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

Ali El Abridi, Fengyu Cai, Liangwei Chen
ali.elabridi@epfl.ch, fengyu.cai@epfl.ch, liangwei.chen@epfl.ch

Decentralized systems engineering (CS-438) - EPFL
Submitted on January 31, 2020

Abstract

This project aims to create an open-auditable e-voting application on top of the current implementation of Peerster based on decentralized tallying and homomorphic encryption by splitting the secret key among multiple trustees. This allows the users to create and participate in elections while ensuring that their votes are *private* i.e., no one knows what they have voted, *verifiable* i..e, their votes have been accounted for, and *trusted* i.e, the final tally has not been tampered with, given that the majority of the trustees are honest.

Table of Content

1. Introduction

All current political systems such as monarchies, republics, and even military dictatorships call to vote, one way or another, on a four-years to seven-years basis. Some means of "getting around" the will of the people were put in place by lobbies or by "representatives" in order to confiscate decision-making, and this in all types of communities e.g. association, trade-union, political party, commune, etc. On the other hand, The current political system contradicts with our way of life and the arrival of new information technologies such as social networks and our ability to interact daily and simultaneously on several issues using the internet. 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, or of a nation (Emperor, King, President), but rather the "biopolitics", the concrete decisions affecting the everyday life of citizens. There will always be a need for an ultimate representative of the state, where legitimacy, history, law and religion are essential to build a nation. The number of highways to be built, the social security of citizens, and even education are concrete subjects on which every citizen can position themselves without the need of a representative.
These new technological means provide the opportunity for everyone to participate in any decision or debate in a local, national, or international way, and this in a virtually immediate manner without the need to go to the polling station to vote, and all the logistics involved in organizing a traditional election process. The aim of the E-voting peerster application based on blockchain, homomorphic encryption and decentralized tallying is to provide this technological means to people in order to voice their choices and opinions without fear of being persecuting or coerced in their decision-making. In order to guarantee the latter, we need to maintain a certain set of election properties for the voters related to privacy and trustworthiness as described in the section below.

1.1 Election Properties and 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:** It is important to give the possibility for external observers, and watchdogs to verify the results of a certain elections even after the tallies have been computed. The purpose of it is to eliminate the possibility of a rigged counting system that will miscount the votes in favor of a certain answer or a certain candidate.
- **Anonymity**: Voter secrecy is a must. It is important to protect both the voters and the content of their votes. A voter could be influenced by lobbyists or external influencers that will force their votes in one way or another. However, keeping both the integrity of the vote and the privacy of the voters is nearly impossible. Utopia aims to make a compromise between the two, and keep the integrity of the vote, and therefore, guarantee the trustworthiness of the results, but do not guarantee the privacy of the voters in order to universally verify that all the casted votes have been counted in the final tally.
- **Reliability**: the application should be reliable and available by all possible means, desktop version, mobile version and so on to guarantee that all kind of users will be able to vote without interruption or discrimination to a certain of user depending on the device they use or their geo-localisation.

- **Trustworthiness**: the application needs to achieve multiple levels of encryptions that cannot be faked nor unveiled by a single entity. Multiple trustees will need to hold a part of the secret keys in order to lower the probability of dishonest subset of them that will rig the voting system, and unveil the content of single ballots.
- **Low-coercion**: coercion is an important factor to take into account in every election. Voters should express their opinion in total secrecy, and if given the chance to, they should be able to reexpress their votes in case that the first one was done under pressure by external people.

2. System Overview

2.1. Decentralized E-voting using Homomorphic encryption

Homomorphic encryption[1] is one of the most widely used cryptographic protocols in secure secret-ballot voting schemes[2] among others such as mix-nets[3] and blindsignatures[4]. Homomorphic encryption is used in the tallying process of the e-voting scheme. Each voter encrypts his/her vote (binary vote) with some associated public information to trace it back to its source. The votes are accumulated in a public bulletin board in which every voter can check that his vote has been accounted for, and that no voter has voted twice *(verifiable)*. When the voting process finishes, homomorphism is applied to the encrypted sum of the votes, which is then decrypted using a decryption key. This decryption key is split among multiple trustees, who collaborate in the election process in a decentralized manner. This property guarantees that the individual votes will not be revealed *(privacy)* or the final results will not be tempered with, as long as there is at least one honest trustee *(trust)*. Helios[4] is a voting scheme that relies mainly on homomorphic encryption similar to the underlying structure explained, alongside with ElGamal, and Chaum-Pedersen Zero-Knowledge proofs cryptographic algorithms (figure 1) . However, Helios E-voting scheme has one major flaws that might be used to temper with the voting system. The votes are sent and aggregated by a centralized Helios Server that serves as a public bulletin in which the different voters send their encrypted votes. The new Peerster solution is based on decentralizing the public bulletin by making all the trustees participating in aggregating the votes using Blockchain as described in section 2.2 while providing a decentralized tallying of the votes using their partial decryption keys similar to Helios.

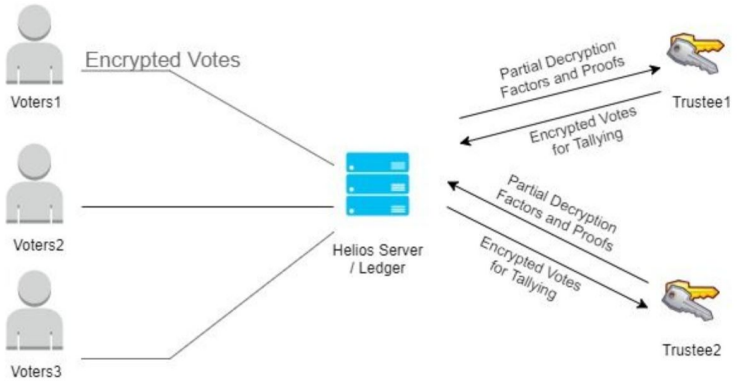

Figure 1. Helios E-voting homomorphic scheme

2.1.1. Homomorphic background and setup

As explained earlier Homomorphic encryption is 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 that the ciphertext of c0*c1 is the decryption of m0+m1. The process of homomorphic encryption between the different trustees (peersters) and the voters, election creators, independant server and tallier is defined as follows:

1. Generate an ElGamal key-pair {generator, prime, secret key, public key}when election creator initialize an election, and the key-pair is held by the independent server.
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 and distribute the different partial decryption keys to the different trustees (peersters) via private message.
3. Each voter encrypt its vote using the public election information available from the independent server about the election such that encryption of a value m is performed as ci = (g^r, g^mi * pk^r) mod p and sends the ciphertext vote to all the trustees via private message.
4. Each trustee accumulate the votes that have been stored in the blockchain and combines the ballots homomorphically by performing Tally = c0 * c1 * c2 … cN such that ci = (αi, βi) when the election ends.
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, and send it to the tallier.
6. For each question and each answer, the independent tallier reassemble the tally by aggregating the decryption factors of the trustees to produce (α, β) and search for its election (question,answer) pair result 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 and publish the results.

2.1.2 Homomorphic Election Initialization (step1)

The election initialization process involves three parties at the beginning before voters are able to vote, which roles are defined below:

- **Election creator:**
  - Creates the election and passes the information such as election description, set of questions and answers to independent server which updates the backend and database of it.
  - Decides on the end the election, and when to tally the votes
  - Could also be one of the voters in his own election or other elections.
- **Independent Server:**
  - Is in charge of maintaining the information about the different election, and provide the public encryption key to the voters.
  - Is in charge of splitting the secret key and sends it to the different trustees at the beginning of its election.
- **Trustee (Peerster):**
  - Receives partial encryption key from the independent server at the beginning of the election.
  - Receives shared secret from the independent server for authentication using non-interactive zero-knowledge proof.

- Is in charge of inserting the encrypted vote into the blockchain for record keeping and validating.
- Is in charge Partially decrypts the encrypted vote and passes the decryption factor to the tallier when the end election signal is received from the creator of the election.

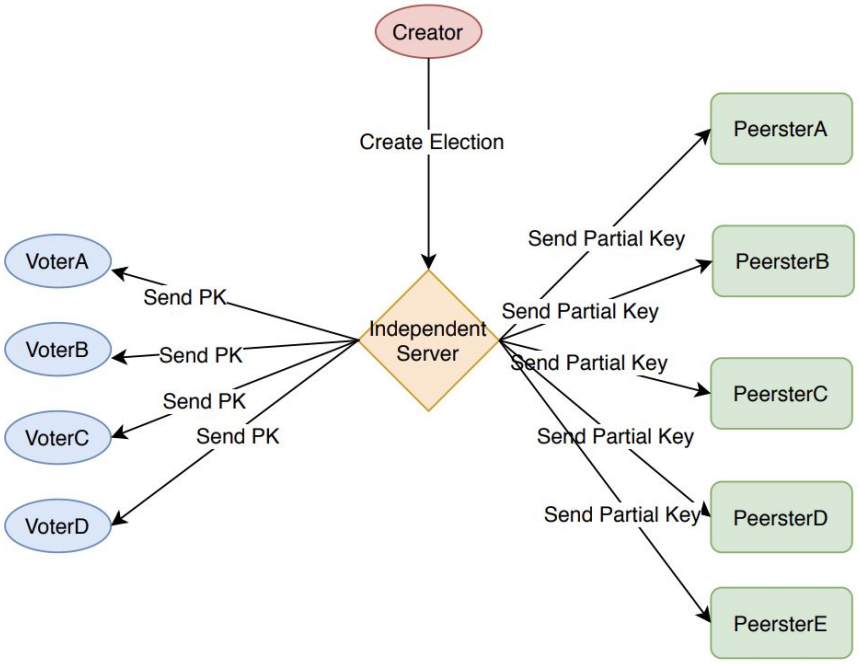

Figure 2. Election initialization process

2.1.2 Homomorphic Voting & Results Tallying (step 2&3)

The voting and results tallying is defining part where voters are able to voice their choices in an election and cast their ballots. It takes part after the homomorphic initialization, and it involves two additionals parties in addition to the three others already presented, i.e, voter and tallier:

- **Voter:**
  - Participates in the election and encrypts his casted ballot (choices) using the public key.
  - After the end of the election, he can view the publicly available election results.
- **Tallier:**
  - Collect the decryption factors from all the trustees at the end of the election.
  - Generate the results of the election by aggregating the different decryptions factors and publish them.

2.2 Blockchain in E-voting

Blockchain, a mechanism providing superior security level than any other database system, has been considered as an ideal tool to develop e-voting system from the day it was invented [5]. The features which make blockchain ideally suitable for election can be summarized into three points: robustness, immutability and consensus. The robustness feature originates from the distributed nature of

blockchain. Since every node has a local copy of the ledgers, a single point failure will not cause the loss of the data. Meanwhile, immutability is achieved through the reference from new ledger to the previous ledger. In specific, each new ledger keep a hash of previous ledger so that a chain of blocks can thus be built. Any attempt to tamper previous entry will then be detected by checking the hashes. Lastly, the ambiguity of history is resolved by requiring nodes to reach consensus before adding a new block to the ledgers. With the aforementioned three common properties, the blockchain can be further divided into three categories: public, private and consortium [5]. While public blockchain is widely used for digital currency such as bitcoin and dogecoin, it has been considered to be unsuitable for e-voting since costly proof-of-work consensus algorithms are required to certify and conduct vote [5]. Private blockchain, on the other hand, grants not only write but also read ability to participants who can authenticate themselves. This property greatly reduce the calculation and maintenance cost of the ledgers and thus have been utilized in several e-voting systems such as BroncoVote [6]. Even though private blockchain has achieved great success in authenticated voting, it may not be extended to scenario not requiring authentication. Moreover, by disabling the public to read the blockchain and thus verify the results, the system may not convince the public of its security. To provide reliable and private election under this scenario, we propose Peerster E-voting system which deploy a consortium blockchain among the trustees. Consortium blockchain, in comparison with its public and private variants, grants the write privilege to a fixed set of trustees while allow everybody to read the blockchain. By instructing voters to communicate with the trustees, the problem of limited range of eligible voters is able to be resolved. Furthermore, allowing the public to read the voting blocks and verify the reliability will attract more users to vote in our system.

2.2.1 Structure of the blockchain

The blockchain contains a connected network with Peersters as nodes. Compared with the original Peerster, the new version has been added with non-interactive zero-knowledge proof for authentication and synchronized round based consensus for the blockchain. The blockchain is designed to have the following functional components:

- **Authentication**:

  Imagine a situation where some hackers manage to run a Peerster instance and attempt to promote a forged vote to the network. Since a trustee may have no knowledge about its peers apart from the total number of peers and its neighbours, the untrusted Peerster may succeed to insert the forged vote into the blockchain. This weakness, which will damage the integrity, motivates us devise an authentication mechanism of the trustees.

  The blockchain system implements the authentication by taking advantage of the non-interactive zero-knowledge proof[8]. From a high level point of view, a shared secret obtained from the independent server will be used to authenticate the trustees. A non-trusted peerster will not be able to obtain the secret as long as the trustee's database is not hacked.

- **Consensus:**

  The consensus component achieve the goal of building a universal, non-modifiable, accurate and robust blockchain containing the encrypted vote. In short, the trustees establish a candidate block at the beginning of each round. They then gossip using the underlying communication layer of Peerster. The trustee validate the proposal by checking the correctness of hash. After receiving all the proposal in a round, every trustee use the same decision algorithm to select the block to append into the blockchain.

2.2.2 Properties of the blockchain

The properties of the blockchain includes:

- **Automated transmission failure recovery**: Thanks to the synchronization of records provided by the blockchain, our system is able to recover from all-but-one failures of transmission.
- **Immutability of records after insertion into the blockchain**: A peerster hacked by attackers may attempt to modify a record or even delete it from the shared database. However, this kind of attack can never works in our system due to the immutability of the blockchain. By modifying or deleting any record in any ledger, any auditors can recompute the hashes from the beginning block to detect this modification.
- **Voter verifiability**: Each voter can access the blockchain after the election. He/She can check that his/her vote has been correctly counted by checking whether the encrypted vote exist in the blockchain.
- **Public auditability**: The blockchain is readable by the public in the sense that each auditor can request for the ledgers and a synchronized blockchain of encrypted votes will be sent to the users after voting ends. After that an auditor may verify that the tallyer has honestly and correctly count by referring to the encrypted votes stored in the blockchain.
- **Automatic synchronization among peerster**: The network connecting the peersters is by nature asynchronous. For instance, some link may be congested or simply broken so that when other peersters have recorded N encrypted votes, a peerster may have received nothing. Our implementation of blockchain will resolve this problem by implementing a mechanism resembling causal reliable broadcast. Specifically, every peerster periodically emits the STATUS packet which serves as a vector clock. Any peerster that finds himself lagging behind will request for the message it lacks. It will then put the missing encrypted records into its local copy of blockchain to synchronize with other peerster. By assuming that the networks containing only perfect links and no peerster will crash, our blockchain promises that every peerster will have the same copy of blockchain in finite amount of time.

2.3 E-voting Auditing

Auditing is an important aspect for the participants to verify that their vote has been correctly protected and counted in the final voting result [3]. According to the Helios system [4], there are two verifications provided to the users. Firstly, it provides the option to simple audition for an individual and simple vote. Before the submission of his or her final decision, individual voting audition could allow users to download the data containing his or her voting plaintext, nonce randomness, and the corresponding cybertext, or ask the user interface to render this on the screen. Later, the user can run locally to verify both the correctness of his voting decision and the correctness of encryption. Also, it is also possible to take advantage of Ballot Encryption Verification (BEV) programs offered by Helios.[4]. What is more, in the Helios system, the user can also verify the election though the Election Tally Verification program. This program downloads the election parameters, the bulletin board of voting, and decrypted votes, etc. The verification program check all proofs and replay the

process of tally based on the decryptions. Finally, it will output the list of anonymous voters and their corresponding vote hash with the verified tally. With the increasing number of auditors involved, the voters could be more confident that the progress of tallying has been performed innocently and the voting result is pure and trustworthy. Unfortunately, the e-voting part is not essential to our Peerster implementation and is only optional. However, the individual vote integrity is now provided by the blockchain instead, and only the final count verification that could be done by any external independent entity has been neglected in our implementation due to time constraints.

3 System functionality & Architecture

3.1. System overall functionality

- The purpose of the Peerster E-voting system is to allow the users (voters) to vote in an election in absolute anonymity, without having to worry about the trustworthiness of the tallying process, and the manipulation of the election results by any of the organisms involved in the election process. The Election UI will allow any entity to create a Peerster election with the set of potential questions and answers proposed to the voters. The election UI will also allow the addition of external trustees (peersters) to the election process, which will hold a partial decryption each that is used to decrypt the final tally in a decentralized manner when the election ends. The Voting UI, will allow voters to cast their ballot for a particular election by encrypting their votes using the public key of the election, and sharing their cast ballot with the different trustees as illustrated in figure 2.
- The decentralized storage of the votes among the election's trustees using blockchain will allow not only the voters to verify that their votes are accounted for, but also the public to check the integrity of the voting process in the later Election Tally Verification. These functions prevent any malicious trustee (or minority of malicious trustees) from hiding their particular votes to the public bulletin. Furthermore, the consensus reached through blockchain will improve the robustness of the system by tolerating transmission errors of the encrypted votes.
- The E-voting Auditing will allow any external watchdog or trusted organism to certify that the election results are genuine, and all the votes have been accounted for.

And in this part, we would like to introduce our system from the perspective of the user.

3.2 Voter and election creator interface

3.2.1 Registration and Login

The user could register their information, and send to the backend server and database. After successful registration, the user can login as shown in the figures.

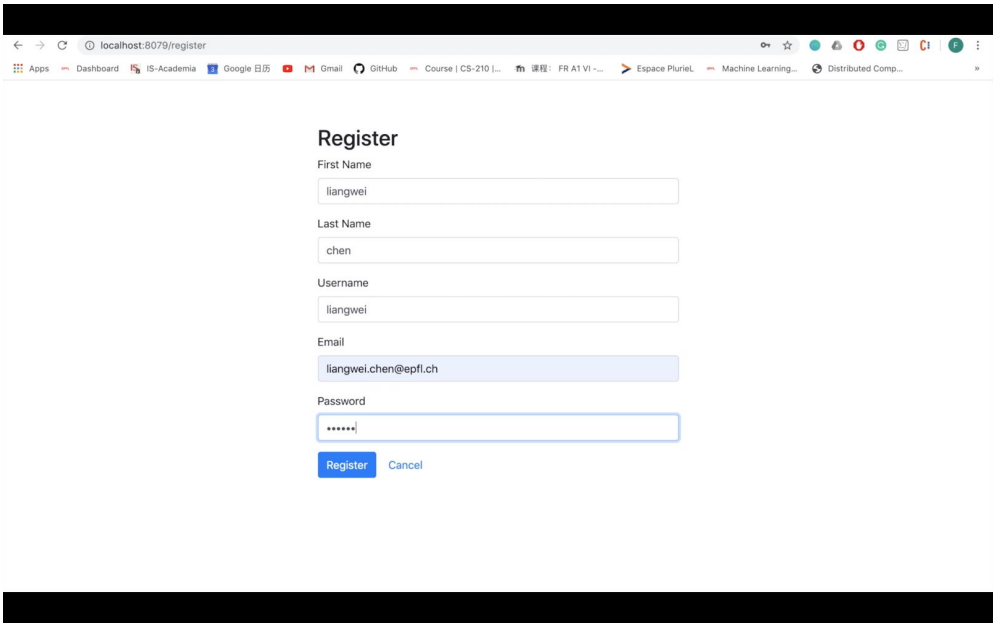

Figure 3. Voter/election creator registration user interface

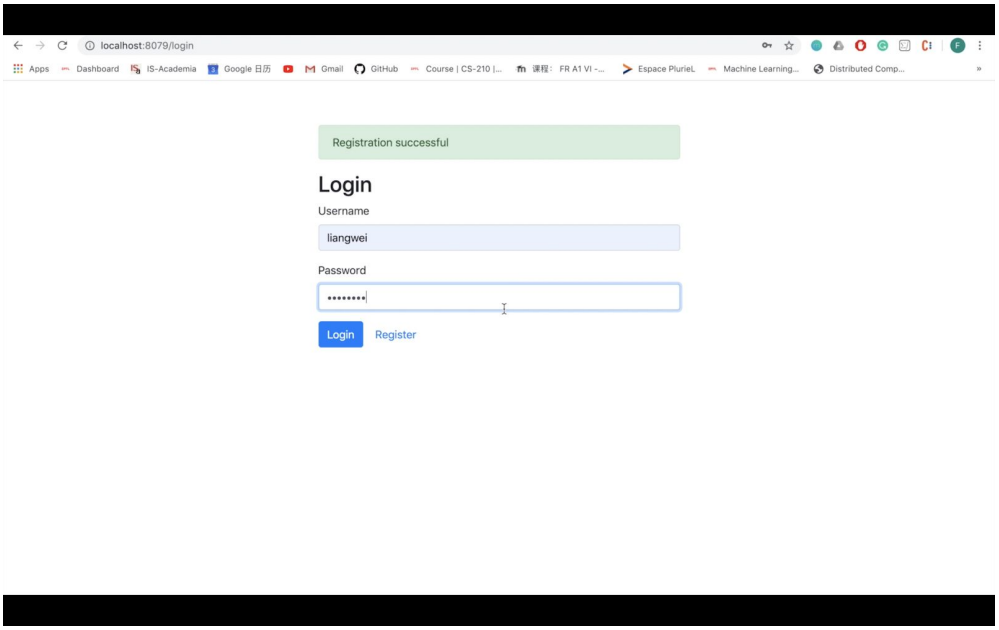

Figure 4. Voter/election login user interface

3.2.2 Voting and Election Creation main interface:

After the step of logging in, the user is authorized to create and election, participate in an election, and view the election result which he has voted. Only the creator of the election has the right to end it. In the figure, the user ali does not have the right to end the election ‘Election1’ but he is able to tally the election ‘Election2’

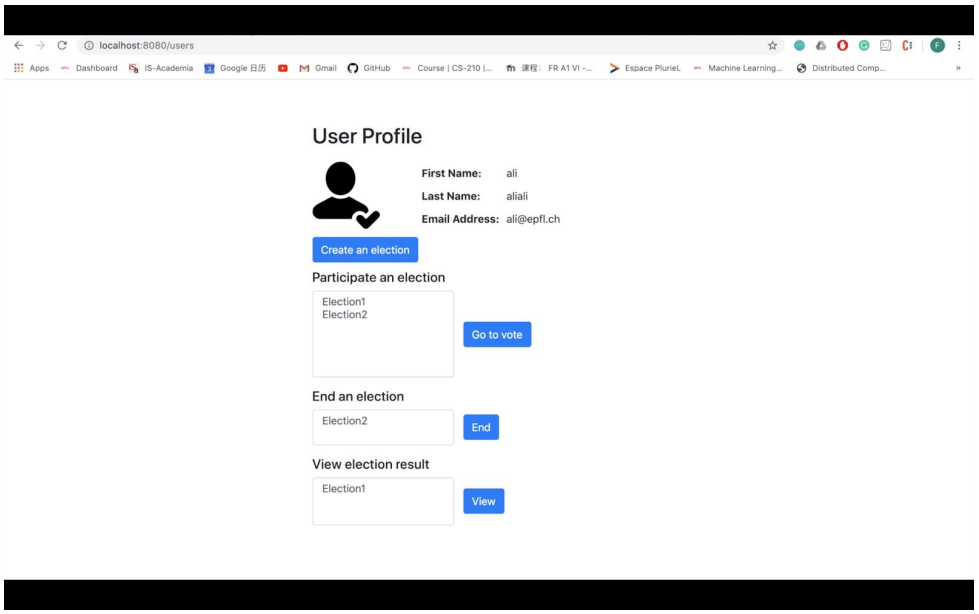

Figure 5. Voter/election creator main user interface for election creating, end and voting

3.2.3 Create an Election

After switching to the election page, the creator could give the election name and description. Also, the number of questions and the number of choices in each election are also customizable.

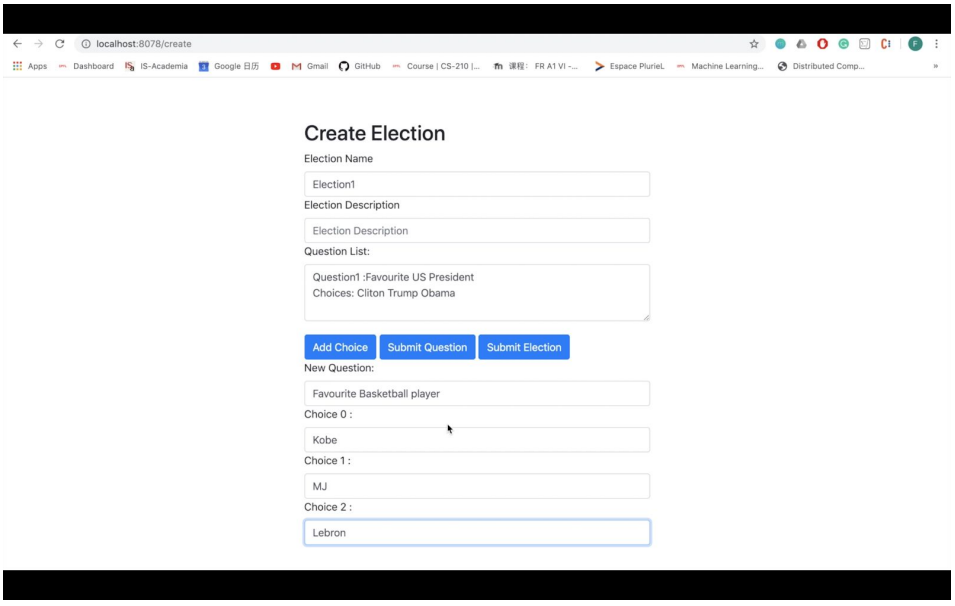

Figure 6. Election creation user interface

3.2.4 Participate an election

After an election has been generated, the election name will display on the profile page of every voter. The voter is able to select this and click the button to join. And after making choices, he could either submit the vote or return to the user profile page.

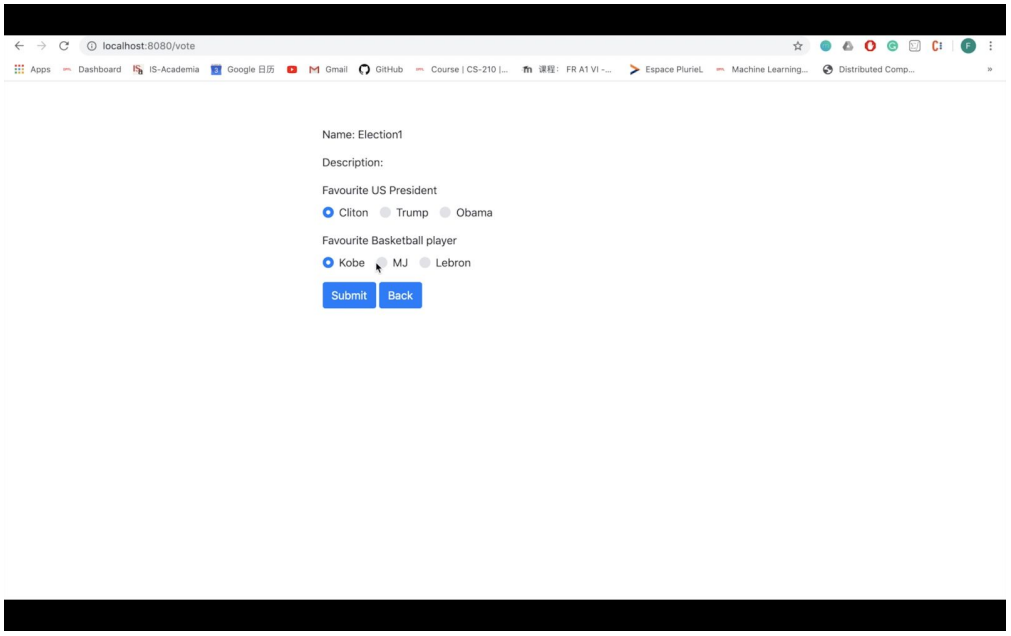

Figure 7. Election voting interface

3.2.5 End the election and view the result

The creator can decide when to stop the election and ask the tallier to collect the vote from trustees. The voter is able to select the election and view its result afterwards.

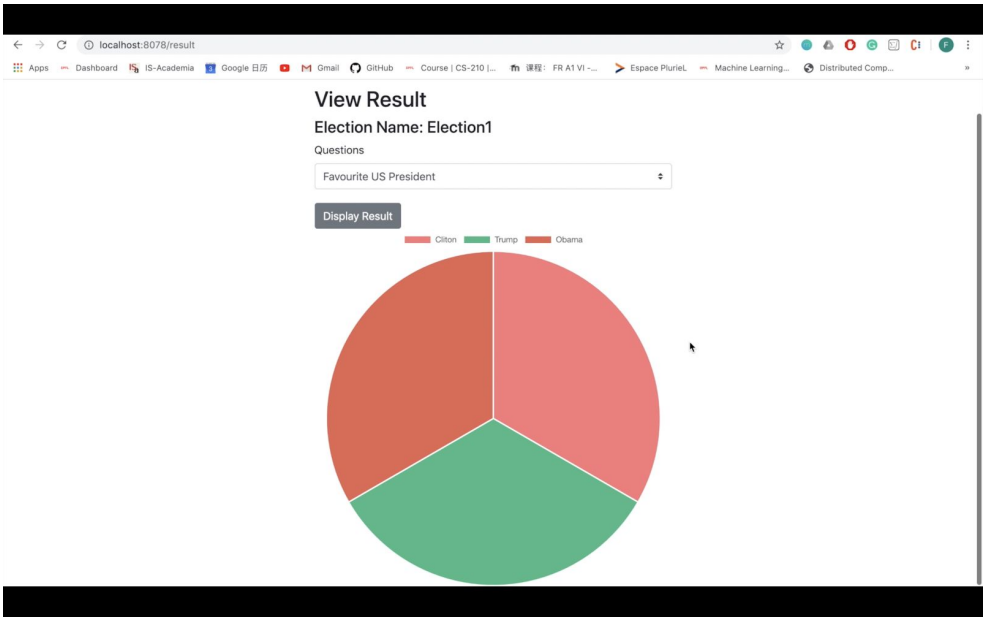

Figure 8. Election result viewing

3.3 Independent Server

As mentioned above, independent server has two functionalities

1. Firstly, it will generate and send the authentication secret to the trustees. Trustees will take it for further peer verification.
2. Secondly, when receiving an newly-created election, it will not only generate the corresponding public key to the voters but also split the private secret to partial private keys, and pass them to the trustees as the following figure.

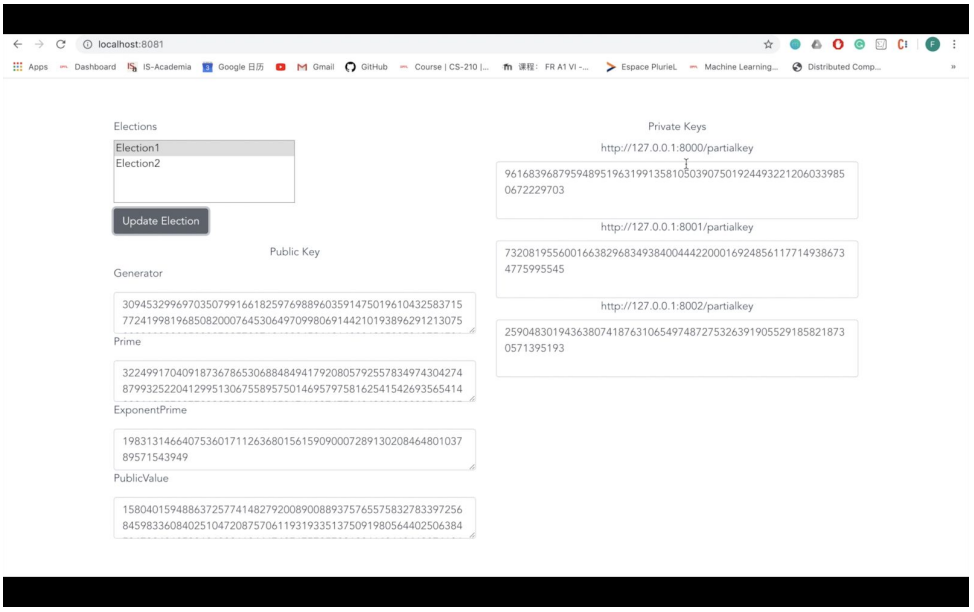

Figure 9: independant server private, partials, and public keys view

3.4 Backend Server and Database

3.4.1 Backend Server

Here, we run backend server to manage the user authentication, and voting information. Also, it will contain the elections and voting. For the voting part, it will only know which voters participate which elections, but hide the voting content. The server is light-weighted and implemented in the framework of Flask.

3.4.2 Database

We host a database to contain the information of voter, election, and voting information. It will run and save on the local disk as three json files.

3.5 Decentralized storage using Blockchain

3.5.1 Authentication

The independent server sends a randomized token to all the trustees for authentication during initialization of the blockchain. This token then serves as a shared secret among the trustees. To conduct authentication, non-interactive zero-knowledge proof has been utilized to verify the holding of the secret.

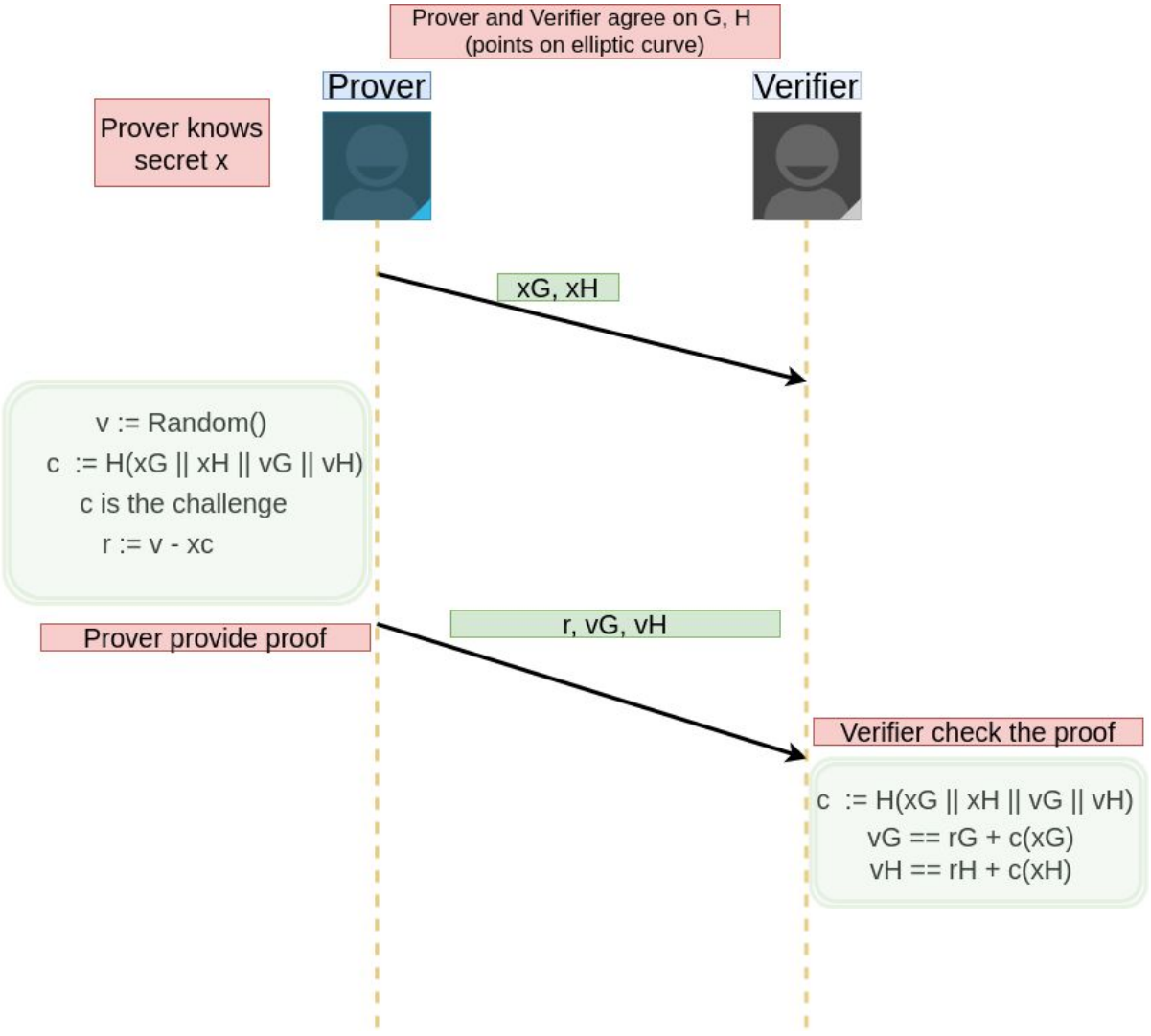

Figure 10. Workflow of non interactive zero knowledge proof for authentication

As illustrated in the above figure, the peerster who attempt to propose an encrypted vote in current round will first generate xG, xH and share them with the others. It then formulate a random challenge using the seed from random oracle. The proof containing vG and vH are sent to the receiving trustee (verifier) afterwards for verification. The difference from the standard NIZKF is that, the secret x is shared among all the trustees. As a result, all the trustee may compute the xG and xH locally before any verification starts.

The advantages of the authentication mechanism are

1. Non-interactivity: The verifier does not need to send any information to the prover

2. Zero knowledge: The secret itself is hidden during the communication step. This invisibility property, which also holds at verifier, can be exploited to generate more general authentication of peerster in other application scenario.

For illustration, we have built a frontend of Peerster to provide fake proposals. The following figure depicts an untrusted Peerster who attempt to propose a vote from voter 2 in election 1.

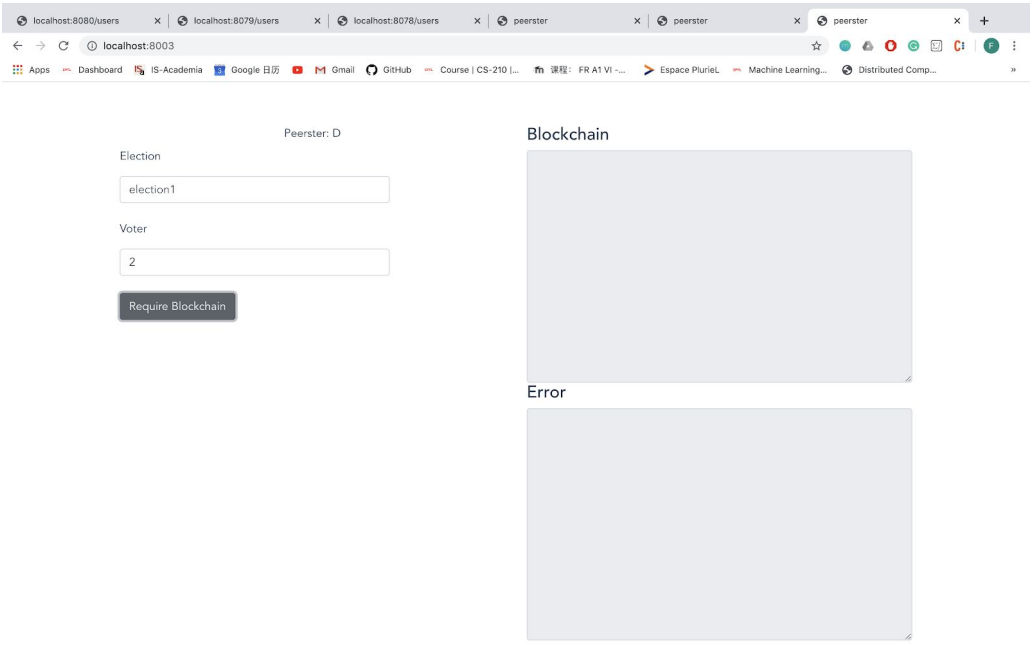

Figure 11. GUI of an untrusted Peerster attempting to propose a vote

The trustees will then be able to capture the attack and log it in their GUI.

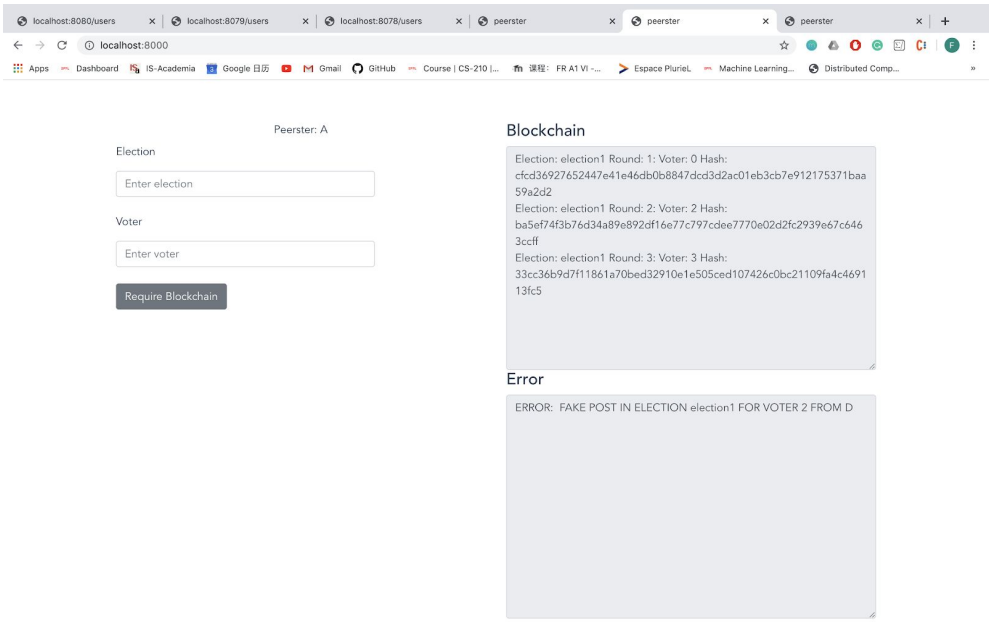

Figure 12. GUI of a trustee who capture the attack

3.5.2 Consensus of the blockchain

The vulnerability of centralized trust of unique trustee is evident in e_voting system since the integrity crashes once the trustee is hacked. To resolve this issue, we distribute the trust among several

trustees and instruct them to construct a ledger of encrypted votes throughout the election. The properties achieved by our blockchain consensus portion include:

1. Universality: All the trustees generate same ledger throughout the election
2. Integrity: Any conflicting vote will be detected by the systems.
3. Robustness: Any valid vote is added to the blockchain in the end of election as long as one trustee received the vote from the independent server.
4. Non-modifiability: Any modification to the blockchain will be detected once the blocks have been added to the ledger and at least one trustee is honest.

Each round of consensus in the blockchain can be illustrated by the following figure:

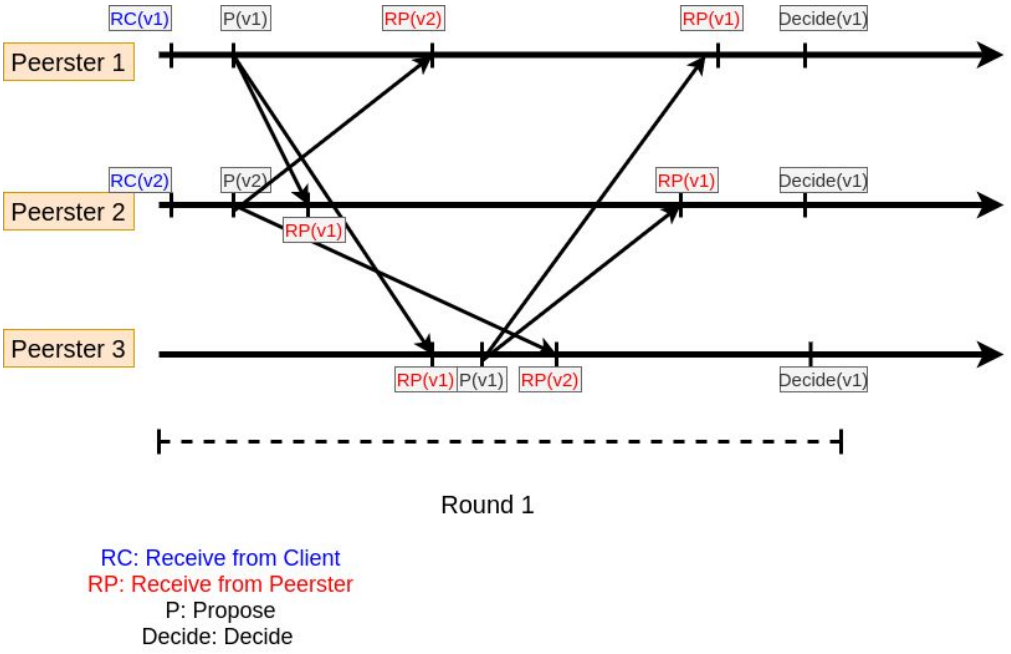

Figure 13. A typical consensus round

1. At the beginning of each round, a Peerster may receive a proposal from the client (independent server) or pop a proposal from its buffer. It then generates a candidate block with a random fitness value and proposes it to other peers using rumor mongering implementation in the Peerster developed from the homework.
2. A Peerster receiving a proposal will check the validity of the proposal by comparing its prevHash field with the currentHash field of the last block in its local blockchain at first. For a valid proposal, it will either create a new proposal from it and transmit the new one to other peers if it has not sent any proposal in current round. Otherwise it will buffer the proposal for block decision at the end of the round.
3. Due to the assumption that no trustee may fail (as needed for partial encryption and decryption), one trustee can expect to obtain proposal from all other trustees. Upon receiving proposals from all trustees at a certain round, a Peerster runs a universal decision algorithm to select the decided block. Specifically, the algorithm works by selecting the proposal with the largest fitness value. To handle the case with duplicated maximals, Peerster id is used to break the symmetry.

Even though the aforementioned consensus algorithm is indeed simple. It provides the desired properties due to our assumptions of trustee and homomorphic encryption. Specifically, integrity is assured by checking that every voter should have the same record in all the peers.

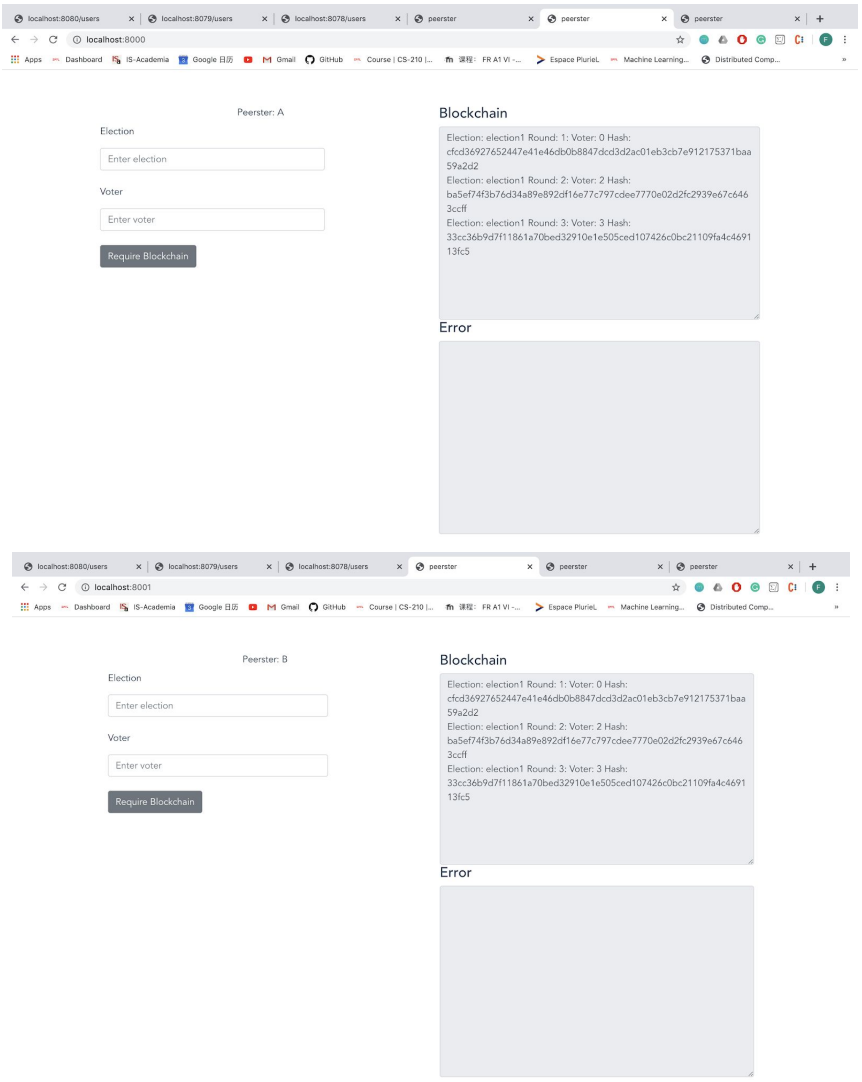

Figure 14. GUI of two trustees illustrating the blocks in the chain

3.6 Miscellaneous

- **The generation of public key and partial private key:** Once receiving a piece of new election information, each peerster will generate its own RSA partial private key. For the public key, we propose to use some technology like secure sum protocol [7] which resembles the idea of round robin. We will further check the feasibility, if not, we will alternately build our system based on the assumptive existence of grouped public key.
- **Progress Diagram:** The whole process graph is displayed in figure 3.
- **UI Implementation:** We implemented the user interface in HTML and Javascript. And specifically, we choose VueJS as the framework for the routing among the different sites.

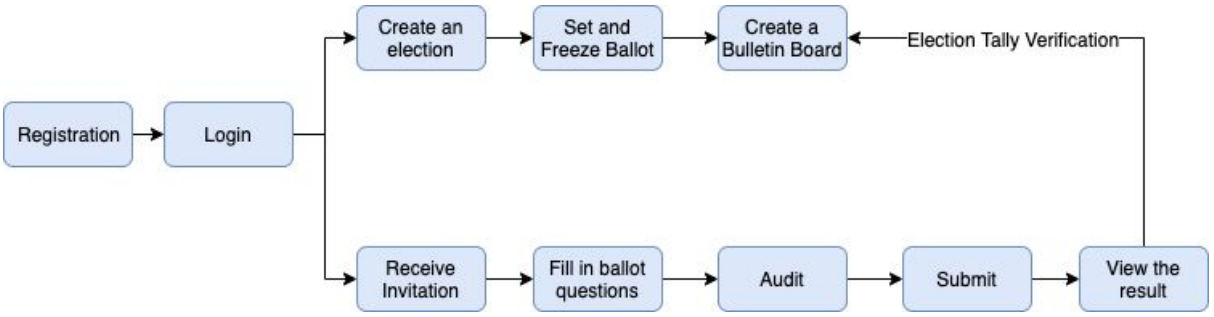

Figure 15. The pipeline diagram of the whole process from the perspective of participant

4. Comparisons with Peerster & Work Distribution

The architecture of this project does not differ much from Peerster's as it focuses more on adding features by extending the current system or re-work some part of it including the private messaging and blockchain. Specifically, the trustees are indeed built upon Peerster. The trustee take advantage of the underlying gossip mechanism of Peerster to perform communication by taking the encrypted vote as a variant of rumor. All common communication stages in Peerster such as rumor mongering, status synchronization and peers routing are exploited to support the trustee. Several aforementioned functionalities such as authentication using non-interactive zero-knowledge proof and blockchain has been added to the Peerster to develop the required trustee.
As for the project responsibilities, the tasks are distributed as follows:

- Decentralized tallying using Homomorphic Encryption (voting, election creating, tallying, trustees) and overall system design: Ali El Abridi, Fengyu Cai
- Decentralized storage using Blockchain and Trustee authentication: Liangwei Chen, Fengyu Cai
- System Engineering including frontend, backend server, tallier, independent server etc. : Fengyu Cai, Liangwei Chen

References

[1] D. Tourky, M. ElKawkagy and A. Keshk, "Homomorphic encryption the “Holy Grail” of cryptography," *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2016, pp. 196-201.

[2] M. Hirt and K. Sako, “Efficient Receipt-Free Voting Based on Homomorphic Encryption,” Advances in Cryptology — EUROCRYPT 2000 Lecture Notes in Computer Science, pp. 539–556, 2000.

[3] N. Islam, K. M. R. Alam, S. Tamura and Y. Morimoto, "A new e-voting scheme based on revised simplified verifiable re-encryption mixnet," *2017 International Conference on Networking, Systems and Security (NSysS)*, Dhaka, 2017, pp. 12-20.
doi: 10.1109/NSysS.2017.7885795

[4] Ben Adida. 2008. Helios: web-based open-audit voting. In Proceedings of the 17th conference on Security symposium (SS'08). USENIX Association, Berkeley, CA, USA, 335-348.

[5]F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa and G. Hjálmtýsson, "Blockchain-Based E-Voting System," *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, 2018, pp. 983-986.

[6]G. G. Dagher, P. B. Marella, M. Milojkovic, and J. Mohler, “Broncovote: Secure voting system using ethereum’s blockchain,” in Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP, Funchal, Madeira - Portugal, Jan 2018, pp. 96–107.

[7]A R. Sheikh, B. Kumar and D. K. Mishra, “A Distributed - Secure Sum Protocol for Secure Multi-Party Computations,” Journal of Computing, USA, Vol. 2, Issue 3, March 2010.

[8]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.