Optimizing e-KYC Process Using Distributed Ledger Technology and Smart Contracts

Hrushikesh Hanbar, Varan Shukla, Chirag Modi(✉), and C. Vyjayanthi

National Institute of Technology Goa, Farmagudi, Ponda, Goa, India
hrushikeshrohit@gmail.com, varanshukla@gmail.com,
{cnmodi,c.vyjayanthi}@nitgoa.ac.in

Abstract. The current know-your-customer (KYC) process in banks needs the customer to go through the entire process of KYC in every bank. This process is redundant, time consuming and costly. To address this problem, we propose a an optimized solution for e-KYC process using distributed ledger technology and smart contracts, which aims at reducing the overall cost of KYC verification process in banks. With the help of the proposed smart contracts, the KYC verification needs to do once for a customer. The result of this KYC verification process is securely shared among other banks with the customer's consent. It ensures that the relation between a customer and a bank is kept hidden from other banks. The proposed e-KYC solution offers efficiency, overall cost reduction and the improved trust with the increased transparency and throughput. The proposed solution is implemented on hyperledger fabric blockchain, and throughput and latency are analyzed to test its feasibility.

Keywords: e-KYC · Optimization · Blockchain · Smart contracts · Hyperledger

1 Introduction

Know Your Customer (KYC) is the process of a business or organization verifying the identity of its clients or customers. It is required to assess the suitability of customers. Especially, banks and other financial organizations are employing the KYC process to make sure that their customers provide the due diligence information required to fulfill the anti-money laundering regulations. Therefore, KYC has become a mandatory and crucial procedure for most of the businesses [1]. However, there is a lot of efforts and cost involved in the KYC verification. In the current system, when a customer opens first account with a bank, the process cost recurs (please refer Fig. 1 [2] as this process is labor and time intensive. Some banks don't have enough staff to attend the number of Anti-Money Laundering (AML) alerts, and it has more than 85% false positive rate. Hence, an adaptive and time efficient system is needed which should offer more granular data. An inefficient system costs banks as they are failing to fulfill KYC AML compliance and thus, fined by governing authorities. In 2018, Dutch bank ING was failed to fulfill Dutch AML compliance, and as a result, it was fined by 900 million dollars [3]. Thus, there is a need of an optimized system which can achieve an efficient KYC and AML, while creating the least amount of friction for banks and its customers.

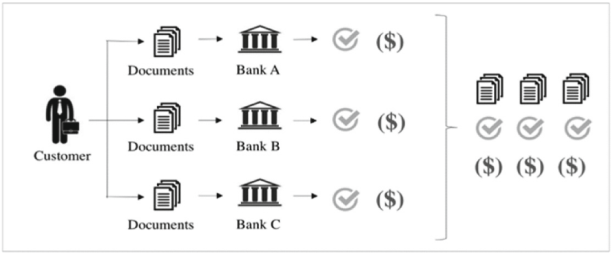

Fig. 1. KYC process and cost structure [2].

To address the above problem, the blockchain/distributed ledger technology (DLT) can play a vital role. A blockchain is a distributed database, in which list of records forming a block are cryptographically linked together and maintained through the consensus of the entire network of transacting peers. These peers can have smart contracts (decentralized replicated applications) which help to carry out the interactions among parties without the involvement of a trusted third party and publishes the results to the blockchain [4]. Blockchain offers data immutability, transparency and trust building features. As KYC verification process needs privacy along with auditability, blockchain can better help to share the customer data among the banks and to reduce the overall effort/cost involved in it [5]. However, the relationship between a customer and a bank must be hidden from other banks while sharing this information due to competitiveness. As a result, customers need to repeat the KYC process at every banks. At the same time, customers should know about sharing of their data in such network. Thus, with the blockchain, customers should do KYC process only once and the result of the same should be shared with other banks. In addition, the cost borne by the first bank for customer verification should be shared by the later banks.

In this paper, we propose the optimized e-KYC process by leveraging the key features of the blockchain and smart contracts. We propose the cryptographic solution to anonymously share customer's data among the banks with the customer's consent over blockchain. Here, the verified KYC documents for a customer are shared with other banks by considering the customer's consent. The proposed e-KYC process addresses the problem of repeated on board of customers in the due diligence process for KYC in different banks. The KYC process cost is shared among all the banks which wish to do the KYC verification for a customer. The privacy of customer-organization relationship is maintained.

The proposed solution is implemented using the hyperledger fabric blockchain with the proof-of-authority consensus method. We analyze the experimental results on throughput and latency of the proposed e-KYC process.

The rest of this paper is organized as follows: Sect. 2 discusses the existing e-KYC solutions with research gaps, while the proposed e-KYC solution is discussed in Sect. 3. Section 4 analyzes the experimental results of the proposed solution. Section 5 concludes our work with references at the end.

2 Background and Related Work

2.1 Existing Efforts on e-KYC Process

In literature, a very few attempts have been made towards simplifying the KYC process and increasing its efficiency without compromising security. We have observed that the e-KYC process needs immutability of the records. The cost of e-KYC should be further reduced and if multiple banks are involved for a customer, then the overall cost should be shared among them. In addition, it should have resistance to compromise capability. We have investigated a role of the national regulator such as RBI in India for a DLT-based e-KYC solution. However, the identity of the individuals is revealed during KYC approval process and storing the documents. Although blockchains offer data transparency and tracking, the privacy and auditability of a customer is also important for many applications. The proposed solution attempts to achieve anonymous KYC details sharing among organizations with the consent of users and cost distribution of KYC process among the involved organizations. To offer anonymity of transactions, zero knowledge proofs are used in most of the proposals. Zerocash [6] offers the decentralized anonymous payments. It guarantees strong privacy for transactions. However, it lacks in auditability. As many applications needs auditability, later Zerocash [6] was enhanced to offer privacy and accountability. In addition, it focuses more on payments and thus, it is very difficult to use in applications which require data or asset transfer.

IBM has developed “Proof-of-Concept Blockchain-based Shared KYC” in January 2018 in collaboration Deutsche Bank and HSBC [7]. Shyft [8], a blockchain-based network allows users to transfer data with regulatory compliance. It has a creditability feature which offers users with a reputation score based on compliance and historical transaction activity. Here, customers, data attesters and data consumers interact with each other via the shyft network, and thus the anonymity of customer relationship is not maintained. Other solutions like kyc-chain.com [9], kyc.legal [10], and Fides (norbloc.com) [11] are proposed use blockchain for KYC verification. However, a successful implementation is not reported yet for document validation by the trusted participants in the network. In addition, it is very much important to offer data validation among the banks with the privacy of the customer-bank relationship. Some solutions such as like KYC services [12] attempts to offer data standardization and harmonization in KYC processes among banks. However, it relies on a trusted third party [1].

2.2 Blockchain and Inter Planetary File System (IPFS)

A blockchain is a distributed database, in which list of records forming a block are cryptographically linked together and maintained through the consensus of the entire network of transacting peers. Bitocin [13], a cryptocurrency is the first and widely used application of blockchain. Later, many other blockchains are evolved. Ethereum [14] is also a cryptocurrency which leverages some features of Bitcoin. It supports smart contracts which can be used for many applications. Bitcoin and Ethereum are considered as public permissionless blockchain technology. However, they lack in offering sufficient level of performance in many applications which require to identify participants, permissioned network, high throughput, low latency and privacy of transactions and data pertaining to business transactions [15].

Hyperledger Fabric [15] is a permissioned distributed ledger technology (DLT) platform which offers access control and can be used for many applications such as banking, healthcare, supply chain etc. It supports smart contracts written in general purpose programming languages. It supports pluggable consensus protocols that make it feasible for various use cases and trust models. It allows to deploy and run different consensus protocols which can address scalability issues and costly mining as in bitcoin and ethereum. A smart contract (chaincode in Fabric) is the self running or custom running program/code to run business logic of an application. Most of the existing blockchain frameworks follow order-execute method. Here, all the transactions are validated through the consensus method. The validated transactions are ordered and propagated in a blockchain network in the form of a block. At the end, each peer executes the block sequentially as per the given criteria and put it on blockchain, once the execution is completed from any of the peer. However, it affects the performance and scalability. In contrast to this, hyperledger Fabric performs execute-order-validate on transactions which addresses the problem of scalability and performance as in order-execute model.

Inter Planetary File System (IPFS) [16] is a distributed system for storing and accessing files, websites, applications, and data. Instead of being location based, IPFS addresses a file by its contents. The address of a content is hash of contents and hence unique to that content only, and thus it offers to store large amounts of data in IPFS, and its links on a blockchain. It secures contents as only hash of the content is recorded on the blockchain. In the proposed e-KYC solution, we use the IPFS as permissioned database and hash of the encrypted documents is recorded in hyperledger.

3 Proposed e-KYC Solution

3.1 Objective and Design Goals

The objective of the proposed e-KYC solution is to optimize e-KYC process using distributed ledger technology and smart contracts with following design goals:

Anonymity of relationships: The proposed solution must keep the relationship between the customer and bank confidential. The banks need to access customer's data. However, the identity of the bank must be hidden, while sharing the data from one bank to another. The consent of a customer is must for the sharing of data from one bank to another.

Minimizing KYC cost: The cost incurred for conducting the KYC process must be shared among all the banks which wish to do the KYC verification for a given customer.

Creditability: Every customer must have a creditability measure which helps banks to build confidence on the customer.

High throughput and low latency: The proposed solution should be able to perform high number of transactions in a given time with low latency.

3.2 Design of the Proposed e-KYC Solution

In the proposed e-KYC solution, we use a permissioned blockchain-Hyperledger fabric whose access is controlled by a central regulator such as RBI in case of bank's KYC in India. The financial organizations such as banks are enrolled in this network of blockchain by the central regulator and are issued identity certificates, as given in [14]. It includes two types of certificates: (1) Enrolment Certificate - Long term and to identify bank. (2) Transactional certificates - Short term and unique to each customer bank relationship. The regulator/identity provider maps all the transactional certificates to their corresponding enrollment certificates as transactional certificates have pseudonyms of the banks instead of their original identity. This helps in maintaining the anonymity of interacting banks. In e-KYC, several entities need to interact with each other through smart contracts. The interaction of entities with the smart contracts, database and each other is given in Fig. 2. The customers interact with the network through the customer portal which acts as an entry point for the customers. It is used to transfer customer's KYC data and credentials, and to interact with banks for giving KYC verification consent and data. The customer is also provided with the enrollment certificate which is a unique identification number given by the regulator/identity provider (e.g. Certification authority) when he/she sign up on the customer portal. Banks have APIs to interact with the customers through their portal. Banks interact with off-chain database to store and retrieve the encrypted documents.

Recording Documents. For the first institute recording the KYC documents, the customer must provide the documents in person and have to be verified by the bank. It is a tedious process and involves a cost. After the customer has been verified for those documents and the consent for these documents and bank has been recorded, the bank then selects a symmetric encryption key such as AES, encrypts these documents and store them in an off-chain database. This off-chain database is an IPFS database. After this, the bank records the hash of these documents along with the link encrypted using the same key on the

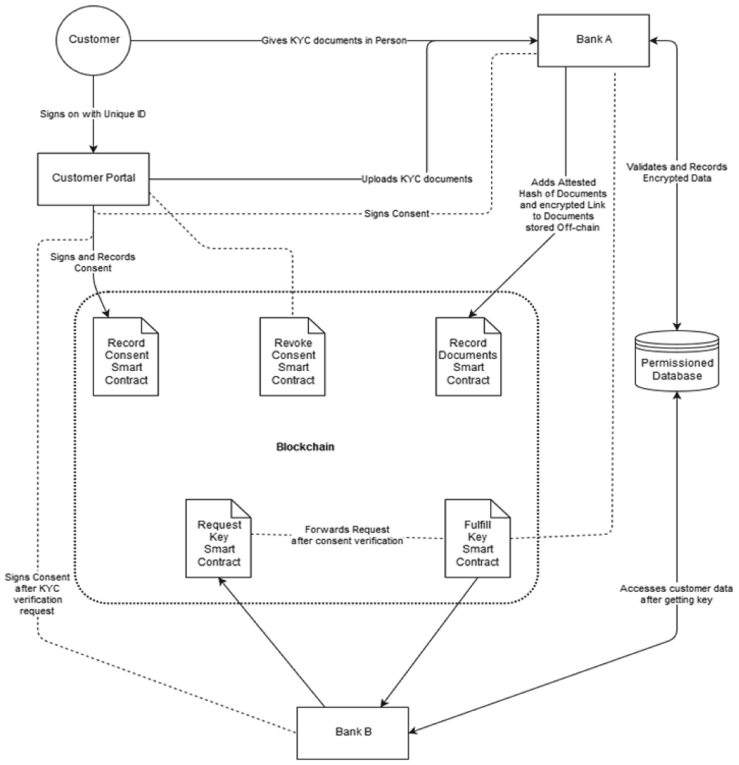
Customer
Gives KYC documents in Person
Bank A
Signs on with Unique ID
Customer Portal
Uploads KYC documents
Signs Consent
Adds Attested Hash of Documents and encrypted Link to Documents stored Off-chain
Validates and Records Encrypted Data
Signs and Records Consent
Record Consent Smart Contract
Revoke Consent Smart Contract
Record Documents Smart Contract
Permissioned Database
Blockchain
Request Key Smart Contract
Forwards Request after consent verification
Fulfill Key Smart Contract
Signs Consent after KYC verification request
Accesses customer data after getting key
Bank B

Fig. 2. Entity interaction in the proposed e-KYC process.

blockchain through record document contract. This link can be later used to share these documents with other banks and hash helps to prove their integrity. The contract for recording the document is as follows:

recordDocument (customerID, documentsList, transactionCertificate, documentsHash, encryptedLink)

- Validate customer signature
- Check for the presence of consent on the ledger by querying
- Records the hash and encrypted link of the encrypted documents on DLT

As shown in Fig. 3, the data stored on blockchain includes the Customer Profile Table, Customer Consent Table, Encrypted Key Table, Hashed Document Table. The customer ID is a primary key in each table. The consent record is also linked by type of document and bank's transactional certificate. Each customer's profile is addressed by a globally unique ID with type, and a list of such IDs and types are recorded in a profile table. The document hash with the link to off-chain is stored by encrypting it using symmetric key (AES). Such AES keys are stored in a separate table by encrytping using the public keys as given in the transactional certificates. Here, the encrypted documents of a customer are stored in this IPFS database. This encrypted data is sent to the permissioned database with a corresponding link. This link can later be used by other banks to get the encrypted data.

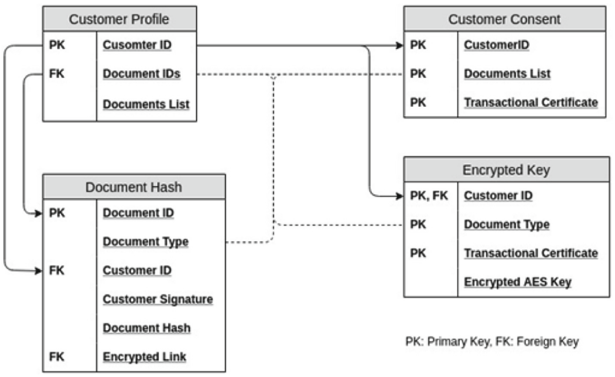

Fig. 3. Data model in the proposed e-KYC solution.

Consent. A customer gives the consent to a bank for accessing KYC details by recording "the consent" on the blockchain. A consent is a structured record including customer ID and the transactional certificates of the bank to which the customer has given consent along with the list of documents to access. The smart contract for recording consent is as follows:

recordConsent (consent, transactionCertificate, customerCertificate, signT, signC)

– Validate customer signature using Hyperledger Fabric CA server API
– Validate bank signature using Hyperledger Fabric CA server API
– Build consent object to be recorded on the ledger
– Write Consent and order the transaction using putState Fabric API

As shown in Fig. 4, through recording consent, the consent is recorded for each bank wishing to share KYC details of the customer before they start the process of borrowing this information from other banks. This consent presents with the first bank which has actually uploaded the documents for the first time. The procedure of recording this consent involves following steps:

1. The customer selects the list of documents through the portal and sends a request to a bank for which he/she wants to give consent.
2. The bank selects a unique transaction certificate for this bank customer relation and replies to customer with this transaction certificate signed with enrollment certificate secret key and consent signed with transaction certificate's secret key. The bank must maintain a map of customers to transaction certificates and key used to encrypt documents for later exchange of key.

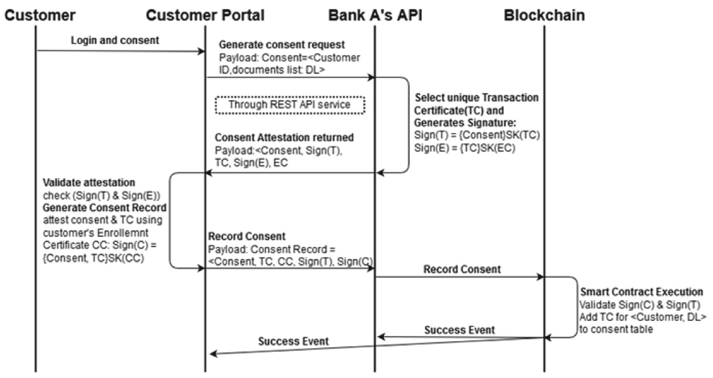
Customer
Customer Portal
Bank A's API
Blockchain
Login and consent
Generate consent request
Payload: Consent=<Customer ID,documents list DL>
Through REST API service
Select unique Transaction Certificate(TC) and Generates Signature:
Sign(T) = {Consent}SK(TC)
Sign(E) = {TC}SK(EC)
Consent Attestation returned
Payload:<Consent, Sign(T), TC, Sign(E), EC
Validate attestation
check (Sign(T) & Sign(E))
Generate Consent Record
attest consent & TC using customer's Enrollemnt Certificate CC: Sign(C) = {Consent, TC}SK(CC)
Record Consent
Payload: Consent Record = <Consent, TC, CC, Sign(T), Sign(C)
Record Consent
Smart Contract Execution
Validate Sign(C) & Sign(T)
Add TC for <Customer, DL> to consent table
Success Event
Success Event

Fig. 4. Customer's consent for bank created through one round trip.

3. The customer then validates these signatures to make sure he/she is interacting with correct bank and then invokes the record consent smart contract on blockchain to record consent and corresponding transaction certificate. The smart contract records the consent on the ledger after validating customer and bank sign.

When other banks with consent from customer wishes to borrow the verified credential, the exchange of key between these two banks happens without knowing each other's identity through requestKey and fulfillKey contracts as given below:

requestKey (customerID, documentsList, transactionCertificate, signT)

- Check for consent and sign
- Select another Banks' Tcert at random
- Ask all banks to fulfil key if Tcert matches

fulfillKey (customerID, documentsList, transactionCertificate_A, transactionCertificate_B)

- Check the presence of consent for the requestor and provider
- Select another Banks' Tcert at random
- Record the AES key encrypted using requestor's TCerts' public key

Figure 5 shows the process of exchanging the key between banks. It performs following steps:

1. The second bank first selects a unique transaction certificate for this relation with the customer and record a request on the blockchain for the key of those documents after signing this request with its transaction certificate's secret key. The requestKey contract selects a random transaction certificate holder, among all those having consent for this customer, who must satisfy this request of key. This is done to ensure that not all banks try to fulfil this request at a time.

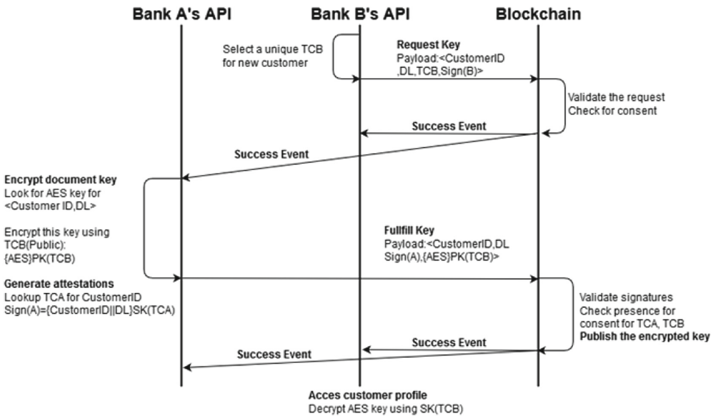
Bank A's API
Bank B's API
Blockchain
Select a unique TCB for new customer
Request Key
Payload:<CustomerID ,DL,TCB,Sign(B)>
Validate the request
Check for consent
Success Event
Success Event
Encrypt document key
Look for AES key for <Customer ID,DL>
Encrypt this key using TCB(Public): {AES}PK(TCB)
Fullfill Key
Payload:<CustomerID,DL Sign(A),{AES}PK(TCB)>
Generate attestations
Lookup TCA for CustomerID
Sign(A)={CustomerID||DL}SK(TCA)
Validate signatures
Check presence for consent for TCA, TCB
Publish the encrypted key
Success Event
Success Event
Acces customer profile
Decrypt AES key using SK(TCB)

Fig. 5. AES encryption key sharing in the proposed e-KYC solution.

2. The success of this request key notifies all banks, including the bank with key to customer credentials selected by the contract. This bank realizes that it is its transaction certificate which has been selected and has key for the credentials of the customer asked in this request. It then looks for the corresponding symmetric key and encrypt this key with the second banktransaction certificate's public key and invokes fulfilKey contract after signing request with its transaction certificate's secret key.
3. The smart contract validates the presence of consent for both transaction certificates and then records this encrypted key on the blockchain.
4. Using this recorded key, second bank is able to verify KYC for a customer who has given consent.

Here, the cost sharing mechanism works as follows: An average cost per customer is decided by unanimous agreement of all the parties involved. A fixed value token system (T) is used to meet such a requirement. To get the customer data from a bank, the requesting bank has to pay his share of cost via this token. This cost is shared among all the banks which currently hold the customer's identity. This ensures no bank single handily exploits the systems for its own benefit and the overall cost of this process is shared among all the banks which desire the KYC data of that particular customer. Suppose, bank *A's* cost is T for customer *C's* KYC verification. When bank B receives confirmation from bank A for customer *C's* KYC verification, it pays cost *T/2* to bank A. Thus, cost to bank A and B becomes *T/2*. Now, bank C needs to do KYC verification for same customer and bank A has given confirmation, then bank C has to pay *T/3* cost to bank A. As a result, bank *C's* cost is *T/3* and bank *A's* cost becomes *T/6*. Thus, the overall cost of KYC verification of a customer is shared among the banks. In addition, the bank fulfilling the key request is provided with an extra fractional reward later to keep them motivated towards this business network.

4 Experimental Results and Analysis

4.1 Experimental Setup

For the performance evaluation of the proposed e-KYC solution, we have implemented a prototype using Hyperledger Fabric [15]. As shown in Fig. 6, we set up a virtual prototype network using Docker on a computer with following specifications: Operating System: Ubuntu 18.04 (Linux), Processor: Intel core i5 8250U, Memory: 8 Gb. We have considered two organizations, namely: Banks and Customer portal with one peer for each organization. Peers are owned by organizations and considered as nodes in the network which host the ledger and smart contracts. Each organization also have a certification authority (CA) and membership service provider (MSP) to issue certificates and to provide role to identities in network.

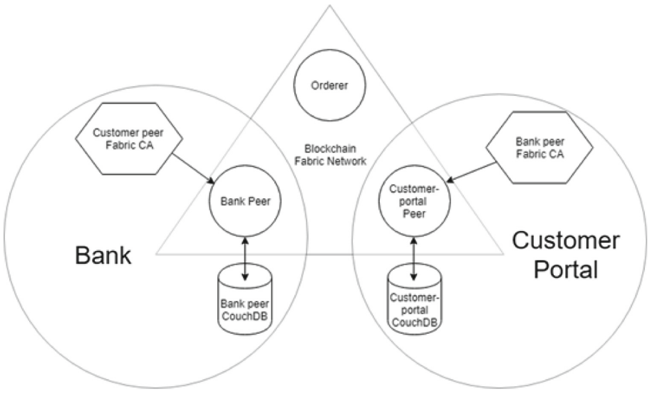

Fig. 6. Experimental setup of the proposed e-KYC solution.

We have generated the required certificates and keys for the banks and customers, with the genesis block which bootstraps the service. We have created a number of transactions to configure the channel. All the Docker containers are launched along with the couchDB containers for each peer. CouchDB serves as the state database to store key value pairs of the ledger and allows rich query against the chaincode data. Docker starts a virtual network, e-KYC network for the services running in container to communicate. The peers are then made to join the channel. A client container is launched to execute commands in the peer containers, which is used to install the proposed smart contracts (KYC-chaincode) in both the peers. Here, chaincodes are written in node.js, through which blockchain is accessed. We have deployed different chaincodes viz; recordConsent, recordDocument, requestKey and fulfillKey in node.js (as discussed in Sect. 3.2). APIs of Banks and customer portals are implemented in node.js. These RESTful APIs are implemented with express.js to record consent and facilitate exchange of keys. They wait for consent record request from a customer and reply with the transaction certificates. We use MongoDB database for banks to store the mapping between transaction certificates, customerID, and the key for the encrypted documents. This key is encrypted using a master key of bank. Bank's API also waits for any key request made for their transaction certificates and invokes the fulfil key smart contract. New users register by Fabric Node SDK through node.js. An admin is enrolled first and his/her ID is used by the customer portal to register new customers. This returns a wallet ID which is used by a customer to interact with the blockchain and perform transaction such as record consent. Same SDK is used to enroll banks and to provide them with enrollment certificates. In addition, it is used at bank API side to get new transaction certificates. All these certificates are X.509 and SHA-256 with ECDSA used for digital signature generation.

The chaincode mentioned above in the proposed e-KYC solution are tested using Hyperledger Caliper, a blockchain benchmark tool [17]. Its functions are subjected to a large number of test transactions at different sent rates. We have derived the results in terms of throughput and latency in executing the transaction at different sent rates.

4.2 Results and Discussion

Figure 7 represents the number of the customer's transactions over chaincode served by the blockchain network with different sent rate of transactions. It shows that the maximum throughput with a transaction over blockchain on the RecordConsent function of chaincode is 5 tps on average. We have observed that the throughput initially increases with the sent rate, but then saturates around 5 tps. Similarly, the latency increases with the increase in sent rate as the transaction requests pile up due to high send rate (please refer Fig. 8).

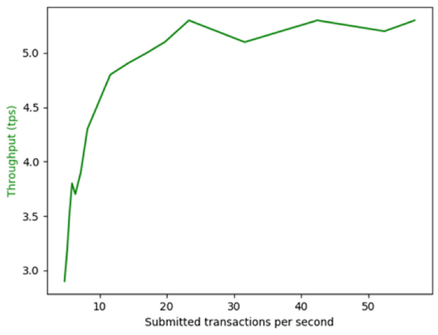

Fig. 7. Throughput of the customer's chaincode in the proposed e-KYC solution.

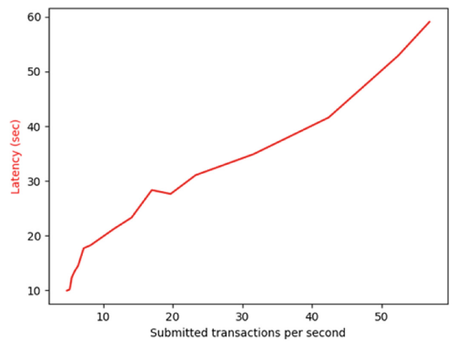

Fig. 8. Latency of the customer's chaincode in the proposed e-KYC solution.

As shown in Fig. 9, the maximum throughput with the bank's transactions over blockchain on fulfillKey chaincode is 9 tps on average. We have observed that the throughput initially increases with the sent rate, but then saturates around 9 tps. In addition, the latency increases with the increase in sent rate as transactions tend to queue up (please refer Fig. 10).

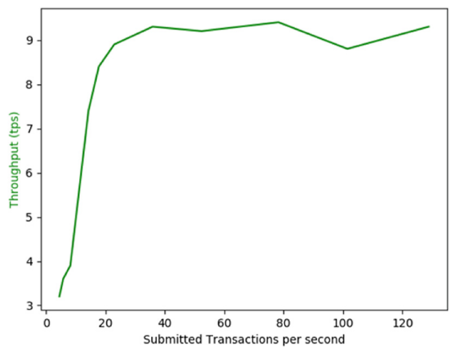

Fig. 9. Throughput of the bank's chaincode in the proposed e-KYC solution.

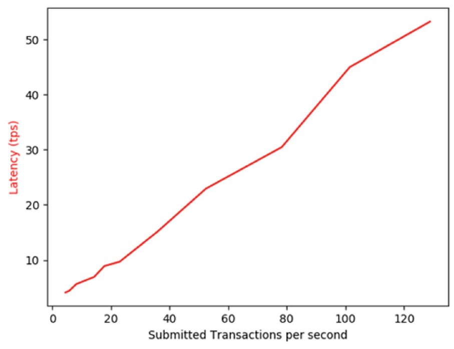

Fig. 10. Latency of the bank's chaincode in the proposed e-KYC solution.

In the proposed e-KYC solution, the transaction certificates act as pseudonames which are used among the banks to share information without their identity. This helps in achieving anonymity of relationships constraint as the customer bank relationship is not exposed while sharing the key for those encrypted KYC documents. The smart contracts make sure that this information is not shared through the network without the consent of the customers. The smart contract validates customer's certificates and signed consent for transactional certificate. This process ensures that a man-in-the-middle attack is not possible and the bank cannot forge a fake consent as it is using digital signature on each transaction.

The overall cost of KYC process is shared among the banks, which wish to do a KYC verification for a given customer. Data transparency and immutability features of the blockchain help in achieving the user's trust. This helps banks to improve confidence of the users.

As per experimental results, the throughput of the proposed e-KYC solution is comparable with the existing process and latency is affordable.

5 Conclusion

The KYC revolution has addressed the problem of money laundering and other unidentified funding. The current KYC process in banks is redundant, time consuming and costly for every customer. To address these problems, we have proposed an optimized e-KYC solution which leverages the benefits of the blockchain and smart contracts. With the help of proposed smart contracts, a customer need to do one time KYC verification process only and results of same is shared among other banks with the consent from a customer, while maintaining the confidentiality of bank-customer relationships, improving the accountability, cost sharing among the involved banks, sufficient throughput and affordable latency as compared to current solutions. We have implemented the prototype of the proposed solution using hyperledger fabric blockchain and evaluated the results in terms of throughput and latency in performing the transactions. The experimental results and analysis of the proposed solution are very encouraging to make the proposed solution standardized and widely utilized in financial institutions. In future, the throughput of the proposed solution can be further improved by incorporating lightweight crypto algorithms and better performance databases in peer nodes for the fast execution of chaincode.

References

1. Rajput, V.U.: Research on Know Your Customer (KYC). Int. J. Sci. Res. Publ. **3**, 541–546 (2013)
2. Parra Moyano, J., Ross, O.: KYC optimization using distributed ledger technology. Bus. Inf. Syst. Eng. **59**(6), 411–423 (2017)

3. Munserman, R., Blackman, A.: ING to Pay $900 Million to End Dutch Money Laundering Probe. https://www.bloomberg.com/news/articles/2018-09-04/ing-to-pay-784-million-in-fines-to-settle-dutch-criminal-case. Accessed 13 Aug 2019
4. Martens, D., Van Serooskerken, T., Alexander, V., Steenhagen, M.: Exploring the potential of blockchain for KYC. J. Digit. Bank. **2**(2), 123–131 (2017)
5. Lootsma, Y.: Blockchain as the newest regtech application-the opportunity to reduce the burden of KYC for financial institutions. Bank. Finan. Serv. Policy Rep. **36**, 16–21 (2017)
6. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from Bitcoin. In: IEEE Symposium on Security and Privacy, pp. 1–56 (2014)
7. Curry, M.: Blockchain for KYC: game-changing RegTech innovation, IBM RegTech In-novations. https://www.ibm.com/blogs/insights-on-business/banking/blockchain-kyc-game-changing-regtech-innovation/. Accessed 13 Aug 2019
8. Shyft. White paper. https://www.shyft.network/pdfs/Shyft-Network-Inc-Whitepaper_V3.1.pdf. Accessed 13 Aug 2019
9. KYC-CHAIN. https://kyc-chain.com/. Accessed 13 Aug 2019
10. KYC Legal. https://kyc.legal/. Accessed 13 Aug 2019
11. Fides. https://www.norbloc.com/. Accessed 13 Aug 2019
12. KYC Services. https://ihsmarkit.com/products/kyc-services.html. Accessed 13 Aug 2019
13. Nakamoto, S.: Bitcoin: A Peer to Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. Accessed 13 Aug 2019
14. Buterin, V.: A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 13 Aug 2019
15. Hyperledger Fabric v1.4: Protocol Specification. https://github.com/hyperledger/fabric/tree/release-1.4/docs. Accessed 13 Aug 2019
16. Benet, J.: IPFS Content Addressed, Versioned, P2P File System. https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf. Accessed 13 Aug 2019
17. Hyperledger Caliper. https://www.hyperledger.org/projects/caliper. Accessed 13 Aug 2019