A Decentralised KYC Verification Process for Banks

Origin of KYC

Know your customer or KYC originated as a standard to fight against the laundering of illicit money flowing from terrorism, organised crime and drug trafficking. The main process behind KYC is that government and enterprises need to track the customers for illegal and money laundering activities. Moreover, KYC also enables banks to better understand their customers and their financial dealings. This helps them manage their risks and make better decisions.

Need for KYC

Taking in from the origin of KYC, we can state that there are four major sectors in banking where KYC is needed. They are as follows:

Customer Admittance: This sector defines making anonymous accounts as restricted entry into the banking system. In other words, no anonymous accounts are allowed. Preliminary information, such as names, date of birth, addresses and contact numbers, is to be collected to provide banking service.

Customer Identification: In the case of suspicious banking transactions through a customer, customer accounts can be tracked and flagged. Further, they can be sent for processing under the bank head office for review.

Monitoring of Bank Activities: Suspicious and doubtful activities in any account can be zeroed in by the bank after understanding its customer base using KYC.

Risk Management: Now that the bank has all the preliminary information and the activity pattern, it can assess the risk and the likelihood of the customer being involved in illegal transactions.

These requirements make the KYC process an essential entity in the banking and financial world. The traditional KYC process is already in place in some banks, but there are major challenges related to the process, and through this case study, we will assess and tackle these challenges. Let's first list the challenges related to the traditional KYC process.

Problems/Challenges in KYC

The disparity in specifications for KYC

Every bank has its own KYC process set up, and customers need to do the KYC again and again for each bank. Due to the lack of KYC standards, compiling each request is time-consuming.

Adverse impact on customer relationship

It becomes irksome for customers to provide the same information to different banking entities and industries. Banks sometimes even follow up with customers to get more details for KYC.

Escalating costs and time for banks

A recent study concluded that overheads of KYC in a bank increase the onboarding cost for a customer by 18% and the minimum time required to 26 days.

Solution Using Blockchain

The blockchain is an immutable distributed ledger shared with everyone involved in the network. Every participant interacts with the blockchain using a public-private cryptographic key combination. Moreover, immutable record storage is provided, which is extremely hard to tamper with.

Banks can utilise the feature set of blockchain to reduce the difficulties faced by the traditional KYC process. A distributed ledger can be set up between all the banks, where one bank can upload the KYC of a customer and other banks can vote on the legitimacy of the customer details. KYC for the customers will be immutably stored on the blockchain and will be accessible to all the banks in the blockchain.

This case study is divided into three parts to achieve the solution. They are as follows:

Phase 1:

Banks add customers and their data on the network.
Whenever any new data is needed to be appended, the ledger could enable encrypted updates of the data.
Mining will make sure that the data gets confirmed over the blockchain and is distributed to all other banks.

Banks can modify the data of the customers present in the database. In phase 1, any bank can modify the data of the customer. In phase 2, we will add admin and bank functionalities, which will provide the necessary restrictions over the network and data of the customers.
Banks can also view customer data.

Phase 2:

Admin functionalities are provided for the system, where an admin can track the actions such as upload or approval of KYC documents performed by banks.
The admin can block any bank from doing a KYC; the admin can also add new banks or remove untrusted banks from the smart contract.
Whenever a new customer enters into the system, a bank initiates a KYC request for the customer with the additional data provided by the customer to the bank. Any bank can raise the KYC request.
Once a KYC request of a customer is added, any bank can upvote or downvote, stating their stand on the data provided by the customer.
The bank can remove the KYC request of the customer.
Now, the customer struct will also store the number of upvotes and downvotes, and the KYC status of the customer. For the KYC status to be true for any customer, the number of upvotes should be greater than the number of downvotes. Also, if one-third of the total number of banks downvote the customer, then the KYC status is set to false even if the number of upvotes is greater than that of downvotes for that customer.

Customers' KYC status will be stored on the chain depending on the number of upvotes/downvotes.

Banks also report the other banks to make sure that the banks are secure and not tampered with for the KYC process. This identifies whether the bank is corrupted and whether it is uploading fake KYC. This rating will help us to judge the bank activities and remove the fraudulent bank from our network.

The admin can anytime disallow the bank from upvoting/downvoting.

Depending on the number of reports and the number of banks present in the network, it will be decided whether any bank is allowed to downvote or upvote. If any bank gets reported more than one-third of the banks present in the network, it will not be allowed to do KYC anymore.

You can use some other logic to identify corrupted banks over the network. (For example, if more than half of the banks report the bank or the upvoted customers by a bank get more than a threshold number of downvotes by the other). If a bank is corrupted, set the 'isAllowedToVote' variable of the bank struct to false.

Phase 3:

In this phase, the smart contract will be deployed over a private network that is put up between various banks.
Banks can use the functionalities of the smart contract over this private Ethereum network.
Banks need to have an account on the private network to interact with the smart contract.