Digital Electoral Rolls

Prof. Subodh Sharma

Hardeep Kaur 2019CS10354
Anjali Sharma 2019CS50422

COD310, Mini Project
Indian Institute of Technology, Delhi

January 7, 2022

Contents

Introduction

Electoral roll of a region/constituency is a list released by the election commission which includes the names of all the registered voters who have the right to vote for specific elections of a region. It helps to detect electoral fraud by enabling the authorities to verify an applicant's identity and entitlement to a vote. It also helps to ensure that a person doesn't vote multiple times.
Traditionally, electoral rolls were maintained in paper form (hard copies) but nowadays electronic electoral rolls are increasingly being adopted. We would discuss about the challenges accompanied by this transition and also propose solutions to them.

1.1 Electoral Rolls: Public or Private

The process of creation and maintenance of electoral rolls is governed by the Registration of Electors Rules, 1960 [4]. According to Rule 22, the Registration Officer (RO) has to 'publish the roll, together with the list of amendments by making a complete copy thereof available for inspection'. Furthermore, ECI may direct the RO to share the full roll with registered political parties . Thus, the electoral roll of a constituency is essentially a public document and can be circulated on ECI's directions.

1.2 Public Digital Rolls: Security and Privacy

We have identified the following two major caveats in the current digital version of electoral rolls.

1.2.1 Availability of Too Much Information

- For each registered voter, the electoral rolls *inter alia* contain the age, sex, address, name of the relative (father/husband), phone number/email (for those who registered online) and the photo identity card number.
- Availability of this much of PII (Personally Identifiable Information) is a serious security threat because digitization allows for uniformity, standardisation and generalisation of data which makes it quite more vulnerable by allowing it to be searched and categorised in a custom way.
- As Bill Davidow argues in one of his articles, exponential growth in computing power has made search operations faster, cheaper and more convenient.This makes it easy to paint a big picture of a community, by collecting bits and pieces of the voters' information *(Aggregation effect, Digital Person)[1].*

1.2.2 Non-Auditability of actions performed by Authorised Officials

- In the current design, a user with higher level of access, such as ERO or above, can insert, update or delete data of a voter from the rolls. There is no means to verify whether the action was requested by the voter or whether the action (say, an update) was performed as requested.
- Although this update would be visible to the concerned voter but most of the times electors consult the database only near elections. Thus, an authorised official or some malicious user (after stealing his user credentials) may tamper the database and this wouldn't be logged anywhere providing limited or no auditability.

1.3 Requirements of Electoral Rolls

We begin the design of secure and privacy preserving electoral rolls by first listing the requirements of the system, i.e., what all services should it be able to provide and what all properties should hold at each instant, also known as the system invariants.

1.3.1 Requirements

1. An unrolled eligible citizen should be able to register as a new voter.
2. A registered voter should be able to update his data, as and when needed.
3. A provision to delete a voter's data due to death or permanent shifting.
4. A voter should be able to inspect the existence of another voter in a constituency and raise an objection if he finds something wrong/fake.
5. The actions of authorized officials and regulators should be auditable.
6. The PII information should be provided on a *need-to-know* basis.

1.3.2 Properties of Database

1. De-duplication while insertion: Only one active entry for each registered voter.
2. Prevention of fake insertions and deletions: Data is modified only after proper verification.
3. Correctness of data: Data is tamper proof and each transaction is recorded in an immutable digital ledger for constructing the proof of correctness.

1.4 What we achieve

An immutable, append only database for maintaining the electoral rolls is the core of our implementation. The data structures along with the cryptographic primitives used guarantee the tamper evidence of the database. With each transaction, metadata is also stored which contains the information about who performed the action, timestamp and hash pointer to previous block of data. This provides auditability of actions of authorised officials and regulators. We have also implemented merkle tree representation of the database for providing cryptographical proofs to queries like checking membership and verifying database consistency.

Transaction Flow: Big Picture

2.1 Specifications: Interactions of actors with the system

Actor	Roles	Interaction with ER
Unrolled eligible citizen/ registered voter	- apply for registration, request for updation or deletion in ER - inspect electoral roll - raise objection	- fill claim forms for insertion, updation, deletion or objection - access his own data - inspection of electoral roll
Regulator	- verify the insertion, updation and deletion requests - check the validity of objection requests by proper verification - keep a check on the deletions made by AOs	- perform verification process for the claim forms - ensure the disposal of claim forms with verified requests - monitor the deletions made by AOs
Authorised Officials	- prepare, manage and update the electoral roll - update polling stations' data to a centralised database - at root level, ensure mapping of PwD or illiterate citizens in ER - ensure EPIC distribution to all eligible voters in a constituency - publish electoral roll	- insert, update & delete a voter from ER - update polling station data - publish electoral rolls
ECI	- administrative - supervision - policy enforcements	- access metadata for supervision - directs the entities as deems fit
Political Parties	- inspect the electoral rolls - track the changes made during the continuous updation period - monitor false deletions	- access the rolls of a constituency and compare with the supplement of continuous updation period to keep a track the changes - monitor the deletions made by AOs
Third-party researchers	- research purposes	- access to metadata for research

Electoral rolls have a number of stakeholders - from BLO to CEO who are involved in the disposal of claim forms, field verification, management and updation of electoral rolls. They have different levels of access to the system depending on the level (constituency, district or state). We refer to them as 'Regulator' and 'Authorised Officials' and assume that the ones at apt level are called by the system based on the action.

2.2 Transaction Flow

Above interactions can be classified into the following categories: Access, Insert, Update, Delete and Objection. We define two general transaction flows: one for Insert and Update action on database, and other for Delete and Objection requests. The finer details of interaction with database are discussed later .

2.2.1 Transaction Flow [Insert/Update]

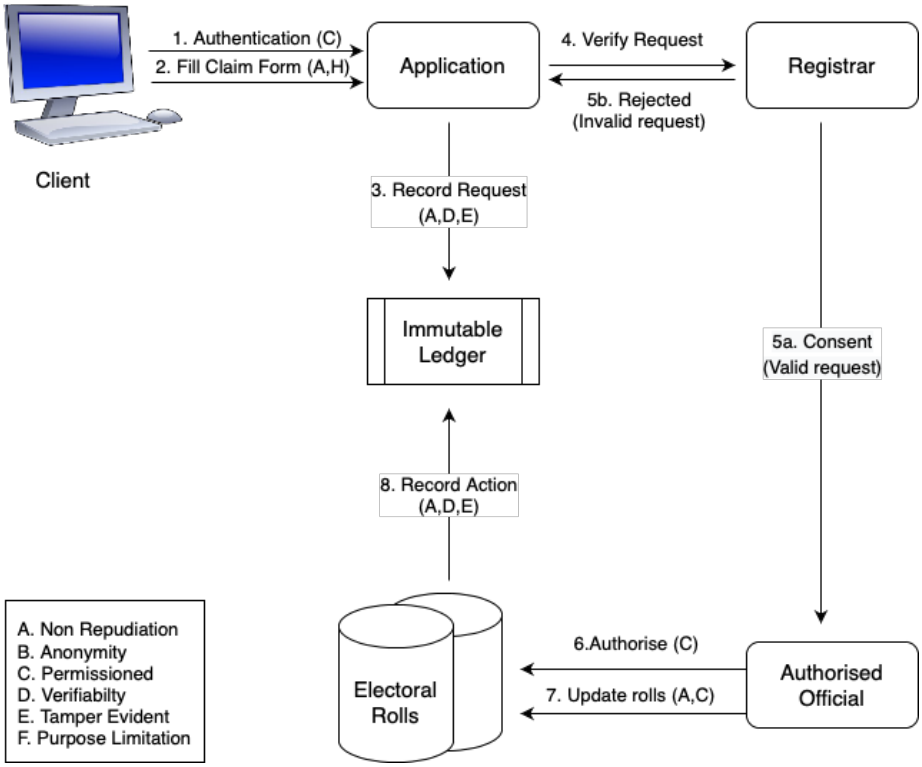

Figure 2.1: Transaction flow for Insert/Update

The letters in parenthesis denote the properties that need to be satisfied at that step.

Brief Explanation of steps

1. The client connects to the application using basic authentication and fills a claim form corresponding to his requested action.
2. The request sent by client is recorded in the immutable ledger for non-repudiability, verifiability and tamper evidence.
3. The request is then sent to the concerned regulator for verification.
4. The regulator would verify the documents provided as proofs and perform other necessary actions (corresponding to the field verification in present system). If the request is found valid, regulator would give his consent.
5. Upon receiving a request from regulator, the authorised official (who lies in the ACL of concerned action) would perform the action on the append only database which would get recorded in the immutable ledger also.

2.2.2 Transaction Flow [Delete/Objection]

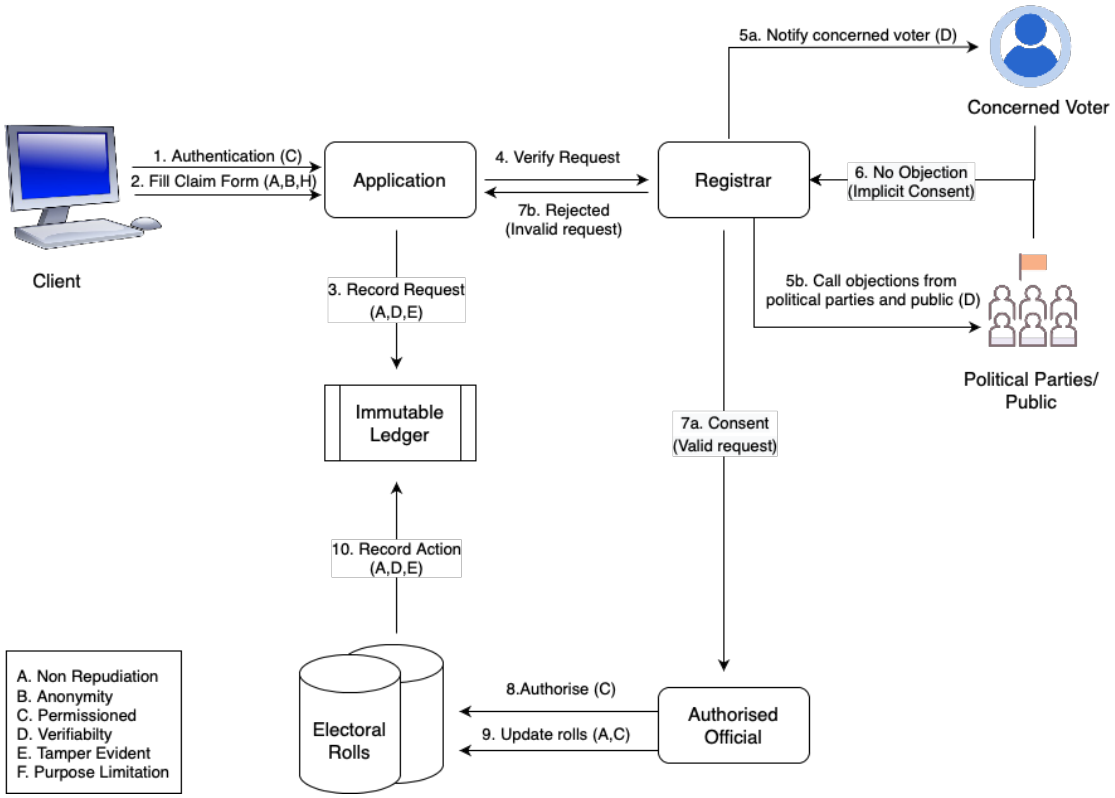

Figure 2.2: Transaction flow for Deletion/Objection

The letters in parenthesis denote the properties that need to be satisfied at that step.

This differs from the previous flow in the verification step. This is in accordance with the guidelines of ECI to prevent any kind of fake deletions.

1. The voter to be deleted is notified via post or email.
2. The list of prospective deletions is made public for calling objections from political parties and general public.
3. If no objections are raised within a certain amount of time, the consent to make the deletions can be assumed.

2.2.3 Properties

- **Non Repudiation:** If a user has made a transaction then he/she cannot deny it.
- **Anonymity:** It should not be feasible to distinguish an individual from at least k others (for large enough k) from their aggregate data.
- **Permissioned:** A person can access some data iff he belongs to ACL of that data.
- **Verifiability:** An individual should be able to view all transactions in which they are one of the participants.
- **Tamper Evident:** Any attempt to tamper with data cannot go unnoticed.
- **Purpose Limitation:** The data requested is used only for the purpose under which it was requested for.

Design

3.1 Immutable append only database

We have used an append only data base to store the electoral rolls, i.e., data can only be appended to it, never deleted. The data structure used is a doubly linked list with hash pointers for each voter. In simple words, instead of assigning a single row to a voter, we keep a hashed chain of blocks among which there is only active block (head of the chain). This active block contains the recent most data about the elector and rest of the chain constitutes the history of that voter.

3.1.1 Hash pointers

Hash pointer is a pointer to the place where information is stored. We also store a cryptographic hash of the information. Hence, a hash pointer differs from a regulator pointer in the fact that it not only helps us retrieve the information but also lets us verify that the information hasn't changed.

Tamper Evidence

Tamper evidence means if someone tries to tamper with a previous/active block of voter's data, it can be detected. We achieve this using hash pointers. For example, consider the data of a voter as represented in our database,i.e, linked list with hash pointers. Let an adversary tamper the data in a middle block (shown by lightning), then hash of the block won't match with hash of this updated data since the hash function is, practically, collision free.

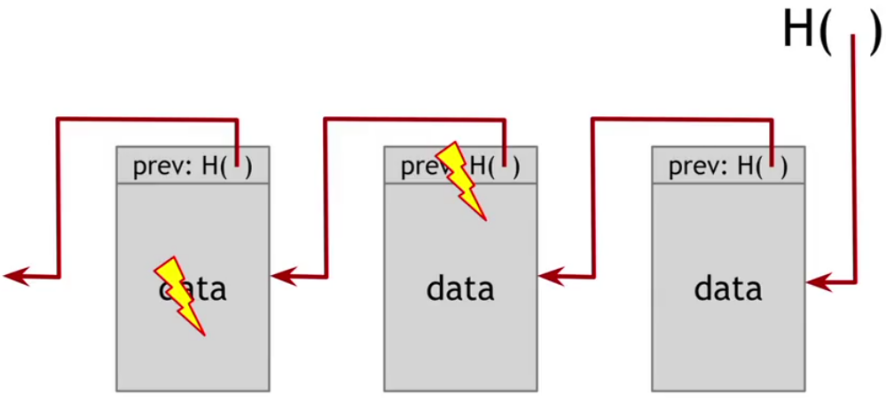

Figure 3.1: Tamper trial with hash chain

Img Source: Link

Thus, we could detect that the block was tampered with. If adversary changed the hash of the block also, so as to match the hash of updated data, then this hash won't match with the 'previous hash' field of the following block. Now if the adversary manages to change that too, the next block would become inconsistent with the hash stored in it. This may go on till the current block, whose hash we know and hence we could detect any change in that. In this way, hash pointers provide immutability and tamper evidence.

3.1.2 Structure of Data Block

The structure of a block in the hash chain for a voter is shown below:

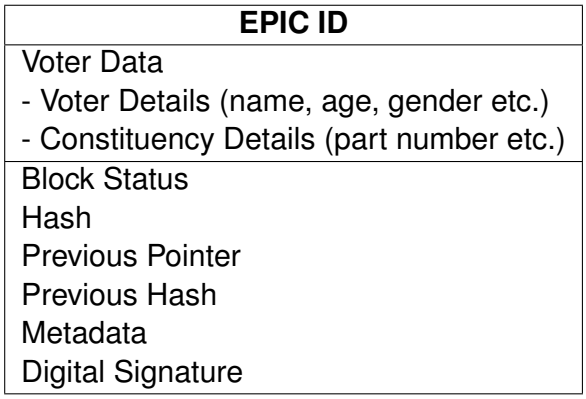

Figure 3.2: Structure of block of data for each voter

Explanation:

- **Voter Data:** It contains data of the voter.
- **Block Status:** It refers to status of block and can be either 0: inactive, 1: active, or 2: deleted. This is needed for the 'append-only' nature of the database.
- **Hash Pointer to previous block:** Previous Hash field and Previous Pointer together constitute the hash pointer to the previous block. This provides tamper evidence of data.
- **Hash:** It stores the hash of Voter Data and Previous Hash Field.
- **Metadata:**
  - It keeps information about who performed the action leading to creation of the block, timestamp of action and necessary proofs for proving the validity of action.
  - Metadata itself is also immutable.
  - It is not accessible publicly and requires administrative privileges.
- **Digital Signature:** We have implemented digital signatures using public key cryptography to achieve non repudiation, message authentication and data integrity. Each block of data contains the digital signature of the writer which makes it independently verifiable at any later instant.

3.1.3 Queries allowed on database

- **Insert:** This corresponds to adding the genesis block of the chain for a given EPIC id. Block status is set to 1. As this is the first block, previous pointer is set to *null*. Previous Hash is set to *zero*. The Hash and Metadata fields are updated accordingly.
- **Update:** Since our database is immutable, updating the existing block of data is not an option. An update is thus performed by appending another block, containing the updated data. This is made the head of the chain and linked to the previous block. The Block Status of previous field is updated to mark it inactive.
- **Delete:** Again the immutability of database leads to inability of deleting a block. The deletion thus corresponds to addition of another block with a block status of 2. It is made to point to the previous block which is marked inactive. Hash and Metadata are set accordingly.
- **Search:** Any search on database would lead to retrieval of data in the active block of the chain linked with the given EPIC id. Search can be

3.2 Immutable Digital Ledger

- Any user can send a request for updation, deletion, objection etc from the portal. These requests are stored in an untamperable (append only) digital ledger to ensure Non- Repudiation and Verifiability.
- The updates to the database are made by Authorised officers with the respective attribute based access. These updates are also stored in the immutable database along with the time stamp and the digital signature of the performer.
- This ensures that no user (even someone with higher levels of access) tampers with the database. All the updates made by AO are also cross checked to be requested by a user along with valid documents,such that no fraudulent changes are made in the electoral roll.

3.3 Merkle Trees

Merkle tree or hash tree is a binary tree in which each leaf represents the hashed value of the record associated with it. The non leaf nodes contain the hash of the concatenation of hashes of the two children. This re-hashing of the concatenation of the child nodes is performed until there is a single hash, the tree root hash.

Signed Root Hash

The tree root hash is very crucial as it acts as a snapshot of all the records in the log. The trusted authority, thus, signs it using its private key so that the client can verify it (using the public key of trusted authority) before using it.

Terminology

- **Record** Data of a voter at some instant. Its hash corresponds to a "leaf" in the Merkle tree.
- **Block** Blocks encapsulate the records along with metadata like whether the record associated with it is active or not. Refer to the block structure shown in Fig 3.2.
- **Verifiable Log** It is the hash tree constructed from the hashed records. The changes to the database are represented by appending new record entries in the log, providing a complete audit trail of the action.

3.3.1 Queries

Merkle trees are useful because they separate the validation of data from the data itself. Moreover, they reduce the query execution time to logarithmic from linear, thus they allow efficient and secure verification of the contents of large data structures. We are using them for two type of queries: data verification and consistency verification.

- **Data Verification** This refers to checking the membership of a voter in the database, based on EPIC Id and other fields. If the elector exists in database, an audit trail or inclusion proof is provided to the query initiator, using which he can generate the root hash and verify the response obtained. Normal search takes O(1) time and providing a proof takes O(log(n)) time.

Inclusion Proof Algorithm

Input : Merkle tree, merkle_node
Output: Inclusion Proof P

```
1  P ←empty set // store the nodes for audit trail
2  if merkle_node is root of Merkle Tree then
3  |  P ←P ∪{merkle_node}
4  |  return P
5  else if merkle_node is left child of its parent then
6  |  P ←P ∪{right sibling of merkle_node}
7  else
8  |  P ←P ∪{left sibling of merkle_node}
9  end
10 parent ←parent of merkle_node
11 P ←P ∪Inclusion_Proof(Merkle Tree, parent)
12 return P
```

Algorithm 1: Inclusion_Proof(Merkle Tree, merkle_node)

- **Consistency Verification** A consistency proof verifies the consistency between a log at two points in time. Since our database/log is append only, it suffices to prove that the log at a given time instant contains the previous version as a prefix. Using the proof, the query initiator can construct the root hash at the previous timestamp and verify it against the stored copy of hashes at different times.
  Remark: This is done on the basis of number of leaves in the earlier version.Also, the auditors have to keep the history of tree root hashes at different instants.

Consistency Proof Algorithm

```
Input  : Merkle tree, timestamp
Output: Consistency Proof P
old_root ←Merkle_Tree.root_history[timestamp]
num_leaves ←| old_root.leaves | // number of records at timestamp
P ←empty set // store the nodes for audit trail
level = log2(num_leaves)
node ←leftmost node at level in tree
k = | node.leaves| // number of leaves in subtree rooted at node
P ←P ∪{node}
if k == num_leaves then
    return P
else
    sibling ←right sibling of node // sibling exists as node lies at
        leftmost position
    done ←False
    while not done do
        sibling_leaves ←|sibling.leaves|
        if num_leaves - k == sibling_leaves then
            P ←P ∪{sibling}
            done ←True
        else if num_leaves - k > sibling_leaves then
            P ←P ∪{sibling}
            sibling ←right sibling of sibling
            k ←k + sibling_leaves
        else
            sibling ←left child of sibling
        end
    end
end
return P
```

Algorithm 2: Consistency_Proof(Merkle Tree, timestamp)

3.4 Access Control

- Actions involving changes to database are performed by authorised entities only and information is provided on a *need-to-know* basis. Anyone could query the existence of a voter in an assembly constituency.
- The authorised officials would be authorised to make changes to the database based on the requests received. This would require two-level authorisation,i.e, digital signatures along with basic authentication.
- The regulators and authorised officials have admin privileges for only those electors who are assigned to them, i.e., fall under their area of authorisation. Thus we have a finer level of control than Role Based ACLs.
- This Attribute Based Access Control is scalable to any number of attributes.

3.5 Digital Signatures

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. We have used RSA (Rivest-Shamir-Adleman) public key algorithm to generate a pair mathematically linked keys: the public and the private key for each Authorised Official. The public key is assumed to be publicly available and the private key is assumed to be present in the system/ device of Authorised Official. Digital Signatures guarantee that the contents of a block are not altered, the authorised official is not impersonated and the database is not tampered with.

3.5.1 Working

- We have seen above that every interaction of an Authorised Official with database (insert /delete /update) corresponds to adding a block to the hash chain of the concerned voter.
- The Authorised Official, while entering a block of data to the electoral roll, encrypts the hash of this data using his private key. This encrypted hash works as a digital signature and is stored in the concerned block itself.
- When a search on database is made, the data in the active block of chain of this voter is retrieved. To verify this data, the retriever first decrypts the signature present in the block using writer's public key. Then this value is matched with the hash of received data.
- If they match, both writer's identity and data integrity are guaranteed. If they don't match, it indicates something wrong with the data.

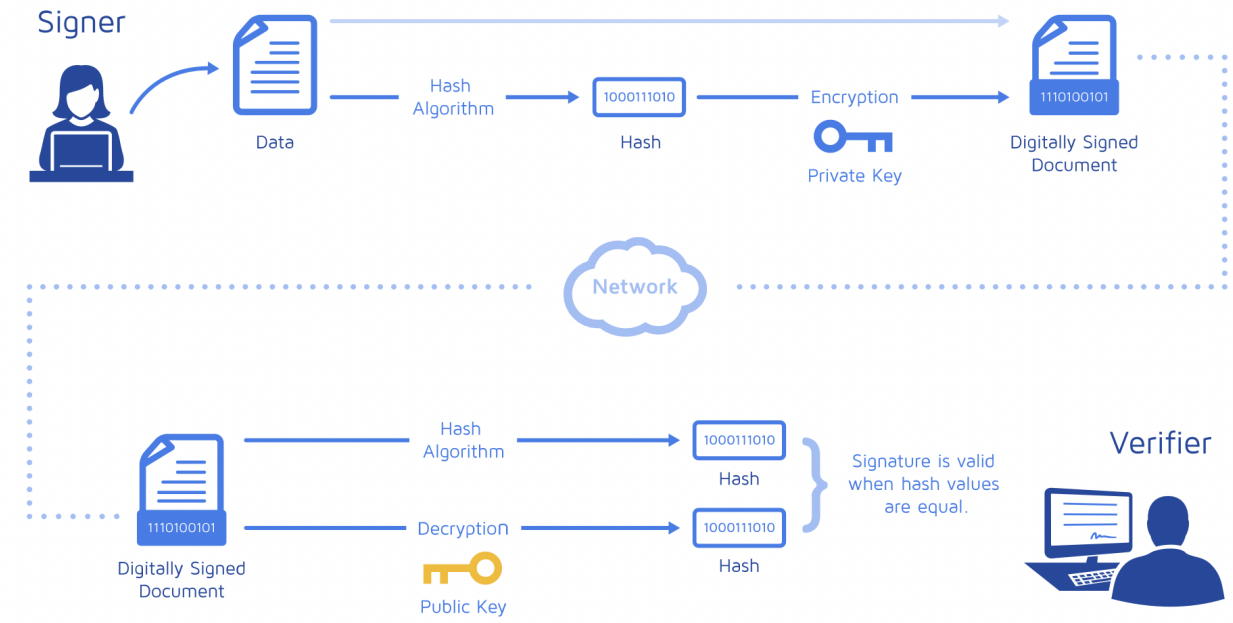

Figure 3.2: Working of digital signatures

Source: Link

Implementation Details

4.1 Libraries and Frameworks Used

- **Framework:** Flask
  Flask is a micro web framework written in Python. It is a microframework as it does not require particular tools or libraries.
  Need to Install: flask, flask-restful,flask-jwt-extended,flask-login,flask-bcrypt

- **Data Platform:** MongoDB Atlas
  MongoDB Atlas is a fully-managed cloud database that handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider of your choice (AWS , Azure, and GCP).
  Need to create an account on MongoDB Atlas

- **MongoEngine**
  MongoEngine is a Python library that acts as an Object Document Mapper with MongoDB, a NOSQL database. It is similar to SQLAlchemy, which is the Object Relation Mapper (ORM) for SQL based databases. *Need to install:* flask-mongoengine

- **Algorithm for Digital Signatures:** RSA
  Our implementation uses RSA to generate the pair of public and private key pair.
  Need to Install pycryptodome

4.2 Working

1. The index page shows an option to start a new session. This would initialise a merkle tree and can be used to populate the database in the beginning.
   api: `http://127.0.0.1:5000/api/index`

2. Then there is an option to signup or login. If the user uses his EPIC Id while signing up, then his account would be linked to his record in the Electoral Roll.
   api: `http://127.0.0.1:5000/api/auth/signup`

3. The home page of a user contains his level of access details and the actions available to him. The Insert/Update/Delete actions are available to Authorised Officials only. Note that an AO can't add/ modify data of a voter who doesn't belong to any of the constituencies under him.
   api: `http://127.0.0.1:5000/api/auth/login`

4. The membership of a voter can be checked using merkle tree implementation also. This would only return the response to query along with a proof of correctness. It won't reveal any other information about the searched voter. Our query provides a generic query, in the sense that the user can search a voter based on more than one fields (as per his choice).
   api: `http://127.0.0.1:5000/api/merkle/member`

5. The consistency of database can also be checked by an Authorised Official. Since we have an append only database, consistency w.r.t. any previous timestamp would translate to the condition that the data at that timestamp is present as a prefix to the current database (in the same order).
   api: `http://127.0.0.1:5000/api/merkle/consistent`

4.3 User Interface

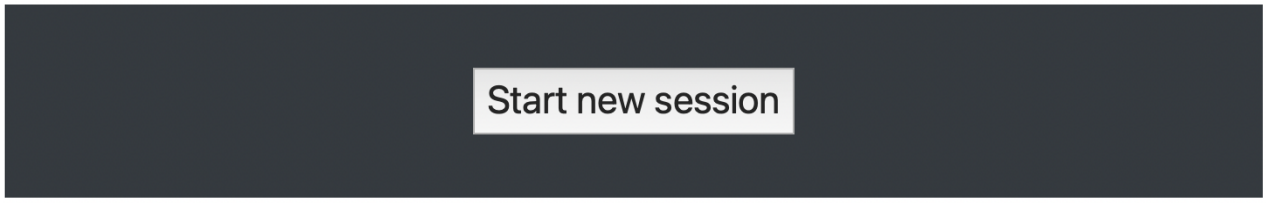

Figure 4.1: Starting a new session

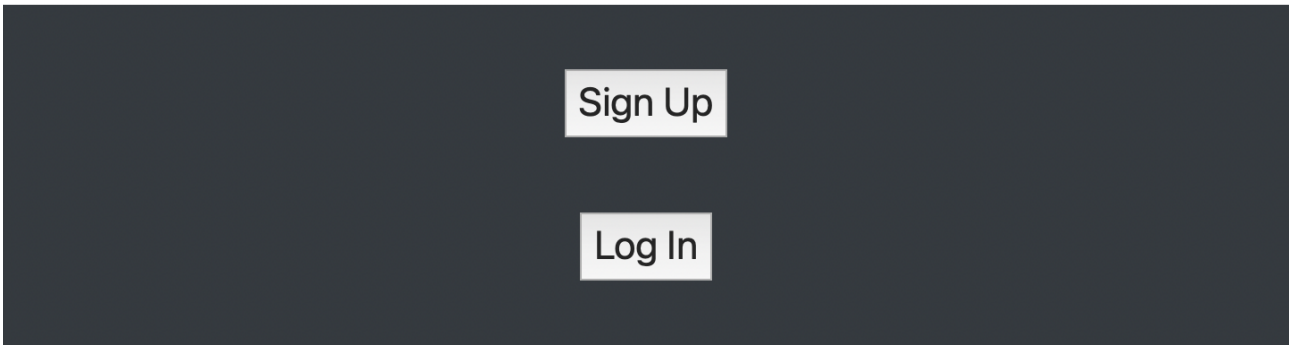

Figure 4.2: Signup or Login to portal

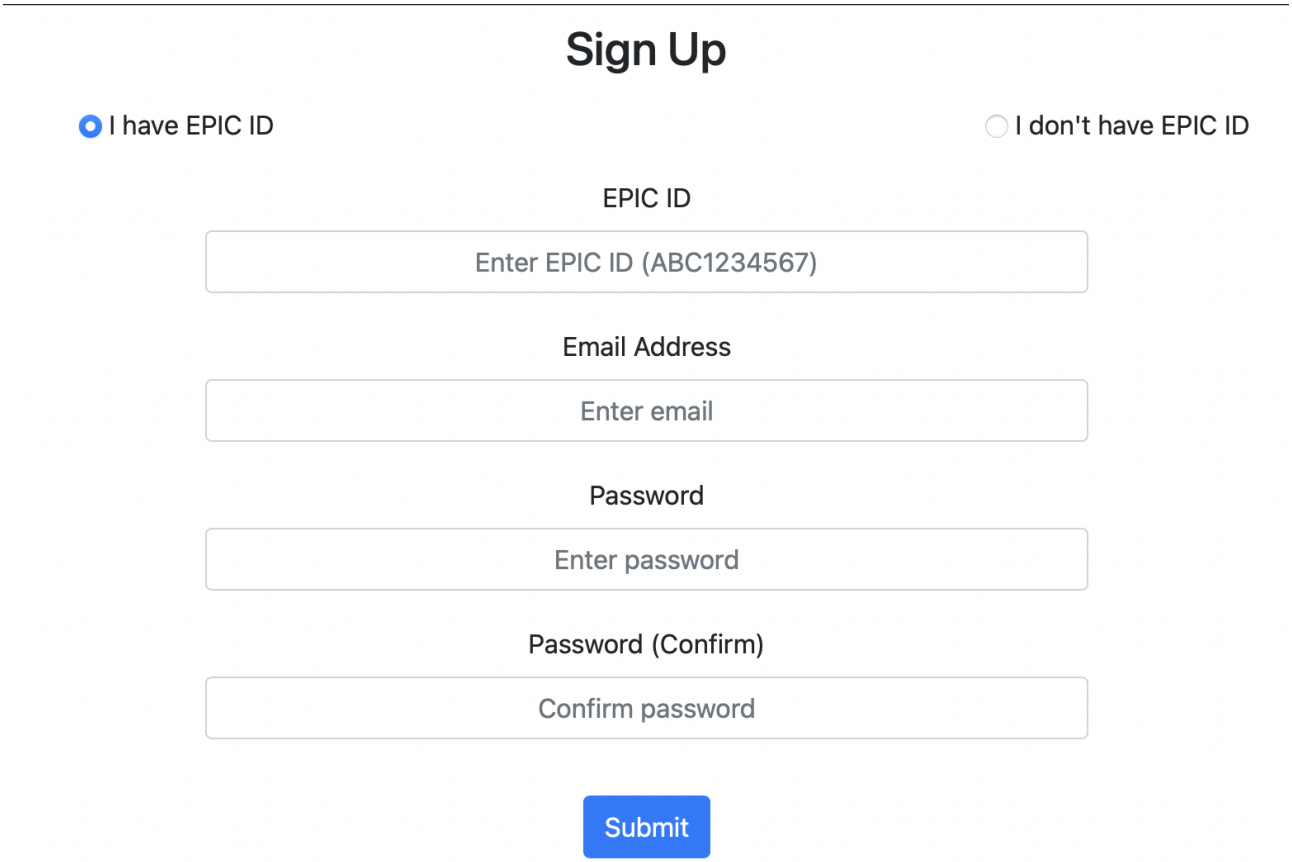

Figure 4.3: Two ways for Signing Up

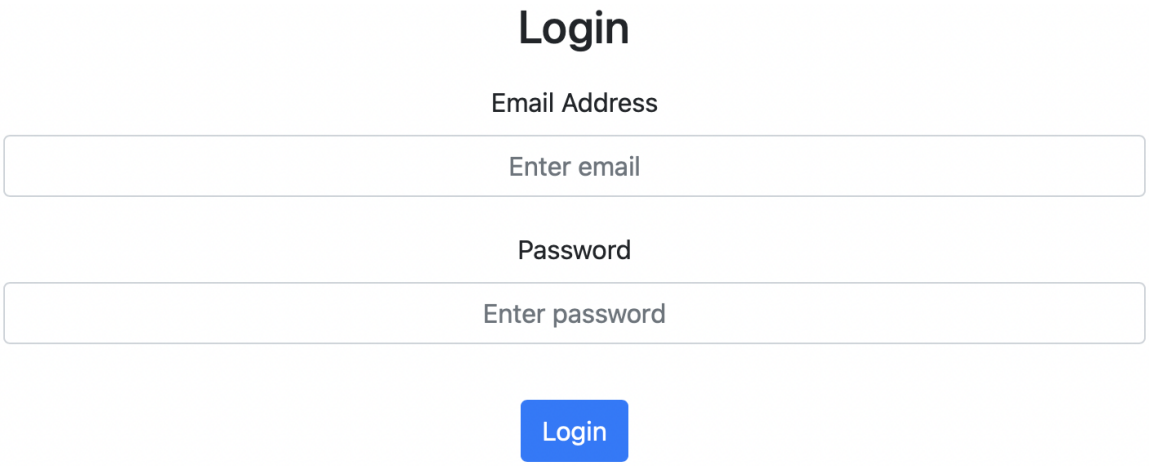

Figure 4.4: Login

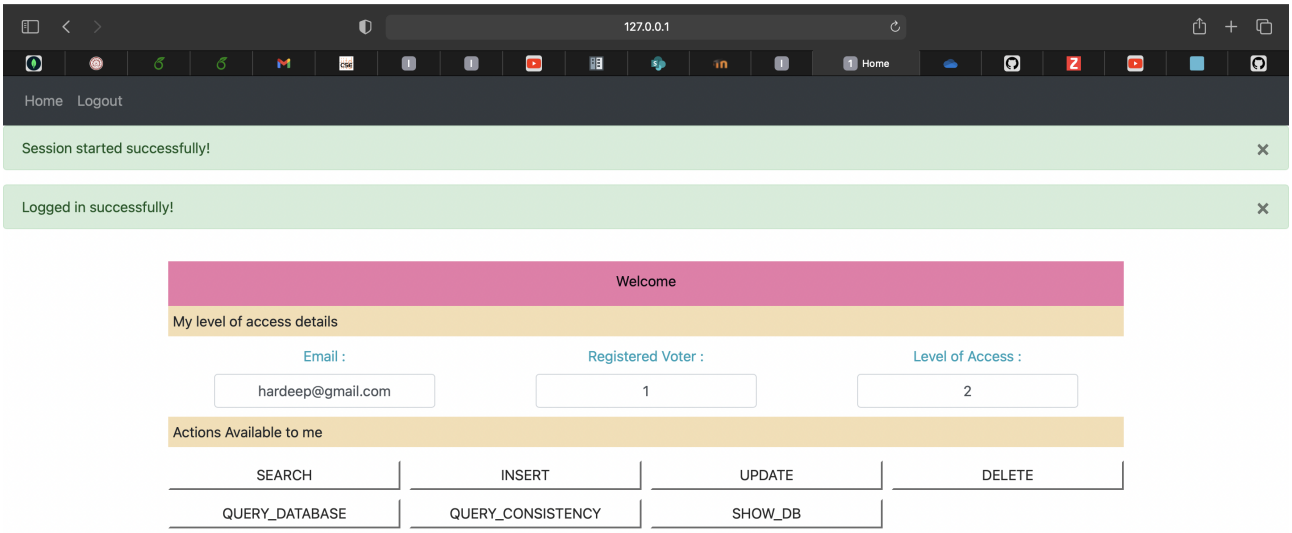

Figure 4.5: Actions available to an Authorised Officer

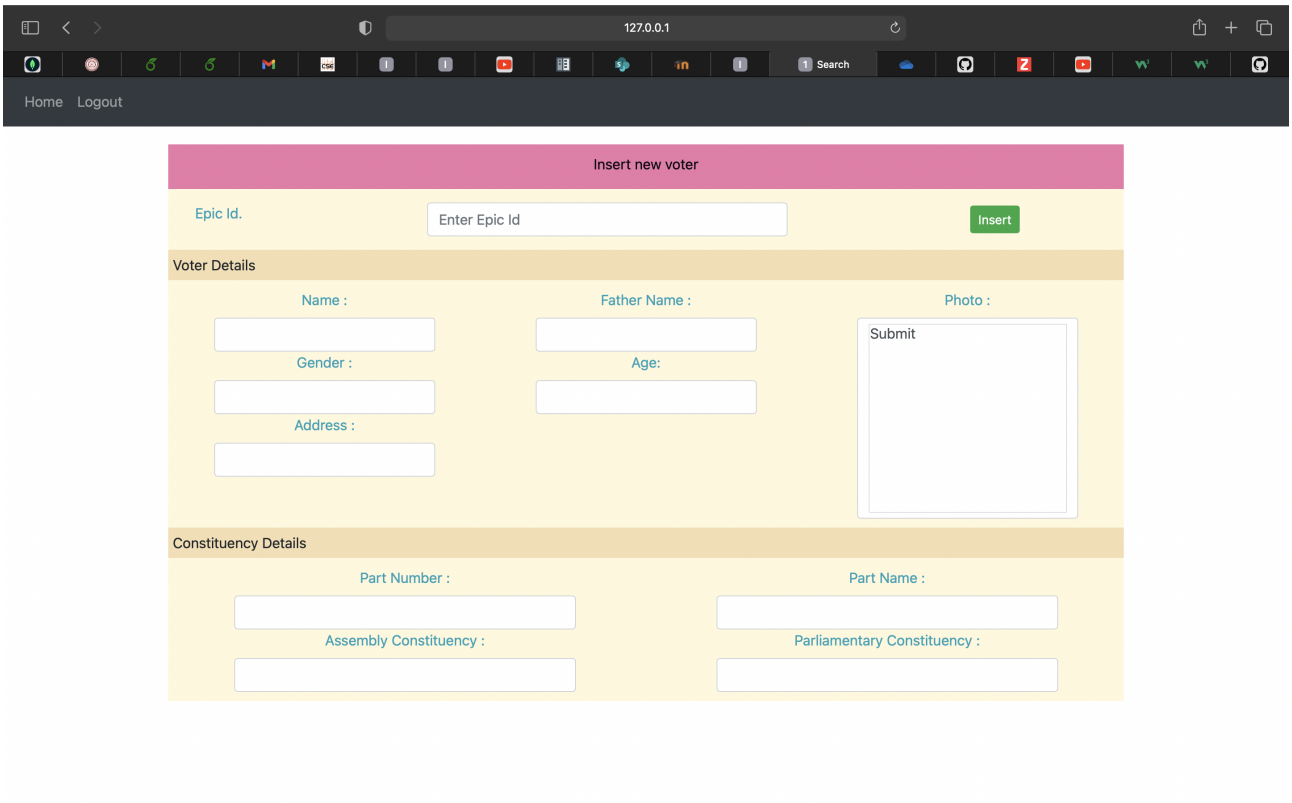

Figure 4.6: Insert voter

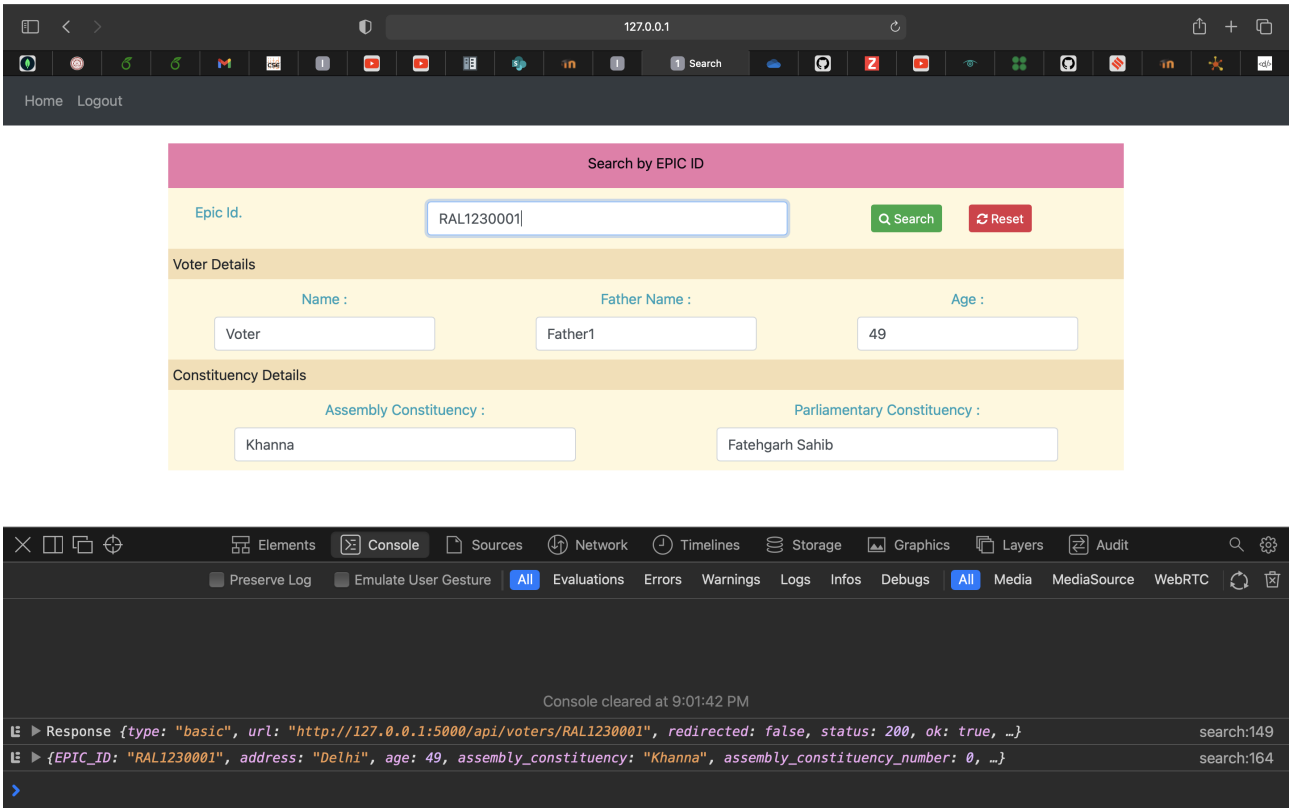

Figure 4.7: Search: Voter exists

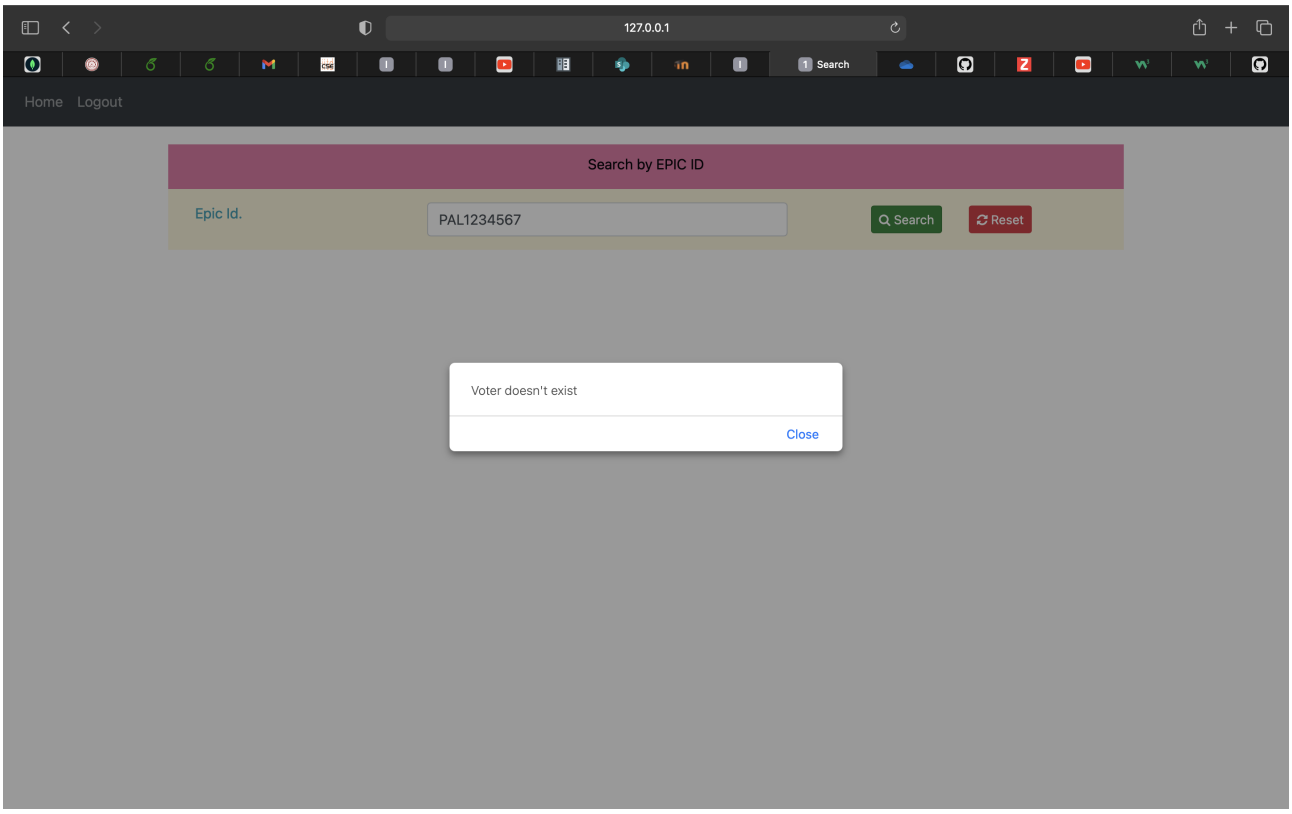

Figure 4.8: Search: Voter doesn't exist

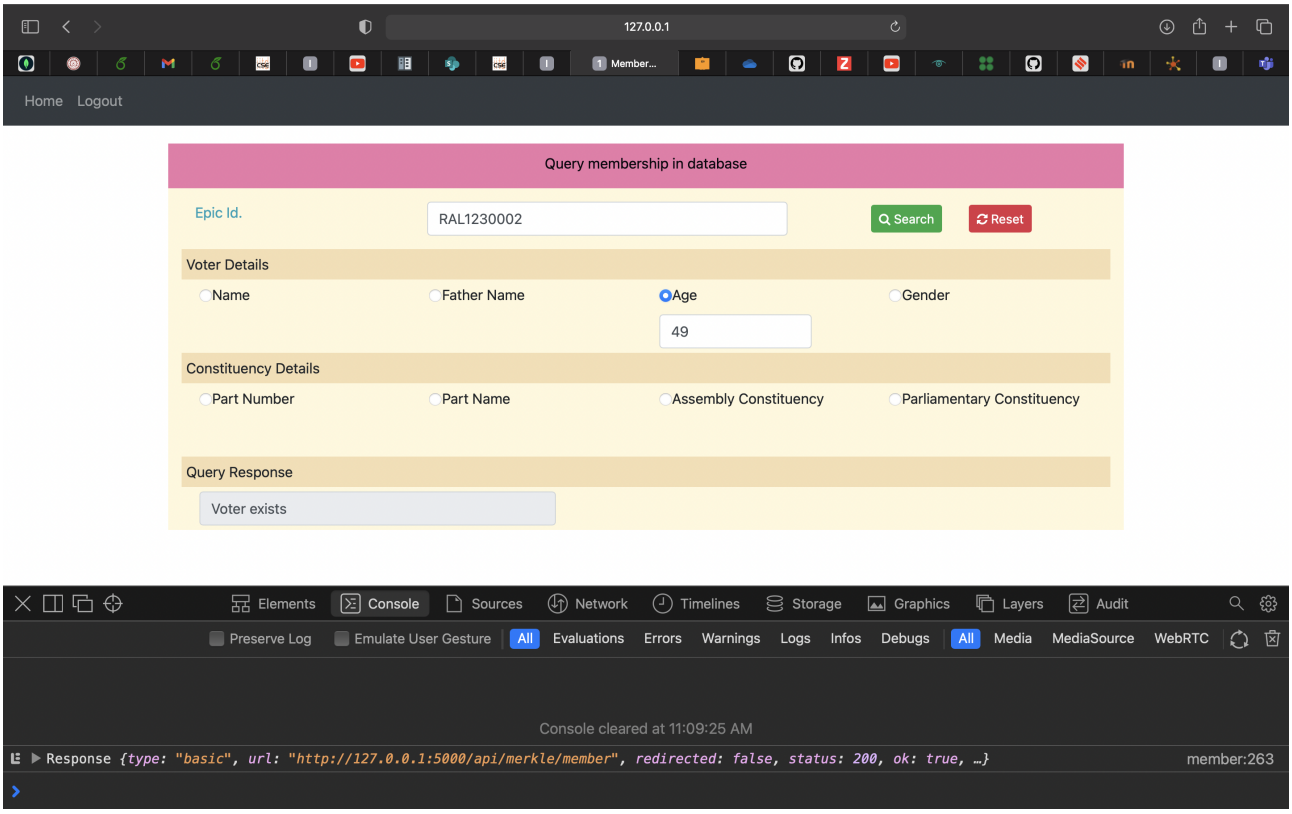

Figure 4.9: Merkle Tree: Membership Query

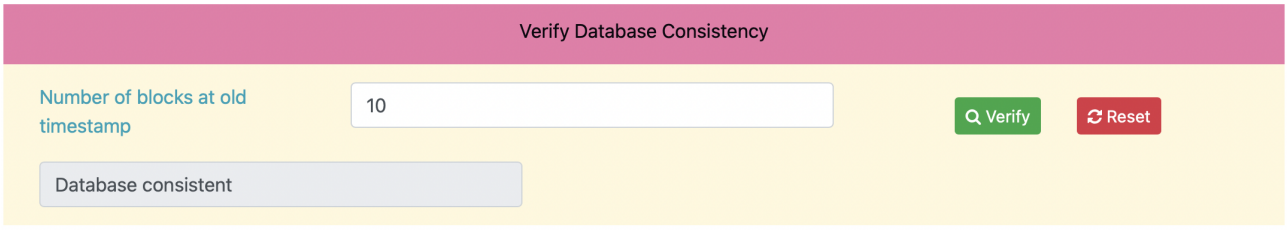

Figure 4.10: Merkle Tree: Consistency Proof

4.4 Further work to be done:

- The integrity of current Electoral Rolls come from the public nature of data. Digital rolls should not be made public in digital form due to the ease of profiling using ML algorithms.
- We can work towards a public database which contains data in encrypted form.
- Whenever a user makes a request for some action on database, he is given a signed receipt (hash) using which he can verify the status of his request.
- This would eliminate the need of a regulator and access control. Note that newer ways of inspection need to be thought upon in this encrypted database.
- Also, the Elector's Registration Rules should be modified for digital nature of data, with the underlying principles intact.

4.5 References

- Solove, Daniel J., The Digital Person: Technology and Privacy in the Information Age (October 1, 2004). [1]
  Available at SSRN:`https://ssrn.com/abstract=2899131`
- Prashant Agrawal, Subodh Sharma, Subhashis Banerjee.Blockchain vs Public Bulletin Board for Integrity of Elections and Electoral Rolls. [2]
- Heather J., Lundin D. (2009) The Append-Only Web Bulletin Board. In: Degano P., Guttman J., Martinelli F. (eds) Formal Aspects in Security and Trust. FAST 2008. Lecture Notes in Computer Science, vol 5491. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01465-$9_1$6.[3]
- Registration of Electors Rules, 1960. [4]
- Hash Pointers and Verifiable Data Structures. [5]
- Digital Signatures