Privacy Preserving Group Ranking

Marc Ilunga Tshibumbu Mukendi

School of Computer and Communication Sciences

Semester Project

June 2016

Responsible
Prof. Serge Vaudenay
EPFL / LASEC

Supervisor
Handan Kilin
EPFL / LASEC

0.1 Motivation

Group ranking is a process used to find a the best candidate from a group. The process involves a party called the ***initiator*** and others parties called ***participants***. The initiator usually has a vector of preferences on some attributes and the participants have vectors of values, each corresponding to the attributes sought after by the initiator. At the end of the process, one or more candidates with attributes best matching the initiator preferences will be selected. This has many applications such as online marketing, personal interest matching, job recruitment and selecting candidates for medical experiment.

However, this process is more and more used in virtual environment in today's society and as a consequence privacy concerns emerge. A trivial approach to Group ranking would leak private informations about all the candidates even if some of the will not be picked at the end of the process. As an example online forms are used to carry such ranking. A naive implementation of the ranking would require that the candidates provide private information(i.e by answering questions on the form). Since some group ranking imply ranking candidates on based on sensitive information, such implementation of group ranking poses a problem to privacy.

Given the necessity of group ranking in a wide range of real world's application and the need to preserve candidates privacy, is there a way to perform group ranking while preserving privacy of all the parties? How do we prevent candidates to learn informations about other candidates? How do we prevent candidates to learn information about the initiator and thus cheating in the process? This is the subject of this paper.

In this report we present and explain a protocol

that performs privacy preserving group ranking and we give a Java implementation of the protocol.

0.2 The protocol

In this section we give a detailed explanation of the protocol.

0.2.1 Framework

The protocol is executed cooperatively by $n + 1$ parties. An initiator plus n Participant P0, P1... Pn. The protocol assumes that the participants are willing to accept the initiator invitation and to submit their private information if selected eventually. The questionnaire given by the initiator is represented as a m-dimensional vector. The initiator holds a m-dimensional vector v0 indicating the preferred values of each of the question of the questionnaire, another m-dimensional vector represents the weight associated to each questions. Furthermore, the questionnaire comprises: "Equal" questions meaning that the initiator is looking for a specific attribute. "Greater than" question means that the initiator is looking for values exceeding some threshold. We assume without loss of generality that the first t questions are"greater than" questions. Finally the answer of each candidates is also represented by a m-dimensional Vector. A complete description of the protocol framework is given below:

Explanation of the protocol In our implementation, the same group will be used for every iteration of the protocol. The implementation uses a prime-order group of at least 1024 bits, thus the computation of discrete log is made difficult.

Secure gain computation First we define what is the gain in the context of this protocol.
Given a criterion vector v0 = [v01, v02,..., v0m] and the weight vector w=[w1,w2,...,wm]. The partial gain value of Pj is
pj = $\sum_{k=t+1}^{m} wkv_k^j$ - $\sum_{k=1}^{t}(w_k(v_k^j)^2 - 2w_k v_k^j v_k^0)$ In terms of dot products, the partial gain is given by $wg \cdot vg_j - we \cdot (ve_j * ve_j) + 2(we * ve_0) \cdot ve_j$

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 $(Q_j X_j, \mathbf{c}'_j, \mathbf{g}_j)$ 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 $\bar{c}_t = c_t/(c'_t)^{x_j}$. Picks $r \leftarrow_R \mathbb{Z}_q$ and updates the ciphertext with $((\bar{c}_t)^r, (c'_t)^r)$.
   - 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.

Figure 1: Framework