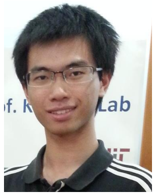
EPFL

E-voting Application Based on Homomorphic Encryption and Decentralized Tallying on Peerster

Fengyu Cai

Liangwei Chen



Ali El Abridi

- Decentralised Software Engineering Project 2019

Friday, January 30, 2020

Introduction

- The starting point of the project is a need to revive democracy through a technological revolution in the decision-making process.
- The objective is not to attack the instances that represent the mystical aspect of a State, of a Nation (Emperor, King, President), but rather the "biopolitics", concrete decisions affecting the everyday life of citizens.
- The whole purpose of the desired platform is to promote legitimate decision-making, to create a breakage with the currently deciding organs that have shown their limitations at all levels.
- the project’s objective is to transform a virtual opinion obtained via a smart device into a vote that counts and has all the legitimacy to make things happen.

Work Contribution

- System Design and Implementation:
  - Including frontend interface, tallier, independent server, database
  - Designed and Implemented by Fengyu ,Liangwei, and Ali
- Homomorphic Encryption
  - By Ali and Fengyu
- Trustee authentication and blockchain consensus
  - By Liangwei and Fengyu

I. Project Requirements

II. System Design and Implementation

III. Homomorphic Encryption

IV. Blockchain

V. Demo

VI. Conclusion

I. Project Requirements

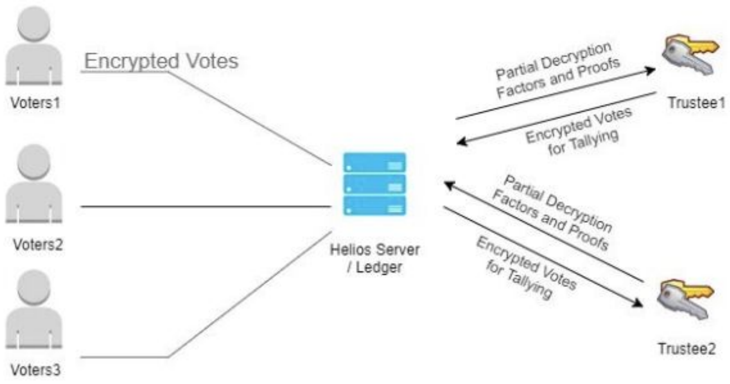
Project requirements

- To achieve the desired objective that we have drawn from the analysis and problem statement, the technological implementation of a voting system will need to have the following characteristics:
  - Open-Auditable
  - Anonymity
  - Reliability
  - Trustworthiness
  - Low-coercion

II. System Design and Implementation

System Overview

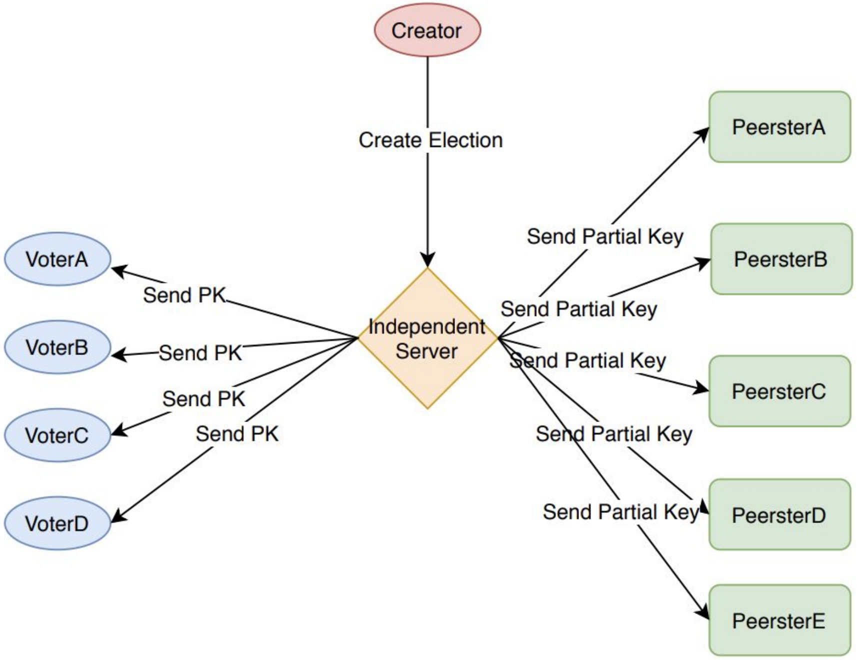
- Proposed Peerster solution:
  - Inspired by Helios, the first open-audit voting system that is publicly accessible
  - Uses end-to-end encryption
  - Distributes the secret key among a number of trustees
  - Uses blockchain to store the votes



Fengyu, Liangwei. Ali

Election Initialization process (Step1)

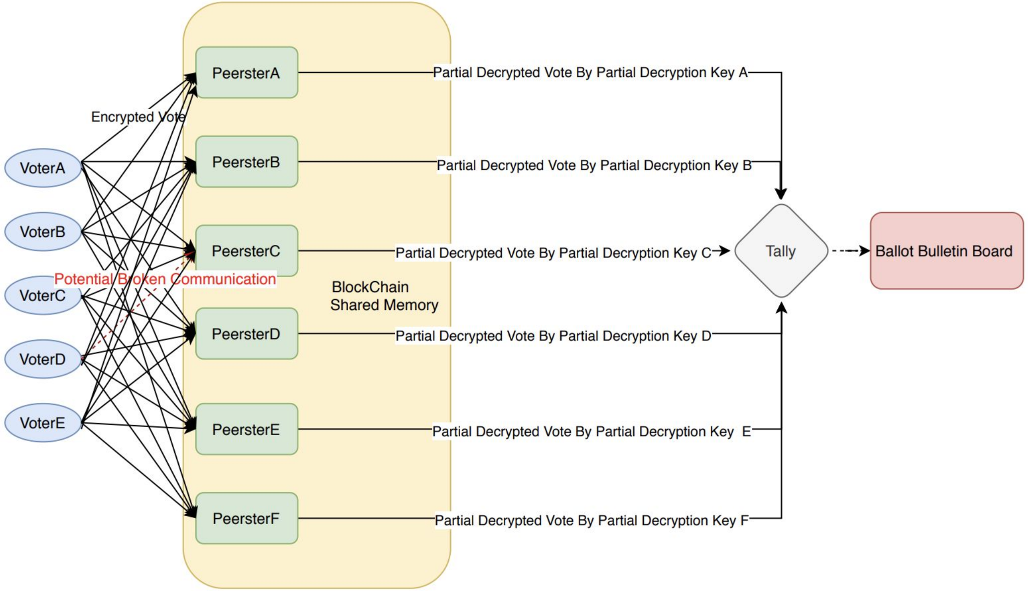
- **Creator:**
  - Creates the election and passes the information to independent server
  - Also updates with the backend and database
  - Decides to tally and end the election
  - Also can be one of the voters
- **Trustee (Peerster):**
  - Receives Authentication Secret and authenticates the peers
  - Partially decrypts the encrypted vote and passes it to the tallier
- **Independent Server:**
  - Generates Authentication Secret to the trustees (Peerster)
  - Once received new election creation, generates public key, splits the secret to partial private keys and sends to the trustees



Fengyu, Liangwei. Ali

Voting and result tallying process (Step 2&3)

- **Voter**:
  - Participates the election and encrypts the election with public key
  - After the end of election, can view the result of the election
- **Tallier:**
  - Collects the partial decrypted factors and homomorphically generates the voting result



Fengyu, Liangwei. Ali

Other Components

- 3 User Interfaces:
  - Voter / Peerster / Independent Server
  - Framework: Vue.js
  - State Management Pattern: Vuex
- User Management Backend:
  - Light-weighted Server: Flask
  - Database: TinyDB
- Blockchain + homomorphic encryption:
  - Peerster: Golang

Fengyu, Liangwei. Ali

III. Homomorphic Encryption

III. Homomorphic Encryption

- A simple and efficient solution for preserving the privacy of users’ votes (open-auditable, privacy, trustworthiness...)
- Peerster uses the additive property of El Gamal Encryption scheme such as
    - Such that the ciphertext of c_0*c_1 is the decryption of m_0+m_1

Setup:

1) Generate an ElGamal key-pair {generator, prime, secret key, public key}
2) Generate trustee partial decryption keys by choosing n-1 random private keys and compute the nth as secret key - (key_1 + key_2 + ... + key_{n-1}) mod p
3) Each voter encrypt its vote using the public election information such that encryption of a value m is performed as ci = (g^r, g^mi * pk^r) mod p.
4) Each trustee accumulate the votes and combines the ballots homomorphically by performing Tally = c0 * c1 * c2 … cN such that ci = (αi, βi)
5) Each trustee then computes a partial decryption factor df = α^key_t mod p such that α is the homomorphic tally of a certain (question, answer) tuple.
6) For each question and each answer, the tallier reassemble the tally by aggregating the decryption factors of the trustees to produce (α, β) and search for its value v by iterating over all potential values such that α = df_0* df_1*…*df_k)mod p and β = modInverse(alpha, p)*β(tally) and such that v = g^{0,1,2…., # of voters} mod p = β

IV. Blockchain

- Authentication
- Consensus

Blockchain

1. Structure
   a. The nodes in the blockchain network are trustees (Peersters).
   b. The network is connected
2. Goals
   a. Build a universal, accurate and non-modifiable ledger of encrypted votes at the end of election
   b. Reduce the trust on single trustee
3. Stages
   a. Authentication
   b. Consensus

Blockchain: Authentication

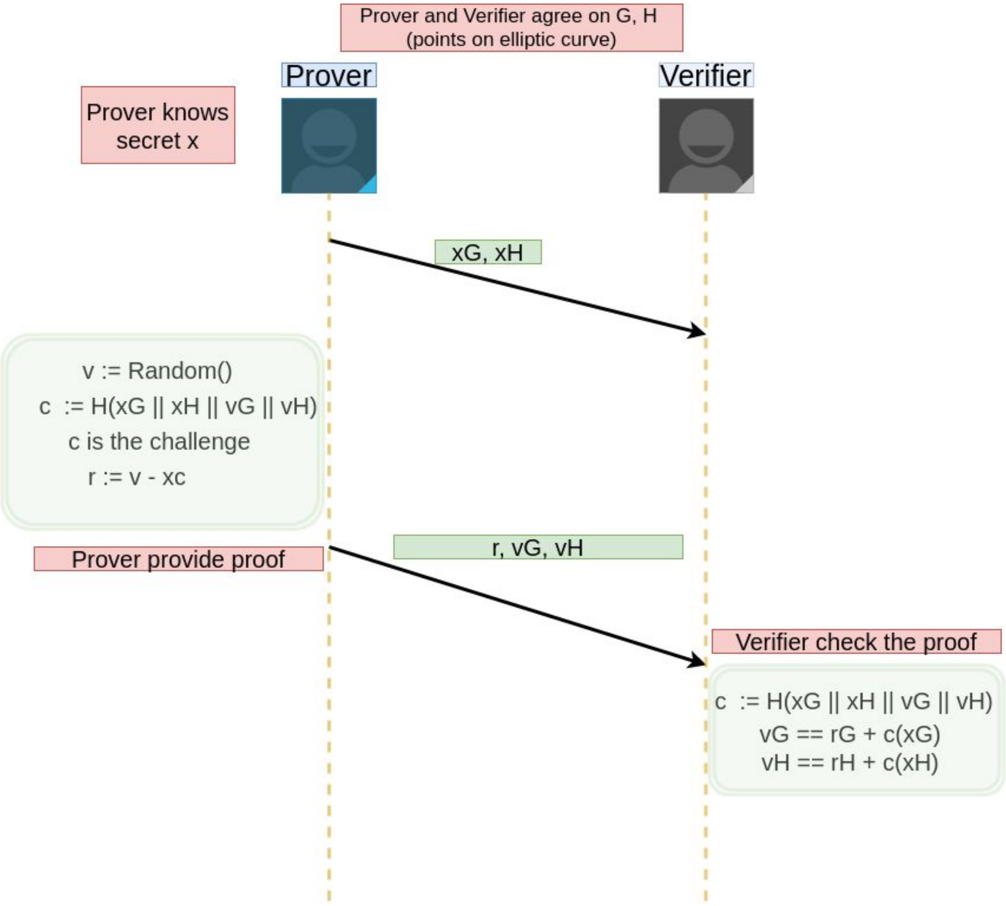
- **Problem:**

  Trustees has no global information of trustees instead of the total number. This fact allows untrusted peerster to propose FAKE votes in the network.

- **Solution:**

  Trustees obtain secret from independent server. Non-interactive zero-knowledge proof is then used to authenticate trustees during communication.

Blockchain: Authentication



- The trustees authenticate themselves using NIZKF as shown in the left hand side.
- Untrusted peers, on the other hand, will not be able to insert into blockchain since they have no knowledge of the secret x.

Blockchain: Consensus

Goals: Build a ledger of of encrypted votes at the ends of each election with properties:

1. Accurate: Any conflicting vote insertion into the blockchain should be detected as long as one peerster is honest

2. Universal: All trustee has same blockchain in the end

3. Non-modifiable: Any modification to the blockchain should be detected as long as one peerster is honest

4. Robustness: Any correct vote is added into the blockchain as long as one trustee receives it

Blockchain: Consensus

Do we need the complicated algorithms?

- Proof of work? Not needed since the trustees collaborate instead of competing to build the ledger

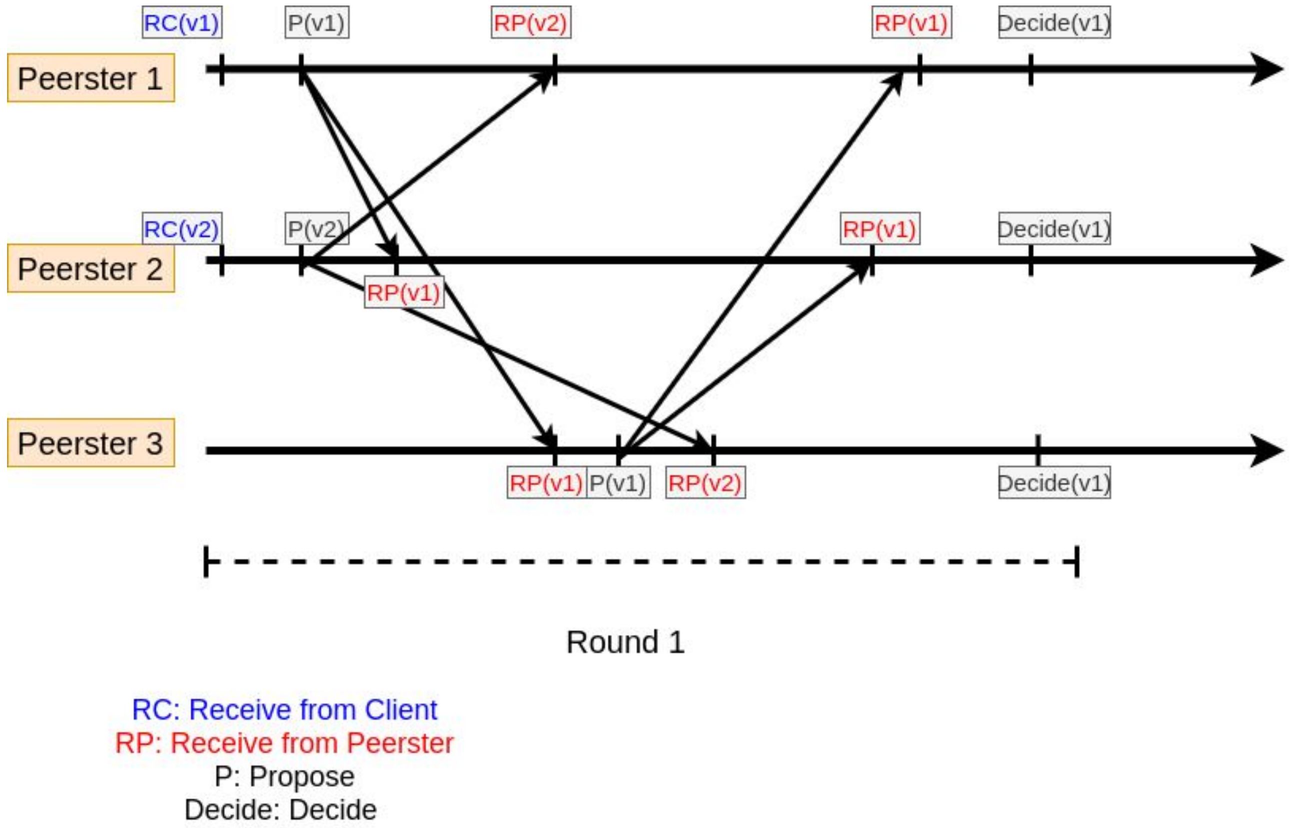
- Proof of state? To some extent……. But no priority need to be imposed on the trustees.

- TLC? No need of 3 tlc round per consensus round.

- Failure handling? Not needed since trustees are assumed not to crash.

These findings motivate us to implement a simple and intuitive round based consensus.

Blockchain: consensus



Technical details:

1. Proposal: proposal can be generated either from client or other peers with random fitness value.
2. Round termination: Receive proposals from all the peers.
3. Decision: Decision is made by selecting highest fitness value. Conflict is resolved by taking the trustee with smallest id.

V. Demo

VI. Conclusion

- Succeeded in implementing an E-voting Peerster based on homomorphic encryption and blockchain that guarantees:
  - Anonymity & trustworthiness (Homomorphic encryption)
  - Integrity (Blockchain)
- Auditing implementation (potential improvement)

References

- David J. Wu. 2015. Fully homomorphic encryption: Cryptography’s holy grail. XRDS: Crossroads, The ACM Magazine for Students 21, 3 (2015), 24--29.
- ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in Cryptology, Proceedings of CRYPTO '84. G. Blakley and D. Chaum (Eds.). Springer, Berlin Heidelberg, 1985, 10--18.
- M. Blum, P. Feldman, S. Micali, "Non-interactive zero-knowledge and its applications", *Proc. 20th Annu. ACM Symp. Theory Comput. (STOC’88)*, pp. 103-112, May 1988.