CS555 Extra Credit

December 3, 2022

CS 555: Cryptography Extra Credit Project

GitHub Repository URL: $https : //github.com/ishaanrc/555 - Project$
$VersionNumber : 295541f88d9f99f36a0d13fa913b0a54a760591a$

1 Languages and Libraries

Language: Python 3

Libraries:
1. math
2. decimal
3. numpy
4. random
5. socket
6. cryptography.hazmat
for algorand libraries refer requirements.txt

2 Specifications

1. There are three parties P_1, P_2, and P_3 each owning the El Gamal encryption of the messages m_1, m_2, and m_3. The parties also have the share of the private key (SK) for El Gamal encryption.

2. We have a Client C with a key pair (private key (SK), public key (PK)) The client wants to pay 10000 ALGO to know result of $(m_1 * m_2) + P_3$ mod p. We have built a protocol achieving what the client wants.

3. In this protocol, client initiates the challenge by contacting the algorand smart contract and sends the public key (PK). At the same time, client C contacts the parties to inform that the challenge has been initiated. The parties now will contact smart contract to get the public key. Once, they receive this,

the parties will send the El Gamal encrypted m1, m2, and m3 to the trusted party along with the shares of the private key owned by each party.

4. The trusted party then computes the private key and decrypts m1, m2, and m3 resp. Now, the trusted party will compute (m1*m2) + m3 mod p. After the completion of the computation, the value is sent to the client by the parties and then the 10000 ALGOS are paid to the parties by the smart contract.

5. The protocol ends when the client receives the encrypted result from the MPC.

3 Assumptions

1. There exists a trusted party that securely receives the three shares of the EL Gamal encrypted private key.

2. Every party P_i sends its input $x_i \in Z_m$ to the trusted party T where x_i is an El Gamaal share of the Private Key (i=1,2,3)

3. The trusted party, T, computes private key using the reconstruct protocol.

4. The trusted party, T, computes m_i by $Dec_k(m_i)$ securely without letting an active or a passive adversary learn any extra information

5. The protocol is secure till t < n corruptions.

4 Security Analysis

1. The private key is distributed into shares and distributed amongst 3 parties. This ensures that a party owns a share of the private key and not the entire private key. Thus, any party won't be able to decrypt the original messages.

2. The shares of parties and the encrypted values of the messages are sent to the trusted party securely. The trusted party reconstructs the private key and decrypts the messages.

3. The trusted party does the required computation and encrypts the result with clients public key, thus ensuring security while sending it to the client.

4. The clients learns the required value by decrypting it with it's own private

key.

5 Security Goals

1. The parties would not know the original values of the messages(m_1, m_2, m_3) since thye won't have access to any of the other parties share of the private key

2. The parties cannot learn the values of other parties.

3. The parties cannot compute the original private key.

4. The shares and the encrypted values of messages are sent securely to the trusted party.

5. The trusted party sends the required ask securely to the client.

6. No parties can learn the actual result ($m1 * m2 + m3$).