secure & trustful mechanism for storing Aadhaar- details and Aadhaar based user-identity validation.

Soln->

Aadhaar data is stored in a distributed manner spanning across hundreds of computers. The data obtained from the user is encrypted.Using Public and Private key mechanism User's Data is signed and the encrypted version is stored on the blockchain. Signature to the block mapping is also stored to easy verification. This encrypted data is stored in a block which is mirrored across all nodes. This encrypted data can be shared with third parties for verification and authentication. This prevents the misuse of Aadhaar Card Number which is linked to a number of services such as Bank accounts and so on.

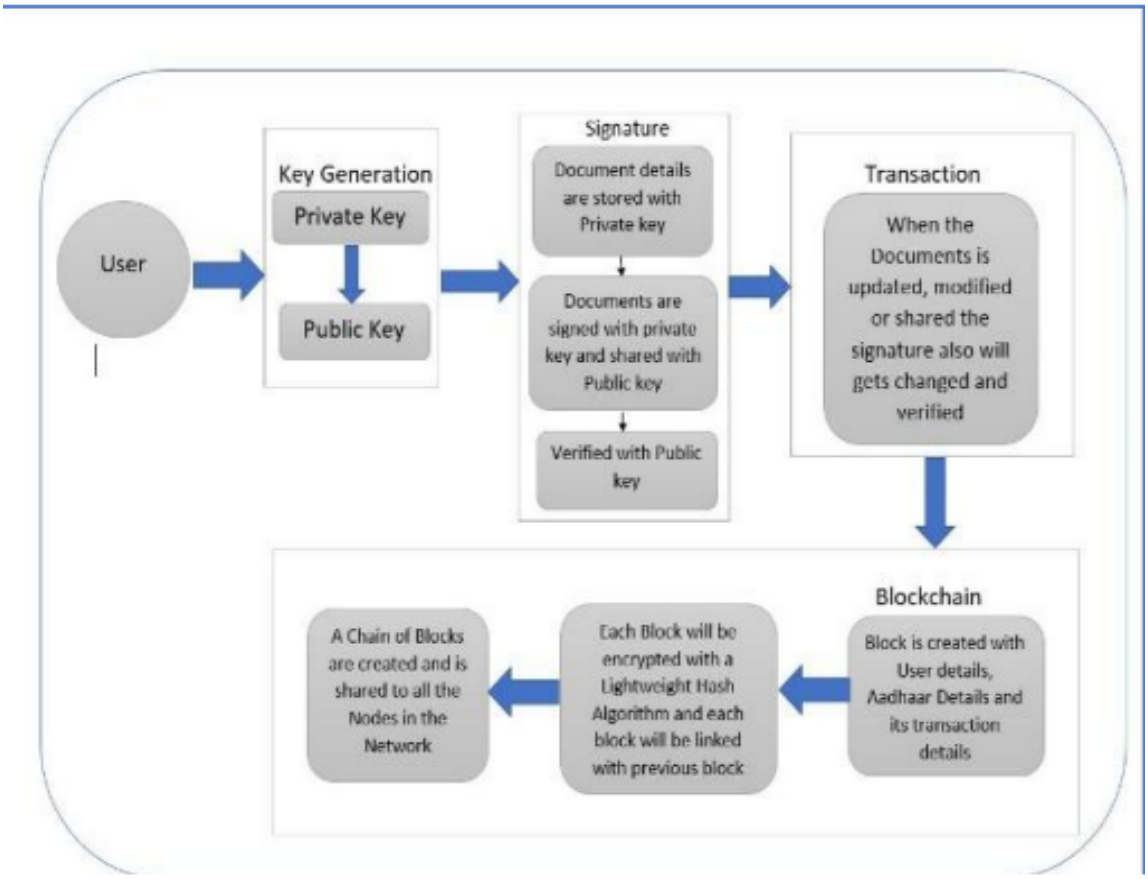
The Ethereum Smart Contracts is a computer protocol which is used to verify a contract or enforce a negotiation with a contract. The Smart Contracts work only when a certain condition is met. This technology can be used in Aadhaar system to ensure restricted access to the user's personal information. The third parties can access only part of the user data if authenticated. This also prevents the unauthorized access of user data and ensures privacy by using the Ethereum Smart Contract.

In order to achieve this we have to create a genesis block in which user identification is uploaded with private key for which a public key is generated. To add Aadhaar details like Address, Iris scan, Finger Print, etc,. and upload with a private key, sign (in general signature part use Elliptic Curve Algorithm) and share in the network which is should be verified

- User Registration is done
- Using Public and Private key mechanism User's Data is signed and the encrypted version is stored on the blockchain
- This ensures data is anonymous

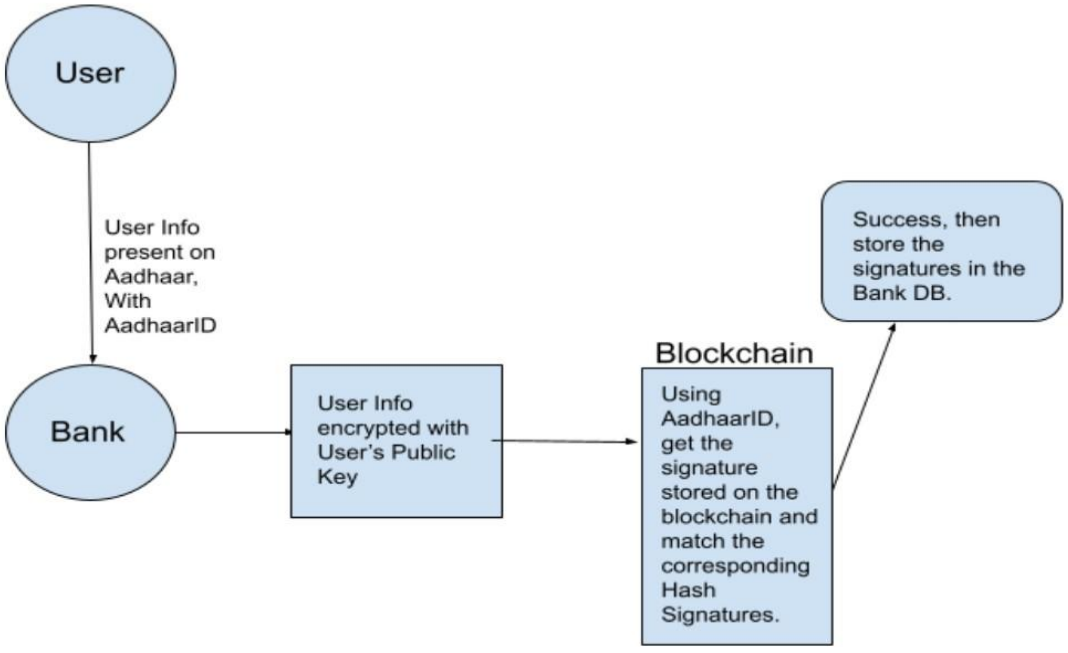
SCHEMA

(i) Registration



(ii) Verification

User-Verification System :



Pseudo code

```
Struct UserDetails{  // Structure for storing user details.
Bytes Aaadhar Id,
Bytes Name,
Bytes DoB,
Bytes fingerprint,
Bytes Address,
Bytes Iris scan,
..
...
};

mapping(bytes=>UserDetails)m;// Mapping for signature to user
//structure.
```

Steps to follow :

1. User gives the user info, as per his/her Aadhaar to 3rd party.
2. The 3rd party can encrypt the same using the user's public key.
3. Ask the blockchain to return the User struct for the particular AadhaarID.
4. The smart contract returns the User hash signatures and they are matched with the current available signatures.
5. If the match is a success, 3rd party can remove the confidential data from their DB and just keep the signatures instead.