Tailored IoT & BigData Sandboxes and Testbeds for Smart, Autonomous and Personalized Services in the European Finance and Insurance Services Ecosystem

D4.8 – Permissioned Blockchain for Finance and Insurance - II

Revision Number	3.0
Task Reference	T4.3
Lead Beneficiary	UBI
Responsible	Konstantinos Perakis
Partners	ENG GFT HPE IBM INNOV SIA UBI UNIC
Deliverable Type	Report (R)
Dissemination Level	Public (PU)
Due Date	2021-05-31
Delivered Date	2021-06-30
Internal Reviewers	FBK, HPE
Quality Assurance	INNOV
Acceptance	WP Leader Accepted and Coordinator Accepted
EC Project Officer	Pierre-Paul Sondag
Programme	HORIZON 2020 - ICT-11-2018
	This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

Contributing Partners

Partner Acronym	**Role**[1]	**Author(s)**[2]
UBI	Lead Beneficiary	Konstantinos Perakis, Dimitris Miltiadou
INNOV	Contributor	John Soldatos, Nikos Kapsoulis, Antonis Litke
IBM	Contributor	Fabiana Fournier, Inna Skarbovsky
FBK	Internal Reviewer	Bruno Lepri
HPE	Internal Reviewer	Giovanni Giuliani
INNOV	Quality Assurance	Filia Filippou

Revision History

Version	Date	Partner(s)	Description
0.1	2021-04-07	UBI	ToC Version
0.2	2021-04-15	UBI	Initial contributions on Section 4 and 5
0.3	2021-04-21	UBI, INNOV, IBM	Contributions on Section 2, 3, 4 and 5
0.4	2021-04-29	UBI, INNOV, IBM	Updated contributions on Sections 2, 3, 4, 5 and 6
0.5	2021-05-11	UBI	Updated contributions on Sections 2, 3, 4, 5 and 6
0.6	2021-05-11	UBI	Finalisation of section 2, 3, 4, 5 and 6
0.7	2021-05-17	UBI	Consolidation of the first version
1.0	2021-05-18	UBI	First Version for Internal Review
1.1	2021-05-24	FBK	Internal Review
1.2	2021-05-24	HPE	Internal Review
2.0	2021-05-25	UBI	Version for Quality Assurance
3.0	2021-06-30	UBI	Version for Submission

[1] Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance

[2] Can be left void

Executive Summary

The document at hand, entitled "D4.8 - "Permissioned Blockchain for Finance and Insurance - II", constitutes a report of the efforts and the produced outcomes of Task T4.3 "Distributed Ledger Technologies for Decentralized Data Sharing" of WP4. Towards this end, the main objective of this deliverable is to deliver the updated documentation that will supplement the information documented in the deliverable D4.7, focusing on the updates in the design specifications of the INFINITECH Blockchain applications, the documentation of the technical details of their first prototype version and the updates on the INFINITECH Blockchain network that hosts these applications. The deliverable builds on top of the outcomes and knowledge extracted in D4.7 in order to provide the updated report of the work performed until M20.

Hence, the scope of the current report can be described in the following axes:

- Present the results of the comprehensive analysis of the key characteristics and offerings of the blockchain technology, as well as the definition of the role of the blockchain technology within the INFINITECH RA. In this analysis, the key characteristics of the blockchain technology and the different implementations based on different approaches are presented. The role of the blockchain technology within the INFINITECH RA is formulated based on the results of this analysis. Although the results remained unchanged from the previous iteration, they are included in the deliverable for coherency reasons.
- Document the updated design specifications of the blockchain applications which are implemented within the context of the INFINITECH project. More precisely, two INFINITECH Blockchain applications are presented, namely the Consent Management and the Know-Your-Customer (KYC) / Know-Your-Business (KYB) applications. For each application, the updated documentation of the addressed business operation, the exploited key functionalities of the blockchain technology and the high-level architecture of the solution are presented.
- Present the technical specifications of the first prototype version of the Consent Management and of the Know-Your-Customer (KYC) / Know-Your-Business (KYB) blockchain applications. For each application, the technical details of the implemented services and functions, as well as the interactions between the services, are documented. Furthermore, the source code structure of each application is presented.
- Document the updated design details of the INFINITECH blockchain network. The details of the updated INFINITECH blockchain network are presented providing the updated documentation of the network topology, defining the role of each node, the services and key blockchain components hosted on each node and the interactions between the nodes.
- Present the list of baseline technologies and tools, composed by dominant open-source software, libraries and frameworks, which are exploited for the deployment of the INFINITECH blockchain network and the INFINITECH Blockchain applications. The list remained unchanged from the previous iteration of the deliverable.

The deliverable constitutes the second iteration of the report of the work performed within the context of T5.4. The outcomes of this deliverable will be used as the basis for the implementation of the upcoming version of the INFINITECH Blockchain applications which will be delivered in deliverable D4.8 on M27. As the development of the blockchain applications is a living process that will last until M27 as per the INFINITECH Description of Action, the final iteration of this deliverable will report the derived optimisations and enhancements from the analysis of the collected feedback from the pilots of the project and the stakeholders of the platform.

Table of Contents

List of Figures

List of Tables

Abbreviations/Acronyms

Abbreviation	Definition
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AML	Anti-Money Laundering
API	Application Programming Interface
BFT	Byzantine Fault-Tolerant
CA	Certificate Authority
CFT	Crash Fault-Tolerant
CRUD	Create Read Update Delete
DLT	Distributed Ledger Technology
DoA	Description of Action
GDPR	General Data Protection Regulation
IoT	Internet of Things
KYB	Know-Your-Business
KYC	Know-Your-Customer
M	Month
MSP	Membership Services Provider
RDBMS	Relational Database Management System
RA	Reference Architecture
UUID	Universally Unique Identifier

1 Introduction

The scope of deliverable D4.8 "Permissioned Blockchain for Finance and Insurance - II" is to document the efforts undertaken within the context of T4.3 "Distributed Ledger Technologies for Decentralized Data Sharing" of WP4. The deliverable D4.8 is prepared in accordance with the INFINITECH Description of Action and constitutes the second iteration of the work performed under Task 4.3. The current document provides the updated documentation related to the specifications of the permissioned blockchain infrastructure that is exploited in the INFINITECH platform, along with the updated design specifications and the implementation details of the blockchain applications that will be developed on top of this infrastructure within the context of the project.

As documented also in the first iteration of the deliverable, the blockchain technology constitutes one of the main ingredients of the INFINITECH platform. To this end, the consortium leverages the tremendous potential of the blockchain technology towards the realisation of blockchain empowered scenarios for the financial and insurance sectors. Through the effective use of the blockchain technology, the delivered solutions enable the financial institutions to utilise innovative solutions with a twofold purpose. At first, the delivered solutions enable the financial institutions to optimise and enhance their current core operational services and processes towards the significant cost reductions and their increased performance. Secondly, the delivered solutions facilitate the design and implementation of new innovative services, products and offerings to their clients.

The deliverable is building directly on top of the preliminary results that were documented in deliverable D4.7, in which the appropriate blockchain infrastructure was designed and integrated in the INFINITECH Reference Architecture (INFINITECH RA) and a set of blockchain-enabled solutions was designed on top of this blockchain infrastructure, to deliver the first prototype version of these blockchain-enabled solutions in order to be exploited by the INFINITECH project's pilots, as well as the stakeholders of the financial sector.

1.1 Objective of the Deliverable

The purpose of this deliverable is to report the outcomes on the work performed within the context of Task 4.3 at this phase of the project (M20). This deliverable constitutes the second iteration, hence its main focus is to document the advancements in comparison with the first iteration and in particular the delivery of the first prototype version of the blockchain applications whose design specifications were documented in the first iteration. To this end, the deliverable provides the supplementary documentation of the blockchain applications by presenting their implementation details for this first prototype version, while also providing the updated documentation of the blockchain infrastructure and the baseline technologies that are leveraged.

The deliverable aims at providing the updated documentation of the information that has been documented in the previous iteration and, for coherency reasons, it contains the information included in the previous iteration, highlighting the updates and optimisations that were introduced where needed. In more detail, the main objective of the current deliverable is to document the updates on the design specifications, as well as to provide the implementation details of the delivered INFINITECH blockchain applications. In addition to this, the deliverable documents the updates introduced in the INFINITECH blockchain network.

The revised information is presented utilising the approach that was followed in the previous iteration. Hence, the results of the thorough analysis of the key characteristics and offerings of the blockchain technology, as well as the role of the blockchain technology within the INFINITECH RA, which remained unchanged from the previous iteration, are presented. Following the role definition of the blockchain technology within the INFINITECH RA, the INFINITECH blockchain applications are presented. At first, the updated design specifications of each application are presented, highlighting the optimisations introduced where applicable. For each application, the addressed business case, the main characteristics of the blockchain technology which are leveraged and the high-level architecture of the application are presented. The detailed use cases that each application addresses, as well as the related sequence diagrams are also documented. Moreover, the deliverable introduces the detailed documentation of the implementation of the first prototype versions of the blockchain applications. For each application, the implemented components are documented providing the technical details of their implemented functions.

Furthermore, the deliverable presents the updated documentation of the INFINITECH blockchain network which is deployed and leveraged by the implemented blockchain applications. Finally, the deliverable documents the baseline technologies and tools which are leveraged for the implementation of the INFINITECH blockchain applications.

It should be noted that according to the INFINITECH Description of Action Task 4.3 lasts until M27, and therefore, the final version of the deliverable will be released on M27 with deliverable D4.9. To this end, the forthcoming iteration of the deliverable will constitute the final documentation of the work performed and will include all the necessary updates and optimisations that will be introduced on the INFINITECH blockchain applications and on the INFINITECH blockchain network taking into consideration the evolvement of the project and the blockchain technology, as well as the feedback that will be received by the pilots and the stakeholders of the project.

1.2 Insights from other Tasks and Deliverables

The deliverable D4.8 is released in the scope of WP4 "Interoperable Data Exchange and Semantic Interoperability" activities and documents the preliminary outcomes of the work performed within the context of T4.3 "Distributed Ledger Technologies for Decentralized Data Sharing". The task is tightly interconnected with the outcomes of WP2 "Vision and Specifications for Autonomous, Intelligent and Personalized Services" in which the overall requirements of the INFINITECH platform are defined. In detail, the outcomes of Task 2.1, as presented in deliverable D2.1 and D2.2 that reported the collected user stories of pilots of the project and the extracted user requirements, are provided as input in T4.3. Furthermore, the specification of the technologies that constitute the fundamental building blocks of the INFINITECH platform and the elicited technical requirements that are linked to these building blocks, as reported by the outcomes of T2.3 in deliverable D2.5 and D2.6, are also provided as input in T4.3.

Last but not least, the outcomes of T2.7, that formulated the INFINITECH Reference Architecture (INFINITECH RA) and that serve as the blueprint for the development, deployment and operation of Big Data, AI and IoT in the finance and insurance sectors, are directly related with the work performed in this task, as the reported outcomes of this report related to the blockchain network and the blockchain applications are integral parts of the INFINITECH platform. Finally, the work reported in this deliverable is tightly connected with the work performed in T4.4 and T4.5 of WP4 as the output of Task 4.3, and especially the designed blockchain network serves as the basis in the activities of both T4.4 and T4.5.

1.3 Structure

This document is structured as follows:

- Section **Error! Reference source not found.** introduces the document, describing the context of the outcomes of the work performed within the task and highlights its relation to the rest of tasks and deliverables of the project.

- Section 2 provides the results of the analysis of the blockchain technology, presents the key characteristics and main components of the technology, and defines the role of the blockchain technology in the INFINITECH RA.
- Section 3 presents the details of a proposed capability of the blockchain platform which will further extend the capabilities of the blockchain platform.
- Section 4 presents the updated designed specifications of the blockchain applications, the use cases addressed by each application, and the corresponding sequence diagrams of each use case. Furthermore, it documents the technical details of the implemented blockchain applications.
- Section 5 documents the updated details of the blockchain network which is leveraged by the blockchain applications, by presenting the updated network topology along with the updated list of services of each node and their interactions.
- Section 6 presents the list of baseline technologies and tools utilised in the implementation of the described network and applications.
- Section 7 concludes the document.

2 The Blockchain technology

Updates from D4.7:

The particular section remained unchanged from the previous version. It presents the key characteristics and offerings of the blockchain technology while also defining the role of the blockchain technology within the INFINITECH Reference Architecture.

2.1 An overview of the blockchain technology

Blockchain is a distributed digital ledger of cryptographically signed transactions that are grouped into blocks, which in turn are cryptographically linked to each other after validation and undergoing a consensus decision. The addition of new blocks increases the tamper resistance of the older ones and are replicated across the copies of the ledger within the network, resolving automatically any possible conflicts through a set of established rules [1]. In this sense, blockchain is a continuously growing, distributed, shared ledger of uniquely identified, linked transaction records organised in blocks that are sealed cryptographically with a digital fingerprint generated by a hashing function and are sequentially chained through a reference to their hash value [2]. Blockchain technology became extremely popular as the basis of cryptocurrencies as well as its implementations such as Bitcoin and Ethereum, which are digital currencies that were designed to work as medium of exchange incorporating secure and verifiable cryptographically signed transactions and controlled cryptocurrency unit generation.

In general, the blockchain technology is composed of multiple technologies related to cryptography, peer-to-peer networks, identity management, network security, transaction processing, (distributed) algorithms and more, that are all leveraged in order to formulate an immutable transaction ledger which is maintained by a distributed network of peer nodes formulating the blockchain network. The key characteristics of the blockchain technology can be grouped as follows [3]:

- *Decentralised:* One of the core characteristics of the blockchain is its decentralised and distributed nature across multiple number of nodes (peers) that provides extensibility, scalability, confidentiality, flexibility and resilience to attacks or misuse.
- *Immutable:* Any transaction record is immutable and reserved forever. Hence, full transactional history is maintained and all records are cryptographically secure. This fact safeguards that the underlying data cannot be tampered and are attestable.
- *Transparent:* The transaction data that formulate the blocks are transparent to each node and each node can introduce an update, based on a set of rules, increasing the transparency and trustworthiness of the technology.
- *Autonomy*: One of the core characteristics of the blockchain is the autonomy offered within the peer network that is regulated by the consensus protocols, where each node can safely and securely transfer and update data. Since the ledger (or actually a copy of the ledger) is shared among multiple nodes (peers), the transparency and trustworthiness are also increased.
- *Open Source*: The blockchain technology is an open source technology with multiple blockchain implementations and variations being available; sustained by various communities and ecosystems, that can be leveraged upon needs.

At a high level, the blockchain technology exploits well-established computer network mechanisms and cryptographic primitives such as cryptographic hash functions, digital signatures, asymmetric-key cryptography, certificate authority mixed with record keeping concepts (such as append only ledgers) [1]. Nevertheless, the blockchain technology has a set of key concepts that includes the distributed ledger that is composed by blocks containing the transaction records, the smart contracts or chaincode and the consensus model.

The heart of the blockchain is the distributed ledger in which all transaction records that are published within the blockchain network are stored in the form of blocks. Being by nature decentralized, the technology takes advantage of both the distributed ownership and the distributed physical architecture of a distributed ledger. Each peer maintains its own copy of the ledger, ensuring that is synced and updated with the same data. As the blockchain network is designed and is operating in a peer-to-peer mode, the blockchain network has an increased resilience to the loss of any node. Every new transaction is checked and verified among all peers before it is accepted and inserted into the ledger and it is referenced to the previous block, enabling an integrity check of invalid transmitted transactions or nodes. Finally, with the utilisation of the cryptographic mechanisms the distributed ledger is tamper evident and tamper resistant.

The nodes that are participating in the blockchain network can be characterised as publishing and non-publishing nodes. The candidate transactions are submitted to the blockchain network via its user through the interacting applications and services. However, it is the role of the publishing node(s) only to publish a block in the blockchain network that contains these transactions via the gossip data dissemination protocol, once they have been validated and authenticated, that will be received by the rest of the (publishing and non-publishing) nodes which in turn will validate and authenticate the received block and accept it in order to be inserted finally in the ledger.

To facilitate all the operations performed in the ledger within the blockchain network, the smart contracts (or chaincode) are leveraged. Smart contracts are the trusted distributed applications that are deployed within the nodes of the blockchain network and encapsulate the business logic of the blockchain applications. Smart contracts include the agreements that the participants of the blockchain network have formulated with regards to the generation of new facts that are added to the ledger and that will update the current and historical state of the facts that are already stored in the ledger. In this sense, the smart contracts enable the creation of new transactions by the users of the blockchain network by invoking the smart contracts' functions. The smart contracts facilitate the controlled access to the ledger, offering a layer of abstraction on top of the aspired transactions, encapsulating and simplifying all the relevant information while also ensuring their compliance with the underlying legal agreements, as well as the automation of the several aspects of the transactions. The implementation and execution of smart contracts varies depending on the blockchain implementation with most popular cases being Ethereum's smart contracts and Hyperledger Fabric's chaincode.

One of the most critical concepts of blockchain technology is the consensus model that is utilised in order to validate a transaction and to keep the ledger transactions synchronized across the blockchain network. Hence, the consensus model undertakes the validation and approval of the candidate transactions and ensures that the copies of the ledger, that are kept within the nodes of the blockchain network, are updated with the same transactions and in the same order. As the blockchain network is composed of multiple nodes, it is very likely that many publishing nodes will compete at the same time to publish new nodes. Additionally, conflicts might be created by nodes publishing new blocks at approximately the same time. Hence, it is evident that a method is required to ensure that transactions will be written to the ledger at the same order as they generated, as well as that malformed or malicious transactions are rejected. For this reason, the blockchains depending on their implementation specifications exploit different consensus models that are available in computer science such as the CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) ordering, while at the same time large research effort is spent on this topic towards the definition of further alternative consensus models capable of better addressing this issue with less trade-offs.

The blockchain implementations can be characterised and grouped into two major high-level categories based on the permission model applied on the blockchain network, the *permissionless* *blockchain networks* and the *permissioned blockchain networks*. Permissionless blockchains are based on open and public blockchain networks where anyone is capable of publishing new blocks or read the blocks of the blockchain anonymously and without granting any permission from any authority. Hence, the implementation of the permissionless blockchains dictates that they are open and available to anyone and anyone can issue new transactions in new blocks and read the transactions included in the existing blocks, thus write and read the ledger. To prevent the malicious usage of the blockchain, these implementations employ a consensus model that requires from the participants to expend or maintain resources through a "mined" native cryptocurrency or through transaction fees when it comes to publishing new blocks. The most common consensus models employed are the "proof of work" or "proof of stake" that are rewarding the participants of new blocks that conform the consensus protocol with a native cryptocurrency. The well-established examples of permissionless blockchains are Bitcoin and Ethereum.

On the other hand, the permissioned blockchain networks are regulated blockchain networks where only authorised users, by a specific authority as defined within the network specifics, are able to maintain the underlying blockchain, while read access and publishing of new transactions are also restricted and regulated. In this sense, the permissioned blockchain networks are formulated only by a set of known, identified and verified participants whose access rights and roles are regulated by an agreed governance model defined by the participants of the networks providing a certain degree of trust and security for all generated transaction records. As the identities of the participants of the network are known and trusted, the consensus models that are employed for publishing new blocks do not require the expense or maintenance of resources. In permissioned blockchain networks the consensus models exploited are usually faster and less computationally expensive, as the mining operations are eliminated and more traditional consensus protocols are adopted, such as the CFT or BFT protocols. Permissioned blockchain networks allow the tight control and protection of the underlying blockchain, and the level of trust between each participant of the network can be reflected on the consensus model that will be used or regulated by the access rights to the data that each participant can obtain. Furthermore, the authorisation of each participant can be revoked in the case of misuse or withdrawal of trust.

Blockchain technology has an enormous potential that has been noticed by several industries that are looking forward to exploit its various advantages and offerings in order to introduce new services, products and offerings to their clients or to rejuvenate their internal processes and legacy systems towards a better performance, reduction of financial costs and increase of trust between the partners involved in business transactions. The mostly adopted area is the cryptocurrencies area where Bitcoin and Ethereum are the most notable cases. Additionally, blockchain is adopted in financial services, insurance services, supply chain, energy trading, sales, digital music, anti-counterfeiting, domain name services and videogames, among others. For the financial and insurance services in particular, the blockchain technology has found many potential use cases for providing blockchain-enabled banking and insurance services that optimise various back-office processes, removing various intermediaries and disrupting various operational processes towards the financial cost reductions and the expansion of the portfolio of offered services to their customers.

2.2 The role of blockchain technology in INFINITECH RA

Within the INFINITECH Reference Architecture (INFINITECH RA), as presented in deliverable D2.13, the blockchain technology has a dual presence and can be exploited in different ways, depending on the scope of the use case that INFINITECH RA aims to address.

Hence, on the one hand, blockchain can be considered as an additional data source type at the infrastructure layer, from which data are accessed and collected with the use of the respective sophisticated data collection mechanisms. On the other hand, blockchain can be considered as a cross-cutting service positioned in the central layer of the INFINITECH RA, in which decentralised applications tailored to the needs of the financial and insurance sectors can be developed and exploited. The blockchain-enabled decentralised applications can be utilised to apply use cases that can optimise, and even revolutionize, core operational processes of the financial or insurance institutions, decreasing their costs and increasing radically their performance and efficiency by reducing or eliminating the need for manual processing or manipulation, while at the same time augmenting their level of trust. Within the context of the WP4, the focus is on the development of such decentralised applications that will showcase the potentials of the blockchain technology.

As with many other industrial sectors, financial and insurance sectors are highly regulated with strict legislations and processes in which security and trust are fundamental aspects. While the blockchain technology is considered as the dominant candidate to disrupt all of the financial and insurance sectors' processes, as it is promising to mitigate the cost of trust and increment the security level in these processes and even business models [4], not all blockchain implementations are deemed as ideal candidates. The reason for this lies on the specific characteristics offered by the two major approaches followed in the blockchain technology, the permissionless blockchain and the permissioned blockchain.

While the permissionless blockchain, with public open networks to which anyone can participate and interact in an anonymous manner, has been adopted in the case of cryptocurrencies, when it comes to more enterprise-oriented use cases, such as the banking institutions or other financial institutions, different requirements arise and are related mainly to the privacy and confidentiality of the data, as well as the underlying business or financial transactions stored in the blockchain, the regulated and strictly controlled access to the blockchain network, the performance of the network with high throughput and low latency, and most importantly the hard requirement of the identifiable and pre-approved identity of the participants of the blockchain network. For all these reasons, among the two major approaches, only the permissioned blockchain technology is considered as the appropriate candidate solution and will be exploited within the INFINITECH RA.

Towards this end, the consortium decided to exploit the Hyperledger Fabric open source enterprise-grade permissioned distributed ledger technology (DLT) platform [5] that is one of the most active projects of the Hyperledger project founded by the Linux Foundation. Hyperledger Fabric has been designed specifically for enterprise use and delivers a set of key differentiating capabilities over other popular distributed ledger or blockchain platforms. One of the main differentiations of Fabric is its highly modular and configurable architecture that promotes the innovation, versatility and optimization for a broad range of industry use cases including banking, finance and insurance [6]. Based on its modular and configurable architecture, Fabric guarantees a high level of confidentiality, resiliency, flexibility, and scalability.

Through its pluggable ordering service, consensus can be achieved with multiple implementations, such as the CFT or BFT and more, based on the requirements of a specific use case or deployment. Fabric offers a private and permissioned blockchain network where all participants can be enrolled based on their cryptographic entities, through a set of pluggable trusted membership service providers that are supported, and can be tailored to the needs of the deployment. Fabric provides its own ordering service implementation named Raft. Raft ordering service is a CFT that is based on an implementation of Raft protocol in the etcd distributed key-value store and it constitutes the first step toward Fabric's development of a BFT ordering service.

Although Hyperledger Fabric's Raft ordering service is CFT based on an implementation of Raft protocol in etcd, it constitutes the first step toward Fabric's development of a byzantine fault tolerant (BFT) ordering service.

In Fabric, smart contracts, referred as chaincode, are operating in an isolated container environment, such as Docker, and can be written in standard programming languages, such as Go and Node.js. Smart contracts offer the required interfaces that are exploited by applications outside of the blockchain network in order to interact with distributed ledger providing the required level of abstraction, as well as increased level of privacy and confidentiality. To further promote privacy and confidentiality, Fabric enables the creation of channels in which the participants own a separate ledger of transactions from the rest of the blockchain network that is visible only to the participants of the channel. Finally, it provides the feature of private data, where collections of data can only be visible and accessible to a portion of the participants of a specific channel.

It is acknowledged that the combination of all these key differentiating capabilities, sets Fabric as one of the best performing platforms in transaction processing and transaction confirmation latency platforms. Its pluggable architecture enables its exploitation in a variety of different use cases of the financial and insurance sectors that are characterized as highly complex and restrictive sectors.

Towards this end, to address the needs of the financial and insurance sectors, a new blockchain capability which will extend the existing capabilities is proposed and a set of trusted distributed applications in the form of chaincode will be developed. The aim of the new blockchain capability will enhance the blockchain for the needs of these sectors. On the other hand, the design distributed applications will address core use cases of the financial and insurance sectors exploiting the benefits of the permissioned blockchain technology. In this sense, the designed distributed applications will effectively leverage the underlying permissioned blockchain infrastructure provided by Fabric that is designed in accordance to the needs and requirements of the stakeholders of the specific sectors. The details of the proposed blockchain capability, that is currently under formulation, are described in Section 3. The design specifications of the designed distributed applications are presented in detail in Section 4 of the current deliverable, while the details of the designed underlying permissioned blockchain network that these distributed applications will be deployed, are documented in Section 5.

3 INFINITECH Blockchain Capabilities

Updates from D4.7:

The work on GDPR compliance in blockchain has not progressed in the elapsed time since the submission of D4.7. It seems that efforts in implementation in this direction won't be done in the scope of the INFINITECH project.

Blockchain based solutions are applications built on top of the selected blockchain platform technology exploiting the provided tools of the blockchain platform, such as smart contracts and client SDKs. Generally speaking, when some use cases require a certain capability currently not existing in the selected blockchain platform, two main approaches can be followed to cope with the specific requirement. The first approach includes the extension of the current platform capabilities with additional ones that are suitable for the needs of the use case, while the second approach requires the design and implementation of a solution at the application layer that the scenarios requiring the specific capability can leverage, if possible.

In the scope of the INFINITECH project complying with the General Data Protection Regulation (GDPR) [7] is an example of such needed capability as described henceforth.

3.1 Addressing GDPR in Blockchain

3.1.1 Motivation

One of the blockchain's stated benefits includes its immutability, i.e. once the data is written on the ledger it cannot be changed or deleted. This assures complete trustworthiness of the ledger's content and supports provenance and non-repudiation use cases. Nevertheless, this seemingly comes in contrast with "the right to be forgotten" which is a basic principle of GDPR.

Highly regulated sectors, such as healthcare, banking, and insurance, need to support GDPR privacy articles, such as a "right to be forgotten", and therefore, they require an approach which can allow the removal of personal data from the chain on demand without violating the blockchain's consistency.

It became evident at the early stages of the project that providing such capability can benefit the aspired INFINITECH scenarios. The aim is to provide such functionality "horizontally" at the level of the blockchain platform and not at the application level, so applications requiring this capability could be built on top of this "extended" platform. Such an approach implies a differentiation from the current Fabric implementation as described in the following sections. The following subsection presents a high-level overview and the design behind the proposed solution.

3.1.2 Description of the solution

Many approaches have been proposed in order to deal with this problem (e.g. [8] and [9]). Some approaches rely on not storing personal data on the blockchain at all, the data is stored off-chain and only the hashes of the data on-chain. Such an approach, while solving the problem of putting personal data on immutable ledger, poses other problems such as performance penalties for complete end-to-end latencies, and consistency and availability challenges in case of multiple administrative domains for this data. Other models have been proposed, based on the fact that encrypted data cannot be retrieved within a reasonable period of time without proper authorization, and therefore, concluding that it can be stored directly in the blockchain. This notion is not robust enough to guarantee the GDPR compliance for the right to be forgotten in systems where the data may be used for more extended periods of time. Forward secrecy is unclear at the quantum computing age.

Our proposed solution is based on changing the transaction content stored at the level of the block, the transaction envelope content, and the way transactions are validated at the Hyperledger Fabric infrastructure level.

The main steps of the approach are as follows (Figure 1):

- Store values and their hashes in the ledger
- Construct block hash from the values' hashes only (not from the values themselves)
- When validating a transaction, ensure hash matches the computed value's hash
- Keep the actual values alongside the signed transaction
- Introduce a new type of transaction – the *redaction transaction*

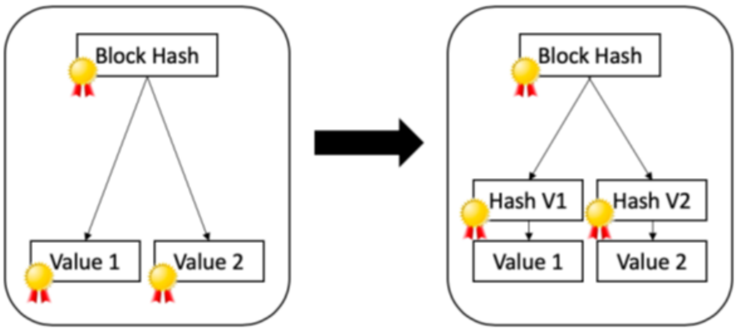

Figure 1: Storing hashes and values on the ledger

The redaction transaction would redact the value itself (set bits to zero), not its hash. The redaction does not entail a change in the value's hash nor in the hash-chain (which is only dependent on the hash value of the original value and not on the value itself). This way the consistency of the chain and accountability is preserved (see Figure 2).

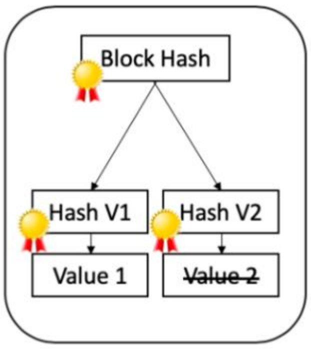

Figure 2: Reduction of transaction value

3.1.3 Use Cases

The following sections provide the initial documentation of all the use cases encapsulated in this extension, describing in detail the information of each use case.

3.1.3.1 Use Case GDPR-1: New transaction submission

As a blockchain client, I would like to submit a transaction proposal to a blockchain supporting value reduction and make sure the data is properly written on the ledger.

Table 1: Use Case GDPR-1: New transaction submission

Stakeholders involved:	Transaction invoker
Pre-conditions:	1. The requesting party is a legal end-user in the network and has write access
Post-conditions:	1. The transaction is written on the blockchain, the data is available for reading
Data Attributes	1. A value to write
Normal Flow	1. The authentication and access roles for the requester are determined 2. In case write access is allowed, the hash of the value being stored on the blockchain is calculated 3. The value and the hash are stored on the blockchain, namely hash as part of the transactions read write set, while pre-image value in a separate structure
Pass Metrics	1. The transaction is available on the chain; value can be read
Fail Metrics	1. Requested information could not be written: • the invoker has no access

3.1.3.2 Use Case GDPR-2: Value reduction

As a blockchain client, I would like to redact a value previously stored by me on the ledger.

Table 2: Use Case GDPR-2: Value reduction

Stakeholders involved:	Transaction invoker
Pre-conditions:	1. The requesting party is a legal end-user in the network and has write access and has previously written a value
Post-conditions:	1. The redacted values (bits set to zero) are written on the blockchain, the original hash is kept
Data Attributes	1. A value to redact
Normal Flow	1. The authentication and access roles for the requester are determined 2. In case write access is allowed, the existence of the value to redact is checked 3. The redacted value is stored on chain
Pass Metrics	1. A value is redacted, and a notification is returned to the invoker
Fail Metrics	1. Value could not be redacted: • the invoker has no access • the requested value does not exist

3.1.3.3 Use Case GDPR-3: Read redacted value

As a blockchain client, I would like to make sure that an attempt to read the value after reduction will return the redacted (all bits “0”) and not the original value.

Table 3: Use Case GDPR-3: Read redacted value

Stakeholders involved:	Any query invoker
Pre-conditions:	1. The requesting party is a legal end-user in the network and has read access
Post-conditions:	1. The redacted value (bits set to zero) and not the original value is returned
Data Attributes	1. The value to read
Normal Flow	1. The authentication and access roles (read) for the requester are determined 2. In case read access is allowed, the redacted transaction payload is read (bits set to zero) and returned to the invoker
Pass Metrics	1. The redacted value is returned to the invoker
Fail Metrics	1. Requested query could not be performed: • the invoker has no access • the value does not exist

4 INFINITECH Blockchain Applications

Updates from D4.7:

The particular section documents the necessary updates related to the advancements on the implementation of the INFINITECH Blockchain applications. In detail, the following documentation is introduced per blockchain application:

- *Consent Management: The implementation details of the first version of the Consent Management (Section 4.1.4).*
- *Know Your Customer (KYC) / Know Your Business (KYB): The implementation details of the first version of the KYC/KYB (Section 4.2.4).*
- *Tokenization: The updated short documentation of the work performed for the specific use case, as well as the details of the future work (Section 4.3.1).*

The benefits of the blockchain technology are leveraged with the development of trusted distributed applications that are deployed within the nodes of the blockchain network. In the blockchain technology, the trusted distributed applications are developed as smart contracts or chaincode in the Fabric terminology. The role of a chaincode is to generate the new facts that will be added to the ledger, which maintains all the facts related to the current and historical state of a set of business objects, in order to introduce the appropriate changes in both the current and historical state. In this sense, the chaincode defines the transaction logic that is responsible for the evolution of the state of the business objects. Furthermore, the chaincode provides a set of interfaces that are utilised by the applications outside of the blockchain network in order to interact with the distributed ledger.

Hence, in the design of any blockchain-enabled solution that effectively addresses the needs of a specific business operation, process or service, the main components included are the external to the blockchain network application and the chaincode that is deployed on the blockchain network. In the design specifications of both components, a specific business logic is encapsulated, whose aspects are clearly defining the permitted and required operations which are executed for a use case of the business operation.

With regards to the design of the chaincode, it is usually written in the GO programming language and it explicitly defines the governance rules for any type of business object utilised in the use case. The structuring of the source code of the chaincode can vary as there is no strict norm of how the source code should be constructed, rather than best practises formulated by the blockchain development communities. However, the basis of the source code is the definition of the business objects on top of which this chaincode will perform all the operations and the functions that will undertake the execution of these operations. Thus, as for every specific business operation different business objects are defined, the whole chaincode is tailored to the needs of each business operation.

Nevertheless, within the context of the INFINITECH project and specifically Task 4.3, the developed chaincode will adhere to the following structuring of the source code that is based on the logical schema of the Data Processing Components as defined in the deliverable D2.5:

- *Blockchain Reader*: The main purpose of this component is to enable the fetching of the requested data from the blockchain ledger. Depending on the underlying use case, the design specifications may vary. However, the context of the functions of the component will remain the same.
- *Blockchain Writer*: The main purpose of this component is to facilitate the submission of new transactions to the blockchain ledger. As with the blockchain reader, the design specification for each use case may vary, nevertheless the context of the functions of the component will remain unaffected.
- *Smart Contract Executor*: The main purpose of this component is to encapsulate the business logic of the designed use case and execute the smart contracts on the blockchain ledger. Hence, the

design specifications of each use case are tailored to the needs of the aspired operation and service and they will orchestrate the use case execution.

- *Blockchain Authenticator*: The main purpose of this component is to perform the authentication of the blockchain network user in order to grant access to a specific channel of the blockchain network. The implementation of the specific component is almost generic and will be utilised across all the implemented use cases.
- *Blockchain Encryptor:* The main purpose of this component is to perform the encryption of the data that are involved and produced within the smart contract execution utilising the AES 256 encryption. The implementation of the specific component is almost generic and will be utilised across all the implemented use cases.
- *Blockchain Decryptor:* The main purpose of this component is to perform the decryption of the data that were encrypted by the Blockchain Encryptor. Again, the implementation of the specific component is almost generic and will be utilised across all the implemented use cases.

In the following sections, the design specifications of the blockchain applications that will be developed within the context of Task 4.3, are presented in detail. In total, two different blockchain applications are presented. For these two applications, namely the Consent Management and the Know-Your-Customer (KYC) / Know-Your-Business (KYB), a thorough description of the addressed business operation is presented, highlighting the use of the blockchain technology on each application accompanied by the high-level architecture of the application. Furthermore, the details of each specific use case of the business operation, that is supported by the blockchain application, is presented. Finally, for each use case of the business operation, the corresponding sequence diagram that illustrates the interactions between the involved stakeholders and the components is documented. The section concludes with a short description of the application related to tokenization use case that is currently formulated within the context of Task 4.4. However, the thorough documentation of this specific use case will be documented within the context of the deliverable D4.10 in accordance with the INFINITECH Description of Action (DoA).

It should be noted that the list of applications that will be presented might be further expanded with additional applications, as well as that the existing applications might be further enriched with additional uses cases and functionalities, as the project evolves and new requirements may arise as the result of the feedback that will be collected by the pilots of the project and the stakeholders of the platform. These updates will be reported in the next iterations of the current deliverable, as planned by the Description of Action of the INFINITECH project.

4.1 Consent Management

Updates from D4.7:
The updates on this iteration involve the Implementation Section (4.1.4) that includes analysis on the current framework and illustrates the realization of the conceptualization of the use case as defined in the previous sections. In addition to this, the paragraph in Section 4.1.1 with the indicative functions, as documented in the previous iteration, has been removed, as it is now covered by the newly introduced Section 4.1.4. Finally, it contains small updates on Table 4 and Table 9.

4.1.1 Description of the solution

Collaborative data sharing between customers, banks and other organizations, enables application of advanced analytics over particular datasets and intelligent support tools for better understanding customers and their financial relationships, thus being critically important in today's financial markets. However, the development of intelligence support tools which will support new customer services that solve business problems, such as improved Know-Your-Customer (KYC) processes and consequently Anti Money Laundering (AML), credit scoring and fraud detection services that can be built upon them, is highly dependent on the customers permission to share data. As a result, the requirement for a trusted and secure sharing mechanism of customer consent arises. In this sense, banks also identify granular permission consent as a key enabler of trust which is vital to maximise data sharing and ensure customers are comfortable with sharing data.

In this blockchain application, we aim at exploiting the blockchain technology and specifically the permissioned blockchain in order to develop a decentralised and robust consent management mechanism, that will enable the sharing of the customers' consent to exchange and utilise their customer data across different banking institutions. Blockchain technology, and its latest advancements, appears as a compelling technology to overcome the underlying challenges of trust improvement due to its decentralised nature and immutability, as well as the impossibility of ledger falsification. The integrity of customer data processing consents and their immutable versioning control are protected by the blockchain infrastructure. The blockchain-enabled consent management mechanism will enable the financial institutions to effectively manage and share their customers' consents in a transparent and unambiguous manner, enabling them to inform the customers at any time about:

a) any customer data they are managing upon their consent
b) the status of their consent (active or revoked/withdrawn)
c) the recipients (financial institutions or peers) of their customer data upon their consent
d) the purpose (or even legal basis) and time period of their customer data sharing to the recipient (financial institutions or peers)

In the same transparent and unambiguous manner, the customers of the financial institution will:

a) be constantly informed for all the requests for sharing their customer data
b) be able to activate or revoke their consents
c) be constantly aware of the active consents they have given to each specific recipient

The blockchain-enabled consent management mechanism guarantees the integrity of the consents through the usage of the blockchain technology, as well as cryptographic techniques and digital signatures incorporated in it. Besides the consent records and their integrity, the blockchain technology will be used to securely maintain the consent history providing the complete consents' versioning. Through the blockchain technology, immutable versioning control is provided that is capable of obtaining the latest version of the consent, as well as the previous versions of the consent and their valid periods, in an indisputable manner. Thus, the blockchain is capable of storing the consents and their complete update history in a secure and trusted manner. In this way, both the financial institutions, as well as their customers are protected as both the consents' integrity and validity periods are secured.

With regards to the consents and their validity period, two different approaches are considered. The first approach provides the ability of an "once off" consent, in which the validity period of the consent is a predefined period. Once this period expires, the consent is automatically revoked and new consent (or an updated consent) is required in order for the recipient to be able to access the customer's data. An example of the cases considered for this consent type is the customer's consent for sharing of KYC data between two financial institutions (banks) for the scenario of a loan origination / application or an account opening. The second approach provides the ability of a permanent (or "regular") consent that has an infinite validity period and it is only invalidated if the consent is withdrawn. An example of the cases considered for this consent type is the Peer-to-Peer (including Person to Person, Person to Organisation, Organisation to Organisation cases) customer data sharing consent in which a customer utilises an interface of a mobile application to select a set of its specific customer data (such as specific accounts, specific transactions or alerts) that they would like to share with an individual person or a group of persons. It should be noted that the consent management mechanism does not save or distribute any customer data. Customer data are exchanged using the respective secured APIs of the financial institution. Instead, the blockchain based consent management mechanism maintains the minimum information that is required in order to formulate a consent agreement between the customer and the respective financial institution.

Figure 3 depicts a high-level architecture of the proposed solution. As depicted, the core elements of the proposed solution are the *Consent Management System, the File Storage and the Blockchain infrastructure*, while the key stakeholders that are involved and are interacting with the proposed solution are the *internal financial institution* that collects and has access to its customer data, the *customer of internal financial institution* whose data are collected by the internal financial institution, and finally the *external financial institution* or peer that aspires to obtain the data of the customer of internal financial institution upon the formulation of a valid and legitimate consent that is formulated between the three parties.

In this architecture, the Consent Management System is the mediator between the involved parties. It receives and processes the requests for a consent formulation from the external financial institution or peer to the internal financial institution, and consequently the customer of the internal institution. By interacting with all three involved parties and upon acceptance of the details and terms of the consent by all of them, the final consent receipt is formulated and stored in digital form within the File Storage. At the last and most crucial step, the minimum information that is required from the formulated consent form is entered in the blockchain infrastructure by invoking the deployed chaincode that is responsible for the generation of a new transaction in the underlying ledger. Additionally, the Consent Management System, is consulting and retrieving the information stored in the blockchain infrastructure, by invoking the deployed chaincode in order to formulate an access control decision depending on the existence and validity of a consent when access to customer data via the internal financial institution's APIs is requested or to retrieve the complete history of consents for a customer upon request.

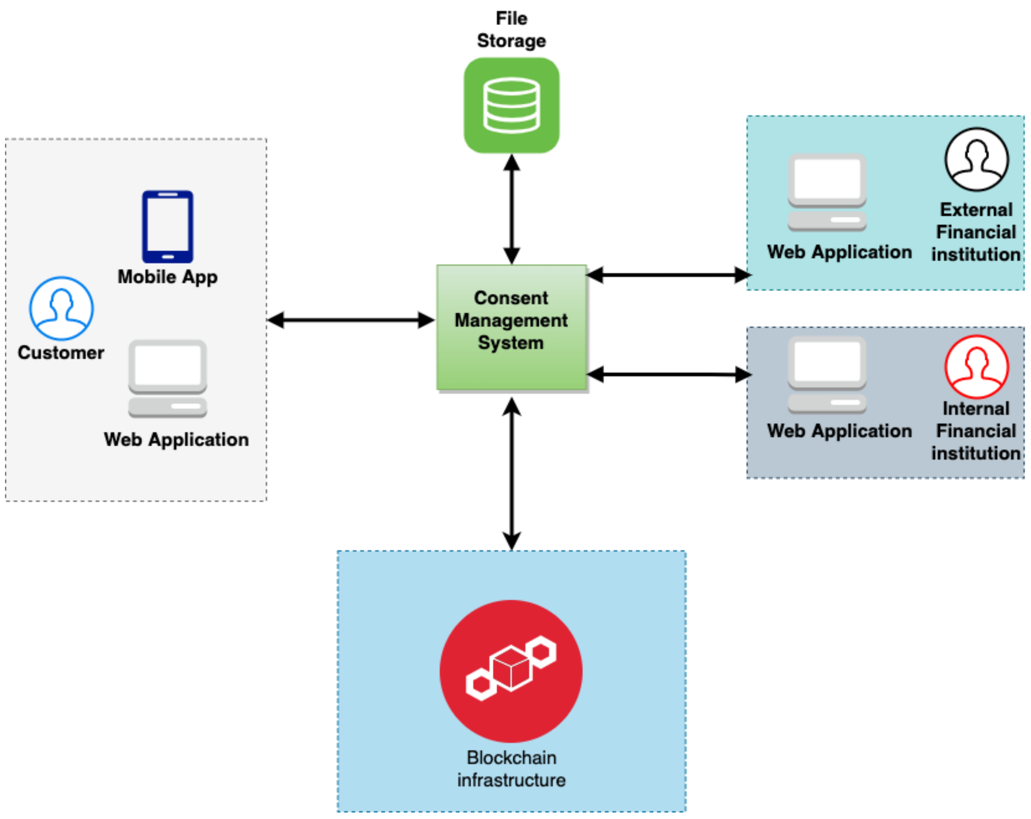
File Storage
Mobile App
Customer
Web Application
Consent Management System
Web Application
External Financial institution
Web Application
Internal Financial institution
Blockchain infrastructure

Figure 3: High-level architecture of the Consent Management solution

One of the core aspects of the design specifications is the definition of the business objects for which the current and historical state will be maintained and updated through the functions of the chaincode. To this end, for the Consent Management application, the initial data schema which defines the core business objects has been defined. The definition was based on the Consent Receipt specification that is proposed by the Kantara Initiative [10] which has been adapted in terms of terminology in order to be aligned with the EU GDPR legislation. Figure 4 depicts the initial data schema of the Consent Management application, while the details of all the entities are documented in Table 4 to Table 7.

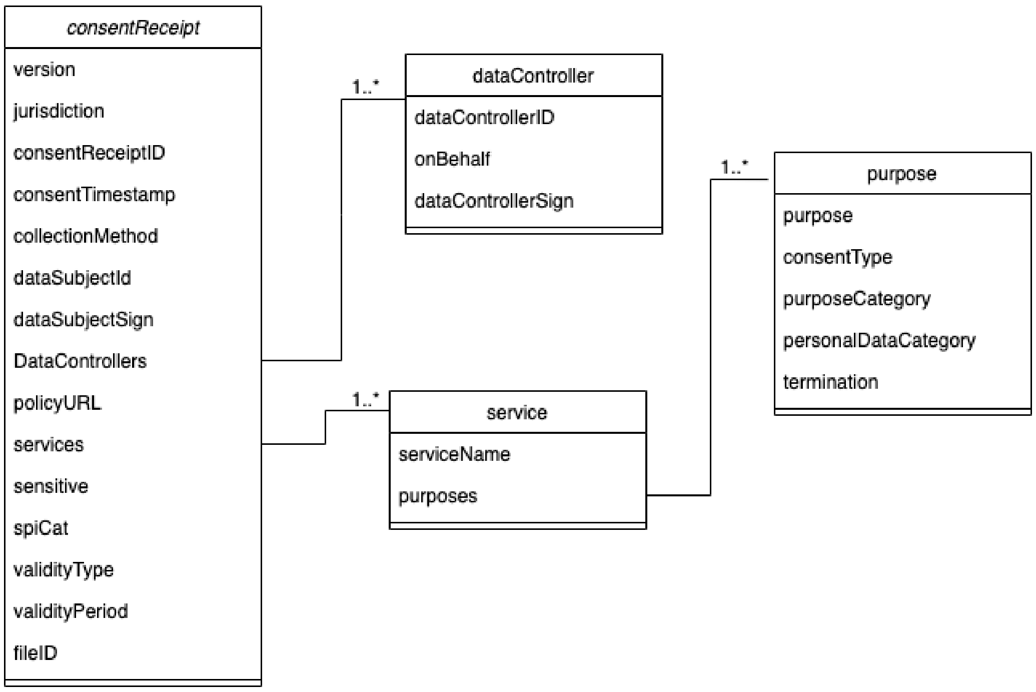

Figure 4: Consent Management Data Schema diagram

Table 4: Consent Management Data Schema details (1)

consentReceipt		
Name	**Type**	**Short description**
version	String	REQUIRED: The version of the consent specification to which a receipt conforms
jurisdiction	String	REQUIRED: The jurisdiction(s) applicable to this transaction i.e. EU and/or any EU-national
consentReceiptID	String	REQUIRED: The unique identifier of each Consent Receipt in UUID-4 format
consentTimestamp	String	REQUIRED: Date and time of the consent transaction in ISO 8601 format
collectionMethod	String	REQUIRED: A description of the method by which consent was obtained, i.e. the Consent Management System
dataSubjectID	String	REQUIRED: The unique identifier of the Data Subject in UUID-4 format
dataSubjectSign	String	REQUIRED: The Data Subject's digital signature in base64 format.
dataControllers	Array [dataController]	REQUIRED: An array of dataController items (see Table 5)
policyURL	String	REQUIRED: A link to the DataController's privacy statement/policy and applicable terms of use in

		effect when the consent was obtained, and the receipt was issued.
services	Array [service]	REQUIRED: An array of Service items (see Table 6)
sensitive	Boolean	REQUIRED: Indicates whether the consent interaction contains personal data that is designated sensitive or not sensitive
spiCat	Array [categories]	REQUIRED: A listing of categories where personal data collected is sensitive. The array must contain the categories if sensitive is TRUE
validityType	String	REQUIRED: The validity type of the consent, either ONCE_OFF or PERMANENT.
validityPeriod	String	REQUIRED: Date and time of the consent expiration in ISO 8601 format. Required when validityType is ONCE_OFF.
fileID	String	REQUIRED: A reference to the file stored in the File Storage
status	String	REQUIRED: Represents the current consentReceipt status. Possible values are ACTIVE, WITHDRAWN, EXPIRED

Table 5: Consent Management Data Schema details (2)

dataController		
Name	**Type**	**Short description**
dataControllerID	String	REQUIRED: The unique identifier of the data controller in UUID-4 format.
onBehalf	Boolean	OPTIONAL: True if a data processor is acting on behalf of a data controller.
dataControllerSign	String	REQUIRED: The Data Controller's digital signature in base64 format.

Table 6: Consent Management Data Schema details (3)

service		
Name	**Type**	**Short description**
serviceName	String	REQUIRED: The service or group of services being provided for which personal data is collected. The ID of the service for which consent for the collection, use, and disclosure of personal data is being provided.
purposes	Array [purpose]	REQUIRED: An array of Purpose items (see Table 7)

Table 7: Consent Management Data Schema details (4)

purpose		
Name	**Type**	**Short description**
purpose	String	OPTIONAL: A short, clear explanation of why the personal data is required
consentType	String	REQUIRED: The type of the consent used by the Data Controller as their authority to collect, use or disclose personal data. The accepted values are EXPLICIT or IMPLICIT.
purposeCategory	String	REQUIRED: The reason the Data Controller is collecting the personal data. The acceptable values are based on the (CISWG) Wiki page [11].

personalDataCategory	String	REQUIRED: A list of defined personal data categories. Personal data category should reflect the category that will be shared as understood by the Data Subject. The acceptable values are based on the (CISWG) Wiki page [11].
termination	String	REQUIRED: Conditions for the termination of consent. Link to policy defining how consent or purpose is terminated.

4.1.2 Use cases

The following sections provide the detailed documentation of all the use cases encapsulated in this blockchain application, describing in detail all information of each use case.

4.1.2.1 Use case CMS-1: Register Customer in the Consent Management System

In order for a customer to be able to receive consent requests, it is mandatory that they are registered in the Consent Management System, creating their profile that will be used in order to receive consent requests. The customer profile shall include the minimum customer discovery and communication details required in order to receive a consent request. The profile information and any consequent private information is not disclosed to any interested party and is not saved in the blockchain infrastructure.

Table 8: Consent Management Use Case CMS-1

Stakeholders involved:	Customer, Internal Financial Institution
Pre-conditions:	A customer willing to provide his/her consent to share his/her customer data upon his/her approval.
Post-conditions:	A customer is registered in the Consent Management System, his /her profile is created and is able to receive consent requests from the Internal Financial Institution.
Data Attributes	Required data based on the Consent Management Data Schema.
Normal Flow	1. The customer (or Internal Financial Institution on behalf of the customer) registers to the Consent Management System filling in the required information explained in the Data Attributes above 2. The customer provides his / her approval to receive new consent requests
Pass Metrics	1. Customer profile is available in the Consent Management System
Fail Metrics	1. No customer profile is available in the Consent Management System

In the same manner, the external financial institution or the peer registers in the Consent Management System in order to be able to initiate consent requests to a customer of the Internal Financial Institution.

Table 9: Consent Management Use Case CMS-1 (2)

Stakeholders involved:	External Financial Institution, Internal Financial Institution
Pre-conditions:	An External Financial Institution interested in obtaining the consent of customer in order to access his/her customer data upon his/her approval
Post-conditions:	An External Financial Institution is registered in the Consent Management System, his /her profile is created and is able to initiate consent requests to Internal Financial Institution.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The External Financial Institution (or Internal Financial Institution on behalf of the External Financial Institution) registers to the Consent Management System filling in the required information explained in the Data Attributes above
Pass Metrics	1. External Financial Institution profile is available in the Consent Management System
Fail Metrics	1. No External Financial Institution profile is available in the Consent Management System

4.1.2.2 Use Case CMS-2: Customer receives a request to provide new consent for sharing his/her customer data

The stakeholder wishing to gain access to the customer data issues a new consent request to the Consent Management System specifying the details of the requested customer data (i.e. specific data, period of usage). The customer receives a notification with all the required information for the consent request in a proper way that it allows him/her to review the request.

Table 10: Consent Management Use Case CMS-2

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	1. A customer profile is available in the Consent Management System capable of receiving new consent requests 2. A new request for customer data is issued by a banking institution or financial organisation to the Internal Financial Institution
Post-conditions:	The customer is notified for the new consent request in order to review and decide about giving his/her consent or denying the access.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. Internal Financial Institution generates the new request for consent to the customer including all the required information described in the data attributes above.

	2. Customer receives the new request in a proper format (e.g., through his/her mobile phone app or a web-based interface provided by the Internal Financial Institution) so that he/she can review and decide whether to provide his/her consent or deny the access to his/her customer data.
Pass Metrics	1. The customer is informed for the new consent request and is able to formulate a decision.
Fail Metrics	1. The customer is not informed for the new consent request

4.1.2.3 Use Case CMS-3: Definition of the consent

The customer reviews and possibly alters the details and conditions of the consent request before formulating his/her decision to give his/her consent or deny the access. In the case of approval, the final consent is defined by the customer and it is submitted to the Consent Management System. In the case of the denial, the request is blocked and the interested party is informed and the processing is finished.

Table 11: Consent Management Use Case CMS-3

Stakeholders involved:	Customer, Internal Financial Institution
Pre-conditions:	A new consent request from an interested party is provided to the customer.
Post-conditions:	The customer formulates the consent form that is ready to be signed by both parties.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The customer reviews, edits and finalises the consent form details (based on the data attributes above). He/she is able to check and alter the proposed attributes and details of the request in a user-friendly way (e.g., through his/her mobile phone app or a web-based interface provided by the Internal Financial Institution). 2. **In case of approval:** The customer provides his/her consent form to the Consent Management System in order to be signed. 3. **In case of denial:** The customer denies the request; the interested stakeholder is informed and the process is finished.
Pass Metrics	1. **In case of approval:** The consent form is ready to be signed by both parties. 2. **In case of denial:** The interested stakeholder is informed and the process is finished.
Fail Metrics	1. **In case of approval:** No consent form is available 2. **In case of denial:** The process is still open

4.1.2.4 Use Case CMS-4: Signing of the consent by the interested parties

Once the consent form is ready, the Consent Management System provides it to both parties (the customer and the interested stakeholder) in order to be digitally signed. The Consent Management System collects the digitally signed consent form from both parties.

Table 12: Consent Management Use Case CMS-4

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A valid consent form from a customer is available.
Post-conditions:	The signed consent form from both parties.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The consent form is sent to the customer by the Consent Management System (e.g., through his/her mobile phone app or a web-based interface provided by the Internal Financial Institution) in order to be digitally signed. 2. The customer provides the digitally signed consent form to the Consent Management System. 3. The consent form is sent to the interested party by the Consent Management System in order to be digitally signed. 4. The interested party provides the digitally signed consent form to the Consent Management System.
Pass Metrics	1. The Consent Management System obtains the digitally signed consent form from both parties.
Fail Metrics	1. The Consent Management System cannot obtain the digitally signed consent form from both parties.

4.1.2.5 Use Case CMS-5: Consent form is entered into blockchain

Once the digitally signed consent form is available, the Consent Management System interacts with the blockchain infrastructure in order to create a new transaction that will be introduced in the blockchain.

In the case of the “once off” consent, where a specific validity period is defined, the Consent Management System is internally handling the validation of consent time period by creating and monitoring the specific timer in order to perform the validation of consent time period. Use Case 1.9 describes the handling performed when the consent time period expires.

Table 13: Consent Management Use Case CMS-5

Stakeholders involved:	N/A

Pre-conditions:	The digitally signed consent form from both parties is available.
Post-conditions:	A new transaction containing all the information of the consent form is available in the blockchain infrastructure.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The Consent Management System retrieves the digitally signed consent form from both parties. 2. The Consent Management System interacts with the blockchain infrastructure and enters the new consent form in the ledger in the form of a new transaction. 3. In the case of the "once off" consent the proper timer is started in the Consent Management System.
Pass Metrics	1. The new transaction containing all the information of the consent form is available in the blockchain infrastructure.
Fail Metrics	1. No new transaction is available in the blockchain infrastructure.

4.1.2.6 Use Case CMS-6: Consent update or withdrawal

The consent form can be updated or withdrawn at any time by the customer side. In the case of the update, the previously described steps Use Case 1.3 to Use Case 1.5 are re-executed and the status of the consent remains "active". On the other hand, the withdrawal of the consent is internally translated to "invalidation" of the smart contract and the status of the consent is set to "withdrawn". In the case of the "once off" consent, the associated timer is restarted when the consent is updated. On the other hand, when the consent is withdrawn the associated timer is stopped.

In all these processes, the consent history is maintained in the blockchain infrastructure with the context of the smart contract. Through the transactions all versions of the consents and their complete update history is maintained.

Table 14: Consent Management Use Case CMS-6

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A transaction with status active containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The customer initiates the consent update process by defining the consent form (in the case of withdrawal the corresponding status is defined). 2. The Consent Management System retrieves the new consent form and sends it to both the customer and the interested stakeholders.

	3. The consent form is digitally signed by both parties and provided to the Consent Management System. 4. The Consent Management interacts with the blockchain infrastructure and introduces the updates with a new transaction. 5. In the case of the “once off” consent, the timer is either restarted (update) or stopped (withdrawal) in the Consent Management System.
Pass Metrics	1. A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Fail Metrics	1. There is no new transaction with updated consent information.

4.1.2.7 Use Case CMS-7: Access Control based on the consent forms

During a data access request to the underlying data management system, the data management system is consulting the Consent Management System in order to validate the consent status between the requesting party and the customer whose data are requested. By utilising the transactions formulated in the previous steps, the Consent Management System is able to formulate the access control decision.

Table 15: Consent Management Use Case CMS-7

Stakeholders involved:	Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A transaction containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	The access control is formulated based on the consent status contained in the latest transaction.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. Upon a data access request, the data management system initiates a request to the Consent Management System to check the consent status between the customer and the requesting party. 2. The Consent Management System retrieves the relevant transaction from the blockchain infrastructure in order to validate the status of the consent for the requesting party. 3. The Consent Management System formulates the approval or denial decision and informs the data management system.
Pass Metrics	1. The data access requests are correctly validated.
Fail Metrics	1. The data access requests are incorrectly validated.

4.1.2.8 Use Case CMS-8: Retrieve complete history of consents

The customer is able to be constantly informed of all the consents given to each specific recipient, as well as of the complete history of the consents. The transaction performed contains all the different versions of the consents of the customer besides the latest one. In this sense, the customer will be able to retrieve at any time his/her consent history per specific stakeholder, for all stakeholders or all the consents given by the customers for a stakeholder.

Table 16: Consent Management Use Case CMS-8

Stakeholders involved:	Customer, Internal Financial Institution
Pre-conditions:	At least a transaction containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	The customer is able to retrieve the complete history of consents by the Consent Management System.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. The customer initiates a request to the Consent Management System to retrieve the active consents per specific stakeholder along with their history or for a specific stakeholder. 2. The Consent Management System interacts with the blockchain infrastructure and retrieves the relevant transactions in order to compile a list of consents that customer has granted. 3. The customer is able to check the requested list of consents in a user-friendly way (e.g., through his/her mobile phone app or a web-based interface provided by the Internal Financial Institution).
Pass Metrics	1. The customer retrieves the request list of consent he/she has granted.
Fail Metrics	1. The customer is able to retrieve the request list of consent he/she has granted.

4.1.2.9 Use Case CMS-9: Expiration of the validity period

In the case of the “once off” consent, the validity period is set to a predefined time period. The moment that a transaction is created in the blockchain infrastructure for this specific type of consent, the Consent Management System creates and monitors the specific timer in order to perform the validation of consent time period. Once this timer is expired, the Consent Management System creates a new transaction for the specific consent with the status set to “expired”.

Table 17: Consent Management Use Case CMS-9

Stakeholders involved:	N/A
Pre-conditions:	A transaction with status active containing all the information of the consent form is available in the blockchain infrastructure whose timer has expired.
Post-conditions:	A new transaction containing the latest information of the consent form is available in the blockchain infrastructure with status set to “expired”.

Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	1. Upon the expiration of the validity timer, the Consent Management System retrieves the latest transaction for the specific consent by interacting with the blockchain infrastructure. 2. The Consent Management introduces the updates with a new transaction where the status is set to "expired".
Pass Metrics	1. A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Fail Metrics	1. There is no new transaction with updated consent information.

4.1.3 Sequence Diagrams

In the previous section, all the relevant use cases of the designed blockchain application were documented. The following sections present the sequence diagrams for each use case, depicting the interactions between the stakeholders and the components of the designed solution, as well as the interactions between the various components of the designed solution.

4.1.3.1 Register Customer in the Consent Management System

In Use Case CMS-1, the customer of the internal financial institution registers in the Consent Management System in order to create the associated profile that will receive consent requests by interacting with the web interface of the Consent Management System and providing the profile details. In the same manner, the representative of the external financial institution or the peer registers in the Consent Management System in order to create the associated profile that will initiate consent requests to the internal financial institution in order to get access to the internal financial institution's customer's data.

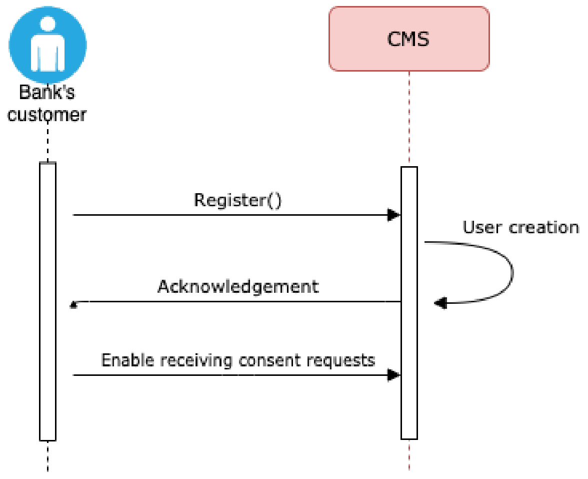

Figure 5: Use Case CMS-1 sequence diagram (customer)

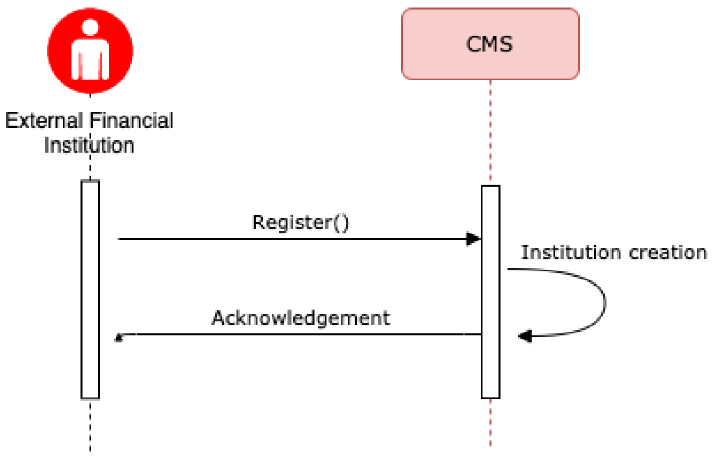

Figure 6: Use Case CMS-1 sequence diagram (external financial institution)

4.1.3.2 Customer receives a request to provide new consent for sharing his/her customer data

In Use Case CMS-2, the external financial institution or peer initiates a request to receive the customer's consent in order to access his/her data. Upon the approval of the internal financial institution's administrator, the Consent Management System interacts with blockchain components in order to check the existence of a consent in the ledger by querying and reading the query results upon decryption. In the case of absence of a valid and active consent, the Consent Management System is informed to transmit the new request to the customer. In case of existence of a valid and active consent, the Consent Management System is informed and approves the request to access the requested data.

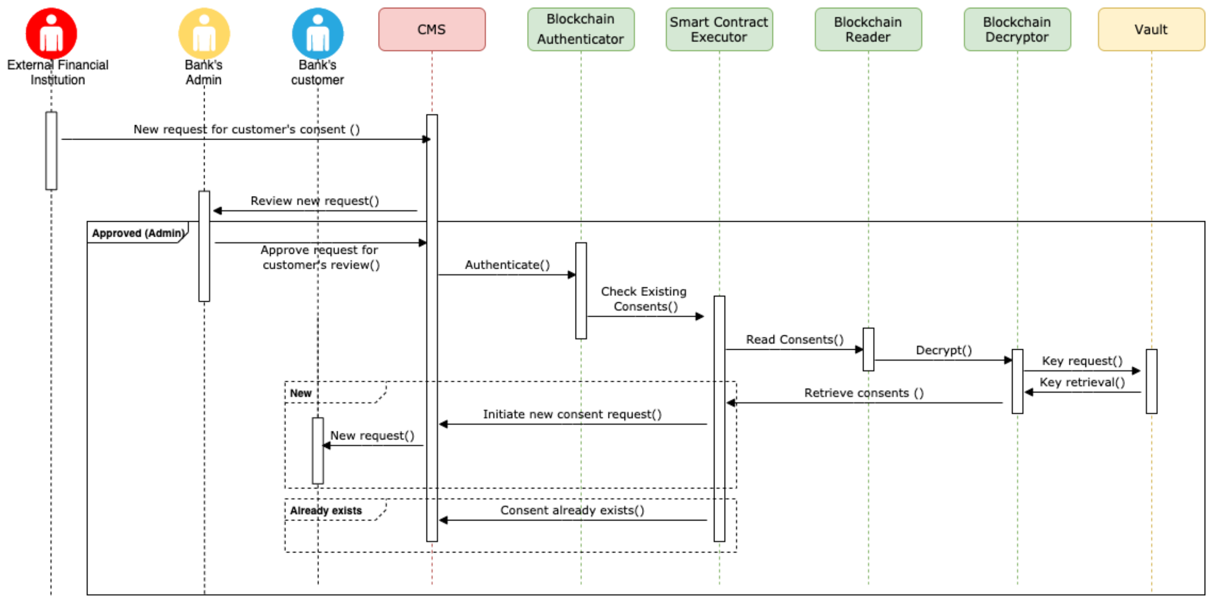

Figure 7: Use Case CMS-2 sequence diagram

4.1.3.3 Definition of the consent

In Use Case CMS-3, the customer receives and reviews the request and in the case of approval, defines the details and conditions of the consent in the Consent Management System and the candidate consent form is created. In the case of denial, the Consent Management System informs the external financial institution accordingly.

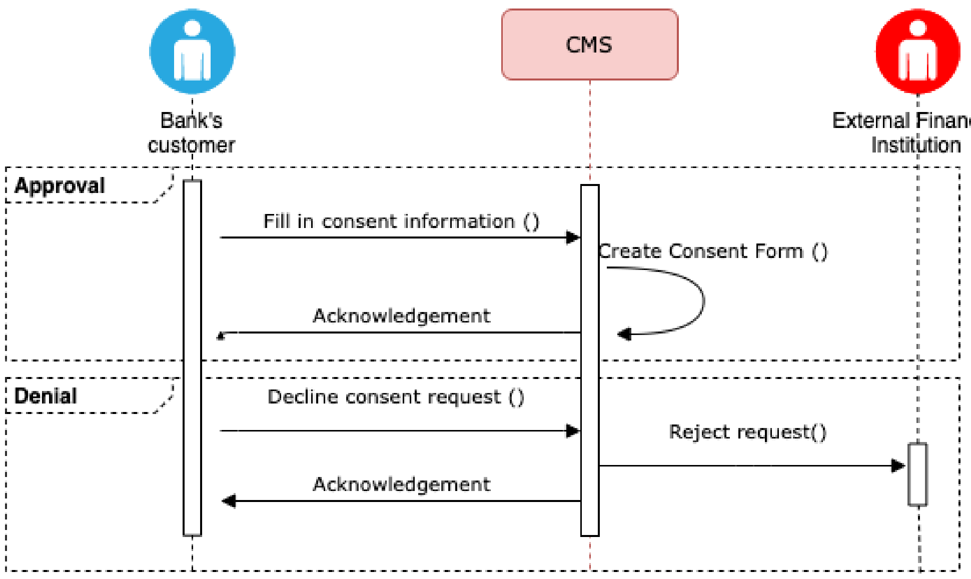

Figure 8: Use Case CMS-3 sequence diagram

4.1.3.4 Signing of the consent by the interested parties

In Use Case CMS-4, the Consent Management System provides the candidate consent form to both the customer and external financial institution and collects the digitally signed consent from both parties.

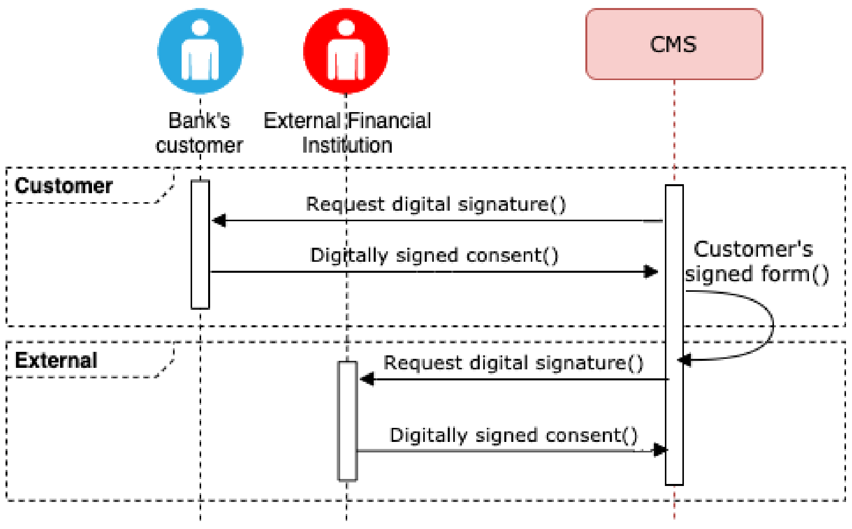
Bank's customer
External Financial Institution
CMS
Customer
Request digital signature()
Digitally signed consent()
Customer's signed form()
External
Request digital signature()
Digitally signed consent()

Figure 9: Use Case CMS-4 sequence diagram

4.1.3.5 Consent form is entered into blockchain

In Use Case CMS-5, the Consent Management System interacts with the blockchain components in order to write the new consent by executing the respective chaincode, encrypting the transaction and writing the new transaction in the ledger. In the case of “once off” consent, the Consent Management System initiates the internal timer. Finally, the Consent Management System stores the formulated consent form in the File Storage.

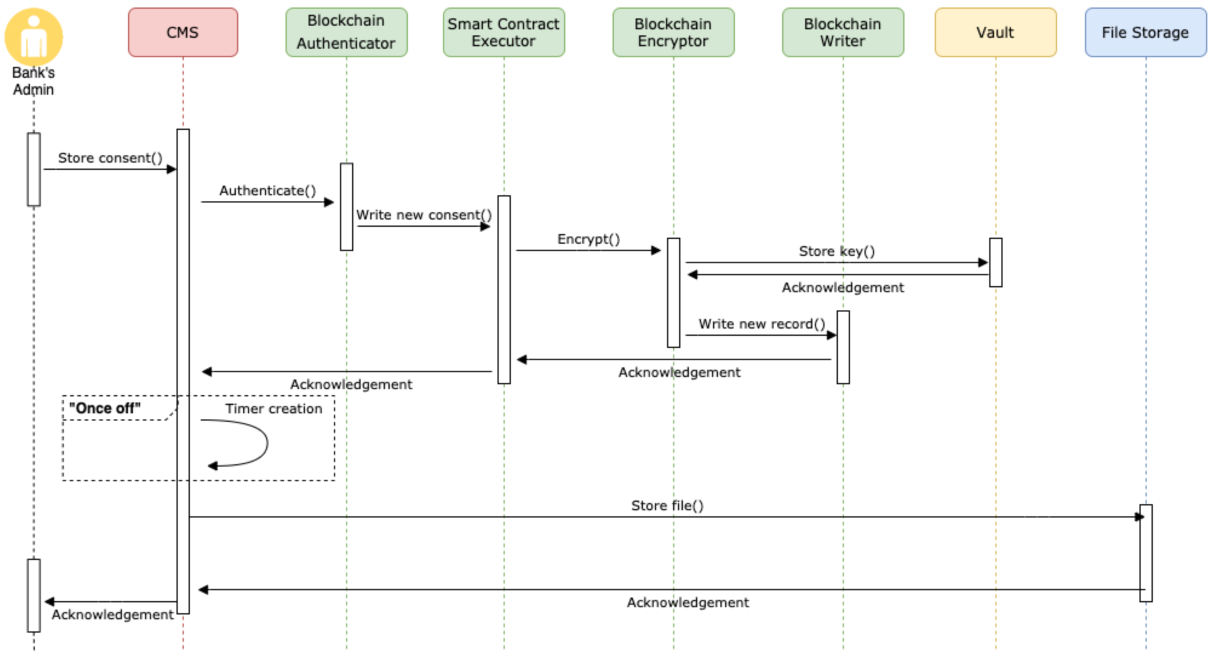

Figure 10: Use Case CMS-5 sequence diagram

4.1.3.6 Consent update or withdrawal

In Use Case CMS-6, the update or withdrawal at any time by the customer side of the consent is handled. In the case of an update, the Consent Management System orchestrates the execution of the sequence diagrams described for Use Case CMS-3 till Use Case CMS-5 in order to generate a new transaction based on the updated consent form that is digitally signed by both parties. In the case of withdrawal, the Consent Management System interacts with the blockchain components in order to retrieve the existing consent from the ledger and update the consent status through a new transaction in the ledger that depicts the new status. Both cases are presented in the sequence diagrams below.

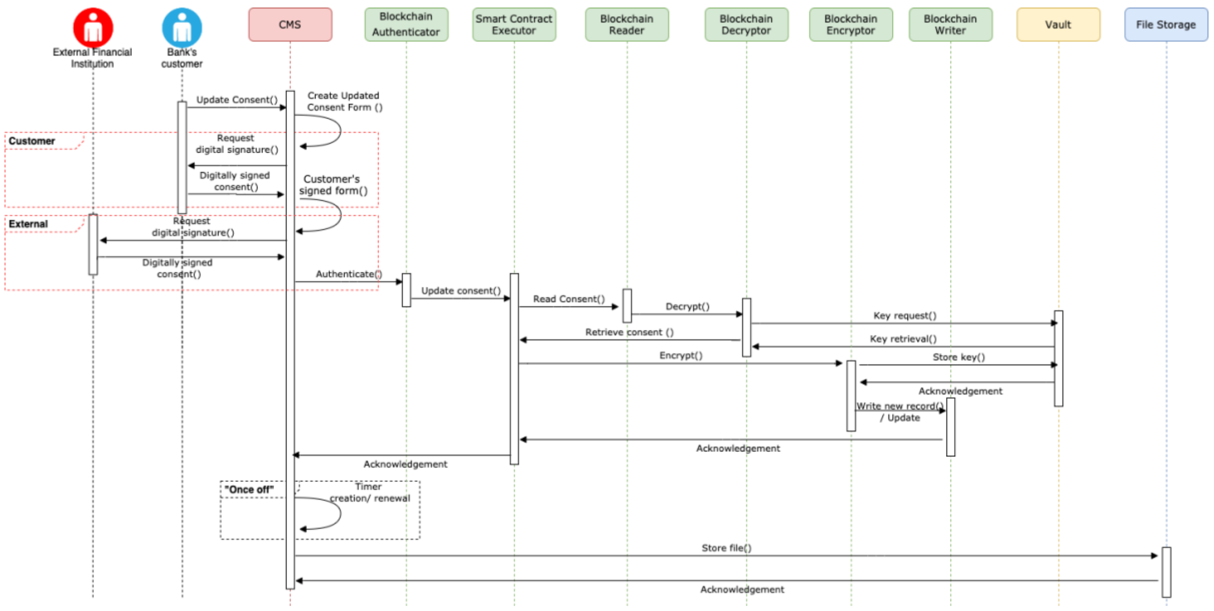

Figure 11: Use Case CMS-6 sequence diagram (update)

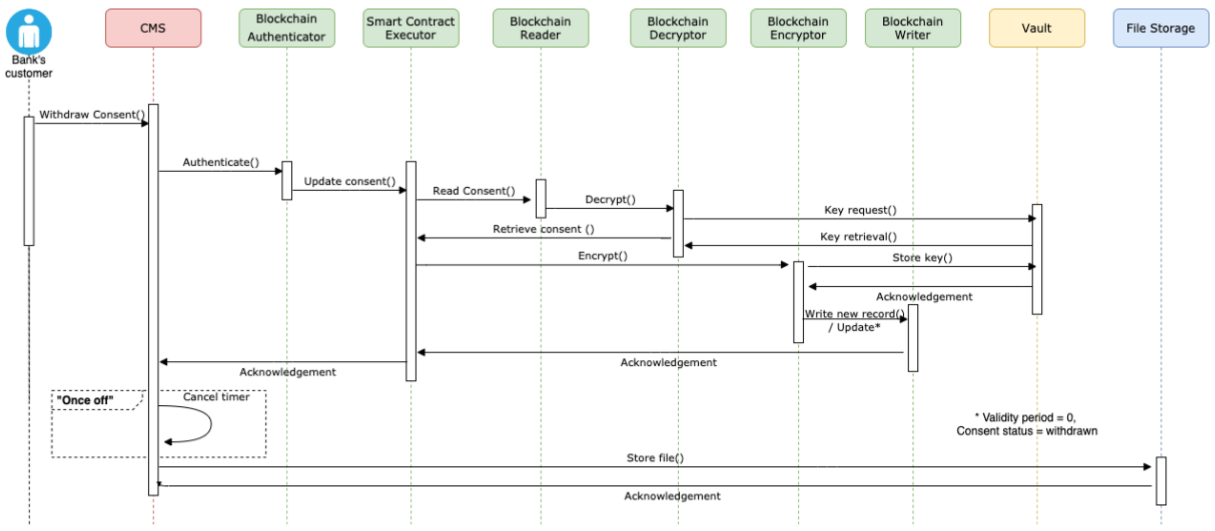

Figure 12: Use Case CMS-6 sequence diagram (withdrawal)

4.1.3.7 Access Control based on the consent forms

In Use Case CMS-7, the Consent Management System receives a new data access request and formulates an access control decision based on the existence of a valid and active consent. To achieve this, it interacts with the blockchain components in order to query the ledger for the existence of a consent between the involved parties for the specific data.

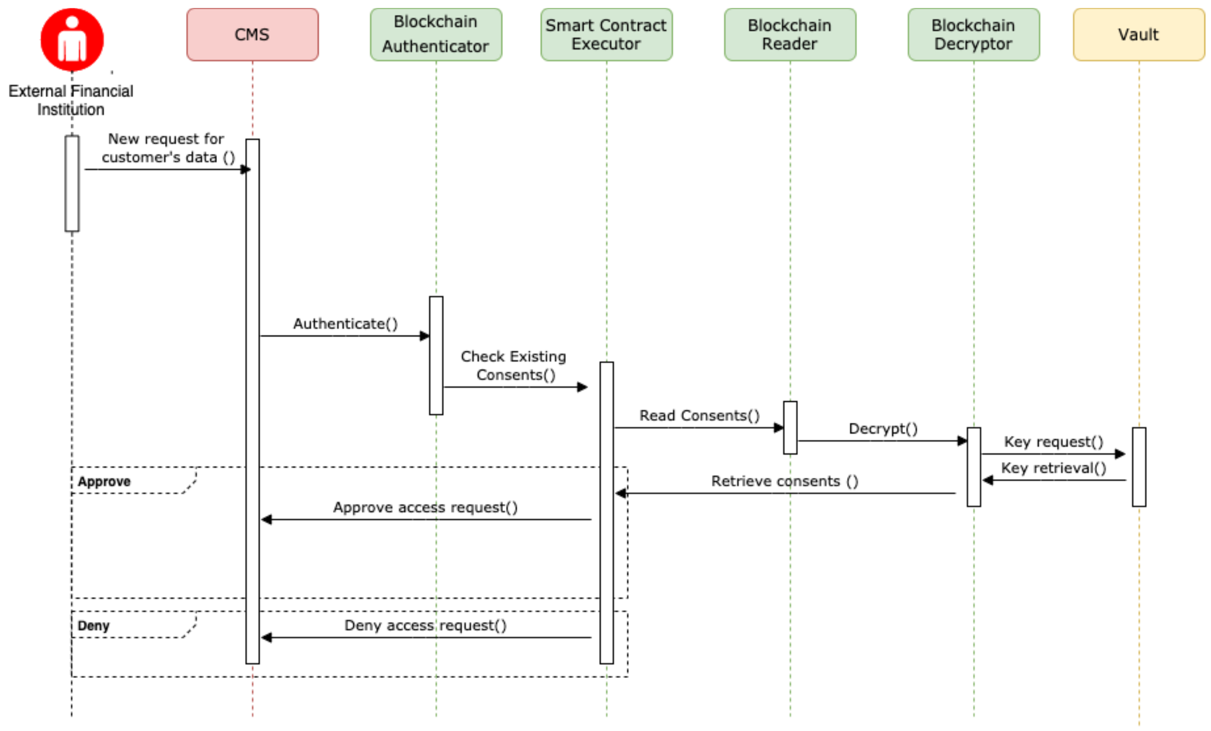

Figure 13: Use Case CMS-7 sequence diagram

4.1.3.8 Retrieve complete history of consents

In Use Case CMS-8, the Consent Management System receives a new request to retrieve all the consents that a customer has given to each specific recipient of his/her data along with their completed history. To achieve this, the Consent Management System interacts with the blockchain components in order to query the ledger and retrieve the required information.

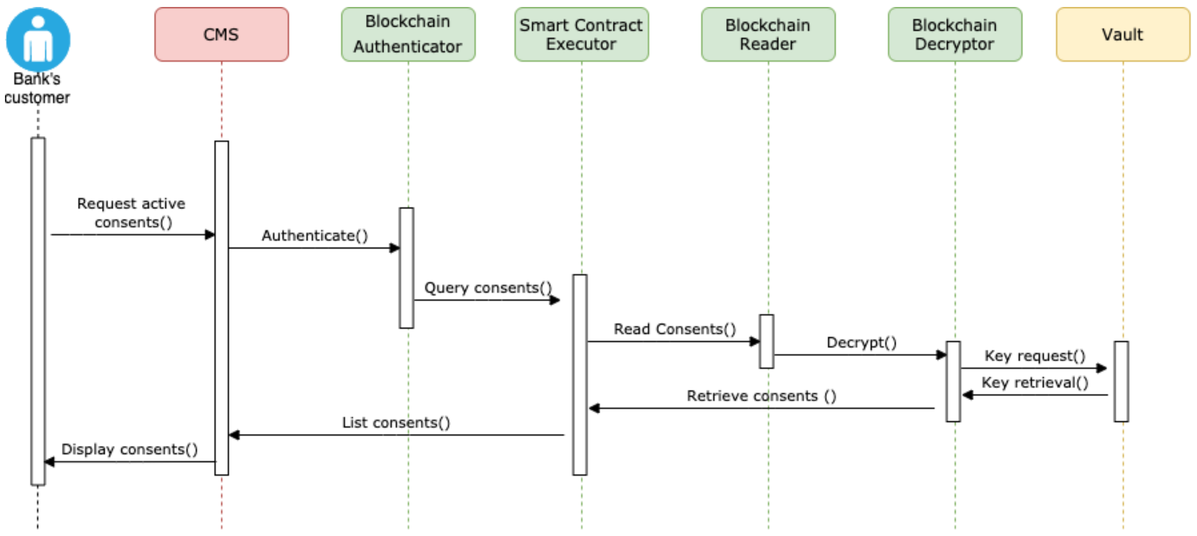

Figure 14: Use Case CMS-8 sequence diagram

4.1.3.9 Expiration of the validity period

In Use Case CMS-9, before the validity period of a “once off” consent expires, the Consent Management System informs the customer in order to initiate an update of the consent as described in Use Case CMS-6. In the case where the timer expires, the Consent Management System interacts with the blockchain components to retrieve the consent and update the consent status via a new transaction.

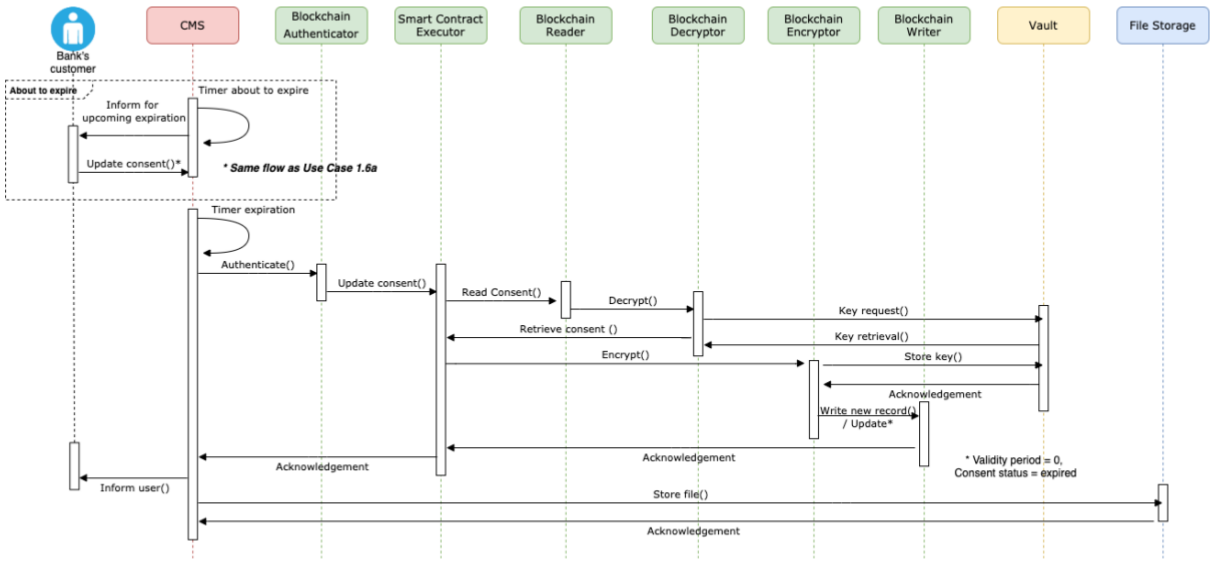

Figure 15: Use Case CMS-9 sequence diagram

4.1.4 Implementation of the Consent Management

In this section, the implementation details of the first version of the Consent Management are presented. The designed and implemented version constitutes the first prototype version which implements a subset of the use cases described in Section 4.1.2. As the project evolves, this prototype will be further enhanced in order to support the additional use cases as per the design specifications documented in the previous section. The scope of the current section is to present in a clear and coherent way the implementation details of the first delivered prototype version by performing a walkthrough of the implemented functions and the structure of the source code.

Based on the design specifications of the Sections 4.1.2 and 4.1.3, as well as the data schema defined in Section 4.1.1, the chaincode is structured at a component level as depicted in Figure 16.

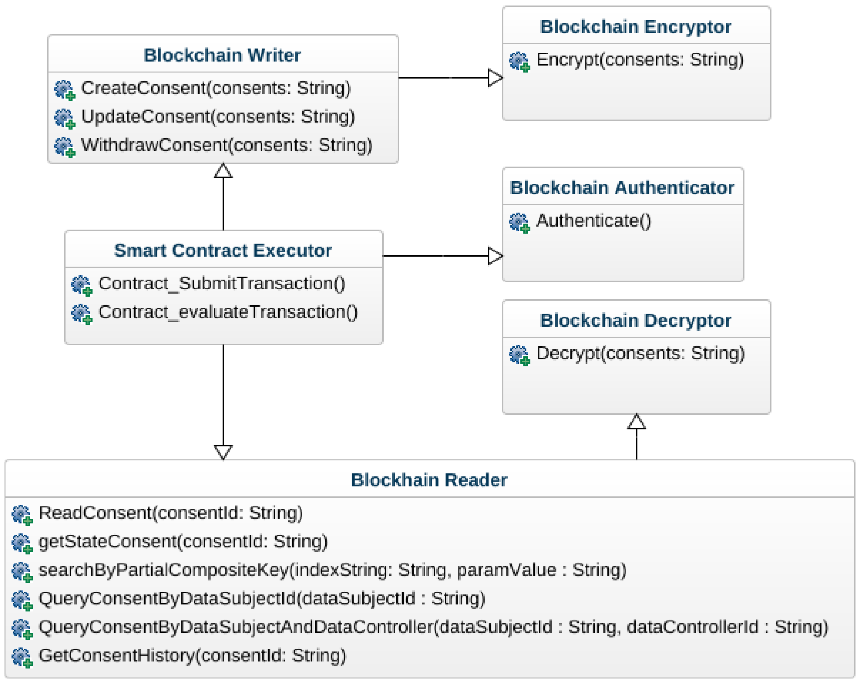
Blockchain Writer
CreateConsent(consents: String)
UpdateConsent(consents: String)
WithdrawConsent(consents: String)
Blockchain Encryptor
Encrypt(consents: String)
Blockchain Authenticator
Authenticate()
Smart Contract Executor
Contract_SubmitTransaction()
Contract_evaluateTransaction()
Blockchain Decryptor
Decrypt(consents: String)
Blockhain Reader
ReadConsent(consentId: String)
getStateConsent(consentId: String)
searchByPartialCompositeKey(indexString: String, paramValue : String)
QueryConsentByDataSubjectId(dataSubjectId : String)
QueryConsentByDataSubjectAndDataController(dataSubjectId : String, dataControllerId : String)
GetConsentHistory(consentId: String)

Figure 16: Consent Management chaincode structure

The *Blockchain Writer* component undertakes the responsibility to submit new transactions to the blockchain ledger. In the Consent Management case, this refers to transactions for a new consent, an updated consent or a withdrawn consent, and it is implemented with the following functions:

- `CreateConsent(consents string)`: The specific function is performing the necessary actions in order to create a new consent receipt into the ledger by updating the world state with a new record and appending new blocks at the end of the ledger. The function receives the data containing the consent information, constructs them into the appropriate format and interacts with the *Encrypt()* function to encrypt them. Once the data are encrypted they are inserted into the ledger.
- `UpdateConsent (consents string)`: The specific function undertakes the responsibility of updating an existing consent receipt into the ledger by updating the world state with a new record and appending new blocks at the end of the ledger. The function receives the updated consent information, constructs it into the appropriate format and encrypts the information via the *Encrypt()* function. Finally, the encrypted information is appended into the ledger.
- `WithdrawConsent (consents string)`: In the same manner as with the UpdateConsent function, the specific function withdraws an existing consent by updating the world state with a new record and appending new blocks at the end of the ledger. The new information is also encrypted via the *Encrypt()* function before it is appended into the ledger.

The *Blockchain Reader* component provides the necessary functions to read the ledger state, fetch a particular consent by id, query the ledger state given different parameters to fetch a set of consents matching those parameters, and get the consents history (history of all updates for particular consent data entity or data entities). The implemented functions are as follows:

- `ReadConsent (consentId string)`: This function is responsible for reading the consent record with the given consentId from the ledger, interacting with the *Decrypt()* function to decrypt the consent information, converting it to the appropriate format and returning it to the invoker.

- `getStateConsent (consentId string):` This is a helper function used in *ReadConsent(), searchByPartialCompositeKey(),* and the query functions to read a given consent record (identified by 'consentId' value) from the ledger.
- `searchByPartialCompositeKey (indexString string, paramValue string):` This is a helper function intended to search the ledger by a partial composite key of consent entities. When storing the consent record in the *CreateConsent()* function, two composite keys are created and stored in the ledger along with the consent data entity itself: a composite key comprised of combining the 'dataControllerId' with 'consentId', and a composite key comprised of the combination of 'dataSubjectId' and 'consentId'. This enables fast searches of all consent records serviced by particular dataControllerId, or belonging to particular dataSubjectId. The function receives the particular index it needs to search the ledger by (either the dataSubjectId index or datacontrollerId index), the partial value of the composite key to use in the search (the particular dataSubjectId or dataControllerId), and it returns a list of decrypted consent records matching this search.
- `QueryConsentByDataSubjectId (dataSubjectId string):` This function uses the *searchByPartialCompositeKey()* function to search for all consents belonging to the specified dataSubjectId. It returns an array of matching decrypted consent entities.
- `QueryConsentByDataSubjectAndDataController (dataSubjectId string, dataControllerId string):` This function uses the *searchByPartialCompositeKey()* function to search for all consents belonging to the specified dataSubjectId and serviced by the specified dataControllerId. It returns an array of matching decrypted consent entities.
- `GetConsentHistory (consentId string):` This function utilizes the Fabric's built-in functionality to fetch the transaction history for the particular consent entity given by consentId value. It returns an array of decrypted consent records, each representing an updated state of the particular consent entity, along with the txId (transactionId) which updated it and the timestamp from when it was updated.

The *Smart Contract Executor* component provides the necessary functions for the execution of the smart contract on the blockchain ledger. In the Consent Management case, this refers to all the operations related to the read and write operations which are performed by the respective functions of the Blockchain Writer and Blockchain Reader components. The implemented functions are as follows:

- `Contract_SubmitTransaction():` The specific function is performing the necessary actions for the execution of the smart contract by invoking the respective functions of the Blockchain Writer (*CreateConsent(), UpdateConsent(), WithdrawConsent()*) along with their required parameters based on the received request.
- `Contract_evaluateTransaction():` The specific function is performing the necessary actions for the execution of the smart contract by invoking the respective functions of the Blockchain Reader (*ReadConsent, GetConsentHistory, QueryConsentByDataSubjectId, QueryConsentByDataSubjectAndDataController,)* along with their required parameters based on the received request.

The *Blockchain Authenticator* component is the component responsible for the authentication operations which are performed in order to ensure the controlled access of the respective users to the blockchain channel utilised by the Consent Management case. For the specific component, the following function is implemented:

- `Authenticate():` The specific function performs the authentication of the user requesting access to the underlying blockchain channel. In the case where the appropriate keys and certificates are provided by the user, access to the specific channel is granted.

The *Blockchain Encryptor* component is the component that undertakes the responsibility for the encryption operations which are performed on the consent data prior to being inserted as new transactions to the blockchain ledger. For the specific component, the following function is implemented:

- `Encrypt (consents string)`: The specific function is performing the necessary actions in order to encrypt the consent data utilising AES 256 encryption. The function receives the data containing the consent information, encrypts them with a randomly generated encryption key which is later stored in a Vault.

The *Blockchain Decryptor* component is the component responsible for the decryption of the consent data which were encrypted by the *Blockchain Encryptor.* Hence, the specific component decrypts the encrypted data when they are read from the blockchain ledger. For the specific component, the following function is implemented:

- `Decrypt (consents string)`: The specific function is performing the necessary actions in order to decrypt the consent data when they are fetched from the ledger. The function receives the encrypted data containing the consent information and decrypts them with the appropriate encryption key as retrieved from the Vault.

As explained also in the architecture of the solution, the Consent Management System is the mediator between the involved parties, namely the external financial institution or peer, the internal financial institution and the customer of the internal institution. It interacts with the underlying blockchain infrastructure where the chaincode resides and provides all the required functionalities to the involved stakeholder in order to formulate the consent. To this end, the Consent Management System is composed of a set of services, as depicted in Figure 17, which are interacting via well-defined APIs in order to provide the required functionalities.

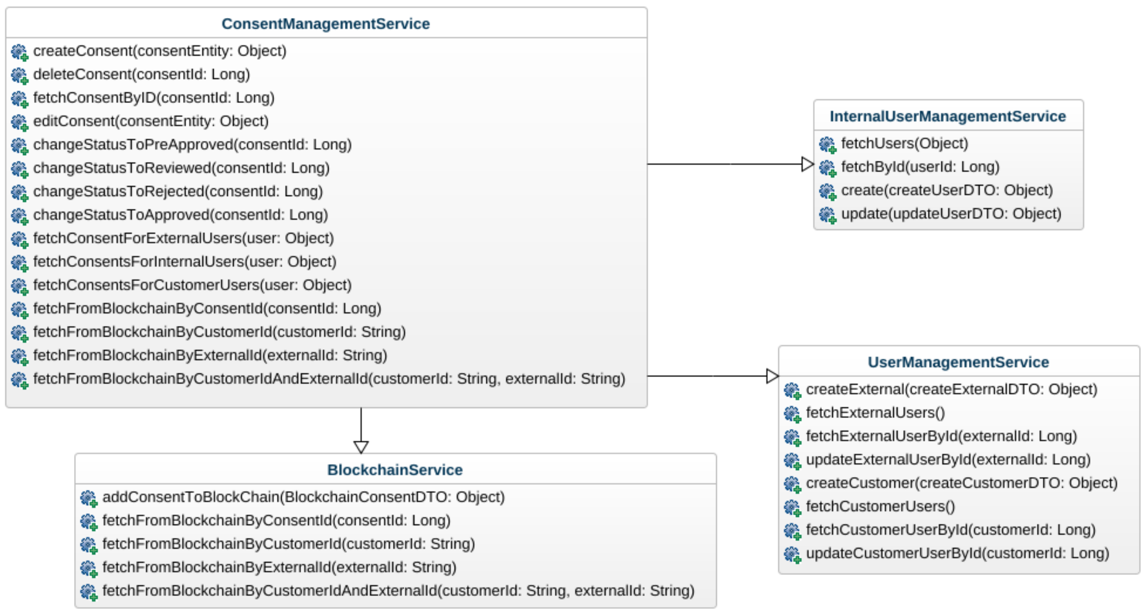
ConsentManagementService
createConsent(consentEntity: Object)
deleteConsent(consentId: Long)
fetchConsentByID(consentId: Long)
editConsent(consentEntity: Object)
changeStatusToPreApproved(consentId: Long)
changeStatusToReviewed(consentId: Long)
changeStatusToRejected(consentId: Long)
changeStatusToApproved(consentId: Long)
fetchConsentForExternalUsers(user: Object)
fetchConsentsForInternalUsers(user: Object)
fetchConsentsForCustomerUsers(user: Object)
fetchFromBlockchainByConsentId(consentId: Long)
fetchFromBlockchainByCustomerId(customerId: String)
fetchFromBlockchainByExternalId(externalId: String)
fetchFromBlockchainByCustomerIdAndExternalId(customerId: String, externalId: String)
InternalUserManagementService
fetchUsers(Object)
fetchById(userId: Long)
create(createUserDTO: Object)
update(updateUserDTO: Object)
UserManagementService
createExternal(createExternalDTO: Object)
fetchExternalUsers()
fetchExternalUserById(externalId: Long)
updateExternalUserById(externalId: Long)
createCustomer(createCustomerDTO: Object)
fetchCustomerUsers()
fetchCustomerUserById(customerId: Long)
updateCustomerUserById(customerId: Long)
BlockchainService
addConsentToBlockChain(BlockchainConsentDTO: Object)
fetchFromBlockchainByConsentId(consentId: Long)
fetchFromBlockchainByCustomerId(customerId: String)
fetchFromBlockchainByExternalId(externalId: String)
fetchFromBlockchainByCustomerIdAndExternalId(customerId: String, externalId: String)

Figure 17: Consent Management System source code structure

The ConsentManagementService is a core service of the Consent Management System and is responsible for the main operations performed during the consent formulation. In detail, the service undertakes the creation, update or editing, as well as the deletion of a consent request, enabling the formulation of the consent through the various steps described in the use cases reported in the

Section 4.1.2. The specific service is responsible for the formulation of the consent prior to being inserted into the blockchain ledger. Finally, it interacts with the BlockchainService in order to perform all the required operations on the underlying blockchain infrastructure. The specific service exposes the *ConsentManagementRestController* through which the various interactions with the rest of the services are realised. The implemented functions are as follows:

- `createConsent(consentEntity: Object)`: The specific function creates the new consent request based on the input received and stores it in the Consent Management System's database.
- `deleteConsent(consentId: Long)`: The specific function deletes an existing consent request from the Consent Management System's database.
- `fetchConsentByID(consentId: Long)`: The specific function fetches an existing consent request from the Consent Management System's database based on the provided ConsentId.
- `editConsent(consentEntity: Object)`: The specific function modifies an existing consent request in the Consent Management System's database based on the consent request data provided.
- `changeStatusToPreApproved(consentId: Long)`: The specific function updates the status of an existing consent request to PreApprove state. This operation is performed when the administrator of the internal financial institution pre-approves the newly received request from an external financial institution and as a consequence the request reaches the customer for review and approval.
- `changeStatusToReviewed(consentId: Long)`: The specific function updates the status of an existing consent request to Reviewed state. This operation is performed when the customer reviews the received consent requests, modifies it or approves it.
- `changeStatusToRejected(consentId: Long)`: The specific function updates the status of an existing consent request to Rejected state. This operation is performed when the customer reviews the received consent requests and rejects it.
- `changeStatusToApproved(consentId: Long)`: The specific function updates the status of an existing consent request to Approved state. This operation is performed when the external financial institution receives the reviewed and approved by the customer consent requests and approves it also.
- `fetchConsentForExternalUsers(user: Object)`: The specific function fetches all the received consent requests from a specific external financial institution.
- `fetchConsentsForInternalUsers(user: Object)`: The specific function fetches all the consent requests that an internal financial institution has received from any external financial institution.
- `fetchConsentsForCustomerUsers(user: Object)`: The specific function fetches all the received consent requests that a specific customer of the internal financial institution has received.
- `fetchFromBlockchainByConsentId(consentId: String)`: The specific function invokes the *fetchFromBlockchainByConsentId* function of the BlockchainService in order to retrieve a formulated consent from the blockchain ledger.
- `fetchFromBlockchainByCustomerId(customerId: String)`: The specific function invokes the *fetchFromBlockchainByCustomerId* function of the BlockchainService in order to retrieve the list of formulated consents for a specific customer from the blockchain ledger.
- `fetchFromBlockchainByExternalId(customerId: String)`: The specific function invokes the *fetchFromBlockchainByExternalId* function of the BlockchainService in order to retrieve the list of formulated consents from an external financial institution from the blockchain ledger.

- `fetchFromBlockchainByCustomerIdAndExternalId (customerId: String, externalId: String)`: The specific function invokes the *fetchFromBlockchainByCustomerIdAndExternalId* function of the BlockchainService in order to retrieve a specific consent between a customer and an external financial institution from the blockchain ledger.

The BlockchainService is another core service of the Consent Management System that is responsible for the insertion, modification and retrieval of the consents to or from the blockchain ledger. While the actual consent is formulated by the ConsentManagementService starting from the initial consent request, once the consent has been formulated the BlockchainService is invoked to insert the formulated consent in the blockchain ledger. On the other hand, when the consents are displayed to the involved stakeholders, the BlockchainService is interacting with underlying blockchain infrastructure in order to retrieve the required information. Finally, any updates performed on the consent information are inserted into the blockchain ledger. The implemented functions are as follows:

- `addConsentToBlockChain(BlockchainConsentDTO: Object)`: The specific function receives the consent information as formulated by the ConsentManagementService and interacts with the blockchain infrastructure in order to insert it into the blockchain ledger.
- `fetchFromBlockchainByConsentId(consentId: String)`: The specific function retrieves a specific consent from the blockchain ledger based on the provided consentId.
- `fetchFromBlockchainByCustomerId(customerId: String)`: The specific function retrieves the list of available consents of a specific customer from the blockchain ledger.
- `fetchFromBlockchainByExternalId(customerId: String)`: The specific function retrieves the list of available consents obtained by a single external financial institution from the blockchain ledger.
- `fetchFromBlockchainByCustomerIdAndExternalId(customerId: String, externalId: String)`: The specific functions retrieves the consent between the specific customer and the specific external financial institution.

The InternalUserManagementService performs the user management operations for the internal financial institution. In particular, the specific service creates the users of the internal financial institution. The implemented functions are as follows:

- `fetchUsers(user: Object)`: The specific function retrieves the detailed list of users of the internal financial institution.
- `fetchById(userId: Long)`: The specific function retrieves the details of a specific user of the internal financial institution.
- `create(createUserDTO: Object)`: The specific function creates a new user of the internal financial institution.
- `update(updateUserDTO: Object)`: The specific function updates an existing user of the internal financial institution.

The UserManagementService is responsible for the user management operations with regards to the customers of the internal financial institution and the users of the external financial institution. In this sense, it provides all the functionalities to create a customer or an external financial institution, as well as update their profile information at any time. The implemented functions are as follows:

- `createExternal(createExternalDTO: Object)`: The specific function creates a new external financial institution.
- `fetchExternalUsers()`: The specific function retrieves the detailed list of the external financial institutions.
- `fetchExternalUserById(externalId: Long)`: The specific function retrieves the details of a specific external financial institution.

- `updateExternalUserById(externalId: Long)`: The specific function updates an existing external financial institution.
- `createCustomer(createCustomerDTO: Object)`: The specific function creates a new customer of the internal financial institution.
- `fetchCustomerUsers()`: The specific function retrieves the detailed list of the customers of the internal financial institution.
- `fetchCustomerUserById(customerId: Long)`: The specific function retrieves the details of a specific customer of the internal financial institution.
- `updateCustomerUserById(customerId: Long)`: The specific function updates an existing customer of the internal financial institution.

Video Link of Solution in INFINITECH Marketplace:
https://marketplace.infinitech-h2020.eu/infinitech/blockchain-enabled-consent-management

4.2 Know Your Customer / Know Your Business

Updates from D4.7:
The updates on this iteration involve the Implementation Section (4.2.4) that includes analysis on the current framework and illustrates the realization of the conceptualization of the use case as defined in Sections 4.2.1, 4.2.2 and 4.2.3. In addition to this, the paragraph in Section 4.2.1 with the indicative functions, as documented in the previous iteration, has been removed, as it is now covered by the newly introduced Section 4.2.4.

4.2.1 Description of the solution

The Know Your Customer (KYC) / Know Your Business (KYB) policies in state-of-the-art financial relations expect that customer parties, either individuals or entire corporations, endeavour verification of their identity. Thus, each financial organization is able to estimate the risks involved with sustaining a new business-customer partnership. In this context, together with the wider Finance field of Anti-Money Laundering (AML) procedures, every financial institution establishes KYC and KYB operations at the time they register a new customer [12].

The KYC/KYB blockchain application resolves the implementation of such industry mechanisms by utilizing blockchain technology as a basic background infrastructure. Security, immutability and controlled transparency are employed inside large enterprise blockchain networks, while they offer efficiency of tasks and effectiveness of transactions within the corporation and its members. Ultimately, blockchain technology simplifies the emerging use cases and directly addresses any possible issues that are created.

Particularly, a KYC/KYB mechanism ensures that the identification and verification of a customer occurs against national and international regulations and laws set by governments, commissions, central banks and financial associations. As both the customer profile information and the relevant laws and rules are subject to changes over time, their update and maintenance become complicated. Moreover, their centralized systems are exposed to data protection and cyber-security risks, which become less expensive to launch while they are led by more sophisticated adversaries, year after year [13].

Blockchain technology and particularly permissioned blockchain networks are capable of providing security to the KYC and KYB processes through decentralization. The concept of decentralization mainly exploits the idea that the information is replicated across all network nodes, and thus sabotaging one or more nodes cannot harm the information integrity and a single point of failure is avoided. In particular, the permissioned blockchain technology promises to keep that sensitive information inside a private network where only privileged parties can access it with an insider invitation. Thus, the customer information is kept safe on a private ledger that offers transparency to a privileged group of legal network participants. Both the customer and the organization are able to perform create, read, write, delete (CRUD) operations on the data under pre-defined access control policies. The various features of permissioned blockchains enable different policy applications that are able to, for instance, separate legal parties into a higher privacy network running inside the initial private one. Improved privacy control and data immutability rule inside the aforementioned technological scenario while they ensure legitimate customer data protection and management together with proper administration of this data by financial enterprises [14].

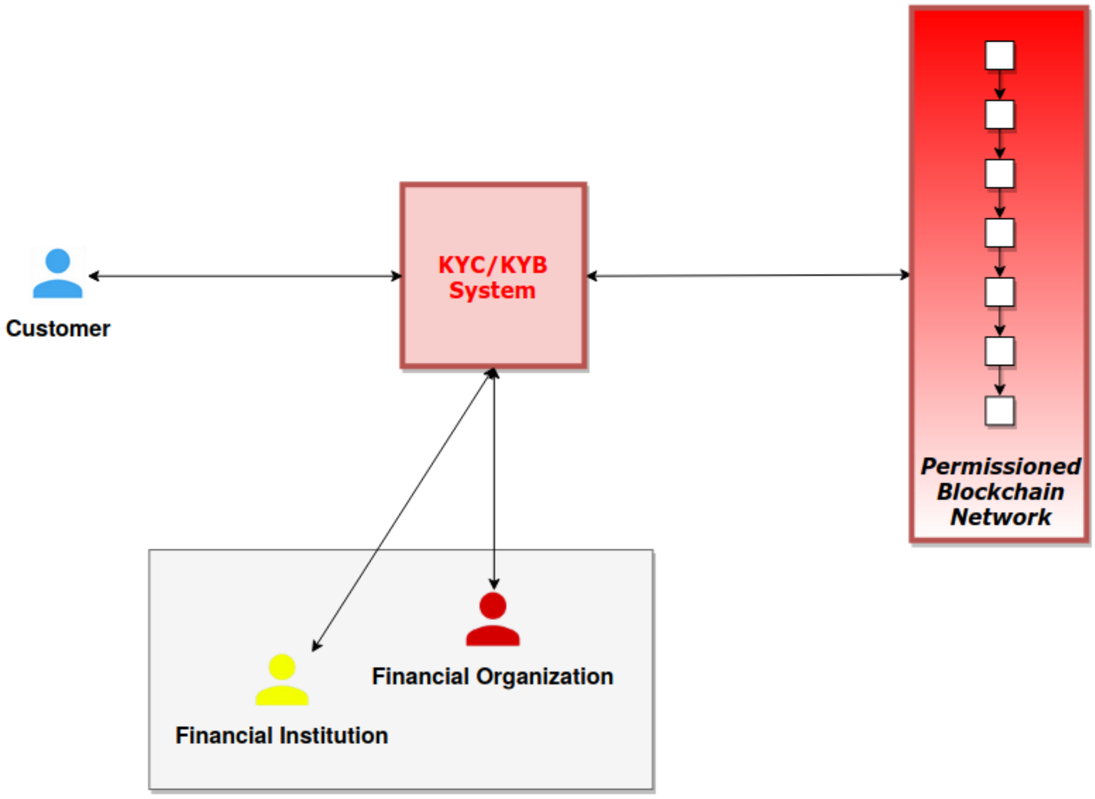

Figure 18: High-level architecture of the KYC/KYB solution

Figure 18 depicts a high-level architecture of the proposed solution. In general, the customer participant (light blue actor), being either an individual or an entire organization, needs to acquire financial services that are offered by the financial institution (yellow actor). For the completion of this transaction, the financial institution requests that the customer identity information is documented in KYC/KYB data after it is legally verified. In that case, the customer participant uploads their KYC/KYB documentation to the KYC/KYB System that interacts with the permissioned blockchain network and stores this information on-chain. When another financial organization (red actor) requires to start business relations with the same customer (light blue actor), the latter's KYC/KYB information is easily and time-efficiently obtained through the access rights provided by the financial institution (yellow actor) which is already having business partnerships with this customer. It is important to clarify that the customer using this kind of technology has declared their consent of sharing their information among privileged participants of the network, while their information privacy is guaranteed. In this system, data security and efficient data management and maintenance rule, for the united underlying blockchain infrastructure with common data structures offers stability, security, sustainability and time-efficiency.

The design specifications of any blockchain application is tightly related to the definition of the business objects for which the relevant functions of the chaincode will be implemented towards the maintenance and update of their current and historical state. In this sense, to facilitate the implementation of the KYC/KYB application, the initial data schema has been defined. The initial data schema is depicted in Figure 19, while the details of all the entities are documented in Table 18 and Table 19.

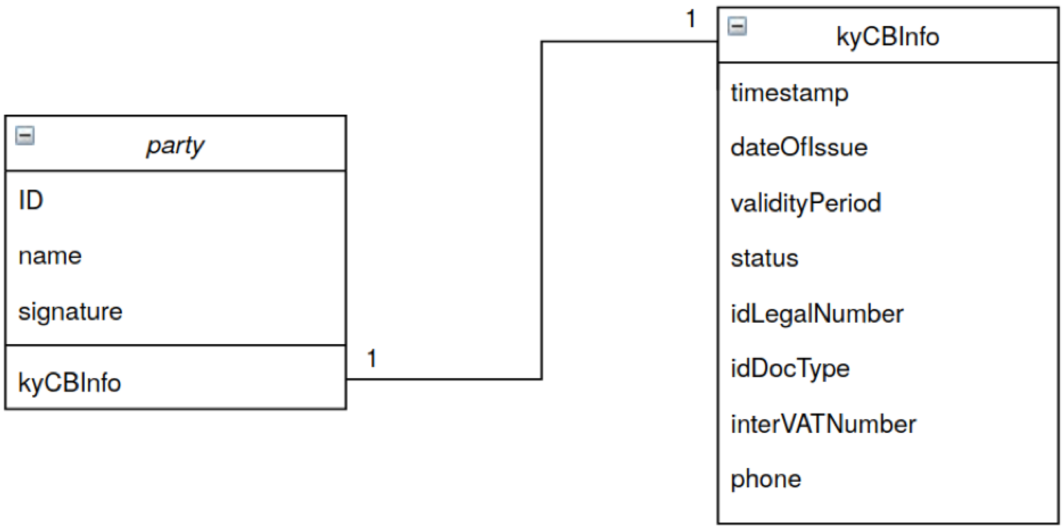

Figure 19: KYC/KYB Data Schema diagram

Table 18: KYC/KYB Data Schema details (1)

party		
Name	**Type**	**Short description**
ID	String	REQUIRED: The identification number used to uniquely identify network participants
name	String	REQUIRED: The name of the participant
signature	String	REQUIRED: The digital signature of the participant
kyCBInfo	String	REQUIRED: The KYC/KYB pointer to the participant's KYC/KYB record

Table 19: KYC/KYB Data Schema details (2)

kyCBInfo		
Name	**Type**	**Short description**
kyCBInfo	Integer	REQUIRED: The KYC/KYB documentation pointer number
timestamp	String	REQUIRED: The date the KYC/KYB documentation is modified
dateOfIssue	String	REQUIRED: The date the KYC/KYB documentation is first accepted
validityPeriod	String	REQUIRED: The period until which the KYC/KYB information is valid
status	Boolean	REQUIRED: The validity of the KYC/KYB documents (either valid or invalid)
idLegalNumber	String	REQUIRED: The alphanumeric number of the legal identity of the participant

idDocType	String	REQUIRED: The document type with which the party is legally approved
interVATNumber	String	REQUIRED: The international VAT identification number of the party in alphanumeric number form
phone	Integer	REQUIRED: The communication phone to contact a participant representative

4.2.2 Use cases

The following sections provide the detailed documentation of all the use cases encapsulated in this blockchain application describing in detail all information of each use case.

4.2.2.1 Use Case KYC/KYB-1: Assertion of customer identity documents

In this use case, the customer asserts their identity documents to the financial organization through the KYC/KYB process. In particular, a customer who represents either an individual or an entire corporation, acknowledges the KYC or KYB information in order to initiate business relations with the financial institutions and organizations of the network. In this context, each enterprise participant of the permissioned blockchain network ensures the legitimacy of their customer and, thus, new business relationships are established.

Table 20: KYC/KYB Use Case KYC/KYB-1

Stakeholders involved:	Customer
Pre-conditions:	1. A private blockchain network is set up.
Post-conditions:	1. The Customer's KYC/KYB information is stored on-chain.
Data Attributes	1. The Customer's KYC/KYB documents
Normal Flow	1. The Customer uploads their KYC/KYB documentation on the blockchain ledger. 2. The documentation information is successfully stored on-chain.
Pass Metrics	1. The documentation information is successfully stored on-chain.
Fail Metrics	2. There is no Customer to upload their KYC/KYB documents.

4.2.2.2 Use Case KYC/KYB-2: Read access to customer identification information

In this use case, as a legal network participant, a financial organization has *read access* to any customer's KYC/KYB documents information. In particular, inside the permissioned blockchain network, each party is able to read the corresponding ledgers. In this context, each financial organization has *read access* to the data that is stored on-chain. By a simple *read access* request, the financial organization can fetch the information of a customer they are interested in. Additionally, upon using the system, each customer approves that their data may be accessed by the financial organization they will initiate business relations with.

Table 21: KYC/KYB Use Case KYC/KYB-2

Stakeholders involved:	Financial organization
Pre-conditions:	1. The financial organization is a legal participant of the network.
Post-conditions:	1. The financial organization has *read access* control over the requested KYC/KYB information.
Data Attributes	1. The financial organization's network attributes 2. The requested KYC/KYB documents
Normal Flow	1. The financial organization requests for *read access* of a specific customer's KYC/KYB documentation. 2. The requested documentation information access is successfully granted.
Pass Metrics	1. The requested documentation information is successfully granted.
Fail Metrics	1. The financial organization is not eligible to access the requested documentation information.

4.2.2.3 Use Case KYC/KYB-3: Sharing of customer KYC/KYB documents

In this use case, the financial organization A shares the KYC/KYB document information of customer B with the financial organization C through the secure and private blockchain network. In particular, each of the financial organizations participating in the blockchain network is eligible for accessing the data stored on-chain. However, depending on the different access control rights granted by the time of joining the network, there exists the case where different organizations are qualified to access different data. In this context, together with the initial customer consent, a financial organization may grant access to another organization or institution for a specific customer KYC/KYB documentation.

Table 22: KYC/KYB Use Case KYC/KYB-3

Stakeholders involved:	Financial organization
Pre-conditions:	1. The financial organizations A and C are legal participants of the network. 2. The requested customer B has submitted their KYC/KYB document information.
Post-conditions:	1. The financial organization C has *read access* control over the requested KYC/KYB information.
Data Attributes	1. The financial organizations' A and C network attributes 2. The requested KYC/KYB documents
Normal Flow	1. The financial organization A requests for sharing of a specific customer's KYC/KYB documentation with a financial organization C. 3. The requested documentation information access is successfully granted.

Pass Metrics	1. The requested documentation information is successfully granted.
Fail Metrics	1. The financial organization A is not eligible to share the requested documentation information.

4.2.3 Sequence Diagrams

In the previous section, all the relevant use cases of the designed blockchain application were documented. The following sections present the sequence diagrams for each use case, depicting the interactions between the stakeholders and the components of the designed solution, as well as the interactions between the various components of the designed solution.

4.2.3.1 Assertion of customer identity documents

In Use Case KYC/KYB-1, a customer provides their documentation information in order to be eligible for business partnerships with the participating financial organizations. In this use case, a customer actor, which is either an individual or an entire enterprise, acknowledges the requested KYC or KYB documents in the KYC/KYB System by uploading them. This action is translated into an internal request that propagates the information inside the blockchain network, though the different blockchain components. Finally, the data is safely stored on-chain, i.e. on the private ledger, and further actions and use cases can emerge afterwards (see Use Case KYC/KYB-2 and Use Case KYC/KYB-3).

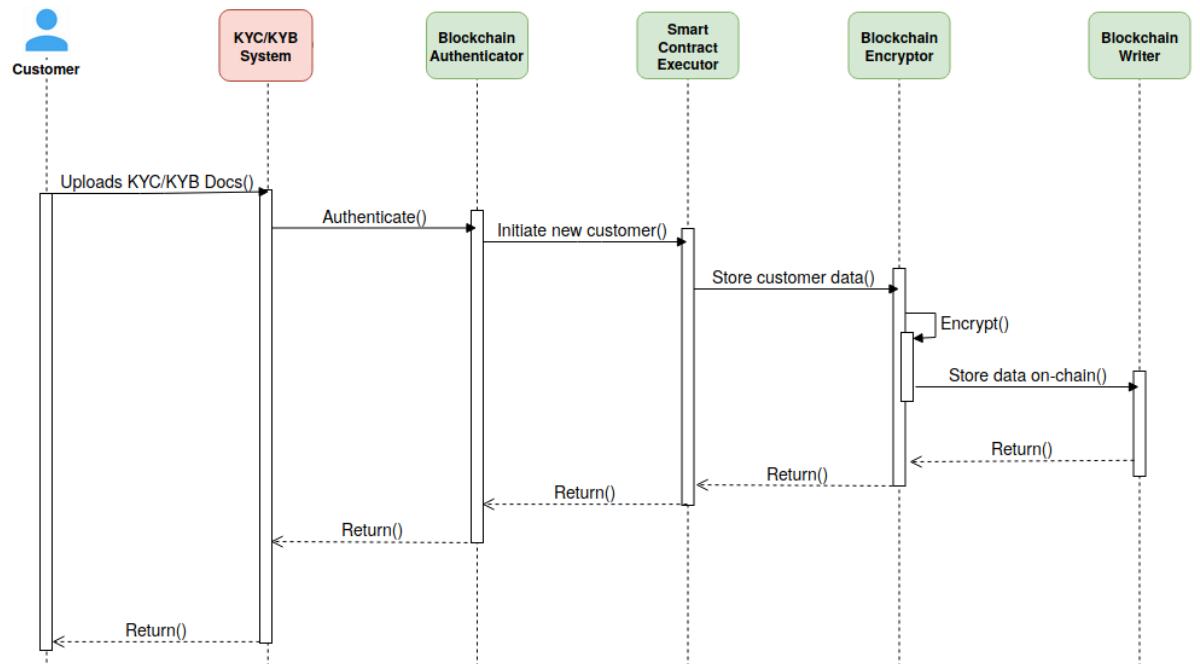

Figure 20: Use Case KYC/KYB-1 sequence diagram

4.2.3.2 *Read access* to customer identification information

In Use Case KYC/KYB-2, the blockchain network parties are constituted of financial organizations and are able to inspect the KYC/KYB documentation information of customers. Through the private and secure network, the different organizations obtain access to a customer's KYC/KYB submitted data in order to estimate whether to establish business relationships with them or not. The customer data submission is already accompanied by the customer's consent of using the KYC/KYB information by the legal parties of the network, i.e. the financial organizations. As in Figure 21, through the sequence of the blockchain components the *read access* is propagated to the financial organization that requested it.

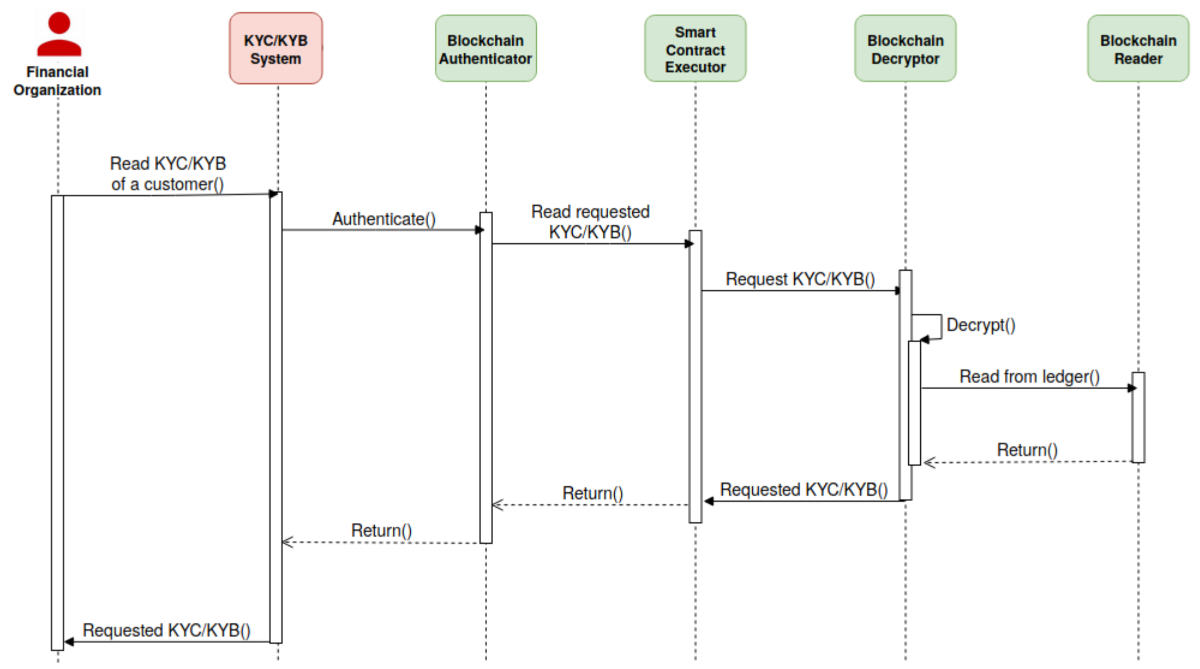

Figure 21: Use Case KYC/KYB-2 sequence diagram

4.2.3.3 Sharing of customer KYC/KYB documents

In Use Case KYC/KYB-3, a sharing of customer (B) information among participant organizations (A and C) takes place. Organization C obtains customer's B information through the cooperation of organization A. Since the customer's data already exists inside the secure and private blockchain network, organization C can request it indirectly. Organization A responds to the request by granting *read access* of customer B information to organization C. In such an efficient system, the customer and the financial organizations benefit from this kind of sharing, since the customer avoids re-entering their KYC/KYB data to a different system of a different organization, while the financial organizations speedily obtain customer information and make decisions upon the establishment of new business relations.

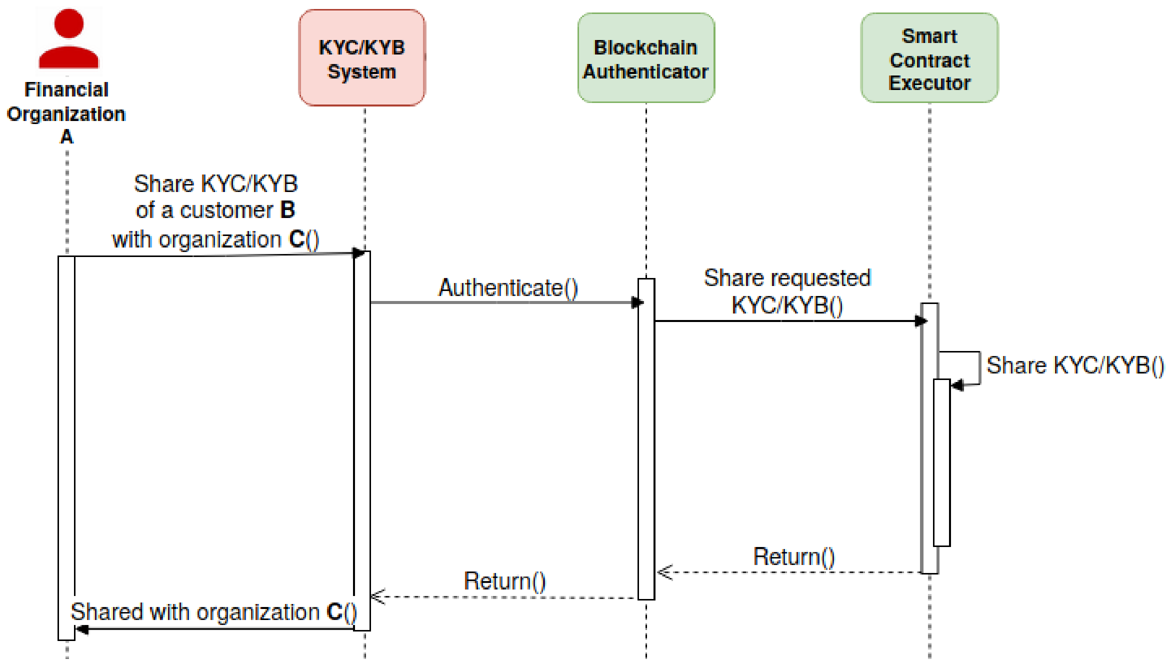

Figure 22: Use Case. KYC/KYB-3 sequence diagram

4.2.4 Implementation of the Know Your Customer / Know Your Business

This section explains the details of the implementation of the Know Your Customer / Know Your Business Use Case Demonstrator. The designed and developed solution has several important parts that are underlined below in order to present in a more clear and coherent way the built framework and the technological manifestation of the initial conceptualized architecture.

Based on the design specifications of the Sections 4.2.2 and 4.2.3, as well as on the data schema defined in Section 4.2.1, the chaincode is structured at a component level as depicted in Figure 23.

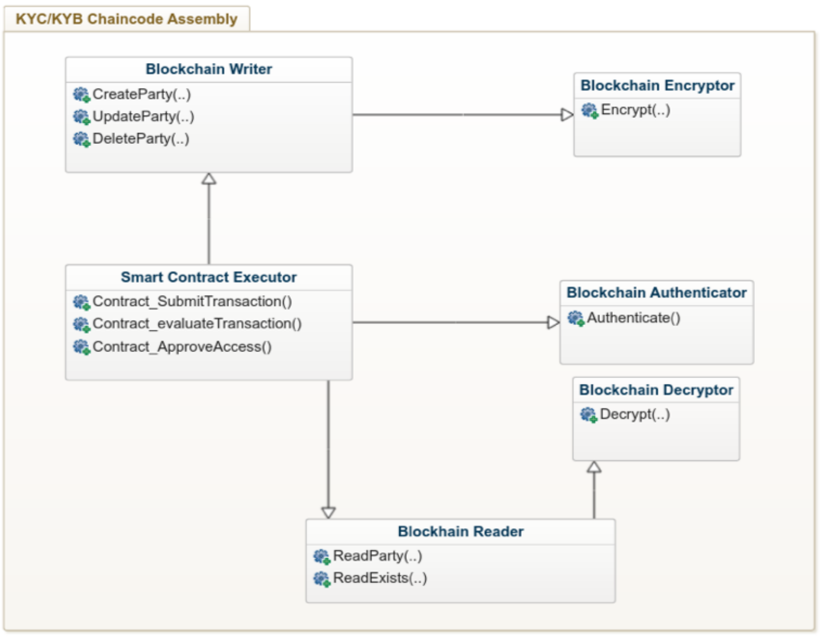

Figure 23: KYC/KYB Chaincode structure.

The *Blockchain Writer* component undertakes the responsibility to submit new transactions to the blockchain ledger. In the Know Your Customer / Know Your Business Use Case, this component refers to the transactions submitted for registering a new client on-chain, updating an existing one and withdrawing the KYC/KYB information of an existent participant. It is implemented with the following functions:

- `CreateParty(id string, name string, signature string, validityPeriod string, idLegalNumber string, idDocType string, interVATNumber string, phone int)`: The CreateParty function is creating a new client on the ledger by updating the world state with the received data and by appending new blocks at the end of the ledger. The "CreateParty" function receives the data containing the client's KYC or KYB information in each function parameter field and converts them into the appropriate format. Afterwards, the party data is organized in a single structure and it is finally submitted to the world state.
- `UpdateParty(id string, name string, signature string, validityPeriod string, idLegalNumber string, idDocType string, interVATNumber string, phone int)`: The UpdateParty function is changing the information of an existing client on the ledger by updating the world state with the newly received details of the participant. The UpdateParty function receives the data containing these details in each function parameter field and as in CreateParty converts them into the appropriate format in order to submit them to the world state.
- `DeleteParty(id string)`: The DeleteParty function withdraws the information of an existing client on the ledger by updating the world state. The DeleteParty function receives the ID of the party that should be deleted and afterwards submits the new withdrawal transaction on the ledger.

The *Blockchain Reader* component is used in order to read the ledger state and fetch a particular party's KYC or KYB submitted information:

- `ReadParty(id string)`: This function is fetching the appropriate party KYC or KYB data from the ledger by querying the ID of the client. After a successful execution of this function, the demanded information is returned in JSON format.
- `PartyExists(id string)`: This function is requesting if a specific party exists on the ledger along with their KYC/KYB information. The party that exists is validated by its own ID, whereas if there is no relevant submission on the blockchain ledger the function claims on the negative.

The *Smart Contract Executor* component's main purpose is to consolidate the business intelligence of the defined use case and execute the chaincodes on the blockchain ledger.

- `Contract_SubmitTransaction()`: The function initiates the on-chain recording of new KYC/KYB information. This function sends the transaction to all dedicated peers of the respective channel in the blockchain network. In case that all of the aforementioned peers endorse the transaction, the endorsed proposal is sent to the ordering service in order to be committed by each of the peers to the ledger of the channel.
- `Contract_evaluateTransaction()`: The function is responsible for initiating the retrieval of KYC/KYB documents of a specific party. This function communicates with a single peer and the requested results will be returned.
- `Contract_ApproveAccess()`: The function is responsible for initiating the access approval of the KYC/KYB documents from financial organization A to B, i.e. the latter being the one that requested the data. This function alters the data view of organization B through the permission of organization A.

The *Blockchain Authenticator* component is responsible for performing the authentication of the blockchain network user in order to grant access to a specific channel of the blockchain network.

- `Authenticate()`: The function is responsible for the authentication of the organization user in order to access a specific channel of the blockchain network. This function receives the appropriate keys and certificates and allows connection of the organization user to the specified smart contract of the respective channel.

The *Blockchain Encryptor* component performs the encryption of the related data that are involved and produced within the smart contract execution.

- `Encrypt (party string)`: The function performs the aforementioned data encryption adopting AES-256 encryption.

The *Blockchain Decryptor* component performs the decryption of the data encrypted by the Blockchain Encryptor component.

- `Decrypt (party string)`: The function performs the reverse procedure of the *Encrypt* function of the Blockchain Encryptor component by decrypting the respective data.

With regards to the KYC/KYB web application, the following analysis presents the essential services of user interface (UI) framework and their functions that are hitherto developed for the user interaction with the on-chain part of the solution.

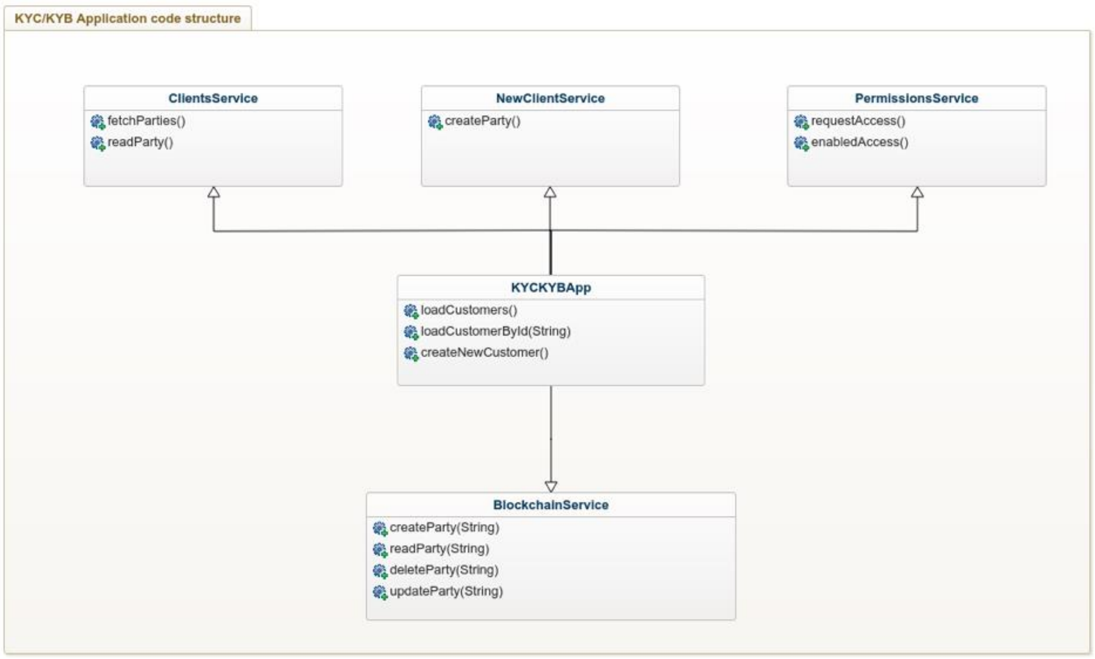

Figure 24: KYC/KYB Application code structure

The *Clients Service* is responsible for retrieving the requested KYC or KYB information from the blockchain ledger and for delivering it on the web interface of the participant channel organization. The implemented functions are as follows:

- `fetchParties()`: The specific function retrieves the KYC or KYB information of all the clients, as seen by the participant channel organization user.
- `readParty()`: The specific function retrieves the KYC or KYB information of a specific client, as seen by the participant channel organization user.

The *New Client Service* is responsible for delivering the KYC or KYB data of a newly inserted client on the specified blockchain endpoints in order to be submitted on the ledger. The implemented function is the following:

- `createParty()`: The specific function creates a new client record on the blockchain ledger by delivering the provided client KYC/KYB information.

The *Permissions Service* is responsible for switching the access control state of a financial organization. The implemented functions are as follows:

- `requestAccess()`: A financial organization requests access control for KYC/KYB information of another organization.
- `enabledAccess()`: The specific function returns in case access control is enabled for a channel organization.

The *KYC/KYB App* constitutes the main user web interface endpoint through which the end-user can initiate all the permitted actions and navigate into the application. KYC/KYB App high-level functions are implemented as follows:

- `loadCustomers()`: This function is the starting point from where the end-user retrieves all the clients' KYC or KYB information. By triggering this function, the full stack integrated mechanism retrieves the requested data back to the user.
- `loadCustomerById()`: This function is exploited by the end-user in order to retrieve the KYC or KYB information of a specific client.
- `createNewCustomer()`: The specific function inserts the data of a new client in the system in order to be handled by the New Client Service.

The *Blockchain Service* is responsible for submitting new use case related data on the blockchain ledger once it is triggered by the web user interface. The functionalities immediately enable the corresponding chaincode functions while this service's operating functions are implemented as follows:

- `createParty(string)`: The specific function prompts the respective chaincode function in order to submit the provided client KYC/KYB information on the blockchain ledger.
- `readParty(string)`: The specific function prompts the respective chaincode function in order to retrieve the required party KYC/KYB information from the blockchain ledger.
- `deleteParty(string)`: The specific function prompts the respective chaincode function in order to remove the requested client from the blockchain ledger.
- `updateParty(string)`: The specific function prompts the respective chaincode function in order to update any of the requested client information on the blockchain ledger.

Video Link of Solution in INFINITECH Marketplace:
https://marketplace.infinitech-h2020.eu/infinitech/kyc-kyb-on-chain-data-governance

4.3 Tokenization

Updates from D4.7:
The current section provides the updated short description of the work performed for the specific use case since the previous version. As this work is performed within the context of T4.4, the completed documentation is available on deliverables D4.10, which was delivered on M14, and D4.11 that will be delivered on M22.

4.3.1 Description of the solution

Digital tokens can play an important role as accelerators of value exchange in any business ecosystem, and particularly in financial and insurance business networks. Representing assets as tokens allows using the blockchain ledger to establish the unique state and ownership of an item, and the transfer of ownership using a consensus mechanism that is trusted by multiple parties. As long as the ledger is secured, the asset is immutable and cannot be transferred without the owner's consent.

The technical core of the tokenization use case focuses on the implementation of ERC 20 standard workflows as a chaincode in Hyperledger Fabric. The implementation of the tokens functionality at the application level frees us, on the one hand from any changes at the infrastructure level of Fabric, and on the other hand gives us flexibility for interoperability with other applications.

The outcomes of the work performed so far can be summarized in the following three assets:

- The blockchain based application that includes the corresponding chaincode that implements the ERC20 standard in addition to the Mint function.
- A recording showing a tutorial of ERC 20 functions, description of the applied blockchain network applied, and the usage of the chaincodes.
- A video targeted for non-technical people explaining the essence and benefits of tokens on a blockchain network.

On the other hand, future work revolves around two main directions:

- Generalization of ERC20 to ERC1155 multi-token standard which allows representation of combinations of fungible tokens and non-fungible tokens within the same contract.
- Collaboration between T4.4 and T4.5 in designing a decentralized and federated machine learning framework in which tokens are a means for incentivizing organizations, trading their data or their insights in a secured and trusted manner.

As documented in deliverable D4.7, the tokenization use case is the outcome of the work performed in Task 4.4 and it is was documented in detail in "D4.10 - Blockchain Tokenization and Smart Contracts – I" on M14 per the INFINITECH Description of Action. Nevertheless, a snapshot of the work performed so far, as well as of the future work, was presented in the current section as a supplementary documentation for the developed blockchain use cases within the context of WP4. The work performed during the period from M15 to M22 will be documented in detail in "D4.11 - Blockchain Tokenization and Smart Contracts – II" which will be delivered on M22 per the INFINITECH Description of Action.

Video Link of Solution:

- ERC 20 tutorial video is available at: https://ibm.box.com/s/2n4ztnj33jt82y2rqosa2rsdpp6k5rdv.
- Video for the essence and benefits of tokens on a blockchain network is available at: https://marketplace.infinitech-h2020.eu/infinitech/tokenization-on-hyperledger-fabric-erc20-chaincode

5 The INFINITECH Blockchain Network

Updates from D4.7:

Since the "horizontal" blockchain capability that addresses the GDPR will not be included, the INFINITECH Blockchain Network has been adjusted, removing the relevant node, the respective network elements and resources that were reserved for this capability.

The cornerstone of every blockchain deployment is the underlying blockchain network in which the immutable transaction ledger is maintained by a distributed network of peer nodes. As explained in Section 2, within the context of the INFINITECH project, the Hyperledger Fabric will be exploited in order to provide the required permissioned blockchain network on which the distributed applications presented in Section 4 will be deployed.

The main elements of the blockchain network are as follows [15] :

- **Peer nodes**: Peers are the nodes that are formulating the blockchain network, hosting the distributed ledger(s) and the smart contract(s) (chaincode), as well as other services of the network.
- **Organisations**: The peers are owned by different organisations that are contributing to the blockchain network as members of the network. Each peer has a digital identity in the form of a digital certificate issued by the Certificate Authority of each organisation.
- **Channel:** A channel refers to the isolated logical structure of a collection of peers which establish a sub-network. In a channel, only its members can see the particular transactions of the specified ledger and can have access to the deployed smart contract (chaincode) as well as all the data being transacted. Each channel is regulated by a specific channel policy as defined by the participants of the channel and is completely isolated as they cannot communicate with other channels. Furthermore, each channel contains all the configurations of communication between the peers along with the list of peers and which peers are endorsing, anchor and leader peers.
- **Ledger:** The distributed shared ledger where the current and historical state of the facts are maintained. Each peer has a copy of the ledger that they share through a channel.
- **Smart Contract (chaincode):** The trusted distributed applications that contain the business logic of the system. They are responsible for all the interactions with the ledger and can be invoked by all the external applications that have access to them.
- **Ordering service:** The purpose of the ordering service, deployed on a peer or a set of peers, is to formulate the transactions into blocks and ensure the delivery of the blocks to the peers of the channel. Furthermore, it guarantees that the transactions are included in the proper order and that all peers have the same updated ledger. The most common ordering services are Solo, Kafka, and Raft. Just like peers, ordering nodes belong to an organization. And similarly to what is done with peers, a separate Certificate Authority should be used for each organization.
- **Membership Services Provider (MSP):** The MSP acts as a Certificate Authority (CA) that issues and manages the certificates utilised in order to authenticate the identity and roles of the members of the network. As Fabric implements permissioned blockchain networks, all participants of the network must be authenticated in order to perform any kind of operation. Everything that interacts with a blockchain network, including peers, applications, administrators and orderers, acquires their organizational identity from their digital certificate and their Membership Service Provider (MSP) definition.

- **Network Policy:** The blockchain network is created and regulated by a network policy that is applied across the whole network with permissions that are determined prior to the network creation by the involved organisations.

To facilitate the implementation of the trusted distributed applications that were presented in Section 4, the consortium designed the suitable blockchain network that is capable of hosting all three applications that will developed in the context of Task 4.3, as well as the tokenization use case that will be developed within the context of Task 4.4. It should be noted that the designed network can be easily adjusted and further expanded in the case where more applications will be designed and developed as the project evolves.

Figure 25 depicts the initial testbed topology of the Hyperledger Fabric network. In total, seven (7) nodes will be set up in order to serve all the applications. Four (4) out of seven (7) nodes will be used purely as the foundation of the blockchain network, hosting the ledger, the chaincodes and the peers together with the single orderer that will be used initially, while the other three (3) will be mainly hosting the external applications that interact with the blockchain network and they will also be providing resources for the required certificate authorities.

In particular, each of the seven (7) nodes will correspond to a single VM, which interacts with the entire blockchain network N. Three (3) organizations (FO1, FO2, FO3) will be created in total, with two peers for FO1 (P1, P4) and one peer for FO2 (P2) and FO3 (P3) respectively, while nodes 1, 2, 3 and 4 will be hosting one peer each as well. In this initial topology of the network, there will exist four (4) channels (C1, C2, C3, C4) with their own copy of the ledger (L1, L2, L3, L4). While (P1, P2, P3) will be added in the first three channels and will be able to deploy smart contracts inside the first three channels (S1, S2, S3), P4 will be added in the C4 channel and will be able to deploy smart contracts inside the corresponding fourth channel (S4). Finally, in node 1, the single orderer (O) will be included.

Regarding nodes 5, 6, and 7, they will be mainly hosting the external applications as well as one Certificate Authority each (CA1 or CA2). Besides Certificate Authority CA1, node 5 will contain the entire KYC/KYB external application, which will be interacting with channels C1 and C2. Two distinct KYC/KYB applications will be developed in order to operate with the two distinguished channels C1 and C2. In a similar manner, besides Certificate Authority CA2, node 6 will be hosting the whole Tokenization external application, which will be interacting with channel C3. Finally, the external application for the Consent Management will be hosted in node 7, which will include CA1 and will be interacting with channel C4.

It should be noted that initial design specifications of the INFINITECH Blockchain Network have been adjusted to remove the node and the respective network elements that were reserved for the GDPR capability, since as it is explained in Section 3, the implementation of this capability will not proceed.

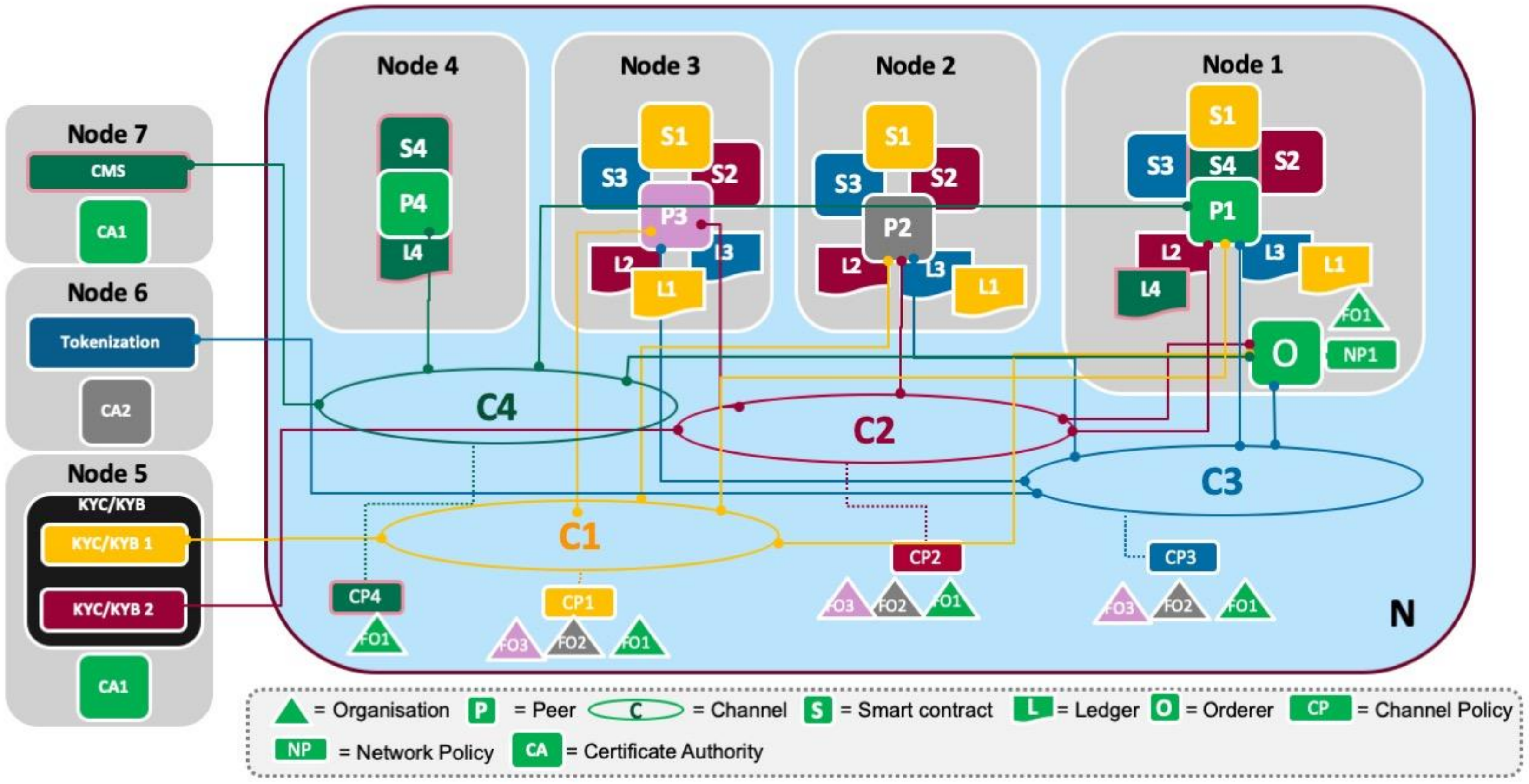

Figure 25: The updated INFINITECH Blockchain network

Regarding the node interactions, the deployment and the network topology, the following table illustrates the number of nodes, the hosted services and the overall hardware resources needed per node.

Table 23: INFINITECH Blockchain network details

Node Name / Service	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7
Orderer	Yes						
Peer	Yes	Yes	Yes	Yes			
CA					Yes	Yes	Yes
KYC/KYB					Yes		
Tokenization						Yes	
CMS							Yes
GDPR							
Hardware Resources	2 vCPUs / 4GB RAM / 120 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk
Total Hardware Resources	18 vCPUs, 40 GB RAM, 1.10 TB Disk storage						

6 Baseline Technologies and Tools

Updates from D4.7:

No updates were introduced with respect to the previous version.

The development of the presented applications is based on a set of carefully selected technologies and tools that will facilitate the development teams to deliver the described functionalities and use cases. Towards this end, the consortium decided to exploit a set of well-established technologies and tools that include open-source software, libraries and frameworks which will enable the smooth and effortless implementation, as well as integration, of the designed use cases. The criteria during the selection process were their level of maturity, their applicability to the designed use cases and the compatibility of each selected technology and tool with the rest ones.

The following table presents the list of core technologies and tools that will be leveraged during the implementation phase of the presented applications.

Table 24: Baseline Technologies and Tools

Software Name	Short description	Version
Hyperledger Fabric	The open source enterprise-grade permissioned distributed ledger technology (DLT) platform.	2.2.0
Fabric - CA	The Hyperledger Fabric CA is a Certificate Authority (CA) for Hyperledger Fabric that will be used to authorise the various components and users.	1.4.8
Vault	Vault provides high-level policy management, secret leasing, audit logging, and automatic revocation to protect sensitive information.	1.5.3
Consul	The Consul storage backend is used to retain Vault's data in Consul's key-value store.	1.8.0
PostgreSQL	PostgreSQL is the RDBMS that will be utilised to store the certificates information by the Fabric CA.	12.0

7 Conclusions

The purpose of the deliverable herewith, entitled "D4.8 - "Permissioned Blockchain for Finance and Insurance - II" was to document the efforts undertaken within the context of T4.3 "Distributed Ledger Technologies for Decentralized Data Sharing" of WP4. To this end, the deliverable included the updated documentation that supplemented the information documented in deliverable D4.7 with regard to the updated specifications of the INFINITECH blockchain network, the updated design specifications of the blockchain applications which are implemented on top of the INFINITECH blockchain network, and the technical details of the first prototype version of the blockchain applications.

More specifically, the deliverable reported the results of the in-depth analysis of the key characteristics and offerings of the blockchain technology and the definition of the role of the blockchain technology within the INFINITECH RA. It should be noted that the results remained unchanged from the previous iteration of the deliverable and they were reported for coherency reasons.

In addition to the results of the analysis of the blockchain technology, the deliverable presented the updates in the design specifications of the blockchain applications which are implemented within the context of the INFINITECH project, as well as the implementation specifications of their first prototype version. In detail, the two distinct applications, namely (i) the Consent Management and (ii) the Know-Your-Customer (KYC) / Know-Your-Business (KYB), were presented. At first, a comprehensive description of the addressed business operation, the exploited key functionalities of the blockchain technology and their high-level architectures were presented highlighting the updates from the previous version, where needed. The design specifications were accompanied by the detailed documentation of the use cases which were addressed along with the respective sequence diagrams. Finally, the implementation details of their first prototype version was presented by documenting the various implemented services and their functions, as well as an overview of their source code structure.

The deliverable also presented the updated design details of the INFINITECH blockchain network, documenting the updated network topology, in which the nodes and their roles as well as the interactions between them were properly defined. The updated INFINITECH blockchain network consists of seven nodes, four of which are utilised as the blockchain network (peers), while three of them serve as the hosting environment for the developed external applications with the blockchain network and the certificate authorities. To support the execution of the implemented blockchain applications, three distinct organisations are registered and are interacting via four distinct channels, while three distinct smart contracts (chaincode) are deployed for the four different ledgers that are hosted by the four peers.

Finally, the deliverable reported the list of baseline technologies and tools that were utilised for the deployment of the INFINITECH blockchain network, as well as the development of the designed blockchain applications. The list remained unchanged from the previous iteration and was selected by applying multiple criteria such as the level of maturity of these baseline technologies, their applicability to the designed use cases and their compatibility with the rest of the technologies, were taken into consideration.

Deliverable D4.8 constitutes the second iteration of the work performed within the context of T4.3. It provided the updated documentation on top of the information presented in deliverable D4.7. It provided the details of the first prototype version of the blockchain application. Nevertheless, T4.3 remains active until M27 as per the INFINITECH Description of Action hence, as the project evolves, new optimisations and enhancements might be introduced as the outcomes of the analysis of the feedback that will be collected by the pilots of the project and by the stakeholders of the platform, as well as of new requirements that may arise. The forthcoming version of the deliverable, namely deliverable D4.9, which will be released on M27 will contain the final documentation of the performed work.

Table 25: Conclusions (TASK Objectives with Deliverable achievements)

Objectives	**Comment**
Exploit the key characteristics and offerings of the blockchain technology	The proposed solutions successfully exploit the key characteristics and offerings of the blockchain technology to effectively address the needs of the financial and insurance sectors with trusted, immutable, decentralised and transparent solutions in the form of blockchain applications.
Provide the blockchain infrastructure which supports the requirements of the financial and insurance sectors	The designed and delivered permissioned blockchain network provides the basis for the implementation of secure and decentralised blockchain applications which can be leveraged to provide innovative services, products and offerings.
Design and deliver the required solutions that address core business cases of the financial and insurance sectors	The delivered solutions (Consent Management, KYC/KYB) address core business use cases of the financial and insurance sectors exploiting the benefits of the permissioned blockchain technology with a set of trusted distributed blockchain applications deployed on robust and secure permissioned blockchain network.

Table 26: Conclusions – (map TASK KPI with Deliverable achievements)

KPI	**Comment**
Design and deliver a permissioned blockchain network	*Target Value = 1* The specific KPI is successfully achieved with the presented solution, namely the INFINITECH Blockchain network, which was designed and documented in detail in the current deliverable. The designed blockchain network consists of seven nodes (four peers and three external application hosts), three organisations, four channels, four copies of the ledger and four deployed smart contracts.
Development of permissioned blockchain solutions	*Target Value >=2* The specific KPI is successfully achieved with the delivered permissioned blockchain applications, namely the Consent Management and the Know-Your-Customer/Know-Your-Business that effectively address two core business use cases of the financial and insurance sectors. The first prototype versions of the aforementioned blockchain applications are delivered within the current deliverable.

Appendix A: Literature

[1] D. Yaga, P. Mell, N. Roby and K. Scarfone, Blockchain technology overview, National Institute of Standards and Technology, 2018.

[2] P. Treleaven, R. Gendal Brown and D. Yang, "Blockchain Technology in Finance," *Computer,* vol. 50, no. 9, pp. 14-17, 2017.

[3] M. Niranjanamurthy, B. N. Nithya and S. Jagannatha, "Analysis of Blockchain technology: pros, cons and SWOT," *Cluster Computing,* vol. 22, no. S6, pp. 14743-14757, 2018.

[4] M. Casey, J. Crane, G. Gensler, S. Johnson, N. Narula and C. A. Wyplosz, The impact of blockchain technology on finance, Geneva: International Center for Monetary and Banking Studies (ICMB), 2018.

[5] "Hyperledger Fabric – Hyperledger," 2020. [Online]. Available: https://www.hyperledger.org/use/fabric. [Accessed 5 September 2020].

[6] "Introduction — hyperledger-fabricdocs master documentation," Hyperledger, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html. [Accessed 03 September 2020].

[7] "EU data protection rules," European Commission, 2020. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en. [Accessed 27 August 2020].

[8] D. Deuber, B. Magri and S. A. K. Thyagarajan, "Redactable Blockchain in the Permissionless Setting," *2019 IEEE Symposium on Security and Privacy (SP),* pp. 124-138, 2019.

[9] G. A. Stevens, B. Magri, D. Venturi and E. Andrade, "Redactable Blockchain – or – Rewriting History in Bitcoin and Friends," *2017 IEEE European Symposium on Security and Privacy (EuroS&P),* pp. 111-126, 2017.

[10] "Consent Receipt Specification – Kantara Initiative," Kantara Initiative, 2020. [Online]. Available: https://kantarainitiative.org/download/7902/. [Accessed 1 September 2020].

[11] "Supporting Material - Archived Groups - Kantara Initiative," Kantarainitiative.org, 2020. [Online]. Available: https://kantarainitiative.org/confluence/display/archive/1+-+Supporting+Material. [Accessed 2 September 2020].

[12] N. Kapsoulis, A. Psychas, G. Palaiokrassas, A. Marinakis, A. Litke and T. Varvarigou, "Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture," *Future Internet,* vol. 12, no. 2, p. 41, 2020.

[13] A. Polyviou, P. Velanas and J. Soldatos, "Blockchain Technology: Financial Sector Applications Beyond Cryptocurrencies," *Proceedings,* vol. 28, no. 1, p. 7, 2019.

[14] I. Karagiannis, K. Mavrogiannis, J. Soldatos, D. Drakoulis, E. Troiano and A. Polyviou, "Blockchain Based Sharing of Security Information for Critical Infrastructures of the Finance Sector," *Computer Security Lecture Notes in Computer Science, Computer Security. IOSEC 2019, MSTEC 2019, FINSEC 2019,* vol. 11981, pp. 226-241, 2020.

[15] “Blockchain basics: Hyperledger Fabric,” IBM Developer, 2020. [Online]. Available: https://developer.ibm.com/technologies/blockchain/articles/blockchain-basics-hyperledger-fabric/. [Accessed 29 August 2020].