ThumbsUp - A Secure Voting System: Design and Implementation with Shamir's Secret Sharing Leveraging Homomorphic Property of Lagrange Interpolation

Christopher Jeffrey - 13520055
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): christopherjeffrey492@gmail.com

***Abstract*—ThumbsUp is a secure voting system designed to keep secret voters vote to everyone but themself, while still allowing the voting to happen. Using Shamir's Secret Sharing and the homomorphic properties of Lagrange interpolation this goal can be achieved with some other benefits such as adding encryption to provide data integrity. Traditional voting systems often expose voters to privacy risks and potential biases. The system facilitates voting on binary yes/no questions and is implemented with Firebase and NextJS, providing real-time capabilities. Although ThumbsUp enhances security, it has limitations, including vulnerability to man-in-the-middle attacks and restriction to binary voting. Despite these constraints, the system demonstrates the potential for a more secure and robust voting mechanism with further development and refinement.**

Keywords—Voting, Secret Sharing, Realtime, Homomorphic Secret Sharing, Web Application, Firebase

I. Ease of Use

The implementation of this voting system would leverage Shamir's Secret Sharing. The implementation of this system leverages shamir's secret sharing, homomorphic encryption.

A. Shamir's Secret Sharing

Secret sharing algorithm is a cryptographic technique, used for hiding information by splitting it into multiple secrets. We call these secrets shares. The idea is that you need all the shares to recover the secret information. It's very easy to visualize this using a treasure map. Imagine there's a secret map that can be used to find the location of the treasure.This map then would be splitted into multiple parts, and then hidden away in multiple locations across the world. The idea is that the person who wants to find the treasure needs to find all the hidden pieces in order to find the hidden treasure. If someone only has one part of it, they wouldn't be able to find the treasure.

We will refer to this information as 'secret', which is the information that we're trying to hide. We would then use this secret to create shares. We refer to this action as generating. We turn shares back to secret, as reconstructing. The implementation can be done in several ways. One of the way to do it is using lagrange interpolation

B. Threshold Scheme

Let's say an information is splitted into 5 shares. You would need all 5 of the shares to be able to recover the information. But sometimes, you just want at least, for example, 3 shares. This would be useful if for example this information is a key to other information, and you want the majority to agree to unlock that information. We refer to this as threshold schemes. Threshold scheme is a scenario where you split the secret information into, let's say, n shares. Then you need at t shares in order to reconstruct the secret information, where t is less than or equal to n. This mechanism was created by Shamir in 1979, called the Shamir Threshold Scheme.

C. Homomorphic Encryption

Homomorphic encryption is the encryption of information (into cipher text), that we would still be able to analyze and work with by doing mathematical computation such as addition and multiplication. Some famous encryption that we use daily in this modern age support this feature, but typical encryption doesn't necessarily support this. There are two types of homomorphic encryption. Partially homomorphic encryption, and fully homomorphic encryption. Partially homomorphic encryption. Partially homomorphic encryption only one operation. That can either be addition, or multiplication. Fully homomorphic encryption supports both addition and multiplication operation.

II. Quick Small Scale Voting

The idea is you want to be able to do computation on a data that is represented using a number, without revealing the number. For example two parties (party A and party B) each give their number, like 12 and 23. They want to compute the sum. Typically, this can be done by giving the information to one of the parties, for example to party A. But such implementation can't be done in this scenario, as party A would know the number form party B. Another way of doing this is by using the service of a trusted third party. Such

implementation would suffice, if there actually is a third party where all the parties involved in the 'number summing' event can be trusted. But that wouldn't always be the case.

Another reason why you would want to hide people's vote is to prevent human biases. For example, if a class wants to decide the class leader out of two candidates, they can simply raise their hand. This would lead to fast decision making, and easy counting. But it has a downside. People might get persuaded when someone they trust, their close friends, or the majority for people has the same vote. This would lead to a less objective voting, with questionable results.

The idea is to create a system where all the information that is exposed in public is sufficient to calculate the result, without revealing any of the voters' vote.

III. Design of ThumbsUp

An implementation is created to demonstrate a secure voting system, which is called ThumbsUp. This implementation uses a yes no question, and would leverage lagrange interpolation partial homomorphic property.

ThumbsUp would be implemented in rooms. Each room is created for a one time voting purpose with their own voters.

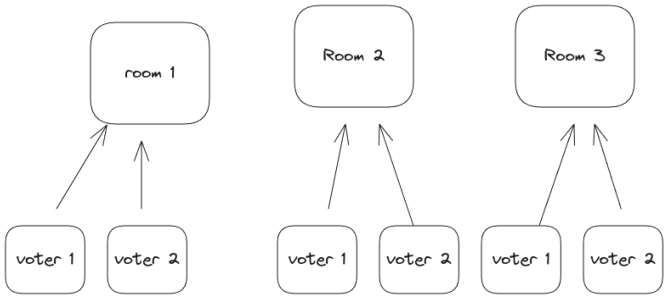

Image 1. An illustration of 3 voting rooms. Each room has 2 voters.
source: writer's archive

There are two roles, room creator and voter. Room creator would be responsible for handling the room which includes the flow of the voting. The second one is the voter.

Here's the flow of the voting

1. Room creator creates the room. They would define:
    a. Room name. This part will be displayed on the room list. People would use this to identify the room they want to join.
    b. People included in that voting room. This number has to be exact. So it's not the maximum number of people that's allowed to join or the minimum number of people that is needed to start the voting, it's the exact amount of people that has to join to allow the voting to happen. The minimum number of people needed is two, while the theoretical maximum is unlimited. The limiting factor for the maximum number of people would tend induced by the practical things, such as the hardware of the network
    c. Question shown in the voting room. This information would be the number one guide information that is used by the voter to determine the context of the voting.
2. Voters find the room and Voters join the room. Voters would then go to a page with a list of rooms where they can choose to join.
3. After all voters join the room, the room creator starts the voting. The voting can only be started once the number of participant is exactly the same amount of voters stated previously.
4. Voters do the voting, it's a yes or no question. The question would be shown on top, to guide the voters.
5. When all voters have done the voting, the calculation can be started. Calculation will be explained more later in this paper.
6. Last, the result will be shown to each voter. Each voter would do the calculation locally on their device.

The voting is done with a binary output, which is yes and no. This binary information would need to be translated into some form that allows addition and multiplication. The binary vote would be encoded into a number. For example we can use 1 for yes, and 0 for no.

When the calculation is started, shares would be generated from each secret. The number of shares would follow the number of voters. If there's 10 voters, each vote would generate 10 shares.

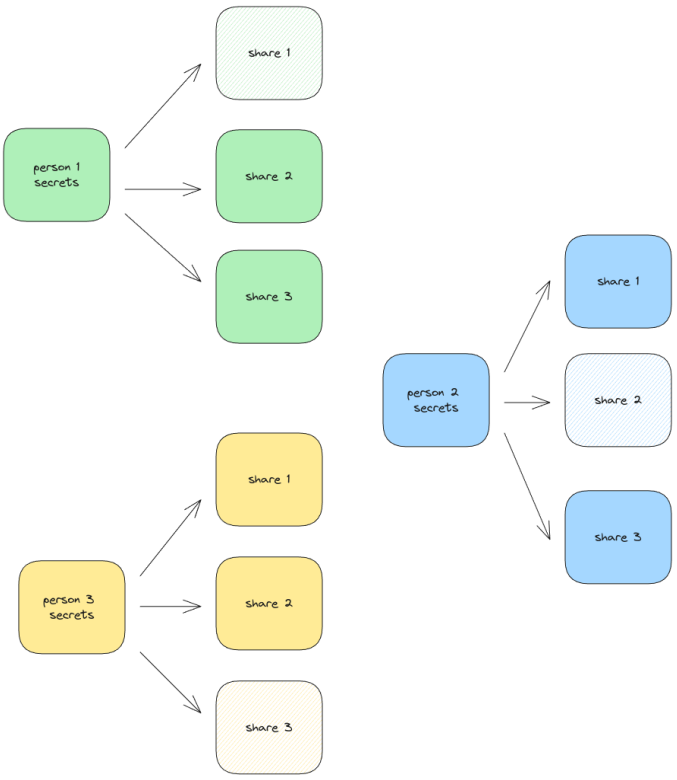

Image 2. A room with three voters. Each secret would generate 3 shares.
source: writer's archive

Number of threshold (t) would be the same as the number of shares (n).

Each voter would keep a single share. This share would never leave them, so that the secret can't be constructed back. The other shares can be sent to the server.

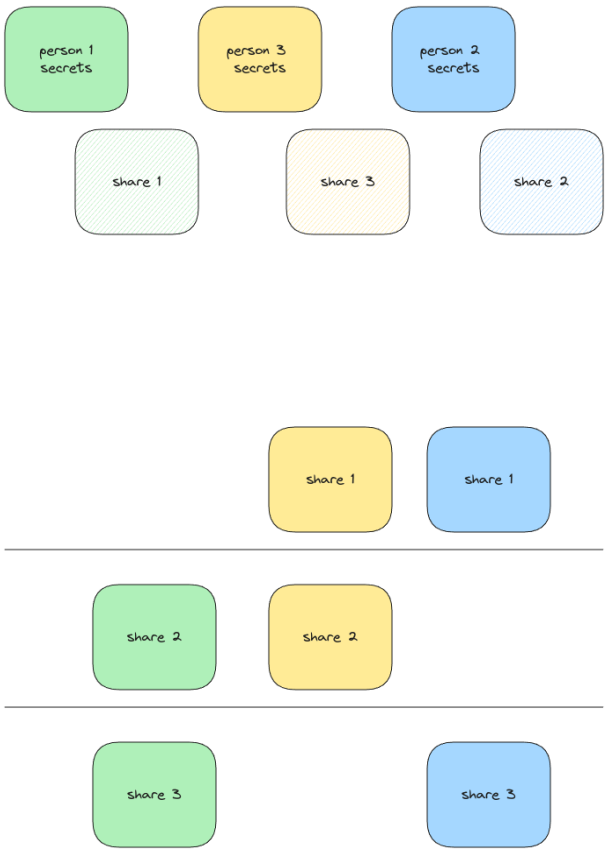

Image 3. The shares that is generated would be shared together, while each of the voter keep 1 share to themselves. source: writer's archive

After everyone has computed their shares and sent their shares, we can continue to the next step which is computing the sum.

The calculation has to be done by every voter. They would grab a single share from each other voter, which corresponds with the index of their kept secret. The homomorphic part comes to play here. Shares from every other voter would be added together. We call this the sum.

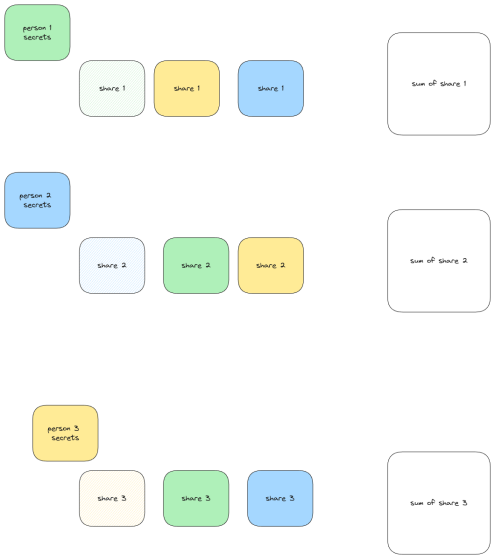

Image 4. Shares of the same index by the other will be taken and added with the share that is kept secret, each generating a sum. source: writer's archive

After every voter calculated their sum, this information would then be shared to the server.

Last, we just need to combine all the sums into a new secret. This new secret is the sum of all secrets.

Image 5. Each sum then would be gathered together and combine to create a single share that would represent the whole sum of the vote. source: writer's archive

The sum of all the secrets then translated back to the number of votes yes and no, according to the encoding. A video demonstration can be seen here to help illustrate the steps.

A. Rules

To be able to pull this off, there are some rules that need to be followed.

- First, everyone has to join first before the voting starts. The number of people is fixed, and can't be changed in the middle of the voting. This is due to the

number of voters who would determine the number of shares, which can't be changed on the fly.
- Everyone has to do the vote first before the calculation of the vote is being done. Calculations require all the shares to be created first.
- The encryption has to be done using the same encryption key. This is done to ensure the homomorphic property of lagrange interpolation.
- To add obfuscation, the secret is going to be slightly changed between all the voters. This is done to make sure the encrypted shares are not the same between each voter. This can be done by simply adding the index of the voter.

B. Real time and Polling

For the implementation of ThumbsUp, the system needs to update the condition of the system as soon as possible. There are two main ways to achieve this, using realtime websocket, or polling. Realtime would be more ideal for this, as polling is worse for the internet.

C. Database Consideration

1. Redis

Redis is a noSQL database which focuses on speed. It was previously used as caching, but it's actually stable enough to be used as a primary storage solution. One of the features that redis has which is very suited for this application, is expiration. This would be useful when we want to remove the room automatically after a certain time. But this route would force polling.

2. GunJS

GunJS is a decentralized database library. Key advantage from GunJS is that it allows peer to peer. This would increase the security of the system, as the information wouldn't be centralized. However, this route would force polling.

3. MongoDB

MongoDB is a NoSQL database. MongoDB describes itself as a document database that can be used for many use cases. Though it's very flexible, it can't be implemented on the client side. Because of this, this would require a separate backend service. This would increase the complexity quite much. If polling is used, race conditions might also happen.

4. Firebase

Firebase is a backend cloud computing services and application development platform provided by Google. It has a plethora of services, from authentication to database. For this application, the most interesting service is the real time database. It has a noSQL structure, and like the name suggests, would allow the system to have real time experience. This would provide a better experience for the user, and would allow the system to be easy on the network. The library that is created by Firebase also is very well established for the NodeJS library.

IV. Stack Used

The implementation of ThumbsUp uses Firebase. This implementation is relatively easy, and allows real time connection directly from the Application's frontend. The Application uses NextJS. This would ensure that no special hardware is required, platform agnostic, and would allow the app to be run pretty much everywhere. The usage of framework would also help the implementation to focus on the cryptographic side instead of the web app side, as many things has been covered already by the framework.

Shamir's Secret Sharing would be implemented using a library https://pypi.org/project/shamirs/ in python. Since the library is in Python, this would force the design to be splitted into smaller pieces, backend and frontend. The backend would be used as a wrapper for the Shamir's Secret Sharing Algorithm. To help ease up the implementation, a framework called FastAPI would be used. It's intended for python and has lots of features to reduce boilerplate which would make it easy to focus on the implementation of the Shamir's Secret Sharing.

V. Thumbs Up in Practice

First, the number of pages and use cases needs to be defined. For the room creator, the use cases would be:

1. Create a room
2. Close a room
3. Trigger start voting
4. trigger start calculate

For the voter, the use cases would be:

1. Find a room
2. Join a room
3. Cast a vote
4. Do calculation
5. See the result

We then use this information to determine pages needed:

1. Landing and Join Room

Landing would be placed at route /, which would be the typical first page to open. This page would be the entry point for both voter and room creator. If they're room creators, they can choose the "Create Room" button. If they're a voter, they can click the "Join Room" button.

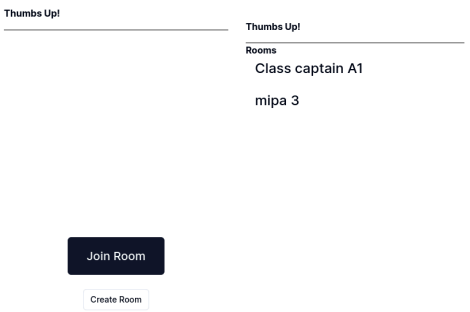

Image 6. Landing page and join-room page.
source: writer's archive

If they click Join Room, a list of rooms will show up. They then can choose the room they want to join.

2. Room Creators' Room

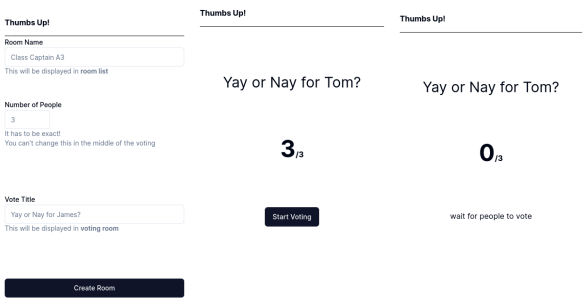

Image 7. Page for creating room, and controlling the flow of the voting.
source: writer's archive

Room creator would control the flow of the program. The main flow controlled by the room creator is:

- Waiting for the people to join,
- start the voting,
- trigger the calculation, and
- lastly deleting the room.

3. Voting Room

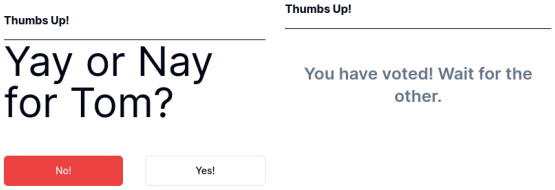

Image 8. Voting Room
source: writer's archive

After all the voters have joined the room, the room creator can start the voting. The voter will be shown a Yes and No question. After all that has finished the result of the voting will be shown.

VI. Codes and Analysis

```
useEffect(() => {
  // update query params
  const query = new URLSearchParams();
  query.set('voterID', voterID.toString());
  query.set('step', step.toString());
  router.push(`/vote/${params.roomID}?${query.toString()}`);
  if (step == 1) {
    // get the secret of my own index from everyone
    const secretsFromEveryone = currentRoom?.voters.map((voter: any) => {
      return voter.secret[voterID];
    });
    // fill in my own secret
    secretsFromEveryone[voterID] = secrets[voterID];

    console.log(secretsFromEveryone);

    // sum them up
    let sum = secretsFromEveryone.reduce((acc: number, secret: any) => {
      return acc + secret.value;
    }, 0);

    sum = sum % secretsFromEveryone[voterID].modulus;

    set(ref(firebaseDB, `rooms/${params.roomID}/voters/${voterID}/sum`), sum);
  } else if (step == 2) {
    // get the sum of my own index from everyone
    const sumsFromEveryone = currentRoom?.voters.map((voter: any) => {
      return voter.sum;
    });
    combineShares(sumsFromEveryone);      You, 3 days ago • checkpoint: main fl
  }
}, [step]);
```

Image 6. example code for calculating the Sum, and combining all the shares back.
source: writer's archive

This implementation relies on every voter to finish the calculation before being able to continue the calculation. So the flow of the program is a huge deal and requires lots of attention. Realtime helps a lot by providing a callback as an entrypoint for starting the next step in the flow. Many libraries are used to help achieve this, such as Jotio for global state management, and shadcn/ui for the base of the UI. Tailwind is also used to speed up the styling process.

Most of the code is implemented in TypeScript, with tsx format. TypeScript is a superset of JavaScript, which is the primary language used in web applications. It offers everything that javascript supports, with additional features that focus on type safety and linting. This feature would help so much during the development, by catching early mistakes that are made by the programmer.

VII. Future Improvement

There are many rooms for improvement based on this implementation, few that sticks out the most are:

1. This implementation is relatively weak. The share that is held back can be interpreted if the sum is known by all the people.
2. Current implementation allows everyone to read all the sum, and all the shares. This can be avoided by using peer-to-peer from all nodes to every other node, though it's quite heavy on the network. But if a person spies on all the connections, they could still be able to decipher the secret.
3. Current implementation only supports Yes and No questions. This can be expanded to a voting scheme which relies on non categorical information, such as rating.

VIII. Conclusion

Here are conclusions that can be derived while implementing ThumbsUp:

1. It has really good potential. It might still be relying on the implementation, but it adds one more layer of obfuscation.
2. If there's a man in the middle using this implementation, it can still guess the vote that is casted by everyone.

Github LInk

The program that implements ThumbsUp is shared publicly, which can be seen here https://github.com/christojeffrey/thumbsup

Video Link at Youtube

A video demonstrating ThumbsUp can be seen here https://youtu.be/Ao4lPbHvBPo

Acknowledgment

Thank you to Dr. Ir. Rinaldi Munir, MT who has helped so much for the material for this paper, and for the love and knowledge he has shared.

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/35-Skema-Pembagian-Data-Rahasia-2024.pdf

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/38-Enkripsi-homomorfik-2024.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Christopher Jeffrey 13520055