Encrypted Ballot

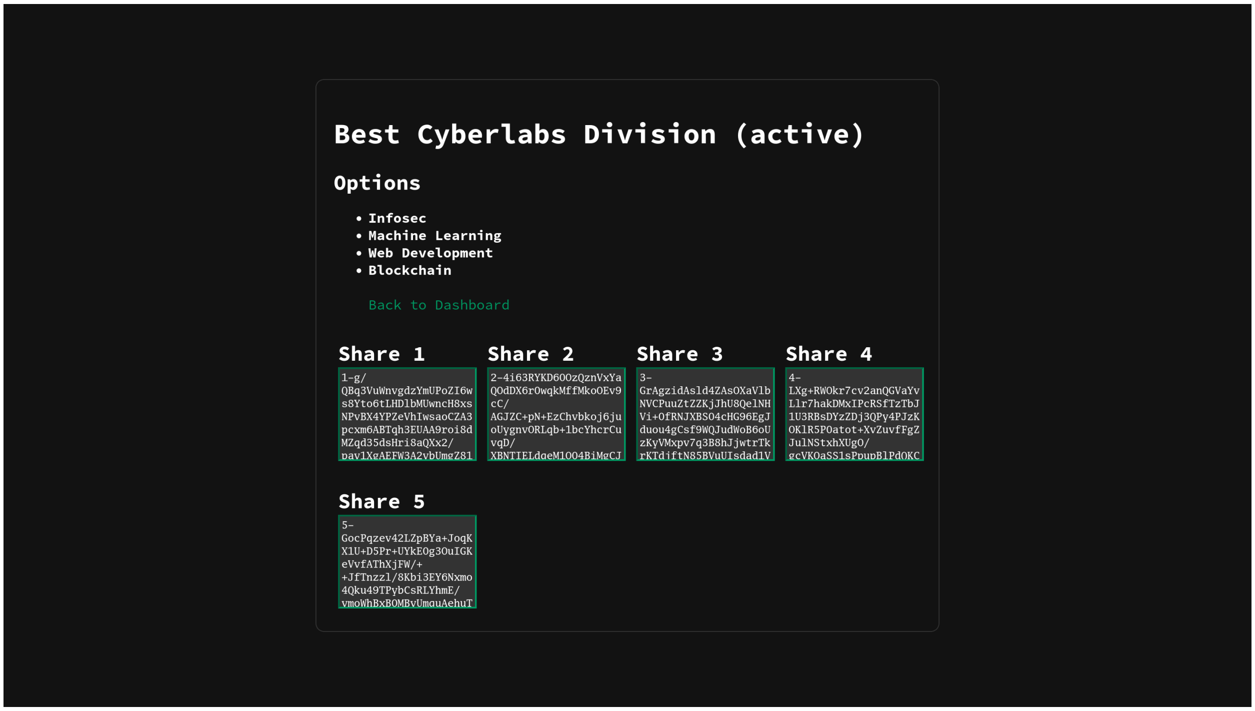
Rachit Kumar Pandey, 24je0677@iitism.ac.in
https://github.com/armoredvortex/woc

Election Creation

The election process begins with an administrator creating a new election. At this stage, a public-private key pair unique to the election is generated. The public key is used for encrypting votes, while the private key is essential for decrypting the results.

To prevent risks associated with a single point of failure, the private key is divided into multiple shares using **Shamir's Secret Sharing** scheme.

> **Problem: Shamirs Secret Sharing**
>
> A secret S is represented as the constant term of a random polynomial $f(x)$ of degree $t-1$:
>
> $$f(x) = S + a_1 x + a_2 x^2 + \dots + a_{t-1} x^{t-1}$$
>
> The admin generates n shares by evaluating $f(x)$ at n distinct, non-zero values $(x_1, x_2, \dots, x_{t-1})$.
>
> Each share is a point $(x_i, f(x_i))$.
> To reconstruct the secret, at least t shares are required and the original polynomial $f(x)$ is recovered hence $S = f(0)$ is revealed.


Best Cyberlabs Division (active)

Options

- Infosec
- Machine Learning
- Web Development
- Blockchain

Share 1

```
1-g/
QBq3VuWnvgdzYmUPoZI6w
s8Yto6tLHDlbMUwncH8xs
NPvBX4YPZeVhIwsaoCZA3
pcxm6ABTqh3EUAA9roi8d
MZqd35dsHri8aQXx2/
pay1XgAEFW3A2vbUmgZ81
```

Share 2

```
2-4i63RYKD60OzQznVxYa
QOdDX6rOwqkMffMkoOEv9
cC/
AGJZC+pN+EzChvbkoj6ju
oUygnvORLqb+1bcYhcrCu
vqD/
XBNTIELdgeM1OO4BiMgCJ
```

Share 3

```
3-
GrAgzidAsld4ZAsOXaVlb
NVCPuuZtZZKjJhU8QelNH
Vi+OfRNJXBSO4cHG96EgJ
duou4gCsf9WQJudWoB6oU
zKyVMxpv7q3B8hJjwtrTk
rKTdiftN85BVuUIsdad1V
```

Share 4

```
4-
LXg+RWOkr7cv2anQGVaYv
Llr7hakDMxIPcRSfTzTbJ
1U3RBsDYzZDj3QPy4PJzK
OKlR5POatot+XvZuvfFgZ
JulNStxhXUgO/
gcVKOaSS1sPpupBlPdOKC
```

Share 5

```
5-
GocPqzev42LZpBYa+JoqK
X1U+D5Pr+UYkEOg3OuIGK
eVvfAThXjFW/+
+JfTnzzl/8Kbi3EY6Nxmo
4Qku49TPybCsRLYhmE/
vmoWhBxBOMByUmguAehuT
```

Voter Registration and Voting

Voters register through a portal, then they are granted access to a list of active elections. Upon selecting an election, a voter can cast their vote. Each vote is encrypted using the election's public key.

Homomorphic Tallying

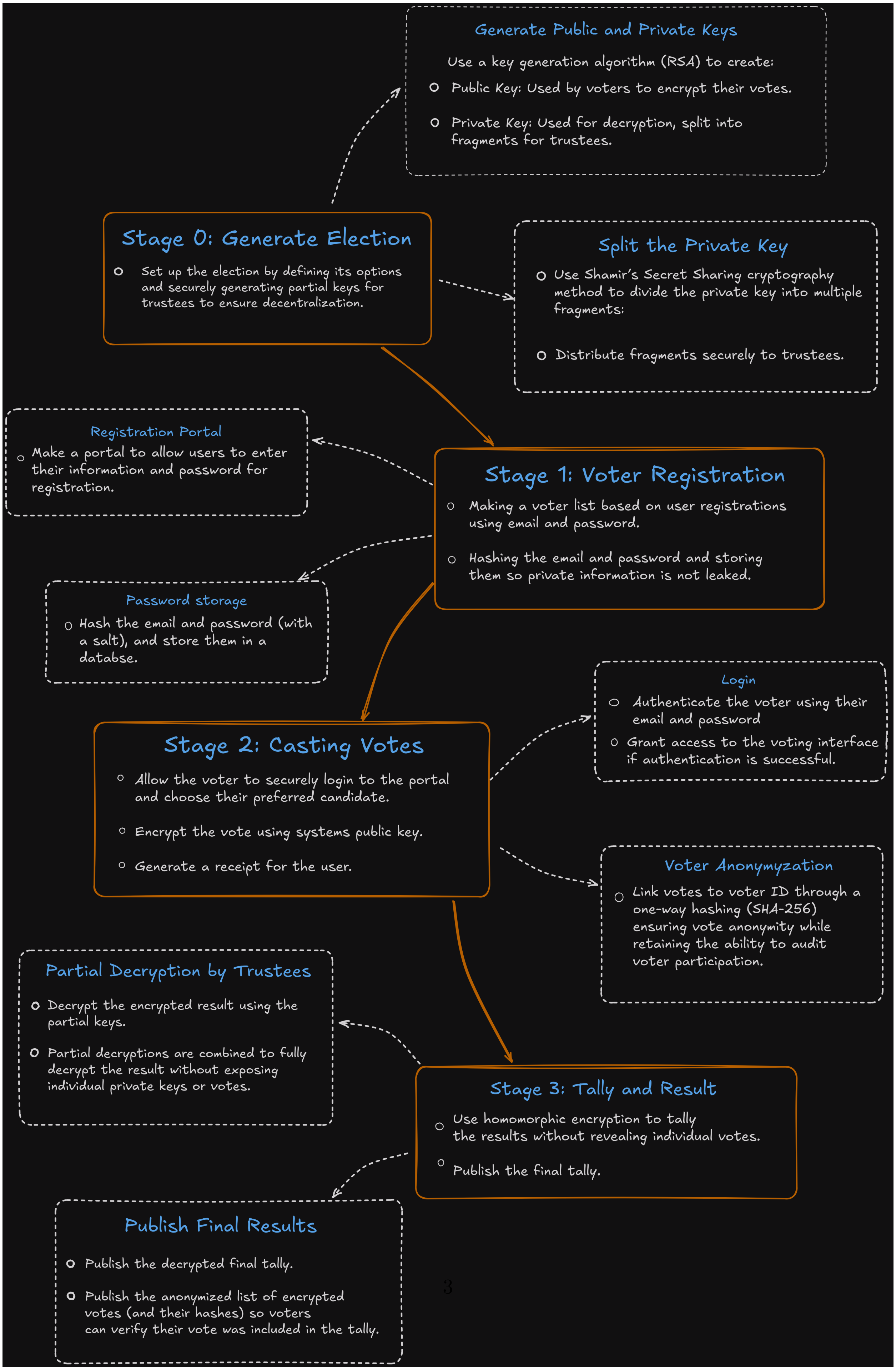
After all the encrypted votes are collected, the encrypted votes to be added directly without requiring decryption at this stage. The Project uses **Paillier Cryptosystem**

> **Problem: Homomorphic Addition**
>
> **Public Key:** (n, g) and **Private Key:** (λ, μ) For plaintext m,
>
> $$c = g^m \mod n^2$$
>
> To Decrypt:
>
> $$m = L(c^\lambda \mod n^2) \cdot \mu \mod n$$
>
> $$L(x) = \frac{x-1}{n}$$
>
> To Add messages, we just multiply the ciphertexts:
>
> $$c_1 \cdot c_2 = g^{m_1+m_2} \mod n^2$$

To reveal the final tally, the system requires the cooperation of trustees. Using their individual shares of the private key, the trustees reconstruct the key only if the minimum threshold is met. This reconstructed key is then used to decrypt the added up ciphertext, giving back the election result.

Closing Thoughts

This Project was really fun to work on and I learned a lot of the math behind how cryptography works behind the scenes.
I have also attached a snippet from my original Proposal in case I missed something.



Stage 0: Generate Election

- Set up the election by defining its options and securely generating partial keys for trustees to ensure decentralization.

Generate Public and Private Keys

Use a key generation algorithm (RSA) to create:

- Public Key: Used by voters to encrypt their votes.
- Private Key: Used for decryption, split into fragments for trustees.

Split the Private Key

- Use Shamir's Secret Sharing cryptography method to divide the private key into multiple fragments:
- Distribute fragments securely to trustees.

Stage 1: Voter Registration

- Making a voter list based on user registrations using email and password.
- Hashing the email and password and storing them so private information is not leaked.

Registration Portal

- Make a portal to allow users to enter their information and password for registration.

Password storage

- Hash the email and password (with a salt), and store them in a databse.

Stage 2: Casting Votes

- Allow the voter to securely login to the portal and choose their preferred candidate.
- Encrypt the vote using systems public key.
- Generate a receipt for the user.

Login

- Authenticate the voter using their email and password
- Grant access to the voting interface if authentication is successful.

Voter Anonymyzation

- Link votes to voter ID through a one-way hashing (SHA-256) ensuring vote anonymity while retaining the ability to audit voter participation.

Stage 3: Tally and Result

- Use homomorphic encryption to tally the results without revealing individual votes.
- Publish the final tally.

Partial Decryption by Trustees

- Decrypt the encrypted result using the partial keys.
- Partial decryptions are combined to fully decrypt the result without exposing individual private keys or votes.

Publish Final Results

- Publish the decrypted final tally.
- Publish the anonymized list of encrypted votes (and their hashes) so voters can verify their vote was included in the tally.