# Verifiable Private Multi-party Computation: Ranging and Ranking

Lan Zhang*, Xiang-Yang Li,†, Yunhao Liu,‡§, Taeho Jung,†
* Department of Computer Science and Technology, TNList, Tsinghua University
† Department of Computer Science, Illinois Institute of Technology
‡ School of Software, Tsinghua University and MOE Key Lab for Information System Security
§ Department of Computer Science and Engineering, HKUST

***Abstract*—The existing work on distributed secure multi-party computation, *e.g.*, set operations, dot product, ranking, focus on the privacy protection aspects, while the verifiability of user inputs and outcomes are neglected. Most of the existing works assume that the involved parties will follow the protocol honestly. In practice, a malicious adversary can easily forge his/her input values to achieve incorrect outcomes or simply lie about the computation results to cheat other parities. In this work, we focus on the problem of verifiable privacy preserving multi-party computation. We thoroughly analyze the attacks on existing privacy preserving multi-party computation approaches and design a series of protocols for dot product, ranging and ranking, which are proved to be privacy preserving and verifiable. We implement our protocols on laptops and mobile phones. The results show that our verifiable private computation protocols are efficient both in computation and communication.**

***Index Terms*—Verifiability, Privacy, Multi-party Computation, Ranking, Ranging, Dot Product.**

## I. Introduction

Privacy preserving multi-parity computation is widely used in different areas. For example, similarity calculation in social networks [8], private voting and auction [9], private data aggregation in sensor networks [6], [7], [12], [17], ranking [10] and oursourced computation [14], [19]. A trusted central server is a simple way to address this problem. The trusted central server will collect the private inputs of all parties, compute the result and disseminate the result to required parties through the secure communication channel. However the server may not always be accessible for all users due to the absence of Internet connection or server failure. Frequent communications with server will cause high expense and security vulnerability. Recently the cracking of databases of some famous online sites causes severe leakage of users' privacy. As a result, many users may not want to reveal their private data even to a server, which also hinders the wide adoption of cloud computing. So there is a strong motivation to design distributed privacy preserving multi-party computation protocols. Note that theoretically, this has been addressed by Secure Multi-party Computation (SMC), which was first introduced in [15]. SMC enables parties to jointly compute a function over their individually held private inputs without any party learning information beyond what can be deduced from the result.

Although theoretically beautiful, generic SMC protocols are extremely expensive. Many successive work, like [3] and [8], provide practical solutions for private multi-parity computation (e.g. set intersection, dot product) based on homomorphic encryption and secret sharing. These existing approaches concentrate on the privacy protection, while the verifiability of user inputs and outcomes are neglected. The correctness of most protocols is based on the assumption that the involved parties will follow the protocol honestly. In practice a malicious adversary may simply forge his/her input value to produce an incorrect result or lie about the outcome of computation [2]. In this way, the malicious adversary can cheat other parties to accept the incorrect result and even compromise other parties' privacy. For example, in a social networking system, an unverifiable private similarity calculation protocol may allow a malicious user to get others's trust by inputting fake attributes or lying about similarity calculation results.

In this work, we focus on the important but neglected problem of verifiable private multi-party computation in an insecure environment without a long-term trustable server. The computation is conducted in a distributed manner, and a trusted server is only occasionally contacted to authenticate the inputs of users. Specifically, we discover two potential attacks on existing protocols for private multi-party computation. We then design verifiable private multi-party threshold-based ranging protocols and ranking protocols suitable for different applications. We prove that these protocols are privacy preserving and verifiable for both input and output values. We thoroughly analyze the performances of our protocols and evaluate them with implementations on laptops and mobile phones. Most of our protocols cost less than 0.5 second on both laptop and mobile phone. Our most expensive protocol (verifiable two-way ranging) takes only 2.56 seconds on laptop and 3.22 seconds on mobile phone, and the largest message size is 3.5 KB. The results show that our verifiable protocols are efficient in both computation and communication.

**Paper Organization:** The rest of the paper is organized as follows. We define our problem and present the adversary model in Section II. In Section III and Section IV, we respectively present our verifiable private multi-party computation protocols for threshold-based ranging and ranking. We report our analysis and evaluation results in Section V, review the related work in Section VI, and conclude the paper in Section VII.

TABLE I
THE FAST VARIANT OF PAILLIER'S CRYPTOSYSTEM

| |
|---|
| Choose two prime numbers $p$ and $q$. |
| **Public key**: |
| modulus $n = pq$ and base $g \in \mathbb{Z}^*_{n^2}$ |
| **Private key**: |
| $\lambda = LCM(p-1, q-1)$ |
| **Encryption**: |
| $c = E(m,r) = g^{m+nr} \mod n^2$ |
| **Decryption**: |
| $m = D(c) = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$, $L(x) = \frac{x-1}{n}$ |
| **Homomorphic**: |
| $E(m_1,r_1)E(m_2,r_2) \mod n^2 = E(m_1+m_2, r_1+r_2) \mod n^2$ |
| $E(m_1,r_1)^{m_2} \mod n^2 = E(m_1 m_2, r_1 m_2) \mod n^2$ |
| **Self-blinding**: |
| $D(E(m_1,r_1)) = E(m_1, r_1+r_2)$ |

## II. PROBLEM DEFINITION AND PRELIMINARY

### A. Verifiable Private Multi-party Computation

There are $n$ parties $P = \{P_1, \dots, P_n\}$, where each party $P_i$ holds a private value $v_i$. $n$ parties wish to compute $f(v_i, v_2, \dots, v_n) = (y_1, y_2, \cdots, y_n)$ by communicating a-mong themselves, without giving away any information about their own values. We say a multi-party computation protocol is *verifiable*, if a malicious party cannot cheat other parties to accept an incorrect result. The *verifiability* of the computation result is usually neglected in existing protocols. In this work, we focus on the verifiable private computation which resists forged inputs and manipulation of computation results as well as preserves the privacy of input values.

### B. Homomorphic Encryption

Homomorphic encryption allows specific types of computations to be carried out on cipher text and obtains an encrypted result which is the cipher text of the computation result of the plain text. In this work, we use the fast variant of Paillier's cryptosystem which is additively homomorphic as an example. The detail is presented in Table I.

Let the public key of the user $P_i$ be $Pk_i$ and the private key be $Sk_i$. We denote the encryption with $P_i$'s public key as $\mathrm{E}_{Pk_i}(\cdot)$, encryption with private key as $\mathrm{E}_{Sk_i}(\cdot)$. Similarly, $\mathrm{D}_{Pk_i}(\cdot)$ and $\mathrm{D}_{Sk_i}(\cdot)$ denote the decryption operations with $P_i$'s keys. For simplicity, when no confusion caused, $\mathrm{E}_i$ stands for $\mathrm{E}_{Pk_i}$ and $\mathrm{D}_i$ stands for $\mathrm{D}_{Sk_i}$ using the Paillier's cryptosystem.

### C. Adversary Models

Many attacks are extensively studied in related private multi-party computation work, *e.g.* [3], [8]. Here, we discover two potential attacks (the compressive sensing based privacy reconstruction and fake signature) to the state-of-art private multi-party computation schemes, which haven't been well studied yet.

*1) Compressive Sensing Based Privacy Reconstruction:* In social networks, considering a user's attributes or relationship as a vector, private dot product is a typical way for profile matching and proximity calculation. However, a user's vector can be reconstructed via multiple rounds of proximity computation. Let $\mathbf{v_k}$ be the $M$-dimension private attribute vector of a user. Each dot product result gives one linear constraint on $\mathbf{v_k}$. In a traditional way as stated in [2], an adversary needs $M$ linearly independent constraints to reconstruct the victim's private vector $\mathbf{v_k}$. Indeed, $\mathbf{v_k}$ is usually $K$-sparse (it has at most $K$ non-zeros) and $K \ll M$. Based on the research in the compressive sensing [1], an adversary can recover the $K$-sparse length-$M$ user private vector $\mathbf{v_k}$ from only $R \geq cK \log(M/K) \ll M$ dot-products, here $c$ is a small constant.

So we suggest that, to achieve privacy preserving dot product in sparse-vector systems like social networks, a protection is needed to resist $o(K \log M)$ queries from the same user or a collusive attack of a group of adversaries.

*2) Fake Signature:* Most existing work of SMC are based on homomorphic encryption systems. Encrypted private values are input, and a series of computation are conducted homomorphically on these encrypted values to generate the encryption of the computation result on these values. A few methods like [2] propose to use a signature of encrypted input from a trusted third party $P_T$ to ensure the authenticity and consistency of the input value. However, we find that if the trusted third party $P_T$ directly signs the encrypted value $\mathrm{E}_i(v_i)$ and the digital signature generation system are homomorphic, the party $P_i$ is able to generate a fake signature for value $k \cdot v_i$ or $v_i^k$ without contacting $P_T$. Here $k$ is a constant picked by $P_i$. As a result, $P_i$ can use $k \cdot v_i$ as the input of a multi-party computation for arbitrary $k$,to cheat other parties to believe an incorrect conclusion.

So we suggest to avoid using homomorphic encryption to directly sign the encrypt value in private multi-party computation.

In the following part of this paper, we will focus on the verifiable private multi-parity computation resisting the adversaries of fake input and outcome, which are ignored by most existing work.

*3) Fake Input and Fake Outcome:*

*Definition 1 (Fake Input Adversary):* In the private multi-party computation, an adversary cheats by inputting an arbitrary value to deviate the result from its true value.

*Definition 2 (Fake Outcome Adversary):* In the private multi-party computation, a malicious party knowing the true result tells a wrong conclusion to trick other parties.

### D. Certificate

In this work, we suppose that there is a trusted third party $P_T$ who is only involved to authenticate the input values of participants. So there is no requirement for long-term involvement by $P_T$. Each participant $P_i$ can contact $P_T$ to authenticate his/her encrypted values. $P_T$ signs the authenticated values and hands $P_i$ his/her certificate consisting the following content and other information in a typical certificate:

$$\mathbf{C}(P_i, v_i) = \langle \mathrm{Sig}(ID_i), \mathrm{E}_{Pk_i}(v_i), \mathrm{Sig}(\mathrm{E}_{Pk_i}(v_i)), Pk_i, Pk_T \rangle$$

Note that the signature algorithm by the trusted authority is not homomorphic.

**Protocol 1:** One-way Threshold-based Ranging Protocol

1) $P_1$ sends $\mathrm{E}_1(\theta, r_1)$ and the certificate $\mathbf{C}(P_1, \mathbf{V}_1)$ to $P_2$. Here certificate $\mathbf{C}(P_1, \mathbf{V}_1) = \langle \mathrm{Sig}(ID_1), \mathrm{E}_1(\mathbf{V}_1, \mathbf{R}_1), \mathrm{Sig}(\mathrm{E}_1(\mathbf{V}_1, \mathbf{R}_1)), Pk_1, Pk_T \rangle$.
2) $P_2$ randomly picks a certificate $\mathbf{C}(P_2, \delta_1 \mathbf{V}_2)$, where $\mathbf{C}(P_2, \delta_1 \mathbf{V}_2) = \langle \mathrm{Sig}(ID_2), \mathrm{E}_2(\delta_1, r_2), \mathrm{E}_2(\mathbf{V}_2, \mathbf{R}_2), \mathrm{E}_2(\delta_1 \mathbf{V}_2, \mathbf{R}_2), \mathrm{Sig}(\mathrm{E}_2(\delta_1, r_2)), \mathrm{Sig}(\mathrm{E}_2(\delta_1 \mathbf{V}_2, \mathbf{R}_2)), Pk_1, Pk_T \rangle$, and another arbitrary number $\delta_2$. For ranging computation, $P_2$ computes
$$\begin{cases} e_1 = \mathrm{E}_1(\delta_1 \mathbf{V}_1 \cdot \mathbf{V}_2 + \delta_2, \delta_1 \mathbf{R}_1 \cdot \mathbf{V}_2 + r_2), \\ e_2 = \mathrm{E}_1(\delta_1 \theta + \delta_2, \delta_1 r_1 + r_2) \end{cases}$$
For verification purpose, $P_2$ computes
$$\begin{cases} e_3 = \mathrm{E}_1(\mathbf{R}_2 \cdot \mathbf{V}_1 + r_2, \mathbf{R}_1 \cdot \mathbf{R}_2 + r_2), \\ e_4 = \mathrm{E}_1(r_2 \theta + r_2, r_1 r_2 + r_2), \\ e_5 = \mathrm{E}_2(\delta_2, r_2), \end{cases}$$
$P_2$ sends $e_1$,$e_2$,$e_3$,$e_4$ to $e_5$ and the certificate to $P_1$.
3) $P_1$ compares $d_1 = \mathrm{D}_1(e_1)$ and $d_2 = \mathrm{D}_1(e_2)$ to determine the ranging result. $P_1$ verifies the value in $\mathbf{C}(P_2, \mathbf{V}_2)$. $P_1$ computes $d_3 = \mathrm{D}_1(e_3)$ and $d_4 = \mathrm{D}_1(e_4)$ to get the information of random number, then $P_1$ computes and checks the equations:
$$\begin{cases} \mathrm{E}_2(\delta_1 \mathbf{V}_1 \cdot \mathbf{V}_2 + \delta_2, \mathbf{R}_2 \cdot \mathbf{V}_1 + r_2) = \mathrm{E}_2(d_1, d_3), \\ \mathrm{E}_2(\delta_1 \theta + \delta_2, r_2 \theta + r_2) = \mathrm{E}_2(d_2, d_4). \end{cases}$$
If they are not all true, $P_1$ learns that $P_2$ is cheating.

**Protocol 2:** Two-way Threshold-based Ranging Protocol

1) $P_1$ sends $P_2$ a randomly picked certificate $\mathbf{C}(P_1, \delta_{11} \mathbf{V}_1)$.
2) $P_2$ computes $\mathrm{E}_1(\theta)$ by himself/herself.
The rest of this step is same as Step 2 in Protocol 1, with the only difference that here we use $\delta_{21}$ and $\delta_{22}$ for $P_2$'s parameters.
3) $P_1$ compares $d_1 = \mathrm{D}_1(e_1)$ and $d_2 = \mathrm{D}_1(e_2)$ to determine the ranging result, and verifies the result as in Step 3 in Protocol 1. If the verification fails, $P_1$ learns that $P_2$ is cheating and terminates the protocol. Otherwise, $P_1$ computes
$$\begin{cases} m_1 = \mathrm{E}_2(\delta_{11} d_1 + \delta_{12}), \\ m_2 = \mathrm{E}_2(\delta_{11} d_2 + \delta_{12}), \\ m_3 = \mathrm{E}_1(\delta_{12}), \end{cases}$$
and sends them to $P_2$. As in Protocol 1 the encrypted information for random numbers by $Pk_2$ are also sent to $P_2$.
4) $P_2$ can compare $dm_1 = \mathrm{D}_2(m_1)$ and $dm_2 = \mathrm{D}_2(m_2)$ to determine the ranging result. $P_2$ verifies the value in $\mathbf{C}(P_1, \delta_{11} \mathbf{V}_1)$, then $P_2$ decrypts the encrypted information of random number. $P_2$ verifies the result if the following equations are true to determine if $P_1$ cheating.
$$\begin{cases} \mathrm{E}_1(\delta_{11} \mathbf{V}_1 \cdot \delta_{21} \mathbf{V}_2 + \delta_{11} \delta_{22} + \delta_{12}) = \mathrm{E}_1(dm_1), \\ \mathrm{E}_1(\delta_{11} \delta_{21} \theta + \delta_{11} \delta_{22} + \delta_{12}) = \mathrm{E}_1(dm_2). \end{cases}$$

## III. Verifiable Threshold-based Ranging Protocol

In this section, we present our two-party threshold-based ranging protocol which is the first privacy preserving ranging protocol supporting verifiability for both input and outcome.

First, we define the threshold-based ranging computation.

*Definition 3 (Threshold-based Ranging):* $P_1$ holds private value $v_1$, and $P_2$ holds private value $v_2$. There is a polynomial function $f(v_1, v_2)$ and a threshold $\theta$. Users $P_1$ and $P_2$ can only determine whether $f(v_1, v_2) > \theta$ or not.

In the verifiable private threshold-based ranging, only the 1-bit comparison result will be exposed to them. This computation can provide better privacy and be widely used in many applications, like [7], [12].

### A. Protocol Design

Our verifiable private threshold-based ranging computation is based on the following observation.

*Theorem 1:* Given a large integer $n$, two arbitrary positive numbers $\delta_1, \delta_2$, $(\delta_1 f(v_1, v_2) + \delta_2 \mod n) > (\delta_1 \theta + \delta_2 \mod n) \Leftrightarrow f(v_1, v_2) > \theta$ if $\delta_1 f(v_1, v_2) + \delta_2 \in [0, n-1]$ and $\delta_1 \theta + \delta_2 \in [0, n-1]$.
Specifically, a party can choose $\delta_1 \le \sqrt{n}$ and $\delta_2 \le \sqrt{n}$, if $f(v_1, v_2) \le \sqrt{n}$.

*1) One-way Protocol:* We first consider the one-way situation that $P_1$ wants to conduct the query, but $P_2$ doesn't need the result. Protocol 1 is the first verifiable private threshold-based ranging protocol. It uses a practical partial homomorphic encryption, the Paillier's encryption. Each $P_i$ needs to acquire a series of certificates $\mathbf{C}(P_i, \delta_k v_i) = \langle \mathrm{Sig}(ID_i), \mathrm{E}_i(\delta_k), \mathrm{E}_i(v_i), \mathrm{E}_i(\delta_k v_i), \mathrm{Sig}(\mathrm{E}_i(\delta_k)), \mathrm{Sig}(\mathrm{E}_i)(\delta_k v_i)), Pk_i, Pk_T \rangle$, from a trusted authority $P_T$ by giving $P_T$ a set of random numbers $\{\delta_k\}$ chosen by $P_i$ and $\mathrm{E}_{Pk_i}(v_i)$ before the protocol launches. $P_i$ can update the certificates once he/she can contact $P_T$. Although the function $f(v_1, v_2)$ in Protocol 1 can be any combination of addition and multiplication. W.l.o.g., we use the dot product $f(\mathbf{V}_1, \mathbf{V}_2) = \mathbf{V}_1 \cdot \mathbf{V}_2$ of two private vectors as an example in the protocol statement. $\mathbf{R}_i$, $r_i$ are random vector and number chosen by $P_i$ as the required input of the Paillier's encryption.

*2) Two-way Protocol:* In the two-way situation, there is a common threshold $\theta$. $P_1$ and $P_2$ both need the verifiable result. A change is required to enable the two-way verifiable ranging. The values $\delta_{11}$ and $\delta_{12}$ are secretly chosen by $P_1$ and $\delta_{21}$ and $\delta_{22}$ are secretly chosen by $P_2$. A straightforward way is to repeat the Step 2 in the Protocol 1 to have $P_1$ compute $\mathrm{E}_2(\delta_{11} \mathbf{V}_1 \cdot \mathbf{V}2 + \delta_{12})$ and $\mathrm{E}_2(\delta_{11} \theta + \delta_{12})$ on the encrypted $\mathrm{E}_2(\mathbf{V}_2)$ homomorphically. However, since $\mathbf{V}_2$ could be a large vector, a simple extension will cost expensive computation. We design a more sophisticated solution and present Protocol 2 to reduce the computation cost. In the Step 3 of Protocol 2, we convert the homomorphic dot product computation on the ciphertext to a simple multiplication between two plain scalars which significantly simplifies the computation. According to our proof (omitted due to limited space), the input random number of Paillier's encryption won't affect the verifiability of the computation. For simplicity, we ignore the process of random numbers in the statement of Protocol 2 and the following protocols.

**Protocol 3:** Participant Comparison Protocol

1) $P_1$ sends certificate $\mathbf{C}(P_1, v_1)$ to $P_2$.
2) $P_2$ randomly picks a certificate $\mathbf{C}(P_2, \delta_1 v_2)$ and another arbitrary number $\delta_2$. $P_2$ computes
$$\begin{cases} e_1 = \mathrm{E}_1(\delta_1 v_1 + \delta_2), \\ e_2 = \mathrm{E}_1(\delta_1 v_2 + \delta_2), \\ e_3 = \mathrm{E}_2(\delta_2), \end{cases}$$
and sends them with the certificate and the encrypted information of random numbers by $Pk_1$ to $P_1$.
3) $P_1$ compares $d_1 = \mathrm{D}_1(e_1)$ and $d_2 = \mathrm{D}_1(e_2)$ to determine the comparison result. $P_1$ verifies the value in $\mathbf{C}(P_2, v_2)$, then decrypts the information of random number and checks the following equations:
$$\begin{cases} \mathrm{E}_2(\delta_1 v_1 + \delta_2) = \mathrm{E}_2(d_1), \\ \mathrm{E}_2(\delta_1 v_2 + \delta_2) = \mathrm{E}_2(d_2). \end{cases}$$
If they are not all true, $P_1$ learns that $P_2$ is cheating.

## IV. Verifiable Ranking Protocol

Here we propose privacy-preserving verifiable ranking protocols in both participants model and aggregator model.

### A. Ranking Problem Definition

*Definition 4 (Participant Ranking):* A party $P_1$ queries the rank of his own private value $v_1$ among $n$ parties's private values $V = (v_1, v_2, ....v_n)$. At the end, $P_1$ only learns the ranking result $R(P_1, P)$, which is an integer. Here $R(P_1, P) = K$ is $v_1$ is the $K$-th smallest in $V$.

*Definition 5 (Aggregator Ranking):* An aggregator $P_a$ wants to rank all $n$ parties's values $V = (v_1, v_2, ....v_n)$. At the end, $P_a$ only learns the ranking result $\langle ID_{r1}, ID_{r2}, \ldots ID_{rn} \rangle$ where $ID_{ri}$ is the ID of the user whose data is ranked $i$-th.

In both models, all participants' values are kept private and only the initiator learns the result.

### B. Protocol Design

*1) Participant Model:* Comparison between two parties is the basic operation of participant ranking. First, we present our verifiable private comparison protocol (Protocol 3) between two participants. In the end of Protocol 3, $P_1$ only learns the verifiable comparison result of $v_1$ and $v_2$, and $P_2$ learns nothing. Based on the verifiable private comparison, $P_1$ computes the rank $R(P_1, P)$ by comparing $v_1$ with each party separately, and counting the values larger than $v_1$ to conclude the rank $R(P_1, P)$.

*2) Aggregator Model:* In the aggregator model, we first design the protocol to compare two parties' private value for the aggregator $P_a$. It requires that, both $P_1$ and $P_2$' values are kept private and at the end of the protocol only the aggregator gets the verifiable comparison result. It is challenging when every party can eavesdrop all the communication. We present our design as Protocol 4.

By Protocol 4, the aggregator is able to get the verifiable comparison result between any pair of parties. Then we leverage the merge sort mechanism to design a parallel scheme for ranking. Only $O(\log n)$ rounds of comparisons will be launched by the aggregator. With the parallel steps 2) to 5) of Protocol 4, time complexity will be reduced compared to a serial scheme. Total $O(n \log n)$ comparisons are required in the worst case.

**Protocol 4:** Aggregator Comparison Protocol

1) $P_a$ launches the comparison by sending $P_1$ and $P_2$ his/her public key $Pk_a$.
2) $P_1$ randomly picks a certificate $\mathbf{C}(P_1, \delta_1 v_1)$. $P_1$ sends the certificate and $m_1 = \mathrm{E}_2(\mathrm{E}_a(\delta_1 v_1))$ to $P_2$.
3) $P_2$ randomly picks a certificate $\mathbf{C}(P_2, \delta_2 v_2)$. $P_2$ sends the certificate and $m_2 = \mathrm{E}_1(\mathrm{E}_a(\delta_2 v_2))$ to $P_1$.
4) $P_1$ verifies the value in $\mathbf{C}(P_2, v_2)$. $P_1$ decrypts $m_2$, then computes the following value and sends them with key $Pk_1$ and the encrypted information of random numbers by $Pk_a$ to $P_a$:
$$\begin{cases} e_{11} = \mathrm{E}_a(\delta_1 \delta_2 v_2), \\ e_{12} = \mathrm{E}_a(\mathrm{E}_2(\delta_1 \delta_2 v_1)), \\ e_{13} = \mathrm{E}_a(\mathrm{E}_2(\delta_1 \delta_2 v_2)). \end{cases}$$
Here $e_{12}$ is calculated based on $\mathrm{E}_2(\delta_2)$ in $\mathbf{C}(P_2, \delta_2 v_2)$ and $e_{13}$ is calculated based on $\mathrm{E}_2(\delta_2 v_2)$ in $\mathbf{C}(P_2, \delta_2 v_2)$.
5) $P_2$ verifies the value in $\mathbf{C}(P_1, v_1)$. $P_2$ decrypts $m_1$, then computes the following value and sends them with key $Pk_2$ and the encrypted information of random numbers by $Pk_a$ to $P_a$:
$$\begin{cases} e_{21} = \mathrm{E}_a(\delta_1 \delta_2 v_1), \\ e_{22} = \mathrm{E}_a(\mathrm{E}_1(\delta_1 \delta_2 v_2)), \\ e_{23} = \mathrm{E}_a(\mathrm{E}_1(\delta_1 \delta_2 v_1)). \end{cases}$$
Here $e_{22}$ is calculated based on $\mathrm{E}_1(\delta_1)$ in $\mathbf{C}(P_1, \delta_1 v_1)$. $e_{23}$ is calculated based on $\mathrm{E}_1(\delta_1 v_1)$ in $\mathbf{C}(P_1, \delta_1 v_1)$.
6) $P_a$ compares $d_1 = \mathrm{D}_a(e_{11})$ and $d_2 = \mathrm{D}_a(e_{21})$ to determine the comparison result. Then $P_a$ decrypts the information of random numbers and checks the equations:
$$\begin{cases} \mathrm{E}_2(d_1) = \mathrm{D}_a(e_{13}), \mathrm{E}_1(d_1) = \mathrm{D}_a(e_{22}), \\ \mathrm{E}_1(d_2) = \mathrm{D}_a(e_{23}), \mathrm{E}_2(d_2) = \mathrm{D}_a(e_{12}). \end{cases}$$
If they are not all true, $P_a$ learns that there's a party cheating.

## V. Analysis and Performance Evaluation

### A. Protocol Analysis

*Theorem 2:* If the Paillier's cryptosystem is semantically secure, Protocol 1 to 3 are privacy preserving.
Note that, in the Protocol 1, because the threshold $\theta$ is chosen by $P_1$, $P_1$ may launch a binary search by adjusting $\theta$ to narrow down the value range of $f(v_1, v_2)$. So we suggest that, a threshold may be given by the system or the query time should be limited.

*Theorem 3:* If the Paillier's cryptosystem is semantically secure, the Protocol 4 is privacy preserving, when neither $P_1$ nor $P_2$ colludes with $P_a$ even if the $P_1$, $P_2$ and $P_a$ can eavesdrop all the communication.
However, if one of the party colludes with $P_a$, they can learn the value of the other party.

*Theorem 4:* Protocol 1 to 3 are verifiable: $P_2$ cannot cheat $P_1$ to accept an incorrect result, and vis versa.

TABLE II
PERFORMANCE OF EACH PROTOCOL WITH REAL IMPLEMENTATION.M=30

| Protocol | Party | Computation(s) (Verification) | | Computation(s) (All) | | Communication | |
|---|---|---|---|---|---|---|---|
| | | Laptop | Phone | Laptop | Phone | (KB) | (Times) |
| 1 | $P_1$ | 0.30 | 0.36 | 0.41 | 0.49 | 1.5 | 1 |
| | $P_2$ | no | no | 2.05 | 2.72 | 3.25 | 1 |
| 2 | $P_1$ | 0.30 | 0.36 | 2.56 | 3.22 | 3.25 | 2 |
| | $P_2$ | 0.30 | 0.37 | 2.35 | 3.09 | 3.25 | 1 |
| 3 | $P_1$ | 0.23 | 0.26 | 0.23 | 0.26 | 1.25 | 1 |
| | $P_2$ | no | no | 0.12 | 0.14 | 3.25 | 1 |
| 4 | $P_1$ | no | no | 0.46 | 0.52 | 3.5 | 2 |
| | $P_2$ | no | no | 0.46 | 0.52 | 3.5 | 2 |
| | $P_a$ | 0.45 | 0.51 | 0.45 | 0.51 | 0.25 | 1 |

*Theorem 5:* Protocol 4 is unconditionally verifiable when at most one party cheats. And when both parties cheat individually without collaboration, $P_a$ can verify the result with a quite high probability.
However, when $P_1$ and $P_2$ collude, they can cheat the aggregator to believe an incorrect result.

All the proof are omitted here due to the space limitation.

## B. Performance Evaluation

Our protocols are designed based on the fast variant of the Paillier's cryptosystem, as in Table I. As in most implementations, we assume that $n$ is 1024-bit, $\lambda$ is 160-bit and the random number is 900-bit. We evaluate our protocols on both laptop and mobile phone. The laptop is Think Pad X1 with i7 2.7GHz CPU and 4GB RAM. The mobile phone is HTC G17 with $1228Hz$ CPU, and 1GB RAM. Table II presents the computation and communication cost of each protocol when the dimension of the vector is 30. The result shows that our protocols are practical in both laptop and mobile phones.

## VI. RELATED WORK

Secure multi-party computation (SMC) was initially introduced in [15]. One line of work on SMC is based on oblivious polynomial evaluation (OPE), *e.g.* [3], [8], [17]. Another line is based on oblivious pseudo random functions (OPRF), *e.g.* [5]. In all these approaches, both the input value and output result are not verifiable. The true result is only revealed to one party, who can cheat other parties by a forge result. Dong *et al.* [2] propose the fist scheme supporting verifiable private dot product between two parties.

There are some work leveraging Verifiable Secret Sharing (VSS) [13] to conduct multiparty computation of secret inputs when a majority of the players are honest, *e.g.* [11]. Secure computation based on VVS needs to share each secret input among $n$ parties and requires at least $t$ parties cooperate to produce the computation result. It results in high communication and computation cost.

Cloud computing enables the computational resource limited users to outsource their workload to the cloud. However, treating the cloud as an untrusted computing platform, privacy and verifiability are two of the main obstacles of its wide adoption. Recently, many work are dedicated to the privacy-assured outsourced computing *e.g.* [14], [18] and cloud data access [19]. There are also some efforts on the verification of the outsourced computation result, *e.g.* [4].

## VII. CONCLUSION

In this paper, we analyze the potential attacks to the state-of-art secure multi-party computation schemes. Previous protocols focus on privacy protection, usually leaving the verifiability neglected. we propose the first verifiable privacy preserving protocols for threshold-based ranging and ranking in different situations, which can resist cheating on both input and outcome. We implement them on phones and laptops. The results show the efficiency of our protocols.

## ACKNOWLEDGMENT

The research is supported in part by National High-Tech R&D Program of China (863) under grant No. 2011AA010100, National Basic Research Program of China (973) under grant No. 2012CB316200 and the NSFC program under Grant No.61103187, No.61272426, No.61202359, No.61272429. The research of Xiang-Yang Li is partially supported by NSF CNS-0832120, NSF CNS-1035894, NSF ECCS-1247944, National Natural Science Foundation of China under Grant No. 61170216, No. 61228202, China 973 Program under Grant No.2011CB302705.

## REFERENCES

[1] BARANIUK, R. Compressive sensing [lecture notes]. *Signal Processing Magazine*, 2007, pp. 118–121.
[2] DONG, W., DAVE, V., QIU, L., AND ZHANG, Y. Secure friend discovery in mobile social networks. *INFOCOM*, 2011.
[3] FREEDMAN, M., NISSIM, K., AND PINKAS, B. Efficient private matching and set intersection. *Advances in Cryptology-EUROCRYPT*, 2004, pp. 1–19.
[4] GENNARO, R., GENTRY, C., AND PARNO, B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. *Advances in Cryptology–CRYPTO*, 2010, pp 465–482.
[5] HAZAY, C., AND LINDELL, Y. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. *Theory of Cryptography*, 2008, pp. 155–175.
[6] XIANG-YANG LI, YAJUN WANG, AND YU WANG. Complexity of Data Collection, Aggregation, and Selection for Wireless Sensor Networks. *IEEE Transactions on Computers*, 2010, pp. 386–399.
[7] HE, W., LIU, X., NGUYEN, H., NAHRSTEDT, K., AND ABDELZAHER, T. Pda: Privacy-preserving data aggregation in wireless sensor networks. *INFOCOM* , 2007.
[8] LI, M., CAO, N., YU, S., AND LOU, W. Findu: Privacy-preserving personal profile matching in mobile social networks. *INFOCOM*, 2011.
[9] PENG, K., BOYD, C., DAWSON, E., AND VISWANATHAN, K. Robust, privacy protecting and publicly verifiable sealed-bid auction. *Information and Communications Security*, 2002, pp. 147–159.
[10] QI, Y., AND ATALLAH, M. Efficient privacy-preserving k-nearest neighbor search. *ICDCS*, 2008.
[11] RABIN, T., AND BEN-OR, M. Verifiable secret sharing and multiparty protocols with honest majority. *STOC*, 1989.
[12] SHENG, B., AND LI, Q. Verifiable privacy-preserving range query in two-tiered sensor networks. *INFOCOM*, 2008.
[13] CHOR, B., GOLDWASSER, S., MICALI, S., AND AWERBUCH, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. *FOCS*, 1985.
[14] WANG, C., REN, K., YU, S., AND URS, K. Achieving usable and privacy-assured similarity search over outsourced cloud data. *INFOCOM*, 2012.
[15] YAO, A. Protocols for secure computations. *FOCS*, 1982.
[16] YE, Q., WANG, H., AND PIEPRZYK, J. Distributed private matching and set operations. *Information Security Practice and Experience*, 2008, pp. 347–360.
[17] JUNG, T. AND MAO, X.F. AND LI, X.Y AND TANG, S.J. AND GONG, W. AND ZHANG, L. Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation. *INFOCOM*, 2013.
[18] LI, X.Y. AND JUNG, T. Search me if you can: privacy-preserving location query service. *INFOCOM*, 2013.
[19] JUNG, T. AND LI, X.Y AND WAN, Z. AND WAN, M. Privacy preserving cloud data access with multi-authorities. *INFOCOM*, 2013.