2012 32nd IEEE International Conference on Distributed Computing Systems

Privacy Preserving Group Ranking

Lingjun Li, Xinxin Zhao, Guoliang Xue, and Gabriel Silva

***Abstract*—Group ranking is a necessary process used to find the best participant from a group. Group ranking has many applications, including online marketing, personal interests matching and proposal ranking. In an online virtual environment, participants want to do group ranking without leaking any of their private information. In this work, we generalize this scenario as a privacy preserving group ranking problem and formulate the privacy requirements of this problem. We propose a fully distributed privacy preserving group ranking framework and prove its security in the honest but curious model. The core of our framework is a novel multiparty sorting protocol, which guarantees that an adversary cannot link the private information to its owner's identity as long as the owner's final ranking is hidden from the adversary. Our protocol is efficient in computational overhead and communication rounds compared to existing works, as demonstrated by our analysis and simulation.**

I. Introduction

In today's society, everyday transactions are increasingly shifting from the real world to online interactions. This shifting has created a demand for new secure process to preserve people's privacy while in this new virtual environment. In the real world, group ranking is a necessary process used to find the best participant from a group. When picking the best participant, a trivial group ranking approach leaks all the relevant information from all participants. In this paper, we are about to propose a novel group ranking framework for the online interactions while preserve participants' privacy.

Our research has implications in many fields that group ranking is done. One example is online marketing. Consider a famous health and nutrition company that wants to promote a new expensive product in an online community. It invites people selected from a group of participants interested in the promotion in a long-term free trial program. To maximize the marketing effect, people chosen are expected to be typical representatives of the target demographic and will also satisfy the company's marketing strategies. For example, the target demographic is specified by attributes set by the company, like age, blood pressure level, *etc*. The company also wants to know some other attributes of the participants for marketing considerations, like number of friends, annual income, *etc*. Traditionally, the company publishes a questionnaire related to the attributes of interest. Every participant fills the questionnaire with personal information and sends it back to the company. For each participant's information received, the company calculates a gain value, which is referred to as the participant's gain. The company will rank the participants according to gain, then invite the top k ranking participants, in its long-term program and record the accepter's information for research and marketing purpose. The true personal information from an accepter could help the company improve the product and develop its marketing strategies. On the other hand, the true personal information is crucial to avoid potential side effect on an accepter. Each participant will be given their individual ranking to ensure fairness.

However, it is neither secure nor necessary to expose lower ranking (from $(k+1)$-th to the last one) participants' private information, like health conditions, annual income, *etc*, to either the company or other participants. For example, a privacy phisher can participate in an online marketing event and collect people's sensitive information if it is exposed in the process. A desired method is to first let the participants privately rank their gains, then inform everyone the individual ranking and, finally, ask the top k ranking participants to submit their information to the company. The company is not willing to disclose how it calculates the gain from an individual information because the calculation may contain trade secrets, like on what ages the product is to expect to have the best effect. In addition, since only top k ranking participants are of interest, providing a low ranking participant's gain could expose unnecessary private information. Without any privacy guarantees, some people would rather not participate if they are worried about their private information being abused. Therefore, a group ranking method to disclose as little unnecessary private information as possible is essential for online marketing.

Such privacy preserving group ranking method is necessary in many other applications that are concerned with protecting people's privacy. For example, by using the method, a person can find "best matched" people from a group by ranking them according to their preferences to a vector of sensitive attributes, like political preference. Another example can be found in business related OSNs, like LinkedIn, where an employer wants to recruit potential employees from a group of candidates for a position, which has a special requirement on the employer's sensitive health information.

The goal of this work is to construct an efficient distributed group ranking framework to meet the privacy demands without the need of a third trusted party. Secure two party computation protocols [1, 2] can be used to protect private information in computing a participant's gain. However, hiding the resulting gain from all parties and giving sufficient information for ranking is challenging. To rank the participants' gains, a secure multiparty (SMP) sorting protocol is needed. Existing protocols [3, 4] are based on the secret sharing (SS) based

Li, Zhao, Xue, and Silva are all with Arizona State University, Tempe, AZ, 85287. Email: {li.lingjun, zhao.xinxin, xue, gsilva}@asu.edu. This research is supported in part by ARO grant W911NF-09-1-0467, and NSF grants 0901451 and 0905603. The information reported here does not reflect the position or the policy of the federal government.

comparison primitive [5, 6] and suffer heavy computational overhead and many communication rounds. A possible alternative is to utilize homomorphic encryption (HE) [7, 8, 9, 10]. Unfortunately, directly applying a partially HE causes privacy leaks, and using a fully HE is still not yet practical [11, 12]. Hence, the second challenge is to construct an efficient SMP sorting protocol for our problem.

In this paper, we present a construction overcoming the above challenges and make following contributions:

(1) We formulate the privacy preserving group ranking problem in the online virtual environment and give formal definitions to reasonable security requirements based on the indistinguishability.

(2) We propose an efficient distributed framework and prove its security in the honest but curious (HBC) model. Our framework leverages the key idea of anonymous group messaging protocols [13, 14] and makes key modifications to the current partial HE-based secure integer comparison protocols to satisfy privacy requirements.

(3) We introduce a concept of *identity unlinkability* and propose an identity unlinkable multiparty sorting protocol, in which each party is given the ranking of the indivdual input but cannot link the inferred information to its owner's identity. This protocol itself is of independent interest to the study of the SMP sorting problem. The proposed protocol has linear communication rounds and resists up to $n-2$ colluders as long as the honest participants' individual rankings are hidden.

The rest of this paper is organized as follows. We present related work in Section II. The group ranking problem is formulated and privacy requirements are presented in Section III. We review several technical preliminaries in Section IV and give our construction of privacy group ranking framework in Section V. The framework is analyzed in Section VI and simulated in Section VII. We will finish with our conclusions in Section VIII.

II. Related Work

In order to rank participant gains, a privacy preserving group ranking framework needs an efficient protocol to let n parties privately sort n integers, which could be viewed as a special case of the SMP sorting problem. In a general SMP sorting problem, the number of parties may be smaller than n and the sorting result may be hidden from all the parties. Finding an efficient protocol for the SMP sorting problem has been stated as an open problem in [15] and [4]. As far as we know, no efficient solution is given to this problem, especially for the special case needed by the group ranking framework.

Existing SMP sorting protocols are built upon SS-based SMP comparison primitives. We will omit SMP notation if it is clear from context. As far as we know, the most efficient SS-based comparison protocol is given by Nishide and Ohta [5], which reduces a comparison to three interval tests. Each interval test can be done by using SS-based addition and multiplication protocols. Overall, the protocol can compare two l-bit integers at a cost of $279l + 5$ invocations of the SS-based multiplication. On top of this comparison primitive, Jónsson *et al.* recently proposed a general SMP sorting protocol in [3] by embedding comparison primitives into a sorting network. Their sorting network is a variant of the merge sort algorithm and needs $O(n(\log n)^2)$ invocations of the comparison protocol. Based on hash tables and the SS scheme, Burkhart and Dimitropoulos gave a probabilistic construction in [4] to find the top k largest integers. Their protocol was shown to be fast by their simulations, but it cannot be guaranteed to terminate with a correct result every time.

Although SS comparison primitive is unconditionally secure, the present protocol [5] suffers heavy computational overhead and many interactions when group size grows to the number of the integers to be sorted. In addition, this primitive requires the number of colluded parties to be less than $(n-1)/2$ since SS-based multiplication protocol needs $2t+1$ parties to do the *degree reduction* [16] in a (t,n)-SS scheme, where $t-1$ is the maximum number of colluders. Compared with existing SMP sorting protocols, our multiparty sorting protocol in this work has linear communication rounds and resists up to $n-2$ colluders.

An alternative to the SS-based comparison primitive is partially HE-based comparison primitive [8, 17], which is semanticly secure. However, directly utilizing it to construct a group ranking framework would expose a party's relative ranking because they are designed for two party scenarios and the comparison result is given to one of the two parties. A possible modification is to output the encryption of the larger integer to one party and the smaller one to the other party. Because $\max\{a,b\} = (a > b) \cdot (a-b) + b$, this modification needs the underlying HE encryption to be fully homomorphic, i.e. additive and multiplicative. Unfortunately, we are not aware of any practical fully homomorphic encryptions for this purpose thus far [11, 12]. Specific to the group ranking problem, we introduce the concept of identity unlinkability and use partially HE-based comparison to construct an identity unlinkable multiparty sorting protocol.

Our idea of identity unlinkable sorting is inspired by the anonymous group messaging problem, in which a group of members want to submit messages anonymously to a data collector. This problem was first proposed and investigated by Yang *et al.* in [18]. Brickell and Shmatikov revisited this problem and gave an efficient solution in [13] which can resist $n-2$ adversaries in a group of n members. The anonymity of their solution is guaranteed by random shuffles and the underlying IND-CCA2 secure cryptosystem. Corrigan-Gibbs and Ford [14] extended this work to an accountable protocol fitting variant size of messages. We leverage the key idea of the random shuffle in [13] and propose an identity unlinkable multiparty comparison protocol.

III. Problem Formulation

In this section, we present our framework model, adversary model and security goals for a privacy preserving group ranking framework.

A. Framework Model

Our framework is cooperatively executed by $n+1$ parties, an initiator P_0, i.e., the company in our motivation example, and n participants $P_1, \cdots, P_n$. We assume that the participants are willing to accept the initiator's invitation and to submit their personal information if selected eventually. We use an m-dimensional vector of attribute names to describe the questionnaire given by the initiator. The initiator holds an m-dimensional private criterion vector $\mathbf{v}_0$, indicating her preferred value for each attribute, and an m-dimensional weight vector, $\mathbf{w}$, denoting the importance of each attribute. Here, an attribute value and a weight value are d_1-bit and d_2-bit unsigned integers, respectively. We abstract the answers of a participant P_j to the questionnaire as an m-dimensional individual information vector $\mathbf{v}_j$, each item of which corresponds to an answer value.

The initiator has different expectations on different kinds of attributes. For some attributes, the initiator expects it is not less than some threshold and the more it exceeds the threshold the better. We call such attributes "*greater than*" attributes. For example, the number of a participant's friends is a "greater than" attributes in our motivation example. For other attributes, the initiator hopes it is around a particular value. Such attributes are called "*equal to*" attributes. Age and blood pressure level are "equal to" attributes. This work will focus on these two kinds of attributes because they are sufficient to solve our motivation problem and have a wide range of applications. without loss of generality, we assume the first t ($0 \leq t \leq m$) dimensions of a vector are "equal to" attributes and the rest are "greater than" attributes. To satisfy the above requirements, we define a participant's gain as follows.

Definition 1. Given a criterion vector $\mathbf{v}_0 = [v_1^0, v_2^0, \cdots, v_m^0]^\mathsf{T}$ and a weight vector $\mathbf{w} = [w_1, w_2, \cdots, w_m]^\mathsf{T}$, the gain value of P_j is

$$g_j = \sum_{k=t+1}^{m} w_k(v_k^j - v_k^0) - \sum_{k=1}^{t} w_k(v_k^j - v_k^0)^2 \ .$$

where $\mathbf{v}_j = [v_1^j, v_2^j, \cdots, v_m^j]^\mathsf{T}$ is P_j's information vector. $\square$

We note that a partial gain

$$p_j = \sum_{k=t+1}^{m} w_k v_k^j - \sum_{k=1}^{t}(w_k(v_k^j)^2 - 2w_k v_k^j v_k^0)$$

is sufficient for group ranking and hides part of criterion information. A partial gain is a signed integer of less than $(\lceil \log m \rceil + d_1 + 2d_2 + 2)$-bit length (including the sign bit). In our framework, we convert l-bit signed integers to l-bit unsigned integers by adding 2^{l-1} to each integer. This conversion retains the order of the original integers. The partial gain of P_j can be represented by dot products, $\mathbf{wg} \cdot \mathbf{vg}_j - \mathbf{we} \cdot (\mathbf{ve}_j * \mathbf{ve}_j) + 2(\mathbf{we} * \mathbf{ve}_0) \cdot \mathbf{ve}_j$. Here, operation $*$ is the element-wise multiplication of two vectors, $\mathbf{a} * \mathbf{b} = [a_1 b_1, \cdots, a_m b_m]^\mathsf{T}$. $\mathbf{ve}_0$ is the "equal to" part of the criterion vector and $\mathbf{vg}_0$ is the "greater than" part. $\mathbf{we}$, $\mathbf{wg}$, $\mathbf{ve}_j$ and $\mathbf{vg}_j$ are defined in this similar fashion.

We now give the formal definition of the group ranking framework.

Definition 2. A group ranking framework is a framework consisting of an initiator and n participants. The initiator publishes a parameter k and holds a criterion vector and a weight vector. Every participant holds an information vector. The framework calculates each participant's gain defined in Def. 1 and ranks the identities $\{P_i\}_{i=1}^n$ according to gain, resulting in an ordered participant identities $\{P_{\pi(1)}, \cdots, P_{\pi(n)}\}$. Here, $\pi(\cdot)$ is a permutation of the sequence $1, \cdots, n$. Eventually, it outputs the ranking i to the participant $P_{\pi(i)}$ and top k ranking participants and corresponding information vectors to the initiator. $\square$

In this work, we rank the participants in non-increasing order of gains. In other words, P_i is ranked higher than P_j if $g_i \geq g_j$.

We assume that there exists a secure channel between each pair of parties and all framework messages are transmitted through the secure channels. Such secure channels can be established by using public key cryptosystems. We *do not* assume a trusted third party during an instance of the framework.

B. Adversary Model

An active attack from outside of the framework could modify, inject, block and replay framework messages. However, the main purpose of this work is to defend against a passive inside adversary whose goal is to get the honest party's private information. This attack is formalized as the so-called HBC model. An adversary in the HBC model does not deviate from the protocols in a framework. However, she may remember all the messages she sends and receives or collude with other HBC adversaries to infer the honest user's private information. When there are more than one adversary, we consider that they collaborate and are controlled by a same adversary.

We consider the adversary is computationally bounded since a powerful adversary with unbounded computation resources is actually very uncommon in practice. An adversary in our framework could carry out any probabilistic polynomial time (P. P. T.) algorithm.

In summary, we consider the security of our protocol in HBC model with a computationally bounded adversary.

C. Security Goals

We state our security goals from the aspect of protecting an honest low ranking participant (ranking between $(k+1)$ and n). Hence, an honest participant in our security goals is an honest low ranking participant if it is not specified. According to definition 2, no framework can protect top k ranking participants from an initiator controlled by an adversary since their private information eventually goes to the initiator. However, we note that a framework achieving our security goals also protects an honest participant ranking higher than k as long as the initiator is not controlled by an adversary because, in this case, a top-k ranking participant exposes the same amount of private information as a low ranking participant does and actually acts the same as a low ranking participant in the view of the adversary. This is true in many cases in which the initiator is a prestige online entity, like a famous health and nutrition company in our motivation example.

The first security goal is to protect an honest party's private vector (i.e., information vector, criterion vector and weight vector). We thus define the security of input data as:

Definition 3. (private input hiding): A group ranking framework is private input hiding if an adversary is not able to get an honest party's private vector. $\square$

A participant and the initiator will cooperatively calculate the gain but we do not want either of them to know the result.

Definition 4. (gain computation secure) A group ranking framework is gain computation secure if a participant is not able to know her gain when the initiator is honest and an initiator is not able to know any honest participant's gain. $\square$

Our next security goal is to protect an honest participant's gain from an adversary if there is only one honest participant. We define *gain hiding* using an adversary's advantage winning the following game between the adversary and a framework oracle under some security parameter $1^{\mathcal{K}}$, which is denoted as $\mathcal{O}(1^{\mathcal{K}})$. Without loss of generality, we assume the honest participant is P_1 and all the rest parties are controlled by the adversary.

Definition 5. (gain hiding) An adversary $\mathcal{A}$ sets the private vectors for parties $P_0, P_2 \cdots, P_n$ and gives two vectors $\bar{\mathbf{v}}_0$ and $\bar{\mathbf{v}}_1$ to $\mathcal{O}(1^{\mathcal{K}})$ such that

$$g_i < \bar{g}_0 < g_{i+1} \Leftrightarrow g_i < \bar{g}_1 < g_{i+1},\ 2 \leq i \leq n. \quad (1)$$

$\mathcal{O}(1^{\mathcal{K}})$ chooses a random bit b from set $\{0,1\}$ (denoted by $b \leftarrow_R \{0,1\}$), sets $\bar{\mathbf{v}}_b$ as P_1's information vector and interacts with $\mathcal{A}$ on behalf of P_1. $\mathcal{A}$ wins the game if Condition (1) holds, $\bar{\mathbf{v}}_b$ is not given to $\mathcal{A}$ and the following value

$$\Pr[\mathcal{A}(1^{\mathcal{K}}, \sigma, b=1)=1] - \Pr[\mathcal{A}(1^{\mathcal{K}}, \sigma, b=0)=1] \quad (2)$$

is a non-negligible function in security parameter, where $\mathcal{A}(1^{\mathcal{K}}, \sigma,\ b=1)=1$ denotes that algorithm $\mathcal{A}$ outputs 1 when the security parameter is $1^{\mathcal{K}}$, her observation from the framework is σ and b is 1. The result of equation (2) is called $\mathcal{A}$'s *distinguishing advantage* or advantage. A framework is gain hiding if for any P. P. T. algorithm $\mathcal{A}$, her advantage is always negligible, i.e., no P. P. T. $\mathcal{A}$ can win the game. $\square$

Definition 6. (gain secure) A group ranking framework is gain secure if it is gain computation secure and gain hiding. $\square$

Definition 5 indicates that it is intractable for an adversary to learn any information about an honest user's gain value except which integer interval it belongs to. The broadness of an integer interval depends on the adversary's gain values. In fact, such privacy leaking is inevitable in a group ranking framework if there is only one honest participant because its final ranking is exposed to the adversary. However, the private information is useless unless one can figure out the owner's identity. Therefore, we introduce the concept of "*identity unlinkability*", which guarantees that an adversary cannot link the obtained private information to its owner's identity as long as there exist at least two honest participants, i.e., each honest participant's individual final ranking is hidden from the adversary. The formal definition is as follows.

Definition 7. (identity unlinkability) $\mathcal{A}$ chooses two honest participants P_i and P_j, and two private vectors $\bar{\mathbf{v}}_0$ and $\bar{\mathbf{v}}_1$. Framework oracle $\mathcal{O}(1^{\mathcal{K}})$ selects a bit $b \leftarrow_R \{0,1\}$ and sets the information vector of P_i (resp. P_j) as $\bar{\mathbf{v}}_b$ (resp. $\bar{\mathbf{v}}_{(1-b)}$). $\mathcal{O}(1^{\mathcal{K}})$ sets the private vectors of the rest honest parties (may not including the initiator) according to the vectors given by $\mathcal{A}$. $\mathcal{O}(1^{\mathcal{K}})$ then participates in the framework interacting with $\mathcal{A}$ on behalf of all the honest parties. After observing the running of protocol *except* the final rankings and information vector of P_i and P_j, $\mathcal{A}$ outputs her guess. We say a framework is identity unlinkable if for any P. P. T. algorithm $\mathcal{A}$,

$$\Pr[\mathcal{A}(1^{\mathcal{K}}, \sigma, b=1)=1] - \Pr[\mathcal{A}(1^{\mathcal{K}}, \sigma, b=0)=1]$$

is a negligible function in $\mathcal{K}$. $\square$

Definition 8. A privacy preserving group ranking framework is a group ranking framework that is private input hiding, gain secure, and identity unlinkable. $\square$

In summary, a privacy preserving group ranking framework would protect an honest party's private input, hide an honest participant's gain value if only its final ranking is exposed, and hide its identity if its final ranking is hidden from the adversary. We emphasize that a privacy preserving framework also protects an honest top-k ranking participant's private information if the initiator is honest.

D. Efficiency Goals

For an online framework, it is desirable to have as few communication rounds among parties as possible and moderate computational overhead.

IV. Preliminaries

In this section, we will review existing research works needed for our framework.

A. Secure Two-Party Dot Product Protocol

For HBC adversaries, Ioannidis *et al.* proposed an efficient protocol in [2], letting two parties, Alice and Bob, securely calculate $\mathbf{w} \cdot \mathbf{v}$. In this case, $\mathbf{w}$ and $\mathbf{v}$ are two vectors with same dimensions held by two parties separately.

- Bob, holding a $(d-1)$-dimensional vector $\mathbf{w}$, chooses a random matrix Q of $s \times s$ dimensions, where s is a random integer. She generates another $(s \times d)$-dimensional matrix X, for which the r-th row is vector $[\mathbf{w}^{\mathsf{T}}, 1]^{\mathsf{T}}$ and the rest of the numbers in the matrix are chosen randomly. Here, r is a random integer in $[1, s]$. Bob then calculates $b = \sum_{i=1}^{s} Q_{ir}$ and $\mathbf{c} = \sum_{i=1, i \neq r}^{s} (\mathbf{x}_i^{\mathsf{T}} \cdot \sum_{j=1}^{s} Q_{ji})$, where

$\mathbf{x}_i^\top$ is the i-th row of the matrix X. Choosing a random d-dimensional vector $\mathbf{f}$ and three random numbers R_1, R_2, R_3, Bob sends QX, $\mathbf{c}' = \mathbf{c}+R_1 \cdot R_2 \cdot \mathbf{f}^\top$ and $\mathbf{g} = R_1 \cdot R_3 \cdot \mathbf{f}$ to the other party, Alice.

- Alice generates a vector $\mathbf{v}' = [\mathbf{v}^\top, \alpha]^\top$, where α is a random number and $\mathbf{v}$ is a $(d-1)$-dimensional vector. Upon receiving data from Bob, Alice computes $\mathbf{y} = QX\mathbf{v}'$, $z = \sum_{i=1}^{s} y_i$, $a = z - \mathbf{c}' \cdot \mathbf{v}'$, $h = \mathbf{g}^\top \cdot \mathbf{v}'$ and sends a, h to Bob.
- Bob computes $\beta = (a + h \cdot \frac{R_2}{R_3})/b$, which is $\mathbf{w} \cdot \mathbf{v} + \alpha$.
- They finally exchange α and β. The desired dot product is $\beta - \alpha$.

The security of a party's input vector depends on the fact that an adversary cannot solve the linear system of equations if the number of equations is less than the number of unknowns [2].

B. Complexity Assumption

Let $\mathcal{G}(1^{\mathcal{K}})$ be a polynomial time algorithm which generates a multiplicative group $\mathbb{G}_q$ with a prime order q, given a security parameter $1^{\mathcal{K}}$. Our framework needs a group such that the decisional Diffie-Hellman (DDH) problem is hard for it. We say the DDH problem is hard for a group $\mathbb{G}_q$ if for any P. P. T. adversary $\mathcal{A}$, the distinguishing advantage of two tuples $(\mathbb{G}_q, g, g^a, g^b, g^{ab})$ and $(\mathbb{G}_q, g, g^a, g^b, g^d)$,

$$\Pr[\mathcal{A}(\mathbb{G}_q, g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(\mathbb{G}_q, g, g^a, g^b, g^d) = 1],$$

is a negligible function in $\mathcal{K}$, where g is a group generator and a, b, d are random elements from $\mathbb{Z}_q$.

Here are two popular groups for which the DDH problem is believed to be hard.

- DL: The group of quadratic residues modulo a safe prime.
- ECC: A prime order subgroup of an elliptic curve with a large embedding degree.

C. IND-CPA Security

IND-CPA stands for indistinguishability under chosen plaintext attack and is defined by a game between a public key encryption oracle and an adversary [19]. Provided a security parameter, the oracle generates a public key and a private key, giving the public key to the adversary. The adversary may do any number of computations, including encryptions. She submits two distinct new messages M_0 and M_1 to the oracle. The oracle encrypts them as $\mathcal{E}_0$, $\mathcal{E}_1$, picks a random bit $b \leftarrow_R \{0, 1\}$ and sends $(\mathcal{E}_b, \mathcal{E}_{1-b})$ back to the adversary. The adversary may do another number of computations (excluding encryption of M_0 and M_1) and eventually makes a guess of b. We say the encryption is IND-CPA secure if for any P. P. T. adversary, the advantage of guessing correct b is a negligible function in the security parameter.

D. ElGamal Cryptosystem

Group $\mathbb{G}_q$ is a prime order multiplicative group for which the DDH problem is hard and g is a generator in $\mathbb{G}_q$. Then, the ElGamal cryptosystem is described as:

- **Key generation:** a private key is a random element $x \leftarrow_R \mathbb{Z}_q$ and the corresponding public key is $y = g^x$.
- **Encryption:** a ciphertext of a message M is of form $E(M) = (My^r,\ g^r)$, where $r \leftarrow_R \mathbb{Z}_q$.
- **Decryption:** the decryption of a ciphertext $(c,\ c')$ is done by $M = c'/c^x$.

If we modify the encryption as $E(M) = (g^M y^r,\ g^r)$, then the ElGamal encryption turns to be an additive homomorphic encryption because $E(M_1) \circ E(M_2) \triangleq (c_1 c_2,\ c_1' c_2') = E(M_1 + M_2)$. Decryption in this case is difficult or impossible because recovering M from g^M is hard in group $\mathbb{G}_q$. However, this does not negatively affect our framework since we only need to verify if $M = 0$, i.e., $g^M = 1$. We note that this modified ElGamal cryptosystem is IND-CPA secure in group $\mathbb{G}_q$.

The ElGamal encryption can be done in distributed way by letting each party choose $x_i \leftarrow_R \mathbb{Z}_q$ and publish $y_i = g^{x_i}$. The joint public key then becomes $y = \prod_{i=1}^{n} y_i$. A ciphertext $(c,\ c')$ encrypted by the joint public key can be decrypted by $c/\prod_{i=1}^{n} c'^{x_i}$.

E. Zero-knowledge Proof Protocol

A zero-knowledge proof [7] protocol is executed by two parties, a prover and a verifier, at the end of which the verifier is convinced that the prover knows some knowledge without learning any information about the knowledge. We are interested in the proof of the discrete logarithm in the HBC model and it is well known that the Schnorr identification scheme [20] is an honest verifier zero-knowledge (HVZK) proof of $x = \log_g y$ [21], where x, y are two elements in a multiplicative group $\mathbb{G}_q$, g is a generator and $\log_g$ denotes the discrete logarithms based g in G. The protocol works as follows:

- The prover chooses a random number $r \leftarrow_R \mathbb{Z}_q$ and sends $h = g^r$ to the verifier.
- The verifier chooses a random number $c \leftarrow_R \mathbb{Z}_q$ and sends back to the prover.
- The prover calculates $z = (r + xc \mod q)$ and sends z to the verifier.
- The verifier verifies $g^z = hy^c$.

An honest prover can always convince an honest verifier and a cheating prover can convince an honest verifier with probability no more than $1/q$. The protocol is zero-knowledge as long as the verifier is honest [21]. We extend this protocol to n verifiers. The new protocol varies at a new second step, in which every verifier selects $c_j \leftarrow_R \mathbb{Z}_q$ and publishes it. The prover calculates $z = (r + x\sum_{j=1}^{n} c_j \mod q)$ and publishes it. Every verifier verifies $g^z = h\prod_{j=1}^{n} y^{c_j}$. The security proof of this protocol, including completeness, special soundness and HVZK can be obtained by extending the original proof of Schnorr protocol. Here, we just briefly show the proof of special soundness [22] by constructing a knowledge extractor, which extracts the knowledge x from a zero-knowledge proof. Given two transcripts $(h, \{c_j\}_{j=1}^{n}, z)$ and $(h, \{c_j'\}_{j=1}^{n}, z')$ (passing the verifier's final verification), the knowledge x can be extracted by computing $(z - z')/(\sum_{j=1}^{n} c_j - \sum_{j=1}^{n} c_j')$ $\mod q$ since the prover commits to the same randomness r.

V. Our Construction

We are now ready to present our framework. There are three phases in the framework: secure gain computation, unlinkable gain comparison and ranking submission. A participant and the initiator cooperatively compute the participant's partial gain (see Section III) in the first phase. All participants then compare partial gains in the unlinkable gain comparison phase and obtain their individual rankings. In the last phase, the participants submit their rankings and information to the initiator. The framework is presented in Fig. 1.

P_0 generates a group $\mathbb{G}_q \leftarrow \mathcal{G}(1^{\mathcal{K}})$, picks a generator g and publishes them. P_0 also publishes a vector of attribute names and an integer k, where $1 \leq k \leq n$.
Private Input: $\mathbf{v}_0$ and $\mathbf{w}$ from P_0; $\mathbf{v}_j$ from participant P_j, $1 \leq j \leq n$.

Secure gain computation:

1) P_0 chooses a random h-bit integer ρ.
2) Every participant P_j generates $\mathbf{w}'_j = [\mathbf{vg}_j^{\mathsf{T}}, (\mathbf{ve}_j * \mathbf{ve}_j)^{\mathsf{T}}, \mathbf{ve}_j^{\mathsf{T}}, 1]^{\mathsf{T}}$. As in the dot product protocol of Sec. IV-A, P_j computes QX, $\mathbf{c}'$, $\mathbf{g}$ and sends them to P_0.
3) Upon receiving $(Q_jX_j, \mathbf{c}'_j, \mathbf{g}_j)$ from a participant P_j, P_0 chooses $\rho_j \leftarrow_R \{0, 1, \cdots, \rho\}$ and constructs $\mathbf{v}'_j = [\rho\mathbf{wg}^{\mathsf{T}}, -\rho\mathbf{we}^{\mathsf{T}}, 2\rho(\mathbf{we} * \mathbf{ve}_0)^{\mathsf{T}}, \rho_j]^{\mathsf{T}}$. P_0 computes $a_j = z_j - \mathbf{c}'_j \cdot \mathbf{v}'_j$, $h_j = \mathbf{g}_j^{\mathsf{T}} \cdot \mathbf{v}'_j$ and sends them back to P_j.
4) Upon receiving (a_j, h_j) from P_0, P_j calculates $\beta_j = (a_j + h_j \cdot R_2/R_3)/b$ and converts it to an unsigned integer (see Sec. III-A).

Unlinkable gain comparison:

5) $\forall 1 \leq j \leq n$, P_j picks private key $x_j \leftarrow_R \mathbb{Z}_q$ and publishes $y_j = g^{x_j}$. P_j proves the knowledge of x_j to the rest of parties (Sec. IV-E).
6) Each participant P_j represents her β_j in binary bits $[\beta_j]_B = [\beta_j^l, \beta_j^{l-1}, \cdots, \beta_j^1]$, encrypts and publishes them as $E(\beta_j)_B = [E(\beta_j^l), \cdots, E(\beta_j^1)]$. Here, the encryption is done using joint key $y = \prod_{j=1}^{n} y_j$.
7) Each participant P_j gets encrypted data $\{E(\beta_i)_B\}_{i=1,i\neq j}^{n}$ from others. For each encrypted data $E(\beta_i)_B$, P_j does following calculation for $1 \leq t \leq l$:
   - $E(\gamma_i^t) = E(\beta_j^t + \beta_i^t - 2\beta_j^t\beta_i^t)$, where $\gamma_i^t = \beta_j^t \oplus \beta_i^t$;
   - $E(\omega_i^t) = E((l - t + 1) - \sum_{v=t+1}^{l}(\gamma_i^t - \gamma_i^v) - \gamma_i^t)$;
   - $E(\tau_i^t) = E(\omega_i^t + \beta_i^t)$.

   P_j sends all the ciphertexts $\mathcal{E}_j = \{e : e \in E(\tau_i) \wedge 1 \leq i \leq n \wedge i \neq j\}$ to P_1, where $E(\tau_i) = \{E(\tau_i^t)\}_{t=1}^{l}$.
8) After receiving the ciphertext sets from all the rest parties, P_1 constructs a vector $V = [\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_n]$. Starting at P_1, each participant P_j applies following steps to each element $\mathcal{E}_i$ $(i \neq j)$ in V:
   - For each ciphertext $(c_t, c'_t) \in \mathcal{E}_i$, replaces c_t by $\tilde{c}_t = c_t/(c'_t)^{x_j}$. Picks $r \leftarrow_R \mathbb{Z}_q$ and updates the ciphertext with $((\tilde{c}_t)^r, (c'_t)^r)$.
   - Permutes the ciphertexts in each set $\mathcal{E}_i$.

   P_j then sends the permuted vector V to P_{j+1}. If P_j is the last one, P_n, she sends the element of the vector back to the corresponding participant.

Ranking Submission:
Upon receiving the final result $\tilde{E}_j$ from P_n, participant P_j $(1 \leq j \leq n)$ decrypts each element ciphertext (c, c') by using $g^m = c/c'^{x_j}$ and checks $g^m = 1$. Let d be the number of zeroes and then the ranking of P_j is $d_j = d + 1$. If $d_j \leq k$, P_j submits $\mathbf{v}_j$ to P_0 as well as the ranking.

Fig. 1. A privacy preserving group ranking framework

In the first phase, each participant P_j invokes the secure dot product protocol in Section IV-A with the initiator P_0 to calculate the dot product, $\mathbf{w}'_j \cdot \mathbf{v}'_j$. The result, output to P_j, is a masked partial gain $\beta_j = \rho p_j + \rho_j$, where ρ, ρ_j are two random integers and $0 \leq \rho_j < \rho$. This technique was proposed and used in [1]. ρ stays consistent for all participants and ρ_j varies for different participants. They are used to hide the partial gain and the private inputs (Lemma 1 and 3). ρ and ρ_j are randomly chosen by the initiator. ρ is of h-bit length and the resulting β_j is a l-bit signed integer, where $l = h + \lceil \log m \rceil + d_1 + 2d_2 + 2$. This number l can be estimated by the initiator and is broadcast to all participants at the beginning of the phase. We note that the β values keep the order of the partial gains in the sense that if $p_i > p_j$ (resp. $p_i < p_j$), $\beta_i > \beta_j$ (resp. $\beta_i < \beta_j$). If $p_i = p_j$, it does not matter if P_i ranks higher or lower than P_j. Prior to the second phase, every participant converts her β value to an l-bit unsigned integer (see Section III-A).

At the beginning of the unlinkable gain comparison phase, every participant generates a key pair of the ElGamal cryptosystem and proves the knowledge of the private key via the multiple verifiers zero-knowledge proof protocol presented in Section IV-E. The joint public key is $y = \prod_{j=1}^{n} y_j$ and no one knows the corresponding secret key, which is $x = \sum_{j=1}^{n} x_j$. The encryption algorithm used in this phase is the modified ElGamal encryption specified in Section IV-D.

The basic idea is to let each participant blindly compare her own β_j with other β values. First, given two l-bit integers in binary representation $[\beta_j]_B = [\beta_j^l, \beta_j^{l-1}, \cdots, \beta_j^1]$ and $[\beta_i]_B = [\beta_i^l, \beta_i^{l-1}, \cdots, \beta_i^1]$, the comparison can be done by calculating $\gamma^t = \beta_i^t \oplus \beta_j^t$, $\omega^t = (l - t + 1) - \sum_{v=t+1}^{l}(\gamma^t - \gamma^v) - \gamma^t$, $\tau^t = \omega^t + \beta_j^t$ for all $1 \leq t \leq l$. If there is a 0 in the final τ values, β_j is smaller than β_i. We note that there is at most one 0 in the τ values. All of the above operations can be interpreted into a combination of additions, subtractions and scalar multiplications of bits. Due to the additive homomorphism of the modified ElGamal encryption, a participant can carry out the calculation over the received n ciphertext vectors and get encrypted τ values. Eventually, P_j's ranking in the group is reflected by the number of zeroes obtained from the comparisons.

However, no one can decrypt them individually because the secret key is spread out across all n participants. A participant then has to ask all other participants to partially decrypt the ciphertext one by one and eventually obtain the τ values (actually g^τ) by getting rid of her own encryption. When a participant is asked by someone else to partially decrypt ciphertexts, she takes this chance to randomly permute the ciphertexts so that the plaintext sequence does not leak the relative ranking information of the participants.

When a participant knows the individual ranking, she submits the individual information to the initiator if it is in top k rankings. A low ranking participant may over claim her ranking. Although this is an active attack and not the main focus of our framework, we argue that this attack can be detected because the selected participant has to submit her information vector and the initiator will then be able to recalculate its gain.

We note that the participants with the same β value calculate same group ranking at the end of the second phase. If there are some participants who have the same β value as the participant who has ranking of k in the group, our framework takes them all as eligible candidates, submits to the initiator and leaves the decision to the initiator. This is reasonable because these participants actually have the same gain value, which is of more interest than the ranking itself in our motivation example. At the same time, the probability of a participant having the same β value as the k-th participant's is $1/2^l$, which is extremely small when l is sufficiently large.

VI. Analysis of the Framework

A. Security

In this section, we will prove that our group ranking framework satisfies all the privacy requirements defined in Section III-C. Intuitively speaking, an adverseary does not know the private input and the resulting gain because they are protected by the secure dot product protocol and the random numbers ρ, ρ_j (Lemma 1). Secondly, an honest participant's original gain value is hidden from the adversary because it is encrypted using the modified ElGamal encryption (Lemma 3). Finally, an honest participant's random permutation (step 8) breaks the original order of ciphertexts and thus hides relative rankings from the adversary (Lemma 4). As in the definition, the honest participant in this section is an honest low ranking participant (ranking lower than $(k+1)$-th one). We start our formal proof with the following lemma.

Lemma 1. *The group ranking framework presented in Fig. 1 is private input hiding.* □

Proof: First, it has been shown in [2] that the immediate results in dot product protocol, i.e., Q_jX_j, $\mathbf{c}'_j$, $\mathbf{g}_j$, a_j, and h_j, do not leak information about input vectors. Therefore, a curious initiator, controlled by an adversary, cannot learn any information about an honest participant's information vector.

We then consider the privacy of the initiator's private vectors. In this case, an HBC adversary controls all participants and tries to calculate the criterion vector $\mathbf{v}_0$ and weight vector $\mathbf{w}$. She cannot get them from a single β_j value because both ρ and ρ_j are unknown. An adversary may pool all n β values together, trying to calculate $\mathbf{v}_0$ and $\mathbf{w}$. However, she even cannot decide the partial gains since there are n equations and at least $n+1$ unknowns in the linear system. Therefore, the adversary cannot get $\mathbf{v}_0$ and $\mathbf{w}$.

In summary, our protocol is private input hiding. ■

In our framework, we encrypt the binary representation of an integer instead of the integer itself. The following lemma guarantees that this does not hurt the security that is needed in our proofs.

Lemma 2. *The bit-wise ElGamal encryption described in step (6) of Fig. 1 is IND-CPA secure provided that modified ElGamal encryption is IND-CPA secure.* □

Proof: We prove by contradiction. If there exists a P. P. T. adversary algorithm $\mathcal{A}$ who can win the IND-CPA game interacting with bit-wise ElGamal encryption oracle $\mathcal{O}_{\mathrm{b}E}$, then we can construct a simulator $\mathcal{S}$ running $\mathcal{A}$ as an internal procedure to win the IND-CPA game with modified ElGamal (Section IV-D) encryption oracle $\mathcal{O}_{\mathrm{E}}$.

We briefly describe the construction of $\mathcal{S}$, which simulates bit-wise encryption oracle $\mathcal{O}_{\mathrm{b}E}$ for $\mathcal{A}$. $\mathcal{S}$ gets public key y from $\mathcal{O}_{\mathrm{E}}$ and sends it to $\mathcal{A}$ as its public key. When $\mathcal{S}$ gets two distinct messages m_0 and m_1 (both are of l bits) from $\mathcal{A}$, it decomposes them as binaries $[m_0]_B = [m_0^l, \cdots, m_0^1]$ and $[m_1]_B[m_1^l, \cdots, m_1^1]$. Let $[d]_B = [m_0^l \oplus m_1^l, \cdots, m_0^1 \oplus m_0^1]$. $\mathcal{S}$ sends two messages $m'_0 = 0$ and $m'_1 = 1$ to $\mathcal{O}_{\mathrm{E}}$ and gets back two ciphertexts $E(m'_b) = (c_b, c'_b)$ and $E(m'_{1-b}) = (c_{1-b}, c'_{1-b})$, where b is $\mathcal{O}_{\mathrm{E}}$'s random choice. $\mathcal{S}$ returns two ciphertext vectors $[e^l, \cdots, e^1]$ and $[f^l, \cdots, f^1]$ to $\mathcal{A}$. For $t = l, \cdots, 1$,

$$e^t = \begin{cases} E(m_0^t), & \text{for } \ d^t = 0 \\ (c_b y^{r_1^t}, c'_b g^{r_1^t}), & \text{for } \ d^t = 1 \wedge m_0^t = 0 \\ (c_{1-b} y^{r_1^t}, c'_{1-b} g^{r_1^t}), & \text{for } \ d^t = 1 \wedge m_0^t = 1 \end{cases},$$

$f^t = e^t$ if $d^t = 0$ and $f^t = ((c_{1-b_e} y^{r_2^t}, c'_{1-b_e} g^{r_2^t}))$ where b_e is the b chosen in e^t. Here, r_1^t and r_2^t are randomly picked from $\mathbb{Z}_q$ each time. When $\mathcal{A}$ makes its guess $\bar{b}$, $\mathcal{S}$ constructs its guess $\bar{b}'$ as $\bar{b}$ if $m_0 < m_1$ or $1 - \bar{b}$ otherwise and sends it to $\mathcal{O}_{\mathrm{E}}$. If $\mathcal{A}$ makes a right guess, then $\bar{b}'$ is equal to b. It is not hard to see that $\mathcal{A}$ has the same observation as in the real bit-wise encryption and thus helps $\mathcal{S}$ to win the game with $\mathcal{O}_{\mathrm{E}}$. This contradicts to our condition that modified ElGamal cryptosystem is IND-CPA secure and the statement holds. ■

Based on the above lemma, we have

Lemma 3. *The group ranking framework presented in Fig. 1 is gain secure.* □

Proof: First, the initiator knows nothing about an honest participant's gain due to the security of dot product protocol [2]. A curious participant gets a randomized partial gain $\rho p_j + \rho_j$ and cannot solve p_j even if she colludes with all other participants. The gain value is thus hidden to both sides. Therefore, the framework is gain computation secure.

Next, we prove that it is also gain hiding by reducing IND-CPA game of the bit-wise ElGamal encryption to the game defined in Definition 5. Assume that there exists a P. P. T. algorithm $\mathcal{A}$ that has a non-negligible advantage in the gain hiding game of definition 5. We will construct a simulator $\mathcal{S}$ that can win IND-CPA game interacting with oracle $\mathcal{O}_{\mathrm{bE}}$. The simulator runs $\mathcal{A}$ as an internal procedure and sets its random tape. $\mathcal{S}$ learns $\bar{\mathbf{v}}_0$, $\bar{\mathbf{v}}_1$ from $\mathcal{A}$, uses the vector $\bar{\mathbf{v}}_0$ as P_1's information vector and gets β_0. $\mathcal{S}$ calculates β_1 for $\bar{\mathbf{v}}_1$ if it can. Otherwise, $\mathcal{S}$ rewinds $\mathcal{A}$, sends $\bar{\mathbf{v}}_1$ to P_0 as P_1's information vector and gets β_1. In these two steps, $\mathcal{A}$ sets the same criterion vector for P_0 and chooses the same random number ρ because it *honestly* reads the random tape, which stays same. Simulator $\mathcal{S}$ gets a public key from $\mathcal{O}_{\mathrm{b}E}$ and publishes it as P_1's choice of public key. Then simulator submits two messages β_0 and β_1 to $\mathcal{O}_{\mathrm{b}E}$ and obtains two ciphertext vectors $E(\beta_b)_B$, $E(\beta_{1-b})_B$. It converts $E(\beta_b)_B$ to a ciphertext vector under joint key y by using the private keys of all other participants, that are extracted from zero-knowledge proofs (Section IV-E). This would not alter the adversary's observation by a non-negligible number because the knowledge error of the zero knowledge proof is negligible. Result ciphertext vector is published by the simulator on behalf of P_1.

The simulator then follows the protocol until step (8), in which it needs to "partially decrypt" ciphertexts using x_1. The simulator does not have x_1, but it knows how many 0's in the τ values calculated by each participant, because it has the private keys of all the rest participants and thus can get all the β values. $\mathcal{S}$ first uses the β values to verify Condition (1) in Definition 5 and aborts if it is not true. For each participant, the simulator creates a new ciphertext sequence $\mathcal{E}$ by encrypting same number of 0's and random numbers for rest non-zeroes using the colluders' joint public key $\prod_{j=2}^{n} y^j$. After randomly permuting each ciphertext sequence, the simulator hands it to the adversary on behalf of P_1. This does not change the adversary's observation by a non-negligible number due to following reasons: 1) $\bar{g}_0$ and $\bar{g}_1$ are in a same interval so that they result in same 0's in the τ values; 2) Non zero τ values have been randomized in the framework so that encrypting random integers is safe; 3) the positions where zeroes appear are randomly permuted. Simulator follows the protocol until the end. Then, $\mathcal{S}$ outputs what $\mathcal{A}$ outputs to $\mathcal{O}_{bE}$.

In at most two steps, $\mathcal{S}$ produces a valid conversation that is indistinguishable from the one between the real framework oracle and the adversary in $\mathcal{A}$'s view. Since $\mathcal{A}$ wins the gain hiding game, $\mathcal{S}$ now wins the IND-CPA game. This contradicts Lemma 2. Therefore, our framework is gain hiding. ■

Lemma 4. *The group ranking framework presented in Fig. 1 is identity unlinkable.* □

Proof: The proof follows the similar idea of the previous proof and we only describe the different simulator actions here.

The two vectors given by $\mathcal{A}$ are $\bar{\mathbf{v}}_0$ $\bar{\mathbf{v}}_1$. The two honest participants pointed by $\mathcal{A}$ are P_i and P_j. The simulator obtains corresponding β_0 and β_1 by handing them to the initiator on behalf of P_i and P_j. The simulator submits β_0, β_1 to $\mathcal{O}_{bE}$, obtains $E(\beta_b)_B$ and $E(\beta_{1-b})_B$, gives $E(\beta_b)_B$ (resp. $E(\beta_{1-b})_B$) to P_i (resp. P_j), and follows the protocol on behalf of all honest parties. This does not change the observation of the adversary because the initiator (controlled by adversary) is unaware of the information of the participant's gains (gain computation secure).

Still, the simulator cannot decrypt in step (8). We note that the simulator now can obtain the private keys of all participants and β values as in the previous proof. If either P_i or P_j receives the message at step (8), the simulator recreates another n encryptions for each participant as in the previous proof (using the joint public key of the rest waiting participants), permutes each ciphertext sequence, and sends them to the next participant. It does not matter how $E(\beta_b)_B$ and $E(\beta_{1-b})_B$ are assigned between P_i and P_j because the sequence of encryptions is randomly permuted and the adversary cannot distinguish them. Then simulator just follows the protocol to finish the framework except submitting the rankings of P_i and P_j to the adversary and gives whatever $\mathcal{A}$ outputs to the bit-wise ElGamal encryption oracle.

$\mathcal{S}$ makes a correct guess with the same probability that $\mathcal{A}$ is able to guess the random bit in the game defined in Definition 7. Therefore, Lemma 2 indicates that our group ranking framework is identity unlinkable. ■

Finally follows our main theorem

Theorem 1. *The framework presented in Fig. 1 is privacy-preserving.* □

B. Efficiency

We first analyze the computational overhead of our framework on each party. Computational overhead is measured by the number of group multiplications. Multiplication in the following analysis means group multiplication unless it is specified in contexts. We note that a group multiplication is equal to $O(1)$ integer multiplications when security parameter $\mathcal{K}$ is fixed. Addition will be omitted due to its smaller computational overhead compared with multiplication.

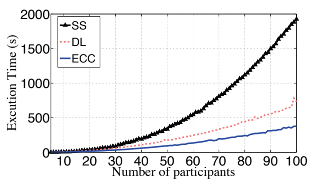
The computation of the gain computation part has $O(s^2)$ multiplications on a participant and $O(nm)$ multiplications on the initiator [2]. Since s is not necessary to be a big number [2] and independent of n, a participant's computational overhead is dominated by the gain comparison part. The first step of the comparison takes each party $O(\lambda)$ multiplications to generate a key pair and $O(\lambda + \lambda n)$ multiplications to prove her own secret key and verify others, where λ is the bit-length of the underlying multiplicative group size (e.g. $\lambda = 160$ in a 160-bit ECC group). Encryptions in step (6) take each party $O(l\lambda)$ multiplications. Step (7) costs $O(l^2 n)$ multiplications due to the computation of $E(\omega_i)$. Each party computes ln^2 exponents in step (8), which takes $O(ln^2\lambda)$ multiplications. Ranking submission part needs each party compute $O(ln\lambda)$ multiplications. In summary, a participant's total computation contains $O(l^2 n + ln^2\lambda)$ multiplications and the initiator computes $O(nm)$ multiplications.

A SS-based integer multiplication protocol [16] needs each participant to calculate $O(nt \log n)$ integer multiplications [6]. Here, t is the maximum number of colluded participants. A SS-based integer comparison protocol [5] requires $279l + 5$ invocations of the multiplication protocol to compare two l-bit integers, which is $O(ltn \log n)$ multiplications. Secure multi-party sorting protocol proposed by Jónsson *et al.* [3] requires $n(\log n)^2$ comparisons, i.e., $O(ltn^2(\log n)^3)$ multiplications. Actually, all comparison based SS sorting protocols require $O(ltn^2(\log n)^2)$ multiplications. If we want to resist $\lfloor n/2 \rfloor$ colluders, Jónsson's protocol and a comparison based SS sorting protocol would need $O(ln^3(\log n)^3)$ and $O(ln^3(\log n)^2)$ integer multiplications, respectively.

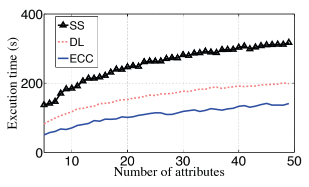
Our framework needs $O(n)$ communication rounds while Jónsson's protocol [3] needs $O((279l + 5)n(\log n)^2)$ communication rounds (at least one round for each invocation of multiplication protocol). The communication overhead of our protocol is dominated by the unlinkable gain comparison part, in which each participant's communication overhead is $O(lS_c + lS_c(n + 1)n) = O(lS_c n^2)$ bits. Here S_c is the bit length of a ciphertext. It is because each party sends l ciphertexts in step (6) and $ln(n + 1)$ ciphertexts in step (8).

VII. Performance Evaluation

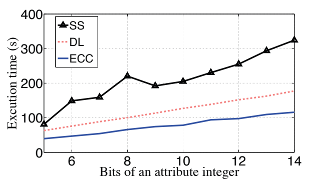
We simulated our framework and the protocol in [3] on an ordinary desktop computer, with an Intel Pentium 4 1.70GHz CPU and 512MB RAM. All protocols were built upon a popular free cryptographic library, Crypto++. Since Jósson's



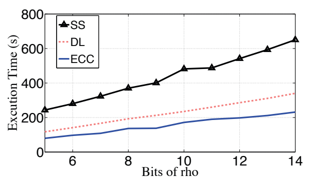
(a) Increasing n



(b) Increasing m



(c) Increasing d_1



(d) Increasing h

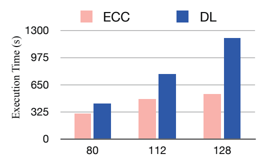
Fig. 2. Evaluating performance under different framework settings

protocol [3] does not deal with secure dot product problem, we used our gain computation part and fed the result β values to Jósson's protocol. In this way, we set up a group ranking framework based on Jósson's sorting protocol and refer to it as the SS framework in this section. We implemented our framework using two groups: the DL group and the ECC group (Section IV-B). The frameworks built on them are referred to as the DL framework and the ECC framework, respectively.

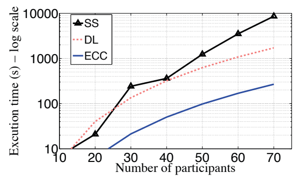
We first compared each participant's computation overhead of the three frameworks using different numbers of participants, different attribute vector dimensions, different bit-length of the attribute integer and different bit-length of parameter ρ, respectively. Initially, all the parameters were set as $n = 25, m = 10, d_1 = 15$, and $h = 15$. For security parameter k, we chose 1024 bits for the DL framework and 160 bits for the ECC framework. In each testing, we fixed all parameters except the one that was under inspection. The results are shown in Fig. 2. First, it is obvious that our frameworks performed better than the SS framework in most cases. The ECC framework gave the smallest execution time because it operates on a much smaller underlying finite field. In Fig. 2(a), when the number of participants grew, execution time of the SS framework increased approximately on the cubic order of n while our frameworks executed in the quadratic time on the order of n. This experimentally verified our previous analysis. In the tests of d_1 and h, the execution time increased in the linear order of n because increasing d_1 or h linearly increases β value's bit number l. And the execution time increased in the logarithm order of n when m increased.

We also tested each participant's computation overhead under different security level. According to the NIST guidance to FIPS140-2 [23], the security of 160-bit, 224-bit and 256-bit ECC group based cryptosystem are equivalent to that of 1024-bit, 2048-bit and 3072-bit DL group based cryptosystem, respectively. They respectively correspond to 80-bit, 112-bit and 128-bit symmetric security level. We then compared the execution time of a participant in the ECC framework with that in the DL framework under different security parameters. All settings are same as previous except that n is 70. The result is shown in 3(a). Compared with the DL framework, the ECC framework used less time while providing the equivalent security and grew slower when security level increased.

We used NS2 to simulate our frameworks and the SS framework on a random generated network, in which each link was 2Mbps duplex and has a latency of 50ms. The network



(a) Evaluation under different security level



(b) Communication latency

Fig. 3. Testing security parameters and communication latency

was generated by randomly deleting edges from the 80-node complete graph until there were 320 edges left. Each time, we only deleted the edge not disconnecting the graph. TCP was employed to deliver messages. ECC and DL frameworks used 160-bit and 1024-bit groups, respectively. We tested each participant's execution time in different frameworks with increment of n. The result is shown in Fig. 3(b). The ECC framework gave the best performance because it has smaller ciphertext size than the DL frameworkand less interaction rounds than the SS framework. It is shown in the figure that the SS framework ran faster than the DL framework at the beginning ($n < 30$) and fell behind after $n = 40$. The reason is that when the group goes large, network congestion becomes serious and the intensive interactions of SS framework dramatically escalate the communication latency.

VIII. Conclusion

In this paper, we have formalized the privacy preserving group ranking problem to meet the privacy demand in the current online environment. An efficient and fully distributed framework was proposed and proved to be secure in HBC model. The efficiency has been theoretically analyzed and experimentally compared with existing works.

References

[1] W. Qiu and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proc. IEEE INFOCOM'11*, Shanghai, China, Apr. 2011.
[2] I. Ioannidis, A. Grama, and M. Atallah, "A secure protocol for computing dot-products in clustered and distributed environments," in *Proc. IEEE ICPP'02*, Vancouver, British Columbia, Aug. 2002.
[3] K. Jónsson, G. Kreitz, and M. Uddin, "Secure multi-party sorting and applications," 2011, http://eprint.iacr.org/.
[4] M. Burkhart and X. Dimitropoulos, "Fast privacy-preserving top-k queries using secret sharing," in *Proc. IEEE ICCCN'10*, Zrich, Switzerland, Aug. 2010.
[5] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," *Public Key Cryptography–PKC 2007*, pp. 343–360, 2007.

[6] I. Damgård, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," *Theory of Cryptography*, pp. 285–304, 2006.

[7] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. ACM STOC'85*. ACM, 1985, pp. 291–304.

[8] I. Damgard, M. Geisler, and M. Kroigard, "Homomorphic encryption and secure comparison," *International Journal of Applied Cryptography*, vol. 1, no. 1, pp. 22–31, 2008.

[9] H. Lin and W. Tzeng, "An efficient solution to the millionaires problem based on homomorphic encryption," in *Proc. IACR ACNS*. New York, USA: Springer, 2005, pp. 456–466.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. IACR EUROCRYPT'99*, Prague, Czech Republic, May 1999.

[11] D. Stehlé and R. Steinfeld, "Faster fully homomorphic encryption," *Proc. IACR ASIACRYPT'10*, 2010.

[12] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" *http://www. codeproject. com/News/15443/Can-Homomorphic-Encryptionbe-Practical. aspx*.

[13] J. Brickell and V. Shmatikov, "Efficient anonymity-preserving data collection," in *Proc. ACM KDD'06*, Philadelphia, USA, Aug. 2006.

[14] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *Proc ACM CCS'10*, Illinois, USA, Oct. 2010.

[15] W. Du and M. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proc. ACM NSPW'01*, NY, USA, 2001.

[16] R. Gennaro, M. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *Proc. ACM PODC'98*, Puerto Vallarta, Mexico, Jun. 1998.

[17] I. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," *Proc IACR ASIACRYPT'04*, pp. 122–135, 2004.

[18] Z. Yang, S. Zhong, and R. Wright, "Anonymity-preserving data collection," in *Proc. ACM KDD'05*, Illinois, USA, Aug. 2005.

[19] O. Goldreich, *Foundations of cryptography: Basic applications*. Cambridge Univ Pr, 2004, vol. 2.

[20] C. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Crypto89*. Springer, 1990, pp. 239–252.

[21] J. Camenisch, *Group signature schemes and payment systems based on the discrete logarithm problem*. Citeseer, 1998.

[22] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Proc. IACR CRYPTO94*. California: Springer, august 1994, pp. 174–187.

[23] *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, NIST, CSE, Mar. 2011, available at http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf.