Privacy Preserving Group Ranking-COMP 515 Project Report

Handan Kilinc

Koc University
Computer Science Department
hkilinc@ku.edu.tr

Abstract. We implement a privacy preserving group ranking protocol in semi-honest adversary mode. We use the ideas in [5] to preserve fairness. In our protocol, there are company and clients and they works distributed. In addition, we analyze its performance and we see that it depends on security parameter and number of clients in the system. In the end we suggest some ideas to develop the protocol both security and performance side.

1 Summary

We design a system that executes group ranking protocol. The group ranking protocol includes a company and clients. The duty of the parties in it is following:

- **Company:** (S)He has a questionnaire. His purpose is the finding best k parties that answer the questionnaire and be chosen according to some criteria of the company.
- **Clients:** They answer the questionnaire of the party and their purpose is to be in best k parties among all parties.

The main problem is preserving privacy against both company and clients. Privacy of the company to protect the criteria of the questionnaire because some dishonest parties can answer wrongly to the questions to be chosen if they know what kind of answers are expected by the company. On the other side, the privacy of parties is hiding all answers from both company and other clients.

Our privacy preserving group ranking (PPGR) protocol satisfy these privacy issues. We use the techniques in [5] purposed by Li et. al for the privacy issues. It consists two part:

1. Calculate gain of a party by using a secure two party computation where the idea is firstly suggested by Yao [9].
2. Clients compare their own gain with using a homomorphic encryption (see Appendix A9 and anonymization.

We can think gain as a score that parties have after they answer the questions. [5] use a specific secure two party computation protocol [3] that only computes vector dot product of two parties and only secure in semi-honest adversaries.

Our PPGR protocol also controls some failure that can be caused by some network failure or some malicious actions of some parties. The detection failure is important because the wrong control of the failure can break the privacy.

The PPGR protocol still needs to be developed because it is not secure with malicious adversaries. We assume that all parties are semi-honest which is not a realistic case. In addition, we see that the efficiency of our protocol depends on the security parameter that is length of the encryption key. We get this result because we test it with using 128,256, 512 and 1024 bits. For security issues 1024 bit is ideal.

2 Discussion

2.1 High Level Code Description

There is two main clients one of them (Client.class) represents the clients in PPGR and the other one (Server.class) represents the company.

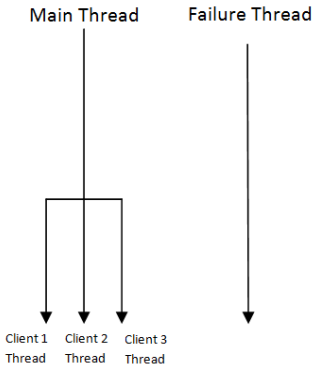

Fig. 1. Threads of Server.class with three connected clients

Server has mainly two threads which is "Main Thread" and "Failure Thread". See Figure 1.

Main Thread: It accepts clients and every connection of the new clients it creates new thread (Client Thread) that is responsible for this client.

⇒ **Client Thread:** It does all phases of the protocol. It has two part. The first one is "Client Acceptance" and the second one "Client Messages for Clients". "Client Acceptance" includes the company and client interaction which is accept the client, then send the question and finally calculate gain with using the protocol in [3]. "Client Messages for Clients" receives messages of the client (sender) for the other clients (receiver) and then deliver the messages to the receiver of the messages. Here, it helps clients to compare their gain and find the final result.

Failure Thread: It is to check failures. It checks all connected clients and if there is failure it aborts the protocol or removes the client from the system. It aborts if they are in "Client Messages for Clients" phase because after this point the protocol cannot continue without failed client. It removes the client from the system if they are in "Client Acceptance" phase since it can accept new client instead of failed client.

Client has only one thread. It firstly connects the server and if the server is in "Client Acceptance" method, he joins the protocol. Otherwise he finish the interaction. If he join, he receives the questions and answers them, and then finally calculate gain with the server (company). If the server is in "Client Messages for Clients" phase, then he receives contact information of the other clients and begins to send messages for clients to the server. Remark that sending these messages to server instead of receiver of the message directly does not break the privacy because all messages are encrypted. In addition, server does not changes these messages because he is company and to change them affect him worse since it can cause him to receive wrong chosen parties.

There are classes to help parties to create the homomorphic encryption and values of secure dot product computation.

2.2 GUI of the Protocol

The protocol begins after company is available. Figure 2 shows the GUI of the company. There are questions with their expected values. Besides, it shows the connected clients with their current situation.

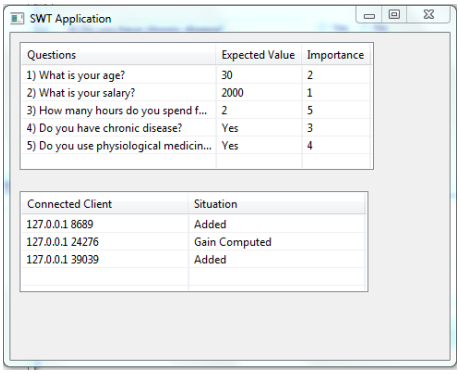

Fig. 2. Company Screen

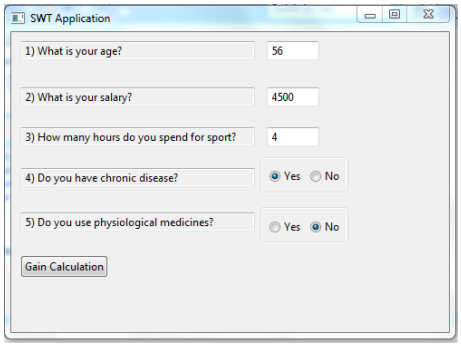

Fig. 3. The GUI of the client where he answers the questions

On the other hand the GUI of the clients are more changeable. After receiving questions from the company, the client has the screen in Figure 3 to answer questions

If the company push "Gain Calculation" button, the gain of him is calculated according to his answers without showing his answers and the criteria of the

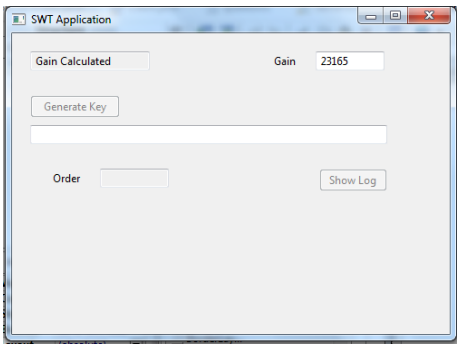

Fig. 4. Result of the gain

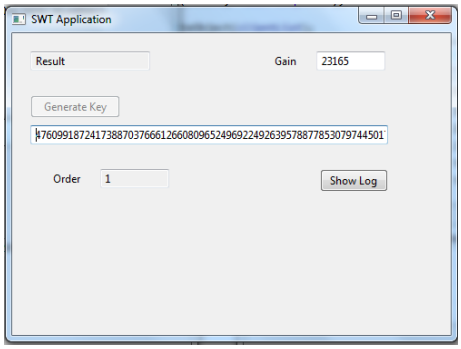

Fig. 5. After "Client Acceptance" mode ends in Company side

company. Then the following screen in Figure 4 is shown. "Generate Key" button is inactive until "Client Acceptance" ends in company side. After this mode, client can generate gain and start the comparison with other clients. Figure 5 shows the calculation result. The result shows the order of the client among others.

If client push button "Show Log", the log screen appears as in 6. As you see, the received and sent messages are not show any information about the client. They are hided by encryption and some randomization.

After receiving the rank result, clients sends message to company if they are the best k client. The company has the update in his screen as Figure 7.

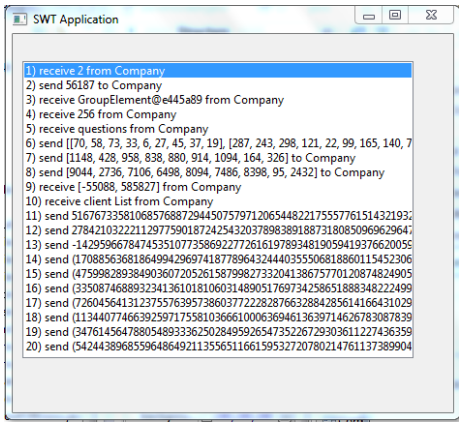

Fig. 6. Log of the clients

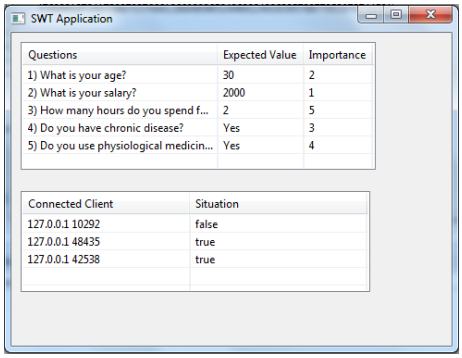

Fig. 7. Final result of the company. Situation "true" shows that the client is accepted

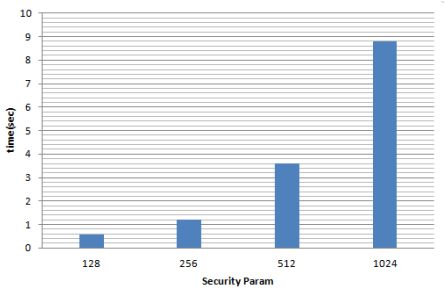

Fig. 8. Security parameter vs Performance in client side

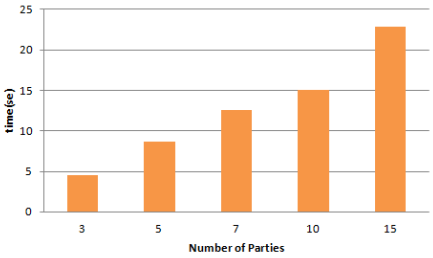

Fig. 9. Number of parties vs Performance in company side

2.3 Results

We have done some performance analysis according to number of clients and security parameter.

We choose different security parameter as in Figure 8 and calculate one of the clients performance in "Client Messages for Clients" phase. The results shows that the performance decreases quadratically while the security parameter increases. Security parameters are important for security and the ideal on is 1024 bit.

We run protocol with using security parameter 128 and different number of clients. We calculate amount of time that company needs to receive final result. The performance and number of clients are linearly dependent as in Figure 9.

2.4 Conclusion and Future Works

We implement the idea in [5] for our PPGR protocol. In addition we use similar randomization techniques in [2, 8, 1].

The main problem of this implementation is that it is secure in semi-honest adversary model. This model is not realistic because the adversaries are malicious in real world and so they do not follow protocol order always. To progress adversarial model in company and client side, there can be used some secure two party computation protocols against malicious adversaries [7, 4, 6] instead od dot product computation in [3]. However it makes the performance lower. So some trade of between security and performance should be decided and according to it the protocol can be designed.

The other drawback of the protocol is that number of messages that is sent. Since they are a lot, the performance is low. So I suggest to design a protocol where the parties depends only previous party. In this case, each part sends and receives message from his previous or next party. It can make performance really high because the current protocol needs some broadcasting.

References

1. J. Brickell and V. Shmatikov. Efficient anonymity-preserving data collection, 2006.
2. H. Corrigan-Gibbs and B. Ford. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 340–350, New York, NY, USA, 2010. ACM.
3. I. Ioannidis, A. Grama, and M. Atallah. A secure protocol for computing dot-products in clustered and distributed environments. In *Proceedings of the 2002 International Conference on Parallel Processing*, ICPP '02, pages 379–, Washington, DC, USA, 2002. IEEE Computer Society.
4. S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, 2007.
5. L. Li, X. Zhao, G. Xue, and G. Silva. Privacy preserving group ranking. In *Proceedings of the 2012 IEEE 32Nd International Conference on Distributed Computing Systems*, ICDCS '12, pages 214–223, Washington, DC, USA, 2012. IEEE Computer Society.
6. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, 2007.
7. J. B. Nielsen and C. Orlandi. Lego for two-party secure computation. In *TCC*, 2009.
8. Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 334–343, New York, NY, USA, 2005. ACM.
9. A. C. Yao. Protocols for secure computations. In *FOCS*, 1982.

A ElGamal Encryption

Key generation:

The key generator works as follows:

– Alice generates an efficient description of a cyclic group Z_q of order q with generator g from group Z_q.

- Alice chooses a random x from 1,...,q-1.
- Alice computes $h = g^x$.
- Alice publishes h , along with the description of Z_g, g, q, as her public key. Alice retains x as her private key which must be kept secret.

Encryption: The encryption algorithm works as follows: to encrypt a message m to Alice under her public key h,

- Bob chooses a random r from Z_q , then calculates $a = g^r$.
- Bob calculates the shared secret $s = h^r$.
- Bob calculates $b = m.s$.
- Bob sends the ciphertext a, b to Alice.

Decryption: The decryption algorithm works as follows: to decrypt a ciphertext (a, b) with her private key x, $m = b/a^x$