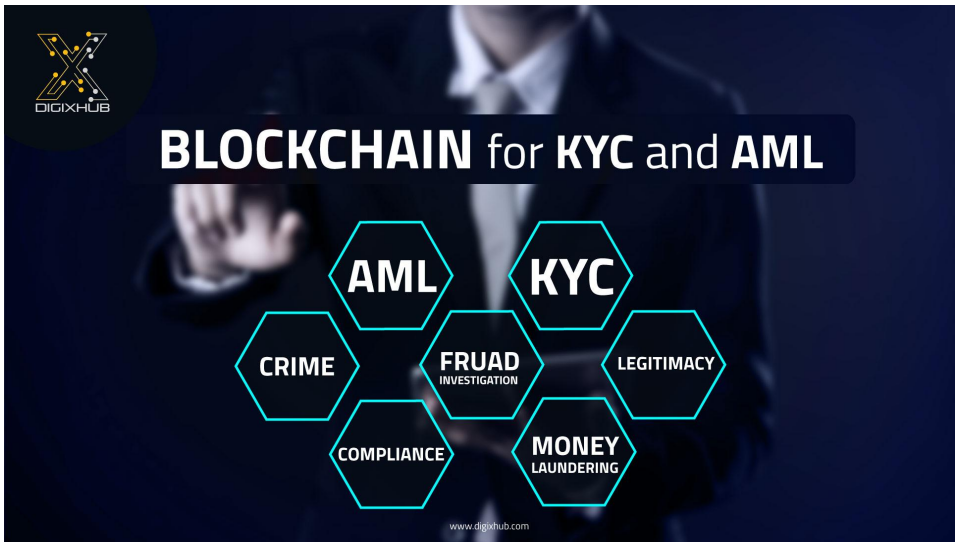
Corporate Banking Blockchain

Blockchain Business Development (CINTV1803E)

Online Oral Exam Based on Written Product (IC)



CBS students:

Akrem Sirag Ademnur Abrahim (S101534) 08-11-2021

Christian Raunkjær (S120504)

Rune Damkjær Licht (S92573)

KU students:

Aleksa Đurđević

Christian Virt

1 Abstract

Systems and concepts that use blockchain technology to improve the know-your-customer (KYC) process have in the existing literature mainly been proposed at a conceptual level and all have missing elements in terms of validation, adoption and implementation. This project hereto proposes a new solution to the cost and time ineffectiveness of the current KYC processes in the Danish corporate retail banking sector by incorporating existing literature with our new Corporate Banking Blockchain innovation.

Our solution introduces the concept of merging KYC processes from different banks with the possibility of balancing risk and cost-efficiency while also making it possible for financial institutions (FI) to dynamically update information related to customers and share this information among participating FIs. By using the Ethereum network we design and implement a functioning prototype showcasing the usability and proof-of-concept of our solution. The result is a stand-alone blockchain-based platform that reduces the costs incurred within the KYC process without any central authority to store the customer's data. The FIs share the initial costs of the KYC process as well as the continuous costs of keeping KYC information and data about corporate clients up to date.

2 Table of Contents

3 Introduction

Our project and innovation is based on the idea of using blockchain in anti-money laundering (AML), Know-Your-Customer (KYC) and Fraud Prevention processes in the Danish retail corporate banking sector.

When corporate clients wish to open a business account, banks must know and obtain a number of documents to be able to enter into a relationship with the customer which are stored on internal data systems alternating between banks (Finance Denmark, 2019). Our innovation would be an open-source distributed ledger platform designed to record, manage, and automate these legal documents between the largest financial institutions (FI) in Denmark. In order to grant read and write permission to FIs other than the one that conducted the KYC process, we would use a private, proof of authority (PoA) based blockchain in which only FIs belonging to the consortium can participate. These two pieces of technology (the distributed database architecture to share data and the blockchain technology to manage permissions) (Mayoana & Ross, 2017) constitute the main innovation of our system called Corporate Banking Blockchain (CBB).

Today the Danish Money Laundering and Terrorist Financing Prevention Act requires banks to prepare adequate written internal rules about customer due diligence, reporting, record-keeping, internal control, risk assessment, risk management as well as training and instruction programmes for their employees in order to forestall and prevent money laundering and financing of terrorism (Finance Denmark, 2019). These internal rules constitute the bank's KYC processes. In practice, the KYC process for a Danish corporate customer wishing to open a business account, is described in appendix 1, showcasing that banks must obtain the customer's business registration (CVR) number, business creation documents, annual reports, documentation for the business ownership structure as well as beneficial owners and ensure that the customer information is updated regularly (Finance Denmark, 2019). Most of these documents are publicly available, but do require the banks to access several databases like CVR, Greens and for example the business creation documents must be acquired and sent by the corporate client, which in practice takes a long time results in a lot of “back-and-forth” making it a pain point for clients and dragging out the entire KYC-authorization process.

Despite the fact that these KYC processes appear as business constricting and time wasting, the AML regulations help societies, governments and banks to prevent financial crime and terrorism financing in a global setting (Mayoana & Ross, 2017). The importance of this was highlighted in 2017 where it was revealed that Danske Bank between 2007 and 2015 had over €200bn of suspicious transactions linked to money laundering and criminal activities originating from Russia, former Soviet states and

elsewhere flow through its Estonian branch (Bjerregaard & Kirchmaier, 2019). Danske Bank saw its market capitalization halved, is expected to receive fines in the range of 10-15bn DKK and are being investigated by a number of international authorities, but most importantly the case triggered regulatory, political and financial shockwaves throughout the world (Bjerregaard & Kirchmaier, 2019).

Especially in Denmark where the Systemic Risk Council, which monitors threats to the stability of the country's financial system, said the trust in Denmark's entire financial sector was tainted by the Danske Bank scandal (Bjerregaard & Kirchmaier, 2019). Following the scandal, The Financial Action Task Force's (FATF) evaluation report on Denmark, estimated that the scale of money laundering in Denmark stands at €2,8bn a year (Finance Denmark, 2019). These numbers caused local and international regulators to further increase AML control by demanding the financial service sector to comply with an increasing number of procedures, laws, regulations and the danish parliament agreed to increase fines for violation of the AML Act by up to 700%, ensuring that the Danish level of fines will be among the absolute top in Europe (Bjerregaard & Kirchmaier, 2019).

Today, the risk of not knowing your customer well enough can therefore be fatal to organisations. The regulatory environment has become increasingly complex, both nationally and globally. High values are at stake, both in monetary terms and in reputation (Mayano et al., 2019). Estimates suggest that the six largest banks in Denmark combined employ about 4.300 AML and compliance staff, tasked with monitoring and ensuring that banks are not misused for money laundering, terrorist financing or other financial crime which amounts to almost 12% of the entire Danish banking workforce (Finance Denmark, 2019). Staff numbers have been rising for a while and are expected to rise further going forward (Finance Denmark, 2019). To compensate these workers the total payroll for bank AML and compliance officers in Denmark stands a staggering 3,38bn DKK (Finance Denmark, 2019).

These regulations and additional staffing requirements drive costs up and challenges efficiency for the banks. The average time a FI takes to 'onboard' a corporate customer is 26 working days (Mayano et al., 2019) and a study by Bain & Co. estimates that risk, governance and compliance costs account for 15-20% of the total "run the bank" cost (Memminger et al., 2018) which for Danske Bank is equivalent of 5,6bn DKK annually (Danske Bank, 2021). Preventing crime thus has the unfortunate side effect of making financial markets less efficient and extremely costly. Moreover, data from a Thomson Reuters survey found that 89% of corporate customers had a bad KYC experience, and 13% had changed their financial institution relationship as a result (Thomson Reuters, 2016). Moreover, data shows corporate customers work on average with 11 FIs, which implies that this – costly – KYC process is repeated on average eleven times for each corporate customer (Mayano et al., 2019). Financial sector regulations are therefore forcing banks to be less effective, increase costs and are

negatively affecting both the customer experience as well as the bank's business opportunities. Our innovation, CBB, is able to solve some of these issues. By helping to gather data and being a single point of entry for KYC-information, CBB creates trust in the financial system and protects it from misuse – together with banks, the bank's customers, regulators and other relevant stakeholders. A win-win where protecting society does not come at the expense of sustaining a friendly business environment.

4 Scope and delimitation

Some limitations and carefully considered assumptions have been necessary in the preparation of this project, and factors deemed more important have been prioritized, which means other aspects of the analysis have been simplified. First, focus is concentrated on the Danish retail banking sector and the 8 Systemic Important Financial Institutions (SIFI) which include Danske Bank, Nykredit Realkredit, Nordea Kredit, Jyske Bank, Sydbank, DLR Kredit og Spar Nord (Finans Danmark, 2021). Secondly, the project focuses on companies as legal entities and the KYC processes on corporate clients as well as the public and corporate business documents used in this KYC-authorization which are not regulated by the data protection rules or GDPR regulations (European Commission, 2021). Thirdly it is assumed that the SIFI banks would be willing to pay for, adopt and use a collaborative blockchain-based platform to effectivise the KYC processes for a mutually beneficial solution and that this platform would be compliant with all financial and data-based laws and regulations.

Important assumptions about our solution is that there exists a *core KYC process* in a regulative environment, e.g. Denmark, where all banks are required by regulations to perform KYC on their clients, with a minimum set of requirements. This core KYC process is something that applies to all banks, in which case they should all be able to agree on the price of such a process. The system also assumes that banks can agree on the cost of updating a KYC process, if e.g. regulations change. For the remainder of this report we assume the notion of a core KYC process and a uniform price. It should be noted that in reality, the complexity of a KYC process may vary from client to client, based on the facilities that the client intends to use at a financial institution. E.g. if bank transfers of a given client exceeds a certain amount, then an *extended KYC process* may be required (Drgon et al., 2020). However, in this report when we refer to a KYC process, we implicitly refer to a core KYC process, shared across all financial institutions in the same legislative environment.

5 The Innovation

5.1 Existing work

Achieving standardized and compliant KYC-information has always been a great challenge due to the vastness of the financial system as well as the sheer number of companies and financial institutions affected (Mayoana & Ross, 2017). This is the problem CBB aims to solve: Creating a single point of entry for reliable and compliant KYC-information. CBB is an extension of the concept originally presented by Mayano and Ross (2019) where CBB would utilize the same blockchain archetype and business model in which the establishment of a smart contract could accommodate dynamic compensation among participating FIs. The existing idea proposes that the blockchain contains KYC processes for each client, and that a fingerprint of the document collection used for the KYC process is stored on the blockchain. Then other banks can share the cost of the KYC process by buying the existing document collection, and verify that the fingerprint of the collection matches the corresponding fingerprint on the blockchain.

However, despite the fact that this paper was later updated by Mayano, et. al (2019), their approach to reducing the costs of KYC, involves certain inherent weaknesses that make their validation and authentication process flawed and prone to errors. Mayano and Ross (2017) suggests that one bank, “The Home Bank", gathers all necessary documents and verifies the validity of these documents. Home banks can therefore upload wrong or falsified documents without any second-hand authentication on the blockchain. Drgon, Georgiu & Kiayias (2020) also discusses the ‘brittleness’ and the ‘one-tier-problem’ of the system proposed by Mayano, et. al (2019). The primary issue is that Mayano, et. al (2019) assumes that all banks are equally capable of correctly performing a KYC process, and that the process cannot fail, e.g. due to falsifications or human errors.

5.2 Our contribution

The brittleness of the system refers to the fact that a corporate client might get approved at “bank A” with potentially falsified documents, and continue to perform no illicit activities with bank A. The client may then approach “bank B”, and get approved using documents from bank A, which were not properly checked by bank A. Then the client may perform illicit activities at bank B, while remaining unsuspicious at bank A. The one-tier problem refers to the fact that a bank may not want to risk using another bank’s documents, since they have to rely on the fact that the KYC process was done correctly. To solve the brittleness problem they propose that banks with some probability must complete a replication of a KYC process, when they attempt to buy documents from another bank. To solve the one-tier problem they propose that banks have an internal rating system. While Drgon, et. al

(2020) proposes a solution to the problems, it creates a system where banks might be unwillingly forced to perform the KYC process themselves, and it also leaves their risk management strategy to be based upon a random probability. This project and CBB aims to solve the brittleness issue and one-tier problem by proposing a different solution. Our solution is that banks may optionally perform a KYC process for some corporate client, or may optionally buy documents from another bank, which is entirely based upon the bank's own risk management strategy. We introduce the concept of *merging* KYC processes together, in order to create a merged collection of documents which can contain several KYC processes from multiple banks. The more document collections that are merged, the safer it is to buy the merged collection, but it also increases the overall price. In this way banks can choose how many replications of a KYC process is necessary, in order to deem a purchase safe while also balancing their own cost-efficiency. Our solution introduces the notion of *participants* and *subscribers*. A bank is a participant of a KYC process if it has conducted the KYC process, and holds a collection of documents. A merged KYC process has multiple participants, corresponding to the number of KYC processes that were merged. A bank is a subscriber to a KYC process, if they bought the rights to the document collection.

The challenge is that we maintain the KYC specifications proposed by Mayano, et. al (2019), while also allowing for the optional merging of KYC processes, which now involves both subscribers and participants. The specifications of the KYC system as proposed by Mayano, et. al (2019) should be:

Proportionality: Costs are shared equally by all financial institutions to ensure the system is fair. *Privacy*: Financial institutions do not want to have their identity revealed. *Irrelevance*: No incentive to be the one who participates in a KYC, or subscribes to it. *No-minting:* A financial institution cannot simulate/fake having performed a KYC process without actually performing it.d

Using blockchain technology we maintain a history of the KYC processes, participants and subscribers, and it will serve as a KYC history which is accessible to all financial institutions, where they can verify if a KYC process should have been performed. This ensures the no-minting property, since it is possible to get compensation, using the blockchain history, and showing the lack of a proper KYC document collection as proof. Proportionality becomes slightly more complicated with merged KYC processes. The cost depends on the number of participants, therefore, it should not be possible to add more participants after a merge has been completed. Otherwise, it might be cheaper to subscribe before a participant was added, rather than if you subscribe afterwards. Irrelevance is secured by making sure that individual KYC processes that are used in a merged KYC process cannot be subscribed to afterwards. If the merged collection does not highlight any issues with the KYC processes it holds, then each individual KYC process can be seen as more secure. Then you might avoid subscribing to the merged collection, since it is more expensive.

One would rather subscribe to an individual process rather than the merged collection to save costs. However, this would induce that it can be less cost-efficient to be a participant. Hence, why subscriptions to the individual KYC processes is disabled. While some banks might not like the idea of getting their individual KYC process disabled, they can simply disallow merging. Banks can also choose the minimum or maximum participants they require for their individual KYC process to be included in a merged KYC process. Banks can use the minimum and maximum participants as a variable to balance risk and cost-efficiency. Privacy is ensured since each bank is identified by a public cryptographic key, where only the central membership service provider of the distributed ledger technology (DLT) knows who the key belongs to. However, banks might be able to deduce the identity of other banks if they intensively analyze the blockchain history, and ask their clients about what other banks the clients are using. Banks should be able to receive a new public cryptographic key if necessary.

The KYC DLT solution is expected to improve on current areas within KYC. The solution will reduce KYC costs for banks, since KYC processes are shared between banks. It will Increase the effectiveness of KYC since the merging of KYC processes will produce even more thorough KYC processes. Which reduces the potential for bank fines, but also reduces the possibility for illicit activities such as money laundering and terrorism financing. Furthemore it will improve customer experience by having a faster and more pleasant KYC interaction with the bank.

The need for a more cost sustainable KYC solution for banks is needed. The usual current solution for banks is simply to hire more employees (Drgon et al., 2020). The use of blockchain has already been considered by Britton (2016) in the context of KYC/AML measures and considers that the KYC framework is probably one of the most suitable for the application of blockchain technology. This is mainly due to the fact that we have a banking system wherein other banks do not completely trust each other. However, banks can still benefit from cooperating in the field of KYC, since they serve many of the same corporate customers and are subject to many of the same risks.

Combating money laundering and terrorist financing is a responsibility common among all banks, and can, as earlier mentioned, also lead to very high fines if not done properly, which provides a further incentive for collaboration between banks. Moreover, under current law, it is not possible to warn other banks against a customer. In other words, if a bank rejects a customer on the grounds of money laundering, and the customer then contacts the bank ”next door”, the bank will not be able to warn the bank next door that the customer is likely involved in money laundering or terrorist financing (Finance Denmark, 2019). CBB can hereto enable the possibility for higher alertness of a bank conducting a KYC process for customers that have previously been “rejected” by other banks, due to the visibility and transparency on the blockchain.

6 Business Process & Business Model

As a next step in our paper, after the introduction of the issues regarding the KYC processes within the banking sector, and showcasing how our CBB not only helps alleviate some of these issues, but is able to further improve on our financial systems, we want to take a deeper dive into the business model and our service itself. We will do this, by first utilizing the CATWOE model developed by Peter Checkland and Jim Scholes (1990). We will use the information gathered from the CATWOE to assist us in drawing a Rich picture, which will help quantifying the specifics of our system, and how it is connected from start to finish. Lastly, we will utilize the Business Model Canvas (BMC) by Alexander Osterwalder & Yves Pigneur(2010), to gain a better overview on the business model of our CBB. This should leave us with most details ironed out, and we will then take on the task of diving into the development process of our prototype.

6.1 CATWOE

The CATWOE analysis is used to assess the impact a business change has from the perspective of the stakeholder(s). It can be used to quickly determine the direction in which a recommendation should shape the outcome. If used efficiently, it should help business stakeholders in addressing concerns based on subject before partaking in said project (Checkland & Scholes, 1999).

The CATWOE analysis consists of six elements, which are described as following: The **'Customer(s)'**, which describes who the output will affect and what the changes will be for these parties as well as their initial reaction. The **'Actor(s)'** are described as the ones carrying out transformation, their roles and how they will react to the impact. Next step is assessing the business activity or **'Transformation'**, where the process is defined. **Weltanschauung** or **'World View'** describes the existence of the business activity. '**Owner(s)'** describes the real change instigators and **'Environment'** the constraints posed by the surroundings such as market, sector, country (Business Change Team, 2020).

By putting yourself in the stakeholders perspective, the following business change is formed:

Customer(s)	**A**ctor(s)	**T**ransformation	**W**orld View	**O**wner	**E**nvironmental
Financial institutes: Banks Credit institutions . Equity & assets management . Tax offices . Bank customers	Bank Managers Bank KYC Employees Node Validators Corporate clients Centralized MSP CBB	Insecure / untrusted input. → Gets verified by validators. → Output now secure/trusted and shareable	As-is: Resource demanding, Locked To-be: Efficient, Transparent	CBB in collaboration with financial institutes	Regulations and laws dictate the need for security and validation. Public trust (needs) to increase in relation to revelations done in the past (e.g., Danske Bank, Pandora Papers, etc.)

Table 1 - CATWOE analysis

This gives an overall assessment of the requirements needed to bring forth the business activity. The ‘Customer(s)’ interest lies in lowering the required time and resources required to uphold the KYC requirements dictated by the ‘Environment’ which consists of the public opinion as well as governmental and legal requirements. The ‘Actor(s)’ are the financial affiliates who utilizes the blockchain solution run by the ‘Owner’, CBB. The ‘Transformation’ makes the initial output more secure, validated and shareable which also acts as our ‘World View’ which is a more transparent process, which resolutes in fraud prevention by knowing your customer better. There can be arguments about an impossibility of the banks giving away ownership by sharing this data, but the requirements from the environment have increased which has also increased the initial cost for running a proper KYC business activity, hence a higher probability of banks joining a collaboration to increase efficiency.

6.2 Rich Picture

The rich picture was presented by Peter Checkland (1999) as a tool that can be used to both explore, define and acknowledge a given situation through diagrams to develop an initial model which can help to create a shared consensus between people. It was developed as a part of Checkland’s soft system methodology (SSM) and it takes form as the first 2 steps. First identify the issue you wish to

address and secondly develop an unstructured explanation on how you see the issue within the context of the diagram (Checkland & Scholes, 1999).

The following rich picture provides an overview of our business model to showcase and illustrate the situation of how our service would shape the environment:

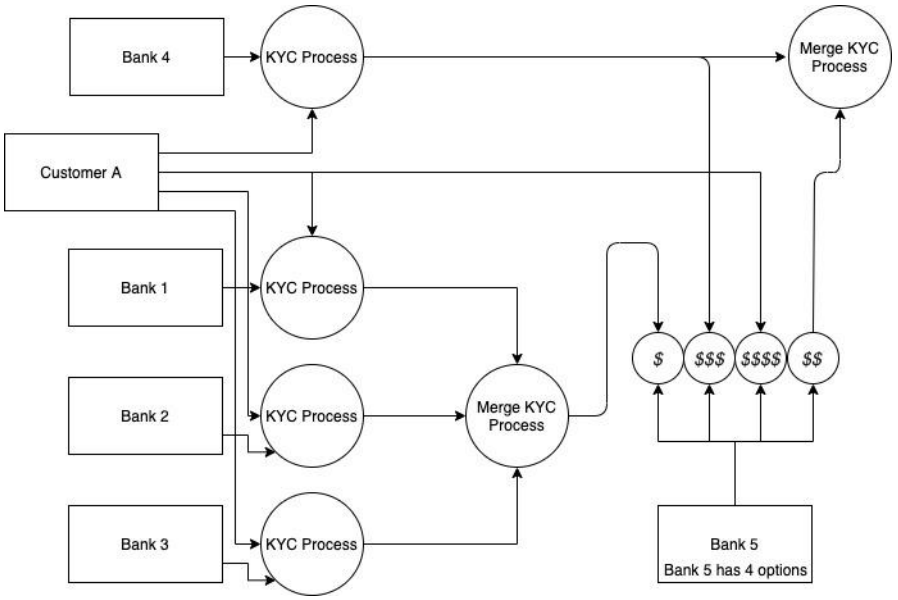


Figure 1: Rich Picture #1 Source: Made using draw.io 7.11.2021.

Figure 1 hereby highlights how we see 'Customer A' approaching 4 different banks, 'Bank 1, 2, 3, and 4'. As mentioned earlier, banks can collude and merge their KYC-process to ensure higher quality and additional safety, though it is costly as each bank has to complete the KYC-process. This is shown through 'Bank 1, 2, and 3'. However, banks can also, depending on their risk assessment, decide to do the KYC-process themselves (see 'Bank 4'). This creates an ecosystem where banks later on ('Bank 5') will be presented with 4 different options when they are approached by 'Customer A'. They can either do the KYC themselves which is very costly and has a medium level risk. They can choose to access the KYC done by 'Bank 4', which is cheaper, but could be a higher risk venture, as they put their trust in 'Bank 4's' KYC process. However, they also have the option to purchase the merged KYC process by 'Bank 1, 2, and 3', which is the least risky venture, as the KYC has been done 3 times and checked up against each other. Furthermore, this option initially will be more costly, than just buying the KYC from 'Bank 4', as 'Bank 5' would have to pay the value equal to a fourth of three KYCs. However, as more banks access this merged KYC, the cheaper it becomes, as the next bank would pay 'Bank 1, 2, 3, and 5' one fifth of the cost of the initial 3 KYCs. So as time goes on part of the cost will be reimbursed to 'Bank 5'. Lastly, Bank 5 has a last and 4th option of doing the KYC alone and then merging it with the KYC from bank 4, which would have an initial higher cost than option 3, and a higher risk assessment as well.

6.3 Business Model Canvas

The Business Model Canvas (BMC) was first introduced in recent times by Alexander Osterwalder and Yves Pigneur in “Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers” (2010). It is a tool that can aid in the facilitation of a more rooted and concrete understanding of a certain company's business model (Osterwalder & Pigneur, 2010). Utilizing the canvas correctly will enable the user to have more insight of the specific value propositions and the channels they exist within (Osterwalder & Pigneur, 2010). It would also be able to showcase how the company takes advantage of their value propositions and channels to generate money. Furthermore, you can also use the BMC to gain a broader understanding about the company's customers.

The BMC traditionally looks at nine different points, but has since its inception had altercations made as some points could become less fruitful in allocating resources to depending on the industry within the scope of the BMC (Osterwalder & Pigneur, 2010). For the scope of our project, we will omit the exact cost structure for running the business and focus on lean operations costs. In addition to bare necessities such as electricity and a running internet connection, we would need servers and data storage to run the business operations. The cost for running operations should aspire to lower the overall costs for the banks despite paying an initial fee to join the ledger. This paywall would also act as a maintenance and development overhead for further progress and for the initial validation price of banks joining the ledger.

Looking at the nine different points first introduced by Osterwalder & Pigneur 2010**.** We identify our *customer segment,* within the scope of our assignment as the 8 Systemic Important Financial Institutions (SIFI). Our *Value proposition* lies within not only simplifying the KYC process of the SIFI banks by building a blockchain based ledger system , but also the guarantee of data being correct by adding a layer of verification that we believe have not been implemented prior in a KYC process within the banking sector. Our *revenue streams* would come from the banks paying a subscription on a recurring basis to be part of the privatized blockchain. (Mayano et al., 2019). Our *communication channels* would be developing as the business grows.

First, we imagine we would have to physically contact the SIFI banks and garner their interest by pushing our business model. It is hereto the ambition of both politicians and the banking industry that Denmark should position itself as forward-looking and proactive in the AML area (Finance Denmark, 2019). Denmark’s AML Task Force hereby recommends that joint IT solutions are one of the main of possible initiatives to combat money laundering for the Danish banking sector, and suggests that a joint data register as an industry-wide AML IT project should be implemented as soon as possible (Finance Denmark, 2019). However, today, the money laundering and terrorist financing challenges

tend to be silo-based, with the three IT providers BEC, Bankdata and SDC as well as Danske Bank and Nordea each developing their own solutions which means that expanding industry-wide IT solutions and their implementation will take time and require substantial resources as well as basic agreement on scope etc. (Finance Denmark, 2019). We would therefore have to convince the SIFI banks that substantial efficiency improvements and financial savings can be achieved by joining the CBB platform where their joint sector investments will result in a more efficient defence against money laundering and terrorist financing than if investing separately and at different paces (Finance Denmark, 2019). Moreover, highlighting the security and tamper-proof abilities of the blockchain technology should be prioritized in the prospecting and customer attraction process, because banks IT-security are one of the industry's main improvement areas (Finance Denmark, 2019). Lastly, the upside of increasing the confidence in the financial sector, facilitating the daily interaction between banks and their customers, minimizing money laundering and finally reducing the KYC costs incurred by banks should be communicated in a way that the banks would instantly be able to see the benefits to them, the industry and society as a whole.

Once the SIFI banks have joined our project and migrated their KYC processes to function through CBB, we would be able to garner the interest of smaller banks by showcasing the effectiveness, savings and security the SIFI banks have achieved. From here on, we could branch out to digital advertisement, as well as continuing the cold calling process to get more banks to join. However, it would be ideal if the banks themselves would start to see that the SIFI is utilizing our services, and therefore would be the ones to establish contact to join the larger network. Our *Customer Relationship* would ideally start out with an automated process where the banks can see if they fit the criterias to join the blockchain. After an initial approval by the automated system, we would have personal relations with the main departments of each bank where they can reach out to specific individuals within the company in case questions or issues would arise. We believe this to be especially important, as much of the data and the service provided would be critical to work and be correct in order for the banks to be able to provide their services. Our daily *key activities* would entail monitoring the blockchain and its system. We imagine a substantial amount of servers depending on the size and growth of our network should be maintained. Furthermore, we would need a department for further sales & development of the customer segments, as well as a department handling the daily communication with the banks varying from simple questions and inquiries to reporting problems such as bugs and so forth. Furthermore, this department would also be in charge of refunds in case one bank pays for a KYC file, and then sees it was not done properly. A department for further development and maintenance of the blockchain would also be needed. Furthermore a department to handle the guarantee of the anonymous file transfers between the banks would be needed. Initially it is hard to find any *key partners* for our business model as the development, support, maintenance, and sale all would lie within the confines of our project itself. However, our service is nothing but an

empty shell without the data supplied by the banks for the KYC processes. So you could argue that even though the SIFI banks are our customers, they are also key partners as our service would not be able to function without their data. In short, they would rely on our system and services, which in term would rely on their supplied data. Lastly, within a BMC you would take a look at the *cost structure* of your company from a top-down view, but in order to do so, more research on the exact structure of the company would be necessary. How would the internal hierarchy look, what positions would we need, how many for each. Furthermore, what type of equipment should we purchase, where should we be situated. Should we offer on-site support, and so on. These are all valid subjects that need thought, but for the sake of the scope of our project, we will leave the details of this part out, as we believe it deserves a larger amount of attention and research which lies in parallel to the actual topics of importance for our project and its learning objectives (Osterwalder & Pigneur, 2010).

7 The Development Process

Our business consists of three main technological parts: The first is the Distributed ledger technology. The second is the centralized file transfer service. And lastly our smart contracts that enables registering KYC processes and facilitates cooperation of KYC between banks. These parts will be further elaborated in the following subsections.

7.1 Distributed ledger technology

Our functioning prototype is developed and can be deployed on the Ethereum network. However, due to scalability issues and the transaction fees on the network, we only use it for our proof-of-concept. The final product will utilise a service like Hyperledger Fabric or Corda R3. We will delve further into the details of Hyperledger Fabric and the properties of the network. A primary difference between Ethereum and Hyperledger, is that Hyperledger utilises Proof-of-Authority (PoA) rather than Proof-of-Work (PoW) to increase scalability and efficiency of transactions. Hyperledger Fabric is an open-source distributed system for permissioned blockchains, which is already used to implement multiple distributed systems across various industries (Androulaki et al., 2018). Hyperledger Fabric can be specified to only require a subset of peers to execute transactions, which improves the potential throughput. In contrast Ethereum employs an order-execute paradigm in which all peers must order transactions, execute, and then solve PoW. Then a successful PoW block must be validated and executed by all other peers. In contrast Hyperledger Fabric uses an execute-order-validate paradigm. In the execution phase a subset of peers called endorsers execute and validate a transaction in a virtual machine, which does not affect the state of the ledger. This enables each endorser to detect non-deterministic operations or DDOS attacks. If enough endorsements are received, then the

transaction is submitted to an ordering service, which orders the transactions. Finally the transaction is sent to all peers which validate and commit/aborts the transaction. Smart contracts (also called chaincodes) can be implemented in standard programming languages (e.g. Java) which makes it easier to find developers for the distributed applications.

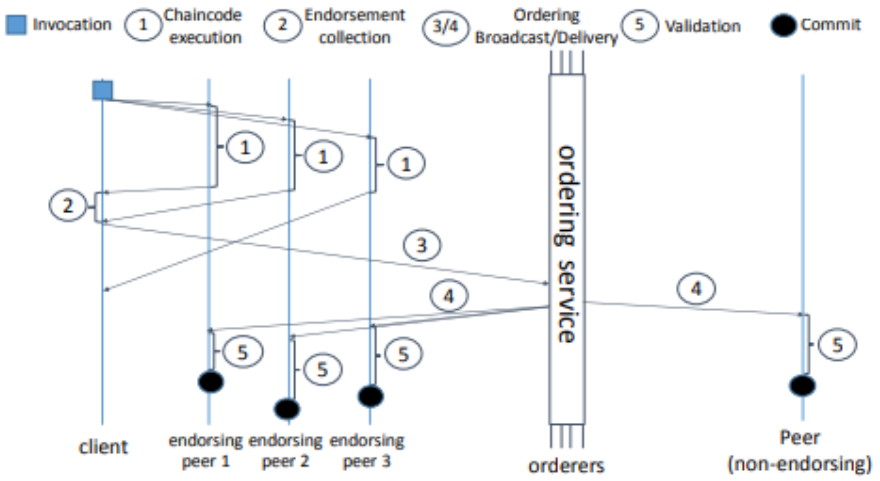


Figure 4: Fabric high level transaction flow.

Figure taken from (Androulaki et al., 2018) that illustrates the execute-order-validation paradigm. The peer at the utmost right-side is not part of the endorsers and simply validates the transaction.

7.2 Centralized file transfer service

We will briefly discuss how the centralized file transfer service could be implemented as part of the DLT solution. A Hyperledger Fabric system has a membership service provider (MSP) that provides peers (financial institutions) with permission to the blockchain. When permission is given by the MSP to a financial institution, the institution receives a public key to represent their identity on the blockchain. Since documents need to be transferred between peers, each peer needs an address of where the documents should be sent. If documents are sent directly between financial institutions, then their identity might get compromised. Instead, the MSP can act as a middle-man, such that a document transfer only specifies which public key the documents should be sent to. The MSP knows the actual identity and IP-address of the recipient of a document collection based on their public key. The MSP can also distribute TLS/SSL certificates to peers which can ensure that all communications are encrypted. The blockchain history will be used to determine if a document transfer should or should not have occured. However, any further details about the centralized file transfer service is outside of our scope at this moment in time, since we are primarily interested in the blockchain design and smart contract specifications.

7.3 Smart contracts

Based on our idea we need the following smart contracts which provides the following functionalities to banks:

- Add KYC process to the blockchain.
- Update KYC process on the blockchain.
- Subscribe to a KYC process on the blockchain. I.e. purchase document collection from another bank.
- Merge KYC processes on the blockchain.
- List KYC processes for a particular customer.
- List price of a KYC process

We also need to consider governance, and the MSP will need smart contracts with the following functionalities:

- Add financial institutions as peers on the blockchain.
- Add tokens to financial institutions to pay for subscriptions.
- Remove tokens from financial institutions in case they cash out.

We have implemented a prototype which supports all of these functionalities. Note that the list of functionalities provided here is not exhaustive of what could be needed in the system, however, it serves as a foundation for the proof-of-concept.
The entire prototype is separated into three smart contracts programmed in solidity, where functionality is divided into three categories:

- Payments: smart contract is PayToken
- KYC processes: smart contract is KYCProcess
- Governance for adding financial institutions: smart contract is Governance

The entire code for the prototype can be found in Appendix 4 or the Github repository[1], and consists of 536 lines of code, where the largest and most complicated smart contract is the one that handles KYC processes.

[1] https://github.com/balzic/KYCBlockchain

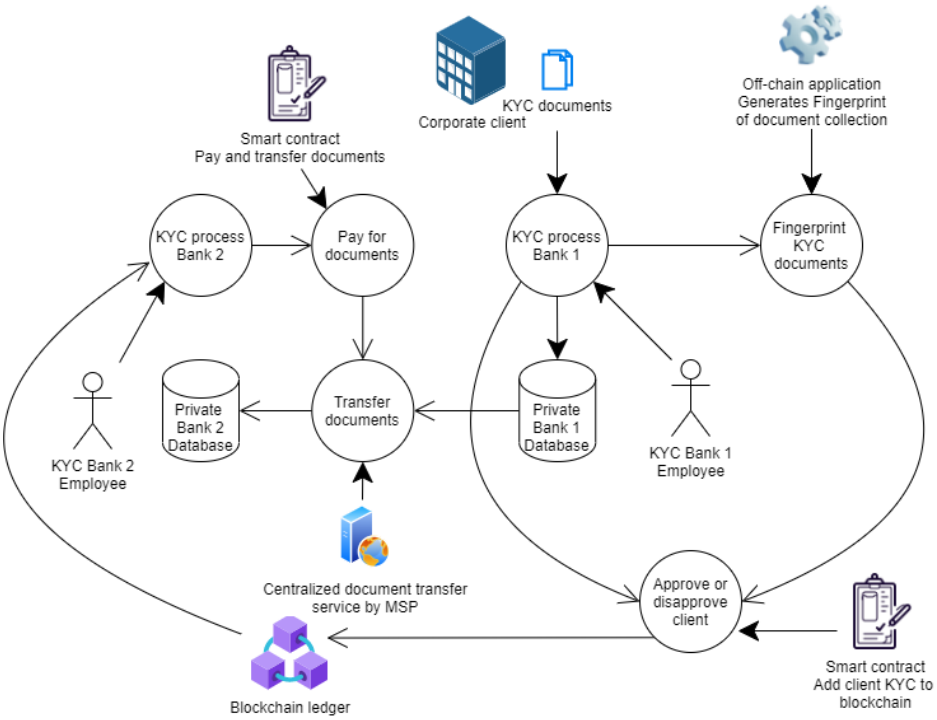


Figure 2: Rich picture: illustrating a basic overview of business processes and services of the KYC distributed ledger technology. Made using draw.io.

Figure 2 shows the basic KYC process, where a KYC employee from bank 1 performs a KYC process using a collection of documents received from the corporate client. The collection of documents are stored in a private database of bank 1. However, a fingerprint of the document collection is created, and a smart contract is used to add the fingerprint to the blockchain together with various parameters. Bank 2 also has to perform KYC for the same corporate client as bank 1 did. The KYC employee from bank 2 can see that a KYC process of the corporate client has been performed. The employee from bank 2 then requests the document collection using a smart contract that subscribes him to the KYC process and pays half of the cost of the KYC process to bank 1. The centralized document transfer service facilitates the transfer of the document collection from the private database of bank 1 to the private database of bank 2.

7.3.1 Payment

In our design considerations, we decided that payment is on-chain with a token that is backed 1:1 with a regular currency. This ensures that the payment history is also on the blockchain and can be used in complaints, and to automate payments. In order for the bank to pay with the token, they must first exchange their regular currency at the MSP for the on-chain token. Banks can at any point withdraw an amount less than or equal to their on-chain token balance, unless perhaps a complaint has been filed and they are under investigation. The implementation of payments can be found in the

PayToken.sol contract shown in Appendix 4. It is rather simple, it has a mapping of balances, and a mapping of locked balances, which can be used in case of complaints. It also has functions to distribute or remove tokens from financial institutions. However, these functions can only be used by the MSP. Furthermore, the function that transfers tokens can only be used by the smart contract that deals with KYC processes, and not financial institutions themselves. This is to ensure that tokens can only be transferred in case a financial institution subscribes to a KYC process.

7.3.2 Adding a KYC process to the blockchain

When adding KYC processes to the blockchain, a set of parameters needs to be specified. We need the following parameters:

1. Client identifier (CVR).
2. Approval status (approved or disapproved?).
3. If the KYC process may be merged with other KYC processes (yes or no?).
4. Minimum participants for merging (one to infinity).
5. Maximum participants for merging (minimum participants to infinity).

When designing this prototype, we had to consider how clients were identified on the blockchain. There is a range of potential solutions, e.g. we could let banks generate a unique identifier whenever they add a new KYC process. This would however, complicate the way that other banks can find KYC processes for specific clients. It is common in many countries and even for the entirety of Europe that a database of unique identifiers for companies exist. E.g. one can go to the business registry at e-justice.europa.eu. However, in Denmark we use CVR numbers for businesses. Our solution is limited to corporate clients that are registered with a CVR number, since this is the main identifier for corporate clients in the blockchain. This approach makes it easy for banks to look up KYC processes for any company with a CVR. The three last parameters are all related to merging of KYC processes. This is where a bank can balance risk with cost-efficiency. These parameters could vary from bank to bank, and perhaps also from client to client.

7.3.3 Updating a KYC process on the blockchain

In order to update a KYC process, one has to be either a subscriber or a participant of the KYC process. It is also not possible to update an individual KYC process, if the KYC process has been used in a merged KYC process. Then the update must occur on the merged KYC process. The list of parameters for updating a KYC process is almost exactly the same as when adding a KYC process. However, when updating one must also specify the identifier of the KYC process being updated. Each

client has a linked list of KYC processes (referred to as KYC nodes in the code). A bank can look up this linked list of nodes and select a node which the bank wants to update.

7.3.4 Merging a KYC process on the blockchain

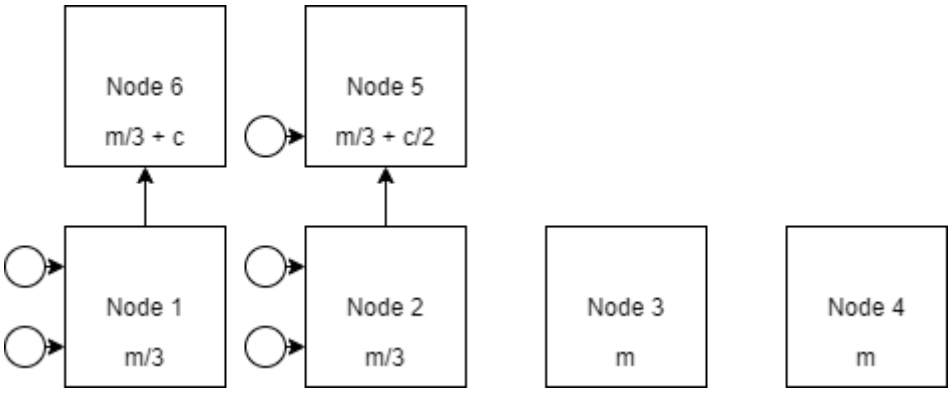


Figure 3: Example of KYC processes on the blockchain for a client. Each node represents a KYC process. The numbers on the nodes show the order in which they were created. The circles represent subscribers. Here, m is the price for a core KYC process. And c is the price for updating a KYC process. Each node shows their price. Source: Made using draw.io.

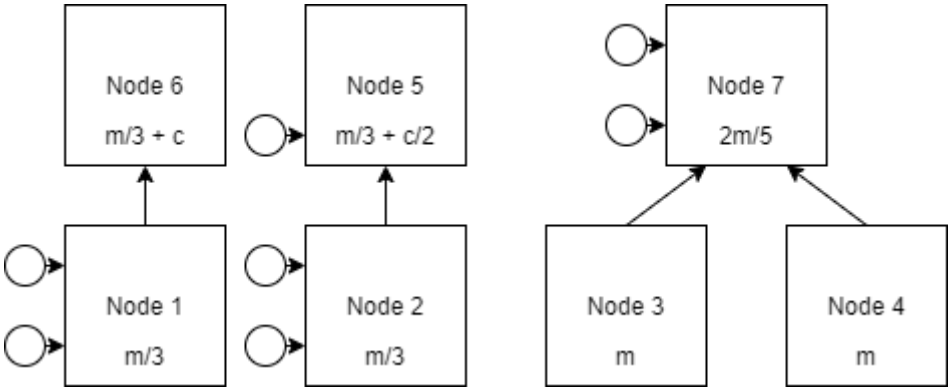


Figure 4: The same example similar to before, however, it now contains a merged node. Notice that the price is 2m/5. There are three participants and two subscribers. A bank noticed that Node 3 and Node 4 had no subscribers, and then decided to conduct KYC and merge with Node 3 and Node 4 to create Node 7. Source: Made using draw.io.

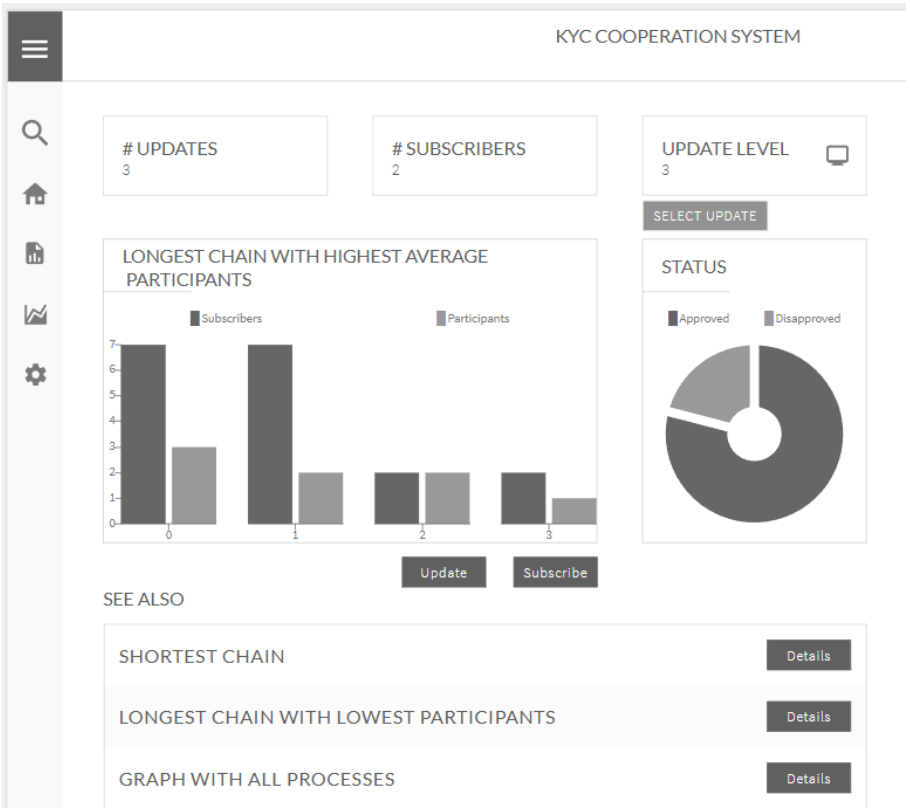
As shown in Figure 3 and 4, the price of a merged node is higher. However, the security of a merged node is also higher since Node 7 contains three document collections from three separate KYC processes, each by a unique bank. In the prototype it is currently only possible to merge nodes, if the merging part also contributes with a KYC process. However, in a complete version it should be possible for existing nodes to merge with other existing nodes. Suppose also that Node 8 was created as an update of Node 1, by an already existing subscriber of Node 1. Then it would also be possible to

merge Node 6 and Node 8 in our prototype. This would create Node 9 with cost $\frac{m}{3} + \frac{2c}{2}$ for each participant and subscriber. We also constructed a formula for the general cost of subscribing to a node A which is $C(A) = \sum_{i=0}^{N} \frac{p_i m_i}{s_i + p_i + 1}$ where p_i is the number of participants, m_i is the price, and s_i is the number of subscribers for update i where $0 \leq i \leq N$ for N updates. That is, when $i = 0$ it represents the initial KYC process with no updates.

The user input required to merge KYC processes is similar to when adding or updating a KYC process, since the prototype currently only supports merging while simultaneously also adding/updating a KYC process. The only extra parameter required is a list of KYC process identifiers (node identifiers) representing the nodes that should be merged.

7.4 Other Design Considerations

The design considerations and decisions mentioned so far have been primarily back-end focused. However, one should also consider the KYC employees and managers of the banks and their experience with the system. Main users of the system are expected to be KYC employees or managers, who most likely do not possess vast technical knowledge, and therefore the given interface should be simple to use and easily understandable.



Source: Example of User-interface of CBB platform. Made using app.moqups.com with at template example

Bank employees should have access to an interface that allows for simple queries, such as getting the longest chain of KYC processes with highest average participants, or the longest chain with lowest participants. It should also be possible to represent all KYC processes in an interactive graph where KYC processes (nodes) can be visualized and selected for subscription, updating or merging.

8 Evaluation, assessment and further perspectives.

The prototype is not necessarily optimal, with respect to storage usage and efficiency of computation. For example, the datatype `uint256` is used for all identifiers and numbers, which is probably not necessary. It is unlikely that the number of KYC processes ever exceeds $2^{256} - 1$for a deployment in Denmark. It might also be possible to optimise some computations. For example, when listing KYC processes, we need to compute the number of nodes for a particular update each time. This could be optimised by storing it in a variable. Going forward, we should reimplement the prototype for the Hyperledger Fabric system and optimise the code.

8.1 Personal Data and GDPR

As mentioned earlier, we have limited ourselves to corporate clients, due to the more strict requirements for storing personal data as a result of GDPR. However, if issues arise due to GDPR even with corporate clients, the system might need some additions. It might also be the case that banks want to use this system for personal clients. To implement this, a whitelist service needs to be implemented such that clients can go to a webpage and give permission when a document transfer occurs. The whitelist service could also be part of the MSP, since it serves as a middleman for document transfers and knows the identity of the banks. This would require an intermediate step that blocks the possibility for subscription until the bank has been whitelisted for a particular customer. This could be implemented as a part of our prototype, but is not currently supported. This solution might be too simple for the complexity of GDPR and further research into the subject is needed.

8.2 Bank Management Concerns

In the paper by Lacity, Steelman & Cronan (2019) it is outlined that from an enterprise perspective it is important for managers to have questions addressed in three main areas, before they can even consider using a blockchain solution. We will phrase the questions with respect to our KYC blockchain solution:

Identity, data and event standardization: Are the identities of corporate clients, the data, and the processes standardized according to business standards, or does it have the potential to become standards? What about connecting with other KYC blockchains?
Governance compatibility/ acceptance: Does CBB govern the system and who is allowed to be in a governance council? Who owns the KYC process data on the blockchain and the rights to the CBB software? Who is liable if something goes wrong, e.g. if a bank is fined for facilitating money laundering, for a customer where they purchased KYC documents from another bank, rather than performing the KYC themselves? *Low Switching Barriers:* Can a bank easily join and leave the system, and also bring data with them?

We will attempt to evaluate our solution based on these questions in the following sections.

8.2.1 Accessing and leaving the system

When it comes to incorporating new clients there needs to be a background check by CBB to ensure their identity as a bank. There also needs to be an education of KYC employees and managers by CBB on how to use the system and platform. Once the banks enter our services, they receive the needed access to the system, that includes their own server to run as a peer and a public key they can use. A bank still has an entire replication of the blockchain on their server in case they decide to leave. Leaving the system would require disabling the bank's access based on their public key, which can result in some technical difficulties for our system, but not for the banks themselves. The problem is that their KYC processes on the system will still be visible. The individual KYC processes could be disabled for subscriptions, but dealing with the merged KYC processes they were involved in would require additional changes to our design and prototype. It would be unwise to disable the merged KYC processes that the bank participates in, since this would affect other banks who are also participants. A potential solution would be that the bank who left the system still gets paid its share on a merged KYC process, and that they can claim it. The documents can then be sent to a new subscriber by banks who also participate in the merged KYC process, since all banks are expected to hold all documents from a merged KYC process. If this is not the case then the merged KYC process must be disabled for further subscriptions, and the subscriber must be refunded.

8.2.2 Governance and liability

CBB will serve as a supporting role for the blockchain as a primary developer, membership service provider, and aid with file transfers. However, any updates to KYC processes or merging KYC processes will be entirely done by banks themselves. Also any updates to the ledger technology must be in agreement with its users. Furthermore, pricing on KYC processes and updates will also be in

agreement with its users. Therefore a large part of the governance is maintained by its users. However, to join the system, banks must agree to let the MSP handle complaints in case they are reported by a bank. A complaint could arise if a bank subscribes to a KYC process, but finds out that the KYC process was not correctly performed (i.e. missing core documents that are part of a core KYC process). A committee should be appointed, consisting of perhaps 2 or 3 people from different banks that are not involved in the complaint, to make a decision on whether or not there are any grounds to the complaint. Both banks in a complaint case will have full anonymity throughout the case. If the complaint is accepted the subscribing bank is refunded, and the bank responsible for the incorrect KYC documents pay for the costs of the committee, which is assumed to be easily measured. However, if the complaint is rejected, the complaining bank pays the costs and gets no refunds.

There are however, even more serious liability issues to be considered. Imagine bank 1 subscribes to a KYC process performed by bank 2, and proceeds to operate with the corporate client based on this KYC process, which may or may not be correct. If one day it is discovered that the client is conducting money laundering using the services of bank 1, then bank 1 might get a considerably large fine. The question begs, should bank 2 be accountable for the possibly correct or incorrect KYC process? There are a few approaches to consider, should we hold bank 2 accountable or should we hold bank 1 accountable, and what if the KYC process is actually incorrect. Our system has a feature to subscribe to merged KYC collections, from multiple banks. Therefore, bank 1 could have chosen a safer option, rather than just subscribing to bank 2. By not holding bank 2 accountable, we actually promote that banks collaborate and merge KYC documents, in order to avoid risks. By holding bank 2 accountable, we risk that nobody wants to use the system due to the potential for fines in the range of several billion DKK. Therefore it seems most likely that banks would use the system if they are only accountable for incorrect KYC processes, which can easily be refunded. This puts more responsibility on banks to consider their risk management strategy when using the system. Balancing risk and cost-efficiency is an inherent property of the system and can be done using the prototype.

8.2.3 Identity, data and process standards

Currently the system uses CVR as identifiers for corporate clients, which is the standard in Denmark. However, if the system was to operate with other blockchains in Europe, it would be better to use EUID which is the standard for business registration numbers in Europe. However, identity standards outside the boundaries of Europe are still to be considered. The data on the blockchain related to KYC is very sparse, since it only involves document collection fingerprint and an approval status. Therefore the banks can freely use current business standards for how KYC documents are ordered and what they should contain. However, our primary innovation is the concept of merging KYC processes, which is entirely new and not related to a business standard. If the system is successful it is likely to

become a new standard due to its benefits. However, the concept of merging documents also includes a lot of restrictions, such as not being able to subscribe or update KYC collections that have been merged. And not being able to merge with KYC collections that have been updated or subscribed to. More collaboration and insights from banks are still needed to figure out if the restrictions are feasible for the banks.

8.3 Validity of assumptions

One of the main assumptions is that the banks would be able to make a significant cost reduction to their current KYC process, and that the extra profits, along with the overall increased efficiency and customer satisfaction would be enough to persuade them to join the system. This also rests on the assumption that the core KYC process exists, which can only be truly confirmed by cooperating with the banks. If the core KYC process does not exist, we would have to rethink the entire pricing system and also how merging works. E.g. you would still only allow merges of KYC processes that are similar, with the same prices to ensure proportionality. It is pretty clear that FIs would be able to save money as the cost function shows that the cost of a KYC process decreases as the number of FIs increase, and it can never be higher than conducting the core KYC process themselves. However, the money saved needs to be less than the fee for entering the system. Since we have not computed the fee yet, it is still an open question. We also assume that the banks can come to agreement with respect to the price of a core KYC process (if it exists), which can only be verified once we have surveyed the banks. However, the assumption might hold if wages throughout the retail banking industry are generally very similar for KYC employees and managers. Then it will only hold if KYC employees from different banks generally spend the same amount of time to perform the core KYC process. If the points about salary and the time spent holds, it might not be difficult for the banks to reach an agreement regarding the price.

Moving forward we must test our assumptions about the core KYC process and the pricing. If the assumptions are incorrect we have two options: We could attempt to create a free market of KYC cooperation, or identify standard KYC procedures that can be uniformly priced across banks. The second approach would not require many changes. However, the free market approach poses a series of challenges. For example, merging could be more complex, since KYC processes would have different prices and possibly different contents. Another challenge is that it would be harder for banks to identify what they are buying without a lot of metadata. It would also be more complex to handle complaints, since it can be unclear what a KYC process is expected to contain.

9 Conclusion

Based on our analysis and highlighted points throughout the project we believe that the state of the current non-colluding banking system, and the heavily regulated KYC domain showcases an environment ripe to take advantage of KYC cooperation between banks. The analysis found that current KYC processes are ineffective, costly and if done incorrectly can lead to damage of brand and have big economical consequences. We have proposed CBB, which utilises distributed ledger technology to facilitate a shared KYC process. The primary innovation lies in the fact that banks can optionally choose how to balance between risk and cost-efficiency. The CBB solution can be used to reduce costs related to KYC by enabling banks to lessen the amount of KYC processes that a single corporate client experiences through their lifetime. Use of CBB should also be able to quicken the KYC process, as banks can choose to access already completed KYC processes depending on their internal risk and profit assessments.

We highlighted that blockchain solutions for bank enterprises must fulfill criteria such as ease of adoption but also the possibility to leave and change systems. Liability is also an important consideration with respect to fines and incorrect KYC processes. An important point is to increase incentive for banks to merge KYC processes, as the banks will reduce risks for fines but also cooperate to limit illicit customer activities. Our work rests on the assumption that a core KYC process exists, which is the bare minimum KYC requirement for all banks. This assumption still remains to be completely verified through cooperation with banks, which is the next step towards finalizing our product. We also need to convert our Ethereum prototype into a prototype that utilises a Hyperledger Fabric network.

We also found that merging KYC processes do put restrictions on subscribing and updating the individual KYC processes, which is why every bank can decide if merging should be possible or not for their own KYC processes. However, we argue that the incentive to merge KYC processes is strong.

10 References

Androulaki, E., Barger, A., Bortnikov, V., & Cachin, C. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *EuroSys '18: Proceedings of the Thirteenth EuroSys Conference*. 10.1145/3190508.3190538

Bjerregaard, E., & Kirchmaier, T. (2019). The Danske Bank Money Laundering Scandal: A Case Study. *Copenhagen Business School, CBS*. . https://doi.org/10.2139/ssrn.3446636

Business Change Team. (2020, November 10). *What Is CATWOE Analysis?* Business Change Academy. https://businesschange.academy/learn/stakeholder/what-is-catwoe-analysis/

Checkland, P. (2000). Soft Systems Methodology: A Thirty Year Retrospective. *Systems Research and Behavioral Science*, *17*, 11-58.

Checkland, P., & Scholes, J. (1999). *Soft Systems Methodology in Action*. John Wiley And Sons Ltd.

Danske Bank. (2021, March 19). *Annual report 2020*. Dansk Bank webpage. https://danskebank.com/da/news-og-insights/nyhedsarkiv/company-announcements/2021/sm04022021

Drgon, M., Georgiou, L., & Kiayias, A. (2020). Robust KYC via Distributed Ledger Technology. *The Third Toronto Fintech Conference*. 10.6084/m9.figshare.13301363

European Comission. (2021). *Do the data protection rules apply to data about a company?* European Comission Webpage. https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/application-regulation/do-data-protection-rules-apply-data-about-company_en

Finance Denmark. (2019, November). *Anti-money laundering and counter-terrorist financing in the Danish financial sector*. Report by Finance Denmark's Anti-Money Laundering Task Force. https://financedenmark.dk/media/40400/anti-money-laundering-and-counter-terrorist-financing-in-the-danish-financial-sector-print.pdf

Finans Danmark. (2021). *SIFI - Systemiske Instituter*. Finans Danmark webpage. https://finansdanmark.dk/aktuelle-emner/sifi-systemiske-institutter/

Gillespie, N. (2013). Restoring Trust in the Financial Services Sector. *University of Queensland, Industry and Parliament Trust*, 1-9. https://espace.library.uq.edu.au/data/UQ_313548/UQ313548_fulltext.pdf?Expires=1634034118&Key-Pair-Id=APKAJKNBJ4MJBJNC6NLQ&Signature=Ss3xkvZGvo0tf3vK0HqTYgidCWQ9HxmZpfKpUo5S-hs4QNAN4O1yZM1xLGod9705MKOFscqEDvSAjXs9vvVJsQhNvZALLx9A~UnRF-0x1L4a039QDENXMTp-Jk4x

Lacity, M., Steelman, Z., & Cronan, P. (2019). Towards Blockchain 3.0 Interoperability: Business and Technical Considerations. *Blockchain Center of Excellence White Paper Series*. https://cpb-us-e1.wpmucdn.com/wordpressua.uark.edu/dist/5/444/files/2019/05/BCCoEWhitePaper012019Open.pdf

Mayano, J. P., Ross, O., & Thoroddsen, T. (2019). Optimised and dynamic KYC system based on blockchain technology. *International Journal of Blockchains and Cryptocurrencies*, *1*(1), 85-106.

Mayoana, J. P., & Ross, O. (2017, November 15). KYC Optimization Using Distributed Ledger Technology. *Business & Information Systems Engineering volume*, *59*, 411-423. https://link.springer.com/article/10.1007/s12599-017-0504-2#citeas

Memminger, M., Baxter, M., & Lin, E. (2018). *Banking Regtechs to the Rescue?* Bain & Company webpage. https://www.bain.com/insights/banking-regtechs-to-the-rescue/

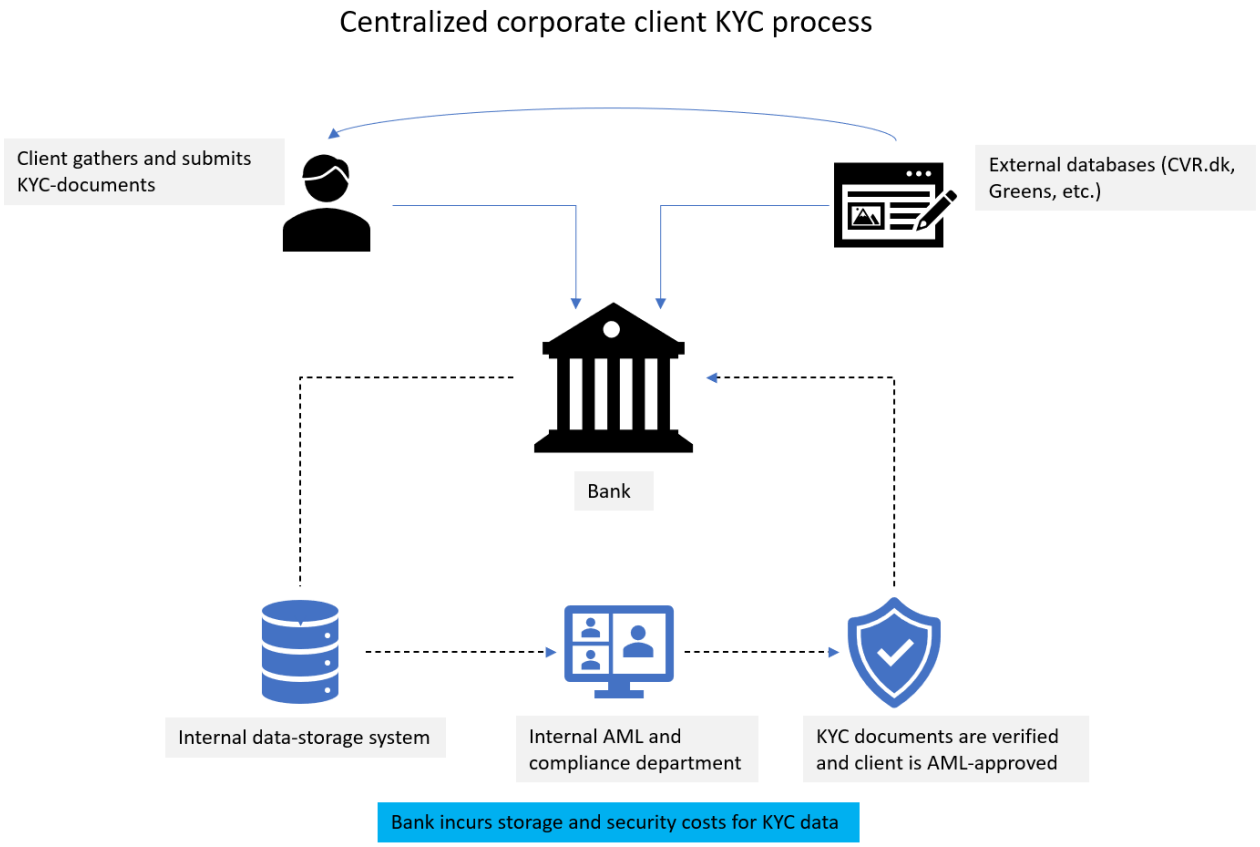
Osterwalder, A., & Pigneur, Y. (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*.

Thomson Reuters. (2016, May 16). *Thomson Reuters 2016 Know Your Customer Surveys Reveal Escalating Costs and Complexity*. Thomson Reuters webpage. https://www.thomsonreuters.com/en/press-releases/2016/may/thomson-reuters-2016-know-your-customer-surveys.html

11 Appendices

Appendix 1

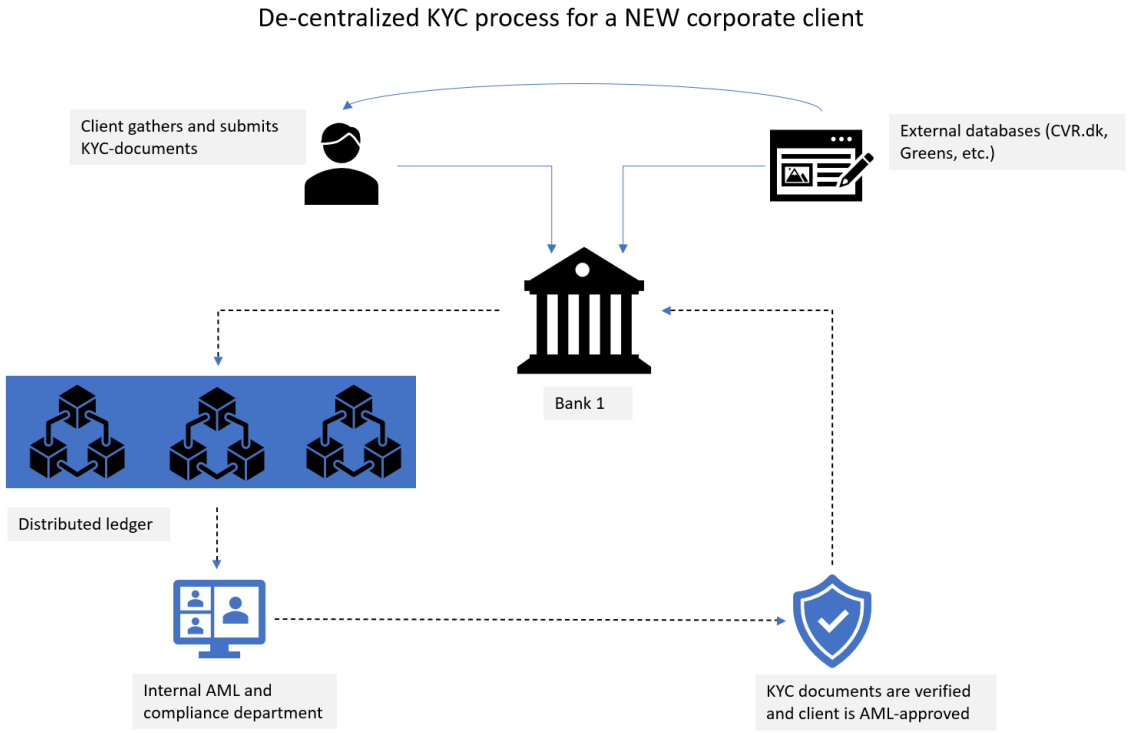
Current KYC scenario



Source: Worked out in PowerPoint based on 2 years of corporate banking experience within one of the largest SIFI banks in Denmark and Danske Bank (2021, March 19). *Annual report 2020*. Dansk Bank webpage. https://danskebank.com/da/news-og-insights/nyhedsarkiv/company-announcements/2021/sm04022021

Appendix 2

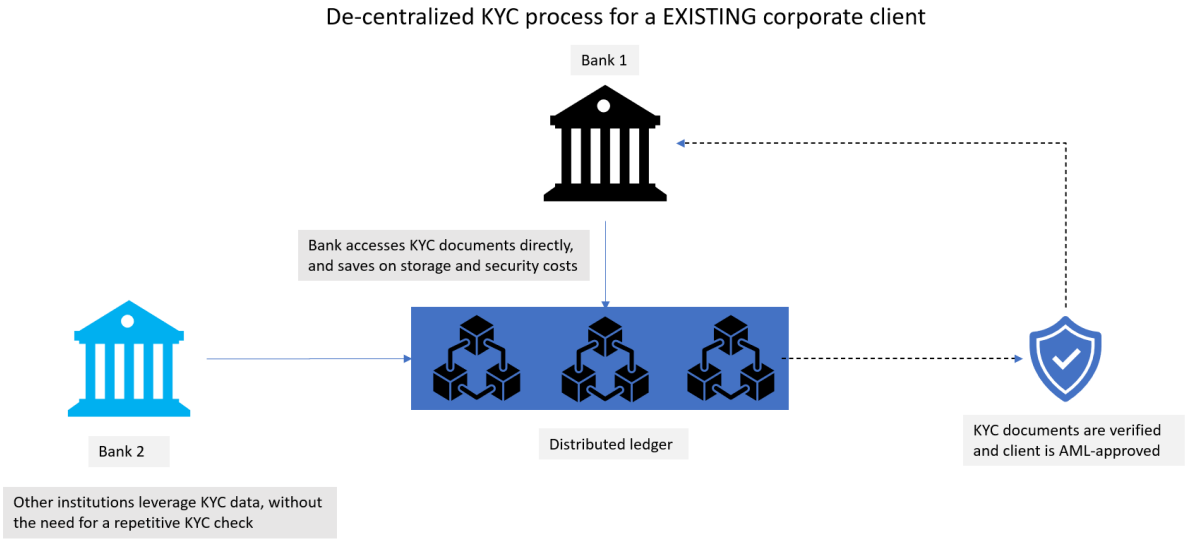
Future KYC scenario for a new corporate client



Source: Worked out in PowerPoint

Appendix 3

Future KYC scenario for a existing corporate client



Source: Worked out in PowerPoint

Appendix 4

Governance.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Governance
{
    address private MSP;
    address private creatorContract;

    modifier MSP_CHECK {
        require(msg.sender == MSP,
          "Must be MSP to use this function");
        _;
    }

    modifier MSP_OR_CREATOR_CHECK {
        require(msg.sender == MSP || msg.sender == creatorContract,
          "Must be MSP or creator contract to use this function");
        _;
    }

    //default maps to false.
    mapping (address => bool) FIs;
    constructor(address _MSP)
    {
        MSP = _MSP;
        creatorContract = msg.sender;
    }

    function addFI(address addr) MSP_CHECK public
    {
        FIs[addr] = true;
    }

    function removeFI(address addr) MSP_CHECK public
    {
        FIs[addr] = false;
    }

    function checkFI(address addr) MSP_OR_CREATOR_CHECK public
        view returns (bool value_)
    {
        value_ = FIs[addr];
    }
}
```

Source: Programmed using the open source web and desktop application Remix IDE https://remix.ethereum.org/

PayToken.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;


contract PayToken
{
    address private MSP;
    address private creatorContract;

    modifier MSP_CHECK {
        require(msg.sender == MSP,
          "Must be MSP to use this function");
        _;
    }

    modifier CREATOR_CHECK {
        require(msg.sender == creatorContract,
          "Must be creator to use this function");
        _;
    }

    mapping (address => uint) private balance;
    mapping (address => uint) private lockedBalance;

    constructor(address _MSP)
    {
        MSP = _MSP;
        creatorContract = msg.sender;
    }

    function showBalance(address addr)
        public MSP_CHECK view returns (uint value_)
    {
        value_ = balance[addr];
    }

    function showLockedBalance(address addr)
        public MSP_CHECK view returns (uint value_)
    {
        value_ = lockedBalance[addr];
    }

    function giveTokens(address addr,
        uint amount) public MSP_CHECK
    {
        balance[addr] += amount;
    }

    function removeTokens(address addr,
        uint amount) public MSP_CHECK
    {
        balance[addr] -= amount;
    }
```

```
55
56      function lockTokens(address addr,
57          uint amount) public MSP_CHECK
58      {
59          require(balance[addr] >= amount);
60          balance[addr] -= amount;
61          lockedBalance[addr] += amount;
62      }
63
64      function unlockTokens(address addr,
65          uint amount) public MSP_CHECK
66      {
67          require(lockedBalance[addr] >= amount);
68          lockedBalance[addr] -= amount;
69          balance[addr] += amount;
70      }
71
72      function transferTokens(address from,
73          address to, uint amount) public CREATOR_CHECK
74      {
75          require(balance[from] >= amount);
76          balance[from] -= amount;
77          balance[to] += amount;
78      }
79  }
```

Source: Programmed using the open source web and desktop application Remix IDE https://remix.ethereum.org/

KYCProcess.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;
import "../KYC/PayToken.sol";
import "../KYC/Governance.sol";

contract KYCProcess
{
    PayToken public payments;
    Governance public governance;
    address public MSP;
    uint public corePrice;
    uint public updatePrice;

    uint private nodeIDCounter;

    //Can add 10000 clients or KYC process a day for 1E70 years before running out of IDs.
    mapping (uint => Client) private clients;
    mapping (uint => KYCNode) private KYCNodes;
    mapping (uint => address[]) private participants;
    mapping (uint => address[]) private subscribers;
    mapping (uint => bytes32[]) private documentHash;

    enum NodeStatus{ Core, Update, Merge }

    struct KYCNode
    {
        uint nodeID;
        bool approved;
        NodeStatus status;
        uint update;
        uint updateOf;
        bool allowMerge;
        uint minParticipantsForMerge;
        uint maxParticipantsForMerge;
        bool merged;
        uint prevID;
        uint nextID;
    }

    struct Client
    {
        uint id;
        bool created;
        uint KYCNodeID;
        uint KYCProcesses;
        uint updates;
    }

    modifier MSP_CHECK {
        require(msg.sender == MSP,
          "Must be MSP to use this function");
        _;
    }
```

```
55
56      modifier FI_CHECK {
57          require(governance.checkFI(msg.sender),
58            "Must be registered FI to use this function");
59          _;
60      }
61
62      event RequestDocuments(address indexed _from, address indexed _to, uint _nodeID, bytes32 docHash);
63
64      constructor()
65      {
66          MSP = msg.sender;
67          payments = new PayToken(msg.sender);
68          governance = new Governance(msg.sender);
69          nodeIDCounter = 1;
70          corePrice = 10000000;
71          updatePrice = 5000000;
72      }
73
74      function getID() private
75          returns (uint)
76      {
77          uint retval = nodeIDCounter;
78          nodeIDCounter += 1;
79          return retval;
80      }
81
82      //Give customer ID
83      function clientExists(uint id) FI_CHECK public
84          view returns (bool)
85      {
86          return clients[id].created;
87      }
88
89      //Give KYC node ID
90      function nodeExists(uint id) private
91          view returns (bool)
92      {
93          return KYCNodes[id].nodeID != 0;
94      }
```

```
95
96      //give node id
97      //give address of FI to check if they participate or subscribe
98      function isParticipant(uint nid, address addr) public
99          view returns (bool)
100     {
101         require(nodeExists(nid));
102         bool retval = false;
103         address[] memory p = participants[nid];
104         for (uint i = 0; i < p.length; i+=1)
105         {
106             if (p[i] == addr)
107             {
108                 retval = true;
109             }
110         }
111         return retval;
112     }
113
114     function isSubscriber(uint nid, address addr) public
115         view returns (bool)
116     {
117         require(nodeExists(nid));
118         bool retval = false;
119         address[] memory s = subscribers[nid];
120         for (uint i = 0; i < s.length; i+=1)
121         {
122             if (s[i] == addr)
123             {
124                 retval = true;
125             }
126         }
127         return retval;
128     }
129
130     function addClient(uint cid) FI_CHECK public
131     {
132         require(!clientExists(cid));
133         clients[cid] = Client(cid,true,0,0,0);
134     }
```

```
135
136     //bytes32 example: 0x0000000000000000000000000000000000000000000000000000006d6168616d
137     //id example: any CVR
138     function addCoreKYCDocuments(uint cid, bytes32 hash, bool approved,
139         bool allowMerge, uint minParticipantsForMerge, uint maxParticipantsForMerge) FI_CHECK public
140     {
141         if (!clientExists(cid))
142         {
143             addClient(cid);
144         }
145
146         uint newNodeID = getID();
147         participants[newNodeID].push(msg.sender);
148         documentHash[newNodeID].push(hash);
149         //first KYC
150         if (clients[cid].KYCNodeID == 0)
151         {
152             KYCNodes[newNodeID] = KYCNode(newNodeID,approved,NodeStatus.Core,0,0,
153                                           allowMerge,minParticipantsForMerge,maxParticipantsForMerge,
154                                           false,0,0);
155             clients[cid].KYCNodeID = newNodeID;
156         }
157         //not first but new last
158         else
159         {
160             uint lastID = findLastNode(cid);
161             KYCNodes[lastID].nextID = newNodeID;
162             KYCNodes[newNodeID] = KYCNode(newNodeID,approved,NodeStatus.Core,0,0,
163                                           allowMerge,minParticipantsForMerge,maxParticipantsForMerge,
164                                           false,lastID,0);
165         }
166         clients[cid].KYCProcesses += 1;
167     }
```

```
168
169     function updateKYCDocuments(uint cid, uint nid, bytes32 hash, bool approved,
170         bool allowMerge, uint minParticipantsForMerge, uint maxParticipantsForMerge) FI_CHECK public
171     {
172         require(clientExists(cid), "Client does not exist");
173         require(nodeExists(nid), "Process does not exist");
174         require(!KYCNodes[nid].merged, "Node cannot be merged");
175         uint newNodeID = getID();
176         participants[newNodeID].push(msg.sender);
177         documentHash[newNodeID].push(hash);
178
179         uint lastID = findLastNode(cid);
180         KYCNodes[lastID].nextID = newNodeID;
181         uint updates = KYCNodes[nid].update;
182         KYCNodes[newNodeID] = KYCNode(newNodeID,approved,NodeStatus.Update,updates+1,nid,
183                                       allowMerge,minParticipantsForMerge,maxParticipantsForMerge,
184                                       false,lastID,0);
185         clients[cid].KYCProcesses += 1;
186         uint old_highest_update = clients[cid].updates;
187         if (old_highest_update < updates + 1)
188         {
189             clients[cid].updates = updates + 1;
190         }
191     }
192
193     function addMergeKYCProcess(uint cid, uint[] memory nids, bytes32 hash, bool approved) FI_CHECK public
194     {
195         updateMergeKYCProcess(cid,0,nids,hash,approved);
196     }
197
```

```
198     function subscribeKYCProcess(uint nid) FI_CHECK public
199     {
200         require(nodeExists(nid), "Process does not exist does not exist");
201         require(!isSubscriber(nid,msg.sender), "Already subscribed");
202         require(!isParticipant(nid,msg.sender), "Already participant");
203         require(!KYCNodes[nid].merged, "Cannot subscribe to merged");
204         uint[] memory priceList = priceKYCProcess(nid,msg.sender);
205         uint nodeID = nid;
206         for (uint i = priceList.length; i > 0; i-=1)
207         {
208             uint price = priceList[i-1];
209             uint n = subscribers[nodeID].length + participants[nodeID].length;
210             uint amount = price/n;
211             for (uint j = 0; j < subscribers[nodeID].length; j+=1)
212             {
213                 payments.transferTokens(msg.sender,subscribers[nodeID][j],amount);
214             }
215             for (uint j = 0; j < participants[nodeID].length; j+=1)
216             {
217                 payments.transferTokens(msg.sender,participants[nodeID][j],amount);
218                 emit RequestDocuments(participants[nodeID][j],msg.sender,nodeID,documentHash[nodeID][j]);
219             }
220             if (!isSubscriber(nodeID,msg.sender) && !isParticipant(nodeID,msg.sender))
221             {
222                 subscribers[nodeID].push(msg.sender);
223             }
224             nodeID = KYCNodes[nodeID].updateOf;
225         }
226
227     }
```

```
228
229     //add function to request documents that will throw event on blockchain?
230
231     function priceKYCProcess(uint nid, address addr) FI_CHECK public
232         view returns (uint[] memory priceList_)
233     {
234         require(nodeExists(nid), "Process does not exist does not exist");
235         uint nodeID = nid;
236         uint level = KYCNodes[nodeID].update + 1;
237         uint[] memory priceList = new uint[](level);
238         while (level > 0)
239         {
240             uint price = priceKYC(nodeID);
241             if (!isSubscriber(nodeID,addr) && !isParticipant(nodeID,addr))
242             {
243                 priceList[level-1] = price;
244             }
245             else
246             {
247                 priceList[level-1] = 0;
248             }
249             nodeID = KYCNodes[nodeID].updateOf;
250             level -= 1;
251         }
252         priceList_ = priceList;
253     }
254
255     //buy price of node
256     function priceKYC(uint nid) FI_CHECK private
257         view returns (uint)
258     {
259         require(nodeExists(nid), "Process does not exist does not exist");
260         uint nodeID = nid;
261         uint level = KYCNodes[nodeID].update;
262         uint cost = updatePrice;
263         if (level == 0)
264         {
265             cost = corePrice;
266         }
267         uint r = participants[nodeID].length;
268         uint p = participants[nodeID].length + subscribers[nodeID].length;
269         return (r*cost)/(p+1);
270     }
```

```
271
272     function updateMergeKYCProcess(uint cid, uint nid, uint[] memory nids,
273         bytes32 hash, bool approved) FI_CHECK public
274     {
275         require(clientExists(cid), "Client does not exist");
276         require(nids.length>0,"Must merge with other processes");
277         uint update = KYCNodes[nid].update; //nid = 0 does not exist, and will default to zero.
278         if (nid != 0)
279         {
280             update += 1;
281         }
282         for (uint i = 0; i < nids.length; i+=1)
283         {
284             uint nodeID = nids[i];
285             KYCNode memory node = KYCNodes[nodeID];
286             require(nodeExists(nodeID), "A Process does not exist");
287             require(node.allowMerge, "A Process does not allow merge");
288             require(node.approved == approved, "All process approval status must match");
289             require(node.updateOf == nid, "A Process is of a different branch");
290             require(node.update == update, "A Process is of a different update level");
291             require(!node.merged, "A Process cannot be merged more than once"); //Maybe?
292             require(node.minParticipantsForMerge <= nids.length + 1,
293                 "Too few processes for merge");
294             require(node.maxParticipantsForMerge >= nids.length + 1,
295                 "Too many processes for merge");
296             require(subscribers[nodeID].length == 0, "A Process cannot have subscribers");
297         }
298         uint newNodeID = getID();
299         for (uint i = 0; i < nids.length; i+=1)
300         {
301             uint nodeID = nids[i];
302             KYCNodes[nodeID].merged = true;
303             participants[newNodeID].push(participants[nodeID][0]);
304             documentHash[newNodeID].push(documentHash[nodeID][0]);
305             for (uint j = 0; j < nids.length; j+=1)
306             {
307                 uint otherID = nids[i];
308                 if (nodeID != otherID)
309                 {
310                     emit RequestDocuments(participants[otherID][0],
311                                           participants[nodeID][0],otherID,
312                                           documentHash[otherID][0]);
313                 }
314             }
315             emit RequestDocuments(participants[nodeID][0],msg.sender,
316                                   nodeID,documentHash[nodeID][0]);
317             emit RequestDocuments(msg.sender,participants[nodeID][0],
318                                   newNodeID,hash);
319         }
320         participants[newNodeID].push(msg.sender);
321         documentHash[newNodeID].push(hash);
322
323         uint lastID = findLastNode(cid);
324         KYCNodes[lastID].nextID = newNodeID;
325
326         KYCNodes[newNodeID] = KYCNode(newNodeID,approved,NodeStatus.Merge,
327                                       update,nid,false,0,0
328                                       ,false,lastID,0);
329         clients[cid].KYCProcesses += 1;
330     }
```

```
331
332     function listKYCProcesses(uint cid, uint updates) FI_CHECK public
333         view returns ( uint[] memory nodeIDs_ )
334     {
335         require(clientExists(cid), "Client does not exist");
336         require(clients[cid].KYCProcesses > 0,
337             "Client have zero KYC processes");
338         require(clients[cid].updates >= updates,
339             "No KYC process has been updated that many times");
340
341         uint[] memory nodeIDs = new uint[](clients[cid].KYCProcesses);
342         uint curNodeID = clients[cid].KYCNodeID;
343         uint nextNodeID = KYCNodes[curNodeID].nextID;
344         uint i = 0;
345         while (curNodeID != 0)
346         {
347             if (KYCNodes[curNodeID].update == updates)
348             {
349                 nodeIDs[i] = curNodeID;
350                 i += 1;
351             }
352             curNodeID = nextNodeID;
353             nextNodeID = KYCNodes[curNodeID].nextID;
354         }
355         uint[] memory retval = new uint[](i);
356         for (uint j = 0; j < i; j+=1)
357         {
358             retval[j] = nodeIDs[j];
359         }
360         nodeIDs_ = retval;
361     }
362
363     function fetchClient(uint cid) FI_CHECK public
364         view returns ( Client memory client_ )
365     {
366         require(clientExists(cid), "Client does not exist");
367         Client memory client = clients[cid];
368         client_ = client;
369     }
370
371     function fetchKYCProcess(uint nid) FI_CHECK public
372         view returns ( KYCNode memory node_ )
373     {
374         require(nodeExists(nid), "Process does not exist");
375         KYCNode memory node = KYCNodes[nid];
376         node_ = node;
377     }
378
379     function fetchParticipants(uint nid) FI_CHECK public
380         view returns ( address[] memory participants_ )
381     {
382         require(nodeExists(nid), "Process does not exist");
383         address[] memory participant = participants[nid];
384         participants_ = participant;
385     }
```

```
386
387     //give client id
388     function findLastNode(uint id) private
389         view returns (uint)
390     {
391         require(clientExists(id), "Client does not exist");
392         uint nodeID = clients[id].KYCNodeID;
393         bool found = false;
394         if (nodeID == 0)
395         {
396             found = true;
397         }
398         while (!found)
399         {
400             uint nextID = KYCNodes[nodeID].nextID;
401             if (nextID == 0)
402             {
403                 found = true;
404             }
405             else
406             {
407                 nodeID = nextID;
408             }
409         }
410         return nodeID;
411     }
412 }
```

Source: Programmed using the open source web and desktop application Remix IDE https://remix.ethereum.org/