E-Voting System using C and Blockchain Concepts

112103001
112103060
112103053

TABLE OF CONTENTS

INTRO

The system aims to create a prototype in C for a more dependable E-Voting model that works around the problems of the current physical system.

ABOUT THE PROBLEM

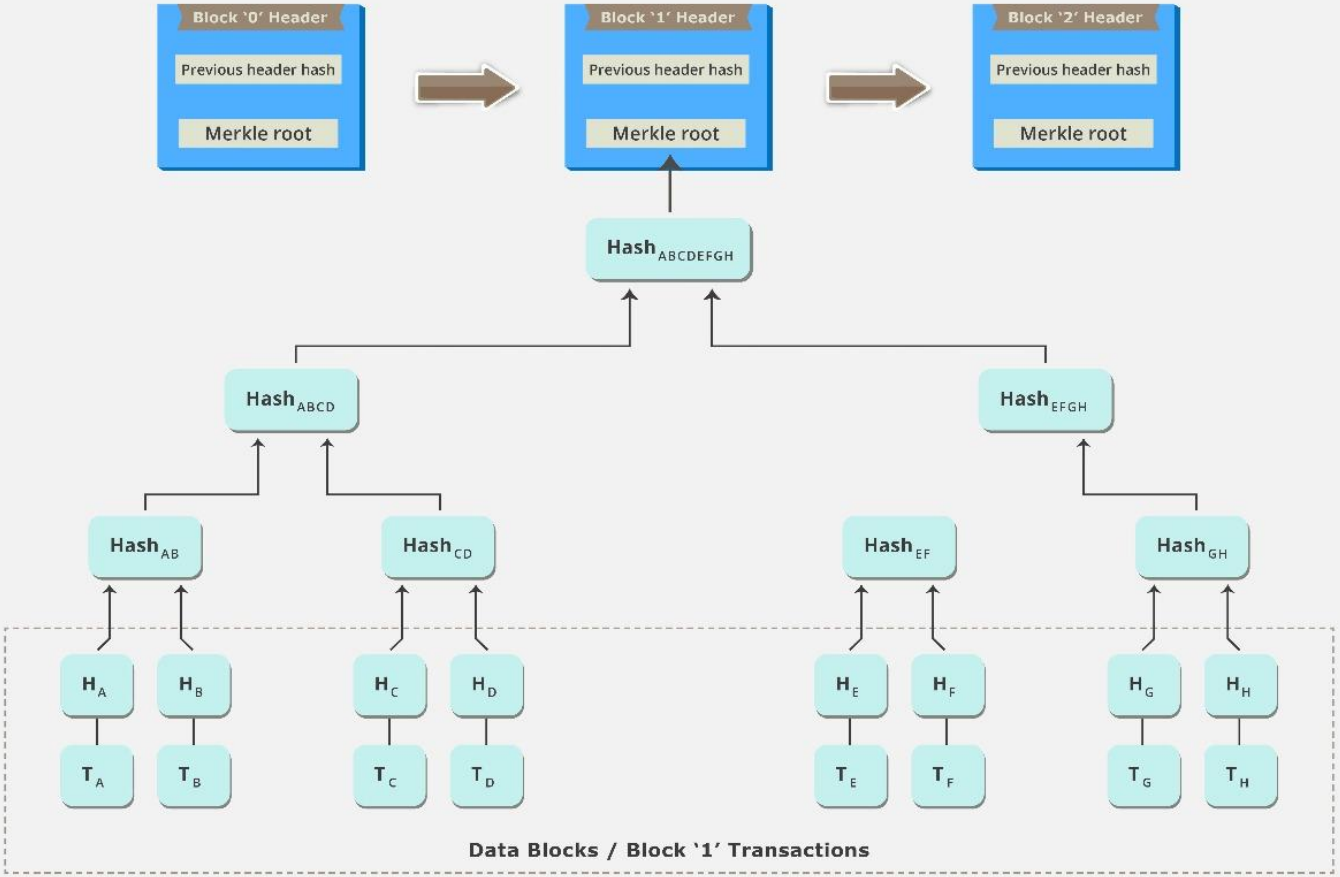
Objective:

- Creating **transparency** in voting system using concepts of blockchain.
- We can trace any **tampering** of the votes by checking the unique cryptographic hashes given to every node.
- More **efficient** and **convenient.** Reduces manpower required to authorize a person's identity.

OUR APPROACH

- We got inspired by the concept of blockchain in which every transaction is stored, encrypted and decentralized. Blockchain focuses on the security and immutability of data.
- This makes E Voting an ideal application for blockchain.
- We decided to implement a small prototype in C inspired by these concepts.
- Various data structures such as Merkle Trees and Hash Maps will be used.

Merkle Tree


Block '0' Header
Previous header hash
Merkle root
Block '1' Header
Previous header hash
Merkle root
Block '2' Header
Previous header hash
Merkle root
Hash ABCDEFGH
Hash ABCD
Hash EFGH
Hash AB
Hash CD
Hash EF
Hash GH
H A
H B
H C
H D
H E
H F
H G
H H
T A
T B
T C
T D
T E
T F
T G
T H
Data Blocks / Block '1' Transactions

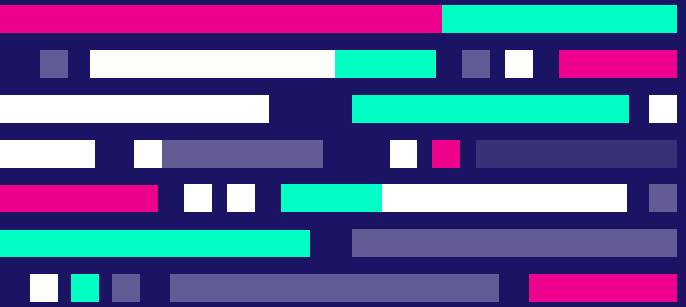
Basic Overview :

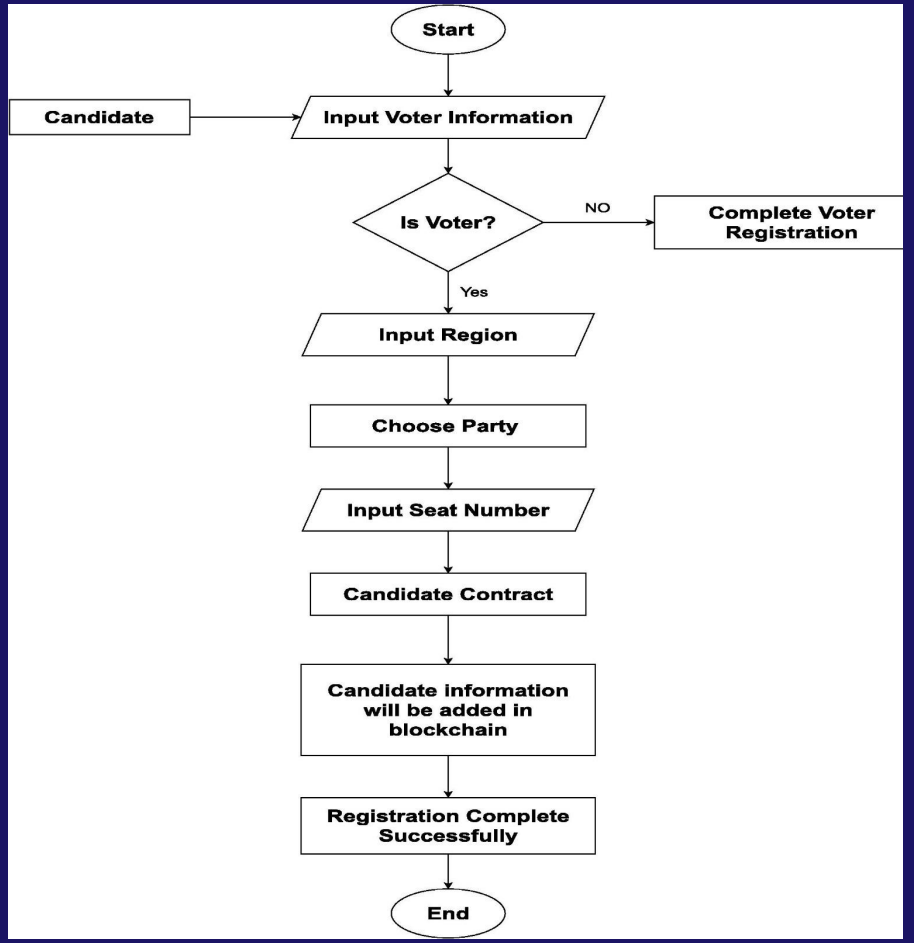
Structure:
Merkle tree is a data structure where each non-leaf node is a hash of it's child nodes.

Our Unique Selling Point:
Create a voting token for every citizen to spend once on the candidate of their choice. This prevents casting of multiple votes by a single voter.

Encryption:
A hash function maps an input to a fixed output and this output is called hash.
The output is unique for every input and this enables fingerprinting of data.
So, huge amounts of data can be easily verified through their hash.
Any tampering can be easily traced by comparing hashes.

OUR SOLUTION





LITERATURE SURVEY :

First Research on Blockchain:
https://bitcoin.org/bitcoin.pdf

E-Voting System Reference Research Paper:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9787540

Recommended course:
https://archive.nptel.ac.in/courses/106/105/106105235/

Dataset:

- Storing the votes of different candidates.

- Voter ID to authenticate the person and token value.

- Candidate IDs to cast your vote.

IMPLEMENTATION

We implemented our project in C with the following data structures

- Merkle trees
- Hash maps
- Doubly linked list

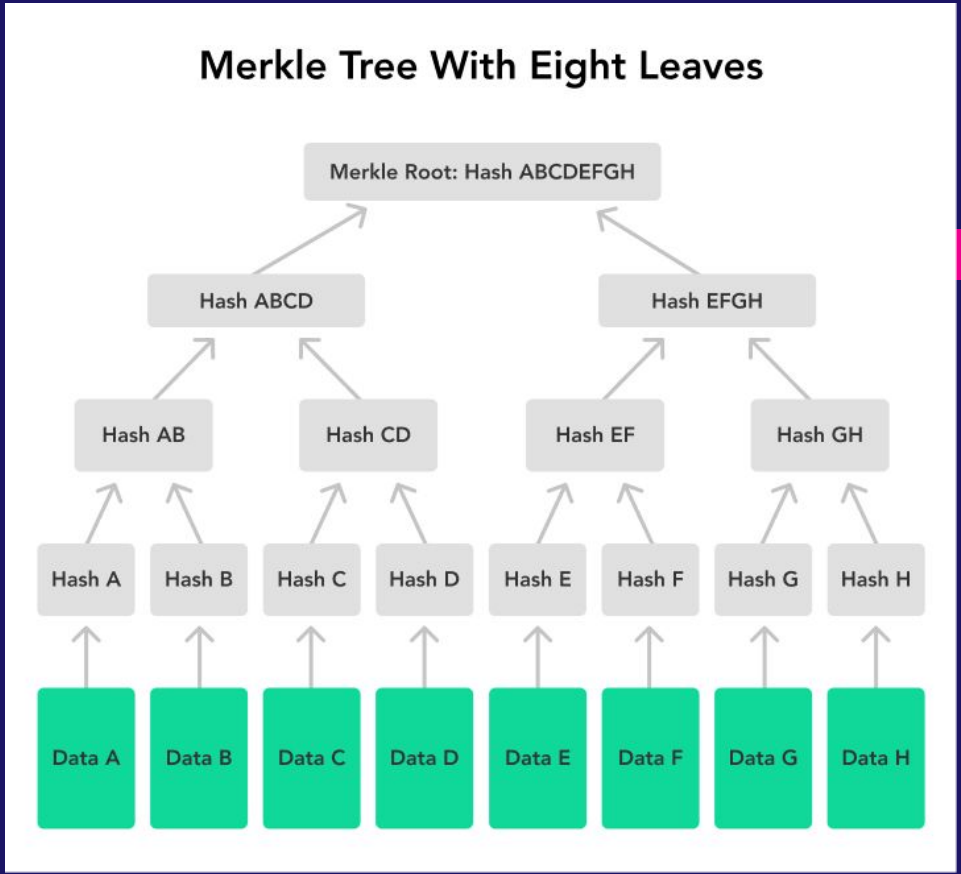
We used multiple csv files during our implementation to make our database strong, and make our system usable.

BASIC PROCESS

- A Region for conducting the voting is selected by the administrator
- The voting cycle starts in which :
  - The voter is asked for their voting ID
  - The ID is authenticated
  - The vote is cast and recorded
  - Vote is hashed
  - It is stored as a transaction in the Merkle Tree
- The root of the tree of a region is stored in a block.
- Many region Blocks are connected in the form of a doubly linked list.
- Blocks are also hashed in a chain.

Merkle Tree

1. Data A is input by user i.e. candidate
2. Hash A is the concat and dual hashing with getCurrentTime function
3. Hash AB with same dual hashing function
4. The root always changes with every new vote leaf entry.
5. It is hence a Complete Binary Tree.



Hashing

It is a way to encrypt data with the unique hash function(s) only the admin has access to. We have two: `unsigned int hash(char *str)` and `unsigned int hash_function(unsigned int input, unsigned int prev_hash)`

1. The first one encrypts a string into one hash. [DATA TO LEAF]

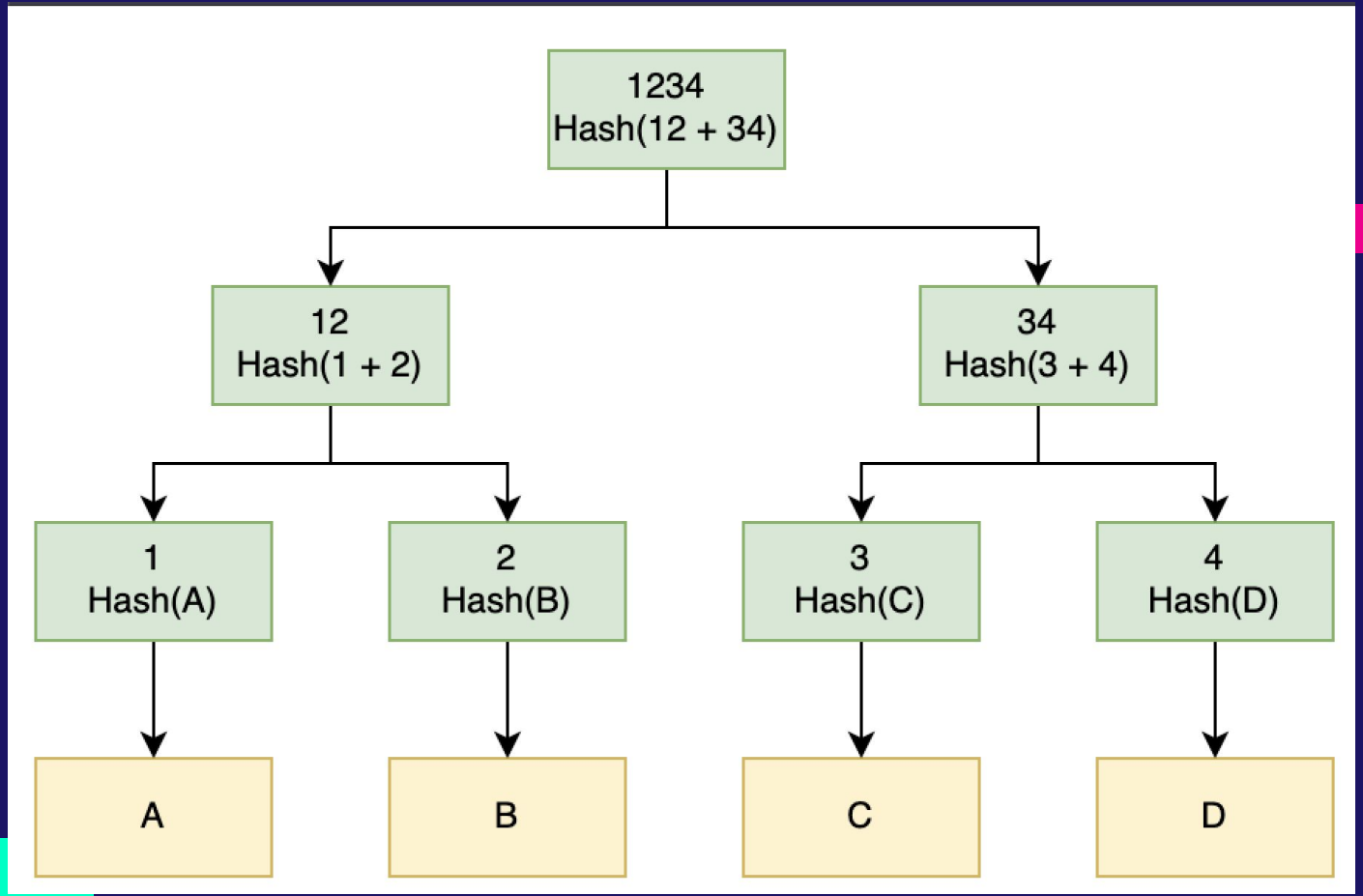
2. The second one encrypts two strings by concat and then hash method to give one hash. [LEAVES TO INTERNAL NODES]

This hashing ensures that each set of transactions has a UNIQUE fingerprint through a hashmap that can only be produced through our UNIQUE & PRIVATE hash functions.

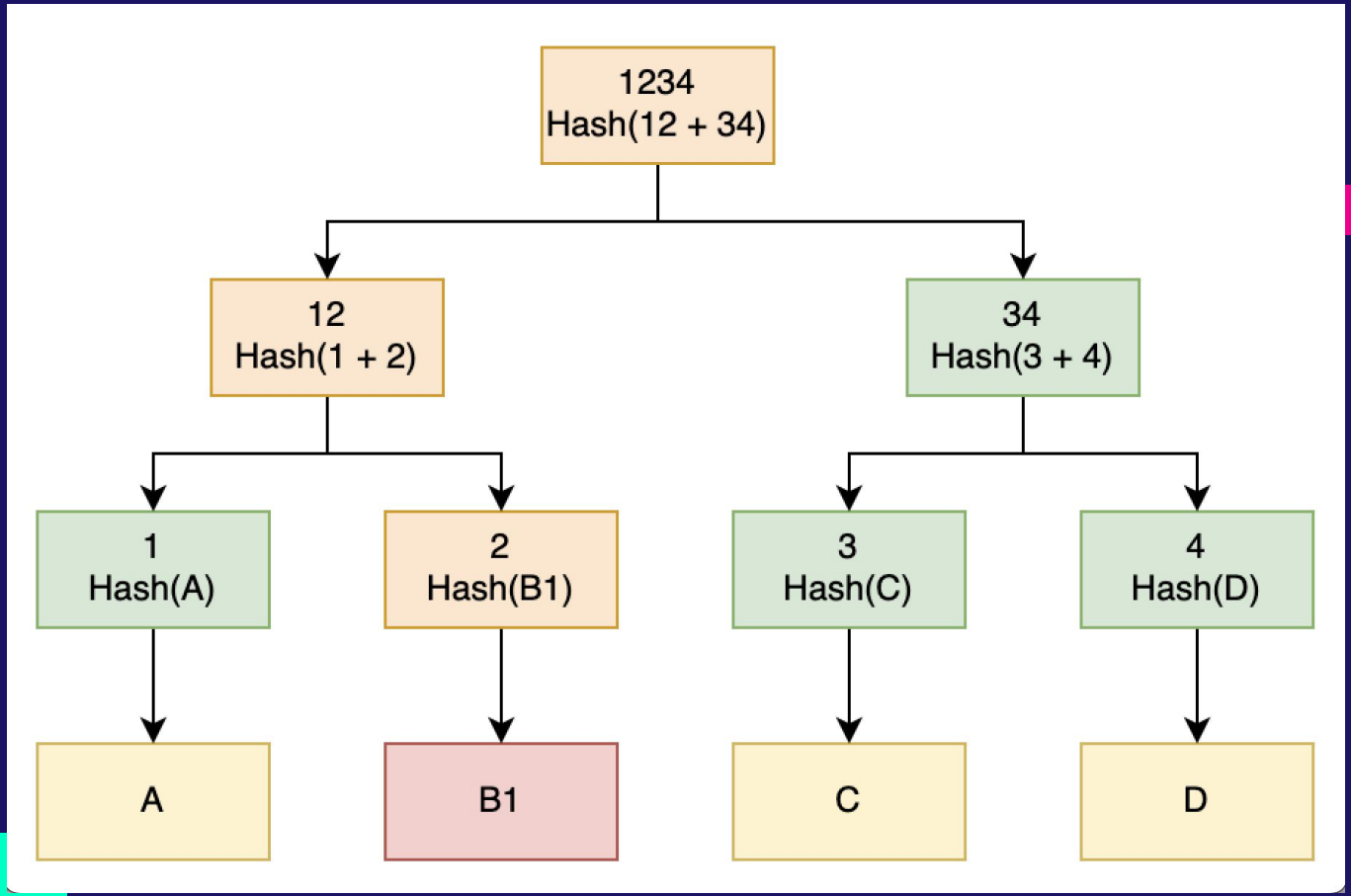
What Challenges Did We Face?

1. Tampering with the stored Merkle Tree CSVs. How to catch?
2. Tampering with the stored Input CSV. How to track?
3. Say both are tampered with?
4. Implemented blockchain in C which increases efficiency and speed as compared to languages like python.
5. Never been implemented in C before, and hence we found no reference coding material and built the project from the ground up.

Verifying Data Integrity



B is altered!!


1234
Hash(12 + 34)
12
Hash(1 + 2)
34
Hash(3 + 4)
1
Hash(A)
2
Hash(B1)
3
Hash(C)
4
Hash(D)
A
B1
C
D

How do we check check overall?

Input would create a merkle that would NEVER be predicted by a malicious actor.

And hence never match the tampered merkle tree in the old csvs.

Steps in our E-voting system

- Checks voter ID in database.
- Checks voting token in database.
- If ID is valid && token is 1, voter is allowed to vote.
- Votes are counted for individual candidates.

Our USP – Voting token

What is a Voting Token?

-It is a boolean value which ensures that one voter can only cast his vote once in the entire process.

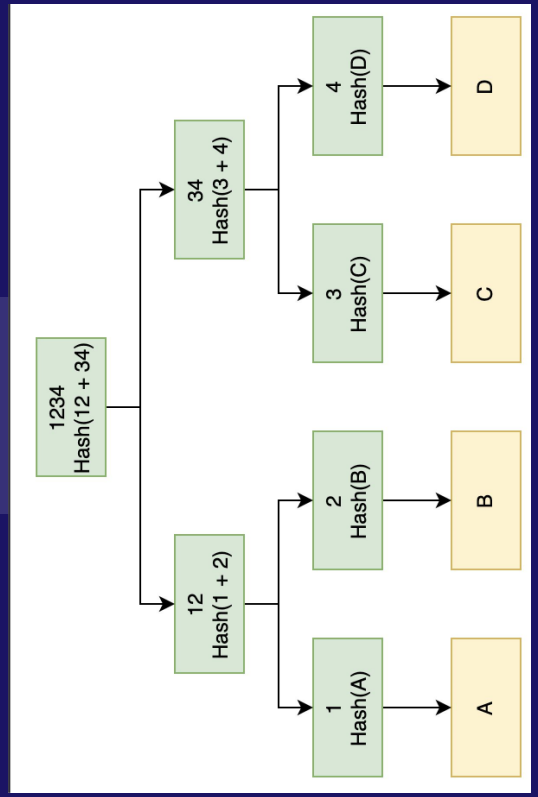
-This eliminates the loophole of a single voter casting multiple votes in an e-voting system and makes system more secure.

How we switch region to region

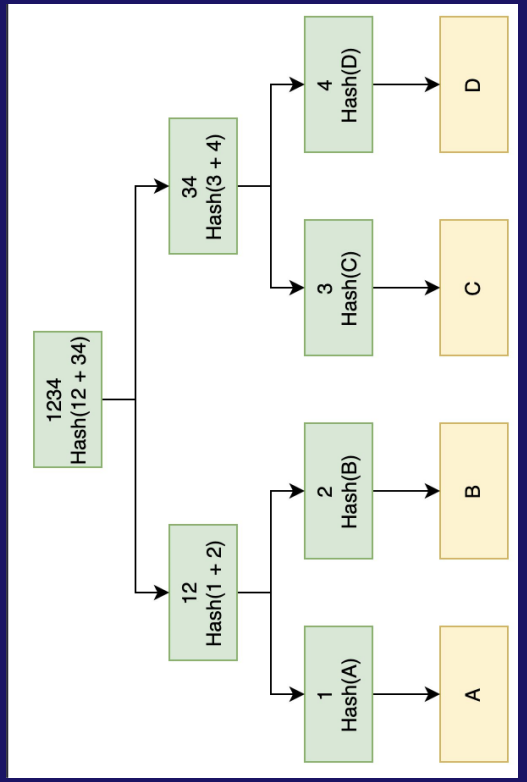
- Separate CSV's are made for each region
- When administrator enters the private region id the CSV for voter IDs as well as the files storing the Hashes for that region are selected.
- Each function dealing with these files goes into the correct file for that region

Security Feature

Old Hash merkle tree

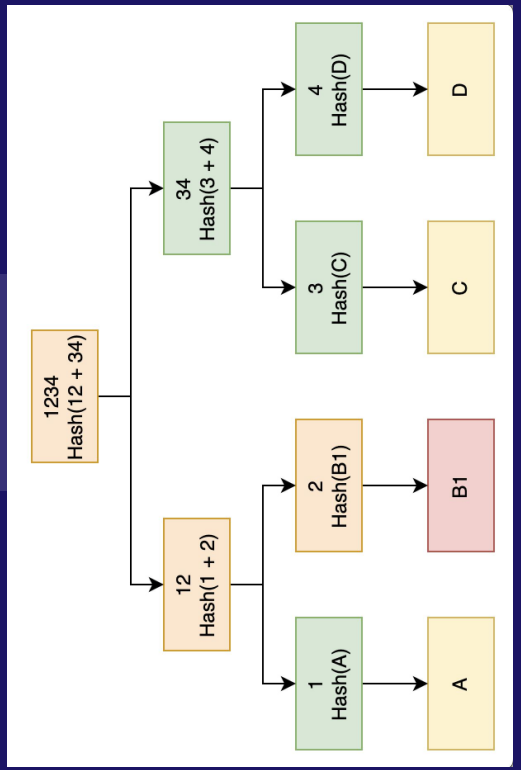


Current hash merkle tree



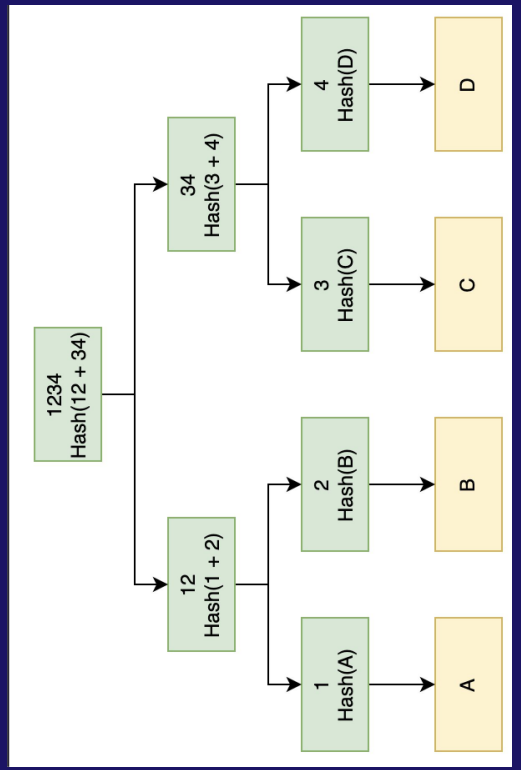
Security features

Tampered tree moved to old

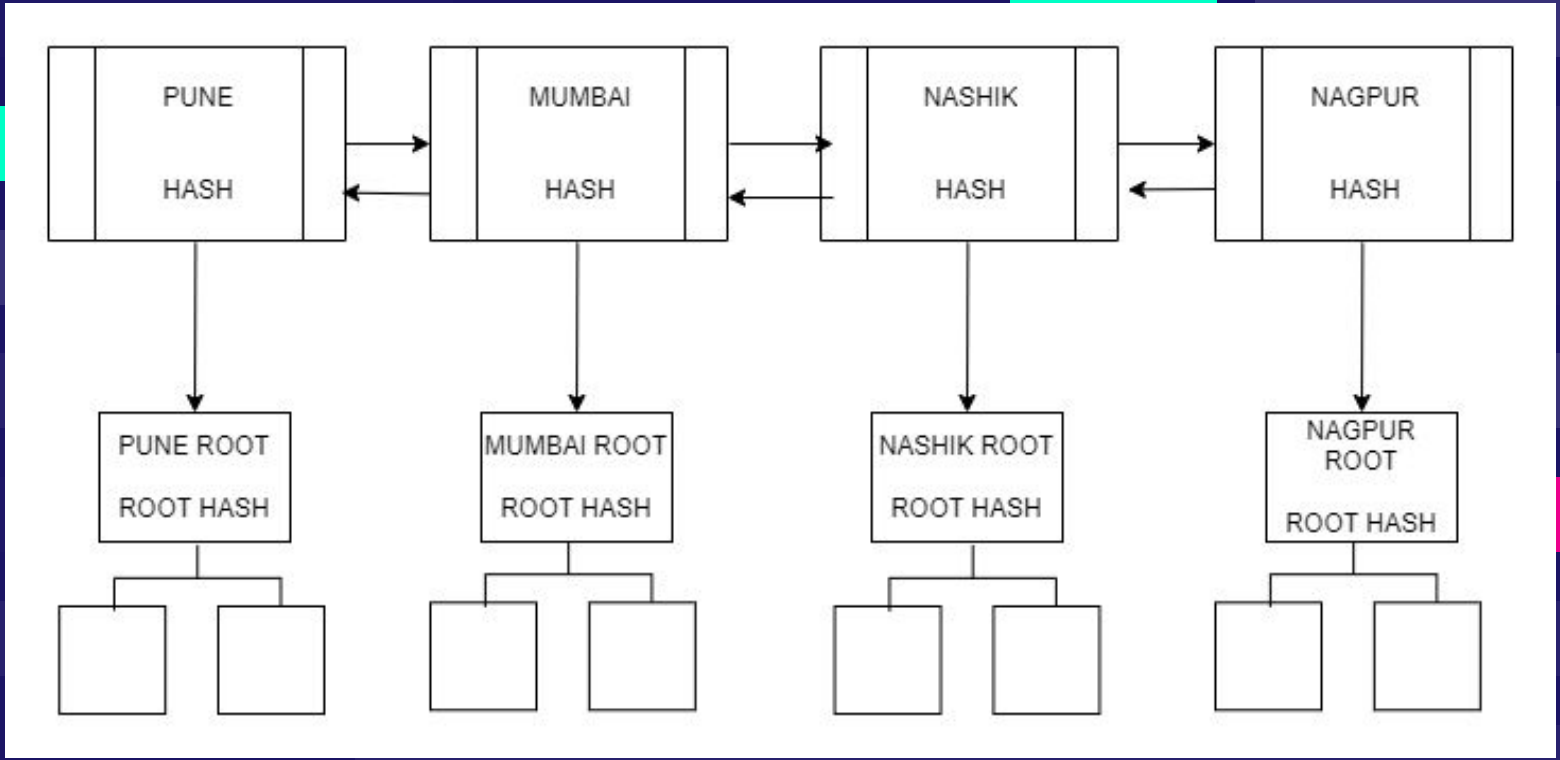


NOT MATCHING

Current hash merkle tree from region cs



BLOCKCHAIN LINKED LIST



Future Scope

- Convert code in a blockchain language like solidity, using the same algorithm and concepts.
- Expand servers and upload the system on a blockchain platform
- Launch it online for users to access voting facilities remotely
- Make the ID verification system more strong, with multiple steps and techniques.
- Create and execute a complex and robust Hashing algorithm like SHA 256
- Create a buffer system to accommodate multiple votes at the same time

THANK YOU!