).

Next we give an analysis on degree of collusion resistance and communication cost of algorithm CRDM. They are given as its two properties: Property 3 and Property 4

**Property 3** CRDM is a privacy-preserving algorithm. Its degree of collusion resistance is \( M - 2 \), where \( M \) is the number of sites.

**Proof sketch.** This property follows from Property 1 and Definition 2, since in CRDM, the private data transferring is solely based on the SecureSum. □

**Property 4** (Communication Cost). For CRDM, communication cost in terms of the message number is estimated as \( M(M-1)+2 \), and in terms of the time requirement is estimated as \( 2MT \), where \( M \) is the number of sites, and \( T \) is the average time for sending a message from one site to another site.

**Proof sketch.** CRDM has two phases. In Phase 1, the set of such itemsets that are locally frequent at at least one site is securely obtained. On the basis of Phase 1, in Phase 2 the set of such itemsets that are globally frequent is securely obtained. For being secure, in either phase, the procedure SecureSum is used. In addition, an additional broadcast message is sent for announcing results. Then by the Property 2, we have this property. □

5. **Performance Study**

A comprehensive analytical and experimental performance study has been conducted. In this section we discuss results of performance study.

5.1 Comparison by Analysis

As stated in Section 2, there are two algorithms similar to CRDM: SFDM [16] and SDDM [10]. In addition, two other approaches can also be used for privacy-preserving data mining: the Crowds-based approach [21] and the MIX-based approach [5, 12, 20]. In the following we give a comparison of Crowds, SFDM, SDDM, and CRDM. We do not consider MIX-based approaches since they require special hardware support, and therefore cannot be applied to our assumed application environment.

**Figure 5** shows results of a comparison on degree of collusion resistance. As it is shown, CRDM outperforms the others significantly. Especially, the more the participating sites, the higher the degree of collusion-resistance. Note that for Crowds-based algorithms, a forwarding probability of 19/20 is used so that a high degree of collusion resistance can be achieved. For SFDM, SDDM, and CRDM, the degrees of collusion resistance are achieved in 100% possibility. For Crowds, however, we cannot assure no privacy violation, and for fairness we assumed a slightly high security (90%).

**Figure 6** shows results of a comparison on communication cost. By Property 4, for CRDM this cost is estimated in two ways: (1) \( M(M-1)+2 \) (line CRDM1), and (2) \( 2M \) by considering the asynchronous ones only (line CRDM2; this is also an estimate of the communication time requirement). Similarly, for SFDM this cost is also estimated in two ways: (1) \( M^2 + 3M + 1 \) (line SFDM1), and (2) \( 3M + 2 \) (line SFDM2). For Crowds-based algorithms, we used the forwarding probability of 19/20 as above to com-
compute the average message number. Figure 6 shows that compared with SFDM (resp. SDDM), CRDM has lower (resp. higher) communication cost in terms of the message number. It is also shown that CRDM has the lowest cost in terms of the communication time requirement, given that we have a network with a wide enough bandwidth.

For CRDM, encrypt communication may be necessary to prevent wiretapping. We use AES for performance study. It is found that in comparison with the cost of data mining, the cost of encrypt communication is too small to degrade performance very much, as explained in the following:

1. It is reported that AES of OpenSSL has a throughput of 20.7 MBytes/sec on an Intel Celeron 1.3 GHz processor. Our experiments show that AES of Bouncy Castle Crypto APIs for JAVA has a throughput of 27.8 MBytes/sec on an Intel Core Duo 1.83 GHz processor. For experiments reported in Section 5.2, however, the private data per site to be sent is even not larger than 20.7 MBytes. Of course, those data are sent in more than one round in practice; and each round would incur some additional overhead for network communication.

2. The time estimation of encrypt communication is given below. A database used in Refs. 7) and 16) is adopted, which has a domain of 1,000 items, and contains 500 K transactions, larger than those used in our experiments. The total number of candidates is about 100 K at 1% support, giving a size of data about 1.5 MBytes per site, given that each piece of privacy is coded with 128 bits. Assume we have 17 sites, and each has a database.

   **Lower bound**: Assume all globally frequent patterns have the same length, meaning that all data are sent within one iteration of CRDM. According to Property 4, the amount of asynchronous data is $3M$ MBytes, where $M$ represents site number. Then the time is $2.5$ (resp. $1.8$) seconds for Celeron 1.3 (resp. Core Duo 1.83) GHz processor.

   **Upper bound**: Let $N$ be the maximum length of globally frequent patterns, implying there are $N$ iterations. Suppose that each iteration takes $2.5$ (or $1.8$) seconds, and $N$ is 8 (This is a large enough value; Reducing support threshold will increase $N$, but accordingly execution time will increase dramatically). Then the time is $20$ (or $14.4$) seconds.

In summary, taking the execution times given in Section 5.2 as reference points, it can be said that comparatively the added time cost is very small.

Using the same database as above we compare CRDM with SFDM in terms of the cost of privacy-preserving protocols. Since they have no difference except for the protocol, this would also be a comparison in terms of the execution time.

1. SFDM has two phases, in parallel with those of CRDM. Phase 1 finds the candidates in three steps: (Step1) Each site encrypts every locally frequent itemset using commutative encryption such as RSA. The encrypted itemsets are then passed to a common site to eliminate duplicates. (Step2) The resulting itemsets are then passed to each site, and each site decrypts each itemset. Phase 2 circulates a message through every site so that each of the locally supported itemsets is tested to see if it is supported globally. Let $T_{E}$ be the time for encrypting an itemset, $T_{D}$ be the time for decrypting an itemset, $M$ be the site number. To be a lower bound estimation, assume each site holds 100 K/M candidates. Ignoring all simultaneously-occurring costs, the time is $100 K (T_{E}+T_{D}+M) + [39\% * 100 K (T_{E}+T_{D})]$. The second part is for the secure comparison at the end of Phase 2, which is derived from the estimation in Ref. 16).

2. For CRDM, the cost occurs mainly when securely dividing itemsets and support counts, which is based on generating random numbers. Let $T_{R}$ be the time for one round of generating and dividing operations. According to Property 4, the time is $2 * 100 K * T_{R} * M$.

Comparing the above two estimations we can safely conclude that CRDM is faster than SFDM. Note that (1) $T_{R} < T_{AES}$, where $T_{AES}$
is the time for encrypting a data block with using AES. (2) By our experiments, AES is 108.6 times faster than RSA, and \( T_E \) is only 5% of \( (T_E + T_D) \).

5.2 Comparison by Experiments

The above analysis convinces us that CRDM has high degree of collusion resistance, and although CRDM has relatively high communication cost in terms of message number, this would not affect performance very much, since most of those messages occur simultaneously. In this section, we verify the analytical results about the communication cost. CRDM is compared with FDM by experiments. The objective is to take FDM as a reference point, and to see how CRDM is different from FDM due to the added communication cost.

5.2.1 Experimental Environment

The testbed is implemented using SUN workstations connected with 100 Mbps Ethernet LAN, which is an active subnet of a campus network. Each workstation has Ultra-SPARC 550 MHz processor and 512 MB memory. CRDM, SDDM, and FDM are all programmed with JAVA.

Synthetic data set is used, which was generated using an IBM data set generator from the IBM Almaden Research center (http://www.almaden.ibm.com/software/quest/Resources/index.shtml). This generator has been used in most data mining studies. The generated data set has \( n_{\text{trans}} \) transactions with an average length of 10 items, where \( n_{\text{trans}} \) is a parameter used in experiments. Number of different items in this data set (i.e., the size of item domain \( I \)) is 1,000, and the frequent patterns have an average length of 4. Using this data set we performed four experiments as follows. Considering the objective of our experiments, we think that this data set is enough to serve our purposes.

5.2.2 Experimental Results

The execution time is used as a performance metric, which is defined as the time from a data mining is started to it is finished, measured in seconds.

Experiment 1: The objective of this experiment is to examine scalability with respect to database size. Results are given in Fig. 7. It is shown that CRDM has higher execution times than FDM, and lower execution times than SDDM. For example, at database size of 50 K, for CRDM, SDDM, and FDM, execution times are 254.7, 266.1, 250.4, respectively for support threshold of 5\%, and 81.3, 83.4, 79.5, respectively for support threshold of 10\%. The reason is as follows. Both SDDM and CRDM are based on FDM, with special protocols being used to preserve privacy. The cost of the protocol results in that both SDDM and CRDM have longer execution times than FDM. On the other hand, since the cost mainly concerns transferring data between sites, and CRDM has the lowest cost in terms of the communication time requirement, CRDM has shorter execution times than SDDM. This coincides with the analytical results in Section 5.1. More importantly, it is shown that CRDM behaves the same way as FDM does, meaning that CRDM and FDM have the similar scalability with respect to database size.

Experiment 2: The objective of this experiment is to examine performance behavior of each algorithm under different support threshold values. Results are given in Fig. 8. It is shown that CRDM has higher execution times than FDM, and lower execution times than SDDM. For example, at support threshold of 5\%, for CRDM, SDDM, and FDM, execution times are 100.4, 110.8, 98.3, respectively for \( n_{\text{trans}} \) of 10 K, 176.9, 185.0, 172.1, respectively for \( n_{\text{trans}} \) of 30 K, and 254.7, 266.1, 250.4, respectively for \( n_{\text{trans}} \) of 50 K. The reason has been explained above. In addition, it is shown that CRDM behaves the same way as FDM does, meaning that CRDM and FDM have the similar performance behavior to FDM with respect to the varying support count.

Experiment 3: The objective of this experiment is to examine scalability with respect to number of sites. Results are given in Fig. 9. For value results, for example, at site number of 17, for CRDM, SDDM, and FDM, ex-
Fig. 8 Results of Experiment 2. \( n_{\text{trans}} \)es are set to 10 K, 30 K, and 50 K, respectively. The number of sites is 5.

Fig. 9 Results of Experiment 3. Support thresholds are set to 5% and 10%, respectively. The \( n_{\text{trans}} \) is 10 K.

eocation times are 107.9, 161.2, 104.6, respectively for support threshold of 5%, and 28.0, 39.1, 26.8, respectively for support threshold of 10%. Again, it is shown that CRDM has higher execution times than FDM, and lower execution times than SDDM, as explained in Experiment 1. By examining Fig. 9 we can conclude that, given an adequate network as used in this experiment, CRDM reacts to the varying site number in a way similar to that FDM does.

Experiment 4: The objective of this experiment is to examine the cost of encrypt communication, and to compare CRDM with SFDM. For being not redundant, here the table is utilized, and all necessary results are summarized in Table 1. (1) By examining the results of CRDM with AES it can be found that, compared with the long execution time of data mining, the cost of encrypt communication is too low to dominate performance, especially when databases are large. This suggests that encrypt communication can be utilized to preserve privacy without worrying its cost. (2) By examining the results of SFDM with RSA (with 512-bit keys) it can be found that, the cost of SFDM’s privacy-preserving protocols is extremely high, and degrades performance seriously, resulting in that execution times of SFDM are far longer than those of CRDM. Note that the results for SFDM were taken with ignoring the cost of secure comparison at the end of its Phase 2; otherwise the difference would be even higher. For both (1) and (2), in the case of larger support values, e.g., 10%, the above observations are expected to still hold, but the difference between any two algorithms given in Table 1 would become smaller since fewer candidates are generated. The above observations have confirmed our analytical results in Section 5.1.

In summary, the added communication cost causes CRDM has slightly more execution times than FDM. Except for that, CRDM behaves link FDM. By the experimental results we are convinced that the low cost of CRDM in terms of the communication time requirement really make it be not so heavy, and CRDM is effective.

Table 1 Execution times (seconds) of FDM, CRDM, CRDM with AES, and SFDM with RSA. Support is 5%.

|                  | Site Num: 5 | Site Num: 17 | Site Num: 17 |
|------------------|------------|--------------|--------------|
|                  | DB: 10K    | DB: 50K      | DB: 10K      |
| FDM              | 98.35      | 250.39       | 104.63       |
| CRDM             | 100.43     | 254.64       | 107.90       |
| CRDM with AES    | 113.60     | 261.20       | 117.22       |
| SFDM with RSA    | 170.33     | 319.75       | 303.88       |

6. Discussions of Possible Extensions

In this section, we discuss some issues related to the possible extension to CRDM. The details, however, are beyond the scope of this paper.

One issue concerns how to improve CRDM’s scalability. As we have seen, for CRDM, its communication cost is proportional to the number of sites. For a very large system, the response time may become unacceptably long. Considering that in most cases we need not so high a degree of collusion resistance, next we give a solution to restrain the communication cost while still maintaining a user-specified level of security.

Procedure SecureSum is revised as follows (Fig. 10 shows an example). Let \( M \) be the number of sites, and assume that user has specified a minimum degree of collusion resistance, denoted by \( D (D \leq M - 2) \). (1) The sites are
divided into $N = \lfloor M/(D + 2) \rfloor$ groups. Group $G_i$ $(1 \leq i < N)$ holds $D + 2$ sites, and $G_0$ the remaining sites. (2) SecureSum is computed in each group in parallel. (3) Site $S_{i,0}$ $(1 \leq i < N)$ sends its result to site $S_{0,0}$. (4) Site $S_{0,0}$ obtains the final result.

Algorithm CRDM is revised as follows. In place of the site $S_0$ in Algorithm 1, site $S_{0,0}$ is used as the coordinator, and the revised SecureSum is used. For the revised CRDM, its degree of collusion resistance is $D$, and its communication cost in terms of the time requirement is estimated as $2(M - (N - 1)D - N + 1)T$.

Another issue concerns the global pruning to cut down the candidate set. As in SFDM and SDDM, we have not considered the global pruning of FDM. On the one hand, the factor of privacy-preserving prevents us from adopting the global pruning technology. On the other hand, in order to apply the global pruning technology a high data skew among sites is needed; otherwise its overhead would degrade performance of data mining algorithms. As a result, CRDM can function well in the homogeneous environment, i.e., the environment where sites have similar frequent itemset distributions. Extending CRDM so that it can also function well in the heterogeneous environment is a challenge.

7. Conclusion

We have proposed a high collusion-resistant data mining algorithm called CRDM. Using CRDM data mining can be done among a group of sites with privacy of every site being preserved. More concretely, for data mining using CRDM, no site would be able to learn contents of a transaction at any other site, what itemsets are supported by any other site, or the specific value of support for any itemset at any other site, unless (to the highest desired level of security) all sites but the victim collude.

Unlike other algorithms that use commutative encryption or random numbers, CRDM uses mainly the technique of sharing a secret among sites. Therefore, CRDM is expected to be very light. Considering the high computation cost of data mining, we think that this property would make CRDM be a practical algorithm, and applicable to data mining in large databases.

We have studied CRDM’s performance analytically and experimentally. Performance results demonstrated that CRDM is an effective algorithm.

For CRDM, in the case of a very large system, communication cost may become a problem. We have given an effective solution to this problem, which can restrain communication cost, but still maintain the desired level of security.

The approach to privacy-preserving data mining proposed in this paper is mainly for the Apriori-based algorithms. Another series of data mining algorithms is the pattern growth ones. To apply the proposed approach to the pattern growth algorithms is our future work.

References

1) Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules, Proc. 20th International Conference on Very Large Data Bases, pp.487–499, Santiago, Chile (1994).
2) Agrawal, R. and Srikant, R.: Privacy-preserving data mining, Proc. ACM SIGMOD Conference on Management of Data, pp.439–450, Dallas, TX (2000).
3) Agrawal, D. and Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms, Proc. 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.247–255, California, USA (2001).
4) Agrawal, S., Krishnan, V. and Haritsa, J.R.: On Addressing Efficiency Concerns in Privacy-Preserving Mining, Proc. 9th International Conference on Database Systems for Advanced Applications, pp.113–124, Jeju Island, Korea (2004).
5) Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, Comm. ACM, Vol.24, No.2, pp.84–88 (1981).
6) Cheung, D.W., Han, J., Ng, V.T., Fu, A.W. and Fu, Y.: A Fast Distributed Algorithm for Mining Association Rules, Proc. Interna-
7) Cheung, D.W., Ng, V.T., Fu, A.W. and Fu, Y.: Efficient Mining of Association Rules in Distributed Databases, *IEEE Transaction on Knowledge and Data Engineering*, Vol.8, No.6, pp.911–922 (1996).

8) Estivill-Castro, V. and Clifton, C.: Preface: Proceedings of the ICDM 2002 Workshop on Privacy, Security, and Data Mining, *Proc. IEEE ICDM Workshop on Privacy, Security and Data Mining*, pp.1–2, Maebashi City, Japan (2002).

9) Evfimievski, A., Srivastava, R., Agrawal, R. and Gehrke, J.: Privacy preserving mining of association rules, *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.217–228, Alberta, Canada (2002).

10) Fukazawa, T., Wang, J., Takata, T. and Miyazaki, M.: An Effective Distributed Privacy-Preserving Data Mining Algorithm, *Proc. 5th International Conference on Intelligent Data Engineering and Automated Learning*, pp.320–325, Exeter, UK (2004).

11) Goldreich, O.: Secure multi-party computation (working draft), Available from http://www.wisdom.weizmann.ac.il/~oded/ pp.html (Sept. 1998).

12) Goldschlag, D., Reed, M. and Syverson, P.: Onion Routing for Anonymous and Private Internet Connections, *Comm. ACM*, Vol.42, No.2, pp.39–41 (1999).

13) Han, J., Pei, J. and Yin, Y.: Mining Frequent Patterns without Candidate Generation, *Proc. ACM SIGMOD international conference on Management of data*, pp.1–12, Dallas, TX, USA (2000).

14) Han, J., Pei, J., Yin, Y. and Mao, R.: Mining frequent patterns without candidate generation: A frequent pattern tree approach, *International Journal of Data Mining and Knowledge Discovery*, Vol.8, No.1, pp.53–87, Kluwer Academic Publishers (2004).

15) Kantarcioglu, M. and Vaidya, J.: An Architecture for Privacy-Preserving Mining of Client Information, *Proc. IEEE ICDM Workshop on Privacy, Security and Data Mining*, pp.37–42, Maebashi City, Japan (2002).

16) Kantarcioglu, M. and Clifton, C.: Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data, *IEEE Transactions on Knowledge and Data Engineering*, Vol.16, No.9, pp.1026–1037 (2004).

17) Lindell, Y. and Pinkas, B.: Privacy Preserving Data Mining, *The Journal of Cryptology*, Vol.15, No.3, pp.177–206 (2002).

18) Okabe, N.: Facility Networking Technology, Ubiquitous Lab, Yokogawa Electric Corporation, Available from http://www.taca.jp/ (June 2004)

19) Oliveira, S.R.M. and Zaiane, O.R.: Privacy Preserving Frequent Itemset Mining, *Proc. IEEE ICDM Workshop on Privacy, Security and Data Mining*, pp.43–54, Maebashi City, Japan (2002).

20) Pfitzmann, A., Pfitzmann, B. and Waidner, M.: ISDN-MIXes: Untraceable Communication with Very Small Bandwidth Overhead, *Proc. GI/ITG Conference on Untraceable Communication in Distributed Systems*, pp.451–462, North-Holland, Amsterdam (1991).

21) Reiter, M.K. and Rubin, A.D.: Crowds: Anonymity for Web Transactions, *ACM Transactions on Information and System Security*, Vol.1, No.1, pp.66–92 (1998).

22) Rizvi, S.J. and Haritsa, J.R.: Maintaining Data Privacy in Association Rule Mining, *Proc. 28th International Conference on Very Large Data Bases*, pp.682–693, Hong Kong, China (2002).

23) Veloso, A.A., Meira, Jr., W., Parthasarathy, S. and Carvalho, M.B.: Efficient, Accurate and Privacy-Preserving Data Mining for Frequent Itemsets in Distributed Databases, *Proc. 18th Brazilian Symposium on Databases*, pp.281–292, Amazonas, Brazil (2003).

24) Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in Privacy Preserving Data Mining, *SIGMOD Record*, Vol.33, No.1, pp.50–57 (March 2004).

25) Zaki, M. J.: Parallel and Distributed Association Mining: A Survey, *IEEE Concurrency, special issue on Parallel Mechanisms for Data Mining*, Vol.7, No.4, pp.14–25 (1999).

(Received December 20, 2006) (Accepted April 10, 2007) (Released July 4, 2007) (Editor in Charge: Takeshi Fukuda) (Paper version of this article can be found in the IPSJ Transactions on Databases, Vol.48 No.SIG11(TOD34), pp.104–117.)
Shintaro Urabe received B.E. and M.E. degrees in software and information sciences from Iwate Prefectural University, Japan, in 2005 and 2007, respectively. His research interests include data mining, ubiquitous computing, and distributed/parallel computing. He is presently with the Hitachi East Japan Solutions, Japan.

Jiahong Wang received the B.E. degree from Northeast University, China, and the M.E. degree from Shenyang Institute of Computing Technology, the Chinese Academy of Sciences, China, both in computer science. He received the Dr. Eng. degree from University of Tsukuba, Japan. Now he is an associate professor at the Faculty of Software and Information Science, Iwate Prefectural University, Japan. His research interests include concurrency control, transaction processing, distributed/parallel processing, data mining, and modeling and performance evaluation. Dr. Wang is a member of IEEE, IEICE, and IPSJ.

Eiichiro Kodama received his master’s degree in mathematical science from the University of Tokyo, Japan, in 1994. From 1994 to 1996, he was at the Hitachi Information Technology Co., Ltd. (Hitachi IT). From 1998 to 2003, he was a Research Associate at Iwate Prefectural University. In 2003, he received Ph.D. degree in information science from Tohoku University, Japan. Currently he is an Associate Professor at Iwate Prefectural University. His main research interests are semantic web, natural language processing and algebraic number theory in mathematics. He is a member of IPSJ, IEICE, JASI, and the Association for Natural Language Processing.

Toyoo Takata received Ph.D. degree in information and computer sciences from Osaka University, Japan, in 1989. In 1989–1993, he was a Research Associate at the Department of Information and Computer Sciences, Osaka University. In 1993–1998, he was an Associate Professor of Graduate School of Information Science, Nara Institute of Science and Technology. Since 1998, he has been a Professor of Software Information Science, Iwate Prefectural University. Dr. Takata is a member of IEEE, ACM, IEICE, IPSJ and SITA.