Secure voting protocol with multiple Central Tabulating Facilities (CTFs)

ISHANK ARORA	**14074009**
AYUSH KEDIA	**14075013**
DIVYAM GOYAL	**14075024**
AKASH GUPTA	**14075002**

SECURE ELECTIONS

- Computerized voting is generally never used for the general elections.
- As it is very hard to design a protocol that can both ensure individual privacy and prevent cheating.

What Makes A Protocol Secure For Elections ?

Ideal Voting Protocol Requirements

1. Only authorized voters can vote
2. No voter can vote more than once
3. No one can determine for whom anyone else voted
4. No one can duplicate anyone else's vote.
5. No one can change anyone else's vote.

Homomorphic Encryption

1. Homomorphic encryption is a form of encryption that allows computation on ciphertexts , generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.
2. The purpose of homomorphic encryption is to allow computation on encrypted data.

The Protocol

Participants

1. V voters and each client represents a voter.
2. C participating Candidates in the election.
3. N Central Tabulating Facilities (CTFs)
4. Central Server.

(1) Central Server:

- Assigns Public and Private keys to each CTFs.
- Receives request from voters situated at client side .
- Keeps track of current number of votes for every candidate.
- Receives request from a voters to access particular booth via a fixed port no and then server sends public key of that CTF to that voter.

- After the voting is finished , server notifies all its client the final result of voting using realtime sockets
- Decrypts the encrypted vote sent by the voter using the private key of that CTF and updates the vote count of the candidate

(2) Client Side(Voter)

- Voter sends requests to join a particular Central Tabulating Facility
- Public key of the requested Central Tabulating Facility is received from server.
- Each voter casts his vote.

- The vote casted is then encrypted using public key assigned by the server of the particular ctf requested.
- Voting is implemented using two level of threads, thereby promoting parallelisation.
- One vote per candidate is ensured by assigning unique ports

(1) Server generates a private and public key pair (K_{pu},K_{pr}) for each CTFs.

(2) Clients connect to the server stating its CTF choice.

(3) Server then, sends the Public Key of that particular CTF to the client.

(4) Client takes the vote (V) as input and encrypts it with Public Key.

$$E=K_{pu}(V)$$

(5) Encrypted vote of client is then sent to the server.

(6) Server then, decrypts the encrypted vote using the private key of the CTF and gets the original vote.

$$V=K_{pr}(E)$$

Advantages

(1) Flexibility in no of CTFs to avoid congestion

(2) No one can know if other person has voted or not

(3) No one can know to whom the other person has voted

(4) No one can cast more than one vote which is ensured by assigning unique ports to each client

(5) Working of each CTF is parallelly performed.

(5) Voters can vote parallelly.

(6) System can accommodate large number of voters as two levels of parallelization is done.

Limitations

(1) An additional requirement “Everyone knows who voted and who didn’t” is not currently supported.

(2) It is impossible to attain total concurrency in python due to the presence of GIL.

Implementation

The Protocol was implemented using python programming language.

Libraries Used:

- phe: For homomorphic encryption
- socket: For implementing client server architecture.
- pickle: For serializing objects
- threading: For implementing threads

Thank You!