Efficient Secure Primitive for Privacy Preserving Distributed Computations

Youwen Zhu[1,2,⋆], Tsuyoshi Takagi[1], and Liusheng Huang[2]

[1] Institute of Mathematics for Industry, Kyushu University,
Fukuoka, 819-0395, Japan
[2] National High Performance Computing Center at Hefei,
Department of Computer Science and Technology,
University of Science and Technology of China,
Hefei, 230026, China
zhuyw@mail.ustc.edu.cn

Abstract. Scalar product protocol aims at securely computing the dot product of two private vectors. As a basic tool, the protocol has been widely used in privacy preserving distributed collaborative computations. In this paper, at the expense of disclosing partial sum of some private data, we propose a linearly efficient Even-Dimension Scalar Product Protocol (EDSPP) without employing expensive homomorphic cryptosystem and third party. The correctness and security of EDSPP are confirmed by theoretical analysis. In comparison with six most frequently-used schemes of scalar product protocol (to the best of our knowledge), the new scheme is a much more efficient one, and it has well fairness. Simulated experiment results intuitively indicate the good performance of our novel scheme. Consequently, in the situations where divulging very limited information about private data is acceptable, EDSPP is an extremely competitive candidate secure primitive to achieve practical schemes of privacy preserving distributed cooperative computations. We also present a simple application case of EDSPP.

Keywords: privacy preserving, distributed computation, scalar product protocol.

1 Introduction

The advances of flexible and ubiquitous transmission mediums, such as wireless networks and Internet, have triggered tremendous opportunities for collaborative computations, where independent individuals and organizations could cooperate with each other to conduct computations on the union of data they each hold. Unfortunately, the collaborations have been obstructed by security and privacy concerns. For example, a single hospital might not have enough cases to analyze some special symptoms and several hospitals need to cooperate with each other to study their joint database of case samples for the comprehensive analysis

⋆ corresponding author.

G. Hanaoka and T. Yamauchi (Eds.): IWSEC 2012, LNCS 7631, pp. 233–243, 2012.

results. A simple way is that they share respective private database and bring the data together in one station for analysis. However, despite various shared benefits, the hospitals may be unwilling to compromise patients' privacy or violate any relevant law and regulation [1, 2]. Consequently, some techniques [3, 4] for privacy preserving distributed collaborative computations were introduced to address the concerns by privacy advocates. Nowadays, a large amount of attention [5–7] has been paid to dealing with the challenges of how to extract information from distributed data sets owned by independent parties while no privacy is breached.

Actually, many privacy preserving problems in distributed environments can essentially be reduced to securely computing the scalar product of two private vectors. Some recent examples are as follows. Murugesan *et al.* [8] proposed privacy preserving protocols to securely detect similar documents between two parties while documents cannot be publicly disclosed to each other, and the main process of their schemes, securely computing the cosine similarity between two private documents, is achieved by scalar product protocol. A privacy preserving hop-distance computation protocol in wireless sensor networks is introduced in [9] and secure scalar product protocol is used to privately compute the value of $\sum x_i y_i$, where x_i and y_i are the private coordinates. Then, the distance $S^2 = \sum (x_i - y_i)^2 = \sum x_i^2 - 2 * \sum x_i y_i + \sum y_i^2$ can be securely obtained. See [6, 7, 10, 11] for more concrete applications of scalar product protocol.

As secure computation of private vectors is fundamental for many privacy preserving distributed computing tasks, several schemes [12–16] have been proposed to perform the secure computation. Du and Zhan presented two practical schemes in [12]: scalar product protocol employing commodity server (denoted as SPP-CS) and scalar product protocol using random invertible matrix (denoted as SPP-RIM). Through algebraic transformation, another scalar product protocol was introduced in [13] (denoted as ATSPP). Based on homomorphic encryption, two solutions for securely computing dot product of private vectors are given in [14] (denoted as GLLM-SPP) and [15] (denoted as AE-SPP) respectively. A polynomial-based scalar product protocol (denoted as PBSPP) was lately presented by Shaneck and Kim [16]. The computational complexity of SPP-RIM and ATSPP is $O(n^2)$ where n is the dimensionality of private vectors. SPP-CS and PBSPP have good linear complexity, but they employ one or more semi-trusted third parties, such as the commodity server in SPP-CS. GLLM-SPP and AE-SPP encrypt the private elements by using expensive homomorphic cryptosystem. As is well known, the public key cryptosystems are typically computationally expensive and they are far from efficient enough to be used in practice. The protocols will be vulnerable to unavoidable potential collusion attacks while employing the semi-trusted third parties. As a result, previous schemes of scalar product protocol are still far from being practical in most situations.

In this paper, we focus on the useful secure primitive, scalar product protocol [12], and propose a simple and linearly efficient protocol for securely computing the scalar product of two private vectors, even-dimension scalar product

protocol (EDSPP). The novel scheme does not employ homomorphic encryption system and any auxiliary third party. Theoretical analysis confirms that the protocol is correct and no private raw data is revealed although it brings about some limited information disclosure. Simulated experiment results and comparison indicate that the new scheme has good fairness and it is much more efficient than the previous ones. As a result, our new scheme is a competitive secure candidate to achieve practical schemes of privacy preserving distributed cooperative computations while disclosing partial information is acceptable. Similar to the existing works [12–16], our protocol is also under semi-honest model [17], where each participant will correctly follow the protocols while trying to find out potentially confidential information from his legal medium records. It is remarkable that the semi-honest assumption is reasonable and practicable, as the participants in reality may strictly follow the protocols to exactly obtain the profitable outputs.

The rest of the paper is organized as follows. Section 2 proposes the new solution for scalar product protocol, and then presents the theoretical analysis of its correctness, security, communication overheads and computation complexity. The performance comparison and experiment results are displayed in section 3. At last, section 4 concludes the paper.

2 Even-Dimension Scalar Product Protocol

2.1 Problem Definition and Our Scheme

In scalar product protocol, there are two participants, denoted as Alice and Bob. Alice privately holds a vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ and Bob has the other private vector $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$, where n is a positive integer. Their goal is that Alice receives a confidential number u and Bob obtains his private output v while the private vector is not disclosed to the other party or anyone else. Here, u and v meet $\boldsymbol{x} \cdot \boldsymbol{y} = u + v$. That is, scalar product protocol enables two participants to securely share the dot product of their confidential vectors in the form of addition.

As a secure primitive, scalar product protocol [12, 14] has extensive privacy preserving applications and an efficient scalar product protocol will boost the practical process of privacy preserving distributed cooperative computation. In this paper, we consider a special case where n is an even number (suppose $n = 2k$, k is a positive integer). Then, at the expense of disclosing partial sum of some private data, we propose an efficient Even-Dimension Scalar Product Protocol (EDSPP). In our scheme, the private data is hidden by stochastic transformation, and each participant obtains a private share of the scalar product of their private even-dimension vectors at last. The novel scheme has linear complexity and no third party is employed. Besides, it just needs a secure channel to securely transmit the data and does not use any public key cryptosystem. The detailed steps are displayed in protocol 1. In step 1.1 of the scheme, the participants protect their private numbers through randomization. Then, step 1.2 works out the secure share of the scalar product of each two dimensions. Finally, they

privately obtain the expected outcomes in step 2. As can be seen from protocol 1, the private vectors are handled two by two dimensions, thus, our new scheme can only compute the dot product of even-dimension vectors.

Protocol 1. Even-Dimension Scalar Product Protocol (EDSPP)

Input: Alice has a private $2k$-dimension vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_{2k})$ and Bob holds another confidential $2k$-dimension vector $\boldsymbol{y} = (y_1, y_2, \cdots, y_{2k})$. ($k \in \mathbb{Z}^+$, $x_i, y_i \in \mathbb{R}, i = 1, 2, \cdots, 2k$)

Output: Alice obtains private output u and Bob securely gets v which meet

$$u + v = \boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{2k} x_i y_i.$$

1: **Step 1**:
2: **for** $j = 1$ to k **do**
3: **Step 1.1**: Alice locally generates two random real numbers a_j and c_j such that $a_j + c_j \neq 0$. Then, she computes $p_j = a_j + c_j$, $x'_{2j-1} = x_{2j-1} + a_j$ and $x'_{2j} = x_{2j} + c_j$, and sends $\{p_j, x'_{2j-1}, x'_{2j}\}$ to Bob by a secure channel. Bob randomly generates two real numbers b_j and d_j which meet $b_j - d_j \neq 0$, and computes $q_j = b_j - d_j$, $y'_{2j-1} = b_j - y_{2j-1}$ and $y'_{2j} = d_j - y_{2j}$. Then, he securely sends $\{q_j, y'_{2j-1}, y'_{2j}\}$ to Alice.
4: **Step 1.2**: Alice locally calculates

$$u_j = y'_{2j-1}(x_{2j-1} + 2a_j) + y'_{2j}(x_{2j} + 2c_j) + q_j(a_j + 2c_j)$$

and Bob, by himself, computes

$$v_j = x'_{2j-1}(2y_{2j-1} - b_j) + x'_{2j}(2y_{2j} - d_j) + p_j(d_j - 2b_j).$$

5: **end for**
6: **Step 2**: Alice obtains $u = \sum_{j=1}^{k} u_j$ and Bob gets $v = \sum_{j=1}^{k} v_j$.

To visually illustrate how our novel scheme works, we give a concrete example as follows. Alice has a 4-dimension vector $\boldsymbol{x} = (2.3, -81.9, 96.7, -27.1)$, and Bob's private vector is $\boldsymbol{y} = (-19.5, -78.1, 39.2, 52.8)$. According to protocol 1, they, by the following procedures, can obtain the scalar product's private shares u and v, which meet $u + v = \boldsymbol{x} \cdot \boldsymbol{y}$, respectively.

- Alice generates random numbers: $a_1 = -53.0$ and $c_1 = 99.8$ for the first two dimensions of $\boldsymbol{x}$. Then, she computes
  $p_1 = a_1 + c_1 = 46.8$, $x'_1 = 2.3 + a_1 = -50.7$, $x'_2 = -81.9 + c_1 = 17.9$,
  and sends $\{p_1, x'_1, x'_2\}$ to Bob. At the same time, Bob randomly selects: $b_1 = 28.7$ and $d_1 = 11.3$ for the first two dimensions of $\boldsymbol{y}$. Then, he computes
  $q_1 = b_1 - d_1 = 17.4$, $y'_1 = b_1 - (-19.5) = 48.2$, $y'_2 = d_1 - (-78.1) = 89.4$,
  and sends $\{q_1, y'_1, y'_2\}$ to Alice.
- Analogously, for the latter two dimensions, Alice and Bob generates random numbers $\{a_2 = -81.1,\ c_2 = -17.5\}$ and $\{b_2 = -56.9,\ d_2 = -31.2\}$, respectively. Alice computes $p_2 = -98.6$, $x'_3 = 15.6$, $x'_4 = -44.6$, and Bob

computes $q_2 = -25.7$, $y'_3 = -96.1$, $y'_4 = -84.0$. Then, they send $\{p_2, x'_3, x'_4\}$ and $\{q_2, y'_3, y'_4\}$ to each other.

– Alice and Bob computes $\{u_1, u_2\}$ and $\{v_1, v_2\}$, respectively, by the following way.

$$u_1 = y'_1(x_1 + 2a_1) + y'_2(x_2 + 2c_1) + q_1(a_1 + 2c_1) = 8074.88$$

$$u_2 = y'_3(x_3 + 2a_2) + y'_4(x_4 + 2c_2) + q_2(a_2 + 2c_2) = 14494.72$$

$$v_1 = x'_1(2y_1 - b_1) + x'_2(2y_2 - d_1) + p_1(d_1 - 2b_1) = -1723.34$$

$$v_2 = x'_3(2y_3 - b_2) + x'_4(2y_4 - d_2) + p_2(d_2 - 2b_2) = -12134.96$$

– At last, Alice obtains the secure share $u = u_1 + u_2 = 22569.6$, and Bob gets his private output $v = u_1 + u_2 = -13858.3$.

If we directly calculates the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$, it is $2.3 * (-19.5) + (-81.9) * (-78.1) + 96.7 * 39.2 + (-27.1) * 52.8 = 8711.3$ which is exactly equal to the sum of $u = 22569.6$ and $v = -13858.3$. It shows the above steps are correct.

2.2 Correctness Analysis

To confirm the correctness of EDSPP, we need to consider,

Theorem 1. *After performing* EDSPP*, Alice's private output u and Bob's secret output v meet $u + v = \boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{2k} x_i y_i$. That is,* EDSPP *is correct.*

Proof. In step 1.1 of EDSPP, there are $x'_{2j-1} = x_{2j-1} + a_j$, $x'_{2j} = x_{2j} + c_j$, $p_j = a_j + c_j$, $y'_{2j-1} = b_j - y_{2j-1}$, $y'_{2j} = d_j - y_{2j}$ and $q_j = b_j - d_j$. Then,

$$x'_{2j-1}(2y_{2j-1} - b_j) = 2x_{2j-1}y_{2j-1} - b_j x_{2j-1} + 2a_j y_{2j-1} - a_j b_j,$$

$$x'_{2j}(2y_{2j} - d_j) = 2x_{2j}y_{2j} - d_j x_{2j} + 2c_j y_{2j} - c_j d_j,$$

$$p_j(d_j - 2b_j) = a_j d_j - 2a_j b_j + c_j d_j - 2b_j c_j,$$

$$y'_{2j-1}(x_{2j-1} + 2a_j) = b_j x_{2j-1} + 2a_j b_j - x_{2j-1}y_{2j-1} - 2a_j y_{2j-1},$$

$$y'_{2j}(x_{2j} + 2c_j) = d_j x_{2j-1} + 2c_j d_j - x_{2j}y_{2j} - 2c_j y_{2j},$$

$$q_j(a_j + 2c_j) = a_j b_j + 2b_j c_j - a_j d_j - 2c_j d_j.$$

According to step 1.2, we have $u_j = y'_{2j-1}(x_{2j-1} + 2a_j) + y'_{2j}(x_{2j} + 2c_j) + q_j(a_j + 2c_j)$ and $v_j = x'_{2j-1}(2y_{2j-1} - b_j) + x'_{2j}(2y_{2j} - d_j) + p_j(d_j - 2b_j)$. Thus,

$$u_j + v_j = x_{2j-1}y_{2j-1} + x_{2j}y_{2j}. \tag{1}$$

There are $u = \sum_{j=1}^{k} u_j$ and $v = \sum_{j=1}^{k} v_j$ in step 2, then, $u + v = \sum_{j=1}^{k}(u_j + v_j) = \sum_{j=1}^{k}(x_{2j-1}y_{2j-1} + x_{2j}y_{2j})$. Therefore,

$$u + v = \sum_{i=1}^{2k} x_i y_i \tag{2}$$

That is, $u + v = \boldsymbol{x} \cdot \boldsymbol{y}$ holds at the end of EDSPP, which completes the proof.

2.3 Security Analysis

In this subsection, we will analysis the security of EDSPP under semi-honest model [17], where each participant correctly follow the protocol while trying to find out potentially confidential information from his legal medium records. Generally, we consider the view of each participant in this protocol and whether some privacy can be deduced from the view.

During the execution of EDSPP, Alice receives y'_{2j-1}, y'_{2j} and q_j, symmetrically, Bob learns x'_{2j-1}, x'_{2j} and p_j.

From y'_{2j-1} and y'_{2j}, Alice cannot learn any information about y_{2j-1} and y_{2j}. While q_j is known to her, the sum of $-y_{2j-1}$ and y_{2j} will be derived by $y_{2j} - y_{2j-1} = y'_{2j-1} - y'_{2j} - q_j$, however, Bob's private numbers y_{2j-1} and y_{2j} are still unrevealed. Analogously, Bob can figure out $x_{2j-1} + x_{2j} = x'_{2j-1} + x'_{2j} - p_j$, while he cannot obtain any more information about Alice's privacy x_{2j-1} and x_{2j}. Therefore, each real element of the private vectors of both participants is not disclosed in EDSPP. If the elements of the vectors are 0 or 1, EDSPP is not secure. GLLM-SPP [14] is more fit for securely computing the scalar product of binary vectors.

Quantification of Disclosure Level. Here, we give the quantification of disclosure level about Alice's private data x_{2j-1} and x_{2j}. While EDSPP has been applied, if $T = x'_{2j-1} + x'_{2j} - p_j$, then, Bob learns that (x_{2j-1}, x_{2j}) is randomly located at the line $T = x_{2j-1} + x_{2j}$, the slope of which is exactly equal to -1.

(1) While $x_{2j-1}, x_{2j} \in \mathbb{R}$, that is, before EDSPP being applied, according to Bob's view, (x_{2j-1}, x_{2j}) is randomly located at two-dimensional real space $\mathbb{R}^2$. After EDSPP, the distribution space of (x_{2j-1}, x_{2j}) is reduced to a line. However, as both x_{2j-1} and x_{2j} are random in Bob's view, then, he cannot extract the original private numbers x_{2j-1} and x_{2j} from their sum $T = x'_{2j-1} + x'_{2j} - p_j$.

(2) While $L \leqslant x_{2j-1}, x_{2j} \leqslant U$ $(L < U)$, then, before EDSPP, (x_{2j-1}, x_{2j}) is randomly located at a $(U-L) \times (U-L)$-square area in Bob's view. At the end of EDSPP, Bob can figure out $T = x'_{2j-1} + x'_{2j} - p_j$ which is equal to $x_{2j-1} + x_{2j}$. Furthermore, $x_{2j-1} = T - x_{2j}$ and $x_{2j} = T - x_{2j-1}$, thus, Bob knows $T - U \leqslant x_{2j-1}, x_{2j} \leqslant T - L$. Then, he obtains

$$\max\{L, T-U\} \leqslant x_{2j-1}, x_{2j} \leqslant \min\{U, T-L\}.$$

According to the range of x_{2j-1} and x_{2j}, it is easy to get $2L \leqslant T \leqslant 2U$.

If $2L \leqslant T < L + U$, then, $\max\{L, T-U\} = L$ and $\min\{U, T-L\} = T - L$. Therefore, Bob can find out $L \leqslant x_{2j-1}, x_{2j} \leqslant T - L$.

If $L + U \leqslant T \leqslant 2U$, then, $\max\{L, T-U\} = T - U$ and $\min\{U, T-L\} = U$. In Bob's view, there will be $T - U \leqslant x_{2j-1}, x_{2j} \leqslant U$.

In this situation, Bob can obtain a more narrow range about x_{2j-1} and x_{2j}, but he cannot exactly deduce the value of them except the following two extreme cases: $x_{2j-1} = x_{2j} = L, T = 2L$ and $x_{2j-1} = x_{2j} = U, T = 2U$.

In general, the new scheme sacrifices some security in a certain level, but the private raw data is still protected especially when the elements of the private

vectors are real number. Alice and Bob disclose nothing but the sum $x_{2j-1}+x_{2j}$, $y_{2j-1}+y_{2j}$ to each other in EDSPP. Besides, two participants carry out symmetric computations, send and receive symmetrical data, consequently, EDSPP is quite fair.

2.4 Communication Overheads and Computational Complexity

The following contributes to the computational cost: (1) In step 1.1 of EDSPP, Alice and Bob respectively generate two random number and perform three additions. In step 1.2, each party performs three multiplications and two additions. All the above operations loop for k times. (2) In step 2, they each carry out $k-1$ additions.

Therefore, the computational complexity of EDSPP is $O(n)$ in total. Here, n is the dimension number of their private vectors and $n = 2k$ in the protocol.

The transmitting data contains $x'_{2j-1}, x'_{2j}, p_j, y'_{2j-1}, y'_{2j}$ and q_j $(j = 1, 2, \cdots, k)$ in EDSPP. Thus, the total communication overheads are $3nb_0$ bits $(n = 2k)$. Here, b_0 is the bit length of a message.

2.5 A Simple Application Case

In many privacy-preserving distributed computations [18, 19], a key step is to securely find out which one of the points holden by one party is nearest to another point of the other participant. For simplicity, we deal with the problem that Alice has two private points $\mathrm{P}_1(P_{11}, P_{12}, \cdots, P_{1d})$ and $\mathrm{P}_2(P_{21}, P_{22}, \cdots, P_{2d})$, and Bob privately holds another point $\mathrm{Q}(Q_1, Q_2, \cdots, Q_d)$. They want to find out which one of P_1 and P_2 is closer to Q without disclosing the private coordinates of each point to each other or anybody else. Here, we use the scalar product of the coordinates as the distance of two points, that is, $|\mathrm{P}_i\mathrm{Q}| = \sum_{j=1}^{d} P_{ij}Q_j$ $(i = 1, 2)$. In fact, comparison of distances measured by other metrics, such as Euclidean distance and consine similarity, can be easily transferred into comparison of the dot products. Based on EDSPP, we present a simple but efficient solution for the above problem.

– Alice locally generates a random positive real numbers α and d random real numbers $r_1, r_2, \cdots, r_d$. Then, she sets the $2d$-dimensional vectors

$$\boldsymbol{P}'_i = (\alpha P_{i1}, r_1, \alpha P_{i2}, r_2, \cdots, \alpha P_{id}, r_d), \quad (i = 1, 2).$$

Bob randomly generates a random positive real numbers β and d random real numbers $R_1, R_2, \cdots, R_d$, and computes his private $2d$-dimensional vector by the following way

$$\boldsymbol{Q}' = (\beta Q_1, R_1, \beta Q_2, R_2, \cdots, \beta Q_d, R_d).$$

– Alice and Bob collaboratively perform EDSPP such that Alice obtains U_1, U_2 and Bob gets his private outputs V_1, V_2 which meet $U_i + V_i = \boldsymbol{P}'_i \cdot \boldsymbol{Q}'$ $(i = 1, 2)$.

– At last, Alice sends $\delta = U_1 - U_2$ to Bob. Then Bob computes $\Delta = \delta + V_1 - V_2$ and finds out the closer one by comparing Δ with 0.

In the above scheme, we can obtain

$$\Delta = (U_1 + V_1) - (U_2 + V_2) = \boldsymbol{P}_1' \cdot \boldsymbol{Q}' - \boldsymbol{P}_2' \cdot \boldsymbol{Q}' = \alpha\beta(|\mathrm{P_1Q}| - |\mathrm{P_2Q}|).$$

Thus, if $\Delta > 0$, P_2 is closer to Q; otherwise, P_1 is closer to Q.

Table 1. Comparison between EDSPP and Existing Schemes

Protocols	Computational Complexity	Employ Third Party?	Security	Fairness
GLLM-SPP [14]	$O(n * \mathcal{H})^\star$	No	CR-sec**	Very Bad
AE-SPP [15]	$O(n * \mathcal{H})^\star$	No	CR-sec**	Good
SPP-RIM [12]	$O(n^2)$	No	L-dis**	Bad
ATSPP [13]	$O(n^2)$	No	L-dis**	Good
SPP-CS [12]	$O(n)$	Yes	IT-sec**	Good
PBSPP [16]	$O(n)$	Yes	IT-sec**	Good
EDSPP	$O(n)$	No	L-dis**	Good

* Suppose the computational complexity of an encryption by homomorphic cryptosystem is $O(\mathcal{H})$. n is the dimension of private vectors.
** Here, IT-sec denotes "information-theoretically secure", CR-sec denotes "the security based on the intractability of the composite residuosity class problem", and L-dis denotes that the scheme will result in limited disclosure about private information of participants. SPP-CS and PBSPP are vulnerable to collusion attacks, though the schemes have the security based on information theory.

3 Performance Comparison and Experiment Results

The communication overheads of EDSPP and each previous scheme are $O(n)$, to demonstrate the special features of EDSPP, we compare it with six most frequently-used schemes (to the best of our knowledge) in table 1. It indicates that EDSPP has the best performance in many aspects except for the security. SPP-CS [12] and PBSPP [16] have the same linear computational complexity as EDSPP, but SPP-CS and PBSPP employ one or more semi-trusted third parties, which results in that they are extremely vulnerable to unavoidable potential collusion attacks. While the third party colludes with one party, the other participant's privacy will be seriously breached. The computational complexity of SPP-RIM [12] and ATSPP [13] are $O(n^2)$ which is bigger than that of EDSPP. GLLM-SPP [14] and AE-SPP [15] use the expensive homomorphic cryptosystem. Additionally, participants execute very similar operations in EDSPP, thus, the scheme has good fairness. In GLLM-SPP [14] the participant, who generates the homomorphic encryption system and encrypts each element of his private vector, will load much more computation and communication than the other one, thus the fairness of GLLM-SPP is very bad.

We implement three most computationally efficient schemes, SPP-CS, PBSPP and EDSPP. In the experiments, each participant is performed on a computer with Intel Core2 Duo 2.93GHz CPU and 2.0GB memory, and the average **ping** time of them is shorter than 1 ms. Figure 1 exhibits the simulated results, which indicates that all the runtime linearly increase with dimension and EDSPP costs least time. While the vectors' dimension are 200 ($k = 100$), the total running time of EDSPP is only a little more than 100 ms which is less than one-third of that of PBSPP and is about one-sixth of the running time cost by SPP-CS.

In summary, the comparative advantages of EDSPP are its simpleness, linear efficiency, good fairness and it does not employ the expensive homomorphic cryptosystem and any auxiliary third party. As ideal security is too expensive to achieve, especially in large-scale systems, and it may be unnecessary in practice, if disclosing partial information about private data is still acceptable, EDSPP will be a competitive low-cost candidate secure primitive for privacy preserving distributed collaborative computations.

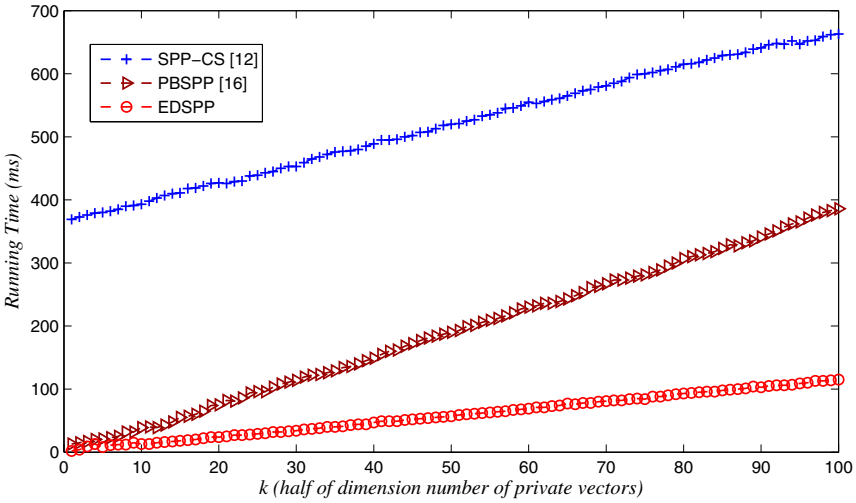

Fig. 1. Running Time of SPP-CS [12], PBSPP [16] and EDSPP ($ms = 10^{-3}s$, the private vectors' dimension $n = 2k$)

4 Conclusion

In this paper, a linearly efficient scheme for scalar product protocol, EDSPP, has been proposed. The protocol has no use of expensive homomorphic crypto-system and third party, which have been employed by existing solutions. Theoretical analysis and simulated experiment results confirm that the novel scheme is a competitive candidate for securely computing the scalar product of two private vectors.

Acknowledgement. This work was supported by the Japan Society for the Promotion of Science fellowship (No. P12045) , the National Natural Science Foundation of China (Nos. 60903217 and 60773032) and the Natural Science Foundation of Jiangsu Province of China (No. BK2010255).

References

1. HIPAA. The health insurance portability and accountability act of 1996 (October 1998), `http://www.ocius.biz/hipaa.html`
2. Cios, K.J., Moore, G.W.: Uniqueness of medical data mining. Artificial Intelligence in Medicine 26(1-2), 1–24 (2002)
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. ACM Sigmod Record 29, 439–450 (2000)
4. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality 1(1), 59–98 (2009)
5. Yang, B., Nakagawa, H., Sato, I., Sakuma, J.: Collusion-resistant privacy-preserving data mining. In: The 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 483–492 (2010)
6. Chen, T., Zhong, S.: Privacy-preserving backpropagation neural network learning. IEEE Transactions on Neural Networks 20(10), 1554–1564 (2009)
7. Bansal, A., Chen, T., Zhong, S.: Privacy preserving Back-propagation neural network learning over arbitrarily partitioned data. Neural Computing and Applications 20(1), 143–150 (2011)
8. Murugesan, M., Jiang, W., Clifton, C., Si, L., Vaidya, J.: Efficient privacy-preserving similar document detection. The VLDB Journal 19(4), 457–475 (2010)
9. Xiao, M., Huang, L., Xu, H., Wang, Y., Pei, Z.: Privacy Preserving Hop-distance Computation in Wireless Sensor Networks. Chinese Journal of Electronics 19(1), 191–194 (2010)
10. Zhu, Y., Huang, L., Dong, L., Yang, W.: Privacy-preserving Text Information Hiding Detecting Algorithm. Journal of Electronics and Information Technology 33(2), 278–283 (2011)
11. Smaragdis, P., Shashanka, M.: A framework for secure speech recognition. IEEE Transactions on Audio, Speech, and Language Processing 15(4), 1404–1413 (2007)
12. Du, W., Zhan, Z.: A practical approach to solve secure multi-party computation problems. In: The 2002 Workshop on New Security Paradigms, pp. 127–135. ACM, New York (2002)
13. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 639–644. ACM, New York (2002)
14. 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)
15. Amirbekyan, A., Estivill-Castro, V.: A new efficient privacy-preserving scalar product protocol. In: The Sixth Australasian Conference on Data Mining and Analytics, vol. 70, pp. 209–214. Australian Computer Society (2007)

16. Shaneck, M., Kim, Y.: Efficient Cryptographic Primitives for Private Data Mining. In: The 2010 43rd Hawaii International Conference on System Sciences, pp. 1–9. IEEE Computer Society (2010)
17. Goldreich, O.: Foundations of Cryptography: Volume II, Basic Applications. Cambridge University Press, Cambridge (2004)
18. Qi, Y., Atallah, M.J.: Efficient privacy-preserving k-nearest neighbor search. In: The 28th International Conference on Distributed Computing Systems (ICDCS 2008), pp. 311–319. IEEE (2008)
19. Shaneck, M., Kim, Y., Kumar, V.: Privacy preserving nearest neighbor search. In: 6th IEEE International Conference on Data Mining Workshops, pp. 541–545. IEEE (2006)