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

D4.10 – Blockchain Tokenization and Smart Contracts - I

Revision Number	3.0
Task Reference	T4.4
Lead Beneficiary	IBM
Responsible	Fabiana Fournier
Partners	IBM, BOUN, ENG, GFT, and HPE
Deliverable Type	Report (R)
Dissemination Level	Public
Due Date	2020-11-30
Delivered Date	2020-11-30
Internal Reviewers	Final
Quality Assurance	Internally Reviewed and Quality Assurance Reviewed
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]
IBM	Lead Beneficiary	Fabiana Founier, Inna Skarbovsky, Wael Shama
BOUN	Contributor	Can Ozturan, Alper Sen
UNP	Internal Reviewer	Tiago Teixeira
ENG	Internal Reviewer	Domenico Messina
INNOV	Quality Assurance	Dimitris Drakoulis

Revision History

Version	Date	Partner(s)	Description
0.1	2020-09-15	IBM	ToC Version
0.2	2020-09-20	IBM	Initial contributions to Sections 1 and 2
0.3	2020-09-30	IBM	Initial contribution to section 3
0.4	2020-10-02	IBM	Inclusion of *mint* function and references
0.5	2020-10-21	IBM	Updated Sections 1-3
0.6	2020-10-28	BOUN, IBM	Contributions to Section 4
0.7	2020-11-05	IBM	Finalisation of section 2, 3, and 4
0.7	2020-11-08	IBM	Section 5 and executive summary
1.0	2020-11-09	IBM	First Version for Internal Review
1.1	2020-11-12	UNP	Internal Review
1.2	2020-11-16	ENG	Internal Review
2.0	2020-11-22	IBM	Version for Quality Assurance
3.0	2020-11-28	IBM	Version for Submission

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

[2] Can be left void

Executive Summary

Blockchain aims to build a trusted digital transaction network envisioned to create a value network. Digital asset tokenization is a vital area to focus on in order to ensure that the digital manifestation reflects the real-world assets movements. From its initial use as the underlying technology for cryptocurrencies, distributed ledger and blockchain technology have expanded to include creating, buying, exchanging, and managing tokens — unique digital representations of things of value. These things of value can be debt instruments, contracts, commodities, services, and much more.

The proliferation of decentralized applications based on blockchain technologies is reaching its all-time-high nowadays because customers are more informed and aware of what "decentralized application" really means: it's a trustless and democratic system, where the trust is not in the hands of a single entity but it relies on algorithms and cryptographic techniques; in such a system every transaction occurs without any intermediary in a transparent and highly secure manner. Corporates, banks and FinTech companies can benefit from all these advantages through the blockchain functionalities of the INFINITECH platform.

The document in hand, titled D4.10 "Blockchain Tokenization and Smart Contracts - I" summarizes the work realized within the context of T4.4 "Tokenization and Smart Contracts Finance and Insurance Services" in work package 4 from month 4 to month 14 of the project. The aim of this task is "to enhance the permissioned blockchain of the project with tokenization functionalities, as means of enabling digital assets trading" (from INFINITECH DoA).

Hyperledger Fabric (simply Fabric) has been selected by the INFINITECH project as the underlying blockchain platform for the project. However, Fabric does not have native tokens functionality. Therefore, the aim of the task is to extend Fabric with tokens functionality to support digital trading in the financial and insurance sectors. To this end, the purpose of this deliverable is to introduce the reader to the world of tokens, to describe the first steps that have been taken to incorporate tokens capabilities as part of smart contracts in Fabric, and to present future directions and collaborations that will leverage this initial work.

Token is a digital representation of ownership of a good, entitlement to a service, or some combination thereof. Blockchain seeks to improve information security and transparency by sharing encrypted data among network members. Due to its emphasis on security and trust, blockchain is a great fit in the financial sector and many scenarios can benefit from it. Examples of potential use cases of tokens in the insurance sector include, among others, claims processing, risk assessment, and fraud detection.

Our starting point is the ERC 20 standard for tokens, chosen for its popularity and familiarity among the partners in the project and the fact that is based on the account model approach for tokens. This deliverable details the implementation of ERC 20 functions (with the addition of one function used for the minting of new tokens) as smart contracts in Fabric. Furthermore, we provide a description of future steps that will be carried out in collaboration with other tasks in the project to leverage this initial work.

It is important to note, that although the deliverable is of type "R" (only report), we provide a full demonstrator of the token workflows implemented as smart contracts in Fabric along with a self-explanatory recording of the work using an illustrative example.

The document in hand constitutes the first version of the blockchain tokenization and smart contracts deliverable. Next versions will include elaboration of the foreseen directions to pursue.

Table of Contents

List of Figures

List of Tables

Abbreviations/Acronyms

Abbreviation	Definition
ACL	**Access Control List**
AML	Anti-Money Laundering
API	Application Programming Interface
BC	Blockchain
CA	Certificate Authority
CAGR	Compound Annual Growth Rate
CBDC	Central Bank Digital Currencies
CLI	Command Line Interface
DAG	Directed Acyclic Graph
DoA	Description of Action
DLT	Distributed Ledger Technology
ERC	Ethereum Request for Comments
EVM	Ethereum Virtual Machine
ICO	Initial Coin Offering
IPFS	InterPlanetary File System
IPO	Initial Public Offering
IT	Information Technology
ITO	Initial Token Offering
KYC	Know Your Customer
M	Month
MSP	Membership Service Provider
NFT	Non-fungible token
STO	Security Token Offering
T	Task
UTXO	Unspent Transaction Output
WP	Work Package
ZKAT	Zero-Knowledge Asset Transfer

1 Introduction

One important aspect of any blockchain, whether public or private[3], is asset tokenization. This refers to the representation of any asset into its digital form for trading which can later be bought, sold, exchanged or redeemed for any other digital or physical value. Assets can be anything tangible or intangible and can vary from luxury wines to gold or data, as well as fiat money[4] such as USD and EUR called stable coins.

The tokenization market is expected to grow from USD 983 million in 2018 to USD 2,670 million by 2023, at a Compound Annual Growth Rate (CAGR) of 22.1% during the forecast period [1]. Therefore, the importance for a blockchain to provide tokens support is becoming of vital importance.

Cryptographic tokens, or 'tokens' for short, are programmable digital units of value that are recorded on a distributed ledger protocol such as a blockchain. 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. Tokens can represent tangible assets, such as goods moving through a supply chain or a financial asset being traded. Tokens can also represent intangible assets such as loyalty points. Tokens can be fungible (that is, having the same value with other tokens of the same class) or non-fungible (unique, not be interchangeable), or a combination of the two. Because tokens cannot be transferred without the consent of the owner, and transactions are validated on a distributed ledger, representing assets as tokens allows reducing the risk and difficulty of transferring assets across multiple parties.

The INFINITECH consortium decided to exploit the Hyperledger Fabric open source enterprise-grade permissioned Distributed Ledger Technology (DLT) platform [2] as the underlying blockchain platform in the project. Hyperledger Fabric (simply Fabric) lacks support for tokens so far. We aim at extending Fabric with tokenization capabilities to be leveraged by the use cases in finance and insurance sectors in the project.

Generally speaking, the work on tokens reported in this deliverable relies on the ERC 20 standard [3] from Ethereum for the tokenization of assets, which is the de-facto standard in many financial and other blockchain applications for fungible tokens. Our first demonstrator implements ERC 20 as a chaincode (smart contract in fabric) on Hyperledger Fabric and can be accessed at: https://ibm.box.com/s/2n4ztnj33jt82y2rqosa2rsdpp6k5rdv.

This deliverable describes the work carried out from month 4 (M04) to M14 of the project around tokenization, gives a brief introductory section to understand the main concepts in the domain and addresses future phases in our work. We assume the reader is familiar with DLTs and specifically Fabric. For a thorough understanding of how Fabric blockchain technology works refer to the "Hands-on Blockchain with Hyperledger" [4].

It should be noted that although this deliverable is typed as Report, we also provide a first implementation and demo of our work in tokenization (refer to Section 3).

1.1 Objective of the Deliverable

The purpose of this deliverable is to report the outcomes of the work carried out within the context of Task 4.4 (T4.4) "Tokenization and Smart Contracts Finance and Insurance Services" from month 4 (M4) to M14 of the project. In this first iteration, the work that has been performed was mainly focused on exploring

[3] *Public* blockchains (also called *permissionless*) are implementations of the distributed ledger where the data inside the blockchain (transactions) is open to the public and everyone can take part as a node while *private* blockchains (also called *permissioned* blockchains) operate inside a previously defined network of participants and therefore more suitable for business scenarios.

[4] https://www.investopedia.com/terms/f/fiatmoney.asp

potential implementation options for tokens on top of Fabric and implementing the ERC 20 tokens standard as smart contracts as first step of enhancing Fabric with tokenization capabilities.

Hence, the main objectives of the deliverable at hand are as follows:

- To document main characteristics of digital tokens and their potential benefits and applicability to the financial and insurance sectors. A background section containing main concepts and terms required for the understanding of the proposed solution is presented.
- To extend Hyperledger Fabric with tokens capabilities.
- To demonstrate the feasibility of the approach by implementing a first demonstrator around ERC 20 tokens standard.
- To address different directions and future steps to be taken to take full advantage of the tokenization mechanism being developed for the INFINITECH platform.

It should be noted that, according to the INFINITECH Description of Action (DoA), Task 4.4 lasts until M30, and therefore two more versions of the deliverable will be released on M22 and M30 with deliverables D4.11 and D4.12 respectively. Hence, the upcoming iterations of the deliverable at hand will extend the content of this document with the necessary updates of the work undertaken, taking into consideration the evolvement of the project and blockchain implementations.

1.2 Insights from other Tasks and Deliverables

Deliverable D4.10 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 T4.4 “Tokenization and Smart Contracts Finance and Insurance Services”. The task is mostly related to T4.3 “Distributed Ledger Technologies for Decentralized Data Sharing” and the corresponding deliverable D4.7 “Permissioned Blockchain for Finance and Insurance” whose first version was submitted in M12 of the project. The designed blockchain network devised in the scope of T4.3 and the technological decisions are an essential input to the activities performed in T4.4.

In addition, it also relates to T4.5 “Secure and Encrypted Queries over Blockchain Data” as this task aims at devising a foundation for creating a personal data market for data thorough tokenization mechanisms; and to T7.4 “Predictive Financial Crime and Fraud Detection” as partner BOUN involved in both Tasks aims at enriching their graphs over public blockchains with outcomes from the work on tokens resulting from T4.4 on permissioned blockchains (refer to Section 4.2).

1.3 Structure

This document is structured as follows:

- Section 1: introduces the document, describing the context of the outcomes of the work performed within the task and highlights its relation to other tasks and deliverables of the project;
- Section 2: provides a background on tokens main concepts;
- Section 3: Tokens implementation presents the details of the tokens implementation carried out so far in the project;
- Section 4: describes future possible directions and extensions to the work in tokens; and
- Section 5 concludes the document.

2 Background

Blockchain (BC) is a distributed ledger system in which information on transaction details is shared and verified by peer-to-peer network participants. The transaction details verified by the network's consensus mechanism can be added to existing blocks and once added into the chain, a block cannot be modified. Because all the transactions within a blockchain system are validated and recorded by consensus of the network nodes, the need for a trusted central entity is eliminated. Centralized consensus sytems are more prone to manipulation thus blockchain can increase security. Through decentralization, blockchain can reduce transaction costs by directly connecting consumers and suppliers and minimizing conflicts or errors that can occur in contracts through automation (i.e., smart contracts) [5].

Digitization of assets is a process in which the rights to an asset are converted into a digital token on a blockchain. Ownership rights are transmitted and traded on a digital platform, and the real-world assets on the blockchain are represented by digital tokens [6].

The public's interest in blockchain began when the world's first cryptocurrency, Bitcoin, appeared in January 2009 [7]. Bitcoin is a case of the successful application of blockchain, which is the first global decentralized cryptocurrency. Unlike a currency, a token operates through a smart contract on top of a blockchain [5].

As a medium of exchange, tokens can act as a currency themselves. In this respect, they can also be termed as the local currency of a blockchain application. In the main, crypto tokens are used beyond mere exchange by leveraging their most salient feature: being programmable. In this regard, they are used to trigger certain functions in the smart contract(s). Moreover, tokens may be linked to off-chain assets. They can serve as means of fundraising, pre-order or investment, as well as building an ecosystem or a community [8].

In the following sub-sections, we cover some key concepts and terms necessary to understand the scope of our work.

2.1 Key terminology

Tokenization is the process of converting rights to an asset into a digital token on blockchain.

Token is a digital representation of ownership of a good, entitlement to a service, or some combination thereof. Digital token is a digital representation of assets or rights, usually governed by a smart contract on blockchain used to accelerate the exchange of value. Token can be seen as a programming concept that standardize a lifecycle and management of owned entities: creation, ownership change, proof of ownership, deletion. Tokens may represent pure digital assets like voting rights or digital identities (digital), be bound to physical objects as in smart property or smart objects (physical), be tied to virtual reality objects (virtual) or represent legal rights granted by law or agreed between parties (legal) [9]. The value of a token depends mainly on supply, demand, and the trust that the participating community has in it, which is based on credibility and service [8].

Asset refers to the full breadth of goods and services that can be tokenized including earnings streams, ownership rights, limited use rights, servicing rights, and many others. Table 1 presents some examples of possible physical and non-physical assets and their respective tokens.

Table 1 - Examples of assets and tokens

Asset type	Asset	Token
Physical	Home ownership	House token
Physical	Car ownership	Car token
Non-physical	Rights to rent a car	Car rental token
Non-physical	Rights to access an application	License token

Non-physical	Rights to redeem bonus	Rewards token
Non-physical	Rights to watch movie	Movie ticket token

Any valuable asset (tangible or not, real or financial instrument) can be tokenized. The resulting digital asset can be considered a coin (or *cryptocurrency*, which serves as mediums of payment) or a *token* (digital representation of a tradable asset or utility).

A blockchain **wallet** is a cryptocurrency (i.e., digital or virtual currency that is secured by cryptography) wallet that allows users to manage different kinds of cryptocurrencies, for example, Bitcoin or Ethereum. A blockchain wallet helps someone exchange funds easily. Transactions are secure, as they are cryptographically signed. The wallet is accessible from web devices, including mobile ones, and the privacy and identity of the user are maintained. Therefore, a blockchain wallet provides all the features that are necessary for safe and secure transfers and exchanges of funds between different parties.

2.2 Tokens classifications

There exist many ways to classify tokens. For example, tokens can be differentiated based on their interchangeability.

Fungible: Interchangeable tokens, having the same value with other tokens of the same class (e.g.; money, loyalty points, etc.). In other words, a fungible token is one that is equal in all aspects to another token of the same category.

Non-fungible: Unique, not be interchangeable (e.g.; art, property title, plane ticket, etc.). Non-fungible tokens (NFT's) can be used in real life for making, managing, and transferring the rights of real-world assets. For example, with the help of real asset-backed non-fungible token, you can automate the land deed process from creation to transfer, all on the blockchain.

Hybrid/semi-fungible: Combines fungible and non-fungible characteristics. For example, a specific airline flight (a fungible thing) and a non-fungible asset, such as a specific seat on that flight. Or a non-fungible token representing a pool of real estate mortgages and a fungible one representing a fraction of that mortgage pool that can be purchased by corporations or consumers as securities.

Table 2 compares between fungible and non-fungible tokens properties.

Table 2 - Fungible versus non-fungible tokens

Fungible tokens	Non-fungible tokens
Interchangeable	Not interchangeable
Divisible	Non-divisible
ERC 20 standard (see Section 2.6)	ERC 721, ERC 1155 (see Section 2.6)

Among the main tokenizable assets, we can distinguish [10]:

Blockchain native currencies: referred to as *cryptocurrencies* and which are used as means of payment within a blockchain network (e.g., Bitcoin). Most cryptocurrencies are "non-fiat" currencies, meaning they are not distributed by a state or government. Thus, the value of a cryptocurrency is not pegged to anything but is instead established by the community of users depending upon what they are willing to exchange for the token and is determined by its ability to purchase goods and services. Stable coins are similar to cryptocurrencies but they attempt to offer price stability and are backed by a reserve asset. Stable coins have gained traction as they attempt to offer the best of both world's—the instant processing and security or privacy of payments of cryptocurrencies, and the volatility-free stable valuations of fiat currencies.

Government issued currencies: CBDC or *Central Bank Digital Currencies*, which are a digital form of fiat money issued by a government or its affiliates, subject to the state's legal, regulatory and monetary policy framework. Fiat money is government-issued currency that is not backed by a physical commodity, such as gold or silver, but rather by the government that issued it.

Securities: *Security tokens* are blockchain native digital bonds, equities, derivatives, and other securities which draw their value from an external, tradable financial assets and are subject to the state's security laws and regulations. A security token can be like a stock and represents ownership of a corporation or company or a contract or debt instrument. A security token can carry with it equity in the organization and, depending on the way the token is distributed, can convey certain rights to the token holder (e.g., voting rights or dividend distribution). These are tokens that are often distributed through initial coin offerings (ICOs) or initial token offerings (ITOs) or security token offerings (STOs).

Access or usage rights: *Utility tokens* are programmable blockchain assets that provide users with future access to a product or service. Utility tokens are generally not designed as investments but rather function as tickets that enable certain user rights and are generally issued through ICOs (Initial Coin Offerings). The value of the underlying commodity or service can be physical (e.g., similar to a commodities contract for oil or corn) or digital (e.g., access to a network). Regardless of what the underlying asset is, these assets are fungible, meaning the items are indistinguishable from one another. A token for a bushel of wheat, kilowatt hour of electricity, or barrel of oil is exchangeable for any other bushel of wheat, kilowatt hour of electricity, or barrel of oil.

Commodities: *Asset-backed tokens* which represent digitized commodities such as gold, power, gas, diamonds, etc. Differently to securities, they are normally not subject to securities' regulations. Asset tokens are like utility tokens in that the value of the token is tied to an underlying asset. However, the underlying asset connected to the token is not fungible, for instance, the token could represent ownership of a plot of land, automobile, or diamond. In addition, the token could represent ownership of a unique digital asset or intellectual property such as a song or patent.

Platform incentives: *Platform or Protocol tokens* are on-chain valuables designed to incentivize and support the execution of operations of decentralized applications; they are generally issued through ICOs.

Information: such as healthcare records, intellectual property or titles.

Contracts: The blockchain technology supports smart contracts, which essentially act as self-executing contractual agreements.

Collectibles: *Crypto-collectibles* are digital collectibles that can be exchanged for money.

Behaviour describes various aspects of a token lifecycle, a capability or restriction on token. Tokens differentiate themselves based on their behaviours. Common behaviours include but not limited to:

Transferable/Non-transferable: Ability/inability to transfer ownership of the token after it was issued.

Sub-dividable/Whole: Decimal places a token can be subdivided into

Singleton: There can only be a quantity of one, where the token class represents the only instrument.

Mintable: Ability to issue new tokens of the class.

Burnable: Ability to remove tokens from the supply.

Role Support: Ability to have roles defined within the class to allow or prevent certain actions.

2.3 UTXO versus account model

For tokens to be useful, they need to be transferable. The transfer of a token on a blockchain is initiated by the owner, creating a transaction. This transaction informs the network about how much tokens are changing hands and who the new owner is.

Generally speaking, any blockchain is a state machine. With each new block the system undergoes a state transition that happens according to the state transition logic defined in its protocol. The user interactions, mostly transactions, are broadcast to the network, and with each new block, a set of them is permanently recorded. The balances of the transacting parties are updated when the system transitions to the new state. The difference between the UTXO (unspent transaction output) and the account model lies in the way the bookkeeping is handled, that is how they record the state and transitioning from one state to another.

In the UTXO model, the movement of assets is recorded as a Directed Acyclic Graph (DAG) between addresses, whereas the account model maintains a database of network states. In the UTXO model, account balances are calculated on the client-side by adding up the available unspent transaction outputs. In the account model, with each new block, the state of the system is updated according to the transactions contained in the block. The number of accounts remains constant and independent of the number of transactions conducted, as long as the number of users or smart contracts remains constant. In the UTXO model, the entire graph of transaction outputs, spent and unspent, represents the global state. In the account model, only the current set of accounts and their balances represents the global state. The conceptual difference is that the account model updates user balances globally. The UTXO model only records transaction receipts. The UTXO model does not incorporate accounts or wallets at the protocol level. The model is based entirely on individual transactions, grouped in blocks. Since there is no concept of accounts or wallets on the protocol level, the “burden” of maintaining a user’s balance is shifted to the client-side. Wallets maintain a record of all addresses controlled by a user and monitor the blockchain for all associated transactions. The sum of all unspent transaction outputs it can control determines the current balance.

The account-based transaction model represents assets as balances within accounts, similar to bank accounts. A transaction in the account-based model triggers nodes to decrement the balance of the sender’s account and increment the balance of the recipient’s account. In an account-based model, there is an account for each participant that holds a balance of tokens. A mint transaction creates tokens in an account, while a transfer transaction debits the caller's account and credits another account. The account model keeps track of all balances as a global state. This state can be understood as a database of all accounts, private key and contract code controlled, and their current balances of the different assets on the network.

The account model offers clear advantages when it comes to enabling smart contracts. First, the account model logic is more intuitive. Adding and subtracting balances makes it easier for developers to create transactions that require state information or involve multiple parties. A signed transaction is valid as long as the sending account has a sufficient balance to pay for the execution. Checking a balance is computationally less expensive than verifying if a transaction output is spent or unspent. This is especially true for code-controlled accounts that interact with other smart contracts. Internal transactions between contracts can be carried out in a virtual machine by adjusting the balances of the contracts. The UTXO model creates computational overhead because all spending transactions must be explicitly recorded. Smart contracts in a UTXO model would need to include logic for choosing which outputs to use when sending assets, and logic to handle change outputs. Because the UTXO model is inherently stateless, it forces transactions to include state information, complicating the overall design.

One benefit of the UTXO model is that it allows for the simpler parallelization of transactions in smart contracts. Multiple UTXOs used in different transactions can be processed at the same time since they all refer to independent inputs. In the account model, the result of a transaction depends on the input state. Executing transactions in parallel must be handled carefully. Generally, transactions affecting the same account should be executed sequentially.

2.4 Tokens benefits

Benefits of tokenization lie mainly in higher liquidity, general programmability, and immutable proof of ownership. More precisely, fractional digital ownership lowers the barriers of entry for investors while it increases the liquidity of the tokenized assets. Furthermore, programmability facilitates automated management of investor rights and compliance regarding the tokenized assets, and thus potentially increases

speed. Finally, the immutable traces of transfers provide evidence that the transfers of the digital ownership have not been tampered with [8].

Token economy is a system of incentives that reinforces and builds desirable behaviours. Some of the benefits of utilizing tokens [11]:

- Reliable, secure and transparent management of assets, from creation to settlement, allowing actors to rely on an immutable record of ownership and legal rights, essential for provenance and traceability purposes (relevant for Anti Money Laundering (AML)/Know Your Customer (KYC) compliance).
- Faster and cheaper transactions, thanks to automated exchanges, disintermediation and the elimination of non-value-adding operations.
- Reduced friction and overhead costs in the management of tokens (such as securities, commodities and real estate), including issuance and transfer.

Tokenized assets benefit from the new medium that they operate in[5]:

- **Simplified exchange**: Tokenization creates a more seamless exchange process. Because all tokens exist on the blockchain, a lot of bureaucracy and headache associated with exchanging a valuable asset is automated and maintained by the blockchain. Each transaction that involves a token has sender, receiver, signatures, time-stamp, and amount-paid all baked into the transactions data. This transaction "receipt" is immutable, verifiable, and made available for all to see.
- **Security**: Tokens are signed by the private key making them difficult to forge/steal.
- **Global access:** Blockchains are ubiquitous. Tokens facilitates international trading opening asses to new markets and opportunities. Physical assets that were previously unreachable by international purchasers, are now made accessible through asset tokenization.
- **Freedom and fairness**: Blockchains allow to build a non-custodial web that makes use of The InterPlanetary File System (IPFS) for storing and sharing data in a distributed file system, storing information that can't be taken down or censored by intermediaries.
- **Liquidity:** Because blockchains bring a global audience, assets are able to achieve larger marketplace participation than traditional siloed markets. For many assets, like real estate, liquidity is a significant obstacle that can lower the value of the asset, as the search for finding the right buyer may prove difficult. By opening up the market to a global pool of purchasers, illiquid assets can more likely find liquidity via increased market participants.
- **Interoperability:** Tokens can standardize the value exchange, significantly improving interoperability between different domains, platforms, and solutions.
- **Programmability:** Tokens can be programmed to behave in a certain way, to match specific business requirements. For example, in the custody of a tokenized property, an owner of the property can make conditional scenarios to where the token can change owners.

2.5 Tokens in financial and insurance sectors

Blockchain seeks to improve information security and transparency by sharing encrypted data among network members. Due to its emphasis on security and trust, blockchain is a great fit in the financial sector [5].

Examples of potential use cases of tokens in the financial sector:

Cryptocurrency/Cross boarder Payments: Sending money internationally using existing payment methods can be both expensive and time consuming due to fees that are taken by middlemen. In addition, there are a considerable number of individuals, especially in the developing world, that do not have access to banking services.

[5] https://realt.co/real-world-asset-tokenization-a-new-form-of-asset-ownership/

Fundraising: The traditional financial markets come with its limitations, for example: intermediary fees, Initial Public Offerings (IPOs) are out of reach for many companies due to the cost of floatation, and clearing and settlement can take multiple days.

Letter of credit: Providing letter of credit (critical for international trading) takes a lot of time and is an expensive process.

Credit risk scoring: Existing credit risk scoring process is associated with high fees, low accessibility to credit, and unfairness.

Decentralized exchanges: How to enable exchange without a centralized middleman. In theory, such an exchange enables a wild west of sorts: marketplaces where participants are anonymous, any token can be bought or sold, and no authorities can shut it down.

Debt issuance: How to prove that the company is capable of developing complex product (like aircraft) to get credit from financial institution without exposing sensitive details of the product.

This is a clear classical play for:

Digital Coin: Store of value and medium of exchange not issued by a central authority

Fiat-backed Token: Representation of FIAT money on a distributed ledger. To maintain a 1:1 ratio between the token and its associated fiat currency there should be a method of backing tokens with real fiat currency.

Physical Asset-backed token: Representation of a physical asset, e.g., gold, oil, etc., on a distributed ledger. The token represents a claim on the underlying asset.

Securities token: Securities tokens represent a share in the company that has completed a token sale (i.e., an equity stake in the company).

Examples of potential use cases of tokens in the insurance sector:

Claims processing: How to lower operational costs related to claims processing and improve data accuracy.

Multinational insurance: How to get transparency into insurance that involves multiple brokers working in each country.

Reinsurance/Risk assessment: How to enable traceability of risks when working with reinsurance especially in less regulated areas/countries.

Fraud detection: How to reduce/prevent fraud especially where fraudulent patterns can only be detected across a wide data set, often across multiple insurers.

Crowdfunded insurance: How to securely extend participation in the insurance space to small infesters.

Cryptocurrency backed loans: Provide a lending platform with benefits based on tokens to borrow fiat currency using cryptocurrencies as collateral.

2.6 Tokens standards

The main technology advancement of the Ethereum blockchain [12] is the introduction of a general purpose and a turing complete smart contract system which is manifested in the Ethereum Virtual machine (EVM). In the EVM, a program code is executed by miners and other network participants who verify state changes. A smart contract in Ethereum is typically written in a high-level programming language like Solidity or Viper and compiled to bytecode that is then deployed on the blockchain. The execution requires sending a transaction to the contract's address while specifying which function is to be called given a set of parameters. These functions in turn can call other smart contracts if they have been programmed to do so. Computationally intensive routines, however, are not suitable, as the execution has a cost attached to it. For each operation supported by the EVM, a gas cost is defined in the Ethereum yellow paper [13], where the main cost drivers are operations that store or change values on the blockchain. A transaction therefore needs to contain a sufficient amount of gas in order to guarantee successful execution. The actual costs of a

transaction depend on how much gas is needed, and the gas price a user is willing to pay for each unit of gas. The transaction costs paid by a user are awarded to the miner that includes the transaction in a new block, as they have to verify and execute the transaction [14].

One of the many Ethereum development standards focus on token interfaces. These standards help ensure smart contracts remain composable, so for instance when a new project issues a token, that it remains compatible with existing decentralized exchanges. Standard token interfaces enable applications such as wallets to recognize tokens and interact with them. The community continuously discusses and establishes standard interfaces for tokens in the programming language Solidity, which is prevalent on Ethereum. The standards in Etherium are account-based. The following standards have been accepted so far.

2.6.1 ERC 20 Token Standard

To provide a common form of tokens in Ethereum, a standard interface referred to as Ethereum Request for Comments (ERC) 20 was introduced. The tokens are fungible.

ERC 20 Token Standard [3] is the most widely used and most general token standard that provides basic functionality to transfer tokens, as well as allows tokens to be approved so they can be spent by another on-chain third party. It lists six mandatory and three optional functions as well as two events to be implemented by conforming Application Programming Interface (API). ERC 20 is the most general and most prevalent token standard. Of the deployed compliant token contracts, over 98 % are ERC 20 compliant [8].

Each ERC 20 token implements the six following functions:

- *totalSupply*: returns the amount of token in existence
- *balanceOf*: returns the amount of tokens owned by a certain account
- *allowance*: Returns the remaining number of tokens that a spender will be allowed to spend on behalf of an owner. This is zero by default.
- *transfer*: moves a certain amount of tokens from the message sender account to a recipient account
- *approve*: Sets a certain amount as the allowance of a spender over the caller's tokens
- *transferFrom*: moves a certain amount of tokens from a sender account to a recipient account

Figure 1 shows the interface contract declaring the required functions and events to meet the ERC 20 standard.

```
// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
```

Figure 1 - ERC 20 token standard interface

Many times we use a two-transaction workflow that uses *approve*(), followed by *transferFrom*(). It allows a token owner to delegate their control to another address and most often used to delegate control to another smart contract for distribution of tokens. For example, if company is selling tokens for an ICO, they can

approve a crowdsale[6] chaincode to distribute a certain amount of tokens. The crowdsale chaincode can then use *transferFrom*() to add tokens from the balance of the owner to the balance of each buyer.

Due to its popularity and simplicity, and being account-based, our work on tokens on top of Fabric started with the implementation of ERC 20 as chaincode (smart contract in Fabric) as described in Section 3.

2.6.2 ERC 721 Token Standard

ERC 721 [15] is another free and open-source standard for implementing tokens over the Ethereum blockchain, just like ERC 20. This token standard defines how to create non-fungible or unique tokens on the Ethereum blockchain, thus enabling the tracking of distinguishable assets. Each asset must have its ownership individually and atomically tracked. This standard requires compliant tokens to implement 10 mandatory functions and three events [8].

In this standard just like ERC 20, ERC 721 defines the minimum interface requirements a smart contract must implement to allow non-fungible tokens to exist (i.e., manage, own, and trade). However, it doesn't mandate a standard for token metadata or restrict adding supplemental functions.

2.6.3 ERC 777 Token standard

ERC 777 Token Standard [16] defines advanced features to interact with tokens while remaining backwards compatible with ERC 20. New functions such as *send* and *authorizeOperator* which replace respectively the functions *transfer* and *approve* from ERC 20. It defines operators to send tokens on behalf of another address, and hooks for sending and receiving in order to offer token holders more control over their tokens. This standard requires compliant tokens to implement 13 mandatory functions and five events [8].

2.6.4 ERC 1155 Token Standard

ERC 1155 is a superior token standard for Ethereum based tokens that allow you to create fungible, non-fungible, and semi-fungible tokens all through one smart contract, in contrast to ERC 20 or ERC 721 which either allow a fungible or a non-fungible token [17]. This standard requires compliant tokens to implement six mandatory functions and four events [8].

2.7 Summary

Blockchain underlying technology has the potential to revolutionize the way assets are stored, represented, and moved. It enables individuals to trade valuable assets anywhere and anytime in a reliable and swift manner and to generate and exchange ownership rights in a formal yet agile way. In this chapter we name some of the potential use cases that can benefit from applying tokens on blockchain such as claims processing, risk assessment, and fraud detection.

This section gives a brief peek into the world of tokens on blockchains. Next section details the selected approach based on implementing the ERC 20 standard as a chaincode on top of Fabric to be our first demonstrator. ERC 20 has been selected to be the test case for implementation due to its popularity and simplicity. In addition, it enables a smooth collaboration with other partners in future phases of the project as presented in Section 4.

[6] Crowdsale is a type of crowdfunding which is done through the issuing of cryptocurrency tokens that are purchased by contributors to finance some project.

3 Tokens implementation

As previously stated, the ERC 20 standard has been selected as first step of tokens implementation in the project. In addition to the six functions of the standard, we also added the *mint* function as described below. ERC 20 standard is for fungible tokens, meaning that different units of an ERC 20 token are interchangeable and have no unique properties.

3.1 Design

Our BC network is composed of (Figure 2)

- Two Organizations (orgs) and one *orderer* which uses a single node Raft ordering service[7]
- Two *peers*
- *Certificate Authority* (CA) per Org
- CLI (Command-Line Interface) Client to interact with chaincode.

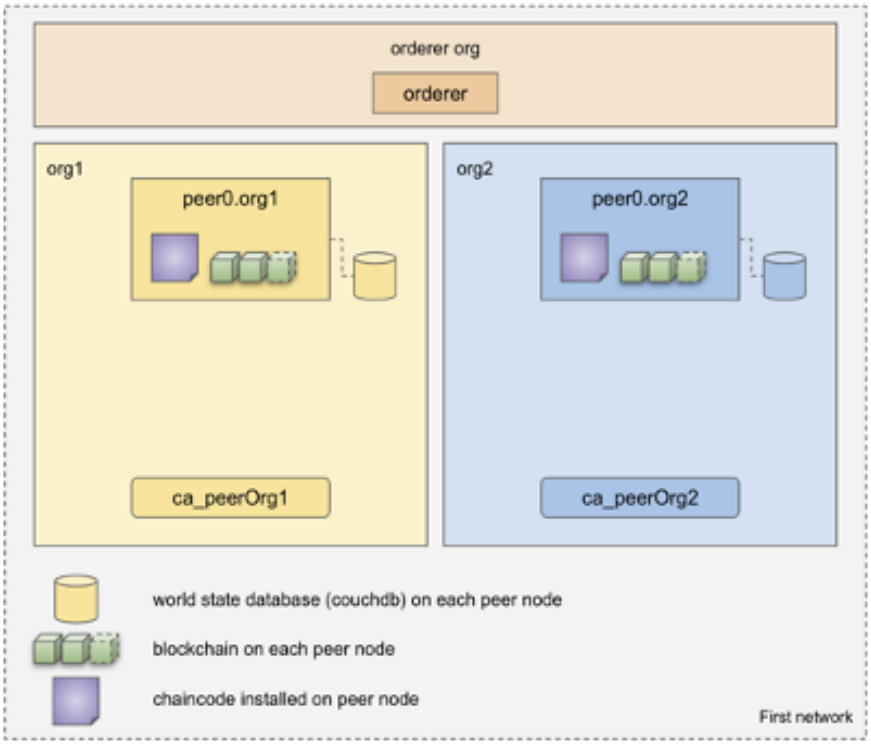

Figure 2 - BC network configuration

Token ownership

- We will use the Membership Service Provider (MSP) of the org
- *ClientIdentity* class in the shim package we can easily get the Id of that particular MSP, and we initialize the world state with this metadata.

[7] New as of v1.4.1, Raft is a crash fault tolerant ordering service based on an implementation of Raft protocol. Raft follows a "leader and follower" model, where a leader node is elected (per channel) and its decisions are replicated by the followers. Raft ordering services should be easier to set up and manage than Kafka-based ordering services, and their design allows different organizations to contribute nodes to a distributed ordering service.

3.1.1 Design assumptions

The following assumptions led the overall solution (no specific order).

- As we have previously stated, we selected Etherium ERC 20 as standard (an account-based model) to be compatible with the rest of the partners in the consortium that develop in Etherium.
- Our implementation includes two organizations representing two token accounts. The *account id* is calculated as a hash of the private key of the user.
- Chaincode development in GO language and demonstrated through CLI command line interface invocations of chaincode's functions.
- Hyperledger nodes (Certificate Authority (CA), orderer, and peer) is deployed using Hyperledger 2.2.0 Docker images to be compatible with the rest of BC demonstrators and is configured and run using Docker Compose. The images used are publicly available in Docker Hub.
    - CA image: hyperledger/fabric-ca:1.4.8
    - Orderer image: hyperledger/fabric-orderer:2.2.0
    - Peer image: hyperledger/fabric-peer:2.2.0

3.1.2 Workflows/use cases

All ERC 20 contracts contain:

Balances

- Allow the token contract to keep track of who owns the tokens
- Each transfer is a deduction from one balance and an addition to another

Balance-allowed

- An owner can delegate authority to a spender to spend a specific amount from their balance

Required functions and events

totalSupply() - Returns the total units of token that currently exist

balanceOf(address) - Returns the token balance of an address

transfer(address, amount) - Transfers amount of tokens to address, from the balance of the address that executed the transaction

approve(spender, amount) - Authorizes spender to execute several transfers up to amount, from the address that executed the transaction

allowance(owner, spender) - Returns the remaining amount that the spender is approved to withdraw from the owner

transferFrom(sender, recipient, amount) - Transfers token amount from sender to recipient. Used in combination with *approve*

Transfer event - Triggered upon successful transfer (call to transfer or transferFrom), even for 0 value transfers

Approval event - Logged upon successful call to approve

We also added

mint(amount) – Creates new tokens and adds them to total supply and to the owner's balance performing the minting. ERC 20 native tokens don't have this feature which makes them a fixed supply token. New tokens can be minted only by an address which is set to be in a Minter role. If no address is in Minter role, then no one can create new tokens.

We will follow the notation of the use cases and sequence diagrams introduced in D4.7 to specify the use cases and sequence diagrams for the functions above.

3.1.2.1 Use Case: Get total supply

ERC 20 is a fixed supply token standard, as such, this function returns the overall fixed amount of tokens within the system. Any authorized party can check the current state of total supply. The returned amount can be changed only by minting additional tokens by a party with a Minter role.

Table 3 - Get total supply use case

Stakeholders involved	Admin, anyone with read access
Pre- conditions	The total tokens amount is initialized/registered, the requesting party is a legal end-user in the network and has read access
Post- conditions	The invoker receives a reply of the total fixed supply amount
Data Attributes	None
Normal Flow	1. The authentication and access roles for the requester are determined 2. In case read access is allowed, the total balance of fixed tokens is returned
Pass Metrics	Requested information is returned to the invoker
Fail Metrics	Requested information cannot be returned if: • The invoker has no access • The supply's amount was not initialized

3.1.2.2 Use Case: Get balance of

Returns the current balance of tokens of the given account. For the given user ID (as appears in the certificate) which is encoded to be the account ID, returns the current balance of tokens.

Table 4 - Get balance of use case

Stakeholders involved	Admin, anyone with read access
Pre- conditions	The requested account exists, the requesting party is a legal end-user in the network and has read access
Post- conditions	The invoker receives a reply of the balance of the requested account

Data Attributes	Account address
Normal Flow	1. The authentication and access roles for the requester are determined 2. The existence of the requested account is determined 3. The current balance of the requested account is returned
Pass Metrics	Requested information is returned to the invoker
Fail Metrics	Requested information could not be returned if: • The invoker has no access • The requested account does not exist

3.1.2.3 Use Case: Transfer

Transfers the specified amount of tokens from the owner's account (invoker) to the recipient's account. Both accounts must be enrolled and authorized in the system, the source account should have at least the amount of tokens required for transfer.

Table 5 - Transfer use case

Stakeholders involved	Account's owner, recipient's account
Pre- conditions	The owner's account exists, and the balance is initialized with non-negative amount of tokens, the owner's account (invoker) is a legal end user in the network and has read access, the recipient's account exists
Post- conditions	The invoker's balance and the receiver's balance are updated with the new amount of tokens, the transaction is written on the chain
Data Attributes	Recipient's address, amount of tokens to transfer
Normal Flow	1. The authentication and access roles for the invoker are determined 2. The existence of the owner's account is determined 3. The current balance of the owner's account is determined. It should at least have the specified amount of tokens to allow transfer 4. The existence of recipient's account is determined 5. The new balances for the owner's account and recipient's accounts are calculated 6. The balances are updated on the ledger
Pass Metrics	1. The balances in the world state are updated 2. The transaction is available on the chain

Fail Metrics	Requested transfer could not be performed if: • The invoker has no access • The owner's account does not exist • The owner's account balance does not have enough tokens • The recipient's account does not exist

3.1.2.4 Use Case: Approve

Approves allowance of the specified amount of tokens from the owner's account (invoker) to the spender's account. This means that the spender's account can use up to the specified amount of allowance tokens from the owner's account (using multiple withdrawals) to transfer to other accounts, until the allowance runs out (useful for third party payments and withdraw workflows allowing transferring payments on behalf of a third party).

Table 6 - Approve use case

Stakeholders involved	Owner's account, spender's account
Pre- conditions	The owner's account exists, and the balance is initialized with non-negative amount of tokens, the recipient is a legal end-user in the network and has read access, the recipient's account exists
Post- conditions	The allowance of the owner's account for the spender's account is updated, the transaction is written on the chain
Data Attributes	Spender's address, amount of tokens for allowance
Normal Flow	1. The authentication and access roles for the invoker are determined 2. The existence of the owner's account is determined 3. The existence of the spender's account is determined 4. The allowance for the spender's account, with the number of allowance tokens is registered on the ledger (new entry created, or previous entry updated with new allowance amount)
Pass Metrics	1. The allowance is updated on the chain 2. The transaction is available on the chain
Fail Metrics	Requested action could not be performed if: • The invoker has no access • The owner's account does not exist • The spender's account does not exist

3.1.2.5 Use Case: Allowance

Returns the current allowance of tokens of a particular spender from the owner's account. This amount will change from initial allowance's value when the spender account invokes *TransferFrom* function to transfer tokens from the allowance either to their own account, or to a third-party account.

Table 7 - Allowance use case

Stakeholders involved	Admin, anyone with read access
Pre- conditions	The token owner's and spender's accounts exist, the invoker is a legal end-user in the network and has read access
Post- conditions	The invoker receives a reply of the current spender's allowance from owner's account
Data Attributes	Owner's account address, spender's account address (allowance recipient)
Normal Flow	1. The authentication and access roles for the invoker are determined 2. The existence of the owner's and spender's accounts are determined 3. The current allowance of the spender's account from owner's account is returned
Pass Metrics	Requested information is returned to the invoker
Fail Metrics	Requested information could not be returned if: • The invoker has no access • The owner's account does not exist • The spender's account does not exist

3.1.2.6 Use Case: Transfer From

Transfers the specified amount of tokens from the owner's account to the recipient's account, adjust the new balance of the owner's account, the new balance of the recipient's account, and the new allowance of the invoker accordingly.

Table 8 - Transfer from use case

Stakeholders involved	Invoker (spender or allowance account), sender's account (from account), recipient's account (to account)
Pre- conditions	• The sender's account exists and the balance is initialized with sufficient amount of tokens • The invoker party is a legal end-user in the network and has read access, it is approved to transfer tokens from owner's account, and has enough allowance

	• The recipient's account exists
Post- conditions	The invoker's balance and the recipient's balance are updated with the new amount of tokens, the allowance of the invoker is updated, the transaction is written on the chain
Data Attributes	Sender's account address, recipient's address, amount of tokens to transfer
Normal Flow	1. The authentication and access roles for the invoker (spender) account are determined 2. The existence of the sender's account is determined 3. The existence of the recipient's account is determined 4. Enough balance of tokens in the sender's account is verified 5. Enough allowance of the invoker is verified 6. The new balances for the sender's account and recipient's accounts are calculated and updated on the ledger 7. The new allowance of the invoker's account is calculated and updated on the ledger
Pass Metrics	1. The balances and allowance in the world state are updated 2. The transaction is available on the chain
Fail Metrics	Requested transfer could not be performed if: • The invoker has no access or was not permitted by sender (using *approve* workflow) to transfer tokens on their behalf • The sender's account does not exist • The sender's account balance does not have enough tokens • The invoker has not enough allowance • The recipient's account does not exist

3.1.2.7 Use case: Mint

Mints additional tokens to the overall total supply. This function can only be performed by an account with the "Minter" role. The minted tokens are added to the total supply and to the minter account balance.

Table 9 - Mint use case

Stakeholders involved	An account who has a role of Minter
Pre- conditions	The owner's account exists, the requesting party is a legal end-user in the network and has write access and approved to mint tokens, the mint amount is a positive number.

Post- conditions	The invoker's balance and total supply balance is updated with the additional amount of tokens, the transaction is written on the chain
Data Attributes	Amount of tokens to mint
Normal Flow	1. The authentication and proper roles for the invoker are determined (the Minter role) 2. The existence of the invoker account is determined 3. Non-negative amount of mint tokens is verified 4. Total supply balance is updated with the amount of newly minted tokens 5. The invoker's balance is updated with the amount of minted tokens
Pass Metrics	1. The balances and allowance in the world state are updated 2. The transaction is available on the chain
Fail Metrics	Requested mint operation could not be performed if: • The invoker account doesn't exist • The invoker account does not belong to organization allowed to mint • The mint amount specified in the function arguments is negative • Failure updating the total supply balance or the minter's account balance

3.1.3 Sequence Diagrams

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

3.1.3.1 Get total supply

In Use Case 3.1.2.1, an authorized account in the blockchain network queries the state of the total supply. Upon the authentication of the user and their role as a reader, the tokenization client interacts with blockchain components in order to check the status of total supply by querying and reading the query result upon decrypting them (Figure 3).

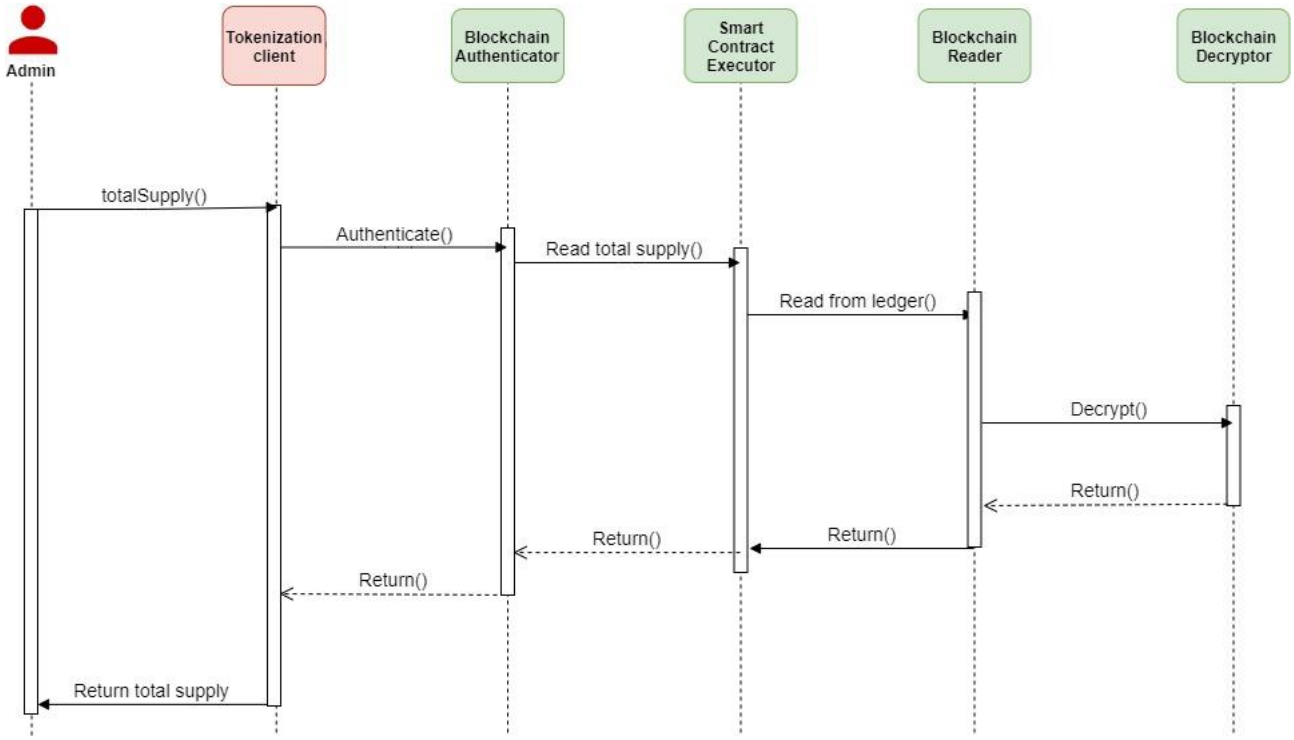

Figure 3 - Get total supply use case sequence diagram

3.1.3.2 Get balance of

In Use Case 3.1.2.2, an authorized account in the blockchain network queries the state of a specified account. Upon the authentication of the user and their role as a reader, the tokenization client interacts with blockchain components in order to check the the balance of the specified account by querying and reading the query result upon decrypting them (Figure 4).

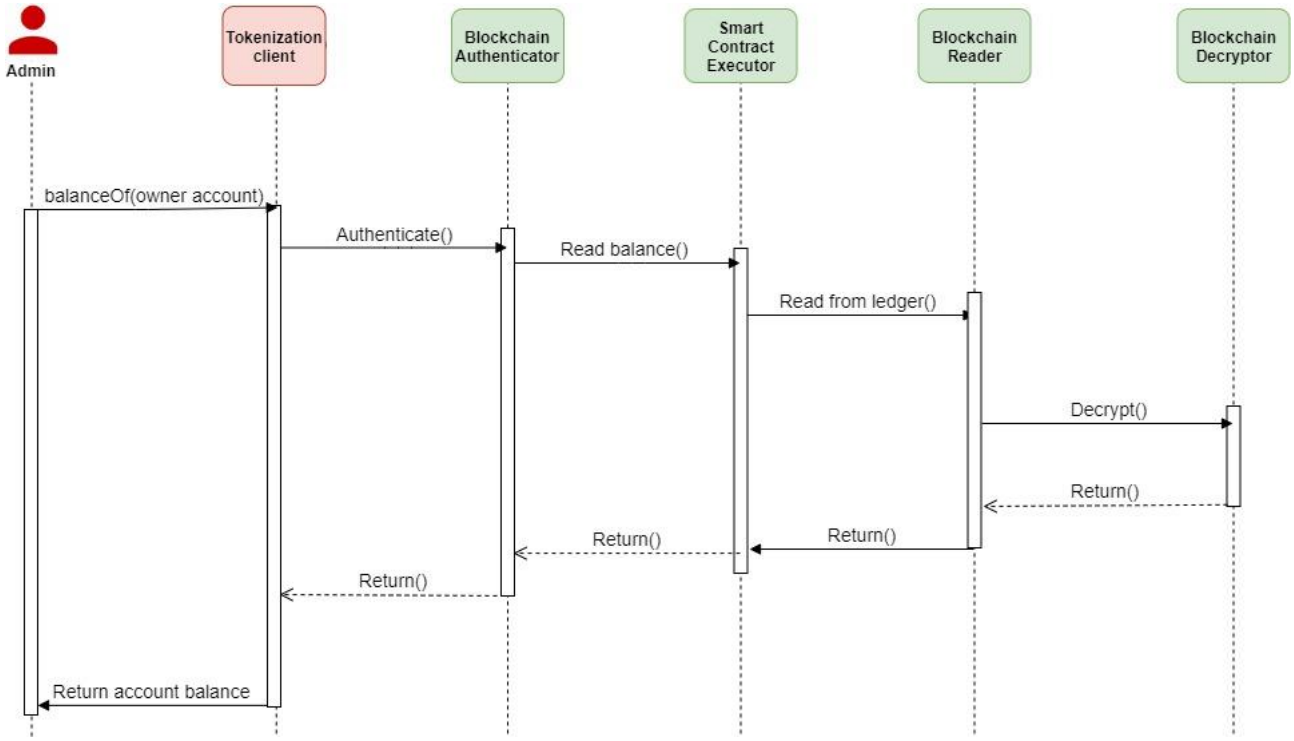

Figure 4 - Get balance of use case sequence diagram

3.1.3.3 Transfer

In Use Case 3.1.2.3, the transfer of tokens from one account to another is handled. In the case of transfer, the tokenization client interacts with the blockchain components in order to retrieve the accounts balance from the ledger, decrypt the query result and update the balance of both accounts, decreasing the amount of tokens in the origin account and increasing the amount of tokens in the recipient account through a new transaction in the ledger (Figure 5).

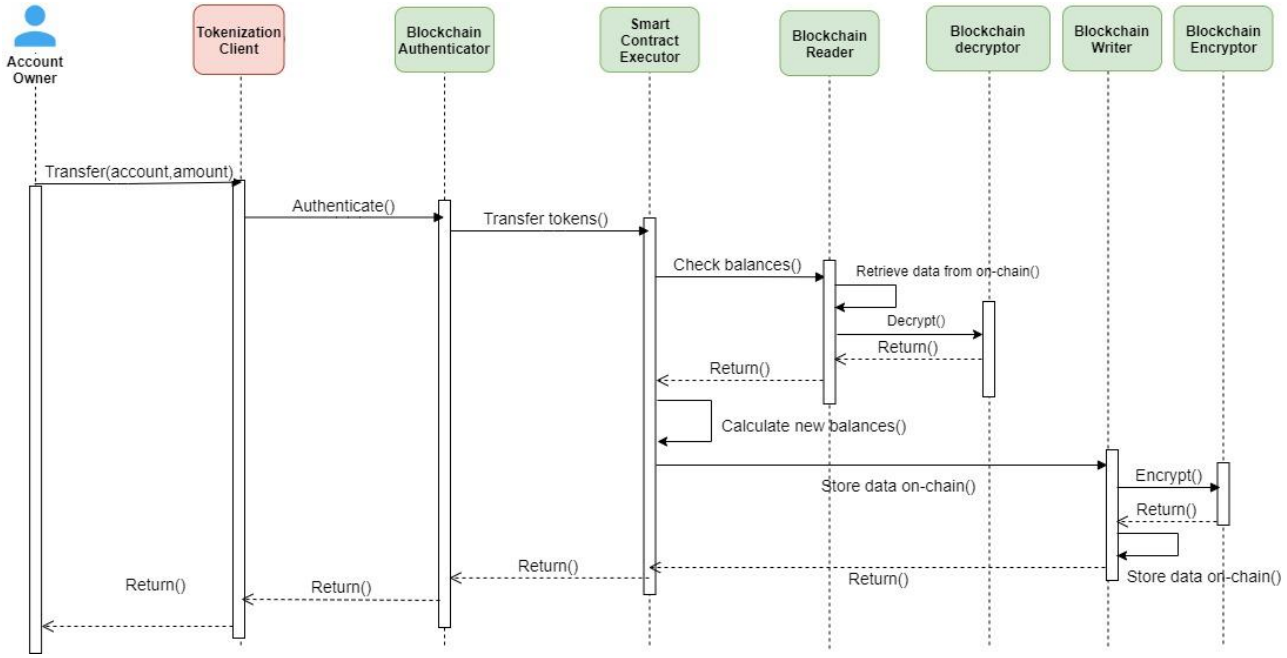

Figure 5 - Transfer use case sequence diagram

3.1.3.4 Approve

In Use Case 3.1.2.4, an account owner is giving to another account (spender) an allowance of a specified amount of tokens from owner's account. This feature is used for approving tokens to be used by spender. It allows spender to transfer the tokens. The amount is added in allowance against approver and spender. For this purpose, the tokenization client interacts with the blockchain components to update the allowance of the spender account from origin account to the specified sum (Figure 6).

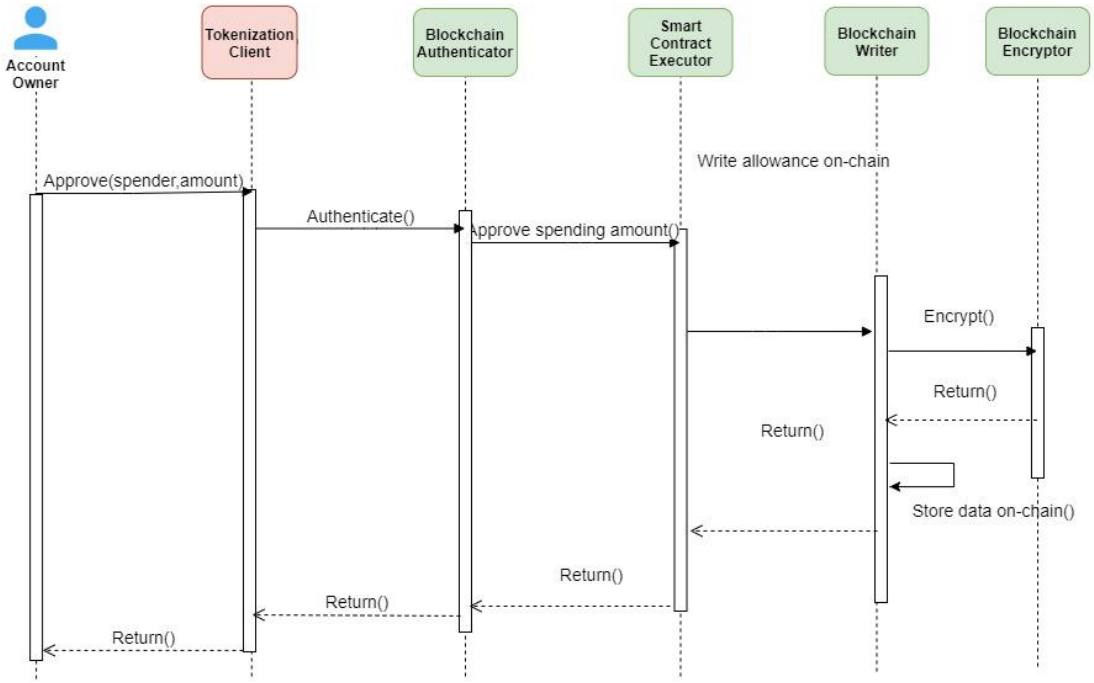

Figure 6 - Approve use case sequence diagram

3.1.3.5 Allowance

In Use Case 3.1.2.5, an authorized account on the blockchain network queries the current remaining allowance of a specified spender account from an origin account. Upon the authentication of the user and their role as a reader, the tokenization client interacts with blockchain components in order to check the remaining allowance by querying and reading the query results upon decrypting them (Figure 7).

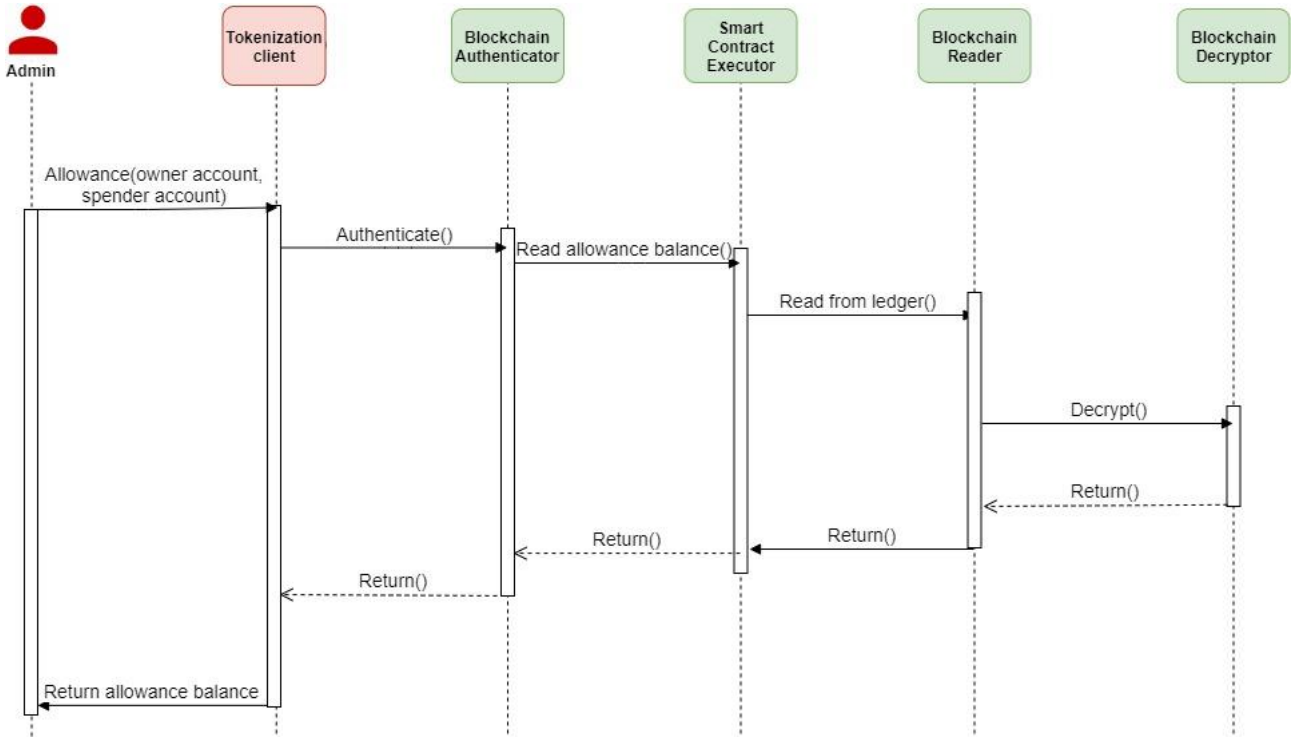

Figure 7 - Allowance use case sequence diagram

3.1.3.6 Transfer From

In Use Case 3.1.2.6, the transfer of tokens from one account to another is handled. In the case of *transfer from*, the tokenization client interacts with the blockchain components in order to retrieve the existing accounts balance and the remaining allowance of the not sender account from the ledger, decrypt the query result and update the balance of both accounts, decreasing the amount of tokens in the origin account and increasing the amount of tokens in the recipient account and also updating the allowance of the sender (Figure 8).

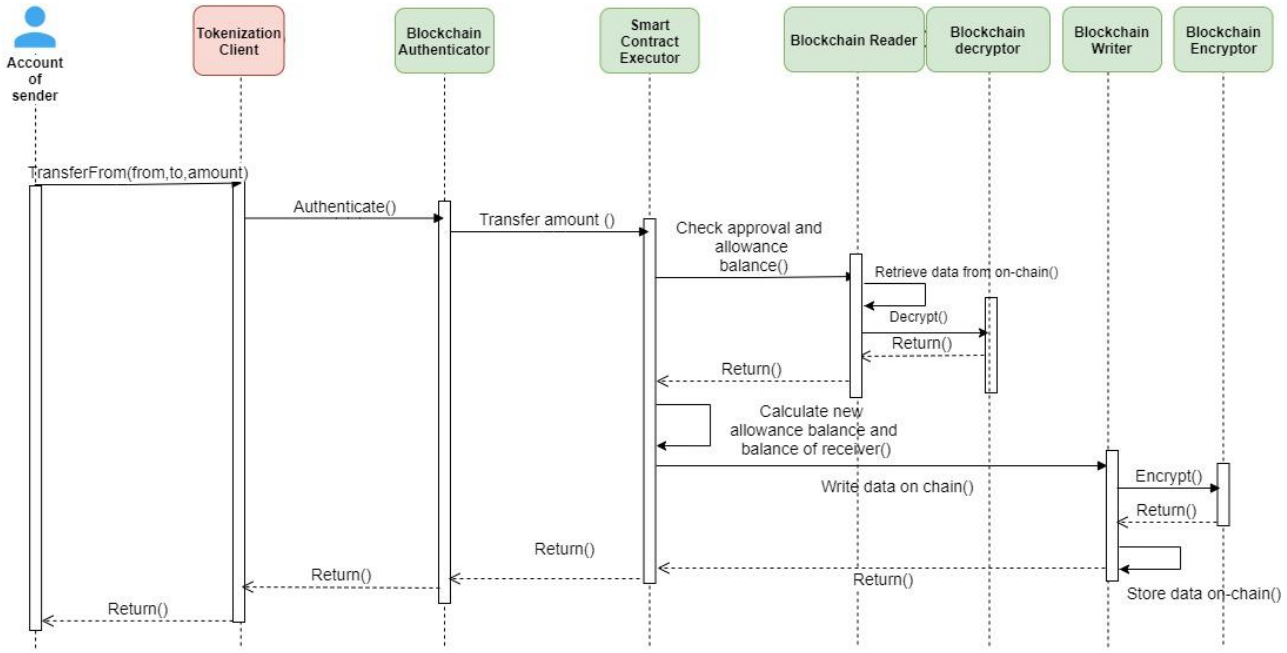

Figure 8 - Transfer from use case sequence diagram

3.1.3.7 Mint

In Use Case 3.1.2.7, minting of new tokens by an account with “Minter” role is performed. The tokenization client interacts with the blockchain components, which in turn, verify that the invoking account has a Minter role and check the amount of tokens to be minted. The blockchain components query the ledger for the current balance of the total supply and the minter account balance. A new transaction increasing the total supply balance and the minter’s account balance is committed to the ledger (Figure 9).

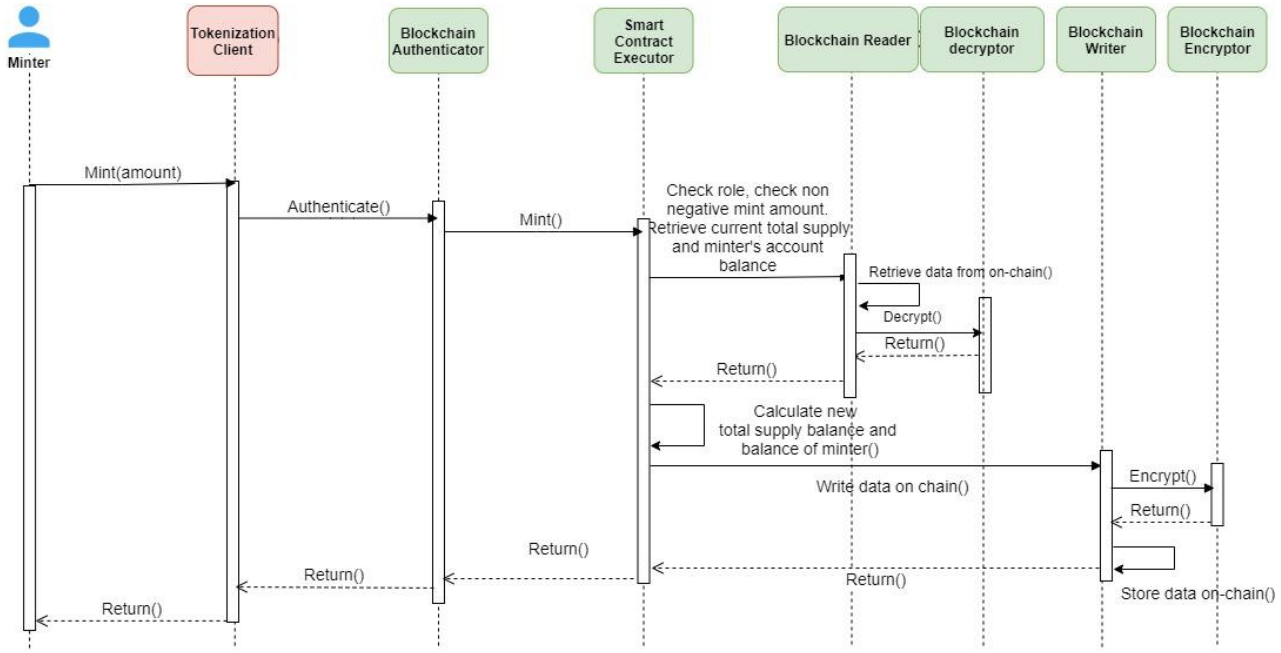

Figure 9 – Mint use case sequence diagram

3.2 Demonstrator

Our initial demonstrator of the ERC 20 implementation is deployed using the BC network presented in Section 3.1. The aim is to demonstrate the different functions workflows.

We assume that only one organization (played by Org1) is in a central banker role and can mint new tokens into their account, while any organization can transfer tokens from their account to a recipient's account. Accounts could be defined at the organization level or client identity level. In our sample, accounts are defined at the client identity level, where every authorized client with an enrolment certificate from their organization implicitly has an account ID that matches their client ID. The client ID is simply a base64-encoded concatenation of the issuer and subject from the client identity's enrolment certificate. The client ID can therefore be considered the account ID that is used as the payment address of a recipient.

The following steps are shown in the demo:

Step 1: mint of tokens (Figure 10)

- Single transaction invoked by Org1 which has the “Minter” role
- Used to add tokens to the overall tokens amount in the system
- The tokens are added to the total supply and to the minter’s account balance (Org1)
- The mint function is invoked with an amount of 20000 tokens (in the demo recording it is done in two steps of 10000 tokens each).
- This is demonstrated by invoking the totalSupply() function by both organizations, returning the current total supply amount (20000 tokens)

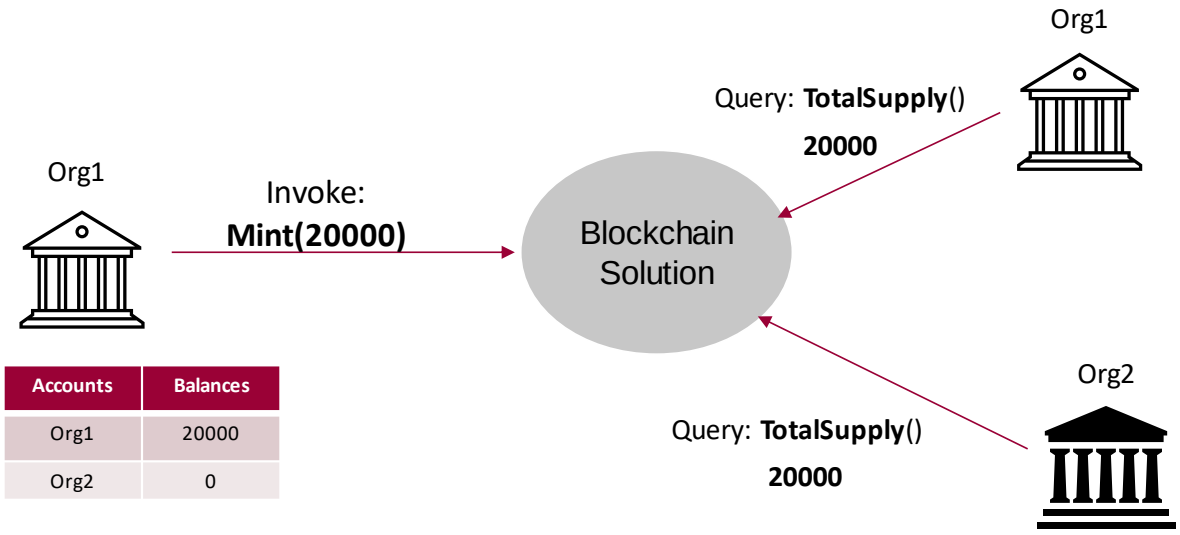

Figure 10 - Mint of 20000 tokens

Step 2: Transfer of tokens (Figure 11)

Used by wallets to send tokens to other wallets. Majority of token transactions happen with this workflow. Let's assume Org1 wants to transfer 1000 tokens to Org2.

- Single-transaction, using the *transfer(address, amount)* function
- Org1 transfers 1000 tokens to Org2, its wallet sends a transaction to the chaincode, with *transfer(Org2_address, 1000)*
- An attempt to transfer larger amount that is currently in the account of the transferring organization (40000 tokens) will result in an exception "client account <originating account address > has insufficient funds"
- The chaincode adjusts Org1 balance (-1000) and Org2 balance (+1000) and issues a *Transfer event.* Calling *clientBalance()* for both of these organizations, verifies that Org1 balance was adjusted to 19000 tokens while Org2 balance was adjusted to 1000 tokens

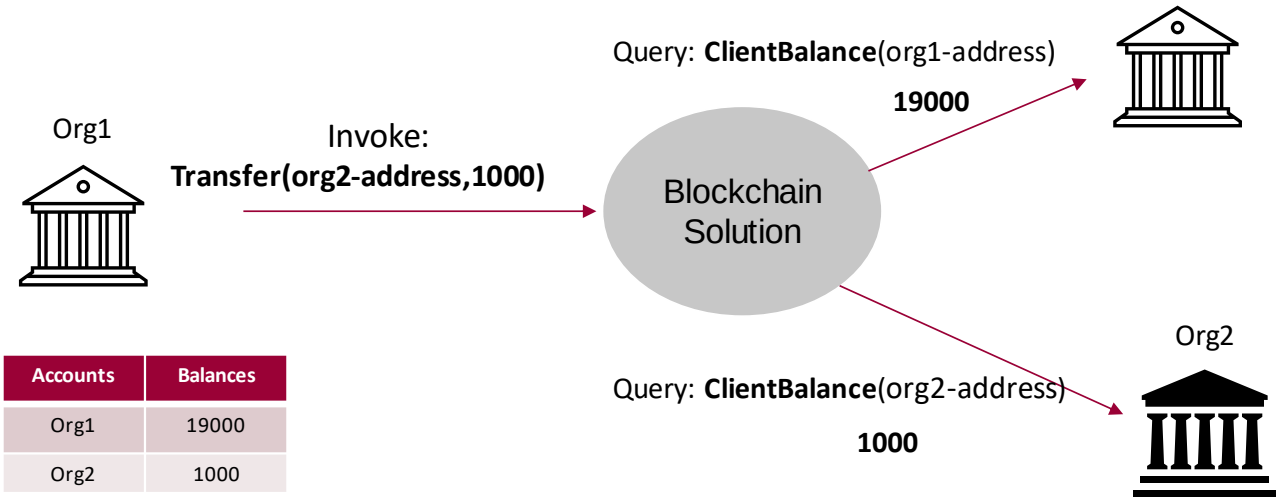

Figure 11 - Transfer of 1000 tokens from Org1 to Org2

Step 3: Approve/Transfer from

Allows a token owner to delegate their control to another address

- A two-transaction workflow that uses *approve(recipient, amount)*, followed by *transferFrom(sender, recipient, amount)*
- In the demo, Org1 calls *approve(Org2_address, 400)* to create an allowance for Org2 of 400 tokens from Org1 account. Calling *allowance(Org1_address, Org2-address*) demonstrates this point – the allowance created for Org2 from Org1 account is 400 tokens. The account balance of Org1 as is seen in *clientBalance(Org1_address)* call is still 19000 as no tokens from the allowance has been transferred yet while the account balance of Org2 is 1000 (Figure 12).
- An attempt by Org2 to transfer larger amount of tokens than what was specified in the allowance, by calling *transferFrom(Org1_address,Org2_address, 3000)* will fail with the exception "client <allowance organization address> approved account has insufficient funds"

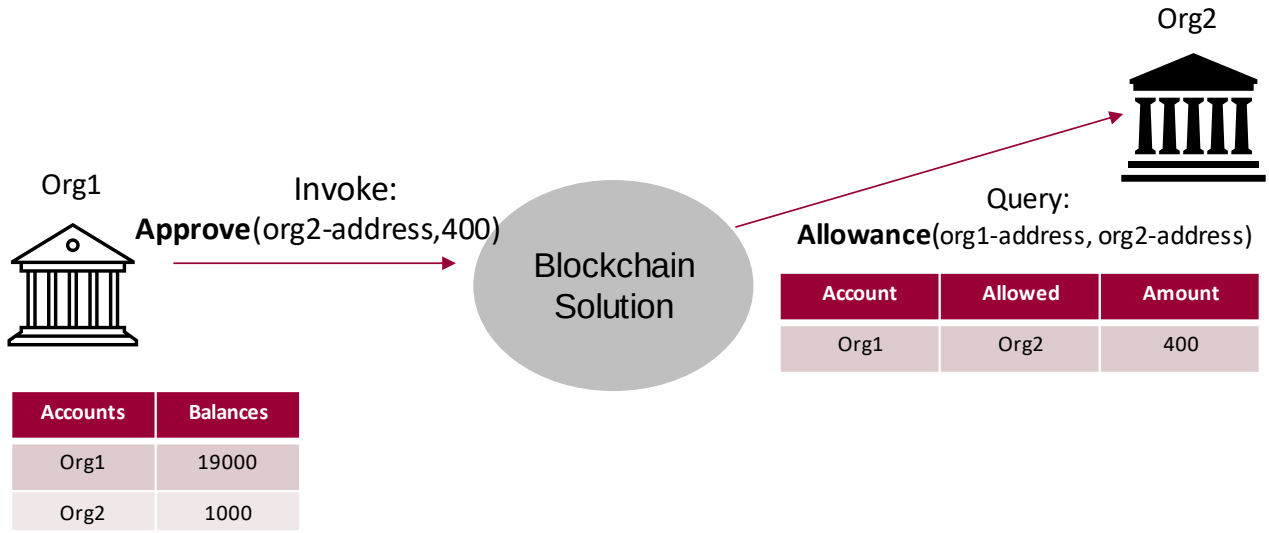

Figure 12 – Authorization of 400 tokens from Org1 to Org2

- Org2 transfers 300 tokens from its allowance from Org1 account to its own account by calling *transferFrom(Org1_address,Org2_address,300)***.** Calling *allowance(Org1_address)* demonstrates that the current allowance of Org2 was reduced to 100 tokens (400-300), while calling *clientBalance(Org2)* shows that Org2 account balance was adjusted to 1300 tokens (Figure 13).

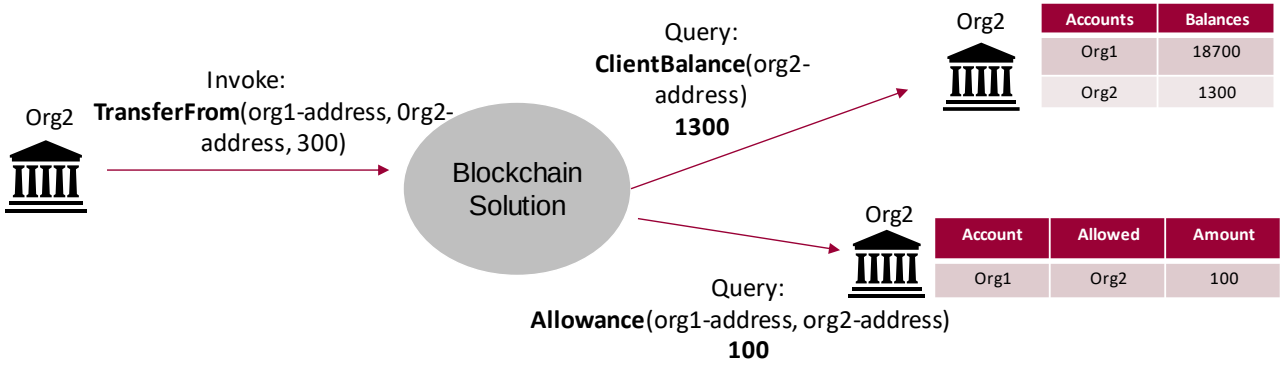

Figure 13 – Transfer of 300 tokens from Org2 to itself from allowance

- Transferring of remaining 100 allowed tokens by Org2 from its allowance from Org1 account to its own account by calling *transferFrom(Org1_address,Org2_address, 100)* will result in the remaining allowance 0. The *totalSupply* of tokens in the system hasn't changed and remains 20000 as the amount of all minted tokens. The balance of Org1 is now 18600 and of Org2 is 1400.

A recording showing a short tutorial of ERC 20 functions, description of the BC network applied, and the usage of the chaincodes (all steps detailed above) can be find at: https://ibm.box.com/s/2n4ztnj33jt82y2rqosa2rsdpp6k5rdv. It will also become available in the INFINITECH market platform, once it is released/launched in the scope of WP8.

3.3 Summary

This section details the design and implementation of ERC 20 functions and the addition of the mint function as chaincode in Fabric. Our demo, publicly available, shows the usage of all the functions along with explanations. Next section presents our insights of how we can extend these workflows to more complex scenarios and how tokenization can be further exploited in collaboration with other tasks.

4 Next steps

There are several natural ways to extend the work carried out so far. In addition, along with the partners of T4.4 “Tokenization and Smart Contracts Finance and Insurance Services” and T4.5 “Secure and Encrypted Queries over Blockchain Data” there are some foreseen directions for collaboration that will leverage the work on tokens as described below. Clearly, the aim is to apply the work to actual scenarios of the pilots in the project.

4.1 Extensions to ERC 20

There are several potential extensions to the work done on tokens as shown in the non-exhaustive list below:

- More complex functions – ERC 20 contains simple functions. This list can be extended with additional functions. In fact, *mint* is an example of such function that has been already implemented.
- Access Control List (ACL) mechanisms to check that an actor can initiate transactions that update the asset. To this end, the chaincode needs to store ownership info of the asset.
- Privacy-preserving exchange of assets with Zero-Knowledge Asset Transfer (ZKAT) - allows transactors to issue assets and request transfer of their assets without revealing anything to the public ledger for the assets being exchanged beyond the fact that the transfer complies with the asset management rules (that each asset is transferred at its owner’s request, and there is no new value created through the transfer).
- Non-fungible tokens – ERC 20 deals with fungible tokens, extension could enlarge the scope to other standards that include NFT, such as ERC 1155 (see Section 2.6 and next sub-section).

4.2 Token Factory

ERC 1155 contract is a more advanced standard than ERC 20 which includes both fungible and non-fungible tokens (see Section 2.6). ERC 20 and ERC 1155 have standard interfaces that need to be provided by Fabric chaincode implementation. Since these tokens can be applied for a wide range of uses by various personnel in an organization with different skills, there is a need for a tool that will enable:

- Design of token contract templates for various purposes and applications
- Customization and instantiation of token chaincode for specific applications
- Deployment of token chaincode from an easy to use interface

Token factory aims to provide a tool that will enable such services. By automating these services with a token factory tool, the users can directly deploy token contracts rather than ask Information Technology (IT) personnel to customize and deploy token contracts. Another advantage may be that standard templates or patterns of verified token chaincodes can be developed for different use cases. These templates or patterns can simply be presented in a menu so users can choose the appropriate template to be configured according to their application. Templates may provide codes for token contract features such as pausable token contracts, token minting functions (periodic, manual, or according to a mathematical function). Simply put, a pausable ERC 20 token is a token that can be paused to prevent any transfers of the token when it is paused. Figure 14 shows the envisioned operation of a token factory.

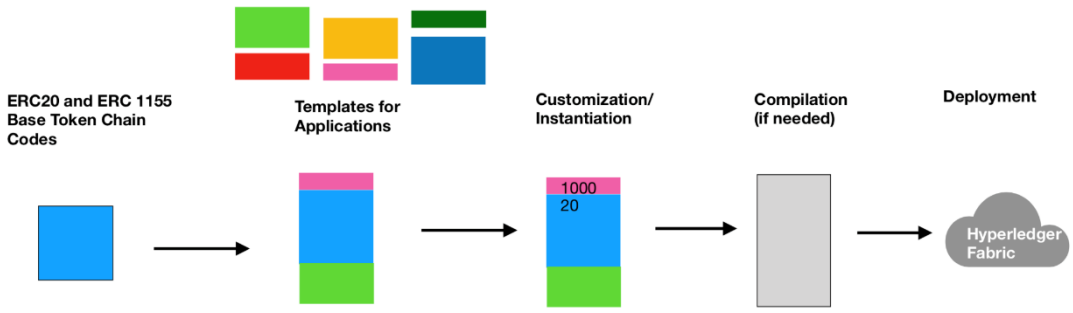

Figure 14 - Token Factory operation

4.3 Scalable transaction graph analysis

Pilot 9 "Analysing Blockchain Transaction Graphs for Fraudulent Activities" aims at developing and deploying a scalable and high performance blockchain transaction graph analysis system for investigating whether customer blockchain account transactions can be traced to fraudulent activities or accounts. Currently, the graph is built from data coming from the public Bitcoin and Ethereum blockchains. The whole raw blockchain data is parsed for cryptocurrency and token transactions which are then extracted and put in a from-to format. Graph analysis is performed to see if customers' addresses can be traced to blacklisted addresses.

The idea is that an additional component will be contributed by T4.4 to the Transaction Graph Analysis System built in scope of T7.4 "Predictive Financial Crime and Fraud Detection". This component will include an ERC 20 and ERC 1155 interfaces for token transactions coming from Hyperledger Fabric. Current public blockchains such as Bitcoin and Ethereum have very low transaction throughput. Bitcoin and Ethereum blockchains can achieve 5-7 and 15 transactions per second respectively. Hyperledger, on the other hand, can achieve 3500 transactions per second. As a result, having a scalable parallel transaction graph analysis system is even more relevant for Hyperledger, since with thousands of transactions per second, the transaction graph's edge set can quickly grow to include billions and billions of edges.

Figure 15 depicts the Hyperledger Fabric interface that is proposed to be developed in task 4.4 next to the Bitcoin and Ethereum interfaces that are developed in WP5 and WP7.

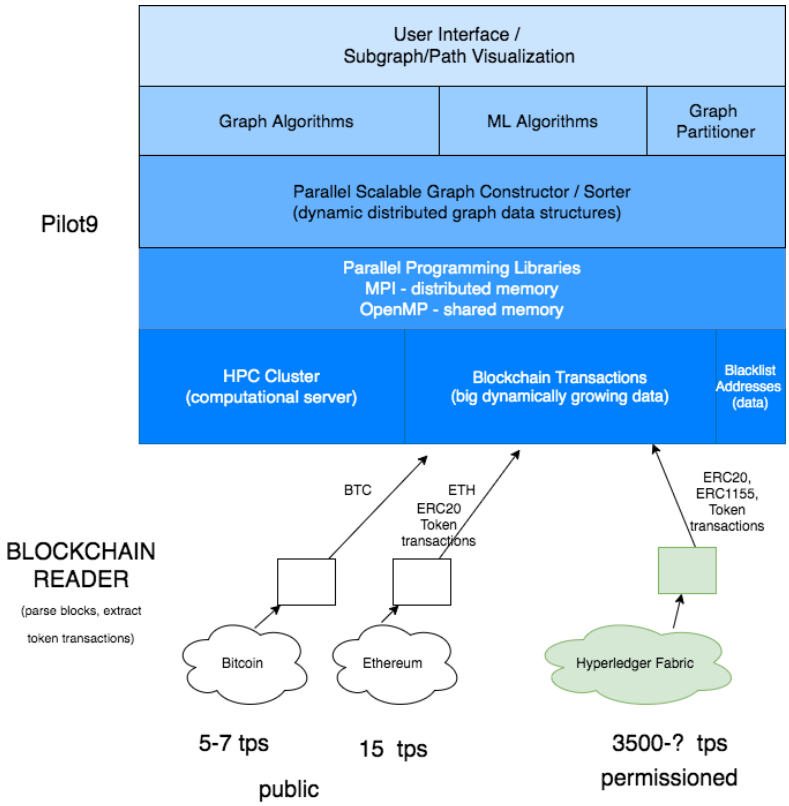

Figure 15 - Pilot 9 scalable transaction graph analysis system with the added Hyperledger Fabric interface for processing ERC 20 and ERC 1155 token transactions

4.4 Collaboration between T4.4 and T4.5

T4.5 "Secure and Encrypted Queries over Blockchain Data" implements and provides a framework for querying encrypted data over the project's permissioned blockchain infrastructure. As stated in the DOA, "in conjunction with the trading and tokenization functionalities of the blockchain, this task will create a foundation for creating a personal data market where customers will be able to trade their data in exchange for tokens on other assets". In other words, T4.5 framework will leverage the tokens implementation in T4.4 "Tokenization and Smart Contracts Finance and Insurance Services" to trade data in a marketplace.

The idea is that a chaincode for data trading (T4.5) will invoke the tokens chaincode (T4.4) to perform operations like transfer of tokens between accounts, checking the balance of accounts, approving a third party to transfer tokens on behalf of account owner, and more. In this way, customers can purchase and trade data using a trading platform.

Figure 12 depicts this idea described by the data flow below. Let's assume we have a BC network with three peers for three organizations insurance companies (A, B, and C).

1. Write data (insurance company A invokes the data sharing chaincode to write insights on the BC)
   1.1. Data committed to ledger of companies A, B, and C
2. Read data (insurance company B attempts to read data)
   2.1. The data sharing chaincode checks
      2.1.1.IF insurance company B has permissioned to read THEN
         2.1.1.1. how many tokens this data costs
         2.1.1.2. INVOKES token chaincode to transfer tokens to insurance company A
   2.2. The read query returns with the requested payload

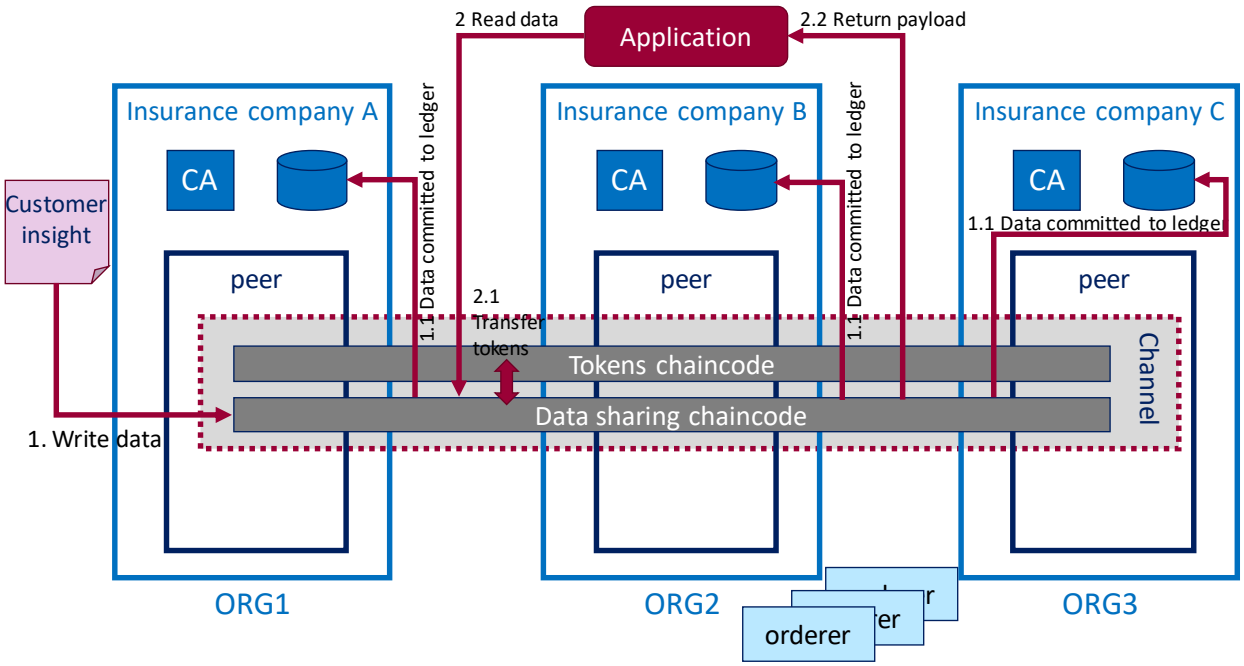

Figure 16 - Envisioned collaboration between T4.4 and T4.5

4.5 Summary

Section 4 describes potential future directions that will leverage the work on tokenization in the scope of T4.4. The aim is to collaborate with other partners and tasks in WP4 to elaborate more complex and realistic scenarios that can be applied to pilots in the project.

Future developments status will be included in next versions of this deliverable.

5 Conclusions

The purpose of the deliverable at hand titled D4.10 “Blockchain Tokenization and Smart Contracts - I” was to report the outcomes of the work performed within the context of T4.4 “Tokenization and Smart Contracts Finance and Insurance Services” in WP4 from M4 to M14 of the project.

Digital tokens can play an important role as accelerators of value exchange in any business ecosystem, particularly in financial and insurance business networks. However, Hyperledger Fabric which is the underlying BC technology selected in the INFINITECH project does not have support for tokens. Therefore, the aim of T4.4 is to extend Fabric with tokens capabilities. This document details the direction taken towards this aim.

At first, we give some background related to the tokens’ world, necessary for understanding the technical aspects of our proposed approach. We present some key terms and concepts and give some examples of potential use cases of tokens in the financial sector. Furthermore, we motivate the implementation of ERC 20 standard workflows as a chaincode in Fabric, which is the core technical work outcome so far. The implementation of the tokens functionality at the application level free us from any changes at the infrastructure level of Fabric on the one hand, and give us flexibility for interoperability with other applications, on the other hand.

Second, we detail the tokens implementation work carried out so far and presents the set of token related functions that have been developed and demonstrated during this first period of the project. We have already implemented the six functions of the ERC 20 standard plus an additional function for minting of tokens. The set of functions are described in detail following the convention introduced in D4.7 for technical descriptions of the use cases and the data flows using sequence diagrams that depict the interactions between the stakeholders and the involved components in the architecture.

Furthermore, we address four potential usages and extensions to the work carried out that will constitute the basis for our next activities in the task. Within these activities we foresee fruitful collaboration with the work done in T7.4 around scalable transaction graph analytics and with T4.5 around a marketplace for data, in which tokens could be used as digital currency to enable trade. Clearly, our goal is to apply the extensions developed in one of the pilots in the project.

It should be stressed out at this point, that the document in hand constitutes the first version of the blockchain tokenization and smart contracts deliverable. Next versions will include elaboration of the foreseen directions to pursue. However, we expect that as the project evolves, we might receive new requirements that might change the course of action. In addition, although the deliverable is of type “R” (only report), we provide a full demonstrator of the token workflows along with a self-explanatory recording of the work using an illustrative example.

As part of our development roadmap we also plan to showcase the INFINITECH Tokenization solution to potential end-users of the solution in the finance sector such as banks, FinTechs and other financial organization. Relevant showcases will be demonstrated as part of stakeholders’ (virtual) workshops, towards receiving feedback regarding the potential adoption and use of blockchain tokenization functionalities by end-users. Stakeholders’ feedback will be taken into account in developing subsequent versions of the tokenization prototype as part of deliverables D4.11 and D4.12.

6 Appendix A: Literature

[1] "Tokenization Market by Component, Application Area (Payment Security, Application Area, and Compliance Management), Tokenization Technique (API-based and Gateway-based), Deployment Mode, Organization Size, Vertical, and Region - Global Forecast to 2023", 2018. MarketsandMarkets Research Report, [Online]. Available at: https://www.marketsandmarkets.com/Market-Reports/tokenization-market-76652221.html?gclid=Cj0KCQjw2or8BRCNARIsAC_ppyY4t6bLXgBQG1pregsjn3hKtxG8goxXR6RgjRXxlP1H7MY2IXUSqGsaAjNAEALw_wcB [Accessed: 20-October-2020].

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

[3] F. Vogelsteller and V. Buterin, "ERC-20 token standard," 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20 [Accessed 1-October-2020].

[4] N. Gaur, L. Desrosiers, V. Ramakrishna, P. Novotny, SA. Baset, and A O'Dowd, Hands-On Blockchain with Hyperledger, 2018.

[5] J.Y. Lee, "A decentralized token economy: How blockchain and cryptocurrency can revolutionize business", Business Horizons (2019). [Online]. Available at: www.sciencedirect.com [Accessed: 10-October-2020].

[6] IBM Institute for Business Value, Moving to a token-driven economy Enabling the digitization of real-world assets, 2018. [Online] Available at: https://www.ibm.com/downloads/cas/YMRKPOJ8 [Accessed 15-October-2020].

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", 2008. [Online] Available at: https://bitcoin.org/bitcoin.pdf [Accessed 19-October-2020].

[8] M. di Angelo and G. Salzer, "Tokens, Types, and Standards: Identification and Utilization in Ethereum" IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), 2020.

[9] L. Oliveira; L. Zavolokina; I. Bauer; and G. Schwabe, "To Token or not to Token: Tools for Understanding Blockchain Tokens", Thirty Ninth International Conference on Information Systems, San Francisco 2018.

[10] L. Monso, "Asset tokenization: What is it and why does it matter?", 2019. [Online]. Available at: https://medium.com/@Metaco/asset-tokenization-what-is-it-and-why-does-it-matter-3f6892273dfe [Accessed: 20-October-2020].

[11] L. Monso, "Asset tokenization: Benefits and challenges ahead", 2019 [Online]. Available at: https://medium.com/@Metaco/asset-tokenization-benefits-and-challenges-ahead-475d737eba57 [Accessed: 20-October-2020].

[12] V. Buterin, "A next-generation smart contract and decentralized application platform, Ethereum Project White Paper", [Online] Available at: https://github.com/ethereum/wiki/wiki/White-Paper [Accessed: 20-October-2020].

[13] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger, Ethereum Project Yellow Paper", 2018. [Online] Available at: https://ethereum.github.io/yellowpaper/paper.pdf. [Accessed: 20-October-2020].

[14] M. Westerkamp , F. Victor, and A. Küpper, "Tracing manufacturing processes using blockchain-based token compositions", Digital Communications and Networks, 2019.

[15] W. Entriken, D. Shirley, E. Evans, and N. Sachs, "ERC-721 non-fungible token standard," 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721. [Accessed 1-October-2020].

[16] J. Dafflon, J. Baylina, and T. Shababi, "ERC-777 token standard," 2015. [Online]. Available: https://eips.ethereum. org/EIPS/eip-777 [Accessed 10-October-2020].

[17] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford, “ERC-1155 multi token standard,” 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1155 [Accessed 10-October-2020].