

A New Efficient Privacy-Preserving Scalar Product Protocol

Artak Amirbekyan

Vladimir Estivill-Castro

School of ICT, Griffith University,
Meadowbrook QLD 4131, Australia
Email: A.Amirbekyan@gu.edu.au

Abstract

Recently, privacy issues have become important in data analysis, especially when data is horizontally partitioned over several parties. In data mining, the data is typically represented as attribute-vectors and, for many applications, the scalar (dot) product is one of the fundamental operations that is repeatedly used.

In privacy-preserving data mining, data is distributed across several parties. The efficiency of secure scalar products is important, not only because they can cause overhead in communication cost, but dot product operations also serve as one of the basic building blocks for many other secure protocols.

Although several solutions exist in the relevant literature for this problem, the need for more efficient and more practical solutions still remains. In this paper, we present a very efficient and very practical secure scalar product protocol. We compare it to the most common scalar product protocols. We not only show that our protocol is much more efficient than the existing ones, we also provide experimental results by using a real life dataset.

Keywords: Privacy Preserving Data Mining.

1 Introduction

Data mining technology allows the analysis of large amounts of data. Analysis of personal data, or analysis of corporate data (by competitors, for example) creates threats to privacy (Estivill-Castro & Brankovic 1999). Moreover, never have globalization and international collaborations placed as much demand on partnerships between governments and/or corporations as they do today. Data mining has been identified as one of the most useful tools for the fight on terror and crime (Mena 2003). However, the information needed resides with many different data holders that must share their data with each other; thus, data privacy becomes extremely important. Parties may not trust each other, but all parties are aware of the benefit brought by such collaboration. In the privacy preserving model, all parties of the partnership promise to provide their private data to the collaboration, but none of them wants the others or any third party to learn much about their private data.

Nowadays, computers can manipulate large databases and perform many data analysis tasks using data-mining techniques. Our purpose is, during such autonomous data analysis, to preserve privacy

of individuals and corporations. Data is now available from companies, shops, medical clinics, hospitals and it can be used to detect patterns to identify individuals who can be dangerous to society or obtain information which will help to make preventative decisions. But such a task is not possible without some level of privacy protection as the companies or hospitals policies protect private information. Thus a method that can achieve a balance on knowledge discovery and privacy protection is highly desirable.

For instance, Peter wants to apply for a loan to buy a house. He goes to Bank *A* and supplies the necessary documentation. Bank *A* uses *k*-nearest-neighbours (*k*-NN) classification to determine the class of an applicant as either RISKY or SAFE. This automated system classifies Peter as RISKY. Obviously, the *k*-NN were retrieved for within the database that Bank *A* holds. It could happen that Bank *A* does not have many clients that are “close” to Peter and this could lead to a wrong classification. Consequently, Bank *A* has a possible loss of profit. It is clear that the larger the database, the more accurate the classification. Thus, if it were possible to somehow combine the databases of bank *A*, bank *B* and bank *C*, classification would have been more precise — Peter could have been classified as SAFE. But this scenario is not possible because privacy restrictions would not allow banks to provide access to each other’s databases. Here is where privacy-preserving data mining comes to the rescue. This is a typical case of so called horizontally partitioned data. In privacy-preserving data mining settings, banks do not need to reveal their databases to each other. They can still apply, for instance, *k*-NN classification but preserve the privacy of their data at the same time.

For most data mining algorithms, the data is encoded as vectors in high dimensional space¹. Therefore, secure computation (preserving privacy) of dot-products (scalar products) is fundamental for many data analysis tasks. The field of statistics is supported by many operations on vectors and matrices because of its long tradition of manipulating vectorial arrangements of data. An examples of data analysis where privacy preservation has been studied and where the dot-product plays a central role is regression analysis (Amirbekyan & Estivill-Castro 2007b, Du et al. 2004, Du & Atallah 2001). Another example where privacy-preserving data analysis uses version of dot-product with some security considerations is the calculation of the product of matrices (Du et al. 2004, Du & Atallah 2001). Examples of core data mining tasks that heavily use scalar products are clustering (Amirbekyan & Estivill-Castro 2006, Estivill-Castro 2004) and decision trees classification (Du &

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at the Sixth Australasian Data Mining Conference (AusDM 2007), Gold Coast, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 70, Peter Christen, Paul Kennedy, Jiuyong Li, Inna Kolyshkina and Graham Williams, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹Attribute-vectors are the common input for learning algorithms like decision trees, artificial neural network or for clustering algorithms like *K*-Means (Vaidya & C. Clifton 2003) or *DBSCAN* (Amirbekyan & Estivill-Castro 2006).

Zhan 2002). All sorts of data mining tasks that require nearest neighbour calculations, such as k -NN classification (Amirbekyan & Estivill-Castro 2007a, Kantarcioglu & Clifton 2004, Shaneck et al. 2006). k -NN queries also enable Local Outlier Detection (Breunig et al. 2000), Shared Nearest Neighbour Clustering (Shaneck et al. 2006) and are supported by associative queries. In turn, privacy-preserving associative queries (Amirbekyan & Estivill-Castro 2007a) are supported by various types of metrics, whose metrics demands privacy-preserving calculations (Amirbekyan & Estivill-Castro 2007a, Vaidya & Clifton 2004). Metrics that compute the distance between vectors use dot-products, in fact, the dot-product is essentially a metric in itself, as illustrated by the cosine metric. Moreover, usually the dot-product is used repeatedly in all this large family of applications. The dot-product is therefore, a central building block. Computing it efficiently becomes essential, as an inefficient or slow fundamental operation has a direct and real impact on the performance of all the protocols that use it. Furthermore, efficiency is not the only issue, but practicality is an important issue as well. There is no use to describe a data mining task using a theoretical protocol for the dot-product if this protocol is in fact, difficult to implement in practice. This, means that a secure dot-product protocol should be easily implementable and not based on some theoretical approaches that are hard to implement, which propagates the lack of implementation of data analysis methods in the privacy-preserving context. Thus, we show that our protocol for a privacy-preserving dot-product is efficient and very practical, as opposed to other available alternatives in the literature.

2 Privacy-Preserving Computation

We study collaboration between several parties that wish to compute a function of their collective databases. In fact, they are to conduct data mining tasks on the joint data set that is the union of all individual data sets. Each wants the others to find as little as possible of their own private data.

To focus the discussion on privacy-preserving collaboration, we will regularly name at least two of the parties as Alice and Bob. In the so called horizontally partitioned data (see Fig. 1), some of the records are owned by Alice and the others by Bob. Obviously, for more than two parties, every party will own some part (a number of records) from the database.

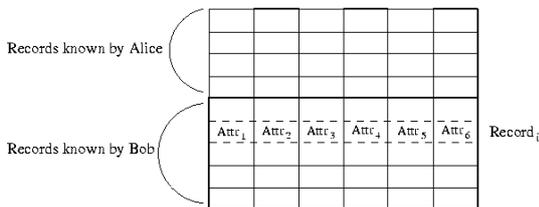


Figure 1: Horizontally partitioned data.

A direct and naive use of data mining algorithms on the union of the data requires one party to receive data (every record) from all other parties, or all parties to send their data to a trusted central place. The recipient of the data would conduct the computation in the resulting union. In settings where each party must keep their data private, this is unacceptable. Note that, for horizontally partitioned data, the more parties are involved, the more records are involved and the larger is the global database.

Our approach is based on the theory developed

under the name of “secure multiparty computation”-SMC (Goldreich 2004). Here, Alice holds one input vector \vec{x} and Bob holds an input vector \vec{y} . They both want to compute a function $f(\vec{x}, \vec{y})$ without each learning anything about the other’s input except what can be inferred from $f(\vec{x}, \vec{y})$. Yao’s Millionaires Problem (Yao 1982) provides the origin for SMC. In the Millionaires Problem, Alice holds a number a while Bob holds b . They want to identify who holds the larger value (they compute if $a > b$) without either learning anything else about the other’s value. The function $f(x, y)$ is the predicate $f(x, y) = x > y$.

Secure multi-party computation under the semi-honest model (Goldreich 2004) has regularly been used for privacy-preserving data mining (Du & Atallah 2001, Du & Zhan 2002, Vaidya & C. Clifton 2003).

Let us give a formal definition for the model of computation for our protocols. The semi-honest party is the one who follows the protocol correctly, but at the same time keeps information received during communication and final output, for further attempt to disclose private information from other parties. A semi-honest party is sometimes called an *honest but curious one* (Ioannidis et al. 2002).

The most common formal definition for the semi-honest model, is the following (Goldreich 2004):

Definition 1 (*privacy w.r.t. semi-honest behavior*): Let $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ be a functionality, and $f_1(x, y)$ (respectively, $f_2(x, y)$) denote the first (resp., second) element of $f(x, y)$. Let Π be two-party protocol for computing f . The view of the first (resp., second) party during an execution of protocol Π on (x, y) , denoted $view_1^\Pi(x, y)$ (resp., $view_2^\Pi(x, y)$), is $(x, r_1, m_1, \dots, m_t)$ (resp., $(y, r_2, m_1, \dots, m_t)$), where r_1 represents the outcome of the first (resp., r_2 the second) party’s internal coin tosses, and m_i represents the i -th message it has received. The output of the first (resp., second) party after an execution of Π on (x, y) is denoted $output_1^\Pi(x, y)$ (resp., $output_2^\Pi(x, y)$), and is implicit in the party’s view of the execution, and $output^\Pi(x, y) = (output_1^\Pi(x, y), output_2^\Pi(x, y))$

- (general case) We say that Π privately computes f if there exists a probabilistic polynomial-time algorithm, denoted S_1 and S_2 , such that

$$\begin{aligned} & \{S_1(x, f_1(x, y), f(x, y))\}_{x, y \in \{0,1\}^*} \\ & \equiv^C \{view_1^\Pi(x, y), output^\Pi(x, y)\}_{x, y \in \{0,1\}^*} \\ & \{S_2(y, f_2(x, y), f(x, y))\}_{x, y \in \{0,1\}^*} \\ & \equiv^C \{view_2^\Pi(x, y), output^\Pi(x, y)\}_{x, y \in \{0,1\}^*} \end{aligned}$$

where \equiv^C denotes computational indistinguishability by (non-uniform) families of polynomial-size circuits. Here $view_1^\Pi(x, y), view_2^\Pi(x, y), output_1^\Pi(x, y)$ and $output_2^\Pi(x, y)$ are related random variables, defined as a function of the same random execution. In particular, $output_i^\Pi(x, y)$ is fully determined by $view_i^\Pi(x, y)$.

This definition basically says, that a computation is secure if the view of each party during the execution of the protocol can be simulated from the input and the output of that party. Thus, for a security proof, it is enough to show the existence of a simulator for each party that satisfies the above equations, and no other party notices that its partner has been replaced by the simulator. Note however, that whatever information is derived or inferred from final result cannot obviously be kept secret.

For instance, Alice, Bob and Charles each hold private numbers and want to calculate the average of their numbers. Using SMC protocols they discovered that the average is m . Assume Charles holds c that is greater than m . This immediately discloses that at least one of the other parties holds a number less than the value m . This information was inferred from the final result and not during the execution of the protocol, thus the protocol still can be considered secure under the semi-honest model.

Theorem 1 (*Composition theorem for the semi-honest model*): (Goldreich 2004) Suppose that g is privately reducible to f and that there exists a protocol for privately computing f . Then there exists a protocol for privately computing g .

This approach became very popular because cryptography offers a well defined model for privacy, which includes methodologies for proving and quantifying it. Moreover, there exist many tools of cryptographic algorithms that can be used for implementing privacy-preserving data mining algorithms.

The SMC literature has a general solution for all polynomially bound computations (Goldreich et al. 1987). This generic “shares” solution computes $f(\vec{x}, \vec{y})$ for a polynomial-time f using private input \vec{x} from Alice and private input \vec{y} from Bob. Alice learns nothing about \vec{y} except what can be computed from $f(\vec{x}, \vec{y})$ and similarly Bob learns nothing about \vec{x} except what can be inferred from \vec{y} and $f(\vec{x}, \vec{y})$. Why, if such a solution exists, is there so much interest in protocols for SMC? The first aspect is that the general solution requires f to be explicitly represented as a Boolean circuit of polynomial size. Even if represented as a circuit of polynomial size in its input, the input must be very small for the circuit to have practical polynomial size. This means, the sub-task that uses this result must be on small inputs, a constraint difficult to meet in data mining applications. Third, the constants involved are not small, so once the circuit is described the parties enter into a protocol, holding shares of the inputs to gates and shares of the outputs of gates. Fourth, the privacy-preserving literature shows the need for practical and efficient solutions that are not based on this general theoretical solutions. It also shows that much more efficient solutions exist for special cases of f .

3 Related Work

Assume the following scenario: Alice holds a vector \vec{a} and Bob holds vector \vec{b} (both with n elements). The goal is to compute $\vec{a}^T \cdot \vec{b}$ securely. We review the most common secure scalar product protocols (Du & Zhan 2002, Vaidya & C. Clifton 2002, Du & Atallah 2001, Ioannidis et al. 2002). Some secure scalar product protocols (Du & Zhan 2002) use a semi-trusted third party’s commodity server. Semi-trusted means that it should not learn any private information from the parties and should not collude with either Alice or Bob. This solution is not considered secure (Vaidya et al. 2006). under the semi-honest model since it uses a third party. The communication cost of this protocol is $4n$, which is 4 times more expensive than in the distributed non-private setting (*DNISP*) cost of a two-party scalar product (the *DNISP* cost of a scalar product is defined as the cost of computing $\vec{a}^T \cdot \vec{b}$ without the privacy constraints, namely one party just sends its data to the other party). The communication cost can be reduced to $2n$ if the commodity server sends just the seeds to the parties.

Another two solutions to the scalar product protocol have been proposed (Du & Atallah 2001, Vaidya

& C. Clifton 2002). Both of these provide privacy preservation without using a third party. However the communication and computation costs are more expensive than if one uses the solution with commodity server. Both solutions have communication cost $O(n)$ but with much larger constants (4 rounds with bitwise cost of $2nM$, where M is the maximum number of bits needed to represent any input value) under the $O(n)$ complexity (recall that n stands for the size of the vectors). Computational cost is also high, for instance, the second solution has a $O(n^2)$ computational cost (Vaidya & C. Clifton 2002).

Two solutions are presented by Du and Atallah for the secure scalar product named Protocol 1 and Protocol 3 (Du & Atallah 2001). Protocol 3 is more efficient than Protocol 1. For Protocol 3, by setting the security parameter $\mu = p^t$, which should be large enough, the communication cost of Protocol 3 becomes $4 \log(\mu \frac{nd}{\log(n)})$, where d is the number of bits needed to represent any number in the input.

For another secure dot-product protocol (Ioannidis et al. 2002). the authors present two kinds of overhead analysis. First, the communication overhead is the amount of extra communications compared to the *DNISP* cost. That is, the communication overhead with respect to the $n + 1$ messages of *DNISP* (where Bob sends his n values and Alice sends the result back). The second type of overhead is the computation overhead with *DNISP* (that requires n products and $n - 1$ additions that Alice performs). Thus, the computation overhead is the extra computation performed. The authors claim that their protocol is more efficient in both overheads than the protocols that use conventional cryptographic techniques. Their experimental results show a total overhead to 4.69 on average. However, we tried to carry out the analysis of this protocol ourselves and we found that this protocol was probably described incorrectly or errors were introduced during publication. Namely, the published description seems to compute a scalar product incorrectly. After private communications with the authors, they acknowledged the problem that we have raised. So, at this stage, we exclude this protocol from further discussion.

4 Review of Tools to be Used

We now describe some cryptographic tools.

4.1 Homomorphic encryption

An encryption scheme is called *additive homomorphic* (Paillier 1999) if it has the following property:

$$E(x_1) \times E(x_2) = E(x_1 + x_2).$$

But other implementations are possible (Naccache & Stern 1998, Okamoto & Uchiyama 1998) including one using RSA. By induction, it is not hard to show that in an homomorphic scheme,

$$E(x_1) \times E(x_2) \times \dots \times E(x_n) = E(x_1 + x_2 + \dots + x_n).$$

4.2 Review of the ADD VECTORS PROTOCOL

The technique was introduced for manipulation of vector operations as the “permutation protocol” (Du & Atallah 2001) and is also known as the “permutation algorithm” (Vaidya & C. Clifton 2003).

In this protocol, Alice has a vector \vec{x} while Bob has vector \vec{y} and a permutation π . The goal is for Alice to obtain $\pi(\vec{x} + \vec{y})$; that is Alice obtains the sum \vec{s} of the vectors in some sense. The entries are randomly permuted, so Alice cannot perform $\vec{s} - \vec{x}$ to find \vec{y} .

Also, Bob is not to learn \vec{x} . Solutions for $P = 2$, that is, two parties, are based on homomorphic encryption.

Protocol 1 1. Alice produces a key pair for a homomorphic public key system and sends the public key to Bob. We denote by $E(\cdot)$ and $D(\cdot)$ the corresponding encryption and decryption system.

2. Alice encrypts $\vec{x} = (x_1, \dots, x_n)^T$ and sends $E(\vec{x}) = (E(x_1), \dots, E(x_n))^T$ to Bob.

3. Using the public key from Alice, Bob computes $E(\vec{y}) = (E(y_1), \dots, E(y_n))^T$ and uses the homomorphic property to compute $E(\vec{x} + \vec{y}) = E(\vec{x}) \times E(\vec{y})$. Then, he permutes the entries by π and sends $\pi(E(\vec{x} + \vec{y}))$ to Alice.

4. Alice obtains $D(\pi(E(\vec{x} + \vec{y}))) = \pi(\vec{x} + \vec{y})$.

This can be extended to the case of $P \geq 3$ vectors, that is $P > 2$ parties are involved. Because the parties do not collude, this protocol accepts faster implementations that do not need to permute the result, because $D(\cdot)$ is known only by Alice, and Alice will get the value $E(\vec{v}_2 + \dots + \vec{v}_P)$, where \vec{v}_i is the vector owned by i^{th} party. The algorithm is as follows:

Protocol 2 1. The 1^{st} party (Alice), generates $E(\cdot)$ and $D(\cdot)$, then sends only $E(\cdot)$ to the other parties.

2. Then, the P^{th} party encrypts his data $E(\vec{v}_P)$ and sends it to the $(P - 1)^{\text{th}}$ party.

3. Next, the $(P - 1)^{\text{th}}$ party encrypts his data $E(\vec{v}_{P-1})$ and using the homomorphic encryption property computes

$$E(\vec{v}_{P-1}) \times E(\vec{v}_P) = E(\vec{v}_{P-1} + \vec{v}_P)$$

and sends this to the $(P - 2)^{\text{th}}$ party.

4. The protocol continues until Alice (the first party) will get $E(\vec{v}_2 + \dots + \vec{v}_{P-1} + \vec{v}_P)$ and she adds her data in the same way.

5. As Alice owns $D(\cdot)$, she decrypts the results and sends them to all the other parties.

Note that in Step 1, the P^{th} party does not need to permute his result because the $(P - 1)^{\text{th}}$ party does not know $D(\cdot)$ to decrypt. It is also the case that the encryption mechanism could be much less costly. For example, if we are prepared to know the distribution of values, although no specific value is revealed, we do not need homomorphic encryption. In this case $E(\cdot)$ could be as simple as adding a random number in a sufficiently large additive field F (or X-OR with a random bit mask) and consequently $D(\cdot)$ will be subtracting the random number previously added.

One can easily also notice that for $P > 2$ (when we do not use permutation) this method can be applied for adding not only vectors but matrices or just numbers as well. This, however, sometimes is called the “SECURE SUM PROTOCOL” (Vaidya et al. 2006).

5 New Privacy-Preserving Scalar Product Protocol

In this section we propose a new simple scalar vector product protocol which is based on the ADD VECTORS PROTOCOL (see Section 4.2). This protocol is very simple and it is easy to implement. Depending on the domain and encryption it is also very efficient. In this protocol again, Alice has a vector \vec{a} and Bob has another vector \vec{b} (both with n elements). Alice

and Bob use the protocol to compute the scalar product $\vec{a}^T \cdot \vec{b}$ between \vec{a} and \vec{b} , such that Alice gets $\vec{a}^T \cdot \vec{b}$. Note that

$$2a_i b_i = a_i^2 + b_i^2 - (a_i - b_i)^2,$$

therefore

$$2 \sum_{i=1}^n a_i b_i = \sum_{i=1}^n a_i^2 + \sum_{i=1}^n b_i^2 - \sum_{i=1}^n (a_i - b_i)^2;$$

thus the protocol works as follows:

Protocol 3 (New Scalar Product Protocol)

1. Alice and Bob apply the ADD VECTORS PROTOCOL for Alice to obtain $\pi_0(\vec{a} - \vec{b})$, where π_0 is a permutation generated by Bob.

2. Alice can obtain

$$\sum_{\pi_0(i)=1}^n (a_{\pi_0(i)} - b_{\pi_0(i)})^2 + \sum_{i=1}^n a_i^2$$

and Bob can compute $\sum_{i=1}^n b_i^2$.²

3. Now Bob can send $\sum_{i=1}^n b_i^2$ to Alice which will allow Alice to compute the scalar product, that is

$$\begin{aligned} 2\vec{a}^T \cdot \vec{b} &= 2 \sum_{i=1}^n a_i b_i \\ &= \sum_{i=1}^n a_i^2 + \sum_{i=1}^n b_i^2 - \sum_{\pi_0(i)=1}^n (a_{\pi_0(i)} - b_{\pi_0(i)})^2. \end{aligned}$$

Alice learns all the values $\pi(a_i - b_i)$ from the information collected during the protocol’s execution. However, this is not enough for Alice to discover any of Bob’s private data, because of the permutation applied by Bob. Essentially, this protocol’s security is the same as the security of ADD VECTORS PROTOCOL. Note also that Alice learns the value of $\sum_{i=1}^n b_i^2$, but again (for $n \geq 2$) is insufficient to learn any of Bob’s private data.

Note that the encryption used in the ADD VECTORS PROTOCOL can be as simple as adding a random vector consisting of the same random number. Alice then adds this random vector to her vector and sends the results to Bob, whereas Bob only adds his vector to the sum vector received and performs a permutation before he sends it back to Alice. However, this solution should be handled carefully, because if Bob knows any of the coordinates in Alice’s vector, then he immediately discovers the random number and consequently all of Alice’s vector.

5.1 Analysis of security for special cases

Assume all the values in Alice’s vector are equal, then there are some information leaks in the ADD VECTORS PROTOCOL. First, Alice learns all of Bob’s values, although not of all of Bob’s information, since she learns them shuffled. That is, she learns the set of values in Bob’s vector. On the other hand, because Bob learns all of Alice’s encrypted values, Bob learns that Alice’s values are equal. Therefore, Bob could detect the pathological case before continuing with the protocol. While this premature abortion of the protocol by Bob will ensure that his values remain

²Note that, if they stop execution in this step they will have scalar product distributed in private shares.

private, Alice would not be protected against the fact that Bob discovered Alice holds a vector with all entries constant (although he does not know the value of this constant).

Since our scalar vector product protocol is based on the ADD VECTORS PROTOCOL, we have to take this issue into account. This leads to consider an analysis of the ADD VECTORS PROTOCOL in more detail. First, at least for regression, it is likely that Alice holds a column in a matrix with all values set to a constant. Since the interpolation matrix holds rows with entries $\langle 1, x_i, x_i^2, x_i^3, \dots \rangle$ for each data point, the first column may be all set to 1. This may be a fact that Bob knows anyways because of the nature of regression, but this should not mean Bob needs to reveal a set of values on one of his vectors. This unpleasant situation can be avoided for our protocol by the fact that the scalar vector product has the following property

$$(\vec{a} + \vec{r})^T \cdot \vec{b} = \vec{a}^T \cdot \vec{b} + \vec{r}^T \cdot \vec{b}.$$

Namely, assume Alice and Bob want to compute $\vec{a}^T \cdot \vec{b}$. Alice can generate a random vector \vec{r} and add to her vector \vec{a} . Thus, even if Alice has a constant vector \vec{a} , the fact that \vec{r} was randomly generated will ensure that $\vec{a} + \vec{r}$ will be always a vector with non-equal values. Now, they can use our scalar product protocol twice for computing $(\vec{a} + \vec{r})^T \cdot \vec{b}$ and $\vec{r}^T \cdot \vec{b}$. Then, Alice can obtain

$$\vec{a}^T \cdot \vec{b} = (\vec{a} + \vec{r})^T \cdot \vec{b} - \vec{r}^T \cdot \vec{b}$$

and send it to Bob. This solution obviously will double the cost of our scalar product protocol, but it still will be efficient than all existing ones and avoid the problem of the ADD VECTORS PROTOCOL.

6 Experimental Results

Privacy is not without cost. In a situation when privacy protection is necessary, protocols that do not ensure some level of privacy will not be considered at all. The performance of privacy-preserving protocols is usually studied with respect to the cost in *DNISP*. Here we have two kinds of overhead, namely communication overhead and computational overhead. Communication overhead of our protocol is very easy to derive. For each entry in the vector, our protocol requires one messages from Alice to Bob, and again one messages from Bob to Alice. Bob needs to send $\sum_{i=1}^n b_i^2$ to Alice, whereas Alice sends the result to Bob. So, the communication cost is $2n + 2$, where n is the dimension of the vectors. In the *DNISP* model, Bob would send his vector to Alice for her to compute the scalar product and Alice will need to send the result to Bob. This is a communication cost of $n + 1$. Thus, the communication overhead is $n + 1$ messages carrying a floating-point value. In special cases when Alice's has a constant vector, the cost is doubled, making it $4n + 4$ and the overhead is $3n + 3$.

In order to calculate the computational overhead, we performed the following analysis. Alice will compute $\sum_{i=1}^n a_i^2$, and Bob will compute $\sum_{i=1}^n b_i^2$ locally. Then, there will be one executions of the ADD VECTORS PROTOCOL which implies n data swaps to implement the permutation π_0 as suggested by (Reingold et al. 1977). Also, there needs to be $n - 1$ random indexes generated to create this permutation. Moreover, we need n encryptions for the ADD VECTORS PROTOCOL, even if they are implemented as adding a random number. While this overhead is linear in the dimension of the vectors, it now involves a slightly

more diverse set of fundamental operations (it is not just additions and multiplications, but there are random bits generated as well as data swaps).

To evaluate the overhead this represents in a realistic setting, we have implemented our algorithm in C++ and with SHELL scripts. We use the CoIL 2000 Challenge (van der Putten & van Someren 2000) dataset which contains information on customers of an insurance company. The data consists of 86 variables with 5821 records and includes product usage data and socio-demographic data derived from zip area codes³. We compute the scalar product for 500 pairs of vectors from this datasets by using our protocol. The scalar product computation without privacy constrains, that is *DNISP*, on average requires 1.413 microseconds of execution time (with a confidence interval of 95% given by ± 0.144709). Our protocol requires 6.37 microseconds of execution time on average (with a ± 0.479215 95%-confidence interval). Thus, the overhead on average is around 4.51 times. Given our analysis, this overhead is reasonable and within the expected range.

7 Final Remarks

While our protocol is efficient, if used without the care for special cases, it is as secure as the ADD VECTORS PROTOCOL. The ADD VECTORS PROTOCOL is not covered under the semi-honest model of multi-party computation, because only one party obtains the output. It also suffer the weaknesses mentioned in Section 5.1, when all the values in Alice's vector are equal.

We need to refer to the semi-honest model of computation to clarify the issue further. It is very hard that two parties compute the sum of respective private values and do not disclose information on such private values because each can subtract its private value to the commonly known sum and discover the other side's value. To fit the semi-honest model, we could formulate the problem better. We call the new problem the *add-vectors permuted-outputs computation*. Here, we require that two parties holding n -dimensional private vectors each find the set of values of the sum of these two vectors (and in a way that all possible permutations are equally likely to correspond to the actual vectorial sum).

Note that the current ADD VECTORS PROTOCOL in the literature could almost achieve a solution to the *add-vectors permuted-outputs computation* problem by Alice applying a permutation σ of her own to the values of $\pi(\vec{a} + \vec{b})$ and sending $\sigma(\pi(\vec{a} + \vec{b}))$ to Bob. However, this modified ADD VECTORS PROTOCOL still suffers from the fact that if all of Alice's values are equal, then Bob finds this fact and Alice finds the set of values in Bob's vector.

The ideal, theoretical semi-honest protocol for *add-vectors permuted-outputs computation* would also leak information. A party with a vector with equal entries engaging in an add-vectors permuted-outputs ideal semi-honest protocol, can retrieve from its input and the permuted output, the set of values of the other party. Moreover, the theoretical solution ensures the other party cannot detect this leak. So, perhaps it is an advantage the way the current ADD VECTORS PROTOCOL operates. It is also an interesting problem if the theoretical solution for the add-vectors permuted-outputs setting can be made implementable. However our protocol does not suffers from this fact. Since, as we have shown in Section 5.1, we can always avoid this situation by adding a small overhead in communication and computation costs.

³This dataset is in UCI KDD repository.

References

- Amirbekyan, A. & Estivill-Castro, V. (2006), Privacy preserving *DBSCAN* for vertically partitioned data, in 'IEEE International Conference on Intelligence and Security Informatics, ISI 2006', Springer Verlag Lecture Notes in Computer Science 3975, San Diego, CA, USA, pp. 141–153.
- Amirbekyan, A. & Estivill-Castro, V. (2007a), The privacy of k -nn retrieval for horizontal partitioned data — new methods and applications, in J. Bailey & A. Fekete, eds, 'Eighteenth Australasian Database Conference (ADC2007)', Vol. 63 of *Conferences in Research and Practice in Information Technology (CRPIT)*, CORE, Australian Computer Society, Ballarat, Victoria, Australia, pp. 33–42.
- Amirbekyan, A. & Estivill-Castro, V. (2007b), Privacy preserving regression algorithms, in 'The 3rd WSEAS International Symposium on Data Mining and Intelligent Information Processing, ISDM 2007', Beijing, China, pp. 37–45.
- Breunig, M., Kriegel, H.-P., Ng, R. & Sander, J. (2000), Lof: Identifying density-based local outliers, in W. Chen, J. F. Naughton & P. Bernstein, eds, 'Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data', ACM, Dallas, Texas, USA, pp. 93–104.
- Du, W. & Atallah, M. (2001), Privacy-preserving cooperative statistical analysis, in 'Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)', ACM SIGSAC, IEEE Computer Society, New Orleans, Louisiana, pp. 102–110.
- Du, W., Han, Y.-S. & Chen, S. (2004), Privacy-preserving multivariate statistical analysis: Linear regression and classification, in B. M. W., U. Dayal, C. Kamath & D. Skillicorn, eds, '2004 SIAM International Conference on Data Mining', Lake Buena Vista, Florida.
- Du, W. & Zhan, Z. (2002), Building decision tree classifier on private data, in V. Estivill-Castro & C. Clifton, eds, 'Privacy, Security and Data Mining', IEEE ICDM Workshop Proceedings, Volume 14 in the Conferences in Research and Practice in Information Technology Series, Australian Computer Society, Sydney, Australia, pp. 1–8.
- Estivill-Castro, V. (2004), Private representative-based clustering for vertically partitioned data, in R. Baeza-Yates, J. Marroquin & E. Chávez, eds, 'Fifth Mexican International Conference on Computer science (ENC 04)', SMCC, IEEE Computer Society Press, Colima, Mexico, pp. 160–167.
- Estivill-Castro, V. & Brankovic, L. (1999), Data swapping: Balancing privacy against precision in mining for logic rules., in 'Data Warehousing and Knowledge Discovery, First International Conference', Vol. 1676 of *Lecture Notes in Computer Science*, Springer, pp. 389–398.
- Goldreich, O. (2004), *The Foundations of Cryptography*, Vol. 2, chapter General Cryptographic Protocols, Cambridge University Press.
- Goldreich, O., Micali, S. & Wigderson, A. (1987), How to play any mental game (extended abstract), in A. Aho, ed., 'Proceedings of the 19th ACM Annual Symposium on Theory of Computing', ACM Press, New York, pp. 218–229.
- Ioannidis, I., Grama, A. & Atallah, M. J. (2002), A secure protocol for computing dot-products in clustered and distributed environments., in '31st International Conference on Parallel Processing (ICPP)', Vancouver, BC, Canada, pp. 379–384.
- Kantarcioglu, M. & Clifton, C. (2004), Privately computing a distributed k -nn classifier, in J.-F. Boulicaut, ed., '8-th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD', Vol. 3202, Springer Verlag Lecture Notes in Computer Science, pp. 279–290.
- Mena, J. (2003), *Investigative Data Mining for Security and Criminal Detection*, Butterworth-Heinemann, US.
- Naccache, D. & Stern, J. (1998), A new public key cryptosystem based on higher residues, in L. Gong & M. Reiter, eds, '5th ACM Conference on Computer and Communications Security', SIGSAC, ACM Press, San Francisco, CA., pp. 59–66.
- Okamoto, T. & Uchiyama, S. (1998), A new public-key cryptosystem as secure as factoring, in K. Nyberg, ed., 'EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques', Springer Verlag Lecture Notes in Computer Science 1403, Espoo, Finland, pp. 308–318.
- Paillier, P. (1999), Public-key cryptosystems based on composite degree residuosity classes., in 'EUROCRYPT', Vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.
- Reingold, E., Nievergelt, J. & Deo, N. (1977), *Combinatorial Algorithms, Theory and Practice*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Shaneck, M., Kim, Y. & Kumar, V. (2006), Privacy preserving nearest neighbor search, Technical Report 06-0149, University of Minnesota, Department of Computer Science and Engineering, 4-192 EECS Building, 200 Union St SE, Minneapolis, MIN, USA. To appear in the 2006 IEEE International Workshop on Privacy Aspects of Data Mining, December 2006.
- Vaidya, J. & C. Clifton, C. (2002), Privacy preserving association rule mining in vertically partitioned data, in 'The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', SIGKDD, ACM Press, Edmonton, Canada, pp. 639–644.
- Vaidya, J. & C. Clifton, C. (2003), Privacy-preserving k -means clustering over vertically partitioned data, in 'Proceedings of the SIGKDD-ACM Conference of Data Mining', ACM Press, Washington, D.C., US, pp. 206–215.
- Vaidya, J. & Clifton, C. (2004), Privacy-preserving outlier detection., in '4th IEEE International Conference on Data Mining (ICDM 2004)', IEEE Computer Society, Brighton, UK.
- Vaidya, J., Clifton, C. & Zhu, M. (2006), *Privacy Preserving Data Mining*, Vol. 19 of *Advances in Information Security*, Springer, New York, NY, USA.
- van der Putten, P. & van Someren, M. (2000), Coll challenge 2000: The insurance company case., Technical Report 2000-09, Leiden Institute of Advanced Computer Science, Amsterdam.
- Yao, A. (1982), Protocols for secure computation, in 'IEEE Symposium of Foundations of Computer Science', IEEE Computer Society, pp. 160–164.