Efficient scalar product protocol and its privacy-preserving application

Youwen Zhu*

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China
E-mail: zhuyouwen@gmail.com
E-mail: zhuyw@nuaa.edu.cn
*Corresponding author

Tsuyoshi Takagi

Institute of Mathematics for Industry, Kyushu University, Fukuoka, 819-0395, Japan
E-mail: takagi@imi.kyushu-u.ac.jp

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 crypto-system and any 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 the most efficient one, and it has good fairness. Simulated experiment results intuitively indicate the good performance of our 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 discuss the application of EDSPP, and present a secure distance comparison protocol based on EDSPP, which can be used in many privacy-preserving computations, such as privacy-preserving k-nearest neighbours computation. Additionally, a hybrid scheme is put forward to securely compute the scalar product of arbitrary-length private vectors.

Keywords: privacy preserving; distributed computation; application; 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 organisations could cooperate with each other to conduct computations on the union of data they each hold. Unfortunately, in spite of potential enormous benefits of the collaboration, it has been obstructed by security and privacy concerns. For example, a single hospital might not have enough cases to analyse some special symptoms and several hospitals need to cooperate with each other to study their joint database of case samples for the comprehensive analysis 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 (HIPAA, 1998; Cios and Moore, 2002). Consequently, some techniques (Agrawal and Srikant, 2000; Lindell and Pinkas, 2009) for privacy preserving distributed collaborative computations were introduced to address the concerns by privacy advocates. Nowadays, a large amount of attention (Yang et al., 2010; Chen and Zhong, 2009; Bansal et al., 2011) has been paid to dealing with the challenges of how to extract beneficial comprehensive information from distributed datasets 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. (2010) 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
Efficient scalar product protocol and its privacy-preserving application

protocol in wireless sensor networks is introduced in Xiao et al. (2010) 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 Chen and Zhong (2009), Bansal et al. (2011), Zhu et al. (2011), Smaragdis and Shashanka (2007), Qi and Atallah (2008), Shaneck et al. (2006), Li et al. (2012) and Dong et al. (2011) for more concrete applications of scalar product protocol.

As secure computation for the dot product of private vectors is fundamental for many privacy preserving distributed computing tasks, several schemes (Du and Zhan, 2002; Vaidya and Clifton, 2002; Goethals et al., 2004; Amirbekyan and Estivill-Castro, 2007; Shaneck and Kim, 2010) have been proposed to perform the secure computation. Du and Zhan presented two practical schemes in Du and Zhan (2002): 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 Vaidya and Clifton (2002) (denoted as ATSPP). Based on homomorphic encryption, two solutions for securely computing dot product of private vectors are given in Goethals et al. (2004) (denoted as GLLM-SPP) and Amirbekyan and Estivill-Castro (2007) (denoted as AE-SPP) respectively. A polynomial secret sharing-based scalar product protocol (denoted as PBSPP) was lately presented by Shaneck and Kim (2010). 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, i.e., the commodity server in SPP-CS and one or more non-collaborative third parties in PBSPP. The protocols will be vulnerable to unavoidable potential collusion attacks while employing the semi-trusted third parties. GLLM-SPP and AE-SPP encrypt all 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. Generally speaking, previous schemes of scalar product protocol are still far from being practical in most situations.

Nowadays, several works (Dreier and Kerschbaum, 2011; Wang et al., 2011; Chida and Takahashi, 2008) have devoted to the practicability of various secure schemes such that their modified solutions run very fast though it may bring about limited but acceptable leakage about private input. In this paper, we focus on the useful secure primitive, scalar product protocol (Du and Zhan, 2002), and propose a simple and linearly efficient protocol for securely computing the scalar product of two private vectors, even-dimension scalar product protocol (EDSPP). At the end of our scheme, each participant will obtain a private output which is a share of the scalar product of their private vectors, and the sum of the two private output numbers exactly equals to the scalar product. Besides, 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 (Du and Zhan, 2002; Vaidya and
Clifton, 2002; Goethals et al., 2004; Amirbekyan and Estivill-Castro, 2007; Shaneck and Kim, 2010), our protocol is also under semi-honest model (Goldreich, 2004), 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 organised as follows. Section 2 gives problem definition and notations in this paper. Section 3 proposes the new solution for scalar product protocol: EDSPP, 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 4. Then, Section 5 discusses the application of our new scheme, and puts forward a secure distance comparison protocol (SDCP) based on EDSPP. We construct a hybrid scheme of generic scalar product protocol (HybridSPP) to securely and efficiently compute the scalar product of arbitrary-length private vectors in Section 6. At last, Section 7 concludes the paper.

2 Problem definition and notations

2.1 Problem definition

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

In this paper, we will deal with the scalar product protocol while $n \in \mathbb{Z}^+$ is even and propose a practical scheme, EDSPP, with high efficiency. Without loss of generality, we suppose $n = 2k$ ($k \in \mathbb{Z}^+$). Then, the problem can be formalised as follows:

- **Input:**
  - Alice: $x = (x_1, x_2, \ldots, x_{2k}) \in \mathbb{R}^{2k}$,
  - Bob: $y = (y_1, y_2, \ldots, y_{2k}) \in \mathbb{R}^{2k}$.
  - Here, $k \in \mathbb{Z}^+$.

- **Output and requires:** Alice obtains $u \in \mathbb{R}$, and Bob gets $v \in \mathbb{R}$ such that
  
  $$u + v = x \cdot y = \sum_{i=1}^{2k} x_i y_i.$$

Additionally, the private inputs, vectors $x$ and $y$, should be kept private to its owner, respectively, throughout the protocol.

Based on EDSPP and the existing GLLM-SPP (Goethals et al., 2004), we also present a hybrid scheme to securely and efficiently compute the scalar product of arbitrary-length private vectors in Section 6.
2.2 Notations

For the convenience of reading, we list the notations this paper uses in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice, Bob</td>
<td>Two participants in scalar product protocol</td>
</tr>
<tr>
<td>$x$</td>
<td>The private vector of Alice</td>
</tr>
<tr>
<td>$y$</td>
<td>The private vector of Bob</td>
</tr>
<tr>
<td>$n$</td>
<td>The dimension of private vectors</td>
</tr>
<tr>
<td>$k$</td>
<td>A positive integer, $n = 2k$</td>
</tr>
<tr>
<td>$[X]$</td>
<td>The set ${1, 2, \cdots, X}$, for any positive integer $X$</td>
</tr>
<tr>
<td>$x_i$</td>
<td>The $i$th dimension of $x$, $i \in [n]$</td>
</tr>
<tr>
<td>$y_i$</td>
<td>The $i$th dimension of $y$, $i \in [n]$</td>
</tr>
<tr>
<td>$x'_i$</td>
<td>Perturbed $x_i$, $i \in [n]$</td>
</tr>
<tr>
<td>$y'_i$</td>
<td>Perturbed $y_i$, $i \in [n]$</td>
</tr>
<tr>
<td>$u$</td>
<td>The private output of Alice</td>
</tr>
<tr>
<td>$v$</td>
<td>The private output of Bob</td>
</tr>
<tr>
<td>$u_j$</td>
<td>The private intermediate data of Alice, $j \in [k]$</td>
</tr>
<tr>
<td>$v_j$</td>
<td>The private intermediate data of Bob, $j \in [k]$</td>
</tr>
<tr>
<td>$a_j, c_j$</td>
<td>The private random numbers of Alice, $j \in [k]$</td>
</tr>
<tr>
<td>$b_j, d_j$</td>
<td>The private random numbers data of Bob, $j \in [k]$</td>
</tr>
<tr>
<td>$\max{X_1, X_2}$</td>
<td>The bigger one of $X_1$ and $X_2$</td>
</tr>
<tr>
<td>$\min{X_1, X_2}$</td>
<td>The smaller one of $X_1$ and $X_2$</td>
</tr>
<tr>
<td>$[X_1, X_2]$</td>
<td>The real number set ${X \in \mathbb{R}</td>
</tr>
</tbody>
</table>

3 Even-dimension scalar product protocol

In this section, we will present our novel scheme, followed by the analysis of its correctness, security, communication overheads and computation complexity.

3.1 The scheme

As a secure primitive, scalar product protocol (Du and Zhan, 2002; Goethals et al., 2004) has extensive privacy preserving applications and an efficient scalar product protocol will boost the practical process of privacy preserving distributed cooperative computations. Here, we consider a special case where the dimension of private vectors $n$ is an even number (that is, $n = 2k$, $k$ is a positive integer). Then, at the expense of disclosing partial sum of some private data, we propose an efficient EDSPP.

In our scheme, to achieve a practical performance, the private data is only hidden by stochastic transformation, and after interchanging their perturbed data, each participant can obtain 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 randomisation. Then, they interchange the perturbed numbers, 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, the scheme can only compute the dot product of even-dimension vectors.

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

- Alice generates random numbers: \( a_1 = -53.0 \) and \( c_1 = 99.8 \) for the first two dimensions of \( x \). 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 \( y \). 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 \( \{p_1, y_1', y_2'\} \) to Alice.

- Analogously, for the latter two dimensions, Alice and Bob locally 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.

- Then, Alice and Bob independently computes \( \{u_1, u_2\} \) and \( \{v_1, v_2\} \), respectively, by the following way.
  \[ u_1 = y_1'(x_1 + 2a_1) + y_2'(x_2 + 2c_1) + q_1(a_1 + 2c_1) = 8,074.88, \]
  \[ u_2 = y_3'(x_3 + 2a_2) + y_4'(x_4 + 2c_2) + q_2(a_2 + 2c_2) = 14,494.72, \]
  \[ v_1 = x_1'(2y_1 - b_1) + x_2'(2y_2 - d_1) + p_1(d_1 - 2b_1) = -1,723.34, \]
  \[ v_2 = x_3'(2y_3 - b_2) + x_4'(2y_4 - d_2) + p_2(d_2 - 2b_2) = -12,134.96. \]

- At last, Alice obtains the secure share \( u = u_1 + u_2 = 22,569.6, \)
and Bob gets his private output
\[ v = v_1 + v_2 = -13,858.3. \]

If we directly calculates the dot product of \( x \) and \( y \), it is
\[
2.3 \times (-19.5) + (-81.9) \times (-78.1) + 96.7 \times 39.2 + (-27.1) \times 52.8 = 8,711.3.
\]
which is exactly equal to the sum of \( u = 22,569.6 \) and \( v = -13,858.3 \). It empirically shows EDSPP is correct.

**Protocol 1**  Even-dimension scalar product protocol

**Require:** Alice has a private \( 2k \)-dimension vector \( x = (x_1, x_2, \ldots, x_{2k}) \) and Bob holds another confidential \( 2k \)-dimension vector \( y = (y_1, y_2, \ldots, y_{2k}) \). \((k \in \mathbb{Z}_+, \text{ for any } i \in [2k], x_i, y_i \in \mathbb{R})\)

**Ensure:** Alice obtains private output \( u \) and Bob securely gets \( v \) which meet
\[
2^1 \sum_{i=1}^{2k} x_i y_i = u + v = x \cdot y.
\]

1: **Step 1:**
2: \( \text{for } j = 1 \text{ to } k \text{ do} \)
3: \( \text{Step 1.1:} \) Alice locally generates two random real numbers \( a_j \) and \( c_j \) such that \( a_j + c_j \neq 0 \). Then, she computes \( p_j = a_j + c_j, \quad x'_j = x_{j-1} + a_j \) and \( x'_j = x_j + c_j \), and sends \( \{p_j, x'_j, x_j\} \) to Bob.

Simultaneously, Bob randomly generates two real numbers \( b_j \) and \( d_j \) which meet \( b_j - d_j \neq 0 \), and computes \( q_j = b_j + d_j, \quad y'_j = y_{j-1} = b_j - y_{j-1} \) and \( y'_j = d_j - y_j \). Then, he securely sends \( \{q_j, y'_j, y_j\} \) to Alice.

4: **Step 1.2:** Alice locally calculates
\[
u_j = x'_j (2y_{j-1} - b_j) + x'_j (2y_j - d_j) + p_j (d_j - 2b_j).
\]
and Bob, by himself, computes
\[
v_j = y'_j (2x_{j-1} + a_j) + y'_j (x_j + 2c_j) + q_j (a_j + 2c_j).
\]

5: **end for**

6: **Step 2:** Alice obtains \( u = \sum_{j=1}^{k} u_j \), and Bob gets \( v = \sum_{j=1}^{k} v_j \).

### 3.2 Correctness analysis

To strictly guarantee 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 \( u + v = x \cdot y = \sum_{i=1}^{2k} x_i y_i \). That is, EDSPP is correct.

**Proof:** In step 1.1 of EDSPP, there are \( x'_j = x_{j-1} + a_j, \quad x'_j = x_j + c_j, \quad p_j = a_j + c_j, \quad y'_j = b_j - y_j \) and \( q_j = b_j - d_j \). Then,
According to step 1.2, we have
\[ x_j = 2y_{j-1} - b_j \]
and
\[ y_j = 2y_{j-1} - d_j \]
Thus,
\[ x_j = 2^{j-1}\text{y}_1 + 2^{j-2}\text{y}_2 + \ldots + 2\text{y}_j + 2\text{y}_{j+1} - a_j b_j, \]
\[ y_j = 2^{j-1}\text{y}_1 + 2^{j-2}\text{y}_2 + \ldots + 2\text{y}_j + 2\text{y}_{j+1} - a_j d_j. \]

There are \( u = \sum_{j=1}^{k} u_j \) and \( v = \sum_{j=1}^{k} v_j \) in step 2, then,
\[ u + v = \sum_{j=1}^{k} (u_j + v_j) = \sum_{j=1}^{k} (x_{j-1}y_{j-1} + x_jy_j). \]
Therefore,
\[ u + v = \sum_{i=1}^{2k} x_iy_i \]
That is, \( u + v = x \cdot y \) holds at the end of EDSPP, which completes the proof.

### 3.3 Security analysis

In this subsection, we will analysis the security of EDSPP under semi-honest model (Goldreich, 2004), 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 discuss the potential disclosure which can be deduced from the view.

During the execution of EDSPP, Alice receives \( y_{j-1}^2, y_j^2 \) and \( q_j \), symmetrically, Bob learns \( x_{j-1}^2, x_j^2 \) and \( p_j \).

From \( y_{j-1}^2 \) and \( y_j^2 \), 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 \( y_{2j} - y_{2j-1} = y_j^2 - y_{j-1}^2 - q_j \), however, Bob’s private numbers \( y_{2j-1} \) and \( y_{2j} \) are still unrevealed. Analogously, Bob can figure out \( x_{2j-1} + x_{2j} = x_{j-1}^2 + x_j^2 - p_j \), 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 (Goethals et al., 2004) 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 \( x_{2j} \) (the degree of disclosure about Bob’s privacy can
be similarly analysed, thus, we do not give the discussion about it any more). While EDSPP has been applied, if \( T = x_{2j-1}^j + x_{2j}^j - p_j \), then, Bob learns that \((x_{2j-1}, x_{2j})\) is randomly located at the line \( T = x_{2j-1}^j + x_{2j}^j \), 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 one-dimension 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 \( T = x_{2j-1}^j + x_{2j}^j - p_j \).

2. While \( L \leq x_{2j-1}, x_{2j} \leq U (L < U) \), 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 \( T = x_{2j-1}^j + x_{2j}^j - p_j \), which is equal to the sum \( x_{2j-1}^j + x_{2j}^j \). Furthermore, \( 2L \leq T \leq 2U, x_{2j-1} = T - x_{2j} \) and \( x_{2j} = T - x_{2j-1} \), thus, Bob knows \( T - U \leq x_{2j-1}, x_{2j} \leq T - L \). Then, he obtains

\[
\max \{ L, T - U \} \leq x_{2j-1}, x_{2j} \leq \min \{ U, T - L \}, (2L \leq T \leq 2U).
\]

As the result, according to Bob’s view, before EDSPP, the length of distribution interval of Alice’s private number is \((U - L)\), after performing the protocol, the length of that interval can be narrowed down to \((\min\{ U, T - L \} - \max\{ L, T - U \})\) by Bob. Here, we use

\[
S_{Alice} = \frac{\min\{ U, T - L \} - \max\{ L, T - U \}}{U - L} \times 100\% \quad (3)
\]

to quantificationally scale the disclosure about Alice’s private data \( x_{2j-1} \) and \( x_{2j} \). As can be derived from the equation (3), \( S_{Alice} \) distributes in the real interval \([0, 1]\), and the bigger scale \( S_{Alice} \), the better privacy-preservation Alice received. At the extreme case \( S_{Alice} = 0 \)(that is, \( \min\{ U, T - L \} = \max\{ L, T - U \} \)), if \( 2L \leq T < L + U \), then, \( \max\{ L, T - U \} = L \) and \( \min\{ U, T - L \} = T - L \). Therefore, Bob can find out \( L \leq x_{2j-1}, x_{2j} \leq T - L \). Consequently, based on the equation (3), we have

\[
S_{Alice} = \frac{T - 2L}{U - L} \times 100\%.
\]

If \( L + U \leq T \leq 2U \), then, \( \max\{ L, T - U \} = T - U \) and \( \min\{ U, T - L \} = U \). In Bob’s view, there will be \( T - U \leq x_{2j-1}, x_{2j} \leq U \). Thus,

\[
S_{Alice} = \frac{2U - T}{U - L} \times 100\%.
\]
While $T$ goes from $2L$ to $2U$ (that is, $(T - 2L)/(2U - 2L)$ goes from 0 to 1.0), the value of $S_{\text{Alice}}$ is visually demonstrated in Figure 1, where the x-axis denoted $(T - 2L)/(2U - 2L)$ instead of $T$, for the convenience of description.

**Figure 1** The value of the security scale of Alice’s private data: $S_{\text{Alice}}$

As we can see from Figure 1, there are two segmented linear relationships between $S_{\text{Alice}}$ and $(T - 2L)/(2U - 2L)$. Except for two endpoints (that is, $S_{\text{Alice}} = 0$) and the case $(T - 2L)/(2U - 2L) = 0.5$ (that is, $T = U + L, S_{\text{Alice}} = 1.0$), $S_{\text{Alice}}$ is bigger than 0 but smaller than 1.0, therefore, Bob can shrink the possible distribution range of Alice’s private data $x_{2j-1}$ and $x_{2j}$ but he cannot find out more information about Alice’s privacy, that is, $x_{2j-1}$ and $x_{2j}$ still cannot be figured out by Bob. Broadly speaking, 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 $x_{2j-1} = x_{2j} = U$, $T = 2U$.

As the continued security analysis of the example of Section 3.1, we present the following results about how Bob can shrink Alice’s private numbers. In that example, Alice has the private input: a 4-dimension vector $x = (x_1, x_2, x_3, x_4) = (2.3, -81.9, 96.7, -27.1)$. We assume that Bob, before performing EDSPP, knows each element of Alice’s vector is a random real number from $\{x_i \in \mathbb{R} | -100 \leq x_i \leq 100\}$, that is, $L = -100$ and $U = 100$. For Alice’s private inputs $x_1$ and $x_2$, Bob can learn the sum $T = x_1 + x_2 - p_1 = x_1 + x_2 = -79.6$ after EDSPP. Furthermore, $\max\{L, T - U\} = -100$ and $\min\{U, T - L\} = 20.4$. Therefore, Bob finds out $-100 \leq x_{2j-1}$, $x_{2j} \leq 20.4$, but $x_1$ and $x_2$ are still hidden in the narrow interval $[-100, 20.4]$. According to equation (3), there is

$$S_{\text{Alice}} = \frac{20.4 - (-100)}{100 - (-100)} \times 100\% = 60.2\%$$
Efficient scalar product protocol and its privacy-preserving application

in the example. It shows that from the disclosed sum \( T = x'_1 + x'_2 - p_1 = x_1 + x_2 \), Bob can shrink the value range of \( x_1 \) and \( x_2 \) by 39.8% (that is, 100% - \( S_{\text{Alice}} = 39.8\% \)).

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 to each participant.

3.4 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 \( n = 2^k \) in the protocol.

The transmitting data contains \( x_{2j-1}, x_{2j}, p_j, y_{2j-1}, y_{2j} \) and \( q_j \) (\( j \in [k] \)) in EDSPP. Thus, the total communication overheads are \( 3nb_0 \) bits \((n = 2^k)\). Here, \( b_0 \) is the bit length of each number.

4 Performance comparison and experiment results

In this section, to demonstrate the special features of EDSPP, we compare it with six most frequently-used schemes (to the best of our knowledge) shown in Table 2. It indicates that EDSPP has the excellent performance in many aspects except for the security. The communication overheads of each scheme are \( O(n) \), however, the cost of EDSPP is lower than that of most previous schemes. Concretely speaking, SPP-CS (Du and Zhan, 2002) and PBSPP (Shaneck and Kim, 2010) have the same linear computational complexity as EDSPP, but the communication cost of the two existed schemes is three to five times more than that of our solution, besides, SPP-CS and PBSPP employ one or more semi-trusted auxiliary third parties, which results in that they are extremely vulnerable to unavoidable potential collusion attacks. While the third party colludes with one participant, the other party’s privacy will be seriously breached. The computational complexity of SPP-RIM (Du and Zhan, 2002) and ATSSP (Vaidya and Clifton, 2002) are \( O(n^2) \) which is bigger than that of EDSPP. GLLM-SPP (Goethals et al., 2004) and AE-SPP (Amirbekyan and Estivill-Castro, 2007) use the expensive homomorphic cryptosystem, thus, they consume much more computation time. Additionally, the better the fairness for individual participant, the more attractive the solution. In the schemes SPPRIM (Du and Zhan, 2002), PBSPP (Shaneck and Kim, 2010) and EDSPP, the participants, who provide private inputs, execute almost exactly the same operations and the private data of each party is hidden by the same secure manner, therefore, they have good fairness for each participant. While performing SPP-CS (Du and Zhan, 2002), ATSSP (Vaidya and Clifton, 2002) or AE-SPP
Y. Zhu and T. Takagi

(Amirbekyan and Estivill-Castro, 2007), as the steps Alice and Bob go and the methods their private inputs are preserved both are very similar but not identical, then their fairness is medium which is inferior to that of EDSPP. In GLLM-SPP (Goethals et al., 2004), the participant Alice, who generates the homomorphic encryption system and encrypts each element of her private vector, will load much more computation and communication than the other one, thus the fairness of GLLM-SPP is bad.

Table 2 Comparison between EDSPP and existing schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GLLM-SPP (Goethals et al., 2004)</td>
<td>$O(n * H)$</td>
<td>$O(n + H)$</td>
<td>-</td>
<td>$b_0(n + 1)$</td>
<td>CR-sec</td>
<td>No</td>
<td>Bad</td>
</tr>
<tr>
<td>AE-SPP (Amirbekyan and Estivill-Castro, 2007)</td>
<td>$O(n * H)$</td>
<td>$O(n * H)$</td>
<td>-</td>
<td>$2b_0n$</td>
<td>CR-sec</td>
<td>No</td>
<td>Medium</td>
</tr>
<tr>
<td>SPP-RIM (Du and Zhan, 2002)</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>-</td>
<td>$b_0n$</td>
<td>L-dis</td>
<td>No</td>
<td>Good</td>
</tr>
<tr>
<td>ATSP (Vaidya and Clifton, 2002)</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>-</td>
<td>$b_0(2n + 3)$</td>
<td>L-dis</td>
<td>No</td>
<td>Medium</td>
</tr>
<tr>
<td>SPP-CS (Du and Zhan, 2002)</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$7b_0n$</td>
<td>IT-sec</td>
<td>Yes</td>
<td>Medium</td>
</tr>
<tr>
<td>PBSPP (Shaneck and Kim, 2010)</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$10b_0n$</td>
<td>IT-sec</td>
<td>Yes</td>
<td>Good</td>
</tr>
<tr>
<td>EDSPP</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>-</td>
<td>$3b_0n$</td>
<td>L-dis</td>
<td>No</td>
<td>Good</td>
</tr>
</tbody>
</table>

Notes: Let $b_0$ be the bit length of each data, and ATP denote the ‘auxiliary third party’. $n$ is the dimension of private vectors. Suppose the computational complexity of an encryption by homomorphic cryptosystem is $O(H)$. 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. Additionally, the computation complexity and communication cost of PBSPP, shown in the table, is evaluated under the condition that only one auxiliary third party is employed; if it hires more third parties, the overheads of computation and communication will rapidly increase.

Furthermore, we implement three most computationally efficient schemes, SPP-CS (Du and Zhan, 2002), PBSPP (Shaneck and Kim, 2010) and our solution EDSPP. In the experiments, each participant is performed on a computer with Intel Core2 Duo 2.93 GHz CPU and 2.0 GB memory, and the average ping time of them is shorter than 1 ms. Figure 2 exhibits the simulated results, which indicates that all the runtime approximately linearly increase with dimension, and EDSPP costs much less time that the other two schemes. 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.

Figure 2  Running time of SPP-CS (Du and Zhan, 2002), PBSPP (Shaneck and Kim, 2010) and EDSPP \((ms = 10^{-3} s, \text{the private vectors' dimension } n = 2k)\)

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.

5 Application of EDSPP

We will discuss the application of EDSPP in this section.

Privacy preserving \(k\)-nearest neighbours (\(k\)NN) computation (Qi and Atallah, 2008; Shaneck et al., 2006) deals with the following situation: Alice has a private dataset \(P = \{p_1, p_2, \cdots, p_m\}\) in which \(p_i (i \in [m])\) is a \(d\)-dimension vector and \(p_i = (p_{i1}, p_{i2}, \cdots, p_{id}) \in \mathbb{R}^d\). Bob privately holds a query point \(q = (q_1, q_2, \cdots, q_d) \in \mathbb{R}^d\). They plan to securely return the index set of \(k\)NN of \(q\) from Alice’s private set \(P\). In privacy preserving \(k\)NN computation, the key step is to securely compare the distances between \(q\) with different vectors in \(P\). In this section, it will be shown that the distance comparison can be securely and efficiently performed by employing EDSPP.

There are several distance metrics, such as Euclidean distance, cosine similarity which is frequently used to evaluate the similarity of two documents in the area of information retrieval and text matching. We study the two types of metrics, and it shows
that the distance comparison, while adopting the metrics, can be reduced to the comparison of dot products. The details are as follows.

- **Euclidean distance.** For simplicity, we use the squared Euclidean distance,

\[ D^2(p_i, q) = \sum_{i=1}^{d} (p_{ii} - q_{i})^2, \]

as it does not alter the return of distance comparison. Then,

\[
D^2(p_i, q) - D^2(p_j, q) = \left( \sum_{i=1}^{d} p_{ii}^2 - \sum_{i=1}^{d} 2p_{ii}q_{i} \right) - \left( \sum_{i=1}^{d} p_{jj}^2 - \sum_{i=1}^{d} 2p_{ij}q_{i} \right).
\]

If we set the \((d + 1)\)-dimensional vectors \(q' = (q_1, \ldots, q_d, 1)\) and

\[ p_i' = (-2p_{1i}, \ldots, -2p_{di}, \sum_{i=1}^{d} p_{ii}^2), \]

then

\[ D^2(p_i, q) - D^2(p_j, q) = p_i' \cdot q' - p_j' \cdot q'. \]

Therefore, Alice and Bob can find out the closer one by comparing \(p_i' \cdot q'\) with \(p_j' \cdot q'\).

- **Cosine similarity.** The cosine similarity of \(p_i\) and \(q\) is

\[ \cos(p_i, q) = \frac{p_i \cdot q}{||p_i|| \cdot ||q||}. \]

Here, \(||p_i|| = \sqrt{\sum_{i=1}^{d} p_{ii}^2}\) and \(||q|| = \sqrt{\sum_{i=1}^{d} q_{i}^2}\). Let \(p_i' = \frac{p}{||p||}\) and \(q' = \frac{q}{||q||}\), then the scalar product \(p_i' \cdot q'\) equals to the cosine similarity of \(p_i\) and \(q\), and they could decide which one of \(p_i\) and \(p_j\) is more similar to \(q\) by judging the smaller one of \(p_i' \cdot q'\) and \(p_j' \cdot q'\).

As the distance comparison can be transformed into the comparison of two scalar products, thus, we present a SDCP using dot product as the distance between different vectors. The protocol is structured based on EDSPP and the highlight is that each original vector and a same length random vector crosswise mix into an even-dimension perturbed vector which is twice as long as the original one, then each participant obtains a share of the scalar product of their perturbed vectors by EDSPP, at last, the closer one can be found out through comparing the scalar product of their perturbed vectors. The details are shown in Protocol 2.

We discuss the correctness, security and overheads of Protocol 2 as follows.

**Correctness.** We present the correctness analysis of Protocol 2 by starting with a theorem.

**Theorem 2:** In Protocol 2, the following equations hold.
Efficient scalar product protocol and its privacy-preserving application

\[
\begin{align*}
\text{iff. } x_1 \cdot y > x_2 \cdot y, \\
\text{iff. } x_1 \cdot y = x_2 \cdot y, \\
\text{iff. } x_1 \cdot y < x_2 \cdot y. \\
\end{align*}
\]

\[
\left\{ \begin{array}{ll}
x'_1 \cdot y' > x'_2 \cdot y' & \text{iff. } x_1 \cdot y > x_2 \cdot y, \\
x'_1 \cdot y' = x'_2 \cdot y' & \text{iff. } x_1 \cdot y = x_2 \cdot y, \\
x'_1 \cdot y' < x'_2 \cdot y' & \text{iff. } x_1 \cdot y < x_2 \cdot y.
\end{array} \right.
\]

\text{Protocol 2} \quad \text{Secure distance comparison protocol}

\textbf{Require:} Alice has two private \(e\)-dimension vector \(x_1 = (x_{11}, x_{12}, \cdots , x_{1e})\) and \(x_2 = (x_{21}, x_{22}, \cdots , x_{2e})\), and Bob holds another confidential \(e\)-dimension vector \(y = (y_1, y_2, \cdots , y_e)\). \((x_{ij}, y_j) \in \mathbb{R})\)

\textbf{Ensure:} They securely find out the closer one of \(x_1\) and \(x_2\) to \(y\) according to the distances \(D(x_1, y) = x_1 \cdot y\) and \(D(x_2, y) = x_2 \cdot y\).

1: \textbf{Step 1:} Alice locally selects a random positive real numbers \(\alpha\) and \(e\) random real numbers \(r_1, r_2, \cdots , r_e\). Then, she sets the \(2e\)-dimensional vectors

\[
x'_1 = (\alpha x_{11}, r_1, \alpha x_{12}, r_2, \cdots , \alpha x_{1e}, r_e), \quad (i = 1, 2).
\]

Bob randomly generates a random positive real numbers \(\beta\) and \(e\) random real numbers \(R_1, R_2, \cdots , R_e\), and computes his private \(2e\)-dimensional vector by the following way

\[
y' = (\beta y_1, R_1, \beta y_2, R_2, \cdots , \beta y_e, R_e).
\]

2: \textbf{Step 2:} 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

\[
u_i + v_i = x'_i \cdot y' \quad (i = 1, 2).
\]

3: \textbf{Step 3:} 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 \(\Delta\) with 0.

Here, \textit{iff.} denotes ‘if and only if’.

\textbf{Proof:} According to the step 1 of Protocol 2, we have

\[
x'_1 \cdot y' - x'_2 \cdot y' = \sum_{i=1}^{e} \left( a\beta x_{1i} y_i + r_i R_i \right) - \sum_{i=1}^{e} \left( a\beta x_{2i} y_i + r_i R_i \right) \\
= \sum_{i=1}^{e} \left( a\beta x_{1i} y_i - a\beta x_{2i} y_i \right) \\
= a\beta \left( \sum_{i=1}^{e} x_{1i} y_i - \sum_{i=1}^{e} x_{2i} y_i \right) \\
= a\beta (x_1 \cdot y - x_2 \cdot y).
\]

Therefore, while \(\alpha > 0\) and \(\beta > 0\), the equation (4) holds. That is, Theorem 2 is correct.

In the steps 2 and 3 of Protocol 2, there is

\[
\Delta = \delta + v_1 - v_2 = u_1 - u_2 + v_1 - v_2 = (u_1 + v_1) - (u_2 + v_2) = x'_1 \cdot y' - x'_2 \cdot y'.
\]

Thus, based on Theorem 2, it can be achieved that if \(\Delta > 0\), then \(x'_1 \cdot y' > x'_2 \cdot y'\) which implies \(x_1\) is closer to \(y\); when \(\Delta = 0\), the distances \(D(x_1, y)\) and \(D(x_2, y)\) are the same; otherwise, \(\Delta < 0\), \(x_2\) is the closer one. Consequently, Protocol 2 can correctly complete the distance comparison.
Security. We first analyse the view of Bob to demonstrate the preservation of Alice’s private data. While performing Protocol 2, Bob can obtain $\alpha x_i + r_t (i \in [2], t \in [e])$ in the step 2 and receive $\delta$ in the step 3. It has been shown $\Delta = \delta + v_1 - v_2 = x'_i - x'_2 - y'_i - y'_2 = \alpha \beta (x_i - y_i - x_2 - y).$ As the randomness and secrecy of $\alpha$ and $r_t (t \in [e]),$ then, $x_2 (i \in [2], t \in [e])$ cannot be deduced by Bob.

To confirm the security of Bob’s privacy, the following theorem is farther proposed.

**Theorem 3:** Throughout Protocol 2, no private information of Bob is disclosed.

**Proof:** In the steps 1 and 3 of Protocol 2, Alice does not receive any data. Based on the security of EDSPP, for Bob’s any private input $y_i (t \in [e]),$ Alice can obtain nothing but $\beta y_i + R_t$ in the step 2. As the random numbers $\beta$ and $R_t$ are kept private to Bob, Alice cannot find out any information about $y_i.$ Therefore, no private information of Bob is disclosed, which confirms the correctness of Theorem 3.

In all, the above analysis confirms the privacy of Alice and Bob can be preserved, especially, Bob’s privacy suffers no disclosure during the execution of Protocol 2.

**Complexity of computation and communication.** The computation cost of step 2 is $O(e).$ In the steps 1 and 3, they generate $(2(e + 1))$ random numbers and perform $2e$ multiplications and three additions. In total, the computation complexity of Protocol 2 is $O(e).

There is no data exchange in the step 1. The communication complexity of steps 2 and 3 are $O(e)$ and $O(1),$ respectively. Thus, Protocol 2 totally consumes $O(e)$ in communication.

### 6 A HybridSPP

Based on EDSPP and GLLM-SPP (Goethals et al., 2004), a HybridSPP are constructed. If the private vectors’ dimension number are even, HybridSPP is the same as EDSPP. While the vectors are $(2k + 1)$-dimension $(k = 0, 1, \cdots),$ the full scheme is displayed as the following.

- Alice sets $a_i = (x_1, \cdots, x_{2k})$ and Bob locally sets $b_i = (y_1, \cdots, y_{2k}).$ Then, they collaboratively perform EDSPP such that Alice securely obtains $u_1$ and Bob privately gets $v_1$ which meet $u_1 + v_1 = a_i \cdot b_i = \sum_{t=1}^{2k} x_t y_t$ (if $k$ is equal to 0, Alice and Bob set $u_1 = 0$ and $v_1 = 0$)
- simultaneously, Alice and Bob collaboratively perform GLLM-SPP (Goethals et al., 2004) such that Alice and Bob respectively obtains private output $u_2$ and $v_2$ which meet $u_2 + v_2 = x_{2k+1} \cdot y_{2k+1}$
- finally, Alice gets the private output $u = u_1 + u_2$ and Bob receives $v = v_1 + v_2.$

Then, we briefly demonstrate the correctness, security and complexity of HybridSPP while the dimension of private vectors is odd.
Correctness. There are $u_1 + v_1 = \sum_{i=1}^{2k} x_i y_i$ and $u_2 + v_2 = x_{2k+1} \cdot y_{2k+1}$, then,

$$u_1 + v_1 + u_2 + v_2 = x_{2k+1} \cdot y_{2k+1} + \sum_{i=1}^{2k} x_i y_i = \sum_{i=1}^{2k+1} x_i y_i.$$ 

Therefore, $u + v = \sum_{i=1}^{2k+1} x_i y_i$ holds, and the hybrid scheme is correct.

Security. There are two subprotocols in HybridSPP and each invoked protocol is independent. The $x_{2k+1}$ and $y_{2k+1}$ are preserved by the homomorphic encryption system that GLLM-SPP (Goethals et al., 2004) invoked. Thus, HybridSPP holds the same security as EDSPP.

Complexity of computation and communication. It has been shown that EDSPP has linear complexity. While securely computing $u_2 + v_2 = x_{2k+1} \cdot y_{2k+1}$ by performing GLLM-SPP, Alice and Bob will respectively do one encryption and send a ciphertext to each other. If $O(\mathcal{H})$ is the computational complexity of an encryption by the homomorphic encryption system, then, the computational complexity of HybridSPP is $O(k + \mathcal{H})$ and its communication cost is $O(k)$, which close to the low overheads of EDSPP, especially when $k$ is large.

7 Conclusions

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 the 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. Then, based on EDSPP, we present a SDCP which can be used in many privacy-preserving applications, such as privacy-preserving $k$NN computation. At last, a HybridSPP is put forward such that we can securely compute the dot product of arbitrary-dimension vectors. Additionally, we analyse the correctness, security, communication overheads and computation complexity of each protocol proposed in this paper.

For the future work, we will devote to the practicability of other secure protocols such that they can be efficiently used in privacy-preserving computations.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The work is supported in part by the National Natural Science Foundation of China (Nos. 61472470, 61370224), and the Open Project Foundation of Information Technology Research Base of Civil Aviation Administration of China.
References


