Publication avant LAFMC

Titre	A Secure Protocol to Maintain Data Privacy in Data Mining
Auteurs	Fodé Camara, **Samba Ndiaye** and Yahya Slimani
Référence	International Conference on Advanced Data Mining and Applications (ADMA)
Editeur	Springer
Pages	417–426
Année	2009
URL	https://link.springer.com/chapter/10.1007/978-3-642-03348-3_40
DOI	https://doi.org/10.1007/978-3-642-03348-3_40
Index	https://www.scopus.com/authid/detail.uri?authorId=6701604512
ISBN	978-3-642-03348-3
Encadreur	Oui
Extrait d'une thèse	Oui

International Conference on Advanced Data Mining and Applications
ADMA 2009: Advanced Data Mining and Applications pp 417-426 | Cite as

A Secure Protocol to Maintain Data Privacy in Data Mining

Authors | Authors and affiliations

Fodé Camara, Samba Ndiaye, Yahya Slimani

Part of the Lecture Notes in Computer Science book series (LNCS, volume 5678)

Abstract

Recently, privacy issues have becomes important in data mining, especially when data is horizontally or vertically partitioned. For the vertically partitioned case, many data mining problems can be reduced to securely computing the scalar product. Among these problems, we can mention association rule mining over vertically partitioned data. Efficiency of a secure scalar product can be measured by the overhead of communication needed to ensure this security. Several solutions have been proposed for privacy preserving association rule mining in vertically partitioned data. But the main drawback of these solutions is the excessive overhead communication needed for ensuring data privacy. In this paper we propose a new secure scalar product with the aim to reduce the overhead communication.

Keywords

Association Rule | Communication Overhead | Secure Protocol | Privacy Preserve | Homomorphic Encryption

Copyright information

About this paper

Cite this paper as:
Camara F., Ndiaye S., Slimani Y. (2009) A Secure Protocol to Maintain Data Privacy in Data Mining. In: Huang R., Yang Q., Pei J., Gama J., Meng X., Li X. (eds) Advanced Data Mining and Applications. ADMA 2009. Lecture Notes in Computer Science, vol 5678. Springer, Berlin, Heidelberg

DOI
https://doi.org/10.1007/978-3-642-03348-3_40

Publisher Name
Springer, Berlin, Heidelberg

Print ISBN
978-3-642-03347-6

Online ISBN
978-3-642-03348-3

eBook Packages
Computer Science

A Secure Protocol to Maintain Data Privacy in Data Mining

Fodé Camara[1], Samba Ndiaye[1] and Yahya Slimani[2]

[1] Cheikh Anta Diop University, Department of Mathematics and Computer Science, Dakar, Senegal
fode.camara@ucad.edu.sn, ndiayesa@ucad.sn
[2] Department of Computer Science, Faculty of Sciences of Tunis, 1060 Tunis, Tunisia
yahya.slimani@fst.rnu.tn

Abstract. Recently, privacy issues have becomes important in data mining, especially when data is horizontally or vertically partitioned. For the vertically partitioned case, many data mining problems can be reduced to securely computing the scalar product. Among these problems, we can mention association rule mining over vertically partitioned data. Efficiency of a secure scalar product can be measured by the overhead of communication needed to ensure this security. Several solutions have been proposed for privacy preserving association rule mining in vertically partitioned data. But the main drawback of these solutions is the excessive overhead communication needed for ensuring data privacy. In this paper we propose a new secure scalar product with the aim to reduce the overhead communication.

1 Introduction

Motivated by the multiple requirements of data sharing, privacy preserving and knowledge discovery, Privacy Preserving Data Mining (PPDM) has been studied extensively in data mining community. In many cases, multiple parties may wish to share aggregate private data, without leaking any sensitive information at their end. For example, different superstores with sensitive sales data may wish to coordinate among themselves in knowing aggregate trends without leaking the trends of their individual stores. This requires secure and cryptographic protocols for sharing the information across the different parties. The data may be distributed in two ways across different sites: horizontal partitioning where the different sites may have different sets of records containing the same attributes; and vertical partitioning where the different sites may have different attributes of the same sets of records. For the vertically partitioned case, many primitive operations such as computing the scalar product can be useful in computing the results of data mining algorithms. For example, [1] uses the secure scalar product over the vertical bit representation of itemset inclusion in transactions, in order to compute the frequency of the corresponding itemsets. This key step is applied repeatedly within the framework of a roll up procedure of itemset counting. The efficiency of secure scalar products is important because they can

R. Huang et al. (Eds.): ADMA 2009, LNAI 5678, pp. 417–426, 2009.

generate excessive overhead communication. Several works have been carried out on privacy preserving association rule mining for vertically partitioned data. Most of them suffer from communication overhead. We challenge this drawback by proposing a new secure scalar product protocol that reduce drastically the communication cost between sites. This feature distinguishes our proposal from the existing ones.

The remainder of this paper is organized as follows. Section 2 provides the state-of-the-art in privacy preserving data mining. In Section 3, we present our proposed approach and we give examples to illustrate the practicability of our protocol. Evaluation of computation and communication costs of our protocol is presented in Section 4. Section 5 discusses the security aspect of our proposal. Section 6 evaluates this work by comparing it with related topics in the privacy preserving association rule mining community. Finally, Section 7 concludes with a discussion of the contributions of our proposal and our current research plans.

2 Privacy and Data Mining

The problem of privacy preserving data mining has become more important in recent years because of the multiple requirements of data sharing, privacy preserving and knowledge discovery. Two main problems are addressed in privacy preserving data mining: the first one is the protection of sensitive raw data; the second one is the protection of sensitive knowledge contained in the data, which is called knowledge hiding in database. We can classify the techniques of knowledge hiding in database in two groups: approaches based on data modification and the approaches based on data reconstruction. The basic idea of data reconstruction is to modify directly the original database D, to obtain a new database D'. According to the way of modifying the original database we can still push classification by distinguishing two families of techniques: techniques based on the distortion and the techniques based on the blocking of the data. The distortion changes a value of attribute by a new value [2] (i.e. change of value 1 in 0), while blocking [3], is the replacement of an existing value of attribute by a special value noted by "?". However approaches based on data modification cannot control the side effects. For example in association rule, hiding a non sensitive rule R_i witch had confidence bigger than threshold $minconf$ (i.e. $conf(R_i) > minconf$), can give a new value such that $conf(R_i) < minconf$. They also make many operations of I/O especially when the original database is too large. Another solution concerns the approaches based on data reconstruction [4]. The basic idea of these approaches is to extract first the knowledge K from the original database D. The new database D' is then reconstructed from K. The basic idea of data reconstruction is inspired by the recent problem of *Inverse Frequent Set Mining*. Opposite approaches use techniques based on data reconstruction that control directly the side effects. The main proposal to solve the problem of the protection of sensitive raw data is the secure multiparty computation. A Secure Multi-party Computation (SMC) problem deals with computing any function on any input, in a distributed network where each

participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participants input and output. Secure two party computation was first investigated by Yao [5,6] and was later generalized to multi-party computation [7,8]. For example, in a 2-party setting, Alice and Bob may have two inputs x and y, and may wish to both compute the function $f(x, y)$ without revealing x or y to each other. This problem can also be generalized across k parties by designing the k argument function $h(x_1, ..., x_k)$. Several data mining algorithms may be viewed in the context of repetitive computations of many such primitive functions like the *scalar product*, *secure sum*, and so on. In order to compute the function $f(x, y)$ or $h(x_1, ..., x_k)$ a protocol will have to designed for exchanging information in such a way that the function is computed without compromising privacy. The problem of distributed privacy preserving data mining overlaps closely with a field in cryptography for determining secure multi-party computations. A broad overview of the intersection between the fields of cryptography and privacy-preserving data mining may be found in [9]. Clifton et al. [10] give a survey of multi-party computation methods.

3 Proposed Approach

3.1 Problem Definition

Scalar product is a powerful component technique. Several data mining problems can essentially be reduced to securely computing the scalar product. To give an idea of how a secure scalar protocol can be used, let us look at association rule mining over vertically distributed data. The association rule mining problem can be formally stated as follows [11]: Let $I = \{i_1, ..., i_n\}$ be a set of items. Let D be a set of transactions, where each transaction T has an unique identifier *TID* and contains a set of items, such that $T \subseteq I$. We say that a transaction T contains X, a set of items in I, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in a database D with confidence c, if $c\%$ of transactions in D that contain X tend to also contain Y. The association rule $X \Rightarrow Y$ has support s in D, if $s\%$ of transactions in D contain $X \cup Y$. Within this framework, we consider mining boolean association rules. The absence or presence of an attribute is represented by a value taking from $\{0, 1\}$. Transactions are represented under the form of strings of 0 and 1, while the database can be represented as a matrix of $\{0, 1\}$. The association rule mining algorithm over vertically distributed databases is based on the classic Apriori algorithm of Agrawal and Srikant [12]. The key issue of this algorithm is the problem of finding the frequent itemsets in a database. To determine if a given itemset is frequent or not, we count the number of transactions, in D, where the values for all the attributes in this itemset are 1. This problem can be transformed into a simple mathematical problem, using the following definitions: Let $l + m$ be the total number of attributes, where Alice has l attributes, $\{A_1, ..., A_l\}$, and Bob has the remaining m attributes, $\{B_1, ...B_m\}$. Transactions are a sequence of $l + m$ $1's$ or $0's$. Let *minsupp* be the

minimal support, and n the total number of transaction in database D. Let $\overrightarrow{X}$ and $\overrightarrow{Y}$ be the columns in D, i.e., $x_i = 1$ if row i has value 1 for item or attribute X. The scalar product of two vectors $\overrightarrow{X}$ and $\overrightarrow{Y}$ of cardinality n is defined as follows: $\overrightarrow{X} \bullet \overrightarrow{Y} = \sum_{i=1}^{n} x_i \times y_i$. Determining if the 2-itemset $< XY >$ is frequent can be reduced to test if $\overrightarrow{X} \bullet \overrightarrow{Y} \geq minsupp$. The generalization of this process to a w-itemset is straightforward. Assume Alice has p attributes $a_1, ..., a_p$ and Bob has q attributes $b_1, ..., b_q$. We want to compute the frequency of the w-itemset $< a_1...a_p, b_1...b_q >$, where $w = p + q$. Each item in $\overrightarrow{X}$ (respectively in $\overrightarrow{Y}$) is composed of the product of the corresponding individual elements, i.e., $x_i = \prod_{j=1}^{p} a_j$ (respectively $y_i = \prod_{j=1}^{q} b_j$).

Now, we can formalize our problem as follows: Assume that 2 parties, for example Alice and Bob, such that each has a binary vector of cardinality n, e.g. $\overrightarrow{X} = (x_1, ..., x_n)$ and $\overrightarrow{Y} = (y_1, ..., y_n)$. The problem is to securely compute the scalar product of these two vectors, e.g. $\overrightarrow{X} \bullet \overrightarrow{Y} = \sum_{j=1}^{n} x_i \times y_i$.

3.2 Security Tools

To define our secure scalar product protocol, we have to use a semantically secure additive homomorphic public-key cryptosystem. Indeed to ensure security in data transmission, we chose a public-key cryptosystem. The security of a public-key cryptosystem is determined by a security parameter k. For a fixed k, it should take more than polynomial in k operations to break the cryptosystem [13] (Section 2). The public-key cryptosystem that have chosen is homomorphic. This choice is justified by the fact that, if given $Enc(x)$ and $Enc(y)$, one can obtain $Enc(x \perp y)$ without decrypting x, y for some operation $\perp$. Furthermore, the homomorphic public-key cryptosystem is additive; that means that a party can add encrypted plaintexts by doing simple computations with ciphertexts, without having the secret key. Informally, this means that for probabilistic polynomial-time adversary, the analysis of a set of ciphertexts does not give more information about the cleartexts than what would be available without knowledge of the ciphertexts. This property is very important in our binary context because an attacker could always encrypt 0 and 1 by using the public key, and then compare the resulting ciphertexts with the received message to decide the value of a bit. One of the most efficient currently known semantically secure homomorphic cryptosystems is Paillier cryptosystem [14]. This cryptosystem has all the four properties described above. In Paillier's case the plaintext space is defined by $P(sk) = Z_N$, with $N \geq 2^{1024}$, e.g. N is a hard-to-factor modulus.

3.3 Algorithm

Before to give our proposed algorithm, we will suppose the following. First, Alice generates a pair of key and a random value r and it computes $Enc_{pk}(x_i, r)$ which she sends to Bob. This message is computationally indistinguishable from the received message since the semantic security of the encryption system guarantees that no extra information is revealed. She sends also the public key to Bob. Bob

Algorithm 1. Private Scalar Product Protocol

Require: N=2 (number of sites; Alice and Bob), Alice vector: $\overrightarrow{X} = (x_1, ..., x_n)$, Bob vector: $\overrightarrow{Y} = (y_1, ..., y_n)$

1: **for** Alice **do**
2: Generates a pair of key (sk, pk);
3: Generates $(Enc_{pk}(x_1), ..., Enc_{pk}(x_n))$ using the public key pk;
4: Sends $(Enc_{pk}(x_1), ..., Enc_{pk}(x_n))$ to Bob;
5: **end for**
6: **for** Bob **do**
7: Computing $\prod_{i=1}^{n} p_i$, where $p_i = Enc_{pk}(x_i)$ if $y_i = 1$ and $p_i = 1$ if $y_i = 0$ then uses the additive property of homomorphic encryption for computing $\prod_{i=1}^{n} p_i = Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$;
8: Sends $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice;
9: **end for**
10: **for** Alice **do**
11: Computes $Dec_{sk}(Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y}))$;
12: Sends the final result to Bob;
13: **end for**

uses the additive property of homomorphic encryption and computes $\prod_{i=1}^{n} p_i$, with $p_i = Enc_{pk}(x_i)$ if $y_i = 1$ otherwise $p_i = 1$. A public-key cryptosystem is additive homomorphic when $Enc_{pk}(x_1, r_1) \times Enc_{pk}(x_2, r_2) \times ... \times Enc_{pk}(x_n, r_n) = Enc_{pk}(x_1 + x_2 + ... + x_n, r_1 \times r_2 \times ... \times r_n)$, where $+$ is a group operation and $\times$ is a groupoid operation. For the sake of simplicity of notations, we will not explicitly include, in the rest of the paper, the randomness as an input of the encryption functions. Then, in step 5, Bob sends $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice. Having the secret key, Alice deciphers the final result what she sends to Bob. Now we describe our proposed algorithm.

To illustrate the behavior of our algorithm, we consider the following first scenario: we suppose that Alice has the vector $\overrightarrow{X} = (1, 0, 0, 1)^T$ and Bob has $\overrightarrow{Y} = (1, 0, 0, 1)^T$. We want to compute securely the scalar product of $\overrightarrow{X} \bullet \overrightarrow{Y}$. We obtain the following:

1. Alice view's: Alice generates a pair of key (sk, pk); to do that it uses the public key pk and generates $(Enc_{pk}(x_1), Enc_{pk}(x_2), Enc_{pk}(x_3), Enc_{pk}(x_4))$, what she sends to Bob.
2. Bob view's: Bob computes $\prod_{i=1}^{n} p_i$, where $p_1 = Enc_{pk}(x_1)$ (because $y_1 = 1$), $p_2 = 1$, $p_3 = 1$ and $p_4 = Enc_{pk}(x_4)$. He uses the additive property of homomorphic encryption to compute $Enc_{pk}(x_1) \times Enc_{pk}(x_4) = Enc_{pk}(x_1 + x_4) = Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$. Finally Bob sends $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice.
3. Alice view's: Using the secret key, Alice computes $Dec_{sk}(Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y}))$ and sends the result to Bob.

As second scenario, we suppose that Alice has the vector $\overrightarrow{X} = (1, 1, 1, 1)^T$ and Bob has $\overrightarrow{Y} = (1, 1, 0, 1)^T$. As for the first scenario, We want to compute securely the scalar product of $\overrightarrow{X} \bullet \overrightarrow{Y}$. This computation gives:

1. Alice view's: Using the public key pk, Alice generates a pair of key (sk, pk); then it generates $(Enc_{pk}(x_1), Enc_{pk}(x_2), Enc_{pk}(x_3), Enc_{pk}(x_4))$, what she sends to Bob.
2. Bob view's: Bob computes $\prod_{i=1}^{n} p_i$, where $p_1 = Enc_{pk}(x_1)$ (because $y_1 = 1$), $p_2 = Enc_{pk}(x_2)$, $p_3 = 1$ and $p_4 = Enc_{pk}(x_4)$. Using the additive property of homomorphic encryption, it computes $Enc_{pk}(x_1) \times Enc_{pk}(x_2) \times Enc_{pk}(x_4) = Enc_{pk}(x_1 + x_2 + x_4) = Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$. Finally, Bob sends $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice.
3. Alice view's: Using the secret key, Alice computes $Dec_{sk}(Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y}))$ and sends the result to Bob.

In this second scenario, although Alice has a constant vector, Bob cannot deduce in no manner this one. Semantic security guarantees that it is not possible for Bob to distinguish between the encryption of a 0 or a 1 value, when r is randomly chosen.

4 Computation and Communication Evaluation

From the communication view point, our protocol needs the following messages: (i) for each entry of the vector, our protocol requires one message; (ii) one message to send the public key; (iii) Bob needs to send $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice; (iv) finally, Alice must send the result of the scalar product to Bob. Hence, the number of messages is $n+3$, where n is the dimension of the vector. In this case we obtain a total communication cost $O(n)$. From the computation view point, Alice performs, in our protocol, n encryptions and 1 decryption. Bob performs less $n-1$ additions. Therefore the computational complexity of our protocol is linear, e.g. $O(n)$.

5 Security Analysis

In this section, we will verify the security of our protocol. This security depends on the semantic secure public-key cryptosystem used. A public-key cryptosystem is semantically secure (IND-CPA secure) when a probabilistic polynomial-time adversary cannot distinguish between random encryptions of two elements, chosen by herself. Paillier [14] recalls the standard notion of security for public key encryption schemes in terms of indistinguishability, or semantic security as follows: We consider chosen-plaintext attacks (CPA), because homomorphic schemes can never achieve security against chosen-ciphertext attacks. To define security, we use the following game that an attacker A plays against a challenger:

$$\begin{aligned} (pk, sk) &\leftarrow KG(.) \\ (St, m_0, m_1) &\leftarrow A(find, pk) \\ b \leftarrow \{0,1\}\ at\ random; c^* &\leftarrow Enc_{pk}(m_b) \\ b' &\leftarrow A(guess, c^*, St). \end{aligned}$$

The advantage of such an adversary A is defined as follows:

$$Adv(A) =\mid Pr[b' = b] - \tfrac{1}{2} \mid.$$

A public key encryption scheme is said to be $\varepsilon - indistinguishable$ under CPA attacks if $Adv(A) < \varepsilon$ for any attacker A which runs in polynomial time.

From this definition, it is quite obvious that the role of the randomness r is crucial to ensure (semantic) security of a public key encryption scheme. In effect, a deterministic scheme can never be semantically secure, because an attacker A could always encrypt m_0 and m_1 by using pk, and then compare the resulting ciphertexts with the challenge one c^*, to decide the value of the bit b.

At this step, we will now give a proof of security for the entire protocol.

Proof. To analyze security let us examine the information propagated by each site taking part in the protocol. All propagated informations in our protocol can be summarized as follows:

1. Alice's view: In steps 1, 2 and 3, for each $x_i, i \in \{1..n\}$, Alice generates a random number r and computes $Enc_{pk}(r, x_i)$; the result of this computation is sent to Bob.
2. Bob's view: Bob receives $Enc_{pk}(x_i, r)$. This message is computationally indistinguishable from the received message since the semantic security of the public key encryption system guarantees that no extra information is revealed [14].
3. In step 4, Bob computes $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y}) = \prod Enc_{pk}(x_i)$ if $y_i = 1$. The homomorphic property of encryption system guarantees that this computation does not reveal the values of $x_{i \in \{1..n\}}$.
4. Then, in step 5, Bob sends $Enc_{pk}(\overrightarrow{X} \bullet \overrightarrow{Y})$ to Alice. The security of the remaining stages is not important because the result of the scalar product is not private.

6 Related Works

Let us reconsider the most popular secure scalar product in order to compare them with our protocol. The protocol in [1] is an algebraic solution which uses a matrix of decision of dimension $n \times n/2$, where n is the dimension of the vectors. Each part encrypts its vector using the matrix of decision. In first step, the primary site namely Alice, generates $n/2$ random values and performs n^2 multiply and addition operations. The responder site, namely Bob, performs $n^2 + 2$ multiply and addition operations. Finally Alice performs $n/2$ multiply and addition operations to obtain the final result. The protocol in [1] thus has a computational complexity of $O(n^2)$. The communication cost is $O(n)$, i.e. $3n/2+2$ messages if we consider that each entry of the vector requires a message. Therefore, the communication overhead is $n/2+1$. In [15], the authors proposed a secure scalar product protocol based on relation $2\sum_{j=1}^{n} x_i y_i = \sum_{j=1}^{n} x_i^2 + \sum_{j=1}^{n} y_i^2 - \sum_{j=1}^{n}(x_i - y_i)^2$. In this protocol each site needs n messages towards the other for send his vector. Bob needs to send $\sum_{j=1}^{n} y_i^2$, whereas Alice sends the result to Bob. In this

case the communication cost is $2n+2$ and the communication overhead is $n+1$ messages. In order to determine the computational overhead, we performed the following analysis. Alice will compute $\sum_{j=1}^{n} x_i^2$, and Bob will compute $\sum_{j=1}^{n} y_i^2$ locally (i.e. $2n-2$ additions and $2n$ multiply operations). Then, we execute the protocol *Add Vectors Protocol* which requires n permutations of values and n encryptions. Finally, the computational complexity is $O(n)$. If Alice has a constant vector (i.e made up only of 1 values), Bob could deduce the values from Alice. To solve this problem the authors in [15] propose that Alice generates a random vector $\overrightarrow{r}$ that is added to her random vector $\overrightarrow{X}$. Thus, even if Alice has a constant vector $\overrightarrow{X}$, the fact that $\overrightarrow{r}$ was randomly generated will ensure that $(\overrightarrow{X}+\overrightarrow{r})$ will be always a vector with non-equal values. Now, they can use the scalar product protocol twice for computing $(\overrightarrow{X}+\overrightarrow{r})\bullet\overrightarrow{Y}$ and $\overrightarrow{r}\bullet\overrightarrow{Y}$. Then, Alice can obtain $\overrightarrow{X}\bullet\overrightarrow{Y}=(\overrightarrow{X}+\overrightarrow{r})\bullet\overrightarrow{Y}-\overrightarrow{r}\bullet\overrightarrow{Y}$ and send it to Bob. This solution obviously will double the cost of the scalar product protocol, but according to the authors it still will be efficient than all existing ones. In [8], Du and Atallah propose a protocol called *Permutation Protocol*. This last uses an additive homomorphic encryption as in [15] and our protocol. In first stage Alice generates a pair of key (sk, pk) and crypt its vector with the public key pk. Then it sends its n encrypted values and the public key to Bob; this phase needs $n+1$ messages. Using the public key it crypt its vector, then uses the property of additive homomorphism to compute $Enc_{pk}(\overrightarrow{X})\times Enc_{pk}(\overrightarrow{Y})=Enc_{pk}(\overrightarrow{X}+\overrightarrow{Y})$. It permutes the entries and sends $\sigma(Enc_{pk}(\overrightarrow{X}+\overrightarrow{Y}))$ to Alice (σ represents the permutation operation). Site Alice performs $Dec_{sk}(\sigma(Enc_{pk}(\overrightarrow{X}+\overrightarrow{Y})))$ to obtain the result, then sends it to Bob. The total number of messages exchanged in this protocol is thus $2n+2$ messages. If we note by *encrypted_msg_time* the time to encrypt a message, *decrypted_msg_time* the time to decrypt a message and *permutation_time* the time to permute the entries, we have $2n\times$ *encrypted_msg_time* + *decrypted_msg_time* + *permutation_time* $= O(n)$. Now let us compare our protocol with the protocols described above. For better analyzing the efficiency of our protocol, we also compare it with $DNSP$ that is the scalar product computation without privacy constraint. In the $DNSP$ model, Bob would send his vector to Alice to compute the scalar product and Alice will need to send the result to Bob. This process requires $n+1$ messages. $DNSP$ requires n multiply and $n-1$ addition operations. Hence, the computational complexity is $O(n)$. Table 1 summarizes comparisons between our protocol and four other ones.

The table 1 highlights the differences between our protocol and four other ones under three metrics: computational complexity, communication cost and the communication overhead. If the computational complexity does not constitute

	DNSP	[8]	[1]	[15]	Our protocol
Communication cost	$n+1$	$2n+2$	$3n/2+2$	$2n+2$	$n+3$
Communication overhead	0	$n+1$	$n/2+1$	$n+1$	2
Computational complexity	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$

Fig. 1. Communication overhead and computational complexity

really a problem because there are several parallel architectures and data mining algorithms, communications will create a bottleneck that can decrease drastically the overall performance of a data mining process. In table 1 we observe that our protocol has a communication overhead of 2. Our protocol needs $n + 3$, while $DNSP$ model needs $n + 1$ without privacy. Table 1 shows clearly that our protocol outperforms of all other protocols.

7 Conclusion and Future Works

The secure scalar product is a very important issue in privacy preserving data mining. Recently, several protocols have been proposed to solve this problem. The evaluation of these protocols shows that they generate an important communication overhead. In this paper, we proposed a private scalar product protocol based on standard cryptographic techniques. This proposal has two features: (i) it is secure; (ii) it reduces the amount of communication between sites. The current version of this protocol uses only binary contexts. In the future, we plan to extend this protocol for numerical data and different kinds of databases (dense and sparse databases).

References

1. Vaidya, J.S., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July 23-26, pp. 639–644 (2002)
2. Oliveira, S.R.M., Zaiane, O., Saygin, Y.: Secure Association Rule Sharing. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS, vol. 3056, pp. 74–85. Springer, Heidelberg (2004)
3. Saygin, Y., Verykios, V., Clifton, C.: Using Unknowns to prevent discovery of Association Rules. ACM SIGMOD Record 30(4) (2001)
4. Guo, Y.H., Tong, Y.H., Tang, S.W., Yang, D.Q.: Knowledge hiding in database. Journal of Software 18(11), 2782–2799 (2007)
5. Yao Andrew, C.C.: Protocols for secure computations. In: Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science, Chicago, Illinois, November 1982, pp. 160–164 (1982)
6. Yao, A.C.C.: How to generate and exchange secrets. In: Proc. of the 27th Symposium on Foundations of Computer Science (FOCS), Toronto, Canada, October 1986, pp. 162–167 (1986)
7. Goldreich, O.: Secure multi-party computation - working draft (2000), http://citeseer.ist.psu.edu/goldreich98secure.html
8. Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: a review and open problems. In: New Security Paradigms Workshop, Cloudcroft, New Mexico, September 2001, pp. 11–20 (2001)
9. Pinkas, B.: Cryptographic Techniques for Privacy-Preserving Data Mining. ACM SIGKDD Explorations 4(2) (2002)
10. Clifton, C., Kantarcioglu, M.: Tools for privacy preserving distributed data mining. SIGKDD Explorations 4(2), 28–34 (2003)

11. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 1993, pp. 207–216 (1993)
12. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, September 1994, pp. 487–499 (1994)
13. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
15. Amirbekyan, A., Estivill-Castro, V.: A New Efficient Privacy-Preserving Scalar Product Protocol. In: Proc. Sixth Australasian Data Mining Conference (AusDM 2007), Gold Coast, Australia (2007)