COLLABORATIVE PRIVACY PRESERVING FREQUENT ITEM SET MINING IN VERTICALLY PARTITIONED DATABASES

Ehud Gudes and Boris Rozenberg
Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel

Abstract: Privacy concerns have become an important issue in Data Mining. This paper deal with the problem of association rule mining from distributed vertically partitioned data with the goal of preserving the confidentiality of each individual database. Each site holds some attributes of each transaction, and the sites wish to work together to find globally valid association rules without revealing individual transaction data. This problem occurs, for example, when the same users access several electronic shops purchasing different items in each. We present two algorithms for discovering frequent item sets and analyze their security, privacy and complexity properties.

Key words: item set, association rule, privacy

1. INTRODUCTION

Data mining technology has emerged as a means for identifying patterns and trends from large quantities of data. Mining encompasses various algorithms such as clustering, classification, and association rule mining. Traditionally, all these algorithms have been developed within a centralized model, with all data being gathered into a central site, and algorithms being run against that data. Privacy concerns can prevent this approach. There may not be a central site with authority to see all the data, or if the data is partitioned among several sites, sites may not want to disclose to each other their individual database, even if they are willing to disclose some information in order to enable the extraction of globally valid knowledge. Clearly if all participating sites are willing to share their data, or trust a third party to do the mining, the privacy problem is solved. We deal here with the problem that each site tries to protect the confidentiality of its database,

without compromising most of the knowledge discovery process. Informally, the problem is to mine association rules across two databases, where the columns in the table are at different sites, splitting each row. The problem was first discussed by Vaidya and Clifton in [3] (called here VDC). Their algorithm requires the intensive use of secure computation and may also have the potential of inferring some information of the individual databases, as will be discussed in Section 2.2, and this motivated our algorithms. Generally, the mining of association rules is divided into two phases: finding frequent item sets with support above a threshold, and finding rules with confidence above a threshold . In this paper we present two algorithms for solving the first problem. The problem of computing the rules can be solved similarly but is not addressed in this paper. The first algorithm is a two-party algorithm which requires a third party, but may also work instead with secure computation. The second is an asymmetric n-party algorithm, which does not require a third party or secure computation. In our protocols, one database is designated the master, and the initiator of the protocol, the other database is the slave. There is a join key present in both databases and the remaining attributes are present in one database or in the other, but not in both. The algorithms are based on two basic ideas. The first idea is the use of fake transactions. In spite of the existence of fake transactions it is possible to compute correctly the frequent item sets. The second idea is not to fix a precise measure of support but rather an approximate measure only, which can be made as precise as needed with a trade-off of the potential amount of inferred private information.

The rest of the paper is structured as follows. Section 2 presents the background and related work. In particular, Section 2.1 reviews the secure computation of a scalar product of two vectors (based on [6]), which we shall use in our algorithm for computing secure intersection of two sets, and Section 2.2 reviews the VDC algorithm [3] and discusses its potential problems. Section 3 presents the two algorithms, and some examples of their use. Section 4 discusses the privacy, security and complexity of the algorithms, and Section 5 is the conclusions and future work section.

2. BACKGROUND

Agrawal and Srikant [1] have proposed the first algorithm for discovering all significant association rules between items in a large database of transactions. However, this work did not address privacy concerns. Later the authors proposed a procedure in which some or all the numerical attributes are perturbed by a randomized value distortion, so that both the original

values and their distributions are changed. The proposed procedure then performs a reconstruction of the original distribution. This reconstruction does not reveal the original values of the data, and yet allows the learning of decision trees. Another paper shows a reconstruction method, which does not entail information loss with respect to the original distribution.

When the data is partitioned among several sites, there are algorithms that assume that all required data is available or can be sent to a central site. Then the existing data mining algorithms for centralized data mining model are executed. Another approach for distributed data mining runs the existing data mining algorithms for centralized data mining model at each site independently and combines the results [2]. But this will often fail to achieve a globally valid result. As pointed in [3], The issues that can cause a disparity between local and global results include:

a) Values for a single entity may be split across sources. Data mining at individual sites will be unable to detect cross-site correlations.
b) The same item may be duplicated at different sites, and will be over-weighted in the results.
c) Data at a single site is likely to be from a homogeneous population, hiding geographic or demographic distinctions between that population and others.

To overcome the above problems, algorithms were proposed for partitioned data between sites. The algorithms that were proposed for horizontally partitioned data (i.e. each site contains basically the same columns). Some of them use cryptographic techniques to minimize the amount of disclosed information or a special architecture. This architecture contains sites that sequentially add noise to the original data, compute the answer with noise and remove the noise from the answer. All these methods work with the assumption that no collusion occurs between the sites and the sites follow the protocol precisely.

The main work on vertically partitioned data is that by VDC [3], which we will discuss later in this section.

There has been much work addressing Secure Multiparty Computation. It was first investigated by Yao[4], and later, after Goldriech proved existence of a secure computation for any function[5] some algorithms based on his Circuit Evaluation Protocol have been proposed. But the general method, which is based on Boolean circuits, is inefficient for large inputs. Atallah and Du [6] proposed a more efficient technique for some cases of multi-party computation problem. One of them is the Two-Party Scalar Product Protocol. VDC uses a similar but more efficient protocol for secure product computation as the basis for their algorithm for privacy preserving association rules mining [3]. We will discuss this in more detail later.

This paper uses the general approach suggested by VDC, but proposes two new algorithms to deal with vertically partitioned data. It also does not provide a zero-knowledge solution, but it discloses less amount of information than [3], and may be more practical in some cases.

2.1 M.J.Atallah and W.Du Scalar Product Protocol

Inputs: Alice has a vector $X = (x_1, x_2, \ldots, x_n)$, and Bob has a vector $Y = (y_1, y_2, \ldots, y_n)$.
Outputs: Alice (but not Bob) gets $X \bullet Y + v$ where v is random scalar known to Bob only.

1 Alice and Bob agree on two numbers p and m, such that p^m is so large that conducting p^m additions is computationally infeasible.

2 Alice generates m random vectors, $V_1, \ldots, V_m$, such that $X = \sum_{j=1}^{m} V_j$.

3 Bob generates m random numbers $r_1, \ldots, r_m$, such that $v = \sum_{j=1}^{m} r_j$.

4 For each $j = 1, \ldots, m$ Alice and Bob conduct the following sub-steps:

4.1 Alice generates a secret random number $k, 1 \leq k \leq p$.

4.2 Alice sends $(H_1, \ldots, H_p)$ to Bob, where $H_k = V_j$, and the rest of H_i's are random vectors. Because k is a secret number known to Alice only, Bob does not know the position of V_j.

4.3 Bob computes $Z_{j,i} = H_i \bullet Y + r_j$ for $i = 1, \ldots, p$.

4.4 Using the 1-out-of-N Oblivious Transfer protocol [8,9], Alice gets $Z_j = Z_{j,k} = V_j \bullet Y + r_j$, while Bob learns nothing about k.

5 Alice computes $u = \sum_{j=1}^{m} Z_j = X \bullet Y + v$

As we'll see in Section 3, we could use the above algorithm to compute the size of the intersection of two sets, however a straightforward use by using binary vectors as membership vectors is not secure enough. This will be explained further in Section 3.1.

2.2 The VDC algorithm [3].

The algorithm uses the apriori-gen function to generate all candidate itemsets. For each found itemset, the algorithm calculates the support as follows:

L_1 = {large 1-itemsets}
for (k = 2; $L_{k-1} \neq \emptyset$; k ++) do begin
  C_k = apriori-gen(L_{k-1})
  for all candidates $c \in C_k$ do begin
    if all attributes in c are entirely at A or B
      that party independently calculates $c.count$
    else
      let A have l of the attributes and B have the
      remaining m attributes.
      construct X on A's side and Y on B's side
      where $X = \prod_{i=1}^{l} A_i$ and $Y = \prod_{i=1}^{m} B_i$
      compute $c.count = X * Y = \sum_{i=1}^{*} x_i \cdot y_i$
    end if
    $L_k = L_k \cup c \mid c.count \geq$ minsup
  end
end
Answer = $\cup_k L_k$

Suppose we want to compute if a particular 5-itemset is frequent with site A having 2 of the attributes and site B having 3 attributes. i.e. A and B want to know if the itemset $I = \{A_a, A_b, B_a, B_b, B_c\}$ is frequent. A creates a new vector X of cardinality n where $X = A_a * A_b$ (component multiplication) and B creates a new vector Y of cardinality n where $Y = B_a * B_b * B_c$. The Scalar product of X and Y provides the (in)frequency of the itemset . The scalar product is computed using a different method of secure computation than the one presented in [6]. As we'll see later we use a minor modification of the algorithm in [6], but we could have used the method of [3] as well.

The VDC algorithm has some problems. First the method requires communication of the two sites, for each item set computation. As we shall see, this is not required in our algorithm. The second problem is that this method may disclose information in certain cases, because in this algorithm each site learns the exact support of each item set. This is explained next.

Assume that site A learns the exact support for some itemset. With this knowledge, A learns the probability that an item in the set supported by A has a property in B, computed as the ratio of the actual support to A's support. For example, in the case when we have a support value of 5% for an itemset *ab,* and exactly 5% of the items on A have item *a,* A knows that at

least those same items on B have property b. This occurs always when the global support equals A's support, not just when A's support is at the threshold.

As we shall see, our algorithm discloses such information in less number of cases. Only when the global support is exactly equal to the threshold and to one of the parties' support value, it discloses the same amount of information as VDC. When the global support value is larger than the threshold, it discloses less information than VDC, and when the support value is less than the threshold, it discloses almost no information.

3. THE ALGORITHMS

We first present the definition of the problem, and the ideas, which are common to both algorithms and then, the two algorithms.

The association rule mining problem can be formally stated as follows [7]: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Associated with each transaction is a unique identifier, called its TID. We say that transaction T contains X, a set of some items in I, if $X \subseteq T$. An association rule is an implication of the form, $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c, if c% of the transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has support s in the transaction set D, if s% of the transactions in D contain $X \cup Y$. Item set X is called large if its support is greater then or equal to the minimal support threshold. Our goal is to find all large itemsets (as noted in the introduction, the problem of finding the rules can be solved similarly, but is outside of the scope of this paper.).

In a vertically partitioned database we have the following situation. There are two databases, DB1 and DB2. In both databases, the domain of the TIDs is the same. We are only interested in large item sets, which contain attributes from both databases. In both databases there are TIDs which are not present, however for those TIDs which are not null, TIDs (DB1) $\cap$ TIDs(DB2) is close to size(DBi), which enables a real-life mining of globally frequent item set. The problem is to find large item sets without each site revealing its own database.

Both algorithms follow the basic collaborative Apriori algorithm. However, before applying the algorithm there is a pre-processing stage that populates the individual databases with fake transactions. After this action, each party can send the resulted database to the opposite party to calculate the frequent itemsets in the resulted database. In the first algorithm the two parties are symmetric. In the first phase one is designated as a master, which initiates the protocol and will get the results, and the other as a slave, which

only contributes its information but will not do any global computations. The master is looking for all large itemsets according to its own real transactions, and check whether the found itemsets are present in the *slave* real transactions. This is done with the help of a third untrusted party (this party is not trusted with the database, but it is trusted with computations). The parties change roles at the end of the phase.

In the second algorithm, there are n slaves and one master. The master computes the globally valid item sets and tells the results to the slaves.

We assume without loss of generality that the number of transactions in each database is equal to N – some number that depends on area of business, and the TIDs range from 1 to N. When fake transactions are introduced, they use "unoccupied" TIDs. When information between the parties is shared, then only information in which some attributes are real (in one of the databases) is of use. That is, a fake transaction whose corresponding TID in the other database is empty, is not considered at all. Its important to note that when each site computes large itemsets it doesn't know whether the attributes corresponding to his real transaction, are real or not!

3.1 Algorithm 1

Preparing phase (executed by two parties):

1 Construct database with fake transactions (Simply put randomly 1 in non existing transactions).
2 Send to the third Party all your real TID.
3 Receive from third Party whether mining is possible (See the Third Party execution phase). If it is then continue.
4 Send the database from one party to the opposite party and receive its database.
5 Build the global database (TIDs from step 1, and attributes from the two databases).
6 Build local true database (LTDB - contains the local real transactions only, but with possibly fake attributes from the other party).

Master's Execution phase:

1 find LL_1 – all large 1-itemsets in the local database from 6.
2 for each $l \in LL_1$:
2.1 send LID = {TID | l present in transaction with id = TID} to third Party
2.2 receive from third Party Ok or Not
3 end
4 $GL_1 = \{l \in LL_1 \mid$ third Party said Ok$\}$
5 for ($k = 2$; $GL_{k-1} \neq \phi$; k++)

5.1 C_k = apriori-gen(GL_{k-1})
5.2 for all transaction $t \in$ LTDB
5.2.1 C_t = subset(C_k, t)
5.2.2 for all candidates $c \in C_t$
5.2.2.1 $c.count$ ++
5.2.3 end
5.3 end
6 $LL_k = \{c \in C_k \mid c.count \geq \text{minsup}\}$
7 for each $l \in LL_k$
7.1 send LID = {TID | l presents in
transaction with id = TID} to third Party
7.2 receive from third Party Ok or Not
8 end
9 $GL_k = \{ l \in LL_k \mid$ third Party said Ok$\}$
10 end
11 send to third Party "Master Finished"
12 Answer = $\cup_k GL_k$
13 reverse roles and execute algorithm again

Third Party execution phase:
1 receive from the two parties their real TID (TID1,TID2)
2 send to the two parties ($|TID1 \cap TID2| \geq$ minsup)
3 if ($|TID1 \cap TID2| \geq$ minsup)
3.1 while (Master not finished) do
3.1.1 receive set of TID's from Master
(MTID, $|MTID| \geq$ minsup)
3.1.2 if [$|MTID \cap TID2| \geq$ minsup] then
3.1.2.1 send "Ok" to Master
3.1.3 else
3.1.3.1 send "NotOK" to Master
3.1.4 end if
3.2 end while
4 else
4.1 send "NotOK" /* mining is not possible!
not enough overlap! */
5 end if
6 end

Slave execution phase:
1 Execute preparing phase.
2 Wait for master to finish.

3.1.1 Secure computation

Instead of a third party, one may use secure computation. The main problem is overcoming the "0,1" problem. Let us discuss it in more detail.

To compute the intersection size of two sets of TIDs (from a common domain), one can use binary vectors where a 1 indicates membership and a zero indicates non-existence. The scalar product of these two vectors is the size of intersection. For example in Algorithm1, the master creates a vector $X = (x_1, x_2, \ldots, x_n)$ such that $x_i = 1$ if transaction with ID = i is a real transaction of the Master and contains the checking item set l, and $x_i = 0$, if transaction number i is a fake transaction or does not contains an itemset l, and the Slave generates a vector $Y = (y_1, y_2, \ldots, y_n)$ similarly. However, if one uses Attallah and Du's protocol [6] directly, there may be a compromise of information. This can be demonstrated as follows:

Suppose one party knows some data values, the equations can be solved to reveal all values. For example, if X is represented as $V_1 = (0,0,1)$, $V_2 = (0,1,0)$, $V_3 = (0,1,1)$, $Y = (1,1,1)$ and $r_1 = 4$, and the first party gets:

$y_3 + r_1 = 7$

$y_2 + r_1 = 5$

$y_2 + y_3 + r_1 = 7$

Since y_1 was eliminated by the "zeroes", the first party now has 3 equations with 3 unknowns. Which it can solve to disclose both the y's and the r.

To reduce this possibility in our protocol, we can restrict the Master not use the value "0" in the representation of vector X, and for the Slave not to reuse the r-values sent once.

The following method explain, how we can divide the vector X into m random vectors $V_1, \ldots, V_m$:

The Master generates m random vectors from Z_p^n such that $X = \sum_{j=1}^{m} V_j \bmod p$

Now, the Master and the Slave can perform a Scalar Product Protocol as is, to find the size of the intersection of vectors X and Y. For example, assume the Master creates vector $X = (x_1, x_2, \ldots, x_{10}) = (1,1,0,0,0,1,1,0,1,0)$ and the Slave generates the vector $Y = (y_1, y_2, \ldots, y_{10}) = (1,1,0,1,0,1,1,0,0,1)$. Suppose the Master and the Slave agree on the number $m = 3$ and p =11 in the protocol. Now, the Master generates following random vectors:

$V_2 = (2,3,4,5,5,7,8,9,1,10)$
$V_3 = (9,7,4,2,1,10,8,5,2,2)$

Now, the Master and the Slave can perform a Scalar Product Protocol as is, to find the size of the intersection of vectors X and Y without having the danger of eliminating Ys because of the "0"s. Note, that only the Slave knows the size of the intersection, and it can send to Master "Ok" if this size $\geq$ minimal support.

3.2 Algorithm 2

In this algorithm we do not use a third party and instead let each slave compute the intersection itself. However, since the slaves have no knowledge of the other databases, their inference from this computation is very limited.

We assume that we have one Master and n Slaves. The Master starts the computation with the first Slave and waits for the last Slave for a positive or a negative result.

Execution phase for each Slave:

1 receive a set of IDs (or "NOT OK") from your left neighbour.
2 if received "NOT OK"
2.1 send "NOT OK" to your right neighbour.
2.2 go to 1
3 else
3.1 intersects the set with your own real IDs and test.
4 if result $\geq$ minimal support
4.1 send resulted SetOfIds to your right neighbour
5 else
5.1 send "NOT OK" to your right neighbor.
5.2 go to 1
6 end if

Now we can describe the algorithm, which is quite similar to Algorithm 1.

Master's preparing phase:

1 Send to your right neighbor all real TIDs. Receive from your left neighbor whether mining is possible. If it is, then continue.
2 Build the global database (TID = TID1 $\cup$ TID2,... $\cup$ TIDn and the attributes from the n databases (i.e. each slave sends the database with fake transactions to the master).
3 Build local true database (LTDB - contains the local real transactions only).

Master's execution phase:

1 find LL_1 – all large 1-itemsets in the local database from 4.
2 for each $l \in LL_1$:
2.1 LID = {TID | l present in transaction with id = TID}
2.2 send LID to the right neighbor
2.3 receive from left neighbor Ok or Not
3 end
4 $GL_1 = \{l \in LL_1 \mid$ Slave said Ok$\}$
5 for ($k = 2$; $GL_{k-1} \neq \phi$; k++)
5.1 C_k = apriori-gen(GL_{k-1})
5.2 for all transaction $t \in$ LTDB
5.2.1 C_t = subset(C_k, t)
5.2.2 for all candidates $c \in C_t$
5.2.2.1 $c.count$++
5.2.3 end
5.3 end
5.4 $LL_k = \{c \in C_k \mid c.count \geq$ minsup$\}$
5.5 for each $l \in LL_k$
5.5.1 LID = {TID | l presents in transaction with id = TID}
5.5.2 send LID to the right neighbor
5.5.3 receive from left neighbor Ok or Not
5.6 end
5.7 $GL_k = \{l \in LL_k \mid$ Slave said Ok$\}$
6 end
7 Answer = $\cup_k GL_k$
8 send Answer to all Slaves

Note that in this case we may also use secure computation, but if the number of databases is greater than 2, this is not really necessary, since each slave may not be able to infer much anyway. This is because when n>2, each slave not only does not know the exact size of the final intersection, but also does not know which exact item set the master is currently testing!

4. ANALYSIS

4.1 Correctness

The two algorithms use the apriori-gen function to generate all candidates to be large itemset. On each iteration k, a large k-item sets that was found by the Master according to its true transactions only, was checked by another site. So, all existed large itemsets where found.

4.2 Security analysis

The security of the approach depends on the algorithm and the method. In algorithm 1 and a Third party, it depends on the integrity of the third party. In algorithm 2, on the integrity of each of the slaves.

If we use secure computation, then the security of our approach is based on the security of the scalar product protocol, which is based on the inability of the slave side to solve $p * m$ equations with $n + m$ unknowns, which requires that $n > m$ (p -1). Since the master does not get the result of the scalar product, just an "OK" answer, it cannot use the linear equations methods anyway.

Computing support for each candidate item set requires one run of the component scalar product protocol. Again, the slave is secured by the same bound, since at each run the master sends it a different intersection set (with different random components).

4.3 Disclosed information

In both algorithms, neither the Master nor the Slave learns the exact support for an item set. They only know, whether some item set is frequent or not. This decreases the probability that the Master or Slaves will learn that a set of items on another site has a given property and it occurs only when the global support value is above or equal to the threshold value, and is also equal to the Master's or Slave's support. For example, if a is an item from the Master's database, and b is an item from the Slave's database, the threshold support is 5%, and the global support is greater or equal 5%, and item set *{a,b}* is present in exactly 5% of the real Master's transactions, then the Master knows that the same transactions are real transactions of the Slave.

In the case when the item set *{a,b}* is present in more then 5% (support threshold) of transactions in the Master's database, say 7%, and *{a,b}* is still a global large item set, the Master knows, that at least 5/7 transactions of the Slave are real transactions, but he is not sure even which are real transactions and which are fake. When the support of a tested item set is less than the support threshold, very little information is disclosed to the master and almost none to the slave.

Note that the master in either Algorithm1 or Algorithm2 never knows the support value, whether we use a third party or secure computation. When we use secure computation, or in Algorithm2 with only 2 parties, the slave does know the exact value of support, but it cannot do anything with it since it

doesn't know which item set the master is currently testing! So knowing the size of the product does not give him much information!

As was explained in Section 2.2, in VDC, in contrast, the Master learns the probability that an item in the set supported by Master has a property in the Slave's database, computed as the ratio of the global 1 support to the Master's support, whether the *item set is frequent or not*!

One important point that we want to stress here is that if the participants don't know the number of fake transactions, then this number has no effect on the amount of disclosed information. It can only affect the performance (redundant computations) of the algorithm!

One problem which can occur in our algorithms, although unlikely, is that some specific value will be disclosed when the database has a large number of overlapping entities. For example, suppose the Master's real IDs are $I_m = \{1,2,3,4,5,6\}$, and the Slave's real IDs are $I_s = \{1,2,3,4\}$ and minsupp = 4. Assume the Master asks the Slave whether $|I_1 \cap I_s| \geq$ minsupp and gets answer "YES" (I_1 - subset of I_m and $I_1 = \{1,2,3,4,5,6\}$). Next question is whether $|I_2 \cap I_s| \geq$ minsupp and he gets the answer "NO" (I_2 - subset of I_m and $I_2 = \{1,2,3,5,6\}$). The Master can come to the conclusion that transaction with ID = 4 is a real transaction of the Slave, and at least 2 transactions from {1,2,3,5,6} are fake. Allowing approximate answers as follows can reduce this possibility.

4.4 The Support approximation method

Let us assume that the Master and the Slave agree on *ac* – approximation coefficient. In both algorithms, whoever compute the intersection size, the third party or the slave, instead of checking whether it is greater or equal than the minimal support, it does the following. It generates a random number c, such that $(\text{minsup} - \text{ac}) \leq c \leq \text{minsup}$, and sends "Okey" to the Master if the size of above intersection $\geq$ c, and "No" in the other case.

Now, for every tested item set, this approximation will use a new value of c. Thus, even for overlapping item sets it will be quite difficult to discover facts such as demonstrated above. This property decreases the exactness of the solution, since it only guarantees that the support will be above minimal-support – ac, but it discloses less information to the opposite side, and decreases the ability of any side to guess the real transactions of the opponent.

Using the above example, the Master cannot distinguish between the following two cases:

C = 3 during the first intersection calculation and C = 4 during the second intersection calculation when {1,2,3} are real transactions of the Slave.

C = 2 during the first intersection calculation and C = 3 during the second intersection calculation when {1,2} are real transactions of the Slave.

On the other hand, if the Master asks the Slave the same question a large number of times, it increases its chance of guessing the correct size of the intersection. This since if ac is not large, the average result will give out some information. To reduce this possibility, we can restrict the Master to check each set only once.

5. CONCLUSIONS AND FUTURE WORK

We presented two algorithms for discovering all large item sets in vertically distributed databases, without the sources revealing their individual transaction values. We compared these algorithms to the previously known algorithm by Vaidya and Clifton [3] and show that it is possible to achieve good security and privacy with complexity cost small enough.

In the future, we will develop precise algorithms for producing rules and not only frequent item sets. We also plan to further investigate the privacy/performance tradeoffs both analytically and experimentally and test these tradeoffs with different support values.

REFERENCES

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, Sept. 12-15 1994.
2. A.Prodomidis, P.Chan, S.Stofo. Meta-learning in distributed data mining systems: Issues and approaches, chapter 3. AAAI/MIT Press, 2000.
3. J.Vaidya, C.Clifton. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In Proceedings of SIGKDD 2002, Edmonton, Alberta, Canada.
4. A.C.Yao. How to generate and exchange secrets. In Proceedings of the 27th IEEE Symposium on Foundations of Computer Science,. 1986, pages 162-167.
5. O.Goldreich, S.Micali, A.Wigderson. How to play any mental game – a completeness theorem for protocols with honest majority. In Proceedings, 19th ACM Symposium on the Theory of Computing, pages 218-229, 1987.
6. W.Du and M.J.Atallah,. Secure multi-party computational geometry. In proceedings of the 7th International Workshop on Algorithms and Data Structures, Providance, Rhode Island, 2001.
7. R.Agrawal, T.Imielinski, A.M.Swami. Mining association rules between sets of items in large databases., In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington, 1993.
8. G.Brassard, C.Crepeau, J.Robert. All-or-nothing disclosure of secrets. In Advances in Cryptology-Crypto86, Springer Lecture Notes in Computer Science, volume 263, 1986.
9. I.Even, O.Goldreich, A.Lempel. A randomized protocol for signing contracts. Communications of the ACM, Vol 28:637-647, 1985.