Know Your Customer using Distributed Ledger Technology

Matus Drgon

MInf Project (Part 2) Report

Master of Informatics
School of Informatics
University of Edinburgh

2021

Abstract

Know Your Customer (KYC) is a verification process financial institutions need to execute before they can start conducting business with new customers. The increasing level of regulations imposed on this process makes it burdensome. Distributed ledger technology (DLT) has been propounded to share the work and cost associated with onboarding a new customer between financial institutions operating with the client. In our previous work, we put forth a design and Solidity smart contract implementation of a DLT-based KYC system that focused on securing the process by introducing a probabilistic mechanism, making the system more robust. The primary objective of this work is to assess the privacy aspect of a KYC system based on a distributed ledger that preserves the probabilistic mechanism. We identify four conditions that cover the privacy of both financial institutions and customers operating via a distributed ledger. We provide design and implementation of two KYC systems on Corda blockchain. These introduce a compromise between fulfilling the privacy requirements and the level of involvement of a trusted third party (TTP) in the network. The first one meets all privacy conditions but requires a TTP to distribute the cost of onboarding a new client between the financial institutions. The second one requires no involvement of a third party but breaches some aspects of the privacy conditions.

Acknowledgements

First, I would like to thank to my supervisor, Aggelos Kiayias, whose expertise and mentorship have immensely helped me throughout this project. Second, I would like to express my gratitude to Lamprini Georgiou, a research assistant at the Blockchain Technology Laboratory at the University of Edinburgh, for her valuable advice on Know Your Customer process from the legal and procedural perspective.

It has been a thrilling journey. I am fortunate and grateful that throughout the two years, we have published a conference paper and our article has recently been accepted to the Journal of Digital Banking.

Table of Contents

Chapter 1

Introduction

1.1 Know Your Customer Process revisited

Know Your Customer (KYC) is an onboarding process financial institutions need to execute to verify new customers before they can start conducting business with them. This onboarding process counters financing of terrorism (CFT) and it is the first measure to prevent anti-money laundering (AML). The AML/CFT directives have been under a lot of scrutiny, with new regulations coming up every 2-4 years, making the KYC process highly regulated. Under the current settings, a financial institution that wants to facilitate a customer has to always beforehand execute the KYC process for this customer. This equally applies to a customer who wants to open up their first bank account, as well as to a customer who is already conducting business with several other financial institutions, and has been verified multiple times. Figure 1.1 depicts this practice for a customer who operates with N financial institutions.

Combined with the tight regulations, this process has unsurprisingly become financially and temporally onerous. According to Thomson Reuters 2017 Global KYC survey, large financial institutions (with 10$bn or more in revenue) on average employ 307 KYC compliance professionals and annually spend $150M to accommodate this process [23]. For major financial institutions, these annual costs can climb up to 500$m [8]. The average waiting period on this process was 26 days, resulting in low customer satisfaction. These inefficiencies stem from the nature of the KYC process under the current settings - always execute the full KYC onboarding verification without considering whether a similar process has been recently accomplished by other financial institutions.

Distributed Ledger Technology (DLT) has been proffered as an instrument for the KYC process that could mitigate its financial and temporal inefficiencies. Parra-Moyano et. al. [21, 22] offer the first comprehensive study on this topic. The authors consulted experts in financial industry and produced design for a KYC system that leverages DLT. According to their design, the DLT-based KYC system would only require the first financial institution (`Bank A`) to execute the KYC process for a customer. The institution would subsequently store the result of this process on the distributed ledger. When the customer would like to start operating with another financial institution (`Bank X`),

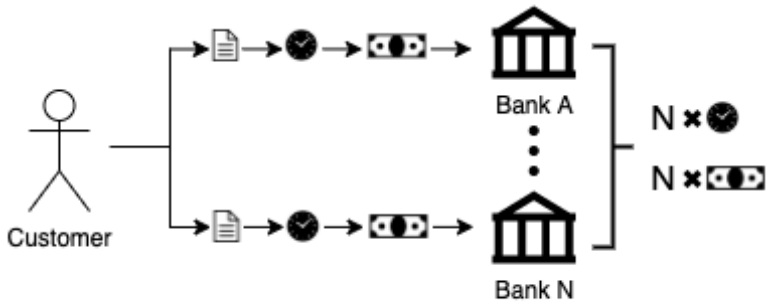

Figure 1.1: Simulation of the current KYC process.

it will not need to repeat the KYC process, but only retrieve the result of this process from the distributed ledger. Bank X would further need to pay an appropriate fee that would be delivered via the distributed ledger to Bank A to fairly share the costs Bank A incurred by onboarding the customer.

This DLT-based KYC system was the theme of our last year's work. We analysed it and found out that the system and its various modifications proposed by current literature, including [25, 19], are all susceptible to a single point of failure - a property we named *brittleness*. If a single FI would make an operational mistake in the KYC process, this mistake would be propagated via the distributed ledger and shared by all other financial institutions which would onboard the customer in the future. The system is brittle in a sense of coming apart when a single financial institution (Bank A in our example) makes a mistake in the KYC onboarding process.

We modified the design of the DLT-based KYC system by probabilistically requiring each financial institution that wants to onboard a customer in the future to repeat the KYC process. When the customer operates with several financial institutions, the single point of failure is effectively eradicated, making the system more *robust*. Under the assumption of collaboration between financial institutions via the distributed ledger, we showed that our robust system could simultaneously decrease the costs incurred by the KYC process and increase the security of the process when compared with how the KYC process is currently accomplished. The security of the system in this context refers to the probability of detecting false negatives - customers who are accepted during the onboarding process and later misuse their accounts for illegal activities, such as money laundering or financing of terrorist activities. Facilitating such a customer might cost a financial institution billions in fines [1]. Finally, we put forth a smart-contract implementation in Solidity programming language on Ethereum blockchain that simulates how customers and financial institutions would interact with the distributed ledger in our proposed DLT-based KYC system.

[1] In the US in 2020, twelve cases of AML non-compliance were detected that resulted in a total of €9.39 billion in fines. Goldman Sachs alone was fined €3.30 billion in that single year. `https://shuftipro.com/blog/record-breaking-fines-on-banks-for-kycaml-non-compliance`

1.2 Procedural background

KYC process can be broadly split into two parts: (1) Customer Identification Program (CIP), where the customer's name, date of birth, and address verified. This usually requires a picture of the customer's ID, drivers license, or passport. (2) Customer Due Diligence (CDD), where the financial institution's aim is to assess the type of transactions the customer will conduct. The depth of due diligence is determined by an estimated risk level of the customer. Common practices include PEP (politically exposed person) search, negative news search, and watchlist check. If the customer presents a moderate or higher level of risk, the customer may need to present previous credit card statements or other references from previous financial institutions.

1.3 Cost-sharing

A key property of a KYC system that utilizes distributed ledger technology for sharing the onboarding cost of a customer is that this cost is fairly shared between financial institutions that operate with the customer. We provided details on how this can be attained in a DLT-based system that probabilistically requires financial institutions to repeat KYC for a customer in our previous work [9]. To summarize the findings of our previous work, assume a customer who has approached $N-1$ financial institutions and would like to open a bank account at Bank X, making it the Nth institution the customer has approached. Further assume that the average cost of executing KYC for this customer is c and the probability of repeating KYC is p. A random decision is made determining whether Bank X has to repeat KYC for the customer. If it does, then it independently repeats KYC process for the customer and can start operating with the client after this has been completed. Bank X is not required to pay any fee as it has incurred a cost by having to repeat the KYC process. Otherwise, Bank X only has to pay a fee and can launch operating with the client as soon as it has paid the fee. The size of the fee is the expected cost a financial institution would face when the customer operates with N financial, the average cost of single KYC for the customer is c, and the probability of repeating KYC is p. This expected cost is:

$$\frac{c + c * (N-1) * p}{N} \tag{1.1}$$

The equation comes from (1) requiring the first financial institution the customer approached to unconditionally execute KYC, which incurred cost c to the institution, and (2) each other financial institution in the future to repeat KYC with probability p. Assume Bank X does not need to repeat KYC and that there are k financial institutions that either executed or repeated KYC for the customer. Note that k approaches $1 + (N-1) * p$ as N goes to infinity. Then each of the k financial institutions receives a fair share of the fee paid by Bank X. The share a single financial institution receives is:

$$\frac{c + c * (N-1) * p}{N * k} \tag{1.2}$$

1.4 Limitations of previous work and objective of this project

The KYC process is a complex business use-case that requires a thorough treatment from legal, procedural, and implementation standpoint. Our previous work touched on the legal aspect in terms of referring to various jurisdictional regulations [4, 16, 12, 11] and recommendations [10] associated with the process. From a procedural perspective, we outlined the design of the system, evaluated how it would fairly distribute the KYC expenses between financial institutions, and quantified the cost efficiency and security of the system under certain assumptions. Finally, from an implementation view, the smart contracts we supplied provided for financial institutions a means of interacting with the distributed ledger to share the KYC costs, use the platform to accept or reject a new customer, and a create a digital profile for the customer.

Privacy is essential in the context of financial industry and it has to be a key property of any customer-onboarding mechanism. The major limitation of our previous work was the neglect of privacy considerations on the distributed ledger. We outlined that a public blockchain network, such as Ethereum, could not be used due to the sensitivity of the customer data. We proposed employing our smart contracts programmed in Solidity on a private blockchain network that is compatible with the Ethereum network (e.g. a private fork of the main Ethereum network or Hyperledger Sawtooth). However, importantly, we did not discuss the privacy aspect in detail and if it would be fulfilled on such a private blockchain.

A private blockchain in this context implies that a party can enter the blockchain only after obtaining a permission. This permission could be either given by a trusted third party (the regulator or a private independent company), or by a consortium of the financial institutions. Such a blockchain ensures zero access from the general public into the network, but does not automatically ensure that the privacy of customers and financial institutions on the network would be fulfilled. As long as the information is propagated via a public network protocol, such as the gossip protocol which is used on Ethereum blockchains, the information regarding onboarding a new client would be freely shared with other parties on the network - including other financial institutions and customers. Even if this information is pseudonymized, a pattern could eventually reveal that could lead to either revealing the real identities of customers behind their digital profiles, or the connection between financial institutions and the customers they operate with.

Privacy can be perceived from the perspective of customers and financial institutions. We identify the following privacy requirements, sorted by the order of importance:

1. The personal documents a customer supplied for the KYC onboarding process, as well as any other documents and confidential customer information, cannot be revealed to any party that was not an intended recipient of these documents or information. In other words, only the financial institution(s) the customer operates with have access to the customer's documents and confidential information.
2. The general public should not have access to the blockchain network.

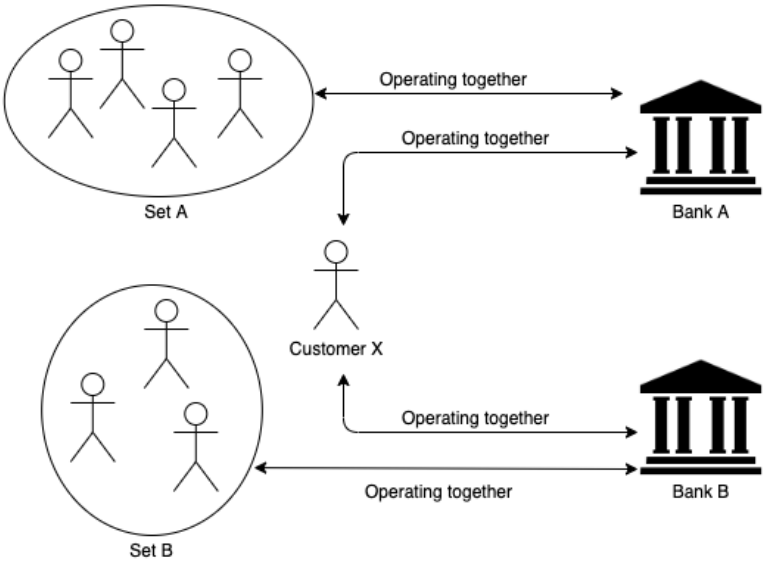

Figure 1.2: Bank A and Bank B both operate with Customer X. Bank A also operates with customers in set A, and Bank B operates with set B. According to the third privacy condition, only Bank A knows that it operates with set A and Customer X. Likewise, only Bank B knows that it operates with set B and Customer X. According to the fourth privacy condition, only Customer X knows that he/she operates with both Bank A and Bank B.

3. The set of customers a financial institution operates with is only known to the financial institution.
4. The set of financial institutions a customer operates with is only known to the customer.

Objective The objective of this project is to supply a design and implementation of a DLT-based KYC system that is robust (i.e. uses probabilistic repetition of the KYC process) and fulfills the privacy aspects we outlined.

Chapter 2

Corda blockchain

2.1 Technological background

Corda is a form of distributed ledger technology that, unlike Bitcoin [18] or Ethereum [3], does not have a single central distributed ledger. Instead, a ledger exists between any non-empty set of parties that executed a transaction with each other. Subsequently, the consensus is not achieved on a central distributed ledger at a network level, but has to be achieved for each ledger in the network and only by the parties with access to the ledger. A ledger is a chain of transactions accessible by a party only if the party was required to sign transactions the ledger is composed of. Figure 2.1 presents a Corda network with three participants - Alice, Bob, and Charlie. The network stores 8 different transactions composing 6 different ledgers. Ledger $\mathtt{L}_1$ is accessible by all parties, $\mathtt{L}_2$ is only accessibly by Bob and Charlie and $\mathtt{L}_3$ is accessible by Alice.

State is an immutable object stored on the ledger. It can present any form of information, such as a request for the KYC process or the result of this process. A state can be known by one or more parties. Figure 2.2 depicts a Corda network with 9 different states. State 1 is shared by all of the participants and everyone knows it. State 2 is only known by Alice and Bob - Charlie does not have access to this state. Similarly, Bob does not know of states 3, 6, and 7. Alice does not know of states 8 and 9. States 4 and 5 are privately known by Alice and Charlie respectively.

Figure 2.3 presents a detailed state view. Each state needs to define its participants - nodes that should be alerted when the state is created or consumed. For instance, when a customer wants to open a bank account at a financial institution and makes a request for this, the participants could be the financial institution and the customer. The state properties and functions are used to store and retrieve the actual information from the state. Each state contains a reference to a contract that provides the validation logic for interaction associated with the state. For instance, when we propose an output state defining the result of KYC for a customer, we want to ensure that the KYC cost is not a negative number. The contract does this validation to enforce a state has desirable properties and is also a form of preventing misuse by malicious parties.

Similar to Bitcoin, Corda uses UTXO (unspent transaction output) model - a design

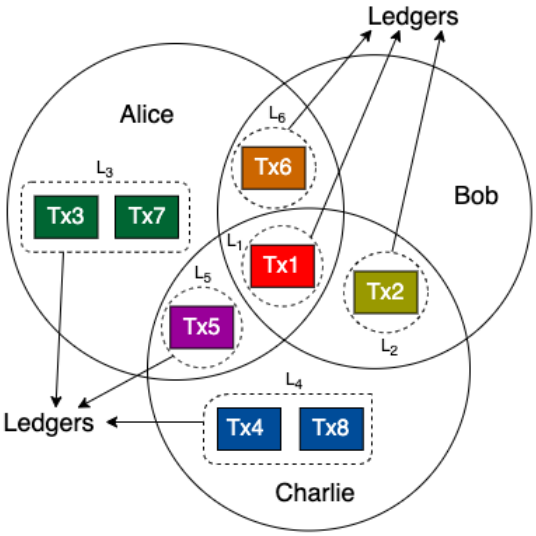

Figure 2.1: Corda network with three participants - Alice, Bob and Charlie. Instead of containing a central distributed ledger that would involve all the transactions (`Tx1`, `Tx2`,...,`Tx8`), Corda has multiple ledgers accessible only by the parties that are involved in the transactions that comprise the ledger. For instance, ledger containing transaction `Tx6`, as well as the transaction itself, is only accessible by Alice and Bob. The ledger containing transactions `Tx3` and `Tx7` is only accessible by Alice.

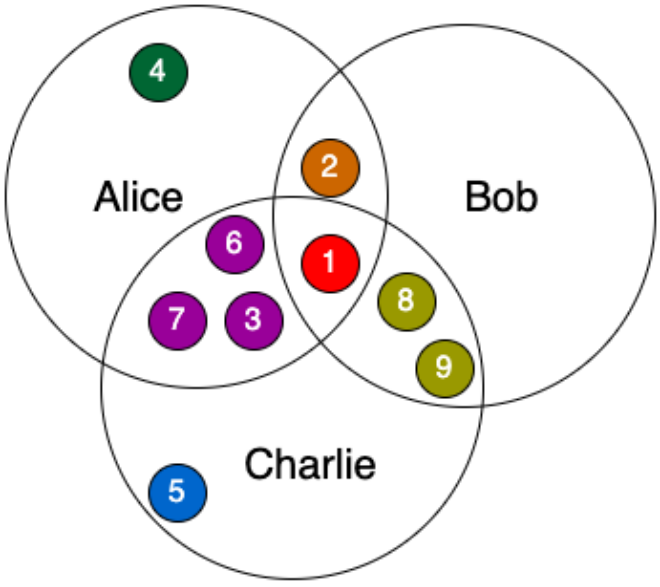

Figure 2.2: Corda network with three participants - Alice, Bob and Charlie. Each circle with a number inside it presents an immutable state stored on a distributed ledger. For instance, states 3, 6 and 7 are only known by Alice and Charlie; states 5 is only known by Charlie, and state 1 is known by everyone. Inspired by: `https://docs.corda.net/docs/corda-os/4.7/key-concepts-ledger.html`

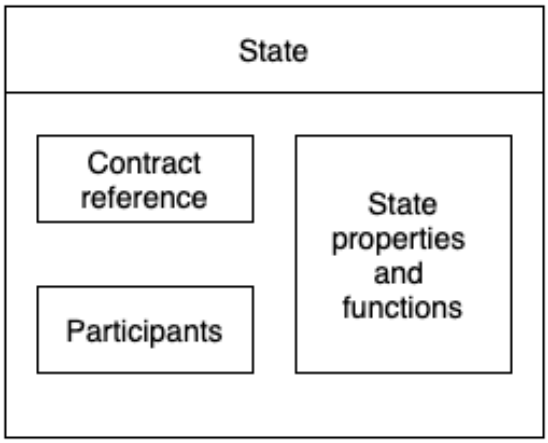

Figure 2.3: State structure. Each state contains reference to a single contract, a list of participants that should get notified when the state is updated, and a list of properties. Figure inspired by `https://docs.corda.net/docs/corda-os/4.7/key-concepts-states.html`

under which a transaction has a set of input states and a set of output states. The input states are spent after the transaction is executed and cannot be used in any future transaction. The output states, on the other hand, can serve as input states in a future transaction. Given the immutability of a state, it is not possible to modify it after it has been defined on the ledger. To update a state, we use the state as an input in a transaction whose output is the updated version of the state.

Figure 2.4 shows the structure of a simple transaction with a single input and output state. The input and outputs states are validated by a contract. Each transaction has a set of required signers who need to sign the transaction. Note that the set of participants in a state does not need to be the same as the set of required signers. A transaction can contain multiple states, each possibly with a different set of participants.

The lack of a global distributed ledger that would record all transactions that occurred in the network and would be accessible by every node stems from using point-to-point communication instead of a global broadcast (e.g. gossip protocol in Bitcoin and Ethereum) [17]. The communication occurs via TLS-encrypted messages sent over AMQP/1.0 [13]. Corda uses flows as an abstraction for the actual communication between nodes. A flow is a sequence of steps nodes execute in order to achieve a ledger update. Figure 2.5 details a flow taken to to execute a transaction. The initiator first retrieves documents from its local storage called the vault. It builds a transaction which is verified by the referenced contract(s). After the initiator signs the transaction, it is sent over to the responder who also verifies the transaction, signs it, and sends it back to the initiator. After the transaction has been verified and signed by both parties, it is checked by notaries for double-spending. If no double-spending attempt occurs, it is committed to the ledger.

A transaction is only valid if it is (1) contractually valid, (2) signed by the required parties, and (3) does not contain double-spends. Contractual validity is accomplished by referencing each state in the transaction with a contract whose validation logic is

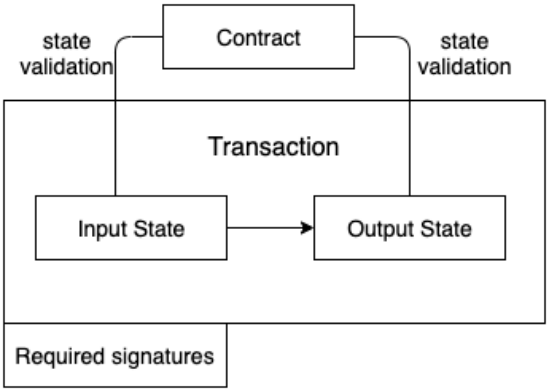

Figure 2.4: Structure of a simple transaction containing one input state and output state. The input state gets spent when committing the transaction to the ledger. The output state is produced by the transaction. The referenced contract validates the input and output state. The box in the left bottom corner includes parties that need to sign the transaction. Figure inspired by `https://docs.corda.net/docs/corda-os/4.7/key-concepts-contracts.html`

programmed according to application specifics. Figure 2.5 shows how both parties need to sign a transaction before it is sent to the notary. The notaries ensure prevention of double-spending on Corda network. Every transaction has to pass through a notary cluster that checks whether the input states in the transaction have already been spent. If they have been spent, the transaction becomes invalid and a double-spending attempt is recognized together with the identities of nodes that made this attempt. Otherwise, the transaction can be committed to the ledger. The type of consensus algorithm a notary runs can be decided upon based on the nature of parties running the notary cluster. This property is called pluggable consensus [2]. In case the notary cluster is run by a trusted party, a high-speed consensus algorithm such as RAFT can be used. If the notary cluster consists of low-trusted parties, a Byzantine Fault Tolerant (BFT) algorithm can be chosen instead.

There are two types of notaries - validating and non-validating [14]. The non-validating notaries only prevent double-spending of the input states and have limited access to the contents of the transaction. This access includes references to the input states that are required to prevent the double-spending. The content of the transaction is hidden ensuring no confidential information the parties deal with in the transaction is exposed to the notary. The validating notaries have a full access to the contents of a transaction, including fully visible input and output states, commands, attachments and signatures. This enables them to validate the transaction more thoroughly, including contractual validation, before committing the transaction to the ledger.

The disadvantage of non-validating notaries is that it does not prevent a denial-of-state attack. This can occur when a state is shared between multiple parties and a single malicious party decides to consume this state selfishly. The non-validating notaries do not have access to input states and contracts referenced by the states. Hence, the non-validating notary cannot determine if an input state is being selfishly consumed by a malicious party. If this occurs, the input state will be consumed and become unusable

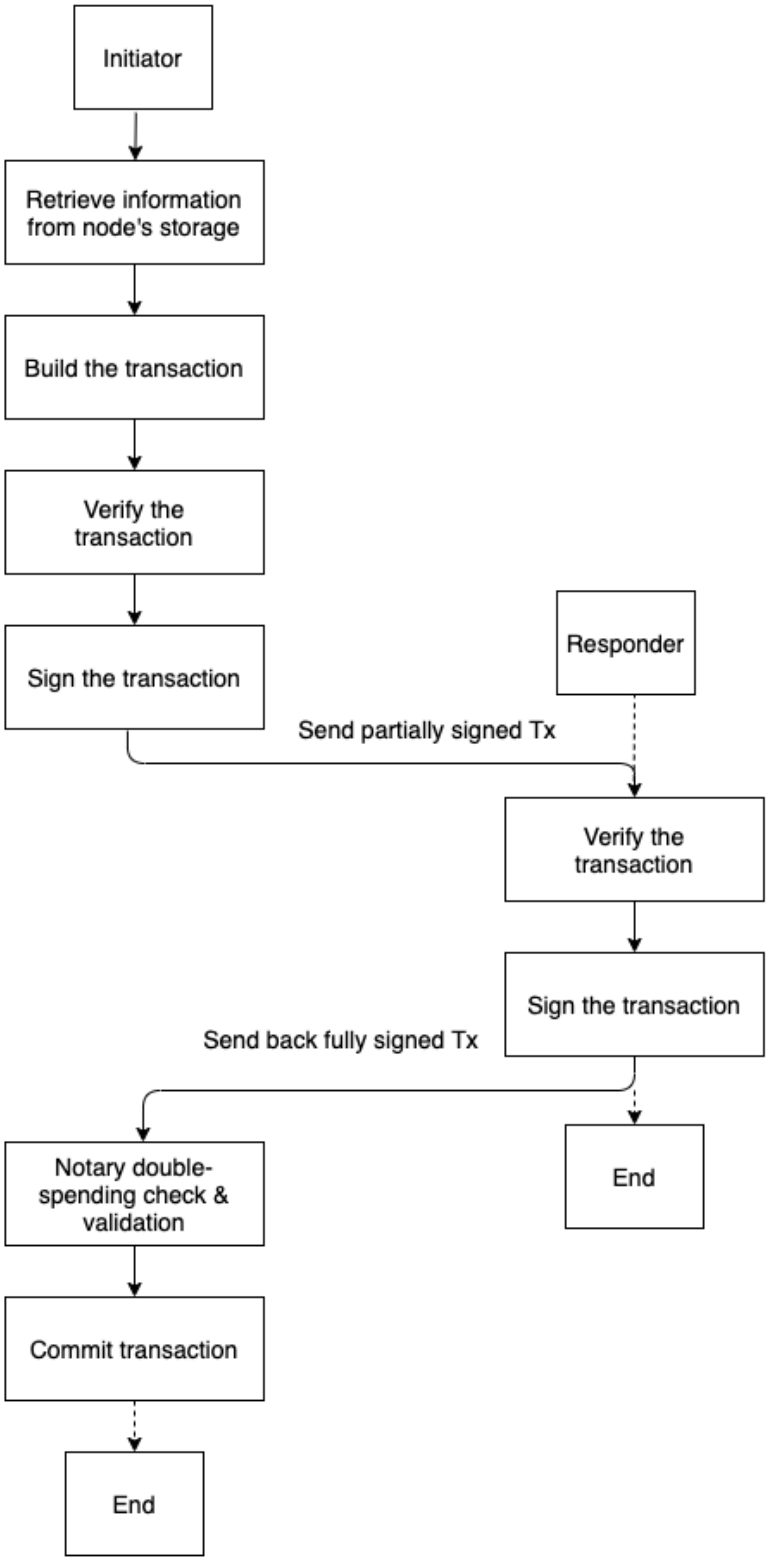

Figure 2.5: An example flow between two parties - Initiator and Responder. The initiator builds, verifies, and signs a transaction. The transaction is then sent to the Responder who verifies it, signs it, and sends it back. The fully signed transaction is checked for a possible double-spending attempt and if no attempt was recognized, it is committed to the ledger. Figure insipired by `https://docs.corda.net/docs/corda-os/4.7/key-concepts-flows.html`

for other parties, creating a denial of state if they collaboratively attempt to spend the state in the future. However, note that all notaries, including the non-validating ones, have access to the identity of the nodes who proposed a transaction. This provides for the conflict to be resolved off-chain as the notaries would have a real identity of the malicious node that selfishly consumed a state.

Corda is a permissioned network only accessible by parties who obtained a permission from the doorman - a party that has authority over who obtains a permission to join the private network. Corda uses X.509 certificate for nodes in the network [7]. Under this certificate, before a node can join the network, it needs to reveal its X.500 name. This includes the name of the person (in case of a customer) or organization (in case of a financial institution), locality (city where the customer or the financial institution operates in), and country of the respective location. This identity is visible to other nodes that have been permitted to join the network.

Key takeaways:

1. Corda does not have a global distributed ledger that would store all transactions that occurred in the network and would be accessible by each node in the network.
2. Corda uses point-to-point communication between nodes instead of globally broadcasting transactions to other peers in the network.
3. Nodes in Corda network have a public identity that is shared with other nodes in the network.
4. A party can join a Corda network only after it has been given a permission from the doorman service.

2.2 Motivation for using Corda for KYC

Corda blockchain was selected as a suitable candidate of distributed ledger technology for the KYC process due to its enhanced privacy measures that stem from avoiding public broadcast of information and the lack of a central distributed ledger. Corda uses point-to-point communication via private ledgers that ensure messages and transactions are only exchanged between the sender and intended recipients. A private ledger in this context presents a chain of transactions that is only accessible by the parties that were involved in signing those transactions.

Note that Corda is a private network accessible only by parties that obtain a permission from the doorman. In this scenario, the doorman could either be a trusted third party (i.e. the regulator or an independent company) or a consortium of financial institutions operating on the network. Hence, due to this design of Corda network, the second privacy condition is automatically fulfilled. The other privacy conditions are discussed in detail in sections 3.3 and 4.3.

2.3 Drawbacks of using Corda for KYC

The first potential downside of Corda is that every node, including all financial institutions and all customers, reveal their real identity in the network. The real identity in this context refers to using a valid instance of `CordaX500Name` class. This contains the name of the person (in case of a customer) or organization (in case of a financial institution), locality (city where the customer or the financial institution operates in), and country of the respective location. It does not reveal any further information about the customer or the financial institution. Note that this does not breach the privacy requirements outlined in section 1.4.

In our previous work, we created a pseudonymized digital profile of each customer on the distributed ledger that would include the following key properties: the result of the KYC process that financial institutions that verified the customer came to, the cost of onboarding the process, the number of financial institutions the customer operates with, and the blockchain addresses of accounts the financial institutions used to onboard the customer. When a financial institution wanted to onboard a customer who already operated with other FIs, this profile provided for effective sharing of the work behind KYC and costs associated with the process - the ultimate benefit of reinventing KYC process with the distributed ledger technology. Figure 2.6 shows a customer who operates with a single FI (Bank A) and approaches another FI (Bank B). Under the assumption that Bank B does not need to repeat the KYC process, which is randomly decided with a certain non-zero probability, Bank B is required to pay its fair share of the KYC cost via the distributed ledger. In a scenario when the customer only operates with Bank A, this share would be $c/2$, where c is the cost Bank A faced when onboarding the client. After this fee is paid, Bank B can onboard the customer without any further complications. The second downside is that this simple cost-sharing between financial institutions cannot be easily replicated in Corda network without compromising the privacy of customers and financial institutions. The outlined cost-sharing between financial institutions operates with the assumption that financial institutions would use pseudonymous blockchain accounts to onboard their customers with a one-to-one relationship between these accounts and their customers. This could hypothetically conceal the real identity of the financial institutions on the ledger. However, this has not been thoroughly investigated by the current literature and remains an open problem. In Corda network, each node has a well-known identity. Hence, referring to figure 2.6, when Bank B would pay the fair share of the KYC cost ($c/2$) to Bank A, both financial institutions would reveal to each other that they operate with the particular customer. This would breach the fourth and partially the third privacy condition outlined in section 1.4.

The third flaw is that we cannot present a global digital identity of the customer due to the lack of a central distributed ledger in the network. The pseudonymous digital profile of customer outlined in figure 2.6 was stored on the central distributed ledger. This profile was used when a financial institution wanted to onboard a new customer and it needed to retrieve the result of the KYC process, find out if it needs to repeat the process, and what fee it would have to pay to fairly distribute the onboarding cost between the financial institutions. We go around this by preserving the customer profile

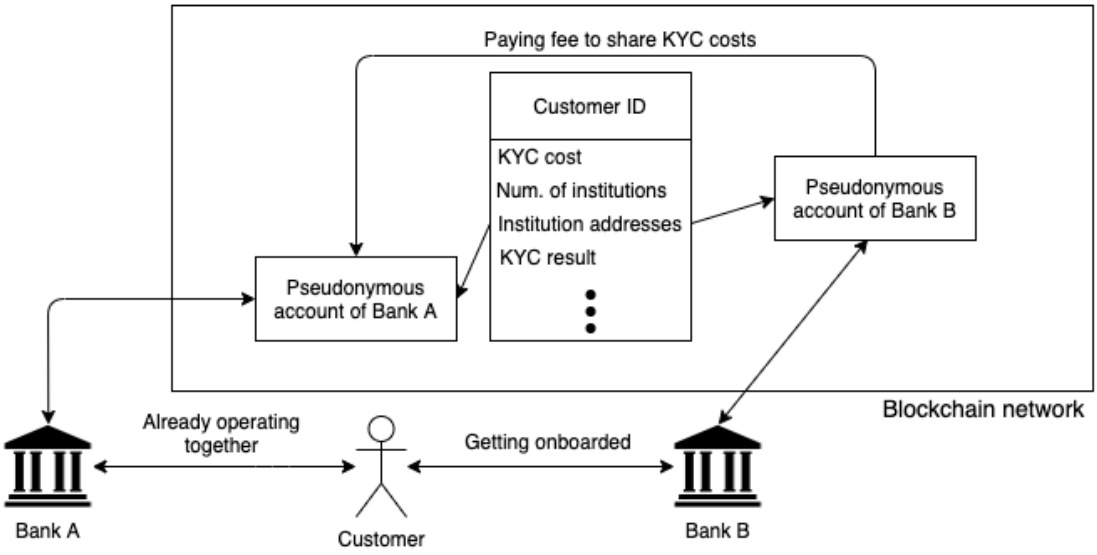

Figure 2.6: Simulation of approaching a new financial institution and cost-sharing mechanism via distributed ledger technology with a central blockchain. This simulation applies to our previous work. Customer operates with `Bank A` and approaches `Bank B`. `Bank B` retrieves the pseudonymous customer's digital identity from the central blockchain and pays an appropriate fee from its pseudonymous blockchain account to a pseudonymous account of `Bank A`. The identity of the two financial institutions is hidden behind their pseudonyms and is not directly revealed. After the fee is paid, `Bank B` can start operating with the customer.

via transaction states as we later explain. Given that this profile was only pseudonymous and accessible by any party on the network, it raised a concern whether there could be a leakage of confidential information about the customer and it vastly limited the scope of information that could stored under this profile. This is overcome by using the transaction states that are only accessible by financial institutions the customer sends them to.

We propose two different designs of Corda-based systems for the KYC process. The first one tackles the drawbacks of Corda network at the cost of increased involvement of a trusted third party. The second works in a fully decentralized manner but partially compromises the privacy conditions outlined in 1.4.

Chapter 3

Centralized Corda-based KYC system

This Corda-based KYC system is called centralized because it relies on a trusted third party (TTP) to (1) proportionately distribute the cost incurred by onboarding a customer between all financial institutions that operate with the customer and (2) provide notary service. The KYC process itself, however, is executed by the financial institutions and does not require any involvement of the third party. The distributed ledger technology serves as a medium for sharing the work and cost associated with onboarding new customers. A financial institution might benefit from using the distributed ledger technology by being able to onboard a new client without executing KYC because the client has been previously verified by other financial institutions. In such a scenario, it pays a fee for this service to the TTP that fairly distributes this fee between financial institutions that executed KYC for this customer.

3.1 Design

When a customer approaches the first financial institution where they would like to open a bank account, the customer needs to provide the FI with documents required for the KYC onboarding process. Due to the point-to-point communication in Corda network, the customer documents are sent directly via the distributed ledger between the financial institution and the customer. Non-validating notaries are used as they do not have access to the states and attachments of a transaction they validate. The double-spending validation requires only references to the input states of the transaction, not the content of the states is not revealed. The financial institution stores the customer documents in its vault, which is a local storage on Corda network only accessible by the institution. The financial institution may also store these documents off the ledger in its own database. This step is voluntary.

Upon retrieving the customer documents, the FI initiates the KYC process. The customer identification process (i.e. assessing customer's identity) can be accomplished by checking the attached documents sent by the customer. However, certain aspects of KYC, such as enhanced due diligence controls (e.g. watch list scan, negative news search, politically-exposed-person check, etc.) still need to be assessed off-ledger. Once finished, the FI either accepts or rejects the customer. This process is outlined in

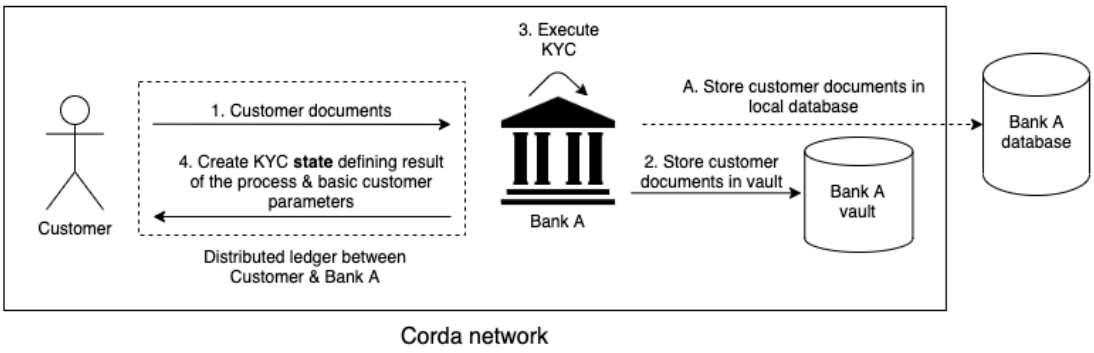

Figure 3.1: A customer approaches the first financial institution and gets onboarded by the institution via the centralized Corda-based KYC system.

figure 3.1.

In our previous work, a financial institution would create a pseudonymous digital profile of the customer on the distributed ledger. This enables the first financial institution (`Bank A`) to share the onboarding costs with any other financial institution (`Bank X`) that starts to operate with the customer in the future. This digital profile is required to confirm that the customer has already been onboarded by a financial institution (Bank A). Without it, Bank X would not know and could not confirm that the customer has been onboarded and verified by another financial institution. Corda network lacks a centralized distributed ledger that would be accessible by other parties operating on the network. Hence, the customer's digital profile is shared via states that serve as inputs and outputs of transactions.

When the same customer approaches a new financial institution (Bank X), the customer presents their documents together with a state that contains the following information: (1) result of the previous onboarding process, (2) whether Bank X needs to repeat the KYC process, and (3) what fee it has to pay to proportionately share the KYC cost with the institution that executed the KYC process (Bank A). Note that the customer at this point only operates with Bank A, which has fully incurred the onboarding cost. The repetition of the KYC process is randomly determined with probability p. This probability is set by the first financial institution the customer operates with. If Bank X does not need to repeat the KYC process, it only has to pay a fair share of the onboarding process (i.e. 50% of the cost incurred by Bank A) and can start operating with the customer immediately. Otherwise, it has to independently repeat the process, in which case it is not required to pay any fee.

If Bank X does not repeat the KYC process and only pays a fee to equally share the onboarding cost with Bank A, Bank X me cannot pay the fee directly to Bank A via Corda network. Having a transaction with another node on the network implies knowing the identity of that node. However, the fourth privacy condition requires that only the customer knows that he/she operates with both institutions. Bank X is only allowed to know that there exists a financial institution that verified this client and it is now going to share the cost of verifying this client with the institution. Bank A receives a fee that in the transaction specifies the customer's ID, so that the institution

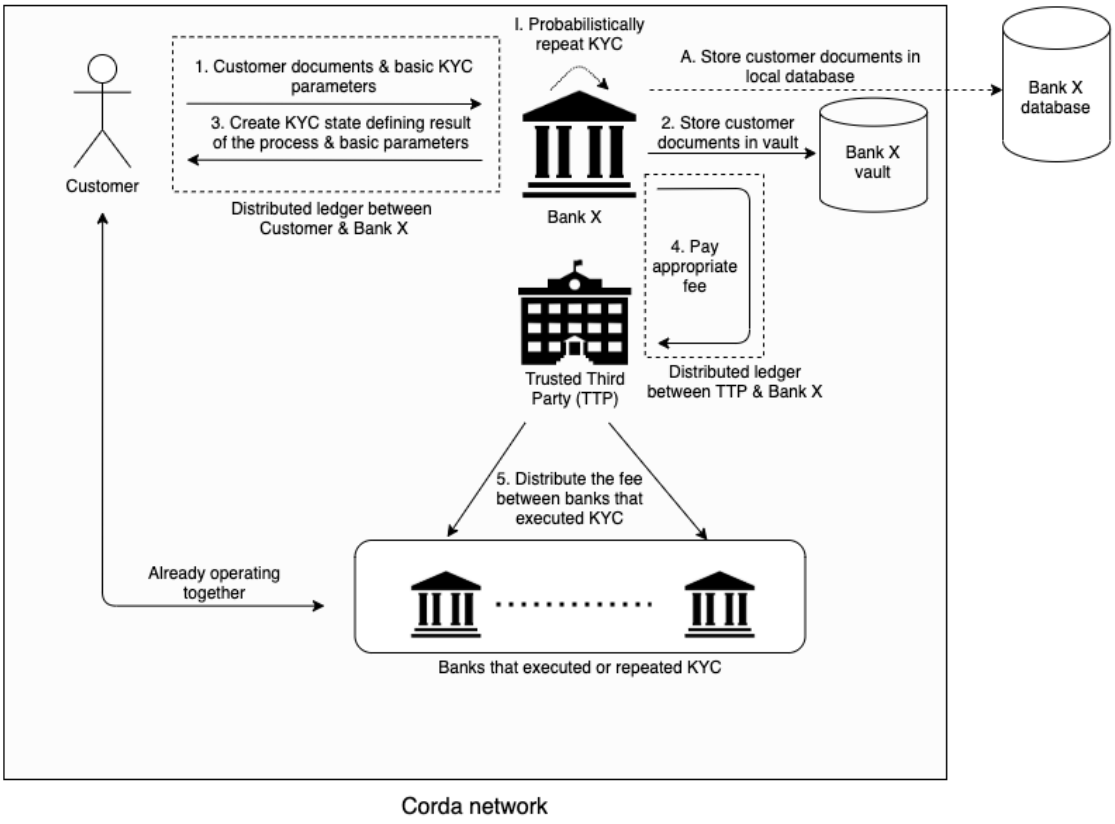

Figure 3.2: The customer approaches another financial institution. If the institution doesn't need to repeat the KYC process, it pays a fee to proportionately share the onboarding cost. This fee is paid to the TTP that distributes it between the financial institutions that executed the KYC.

would know which customer of its customers now starts operating with another financial institution, but does not know the identity of this FI. Hence, Bank X sends the fee to a trusted third party (TTP), which could be the regulator or a private company. The TTP would then send the fee to Bank A. the recipient financial institutions. Figure 3.2 details this process when the customer operates with and has been verified by multiple financial institutions. In this scenario, the size of fee Bank X pays is specified by equation 1.1 and each financial institution that executed or repeated the KYC process for the customer receives an equal share of the fee specified by equation 1.2.

3.2 Implementation

KYC is a business process that might be fairly time consuming. When a node in Corda network has a flow in a suspended state, it cannot be updated to a possibly new release of Corda. Presenting long business processes in a single flow is therefore undesirable

and it is recommended to break a longer flow into a series of shorter flows instead[1]. In our implementation, we implement the following flows:

- `FirstRequestFlow` - initiated by a customer and used when the customer would like to open a bank account at a financial institution of their choice.
- `AccomplishKYCFlow` - initiated by a financial institution and used when the financial institution executed KYC and wants to store the result in the customer's profile.
- `RequestFlow` - initiated by a customer and used when a customer would like to start operating with a financial institution and has already been onboarded by at least one other FI via Corda.
- `RepeatKYCFlow` - initiated by a financial institution and used when a financial institution repeated the KYC process.
- `UpdateKYCFlow` - initiated by a financial institution and used when the financial institution needs to update the KYC process (e.g. due to a regulatory change).
- `DownloadDocumentsFlow` - initiated by a financial institution to download documents provided by the customer for the KYC onboarding process.
- `ShareCostFlow` - initiated by the trusted third party to distribute a fee paid by a financial institution that can start operating with a customer without executing KYC between financial institutions that executed or repeated KYC for the customer.

`FirstRequestFlow`, outlined in figure 3.3, is initiated by a customer when he/she would like to open a bank account at a financial institution and has not yet been onboarded by another financial institution via the Corda-based KYC system. This either means that the customer approaches the very first financial institution where they would like to open an account, or the customer is already operating with other financial institutions, but none of these use the Corda-based KYC system to onboard their customers. For instance, they might be onboarding their customers the traditional way (see Fig.1.1) that does not take advantage of distributed ledger technology.

`FirstRequestFlow` simulates step 1 in figure 3.1. To realistically replicate the KYC process which customer begins by approaching a financial institution (either physically or via a mobile/web application), this flow is also started by the customer. The customer uploads their documents and requests a customer ID from the financial institution. This ID serves as a unique identifier of the customer on Corda network, but is only accessible to financial institutions the customer approaches. Upon receiving the ID from the financial institution, the customer builds transaction `Tx1`, verifies it via a referenced smart contract(s), signs it, and sends it to the financial institution. The financial institution also verifies and signs the transaction, and sends it back to the customer. The customer sends the transaction to a notary who validates the transaction input states to prevent double-spending and commits it to the ledger if no double-spending attempt is detected. Note that once the flow is initiated, the process of build-

[1] `https://docs.corda.net/docs/corda-os/4.6/flow-state-machines.html`

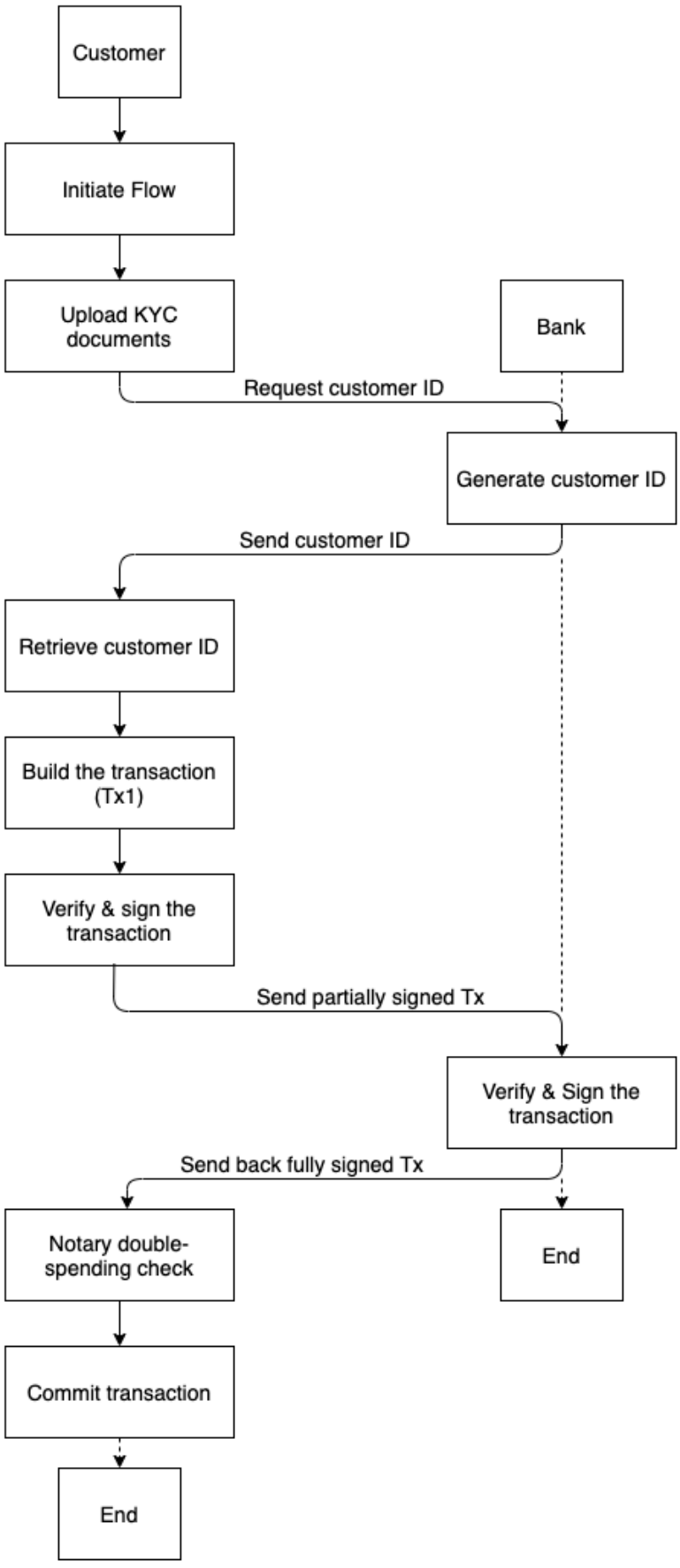

Figure 3.3: `FirstRequestFlow` initiated by the customer that creates the customer's ID and produces `Tx1`.

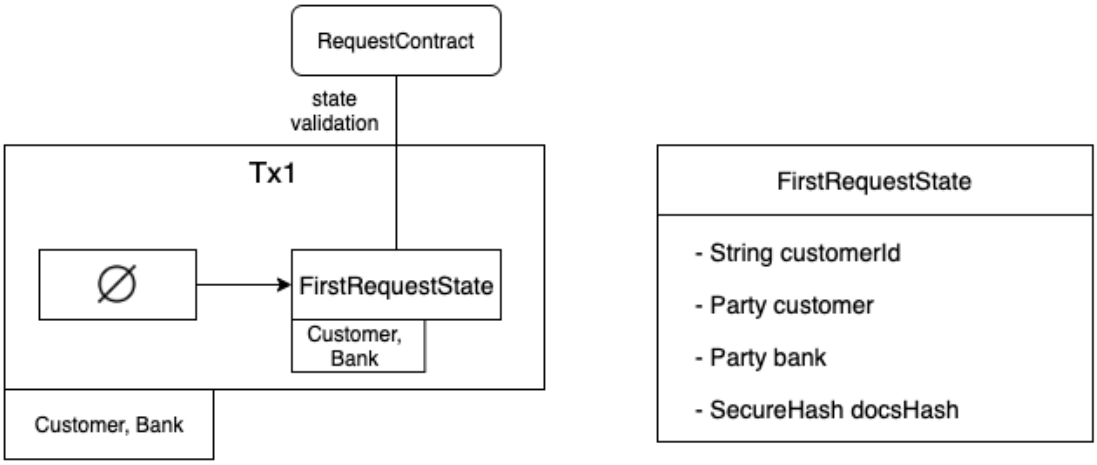

Figure 3.4: A detailed view of `Tx1` and `FirstRequestState`.

ing, verifying, signing and sending a transaction is automated by the flow logic and smart contracts and does not require further involvement from either side.

Transaction `Tx1` is detailed in figure 3.4. Note that the transaction has no input states and so double-spending cannot occur. The transaction has to be signed by both the customer and the financial institution. The referenced `RequestContract` ensures that a malicious customer cannot produce arbitrary output states and that the transaction has a single output state of type `FirstRequestState`. This output state stores identities of the customer and of the financial institution, the customer's ID, the hash of the customer documents, and a unique identifier of the state. Figure 3.4 illustrates these private state properties; each property has an associated public getter method. Note that the customer's ID (`customerId`) is used to uniquely represent the customer, while the unique state identifier (`stateId`) uniquely represents the state.

Each state also needs to overwrite a `getParticipants` method that returns the participants of the state. These participants are parties that should be notified when the state is created or consumed. The participants of this state are the financial institution and the customer. The set of participants does not necessarily equal the set of required signers for a transaction. A transaction might have multiple input and/or output states, each with a different set of participants. Assuming a transaction gets signed by all required signers, a required signer is a party that will have the committed transaction on a distributed ledger that is shared with other required signers of the transaction. This distributed ledger will have stored any previous and future transactions that were signed by exactly the same set of signing parties.

In order for Bank A to obtain access to the documents sent by the customer, it needs to download them into its vault (internal storage of the financial institution on the Corda network). To accomplish this, Bank A executes `DownloadDocumentsFlow`. This is a simple flow that neither produces a transaction, nor consumes or produces any states. It is not sent over to any party and does not require any signatures. This presents step 2 in diagram 3.1.

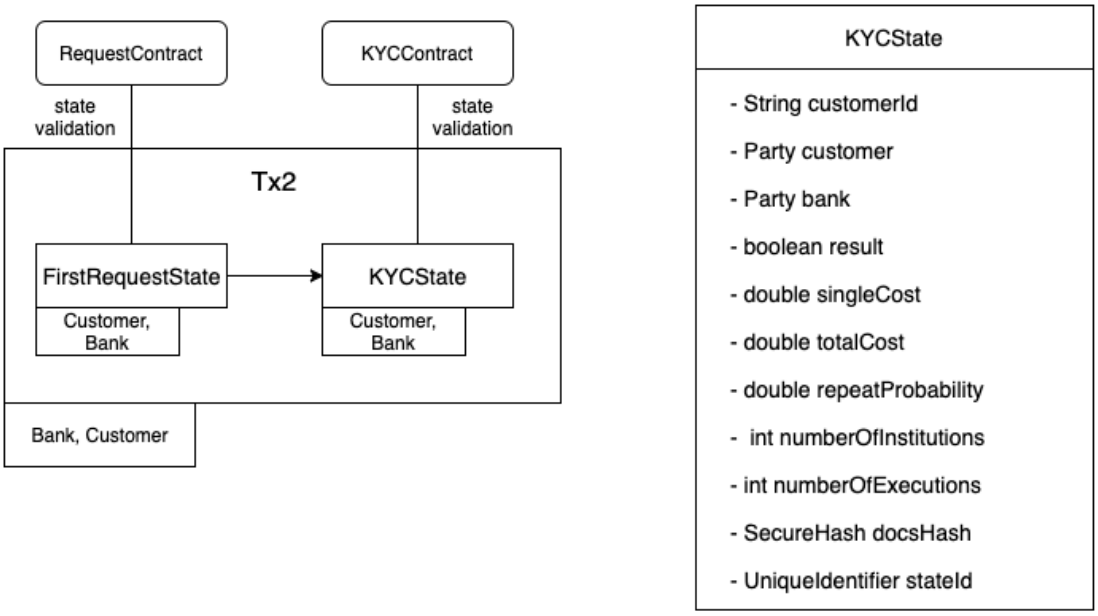

Figure 3.5: A detailed view of `Tx2` and `KYCState`.

Upon retrieving the documents, Bank A performs the KYC process, simulating step 3 from figure 3.1. The customer identification part of KYC, which confirms the customer's identity, can be checked directly from the documents supplied by the customer via the distributed. However, certain aspects of the enhanced due diligence, including watchlist scan, negative news search, politically exposed person scan, etc. currently cannot be directly executed on the ledger as they require use of a separate database (e.g. a database containing politically exposed people). This process could be automated by integrating a Corda application (CorDapp) presenting this KYC system with existing systems of the financial institutions but this goes beyond the scope of this work. Our implementation delivers sharing of the KYC documents that can be retrieved by the financial institution via Corda network and directly used for customer identification program.

After the financial institution executed the KYC process, it can initiate `AccomplishKYCFlow` to store the result of this process on the ledger shared with the customer. This flow is detailed in figure 3.6 presents step 4 from figure 3.1. The financial institution retrieves from its vault request state for the particular customer based on the `stateId` property. It builds transaction `Tx2`, detailed in figure 3.5, that consumes this state and produces a `KYCState`. `KYCState` state includes information about the result of the KYC onboarding process (i.e. whether the financial institution accepts or rejects the customer), the cost of this process, and the probability of repeating this process by a future financial institution that the client would like to be operate with. This information is required for a future financial institution the customer might decide to approach. The full list of the state's properties is shown in figure 3.5.

`Tx2` references `RequestContract` and `KYCContract`. These ensure that the transaction has an input and output state of appropriate type and cannot be misused by a malicious financial institution to pretend to onboard a customer who did not file such

a request. Once the transaction is built, verified via the smart contracts, signed by both parties, and checked by the notaries for double-spending, it is committed to the ledger. The notaries ensure that the financial institution does not attempt to use a single request state made by a customer in multiple transactions to come to multiple KYC states.

When the customer would like to open a bank account at another financial institution, he/she initiates `RequestFlow` outlined in figure 3.7. This flow simulates step 1 in figure 3.2. The flow is similar to `FirstRequestFlow` in a way that both fulfill the same purpose - a request made by a customer to be onboarded by a customer-selected financial institution. The crucial difference between the flows is that the customer has already been onboarded by at least one financial institution. This implies that the customer already has a digital profile and that the KYC process has been executed for the customer. Hence, the new financial institution (Bank X) has to only conditionally repeat the KYC process. Whether Bank X repeats the process is dependent on a random number generator implemented on the customer side to ensure the financial institution has no impact over the outcome of this. If Bank X does not need to repeat the KYC process, it has to pay the fee specified by equation 1.1 to fairly distribute the cost associated with onboarding the customer between all financial institutions that operate with the customer.

`RequestFlow` introduces a new required signer - the trusted third party (TTP) that redistributes the fee paid by Bank X. The transaction built in this flow (`Tx3` detailed in figure 3.8) needs to be signed by three parties - the customer, who is the initiator of the flow, the financial institution, which responds to the customer's request, and a trusted third party (TTP) that later controls the proportional distribution of the total cost associated with onboarding the client between the financial institutions that operate with the client. The value of the fee Bank X should pay, that follows equation 1.1, is specified when the flow is initiated by the customer. The TTP is included in this transaction so that it sees the fee the financial institution should pay when the request is being made by the customer. This ensures the financial institution cannot process the request and attempt to pay a smaller fee to the TTP pretending this was the fee it should pay. The value of the fee is currently inputted by the customer. In an industry-applied setting, the CorDapp (i.e. Corda mobile application) would not enable the customer to set the value of this fee and would rather take it directly from the customer's storage. From implementation perspective, this would be easier to enforce on the application level as taking this value directly from a previous `KYCState`, from which the cost of onboarding the customer and thus the fee Bank X has to pay are derived, would breach the privacy by revealing identity of the previous financial institution to Bank X.

`Tx3` has two output states: `RequestState` and `MonetaryState`. `RequestState` is an augmented `FirstRequestState` class that with additional properties, including the result of the KYC process, the cost of the KYC process, and whether the financial institution needs to repeat the process. For a full list of the properties, see figure 3.9. `MonetaryState` presents a state with the specified monetary value that was given by the financial institution to the trusted third party. If a financial institution needs to repeat the KYC process, this value is set to zero. Otherwise, this value follows equation 1.1 and the financial institution can start operating with the customer upon completion of this flow. Note that Corda at this point does not operate with a currency that could

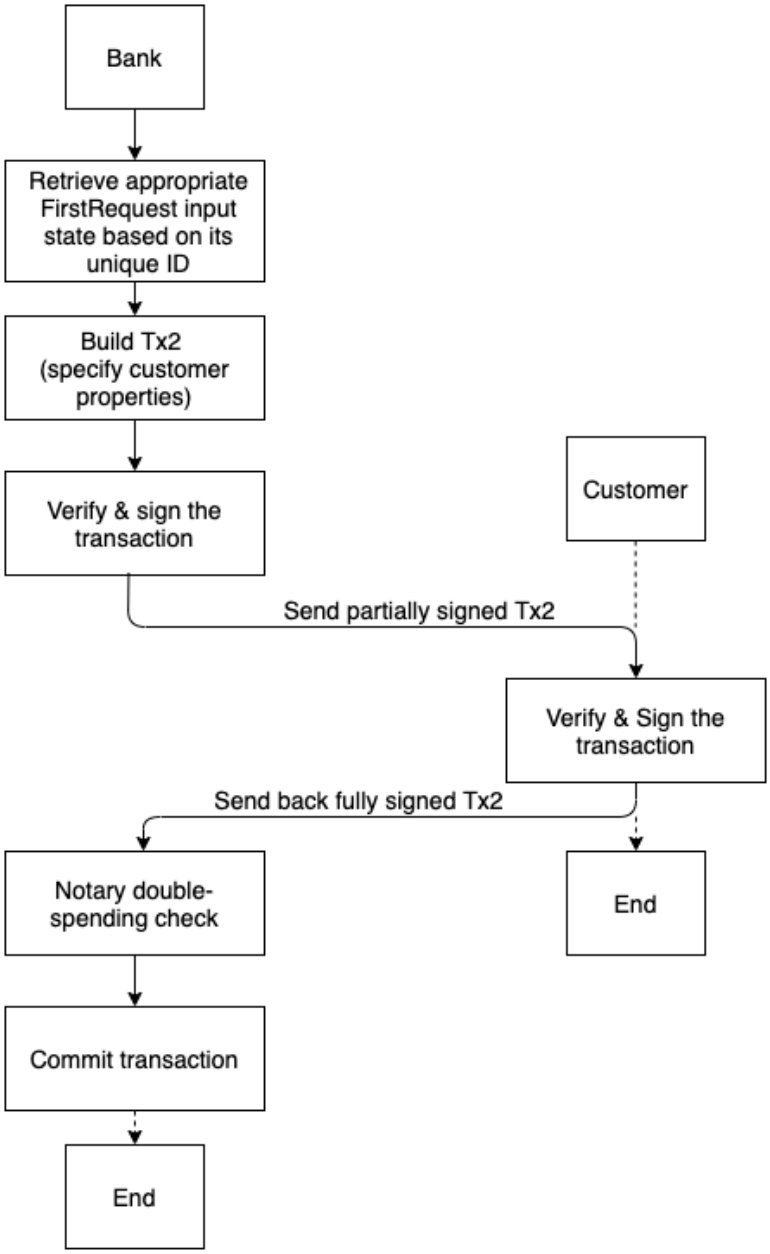

Figure 3.6: `AccomplishKYCFlow` initiated by the FI when it records the result of the KYC process on a ledger shared with the customer.

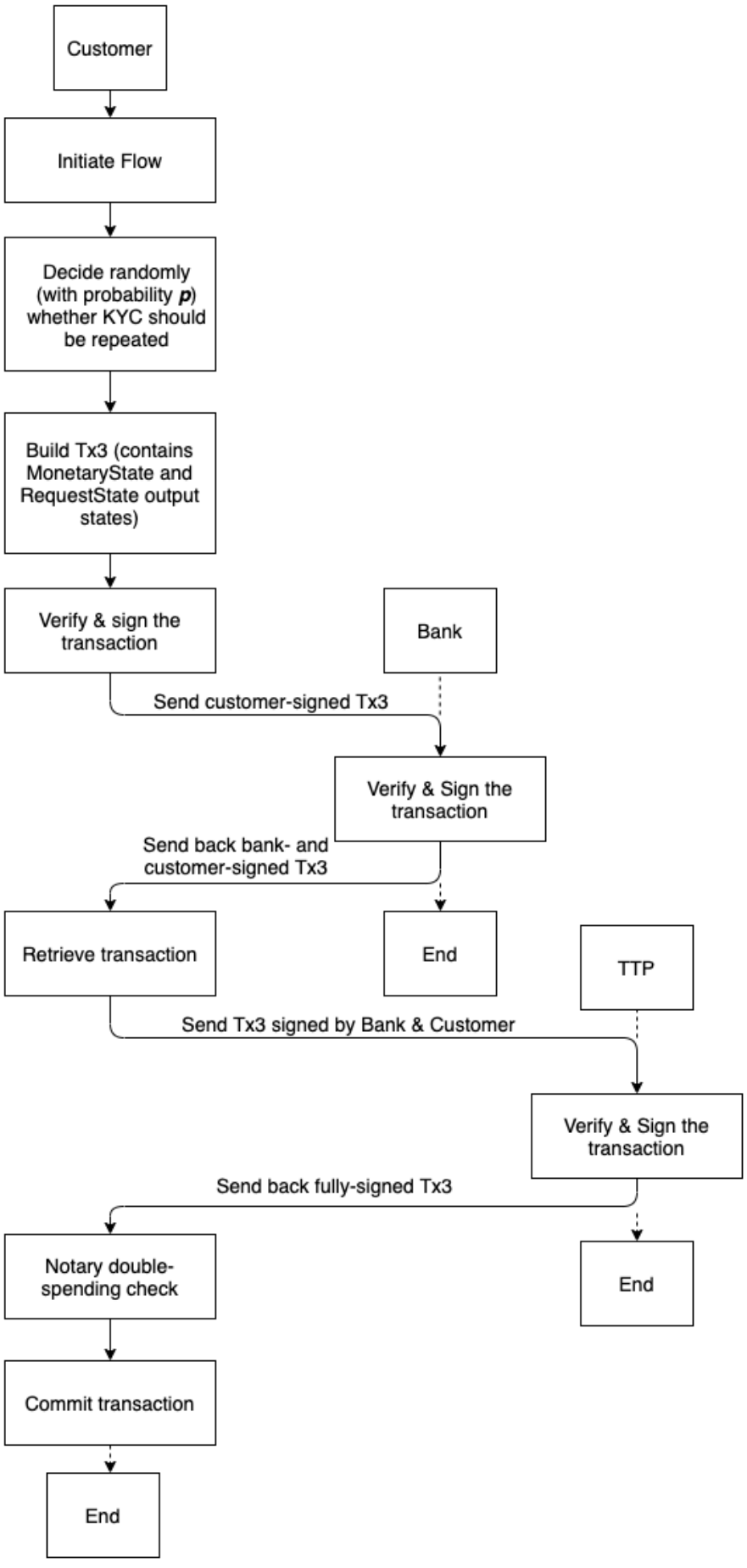

Figure 3.7: `RequestFlow` initiated by the customer when they would like to open a bank account at a financial institution after having been onboarded by at least one other FI.

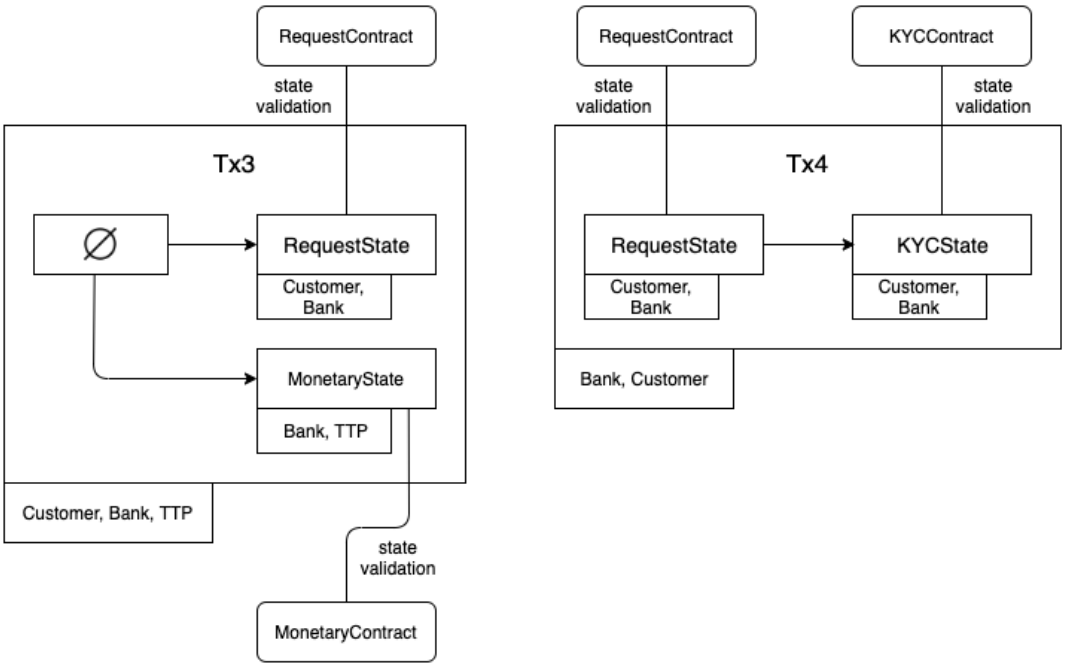

Figure 3.8: A detailed view of `Tx3` and `Tx4`.

be easily sent between the two parties. Hence, `MonetaryState` rather presents that the TTP is now an owner of a state that was issued by the financial institution and has a specified monetary value.

If Bank X has to repeat the KYC process, it executes the process based on documents the customer supplied in `RequestFlow`. When `Bank X` accomplishes this task, it initiates `RepeatKYCFlow` to record the result of this process on the ledger it shares with the customer. Repeating the KYC process and realizing `RepeatKYCFlow` present steps *I* and 3 in figure 3.2. `RepeatKYCFlow` has the same structure as `AccomplishKYCFlow` with the exception that `RepeatKYCFlow` produces a transaction of format `Tx4` illustrated in figure 3.8. `Tx4` consumes input state of type `RequestState`, as this flow is a continuation of a business process that initiated with the `RequestFlow`.

`ShareCostFlow` is initiated by the TTP to distribute the fee it received from Bank X between the financial institutions that executed or repeated the KYC process for the customer. It simulates step 5 in figure 3.2. The flow always has two required signers - the TTP and the financial institution that receives a part of the fee from the TTP. The flow has no input states and has a single output state of type `MonetaryState` which defines that the recipient financial institution is now the owner of a certain monetary value previously owned by the TTP.

The KYC process is a subject to regulations which might change over time. When this occurs, the process might have to be updated. Similarly, certain jurisdictions require financial institutions to revisit the process within a certain time period (e.g. every six months in China). For this purpose, we created `UpdateKYCFlow` that is used by a financial institution when it requires to update the process for a customer. The flow follows the structure of `AccomplishKYCFlow` and `RepeatKYCFlow` but produces a transaction

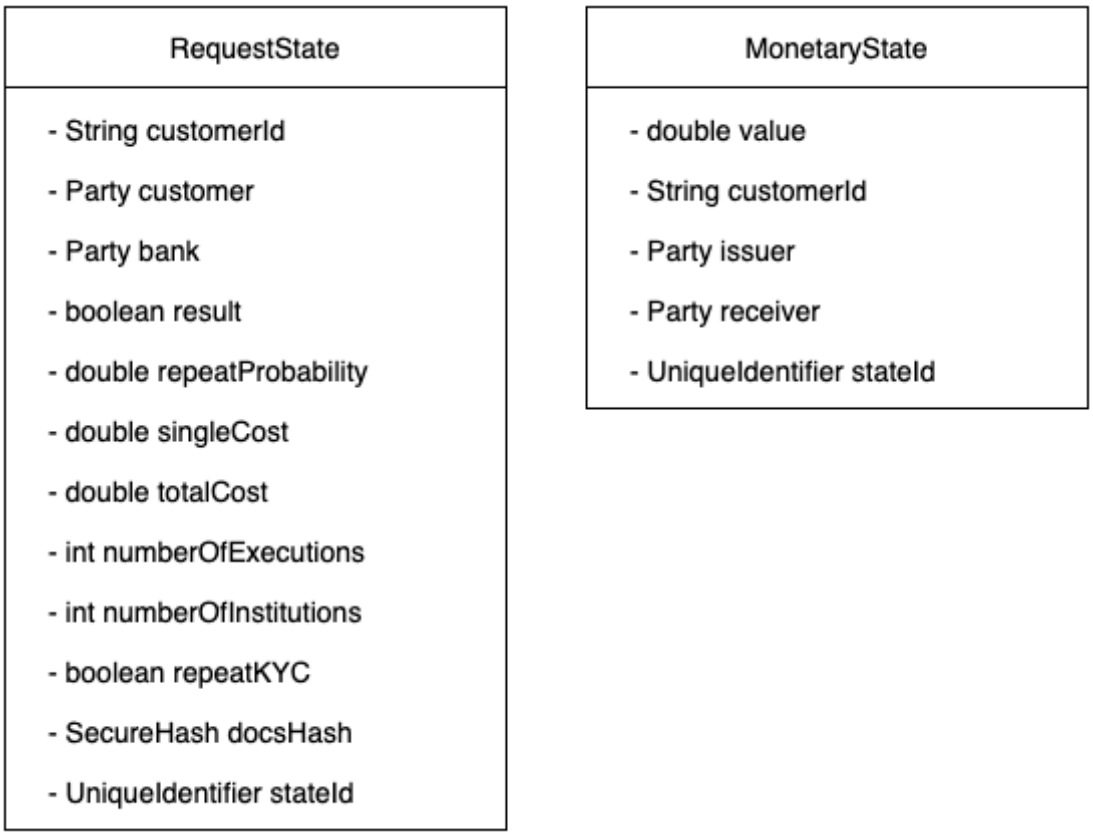

Figure 3.9: A detailed view of `RequestState` and `MonetaryState`.

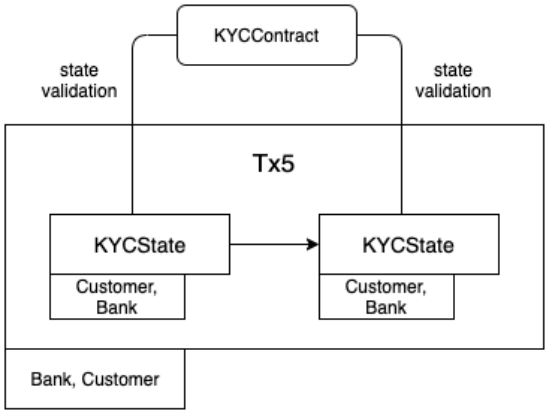

Figure 3.10: A detailed view of `Tx5` created in `UpdateKYCFlow`.

which consumes `KYCState` in the input and produces an updated version of the state in the output. This transaction (`Tx5`) is portrayed in figure 3.10. `Tx5` needs to be only signed by the financial institution as the process might be updated or refreshed without the awareness of the customer.

3.3 Evaluation

The major benefit of the centralized Corda-based KYC system is that it meets the privacy requirements specified in section 1.4. The first privacy condition, regarding the customer's documents and other confidential information, is preserved due to the point-to-point communication and the use of non-validating notaries. This private communication ensures that when a transaction between two parties is made, then any attachments, input and output states of this transaction are exclusively accessible by the two parties. The non-validating notaries only have access to the references of the input states to check for double spending. The actual input and output states, as well as the attachments, are hidden from the notaries.

The second privacy condition is fulfilled due to operating on a private blockchain network that has a doorman service. The third and fourth privacy conditions are met due to using the point-to-point communication between financial institutions and their customers, and due to letting a trusted third party distribute the fee between financial institutions that executed or repeated the KYC process for a customer. This prevents a financial institution from knowing which customers the other institutions operate with.

The robustness property is preserved by requiring financial institutions to probabilistically and independently repeat the KYC process. The digital profile of the customers is preserved by using the transaction states. The main downside is that this system requires a trusted third party to maintain proportionate sharing of KYC costs between the financial institutions.

The system uses non-validating notaries by default. This is sufficient to prevent double-spending. The denial-of-state attack should not be a real threat as the malicious party that exploits this vulnerability is traceable and would face legal repercussions. Optionally, if the TTP is fully trusted, validating notaries could be used instead. We do not recommend this, however, as this would give the notaries (i.e. the regulator or a private company) a full access to the input and output states of transactions, as well as access to any attachments. This level of access highly exposes confidential customer data that should only be given if the trusted third party is fully reliable.

Chapter 4

Decentralized Corda-based KYC system

This Corda-based KYC system is called decentralized because it requires no involvement of a trusted third party in the Corda network.

4.1 Design

The decentralized Corda-based KYC system is in many aspects the same as the centralized system outlined in section 3. Figure 3.1 that captures how a customer approaches the first financial institution where they would like to open a bank account equally applies to this decentralized system. The difference occurs when the customer approaches another financial institution (Bank X) and this institution needs to pay a fee to equally distribute the onboarding cost between all financial institutions operating with the customer.

Instead of paying the fee to an intermediary trusted third party, Bank X directly distributes the fee in an equal manner between the financial institutions that executed or repeated the KYC process for the customer. Subsequently, this maintains the onboarding cost fairly shared between all financial institutions operating with the customer, including those that did not execute or repeat the KYC, as we showed in our previous work. The fee Bank X pays is specified by equation 1.1. Each financial institution that executed or repeated the KYC process for the customer receives from Bank X an equal share of the fee specified by equation 1.2. Figure 4.1 outlines this process.

To provide a fully decentralized system, the notary service should be provided directly by the financial institutions. In order to avoid a breach of the third and fourth privacy conditions, the notary service for a transaction between a customer and a financial institution should be provided by the financial institution. The notary of a transaction between two financial institutions should be either (or both) of the two financial institutions. If this is fulfilled, the notaries can be validating. Otherwise, the notaries have to be non-validating.

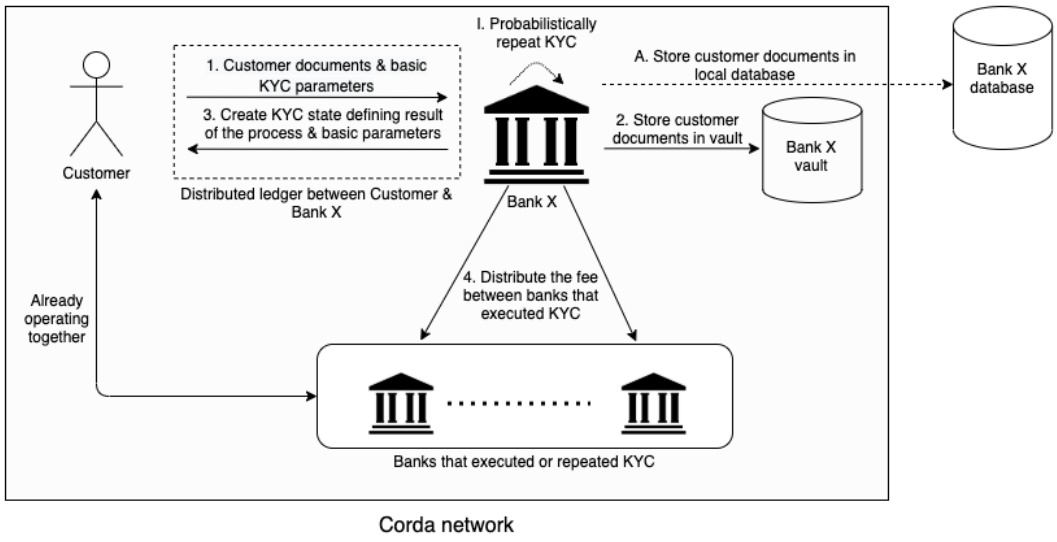

Figure 4.1: The customer approaches another financial institution - `Bank X` - that directly redistributes the fee between the financial institutions that executed the KYC to equally share the onboarding cost.

4.2 Implementation

The decentralized Corda-based KYC system consists of seven flows, six of which are the same as in the centralized system introduced in section 3. The first difference is in `DecentralizedRequestFlow`, which is an adapted version of `RequestFlow` that operates with a decentralized form of the request state (`DecentralizedRequestState`). The second difference is in how `ShareCostFlow` is applied to share the onboarding cost.

The process of approaching and getting onboarded by the first financial institution remains the same as in the centralized system from chapter 3. It is presented by `FirstRequestFlow` and `AccomplishKYCFlow` whose implementations remain identical.

A customer commences `DecentralizedRequestFlow` when they approach a financial institution (Bank X) where they would like to open a bank account after having been onboarded by at least one other FI via the decentralized Corda-based KYC system. This situation corresponds to step 1 in figure 4.1. This flow, depicted in figure 4.2, flows between the customer and the financial institution and builds transaction $\texttt{Tx3}_{Decentr.}$ illustrated in figure 4.3. This transaction consumes no input states as it simulates the request initiated by the customer without prior interaction with the financial institution. The transaction has to be signed by the customer who initiates this request flow and by the financial institution. Upon signing the transaction, the FI acknowledges the customer's request to be onboarded. The transaction has a single output state of type `DecentralizedRequestState`. The state is an augmented ver-

sion of `RequestState` that contains an additional key property - `X.500` identities of all financial institutions that previously executed or repeated the KYC process for the customer. The X.500 standard reveals an organisation's name, the city and country it is registered in. This list is imperative for sharing the total cost incurred by onboarding the customer. If it wasn't supplied and Bank X would not require to repeat the KYC process, it could onboard a client without paying any fee and doing any verification, benefiting from the system at the expense of financial institutions that previously verified the customer.

Note that unlike `Tx3` from figure 3.8, $\texttt{Tx3}_{Decentr.}$ does not contain any `MonetaryState` in the output. This is due to the following reason: assume the financial institution approached by the customer is called `Bank X` and it is the *Nth* institution the customer approached. If the probability of repeating the KYC process is p, then there are $K \approx 1 + (N-1) * p$ institutions that either executed or repeated the KYC process for this customer. The one comes from the first financial institution that unconditionally had to execute the KYC process, and the following $N-1$ institutions had to repeat it with probability p. We can see that K grows as the customer operates with more financial institutions and the probability of this process is set higher. Given that in this decentralized system Bank X needs to distribute the fee between all K financial institutions, it would need to create a transaction that would gather signatures from all K financial institutions. Obtaining a signature from each financial institution might become cumbersome when K is relatively large and slow down the customer's request to be onboarded. The monetary states were therefore not created directly in transaction $\texttt{Tx3}_{Decentr.}$.

To keep the system proportionate and distribute the fee, Bank X would initiate flow `ShareCostFlow`. This is a simple flow between Bank X and Bank J, where $J \in \{A, ..., K\}$, that is initiated by Bank X and presents a monetary transfer of certain value from Bank X to Bank J. The flow needs to be executed K time and each of the K financial institutions receives an amount of money prescribed by equation 1.2. When `ShareCostFlow` is executed between Bank X and Bank A, transaction `TxA` illustrated in figure 4.4 is committed to the ledger shared by the two financial institutions, signaling Bank A is now the owner of a monetary state of a certain value that was previously owned by Bank X. The same process applies to all institutions Bank J where $J \in \{A, ..., K\}$.

The use of an additional flow for the monetary transfers ensures that the request made by the customer, as well as the cost distribution between the financial institutions, cannot be put to a halt by having a single financial institution delayed with signing a transaction.

The rest of the flows and transactions work the same way as described in section 3.2. Indeed, the execution of the KYC process is otherwise identical and the only difference is the sharing of the total cost incurred by this process.

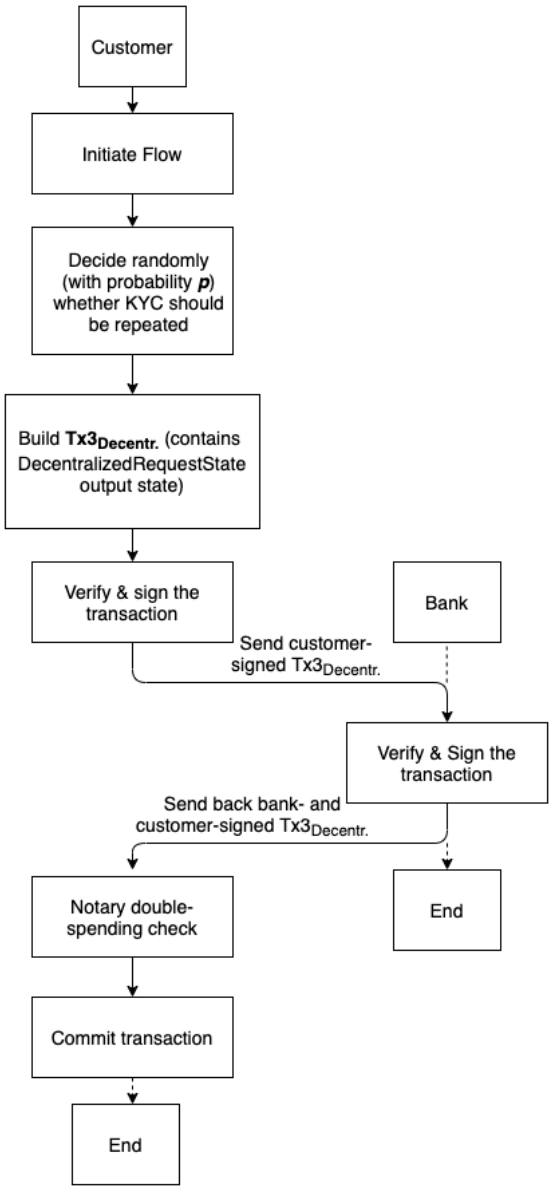

Figure 4.2: `DecentralizedRequestFlow` initiated by the customer when they would like to open a bank account at a financial instituion after having been onboarded by at least one other FI.

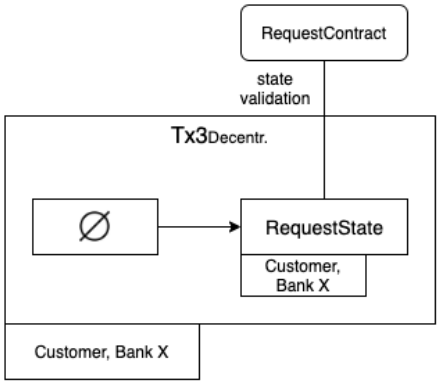

Figure 4.3: A detailed view of $\texttt{Tx3}_{Decentr.}$

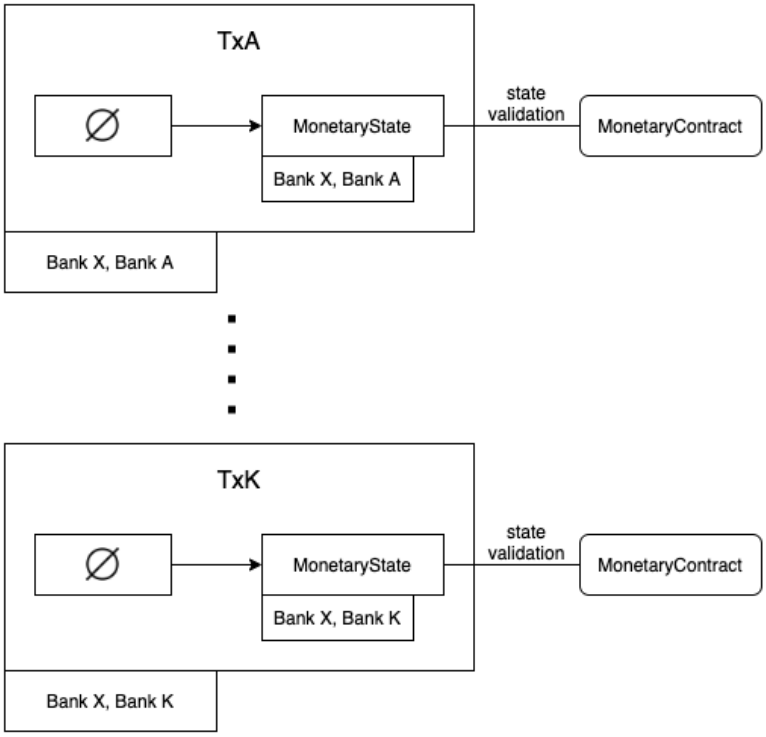

Figure 4.4: Transactions created by `Bank X` to share the cost of onboarding a customer.

	Bank A	Bank B	Bank C	Bank D	Bank E
Customer 1	-	-	**1**	-	-
Customer 2	**1**	2	5	**4**	3
Customer 3	-	-	**1**	-	-
Customer 4	-	**1**	2	**3**	4

Table 4.1: A matrix view of the customer and financial institutions they operate with. The numbers indicate in which order a customer started operating with a financial institution. The bold numbers signal that the financial institution executed or repeated KYC. The order was chosen arbitrarily.

4.3 Evaluation

Similar to the centralized system, the robustness property is retained by requiring financial institutions to probabilistically repeat the KYC process. The customer profile is sustained via the input and output states in transactions. The first privacy condition is fulfilled due to the point-to-point communication used for the exchange of documents and the result of the KYC process between a financial institution and a customer. The second privacy condition is satisfied due to the doorman. The doorman could be a consortium of financial institutions operating on the ledger to avoid involvement of a third party. A customer does not reveal to the doorman the financial institution(s) they would like to operate with, implying no violation of the privacy conditions we outlined.

The core benefit of this decentralized system is that it minimizes the involvement of a trusted third party (i.e. the regulator or an external company) in the network. Involving an external company or the regulator would incur additional cost and could make the process ponderous. The aim of the distributed ledger technology is to make KYC temporally and financially more efficient. Minimising the involvement of a TTP is therefore desirable. In this decentralized scenario, the TTP is no longer needed for the equal distribution of the costs associated with onboarding customers. The cost sharing would be accomplished by the financial institutions themselves by transferring the monetary states via the Corda network.

The downside of this decentralized system is a partial breach of the fourth, and a minimal breach of the third privacy conditions we outlined in section 1.4. This stems from sharing the customer onboarding cost via Corda network, where the identity of nodes is well known, directly by the financial institutions. When a financial institution pays a fee for onboarding a customer, it has to know the list of financial institutions that either executed or repeated the KYC process. This reveals that these financial institutions also operate with the same client. Thus, when a financial institution is about to onboard a new client, it knows of all other financial institutions that executed or repeated the KYC process for this customer.

To show how much information the financial institutions reveal between each other, consider the scenario in figure 4.5. Five financial institutions are operating with four customers. A blue line indicates that a customer operates with a financial institution and the institution either executed or repeated the KYC process for this customer. A black line indicates that a financial institution operates with a customer but neither

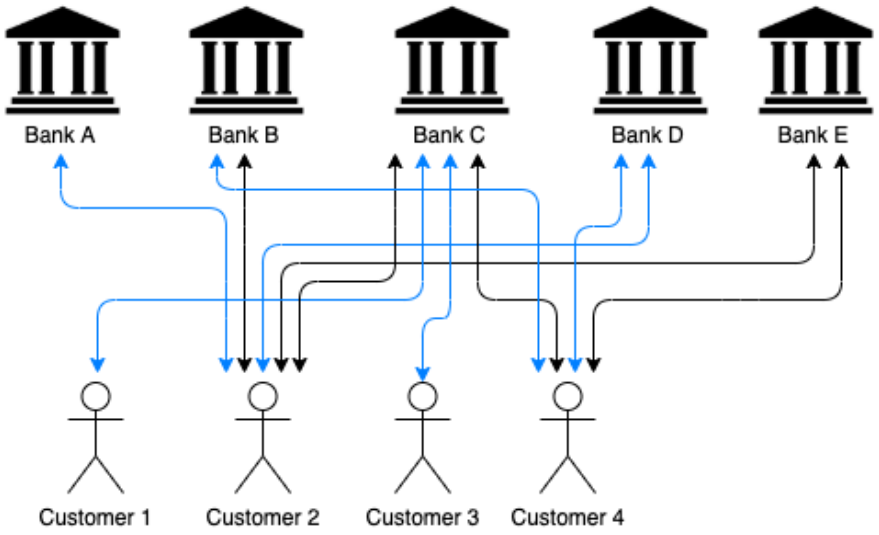

Figure 4.5: A situation depicting 5 financial institutions operating with 4 customers that were onboarded via the Decentralized Corda-based KYC system. A blue line between a financial institution and a customer indicates that the FI either executed or repeated the KYC process for the customer. A black line indicates that a financial institution did not have to execute or repeat the KYC process, but only had to pay a fee to start operating with the customer.

executed, nor repeated the KYC process for the customer. Such an institution had to pay a fee to operate with the customer. Assume that the order in which the financial institutions started operating with the customers is given by table 4.1. The following information is then available to the financial institutions:

- Bank A knows that Bank B, Bank C, and Bank E operate with Customer. Bank A knows this because it receives a fee from these FIs so that they can start operating with the customer. It does not know of Bank D because it does not receive a fee from this FI. Bank D has to repeat KYC for the customer which already incurs a cost for the institution.
- Bank B knows that (1) Bank A executed KYC for Customer 2 and (2) Bank C and Bank E operate with Customer 4. It knows (1) because Customer 2 supplies a list of financial institutions that verified the customer to Bank B. This list at this point only contained Bank A. It knows (2) because it received a fee from the FIs.
- Bank C knows that (1) Bank A executed KYC for Customer 2 and (2) Bank B executed KYC for Customer 4. It knows (1) because Customer 2 provides it with the list of FIs that executed/repeated KYC, which contained only Bank A when the customer approached Bank C. It knows (2) for the same reason - Customer 4 tells Bank C that he/she operates with Bank B so they can fairly share the onboarding cost.
- Bank D knows that (1) Bank A executed KYC for Customer 2, (2) Bank B executed KYC for Customer 4, and (3) Bank E operates with Customer 4. It

knows (1) and (2) because the customers share this information with the institution when they make a request to start operating with Bank D. It knows Bank E operates with Customer 4 because it receives a part of the onboarding cost from the institution.

- Bank E knows that (1) Bank A executed KYC for Customer 2 and (2) Bank B and Bank D executed and repeated KYC for Customer 4 respectively. Both facts are revealed to Bank E when it is approached by the customers so that Bank E can make a payment to the financial institutions to share the onboarding cost.

Let us define the following terminology - when we say that Bank I reveals its identity to Bank J with respect to Customer C, we mean that Bank J now knows that Bank I operates with Customer C. Assume Bank X is a financial institution approached by Customer C. The following rules then specify how, with respect to Customer C, the identity of Bank X is revealed to other financial institutions, and how other institutions' identities are revealed to Bank X, based on (1) when Bank X is approached by the customer and (2) whether it has to repeat KYC for the customer:

1. Bank X is the first financial institution approached by the customer. Bank X's identity is revealed to all financial institutions the customer will operate with in the future and Bank X knows identities of all these institutions.
2. Bank X repeats KYC for a customer when the customer has been verified by a non-empty set M of financial institutions, and operates with a non-empty set P of financial institutions. Note that M is a subset of P. The identity of Bank X is then revealed to each financial institution in set M, and to any financial institution that starts operating with the client in the future. Similarly, Bank X knows identities of all financial institutions in set M, and of any financial institutions the customer operates with in the future.
3. Bank X starts operating with the customer, without having to repeat KYC, when the customer has been verified by a non-empty set M of financial institutions, and operates with a non-empty set P of financial institutions. Note that M is a subset of P. The identity of Bank X is then revealed to all financial institutions in M and Bank X knows identities of all institutions in M. It neither knows of future institutions the customer operates with, nor the institutions know of Bank X.

We cannot thwart that the first financial institution executing KYC for a client will know of other institutions operating with the client and vice versa. However, we can minimize breach of the fourth privacy condition when a financial institution is in situation 2. or 3. by reducing set M. This can be achieved by decreasing the probability p of repeating the KYC process. Two extremes are: (1) $p = 0$ implying $M = \emptyset$, and (2) $p = 1$ implying $M = P$. The latter scenario completely breaches the fourth privacy condition. Note that the smaller the probability p is, the less robust and more brittle the KYC system becomes. Hence, this decentralized system introduces a *compromise* between robustness and breach of the fourth privacy condition.

The third privacy condition is fulfilled if a node on Corda network does not know any customers a financial institution operates with. Hiding this information from cus-

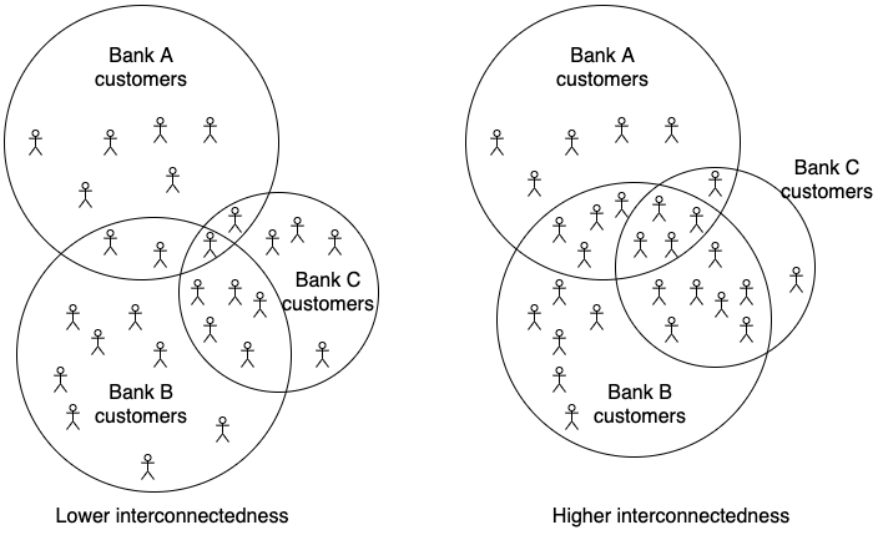

Figure 4.6

tomers in the network is trivial. A customer is always only involved with transactions he/she shares with a financial institution and has no way to learn about institutions other customers operate with. Hiding this information from other financial institutions is more involved. This condition can be fully met only if the fourth condition is fulfilled. If Bank X is a financial institution that executes/repeats KYC for a customer, all FIs operating with the customer in the future will know that Bank X operates with the customer too and vice versa. This means the financial institutions would know of at least one custoemr Bank X operates with. However, financial institutions operate with thousands to hundreds of millions of clients. We do not quantify the extent to which this condition is fulfilled, but for each financial institution, it intuitively depends on how many of its customers operate with multiple financial institutions, and how many institutions each such customer operates with, out of all of its customers.

Figure 4.6 gives an example of Corda network with three financial institutions and two different scenarios - the first has a lower interconnectedness of customers between the financial institutions and the second has a higher level of customer interconnectedness. In both scenarios, Bank B can identify a higher proportion of customers of Bank C than Bank A can identify. In the second scenario, the financial institutions can identify a higher proportion of the customers the other institutions operate with. We can see that breach in the third privacy condition is not constant across the institutions. The third privacy condition is fulfilled to a larger extent for Bank A than it is for Bank B. In the second scenario with higher customer interconnection, Bank C is almost entirely exposed to Bank B.

In our previous work, we outlined that financial institutions might feel skeptical about using a KYC system in which a financial institution does not know who executed the KYC process for a customer. Consider a scenario in which Bank X is a large financial institution with a strong and long reputation on the market. When Bank X wants to onboard a customer who operates with one other financial institution - Bank A - and Bank X does not need to repeat the KYC process, then it solely relies on how Bank A

executed the process for this customer. However, to meet the fourth privacy condition, Bank X does not know that Bank A was the financial institution that executed the KYC process for this customer. From the viewpoint of Bank X, it is only known that a single financial institution executed KYC for this customer. Our centralized system proposed in 3 that fulfills the privacy conditions also inherits this flaw.

The perk of breaching the fourth privacy condition and revealing the identity of financial institutions that previously executed or repeated KYC for a customer (Bank A) to a financial institution that is about to onboard this customer (Bank X), is that Bank X now knows how much it can rely on the result of this process. If Bank X deems Bank A untrustworthy, it can execute additional controls of the customer outside the distributed ledger system to mitigate the chance of onboarding a malicious customer. If Bank X considers Bank A a reliable financial institution, then it could take full advantage of the distributed system, share the onboarding cost with Bank A and start conducting business with the customer straight away.

Chapter 5

Final remarks

5.1 Future work

The first potential step in future work could be to build a full CorDapp with a graphical interface so that the financial institutions and customers would not need to interact via a command line.

The second field for future work is to fully utilise the customer's digital profile. The distributed ledger technology is currently used to share the work and costs associated with onboarding a new client. However, it is only partially used for the execution of the process itself. It can be used for customer identification program, but some other aspects of KYC, including the negative news check or the politically exposed person scan remain outside the ledger. The customer's digital profile includes the customer's documents used for the onboarding process, information on how many financial institutions operate with the customer, the probability of repeating KYC for the customer, etc. A full list of properties this digital profile involves is captured in figure 3.5. Th customer profile in our previous work could not store further information about the customer as it was pseudonymously presented on a central distributed ledger accessible by any party on the blockchain network. Had it stored whether the customer is a politically exposed person, the customer's country of taxation, residence and so forth, the pseudonymous profile could quickly reveal the real customer's identity to any financial institution and customer operating on the permissioned blockchain platform. Therefore, the depth of information captured by the customer profile had to be minimized.

The point-to-point communication on Corda network opens doors for sharing confidential information between two parties without revealing this information to anyone else. We broadened the customer profile with the customer documents. In the future, the customer profile could be further expanded with other risk assessment parameters, including geographical risk (e.g. customer's country of residence, operation, taxation, etc.), risk associated with account type, industry, occupation, watch list check, and many more [24]. Designing this profile should be accompanied with advice from professionals in the financial industry and legal representatives who have expertise in the procedural and legal background of this process. An enhanced customer profile would enable financial institutions to minimize the off-ledger work required in KYC and fully

utilize the distributed ledger technology for onboarding new customers. The platform could eventually be used for real-time sharing of suspicious customer behavior.

Should structured data be gathered about the customers, machine learning could be leveraged to predict the level of risk the customer presents at the onboarding state. The depth of the due diligence and enhanced due diligence a financial institution might carry out could be established based on this risk level. Using machine learning for credit risk analysis is now a common practice and has been considerably reviewed by existing literature [15, 1, 6, 20]. The use of machine learning for predicting the level of risk a customer presents at the onboarding stage is a new phenomenon and it is getting attention as rule-based models are getting replaced by models leveraging artificial intelligence [5, 27]. The literature exploring the use of machine learning for onboarding-risk assessment is limited and based on the current KYC scheme, which does not consider potential benefits the distributed ledger technology could bring. A significant benefit our proposed systems bring is the probabilistic repetition of the KYC process. The probability of repeating KYC for a customer could be established based on the level of risk the customer presents. A high-risk customer could be associated with a higher probability of repeating the KYC process. This would ensure that as high-risk customers would operate with more financial institutions, the customer would be independently verified several times, increasing the chances of exposing a potentially malicious customer.

5.2 Conclusion

A thorough KYC process is the first step to prevent anti-money laundering and financing of terrorist activities. The process has been under a lot of scrutiny with new regulations coming up every few years. Consequently, the process has become expensive and time-consuming, resulting in low customer satisfaction. The distributed ledger technology has been proposed to address these inefficiencies by sharing the work and the cost behind this process between financial institutions via the distributed ledger.

In our previous work, we proposed a KYC system that leveraged the DLT and introduced a probabilistic repetition of the KYC process. This made the system more robust and increased chances of exposing malicious customers - false negatives that if facilitated by a financial institution, they might later cause the institution billions in fines [26]. However, we neglected one core attribute a DLT-based KYC system should provide for - privacy.

In this work, we utilized the probabilistic repetition of the KYC process to keep its robustness and put a heavier focus on the privacy aspect. We defined four privacy conditions, each being of distinct importance, that cover the privacy of financial institutions and customers operating via a distributed ledger platform. We delivered design and implementation on Corda network of two distinct systems - a centralized one that involved a trusted third party in the form of a regulator or external company, and a fully decentralized one that would not require any collaboration with a third party.

The two systems present a natural trade-off between involvement of a third party and privacy. The centralized system meets all privacy conditions but requires a synergy

between the financial institutions and a trusted third party. This might increase the costs and bureaucracy. The fully decentralized system requires no involvement of a third party, but only fulfills the first two privacy conditions. It partially breaks the third and the fourth privacy condition. We saw that the last two privacy conditions are not fully breached and the extent of the leak of information depends on the probability of repeating KYC for customers, and the interconnectedness of customers between financial institutions. The former introduces a compromise between robustness and breach of the fourth privacy condition. Under certain settings, such as when the banking clients operate with several banks and would do not mind that these banks know of each other, implying the fourth privacy condition would not be relevant, this system might be desirable.

Bibliography

[1] Majid Bazarbash. Fintech in financial inclusion: machine learning applications in assessing credit risk. 2019.

[2] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 1:15, 2016.

[3] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[4] Colin Powell Charles Freeland. Customer due diligence for banks. 2001.

[5] Ting-Hsuan Chen. Do you know your customer? bank risk assessment based on machine learning. *Applied Soft Computing*, 86:105779, 2020.

[6] Jacky CK Chow. Analysis of financial credit risk using machine learning. *arXiv preprint arXiv:1802.05326*, 2018.

[7] Corda. Documentation and training for corda developers and operators, Accessed on April 1, 2021.

[8] Kelvin Dickenson. The future of kyc: How banks are adapting to regulatory complexity, 2019.

[9] Matúš Drgon, Lamprini Georgiou, and Aggelos Kiayias. Robust kyc via distributed ledger technology. 2020.

[10] FATF. Fatf 40 recommendations, 2004.

[11] FCEN. History of anti-money laundering laws, Accessed on April 1, 2021.

[12] FDIC. Bank secrecy act, anti-money laundering, and office of foreign assets control, 1970.

[13] Mike Hearn. Corda: A distributed ledger. *Corda Technical White Paper*, 2016, 2016.

[14] Tommy Koens, Scott King, Matthijs van den Bos, Cees van Wijk, and Aleksei Koren. Solutions for the corda security and privacy trade-off: Having your cake and eating it. 2020.

[15] Jochen Kruppa, Alexandra Schwarz, Gerhard Arminger, and Andreas Ziegler. Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications*, 40(13):5125–5131, 2013.

[16] YVONNE Lootsma. From fintech to regtech: The possible use of blockchain for kyc. *Fintech To Regtech Using block chain*, 2017.

[17] Debajani Mohanty. Corda architecture. In *R3 Corda for Architects and Developers*, pages 49–60. Springer, 2019.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[19] Vimalkumar Pachaiyappan and R Kasturi. Block chain technology (dlt technique) for kyc in fintech domain: A survey. *International Journal of Pure and Applied Mathematics*, 119(10):2108, 2018.

[20] Trilok Nath Pandey, Alok Kumar Jagadev, Suman Kumar Mohapatra, and Satchidananda Dehuri. Credit risk analysis using machine learning classifiers. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 1850–1854. IEEE, 2017.

[21] José Parra-Moyano and Omri Ross. Kyc optimization using distributed ledger technology. *Business & Information Systems Engineering*, 59(6):411–423, 2017.

[22] José Parra-Moyano, Tryggvi Thoroddsen, and Omri Ross. Optimized and dynamic kyc system based on blockchain technology. *Available at SSRN 3248913*, 2018.

[23] Thompson Reuters. Global know your customer survey, 2017.

[24] Oracle Financial Services. Know your customer risk assessment guide, 2014.

[25] Prince Sinha and Ayush Kaul. Decentralized kyc system. *International Research Journal of Engineering and Technology (IRJET)*, 5(8):1209–1210, 2018.

[26] Oliver Smith. Record-breaking fines on banks for kyc/aml non-compliance, January 15, 2021.

[27] Anuraj Soni and Reena Duggal. Reducing risk in kyc (know your customer) for large indian banks using big data analytics. *International Journal of Computer Applications*, 97(9), 2014.