Unified KYC

Introduction:

KYC (know your customer) is a process that establishes trust between the organization and the customer. Some organizations make it compulsory for their customers to submit their Government approved ID as proof of identity for the KYC. Hence, this is an essential process that is often done to establish a sense of trust between the organization and the customer. The data stored during the KYC process is often used by the organization for providing services to the customer.

Methods/background:

KYC is usually done by organizations according to their requirements. So the type and number of documents vary as per the organization. However, the general type of document that is required by most of the organizations includes ID and address proof. These documents along with some other information like name, date of birth, parent's details, etc complete the KYC process. Most of the KYC processes require verification. Some organizations use physical verification which is done manually. This verification is done with the help of humanitarian aid. This method consumes time and financial resources. With the advent of AI and machine learning, KYC can be done without the help of human intervention.

The documents submitted at the time of KYC are stored in the form of digital or paper-based documents. The majority of the organizations store these documents digitally. These documents are stored in a centralized repository or database. This database is generally in control of the database administrator.

Problems:

The documents that are stored digitally contain personal information about the customers. The customer is not in control of their private information. Hence the customer has to blindly trust the organization. In case of an attack, the information leak can create problems for the organization and the customers. The data is often leaked to a 3rd party for financial profits.

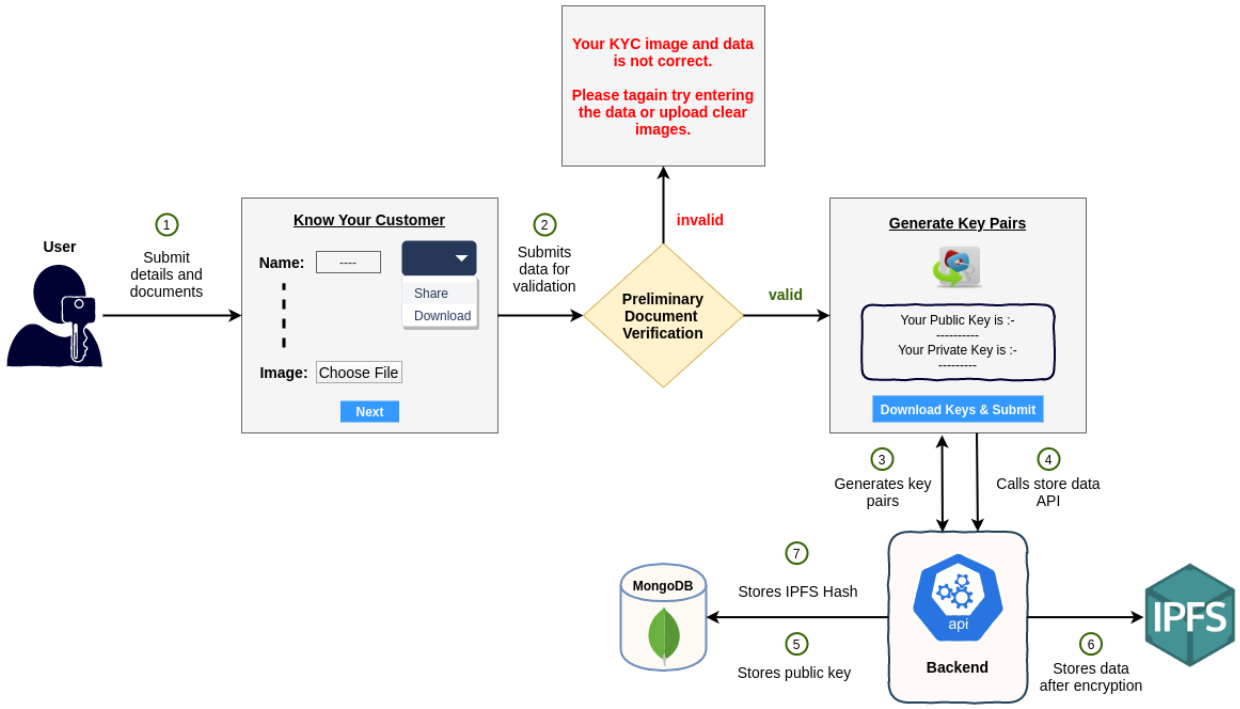
In many cases, the customer has to seek services from multiple organizations. Thus, the customer has to perform the KYC process many times. This lack of coordination between the organizations may cause customers much inconvenience to the customer.

Proposed solution:

We propose a solution to the above problems using a self-sovereign, verifiable, and independent system. The solution leverages the transparency and trust provided by decentralized distributed ledger technology. The proposed solution verifies the KYC documents using AI and machine learning. The authenticity of the shared KYC documents can be verified on the blockchain. The KYC documents are protected by end to end encryption. The users of the system enjoy full control over the distribution and management of the KYC documents.

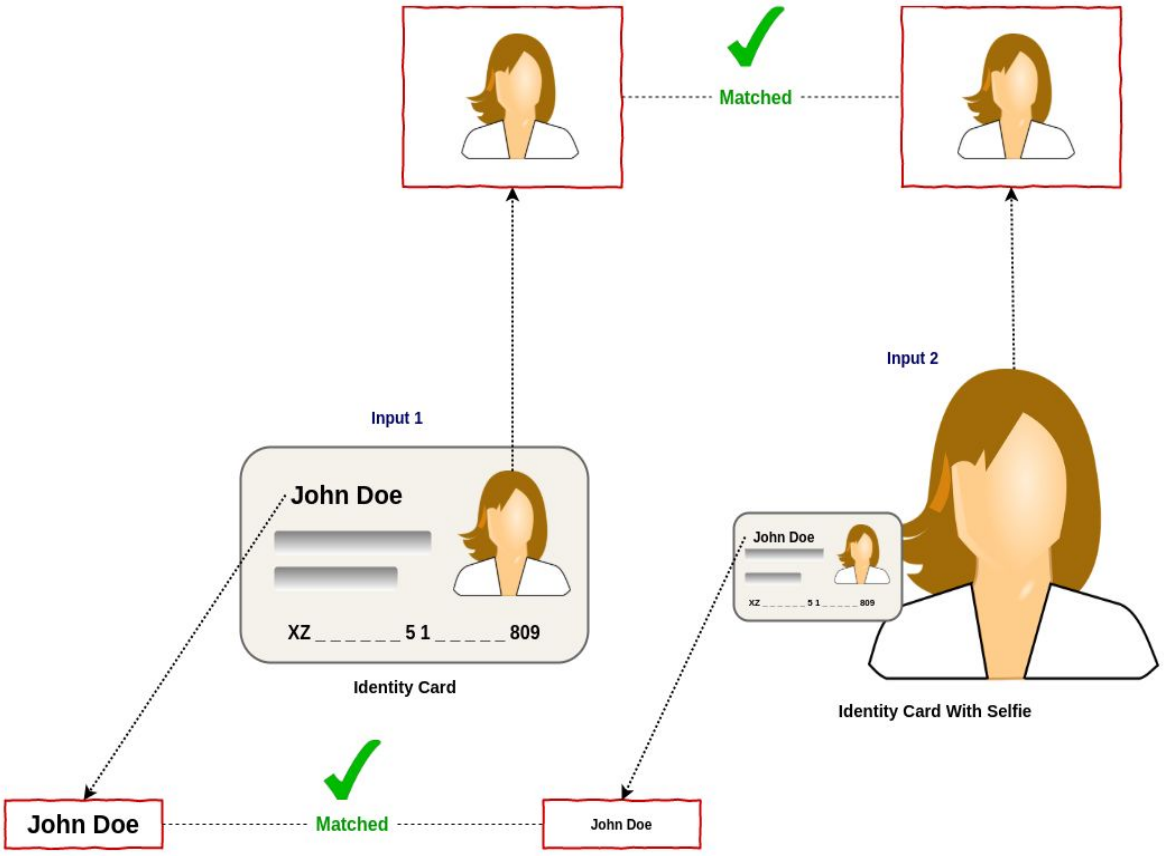
The solution is described in detail below:-

Step 1: (Submit KYC)



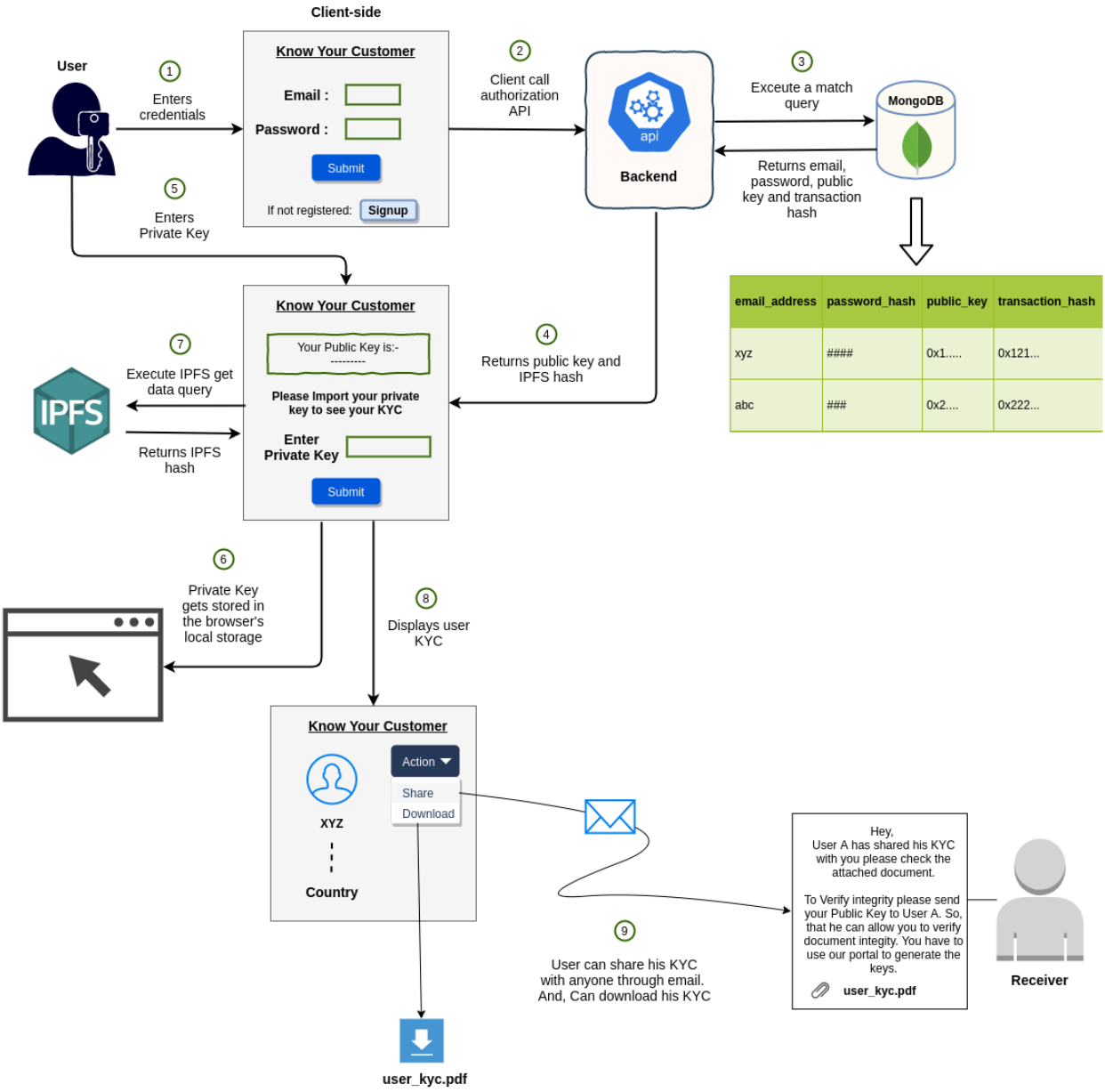
1. Users will fill the KYC form. Which will include all necessary details like name, address, city, etc.
   The user will also upload an identity document image and a selfie with the document.
2. Preliminary KYC verification **(discussed in Preliminary KYC Verification section).**
3. After verification, the system will generate a key pair for the user.
4. The user will then click the submit button. On the click of the submit button, the system will call the backend API to store the user's public key and KYC checksum with his email.
5. The backend will call a database method to store the user's public key in the database.
6. The backend will store encrypted KYC data on the IPFS. The IPFS will return a multihash, which will be used to fetch the data in the future.
7. After getting the IPFS hash, the system stores the IPFS hash in the Database.

➔ **Preliminary KYC Verification**



1. The user will upload the image of the document (ID image).
2. The user will upload a selfie with the same document (selfie image).
3. All the information from the ID image and selfie image will be extracted. For example name, DOB, and facial descriptors.
4. The facial descriptors extracted from the ID image and the selfie image will be compared using euclidean distance.
5. The personal information extracted from the ID image and the selfie image will be matched using OCR (optical character recognition).
6. After successful verification of the KYC. Users can generate the key pair and proceed further.

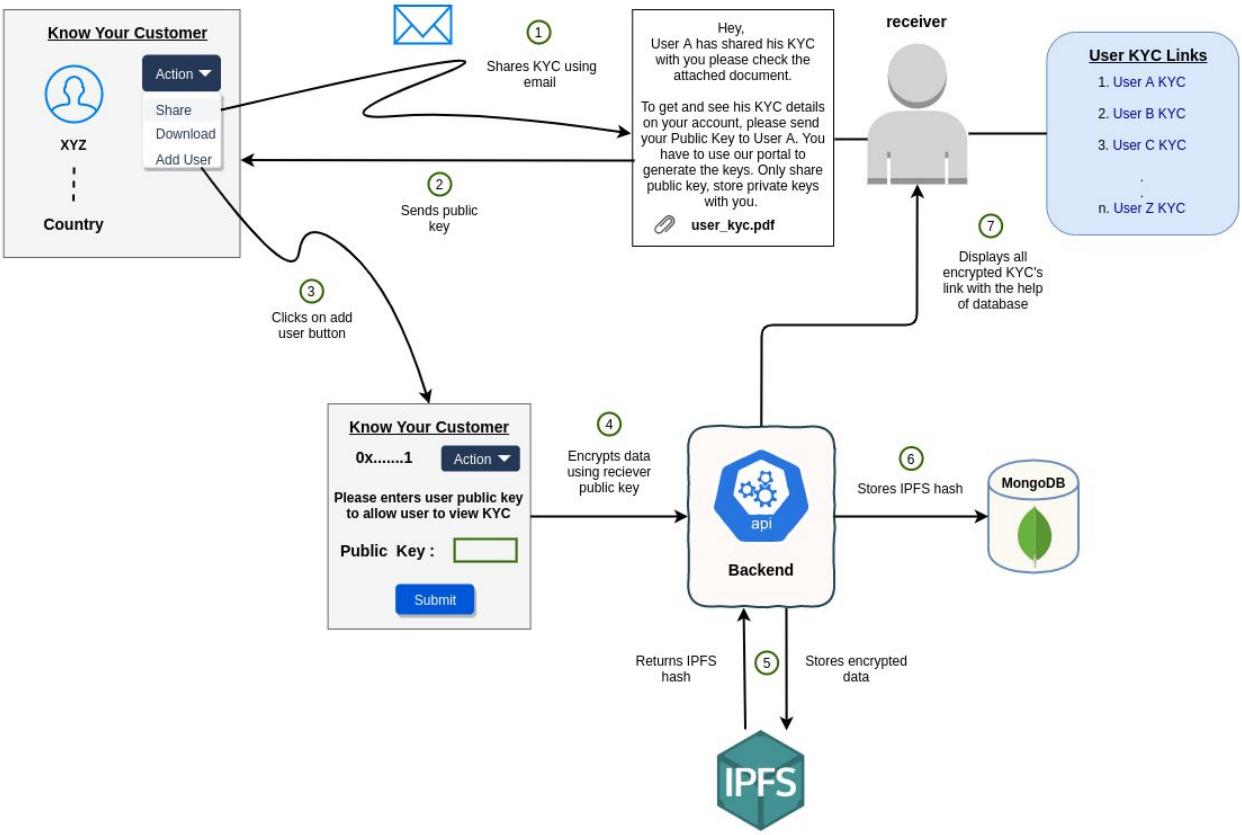
Step 2: (Access Your Own KYC)



1. The user will enter their credentials to log in to the system.
2. The client will call the backend authorization API.
3. The backend will validate the user-provided credential with the credentials stored in the database with the help of a database query. If credentials will be valid, the backend will fetch the user's public key and IPFS hash from the database.
4. If valid credentials, then the backend will display the user's public key, with a dialogue box to import the private key on the client. Private key will always be in control of the user. The private key will be used to decrypt encrypted KYC data. If the credentials will be invalid, the backend will display a message that the provided credentials are invalid.

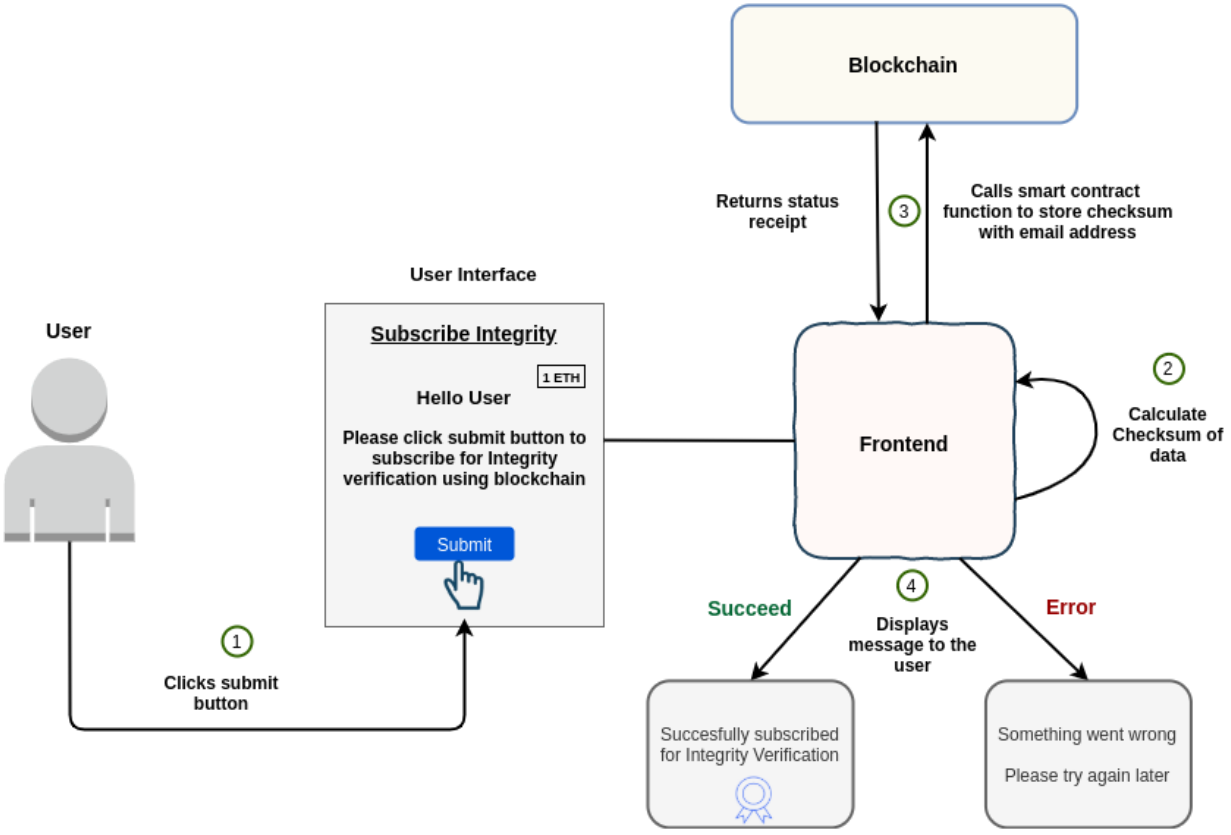
5. Then, the user will enter his private key.
6. The private key will be stored on the client-side.
7. The encrypted data will be fetched from IPFS using IPFS Hash.
8. The data fetched from the IPFS will be decrypted using the private key stored in the browser storage as described in step 6. Now the KYC will be visible to the user.
9. Users can download or share their KYC. Process for that is described below.

Step 3: (Share KYC)



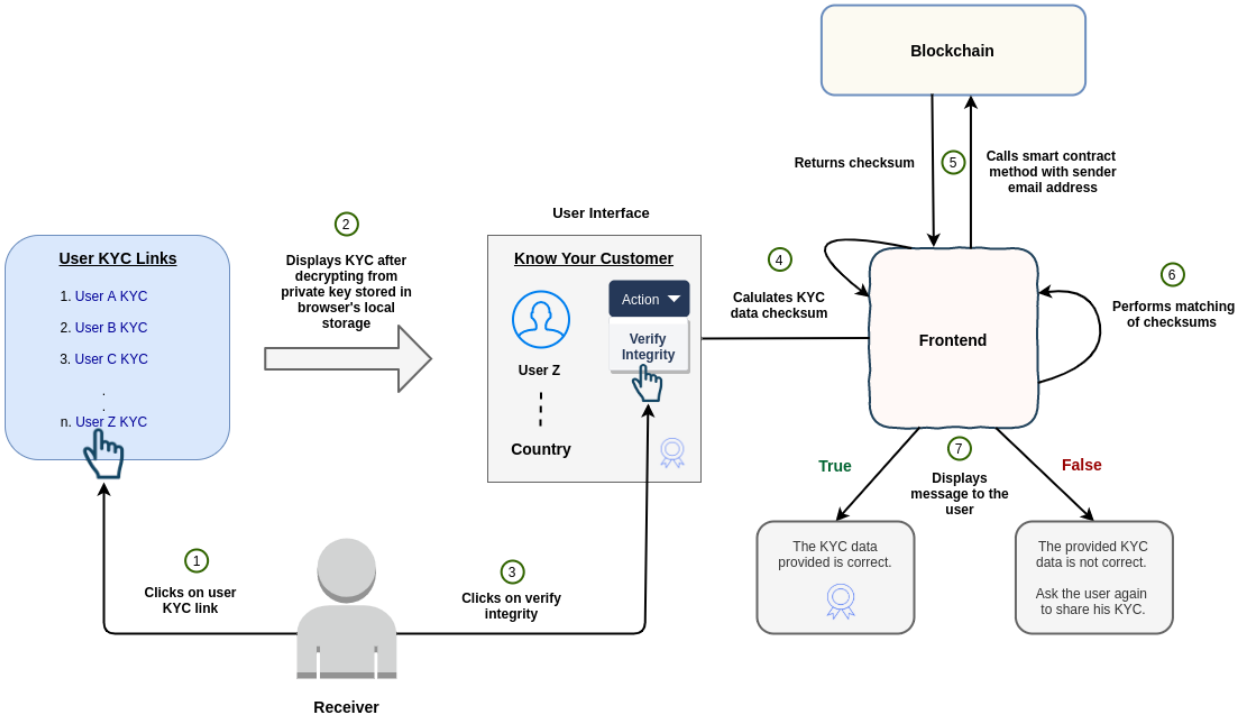
1. The user will share his KYC through email. In that email, the receiver will be asked to share his public key by generating a key pair from our application.
2. After generating the key pair, the receiver will share his public key to the user. (Only for the first time users)
3. After receiving the public key, the user will click the add public key button on the frontend. The user will then fill the user's public key on the frontend.
4. The user's KYC data will get encrypted using the receiver's public key.
5. Encrypted KYC data will get stored in the IPFS and, IPFS will return a multihash called IPFS hash.
6. IPFS hash will then get stored in the database with the user's public key.
7. When the user will log in to the system, the backend with the help of the database will show him all associated KYC's. After importing his private key, he will be able to see the KYC because all the KYC's have been encrypted with his public key.
8. The receiver can verify the integrity of the data using blockchain, If the user has activated the Integrity subscription using blockchain.

Step 4: (Subscribe for KYC Integrity)



1. User clicks on the submit button to subscribe for integrity verification using blockchain.
2. System calculates KYC data checksum.
3. System calls a smart contract method to store KYC checksum with the user's email address. So, in the future the KYC checksum can be fetched using the user's email address.
4. If the transaction is successful, then the system will show a message that You have successfully subscribed for integrity verification. Otherwise, system will show error message,

Step 5: (Check KYC Integrity)



1. The receiver will click on a user KYC. For example, In diagram receiver clicks on user z.
2. The system will display the user's KYC after decrypting it using the private key which the receiver will import after logging in to the system.
3. After decryption of the user's KYC, the receiver will click the Verify Integrity button to check the integrity of the KYC.
4. Decrypted KYC data checksum is calculated.
5. The frontend will call a smart contract function to fetch KYC checksum using the sender's email address.
6. KYC checksum fetched from step 4 and step 5 will be compared for equality.
7. In either case, True or False, the system will display a message to the user through the frontend.

Benefits:

1. The KYC details can be shared with anyone using the system with a valid key pair. This increases reusability.
2. KYC documents are end to end encrypted. The private keys are not accessible at the server-side.
3. The integrity of the KYC documents is ensured using the checksum stored in the blockchain.
4. The solution provides a reliable preliminary KYC document verification. So that fake KYC documents will be rejected at an early stage.

References:

1. Decentralized KYC System