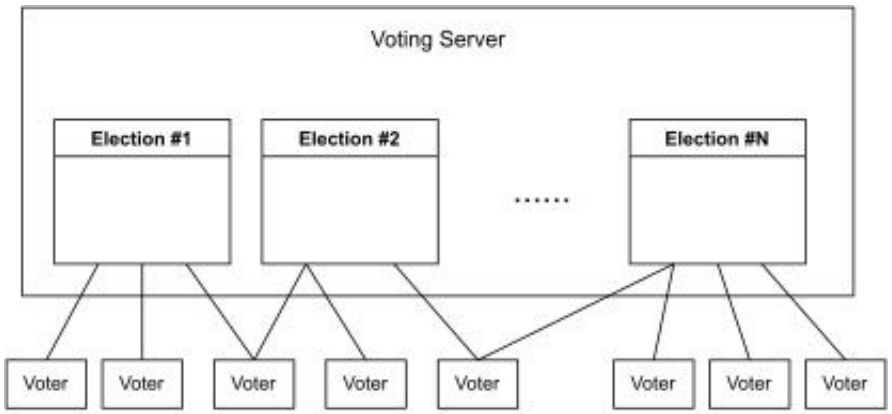
(FTC) Online Voting – Part II. Single-Node e-Voting Server

In the project, you will implement the voting logic of the e-Voting server. We assume that there is just one single voting server. The voting clients will connect to the voting server to participate in the elections.



A. Local Server API

`RegisterVoter(Voter) returns (Status)`

We can register a new Voter by calling the RegisterVoter API.

Return Value

Status.code=0 : Successful registration

Status.code=1 : Voter with the same name already exists

Status.code=2 : Undefined error

Parameters

Voter.name : Name of the voter (e.g., “王大明”)

Voter.group : Group of the voter (e.g., “同學”)

Voter.public_key: The voter's ed25519 public key (based on the libsodium implementation).

`UnregisterVoter(VoterName) returns (Status)`

This API is used for unregistering a voter with the name VoterName.

Return Value

Status.code=0 : Successful unregistration

Status.code=1 : No voter with the name exists on the server

Status.code=2 : Undefined error

Parameters

VoterName : Name of the voter (e.g., “王大明”)

B. RPC APIs

```
rpc PreAuth (VoterName) returns (Challenge)

rpc Auth (AuthRequest) returns (AuthToken)
```

The RPC APIs use authentication tokens to verify the identities of the voter clients. An authentication token can be acquired by a challenge and response handshake with the voting server. First, a voter client calls `PreAuth(VoterName)` to get a challenge, where the VoterName is the name of the voter client. The voting server will return a Challenge.

The voter client will respond to the challenge by calling `Auth(AuthRequest)`, where

AuthRequest.name is the name of the voter client.

AuthRequest.response is the detached signature of Challenge.

The voting server needs to verify if AuthRequest.response is a valid ed25519 signature signed by the voter client's ed25519 secret key. If the AuthRequest.response is valid, the server will return a random and unique authentication token AuthToken back to the client.

An authentication token will be expired after **one hour** since its creation. When the authentication token gets expired, the voter client needs to perform the authentication process again to get a new authentication token.

It is the responsibility of the server to track the life of the authentication tokens. The server needs to keep the mapping between the client identities and the active authentication tokens.

```
rpc CreateElection (Election) returns (Status)
```

A registered voter client can create a new election on the sever by calling `CreateElection (Election)`, where

Election.name: The name of the election (e.g., "學生會會長選舉")

Election.groups: The list of groups eligible to participate in the election (e.g., {"大學部學生", "研究所學生"})

Election.choices: The list of choices (e.g., {"林帥哥", "鄭美女"}}

Election.end_date: The date and time when the election will be closed (e.g. `2023-01-01T00:00:00Z`)

Election.token: The authentication token.

Return Value

Status.code=0 : Election created successfully

Status.code=1 : Invalid authentication token

Status.code=2 : Missing groups or choices specification (at least one group and one choice should be listed for the election)

Status.code=3 : Unknown error

```
rpc CastVote (Vote) returns (Status)
```

A registered voter client can cast a vote in an ongoing election on the server.

A voter client can only cast once in each election.

Return Value

Status.code=0 : Successful vote

Status.code=1 : Invalid authentication token

Status.code=2 : Invalid election name

Status.code=3 : The voter's group is not allowed in the election

Status.code=4 : A previous vote has been cast.

Parameters

Vote.election : Name of the election (e.g., "學生會會長選舉")

Vote.choice_name: The choice (e.g., "林帥哥")

Vote.token: The authenticaiotn token

```
rpc GetResult(ElectionName) returns (ElectionResult)
```

One can query the result of a completed election by calling `GetResult(ElectionName)`.

Return Value

When the query is successful

ElectionReuslt.status = 0

VoteCount.counts: The list of choices and the ballot counts

When the query is unsuccessful

ElectionResult.status = 1: Non-existent election

ElectionResult.status = 2: The election is still ongoing. Election result is not available yet.

Parameters

ElectionName: Name of the election to be queried

Please prepare the following for submission

1. Package your source code in a zip file.
2. Prepare a README describing how to build and run your code
3. Write a report that describes the status of your implementations (which parts are implemetned & which parts are not implemented). Also, in the report, please include the evaluation results to demonstrate the correctness of your implementation.
4. Upload everything (code, README, and report) to E3.

For questions regarding the project, please use the forum in E3 system or contact TA <m2955121314.11@nycu.edu.tw>