Secure Two-Party Distance Computation Protocol Based on Privacy Homomorphism and Scalar Product in Wireless Sensor Networks

Haiping Huang*, Tianhe Gong, Ping Chen, Reza Malekian, and Tao Chen

Abstract: Numerous privacy-preserving issues have emerged along with the fast development of the Internet of Things. In addressing privacy protection problems in Wireless Sensor Networks (WSN), secure multi-party computation is considered vital, where obtaining the Euclidian distance between two nodes with no disclosure of either side’s secrets has become the focus of location-privacy-related applications. This paper proposes a novel Privacy-Preserving Scalar Product Protocol (PPSPP) for wireless sensor networks. Based on PPSPP, we then propose a Homomorphic-Encryption-based Euclidean Distance Protocol (HEEDP) without third parties. This protocol can achieve secure distance computation between two sensor nodes. Correctness proofs of PPSPP and HEEDP are provided, followed by security validation and analysis. Performance evaluations via comparisons among similar protocols demonstrate that HEEDP is superior; it is most efficient in terms of both communication and computation on a wide range of data types, especially in wireless sensor networks.

Key words: secure two-party computation; privacy-preserving; wireless sensor networks; scalar product; distance calculation; privacy homomorphism

1 Introduction

1.1 Problem statement

In open ad hoc networks, including wireless sensor networks, the privacy of location information is usually considered as important to maintain[1], and is an important target for adversaries. For example, in military wireless sensor networks, how to protect private data of nodes, especially the location of vital nodes, has become a research hotspot: The concern is that enemies might obtain military secrets that would grant them a battle advantage through localization. Another research motivation comes from drivers in automotive traffic. They would like their location to remain private to avoid push notifications from location-based service providers who can obtain their positions through many types of sensors. Both the military and the traffic applications may be viewed as problems of privacy protection in wireless sensor networks. Current research on location-privacy preservation issues in WSN focuses on routing-level privacy problems, and proposes ways to protect the source-location information of delivered messages from being picked up by adversaries who fake message routing[2]. Most solutions, however, do not involve protecting data-level location privacy, frequently require excessive encryption and decryption operations or a trusted third party, and impose computational overhead and network costs unsuitable for WSN with their limited energy and computation capability.

Achieving data transfer related to position between two sensor nodes without revealing their respective location information has therefore become a sensitive
issue in WSN. For instance, a sensor node on one armored vehicle wants to calculate the distance between another armored vehicle and itself, but does not want to publish location information on wireless channels. This is a typical issue in secure two-party distance computation in WSN environments; this paper will focus on it by means of privacy homomorphism, rather than traditional secure measurements such as authentications or authorizations.

With the development of the privacy protection technologies required for the Internet of Things, secure multi-party computation, derived from Yao’s “millionaires problem”[3], has been widely used[4].

1.2 Proposed approaches

This paper focuses on secure two-party distance computation based on a privacy homomorphism strategy, which is an important branch of the field of Privacy-Preserving Computational Geometry (PPCG)[5]. Two-party protocols are the foundation for multi-party protocols.

Secure two-party privacy distance calculation protocols used in WSN can be described as follows. Suppose there are two sensor nodes, A and B, involved in this protocol. Node A has one private vector (an n-dimensional point vector) \( P = (x_1, x_2, ..., x_n) \). Node B holds another private vector (an n-dimensional point vector) \( Q = (y_1, y_2, ..., y_n) \). Two participants take their own secret vectors as inputs to the protocol and will achieve the final result through cooperation calculations without disclosing their respective privacy, i.e., both A and B know nothing about the other side’s inputs, but obtain the distance between them. We discuss the Euclidean distance (denoted as \( d(2) \)) calculated in Eq. (1).

\[
d(2) = \sqrt{\sum_{i=0}^{n} |x_i - y_i|^2}
\]

Currently, most secure two-party distance computation protocols are realized based on secure scalar product protocols[6], which frequently appear in such applications as location privacy protection, secure seeking for nearest-neighbor, safety search for spatial range, and private data mining. However, most current protocols cannot achieve satisfactory performance, or have defects such as only supporting a single data type. In response, the design of secure, effective, and novel protocols for WSN is the focus of this paper.

In this study, a semi-honest model is adopted, and two well-designed protocols based on privacy homomorphism and scalar product are proposed, which are distinguished from prevalent solutions, especially in secure multi-party computation, and would be applied in wireless sensor networks. A “calculation of indiscernibility” method is employed to demonstrate the security. Computational and communicational complexity, network delay, and availability are compared with those of previous works and are shown to be better.

The remainder of this paper is organized as follows. In Section 2, we formalize the semi-honest model and homomorphic encryption mechanism. We then present a novel secure two-party scalar product scheme, based on which we propose a secure two-party distance computation protocol in Section 3. The security analysis and communication and computation performance evaluation are outlined in Section 4. We review related works in Section 5. We describe our conclusions in Section 6.

2 Computation Model

2.1 Semi-honest model

In this paper, we make use of a semi-honest model[6] in our two-party computation protocol, where two participants and other parties (for instance, a WSN base station) faithfully carry out protocols. In doing so they provide their real data for calculations without causing intentional errors such as deliberately interrupting the execution of protocols. They save all of the intermediate information obtained in the calculation process and the final result, try to deduce the other side’s secrets from this information, and breach the privacy of both sides. A semi-honest participant is also called “honest but curious”. Generally, the “semi-honest” secure two-party computation model can be described as follows, which will be used in the security proof of the protocols.

There exists a mapping function \( f : \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^* \times \{0, 1\}^* \) where \( f_1(x, y) \) is the first element. And \( \pi \) denotes a secure two-party computation protocol. During the implementation of \( \pi \), the view of the first party \( \text{view}_1^\pi(x, y) \) is represented by \( (x, r^1, m_1^1, m_2^1, ..., m_l^1) \) (analogously \( (y, r^2, m_1^2, m_2^2, ..., m_l^2) \)) denotes the view of the second party \( \text{view}_2^\pi(x, y) \), where \( x \) is the input, \( r^1 \) denotes the result of random coin toss, and \( m_i^1 \) stands for the \( i \)-th intermediate message received by the first party. When \( \pi \) is completed, the output of the first
party is marked as output\(^2\)\((x, y)\) (the output of the second party is output\(^2\)\((x, y)\)), and the output of \(\pi\) is output\(^\pi\)\((x, y) = (\text{output}\(^2\)\((x, y)\), \text{output}\(^2\)\((x, y)\)).

Generally, if there exists a probabilistic polynomial-time algorithm (denoted by \(S_1\) or \(S_2\)) to execute the privacy computation of \(f\), we can conclude that

\[
\{(S_1(x, f_1(x, y), f_2(x, y)))\}_{x, y} \equiv \\
\{(\text{view}\(^2\)\((x, y)\), \text{output}\(^2\)\((x, y)\))\}_{x, y} \equiv \\
\{(f_1(x, y), S_2(x, f_2(x, y)))\}_{x, y} \equiv \\
\{(\text{output}\(^\pi\)\((x, y)\), \text{view}\(^2\)\((x, y)\))\}_{x, y}
\tag{2}
\]

where the meaning of “\(\equiv\)” is “calculation of indiscernibility”, output\(^\pi\)\((x, y)\) is entirely determined by \text{view}\(^2\)\((x, y)\), and both are stochastic variables. We call \(S_1\) or \(S_2\) as time-simulator.

2.2 Homomorphism encryption model

The Homomorphic Encryption Scheme (HES) is a common public-key system used in secure multi-party computation, whose encryption and decryption are denoted by \(E(\cdot)\) and \(D(\cdot)\). One type of encrypted operation on plain-text is marked with “\(\odot\)”, and can be equivalent to another operation (denoted “\(\otimes\)”) on the cipher-text, as shown in Eq. (4).

\[
E(x) \odot E(y) = E(x \odot y)
\tag{4}
\]

There are two kinds of encryption forms: one is Multiplicative Homomorphism as in Eq. (5), and ElGamal Encryption Algorithm\(^7\) is its specific case; another, that satisfies Eq. (6), is named Additive Homomorphism, such as the famous Paillier Encryption Algorithm\(^8\).

\[
E(x) \otimes E(y) = E(x \times y)
\tag{5}
\]

\[
E(x) \otimes E(y) = E(x + y)
\tag{6}
\]

Similarly, we can deduce the form of exponential encryption calculation in terms of Eq. (7).

\[
E(nx) = E(x)^n
\tag{7}
\]

3 Protocol Description

3.1 Secure scalar-product protocol based on privacy homomorphism

In this section, a novel Privacy-Preserving Scalar Product Protocol (PPSSP) based on homomorphic encryption is proposed for WSN. We describe a scalar product problem as follows: Node \(A\) owns a secret vector \(P\) and Node \(B\) holds another, \(Q\); they will securely calculate the result of \(P * Q\), where the two participants can deduce nothing of the other side from the result or any intermediate information. The execution of the protocol is divided into four phases: input phase, calculation phase, output phase, and correctness proof phase. Node \(A\) and Node \(B\) are semi-honest participants in the procedure.

1) Input phase

Node \(A\) selects a private vector \(P = (x_1, x_2, ..., x_n)\) as its input, and Node \(B\) chooses another \(Q = (y_1, y_2, ..., y_n)\) and keeps it secret.

2) Output phase

The result of protocol implementation is the scalar product of \(P\) and \(Q\), shown as Eq. (8). We also use \(f(P, Q)\) to represent the final result shared by the two participants, which satisfies Eq. (9), where Node \(A\) holds the value of \(f_a\) and Node \(B\) controls that of \(f_B\). Finally, Node \(A\) reveals \(f_a\) and Node \(B\) declares \(f_B\) publicly so that they can obtain the result \(f(P, Q)\) according to Eq. (9).

\[
P \otimes Q = \sum_{i=0}^{n} x_i y_i
\tag{8}
\]

\[
f_B = f(P, Q) + f_A
\tag{9}
\]

3) Calculation phase

Step 1: Node \(A\) generates a random vector \(Ra = (ra_1, ra_2, ..., ra_n)\), and for each \(ra_i\) \((i = 1, 2, ..., n)\), makes \(x'_i = ra_i + x_i\) a positive integer. To protect secret data, two vectors \(P_1\) and \(Q_1\) are defined and kept by Nodes \(A\) and \(B\), respectively. Node \(A\) calculates \(P_1\) in terms of Eq. (10). Subsequently, Node \(A\) creates a pair of keys \((E, D)\), where \(E(\cdot)\) denotes homomorphic encryption and \(D(\cdot)\) represents homomorphic decryption. Node \(A\) uses \(E(\cdot)\) to encrypt \(P_1\) and executes Eq. (11), and sends \(E(P_1)\) together with \(Ra\) and \(E(\cdot)\) to Node \(B\).

\[
P_1 = P + Ra = (x'_1, x'_2, ..., x'_n) = (ra_1 + x_1, ra_2 + x_2, ..., ra_n + x_n)
\tag{10}
\]

\[
E(P_1) = (E(ra_1 + x_1), E(ra_2 + x_2), ..., E(ra_n + x_n))
\tag{11}
\]

Step 2: After Node \(B\) receives \(E(P_1)\), \(Ra\) and \(E\), it generates a vector \(Rb = (rb_1, rb_2, ..., rb_n)\) stochastically, and for each \(rb_i\) \((i = 1, 2, ..., n)\), makes \(y'_i = rb_i + y_i\) a positive integer. Node \(B\) obtains \(Q_1\) according to Eq. (12), followed by selecting a random number \(v\), and calculating \(s_1\) and \(s_2\) based on homomorphic encryption, shown as Eqs. (13) and (14). Then Node \(B\) sends \(Rb\), \(s_1\), and \(s_2\) to Node \(A\).

\[
Q_1 = Q + Rb = (rb_1 + y_1, rb_2 + y_2, ..., rb_n + y_n)
\tag{12}
\]

\[
E(Q_1) = (E(rb_1 + y_1), E(rb_2 + y_2), ..., E(rb_n + y_n))
\tag{13}
\]

\[
E(B) = (E(s_1), E(s_2))
\tag{14}
\]
\[ s_1 = E(v) \prod_{i=1}^{n} E(x'_i y'_i) \quad (13) \]
\[ s_2 = \sum_{i=1}^{n} r_{ai} y'_i \quad (14) \]

Step 3: Node A decrypts \( s_1 \) and calculates \( u \) on the basis of Eq. (15) so that it can share the scalar product result \( P \cdot Q \) with Node B in accordance with Eq. (16), where \( f_A = -u \), \( f_B = -v \), \( f(P, Q) = P \cdot Q \), satisfying the form of Eq. (9).

\[
\begin{align*}
u &= D(s_1) - s_2 - P \ast Rb \quad (15) \\
\ast &= P \ast Q + v \quad (16)
\end{align*}
\]

(4) Correctness proof phase

Proof: we can substitute the detailed expressions from Eqs. (10), (12) – (14) for variables \( x_i, y_i, s_1 \), and \( s_2 \) in Eq. (15), and thus obtain Eq. (17),

\[
u = D(s_1) - s_2 - P \ast Rb =
D(E(v) \prod_{i=1}^{n} E(x'_i y'_i) - \sum_{i=1}^{n} r_{ai} y'_i - \sum_{i=1}^{n} x_i r_{bi} =
\sum_{i=1}^{n} r_{ai} (y_i + r_{bi}) - \sum_{i=1}^{n} x_i r_{bi} =
\sum_{i=1}^{n} (x_i + r_{ai}) (y_i + r_{bi}) -
\sum_{i=1}^{n} r_{ai} (y_i + r_{bi}) - \sum_{i=1}^{n} x_i r_{bi} =
\sum_{i=1}^{n} x_i y_i + v \quad (17)
\]

where \( P \ast Q = \sum_{i=1}^{n} x_i y_i \). We conclude that Eq. (15) is equivalent with Eq. (16), both of which are in keeping with the form of Eq. (9). The correctness proof is thereby completed.

3.2 Secure two-party distance computation protocol based on scalar product

A novel Homomorphic Encryption-based Euclidean Distance Protocol (HEEDP) with no third parties is put forward in this section, where we make use of PPSPP introduced in Section 3.1 to complete the scalar product calculation. Four phases are involved in this protocol, including input, output, calculation, and correctness proof, where the input and output phases of HEEDP are the same as those of PPSPP, except for substituting the distance square \(|PQ|^2\) for the scalar product result \( P \cdot Q \) in the output phase. We therefore only describe the latter two phases.

(1) Calculation phase

Step 1: Node A and Node B jointly execute the calculation phase of PPSPP; Node A gets \( u \), and Node B retains \( v \).

Step 2: Node A calculates \( u' \) according to Eq. (18), and Node B gains \( v' \) in terms of Eq. (19) so that they can share the distance result \(|PQ|^2\) in accordance with Eq. (20), where \( f_A = -u' \), \( f_B = v' \), \( f(P, Q) = |PQ|^2 \), satisfying the form of Eq. (9).

\[
u' = -2u + \sum_{i=1}^{n} x_i^2 \quad (18)
\]
\[
v' = 2v + \sum_{i=1}^{n} y_i^2 \quad (19)
\]
\[
u' = |PQ|^2 - v' \quad (20)
\]

(2) Correctness proof phase

Proof: we proved the equality \( u = v + \sum_{i=1}^{n} x_i y_i \) in Section 3.1, so we can deduce Eq. (21) by expanding Eq. (18). Then in terms of Eq. (19), we substitute \( v' = \sum_{i=1}^{n} y_i^2 \) for “2v” in Eq. (21), and we obtain Eq. (22) where \( f_A = -u' \), \( f_B = v' \), and \( f(P, Q) = |PQ|^2 \), satisfying the form of Eq. (9). The correctness proof is thus completed.

\[
u' = -2u + \sum_{i=1}^{n} x_i^2 =
-2(v + \sum_{i=1}^{n} x_i y_i) + \sum_{i=1}^{n} x_i^2 =
\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i - 2v \quad (21)
\]
\[
u' = \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i - 2v =
\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i - 2v + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} y_i^2 =
\sum_{i=1}^{n} (x_i - y_i)^2 - (2v + \sum_{i=1}^{n} y_i^2) =
\sum_{i=1}^{n} (x_i - y_i)^2 - v' \quad (22)
\]
4 Performance Evaluation

The security evaluation and efficiency analysis of PPSPP and HEEDP will be illustrated in this section.

4.1 Security proof

Theorem 1: In PPSPP, Node A and Node B cannot get each other’s secret input.

Proof: This protocol assumes that Node A and Node B are semi-honest participants, and they only share the result $P \ast Q$; they cannot get each other’s private vector. The security analysis of vital steps of PPSPP follows.

1) In the input phase, both Node A and Node B take data disguises for their respective private vectors $P$ and $Q$ with $\text{Ra} = (r_{a1}, r_{a2}, ..., r_{an})$ and $\text{Rb} = (r_{b1}, r_{b2}, ..., r_{bn})$. Elements in both $P$ and $Q$ are integers, which is convenient for hiding real data.

2) In Step 1 of the calculation phase, Node A encrypts $P_1$, and sends $E(P_1)$, $\text{Ra}$, and $E(\cdot)$ to Node B. However, Node B cannot decrypt $E(P_1)$, so Node B cannot deduce those elements in vector $P$.

3) In Step 2 of the calculation phase, Node B calculates $E(v + \sum_{i=1}^{n} x'_i y'_i)$ according to additive homomorphic encryption, where Node B uses a random number $v$ to realize data disguises. For example, $y_i$ is confused by $r_{bi}$, so even if $n = 1$, Node A cannot acquire $y_1$ from equations $s_2 = r_{a1}$ and $y'_1 = r_{a1}(r_{b1} + y_1)$. Node A cannot get any information about Node B’s secret vector $Q$ only from Node A’s intermediate data $s_1$ and $s_2$.

In summary, no information is revealed in any step of the PPSPP protocol in order to ensure the privacy of Node A and Node B.

Theorem 2: PPSPP (denoted by $\pi$) is privacy-preserving during the calculation of the scalar product.

Proof: Because only two parties are involved in the calculation, and are semi-honest participants, the proof can be simplified to “one party tries to deduce the other party’s private data in terms of its own input as well as all the intermediate information and the final output”. If the final output shared by Node A and Node B is the same as that deduced from the ideal model, we just need to prove that all of the intermediate information gained by the attacker who forces one party to implement PPSPP correctly, is “calculation of indiscernibility” as the information generated in the ideal environment.

The attacker will construct a simulator $S_1$, which will simulate all the intermediate information during the execution of PPSPP. We can discretionarily decide Node A (or Node B) is captured by the attacker so that the simulator $S_1$ is used to simulate the view of Node A: view$^i_1(P = (x_1, x_2, ..., x_n), Q = (y_1, y_2, ..., y_n))$. Thus this proved process is transformed into verifying that $(S_1(P, f_1(P, Q), f_2(P, Q)))$ and $\text{view}^i_1(P, Q)$ are “calculation of indiscernibility”, where $\text{view}^i_1(P, Q) = (P, r^1, m^1_1, m^1_2, ..., m^1_l)$. The input of Node A, i.e., $P = (x_1, x_2, ..., x_n)$, and the final scalar product result $P \ast Q$ are regarded as the inputs of $S_1$.

1) Input phase of PPSPP

The attacker captures Node A and notifies $S_1$, and acquires all of A’s data including its secret input $P$ and the final calculation result $P \ast Q$. Node A knows nothing about Node B’s private vector $Q$, which just appears a series of random bits for others. So $S_1$ has to select a simulative vector $Q' = (y'_1, y'_2, ..., y'_n)$ according to $(P, P \ast Q)$ and Eq. (8), where $y'_i$ is the replacement value for $y_i$, which also satisfies $P \ast Q = \sum_{i=1}^{n} x_i y'_i$.

2) Calculation phase of PPSPP

In Step 1, Node A generates a random vector $\text{Ra} = (r_{a1}, r_{a2}, ..., r_{an})$ and a pair of keys $(E, D)$ for homomorphic encryption, and A gets $P_1$ and $E(P_1)$ in terms of Eqs. (10) and (11), respectively. However, $S_1$ has to select another simulated random vector $\text{Ra}' = (r'_{a1}, r'_{a2}, ..., r'_{an})$, and according to Eq. (10), $S_1$ will obtain $P'_1 = P + \text{Ra}' = (x'_1, x'_2, ..., x'_n) = (r'_{a1} + x_1, r'_{a2} + x_2, ..., r'_{an} + x_n)$ and $E(P'_1)$ (the result of encryption of $P'_1$). Thus the attacker obtains all of Node A’s information in Step 1.

In Step 2, owing to the absence of Node B’s random vector $\text{Rb}$ and random number $v$, $S_1$ can only calculate $S'_1 = E(\cdot') \prod_{i=1}^{n} E(x'_i y'_{i'})$ and $S'_2 = \sum_{i=1}^{n} r_{a1} y'_{i'}$ in light of Eqs. (12)–(15), where $\text{Rb}' = (r'_{b1}, r'_{b2}, ..., r'_{bn})$ and $\nu'$ are imaginary data from $S_1$, which creates them by means of a stochastic coin toss by Node A. So $S_1$ simulates all the intermediate data of Node B in Step 2.

In Step 3, $S_1$ acquires $u' = D(s'_1) - s'_2 - P \ast \text{Rb}'$.

3) Output phase of PPSPP

These simulated data such as $u', \nu', Q' = (y'_1, y'_2, ..., y'_n)$ also satisfy the form of Eq. (9), i.e., $f(P, Q') = f(P, Q) + f_A$, where $f_A = -u'$, $f_B = -\nu'$. After two participants exchange their respective information, both of them obtain $f'_A$ and $f'_B$ so that they can
derive \( f(P, Q) \). Here \( S_1 \) simulates all of Node A’s data \( \{ f_1, f_2, f(P, Q) \} \).

In above Steps 1 to 3, all information handled by the attacker is not computationally distinguishable from that obtained in the normal execution of PPSP. Because \( S_1(P, P \ast Q) = \{ P, r_1, s'_1, s'_2, P \ast Q \} \), the view of Node A is \( \text{view}_A^P(P, Q) = \{ P, r_1, s_1, s_2 \} \), and the reasonable output of PPSP is \( \text{output}_1^P(P, Q) = P \ast Q \), we can obtain \( \{ S_1(P, P \ast Q), P \ast Q \} = \{ \text{view}_A^P(P, Q), \text{output}_1^P(P, Q) \} \).

Likewise, if an attacker suborns Node B, he will gather all information of Node B by creating another simulator \( S_2 \). The view of Node B is \( \text{view}_B^P(P = (x_1, x_2, \ldots, x_n), Q = (y_1, y_2, \ldots, y_n)) \), and we are also able to obtain the same results: \( \{ f_1(P, Q), S_2(Q, f_2(P, Q)) \} \) and \( \{ \text{output}_2^P(P, Q), \text{view}_B^P(P, Q) \} \) are thus a “calculation of indiscernibility”.

**Theorem 3:** HEEDP is privacy-preserving during the calculation of distance, where Node A and Node B cannot get each other’s secret input.

**Proof:** (1) In HEEDP, Node A and Node B exchange their respective information when and only when they jointly execute PPSP; (2) Through Theorems 1 and 2, PPSP has been proven secure. So Theorem 3 is thereby proven as well.

### 4.2 Efficiency analysis

In this section, the communication round complexity, computational complexity, and computational data type of HEEDP and other similar protocols will be discussed. For the convenience of comparison with HEEDP, we choose protocols based on the privacy homomorphism technique, including those documented in Refs. [4, 9–12]. Reference [4] only provides a secure two-party scalar product protocol, which can be supposed to be used in secure distance calculations, called “Lu’s protocol”. The protocol in Ref. [9] is named the “Amirbekyan protocol”. Similarly, the solutions in Refs. [10–12] are called “Luo’s protocol”, “Zhong’s protocol”, and “Rane’s protocol”, respectively. Because most of the safe distance calculation protocols are based on the Paillier encryption scheme, in order to facilitate comparisons, HEEDP also utilizes the Paillier scheme. The comparison details are displayed in Table 1.

#### 4.2.1 Communication round complexity

In HEEDP, Node A and Node B communicate with each other twice from Step 1 to Step 2, so the communication round complexity is 2. In addition to Lu’s protocol and Rane’s protocol, whose communication cost is proportional to original vector dimension \( n \), others have achieved satisfactory communication efficiency.

#### 4.2.2 Computational complexity

We ignore the computational cost of creating random numbers and the key pair for homomorphic encryption, which can be completed in the preprocessing stage; only the calculation phase is considered, whose primary computational cost relies on vectors’ dimensions and the complexity of homomorphic encryption. In PPSP, in order to calculate a scalar product with \( n \) dimensions, Node A needs to carry out \( n \) encryptions, i.e., \( E(P_1) = (E(r_{a1} + x_1), E(r_{a2} + x_2), \ldots, E(r_{an} + x_n)) \), in Step 1, Node B has to take one encryption, \( n \) modular multiplications, and \( n \) modular exponentiations, i.e.,

\[
s_1 = E(u) \prod_{i=1}^n E(x'_i)^{y_i}, \text{ in Step 2.}\n\]

In Step 3, Node A completes one decryption, i.e., \( u = D(s_1) - s_2 - P \ast Rb \). So the computational cost of HEEDP based on PPSP includes \( (n + 1) \) encryptions, \( n \) modular exponentiations, \( n \) modular multiplications, and one decryption. We have agreed that if one protocol uses the Paillier homomorphic encryption scheme, which needs \( 2\log N \) modular multiplications (the modular operator is \( N^2 \)) for each encryption or decryption, and \( 2d \)

### Table 1. Efficiency comparisons between HEEDP and other protocols.

| Protocol                  | Communication round complexity | Computational complexity for 2-dimensional vector | Computational complexity for 3-dimensional vector | Data type |
|---------------------------|-------------------------------|-----------------------------------------------|-----------------------------------------------|-----------|
| HEEDP protocol            | 2                             | \( (2n + 4)\log N + 2nd + n \)                 | \( 8\log N + 4d + 2 \)                         | Real      |
| Lu’s protocol             | \( n \)                       | \( (6n + 2)\log N + (2n + 2)d + n \)          | \( 14\log N + 6d + 2 \)                       | Real      |
| Amirbekyan’s protocol     | 2                             | \( 6n\log N + n \)                            | \( 12\log N + 2 \)                           | Integer   |
| Luo’s protocol            | 2                             | \( 6n\log N + 2nd + n \)                       | \( 12\log N + 4d + 2 \)                       | Real      |
| Zhong’s protocol          | 2                             | \( (2n + 4)\log N + 2nd + n \)                 | \( 10\log N + 6d + 3 \)                       | Integer   |
| Rane’s protocol           | \( n \)                       | \( (2n + 2)\log N + 2nd + n + 1 \)            | \( 8\log N + 6d + 4 \)                        | Real      |
modular multiplications at most (the modular operator is \( N^2 \)) for each modular exponentiation, where \( d \) indicates the number of bits of processed data. So the computational complexity of HEEDP can be specified as \((2n + 4)\log N + 2nd + n\) modular multiplications (the modular operator is \( N^2 \)).

Lu’s protocol has higher computational complexity than other protocols: \((6n + 2)\log N + (2n + 2)d + n\). Amirbekyan’s protocol, without consideration of the computational expense of the permutation replacement, needs \(2n\) encryptions, \(n\) modular multiplications, and \(n\) decryptions, a total of \(6n\log N + n\) by using the Paillier homomorphic scheme. Since no modular exponentiations appear in Amirbekyan’s protocol, it achieves low complexity but cannot deal with the situation where \(n = 1\). The computational complexity of Luo’s protocol is obviously higher than that of HEEDP, with a cost of \(6n\log N + 2nd + n\). Though Zhong’s protocol\(^{[11]}\) has the same computational complexity as HEEDP, it does not take the case of \(“n < 3”\) into account. Even with an unsatisfactory communication round cost, the computation complexity of Rane’s protocol is just \((2n + 2)\log N + 2nd + n + 1\) times modular multiplication (modular operator \( N^2 \)).

4.2.3 Computational data type

The disadvantage of Amirbekyan’s protocol and Zhong’s protocol is due to limitations within the integer range produced by direct encryption of private vectors without data disguises, while the data type of HEEDP and other protocols can be expanded to the real range.

As seen in Table 1, HEEDP is a comprehensive protocol, with the best communication round complexity, satisfactory computational complexity, and a broad calculation range for real numbers.

4.2.4 Availability in wireless sensor networks

4.2.4.1 Network lifetime analysis

It is necessary to verify the availability of HEEDP in wireless sensor networks. We design simulation experiments related to distance to observe changes in network lifetime and energy consumption in the following three scenarios: calculating the distance between Nodes A and B which directly exchange their location information without a secure strategy; using HEEDP to do the distance calculation; and realizing the computation through encryption, decryption, and a reliable third party (the base station) based on Elliptic Curves Cryptography (ECC), which is considered the most lightweight public key method in WSN.

To achieve data-level privacy protection in the third scenario, Nodes A and B must encrypt their locations separately using the public key of the base station, and the base station must decrypt them and calculate the distance between Nodes A and B; then the base station must encrypt the distance value by the public keys and send it to A and B. Finally, Nodes A and B decrypt it and obtain the result. If Nodes A and B cannot communicate with the base station within one hop, this method will cause heavy network loads.

In order to ensure the validity of comparisons, we make some assumptions about the three scenarios. As shown in Fig. 1, a distance calculation group would consist of any two nodes in the network. Once a node joins a group it cannot join other groups, unless its partner “dies”. If this happens, the node can find another single node with which to form a new group. In our simulation experiments, nodes prefer nearby nodes with which to form a group. All groups repeatedly calculate the distance between the two members, until the network life-cycle concludes. The base station does not participate in any group, and all packets are the same size which is divisible by 16. The Leach protocol is used for clustering and routing, to ensure path accessibility. Detailed simulation parameters are shown in Table 2.

We set two cases, “\(n = 3, d = 64\) bit” and “\(n = 8, d = 96\) bit”, to reflect the trend of network lifetimes, based on the comparisons of HEEDP and ECC, seen in Figs. 2a and 2b. We note that with the increase in sensor nodes, the network lifetime of both HEEDP and ECC appears to decline after an initial growth. There are different reasons for these phenomena for HEEDP and ECC. There still exist some active sensor nodes in HEEDP because they cannot find another neighbor in their communication ranges when the number of nodes is 100; however, the lifecycle is
network lifetime (s)  
Average remainder energy of nodes (J)  

| Parameters                          | Setting value |
|------------------------------------|---------------|
| Distribution of sensor nodes       | Uniform distribution of from 100 to 400 nodes, with the base station in the center |
| Clustering and routing protocol    | Leach         |
| The number of dimensions of location information | $n = 3, n = 8, and n = 12$ |
| Original energy of node            | 20 J (infinite energy for the base station) |
| Energy consumption when sending a packet (at most $d = 128$ bit) | 0.35 J |
| Energy consumption when receiving a packet (at most $d = 128$ bit) | 0.15 J |
| Energy consumption when processing a packet (at most $d = 128$ bit) | 0.0005 J |
| Energy consumption when encrypting or decrypting one bit with ECC (at most $d = 128$ bit) | 0.0005 J |
| Energy consumption when processing a modular multiplication (modular operator $N^2$) (at most $d = 128$ bit) | 0.001 J |
| Communication radius of node       | 50 m          |
| modular operator $N^2$             | $N = 79, N^2 = 6241$ |

Table 2 Simulation parameters settings.

**Fig. 2** Comparisons of network lifetime between HEEDP and ECC.

**Fig. 3** Comparison of average remainder energy among three scenarios.

station for each node in the network, the secure distance calculation based on ECC would lead to more energy consumption, compared with HEEDP. Similarly, if 100 nodes are uniformly distributed in a $400 \times 400$ field, at last some active nodes cannot establish a path to the base station before the lifecycle ends. Enhancement of sensor nodes could create more paths to the base station, which could prolong the network lifetime. However, when the number of nodes is 250 or more, many nodes will play the role of relay to achieve more paths, which will cause the shortening of the lifecycle due to greater energy consumption. It is worthy of notice that parameters $n$ and $d$ have more influence on the lifetime of HEEDP, which in Fig. 2b is less than that shown in Fig. 2a; and the larger $n$ and $d$, the greater the computation cost of HEEDP. However, the increase of the values of $n$ and $d$ has little effect on the decline of ECC, even though it increases computational overhead.

Figure 3 shows comparisons of average remainder energy of three scenarios which have completed three-quarters of their respective lifetimes by executing their protocols, where the value of $n$ is 12. The unsafe scenario obtains the best efficiency, and the scenario based on HEEDP also achieves more satisfactory performance than that of ECC when $d < 96$ bit. However, with the continuous increase of the value $d$, the scenario based on HEEDP would consume a little more energy than that of ECC, whose defect is in its energy consumption, caused by abundant communication frequencies once the value of $d > 0$.

Generally, the value $n$ is less than 3 and the bits of location data in a sensor node are fewer than 128, though the increase of location dimensions $n$ and bit number $d$ would bring heavy network overhead for HEEDP, as seen in Figs. 2 and 3.

**4.2.4.2 Data processing delay analysis**

Data processing delay may cause serious backlogs of packets in sensor nodes, so it should be taken into
consideration in order to prolong the network lifetime and guarantee Quality of Service (QoS), especially in real-time location-related scenarios such as intelligent traffic guidance. One of the most important impact factors is the speed of data encryption or decryption, which implies that encryption algorithms should be as efficient as possible. For the purpose of analysis and comparisons of data processing delays, simulation experiments on RSA, ECC, and HEEDP are conducted, and the total time consumed by the three methods is recorded. The reason RSA and ECC are chosen is that RSA is one of the most influential public encryption algorithms, and ECC is considered one of the most lightweight public key algorithms in WSN. To quantify these experiments, \( n \) vectors are encrypted at one time, where each vector contains \( n \) elements.

In the first experiment, \( \rho \) is fixed at 100, and \( n \) changes from 1 to 10. The result can be seen in Fig. 4a. Meanwhile, the second experiment fixes \( n \) at 10 and \( \rho \) varies from 0 to 200, as shown in Fig. 4b.

It can be seen that HEEDP has the most satisfactory performance among these three algorithms. This is because no power operations are involved in data processing. Although ECC is considered a lightweight algorithm, it consumes much time because of scalar multiplications and quadratic residue judgments. Moreover, both RSA and ECC consume lots of storage for sensor nodes, which further increases data processing delay.

5 Related Work

In location privacy applications, the location information is usually represented as a multi-dimensional vector, for example, two- or three-dimensional coordinates. A scalar product protocol is exactly suitable for multi-dimensional vector operations, so it becomes the basis of most secure distance calculation protocols. Scalar product protocols can be generally divided into two categories: one is based on a semi-honest third party, and the other achieves the correct result by the two parties without any aid from other parties. Many of these have been proposed, with different security levels and computational complexities [4, 9, 10, 13–15].

Protocols in Refs. [13, 14] use a semi-honest third party. Reference [13] proposes a scalar product protocol whose communication complexity is \( 4n \) (\( n \) denotes the number of dimensions of the original input), with four times the cost of Distributed Non-Private Setting (DNPS) that is considered the standard cost of a two-party scalar product protocol without any privacy calculations. Reference [14] presents a secure two-party quantum scalar product scheme via quantum entanglement and quantum measurement with the help of a non-colluding third party; this scheme is proven to be secure under various outside and inside attacks. However, the third-party involvement incurs extra overhead, and in many temporary-building or poor-link-quality wireless networks, it is difficult to find a permanent trusted third party.

At the same time, other scalar product protocols without a third party are being constantly proposed, where the most representative one is based on privacy homomorphism, such as protocols in Refs. [4, 9, 10]. In intelligent medical treatment applications, to achieve the tradeoff between the privacy disclosure and the high reliability of personal health information in healthcare emergency, Ref. [4] introduces an efficient user-centric privacy access control method, which is based on attribute-based access control and a new privacy-preserving scalar product computation technique, and
allows a medical user to decide who can participate in the opportunistic computing to assist in processing his personal health information. Amirbekyan’s protocol in Ref. [9] is based on the permutation replacement method, and makes use of privacy homomorphism and data confusion to achieve privacy protection. Luo et al. [10] suggested a homomorphic scalar product scheme and expanded it into a secure two-party distance calculation. However, the computational complexity of these protocols is higher than that of most third-party protocols. Their main computational cost is derived from homomorphic encryption operations, for example, the computational complexity of the proposed schemes in Refs. [4, 9, 10] is $O(n^3)$. Based on the data conversion technique, Ref. [15] puts forward a protocol with a smaller computational cost of $O(n^2)$, but attended by different degrees of privacy information leakage, especially when the vector dimension is small or some parameters are given special values.

Secure scalar-product-based two-party distance computation protocols are of widespread interest [10–12, 16, 17]. In the early research stages, assuming the existence of a third party is the design principal of two-party distance calculation. Through a third party, two participants attempt to obtain the distance value with no leakage of private data. The proposal in Ref. [11] describes a safe distance calculation solution aimed at two-dimensional vectors, containing an oblivious third party. However, as mentioned above, finding a reliable third party is difficult, and it may also become a security bottleneck. So secure two-party distance calculation protocols without a third party are paid more attention, such as the aforementioned studies [10, 12, 16, 17]. Reference [12] describes and analyzes a privacy-preserving approximation protocol for the L1 distance that keeps the computation overhead manageable by performing a Johnson-Lindenstrauss embedding into L2 space, and performing a secure two-party computation of L2 distance using Paillier homomorphism encryption. Similarly based on privacy homomorphism, Reference [16] gives an effective solution to some specific geometric problems, such as the distance between two private points.

Many researchers have designed safe protocols related to location privacy to apply in WSN. Reference [17] adopts traditional secure methods with greater network cost, and discusses the key concepts in source location privacy, such as anonymity, unobservability, etc. Then, it presents an overview of the solutions that provide source location privacy within a WSN, in relation to assumptions about the adversarys capabilities. Reference [18] proposes a Location Privacy Routing (LPR) protocol that is easy to implement and provides path diversity. Combined with fake packet injection, LPR is able to minimize the traffic direction information that an adversary can retrieve from eavesdropping. However, LPR provides a kind of security routing guarantee but not data-level privacy protection. Similar to Ref. [18], Ref. [19] introduces a novel attack to locate source nodes in WSNs, called Hotspot-Locating, which uses a realistic adversary model. It also proposes a source location privacy-preserving scheme that creates a cloud of fake packets around the source node, varies traffic routes, and changes the packets appearance at each hop. Reference [2] introduces a data-level location privacy problem, for example, the Privacy-Preserving Hop-distance Computation (PPHC) problem, and a protocol based on data disguise techniques is proposed, whose advantage is that it does not require a trusted third-party or encryption operations, and thus generally has much better performance than traditional solutions.

6 Conclusion

This paper puts forward a secure two-party distance calculation protocol, HEEDP, based on privacy homomorphism and the PPSPP scalar product protocol. Owing to the satisfactory security and remarkable performance on communication and computation complexity and data types, HEEDP can be considered a comprehensive protocol, suitable for networks requiring high security and load balancing of communications or calculations, such as wireless sensor networks. The protocol is implemented based on a semi-honest model, for instance, if malicious attackers capture some sensor nodes, and deliberately terminate the execution of protocols or provide fake data for calculations, two participants cannot obtain true results. Some current research work has already begun for dealing with a completely malicious model [20], which is the focus of our coming work.

Acknowledgment

The authors would like to thank the anonymous reviewers of this paper for their objective comments and helpful suggestions. This work was sponsored by the National Natural Science Foundation of China (No. 61373138), the
Natural Science Key Fund for Colleges and Universities in Jiangsu Province (No. 12KJA520002), the Key Research and Development Program of Jiangsu Province (Social Development Program) (No. BE2015702), the Postdoctoral Foundation (Nos. 2015M570468 and 2016T090485), the Sixth Talent Peaks Project of Jiangsu Province (No. DZZX-017), the Fund of Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks (WSNLBZY201516), and the Science and Technology Innovation Fund for Postgraduate Education of Jiangsu Province (No. KYLX15_0853).

References

[1] C. Yu, D. Yao, X. Li, Y. Zhang, L. T. Yang, N. Xiong, and H. Jin, Location-aware private service discovery in pervasive computing environment, Information Sciences, vol. 230, pp. 78–93, 2013.
[2] M. Xiao, L. Huang, H. Xu, Y. Wang, and Z. Pei, Privacy preserving hop-distance computation in wireless sensor networks, Chinese Journal of Electronics, vol. 19, no. 1, pp. 191–194, 2010.
[3] A. Yao, How to generate and exchange secrets, in Proceedings of The 27th Annual Symposium on Foundation of Computer Science, Toronto, Canada, 1986, pp. 162–167.
[4] R. Lu, X. Lin, and X. Shen, SPOC: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency, IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 3, pp. 614–624, 2012.
[5] L. Liu, C. Wu, and S. Li, Two privacy-preserving protocols for point-curve relation, Journal of Electronics, vol. 29, no. 5, pp. 422–430, 2012.
[6] C. Hazay and Y. Lindell, Efficient Secure Two-Party Protocols: Techniques and Constructions, Springer, 2010.
[7] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory, vol. IT-31, no. 4, pp. 469–472, 1985.
[8] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 1999, pp. 223–238.
[9] A. Amirbekyan and V. Estivill-Castro, A new efficient privacy-preserving scalar product protocol, in Proceedings of the Sixth Australasian Conference on Data Mining and Analytics, Gold Coast, Australia, 2007, pp. 209–214.
[10] Y. Luo, L. Huang, G. Chen, and H. Shen, Privacy-preserving distance measurement and its applications, Chinese Journal of Electronics, vol. 15, no. 2, pp. 237–241, 2006.
[11] H. Zhong, Y. Sun, and F. Yan, Protocol for privacy-preserving space closest-pair of points, Computer Engineering and Applications, vol. 47, no. 4, pp. 87–89, 2011.
[12] S. Rane, W. Sun, and A. Vetro, Privacy-preserving approximation of L1 distance for multimedia applications, in IEEE International Conference on Multimedia and Expo (ICME), Singapore, 2010, pp. 492–497.
[13] W. Du and Z. Zhan, Building decision tree classifier on private data, in Proceedings of the IEEE International Conference on Privacy, Security and Data Mining, Maebashi Terrsa, Japan, 2002, pp. 1–8.
[14] L. He, L. Huang, W. Yang, and R. Xia, A protocol for the secure two-party quantum scalar product, Physics Letters A, vol. 376, pp. 1323–1327, 2012.
[15] J. Vaidya and C. Clifton, Privacy preserving association rule mining in vertically partitioned data, in Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002, pp. 639–644.
[16] S. Li and Y. Dai, Secure two-party computational geometry, Journal of Computer Science and Technology, vol. 20, no. 2, pp. 258–263, 2005.
[17] M. Conti, J. Willemsen, and B. Crispo, Providing source location privacy in wireless sensor networks: A Survey, IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1238–1280, 2013.
[18] Y. Jian, S. Chen, Z. Zhang, and L. Zhang, Protecting receiver-location privacy in wireless sensor networks, in Proceeding of IEEE Infocom, 2007, pp. 1955–1963.
[19] M. Mahmoud and X. Shen, A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 10, pp. 1805–1817, 2012.
[20] B. Yang, Y. Yu, and C. Yang, A secure scalar product protocol against malicious adversaries, Journal of Computer Science and Technology, vol. 28, no. 1, pp. 152–158, 2013.

Tianhe Gong received the BEng degree from Nanjing University of Posts and Telecommunications, China, in 2014. He is currently with successive postgraduate and doctoral programs of study with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, China. His research interests include information security and privacy protection of wireless sensor networks.

Ping Chen received the BEng degree and MEng degree from Nanjing University of Posts and Telecommunications, China, in 2006 and 2013, respectively. She is currently an engineer with the Trend Micro, Nanjing, China. Her research interests include information security and privacy protection of wireless sensor networks.
Haiping Huang received the BEng degree and MEng degree in computer science and technology from Nanjing University of Posts and Telecommunications, China, in 2002 and 2005, respectively; and the PhD degree in computer application technology from Soochow University, China, in 2009. From May 2013 to November 2013, he was a visiting scholar with the School of Electronics and Computer Science, University of Southampton, UK. He is currently a professor with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, China. His research interests include information security and privacy protection of wireless sensor networks.

Tao Chen is currently an undergraduate with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, China. His research interests include information security and privacy protection of wireless sensor networks.

Reza Malekian received the BEng degree from Shomal University, Iran, in 2002, the MEng degree in information engineering/secured telecommunications from Iran University of Science and Technology in 2006, and the PhD degree in computer science from Universiti Teknologi Malaysia in 2009. He is currently a senior lecturer with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa. His research lies in the area of advanced sensor networks, Internet of Things, and mobile communications. He is also a professional member of the British Computer Society and registered as a Chartered Engineer in Engineering Council of UK.