University of Piraeus

School of Information and Communication Technologies

Department of Digital Systems

Postgraduate Program in "Digital Systems Security"

Master Thesis Title:

"Blockchain technologies and smart contracts in the context of the Internet of Things"

Sotirios Stampernas

Supervisor: Sokratis Katsikas, Professor, University of Piraeus

Advisor: Dr. Pankaj Pandey, Research Scientist, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Gjovik, Norway

April 2018

Πανεπιστήμιο Πειραιώς

Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών

Τμήμα Ψηφιακών Συστημάτων

Π.Μ.Σ. «Ασφάλεια Ψηφιακών Συστημάτων»

Τίτλος Μεταπτυχιακής Εργασίας:

"Τεχνολογίες αλυσίδας συστοιχιών και έξυπνα συμβόλαια στο πλαίσιο του Διαδικτύου των Πραγμάτων"

Σωτήρης Σταμπέρνας

Επιβλέπων Καθηγητής: Σωκράτης Κάτσικας, Καθηγητής Πανεπιστημίου Πειραιώς

Σύμβουλος: Dr. Pankaj Pandey, Research Scientist, NTNU

Απρίλιος 2018

Abstract

Distributed ledger technology, is a set of technologies where a ledger is maintained by a number of peers without needing a single central authority. From this family of technologies, blockchain has recently become very popular. Blockchain is a distributed, transactional database which is shared across all the nodes of the network system , acting as a public ledger. Every node, usually most of the times has a full copy of the current blockchain , which contains every transaction that has ever been executed. Each block contains a hash of the previous block; the linking of those two together, constitutes the blockchain. This is the main technology underneath cryptocurrencies that started with Bitcoin. The recent focus on blockchain came as a result of the commercial success of Bitcoin and the consequent attention it caught by the researchers and industries. In this thesis we deal with the current status of the blockchain in addition to an application beyond cryptocurrencies, named smart contracts combined with IoT. Furthermore, we compare five blockchain platforms enabling smart contracts focusing on their general description, their main technological properties and their financial data. Additionally, we will perform a cost-benefit analysis and security assessment of smart contracts to highlight the related issues providing a set of possible solutions. Finally, we present a hands-on example to show the process and the flexibility in building smart contracts in an IoT environment.

Keywords: Blockchain technology, Smart contracts, Internet of Things, Distributed ledger technology, Bitcoin, Ethereum, Consensus mechanism, Security assessment, Blockchain platform evaluation

Περίληψη

Η τεχνολογία κατανεμημένου καθολικού είναι μια βάση δεδομένων όσον αφορά τις συναλλαγές που, αντί να αποθηκεύεται σε μια κεντρική τοποθεσία, κατανέμεται σε ένα δίκτυο πολλών υπολογιστών. Συνήθως, όλα τα μέλη του δικτύου μπορούν να διαβάζουν τις πληροφορίες και, ανάλογα με τις άδειες που τους έχουν δοθεί, να προσθέτουν στοιχεία. Ο γνωστότερος τύπος τεχνολογίας κατανεμημένου καθολικού είναι η αλυσίδα συστοιχιών (blockchain). Όπου οι συναλλαγές ομαδοποιούνται σχηματίζοντας συστοιχίες (blocks) οι οποίες συνδέονται μεταξύ τους με χρονολογική σειρά δημιουργώντας μια αλυσίδα (chain). Αυτή είναι και η τεχνολογία πίσω από τα κρυπτονομίσματα τα που ξεκίνησαν με την έλευση του Bitcoin. Η πρόσφατη επικέντρωση στην τεχνολογία της αλυσίδας συστοιχιών προήλθε από την εμπορική επιτυχία του Bitcoin και της προσοχής που έτυχε από τους ερευνητές και τις βιομηχανίες. Σε αυτή την διπλωματική εργασία ασχολούμαστε με την τρέχουσα κατάσταση της αλυσίδας συστοιχιών και επιπλέον με μια εφαρμογή πέραν των κρυπτονομισμάτων, που ονομάζεται έξυπνο συμβόλαιο. Επιπλέον, συγκρίνουμε πέντε πλατφόρμες αλυσίδας συστοιχιών που μπορούν να υποστηρίξουν έξυπνα συμβόλαια επικεντρώνοντας την ανάλυση σε μια γενική περιγραφή τους, στις κύριες τεχνολογικές ιδιότητές τους και στα χρηματοοικονομικά τους στοιχεία. Επιπρόσθετα, θα παρουσιάσουμε μια ανάλυση κόστους-οφέλους και μια αξιολόγηση ασφάλειας των έξυπνων συμβολαίων σημειώνοντας τα σχετικά θέματα παρέχοντας ένα σύνολο πιθανών λύσεων. Τέλος, θα δείξουμε μια πρακτική εφαρμογή και τη διαδικασία και την ευελιξία της διαδικασίας δημιουργίας ενός έξυπνου συμβολαίου σε ένα περιβάλλον με το διαδίκτυο των πραγμάτων (Internet of Things).

Λέξεις κλειδιά: Τεχνολογία αλυσίδας συστοιχιών, Έξυπνα συμβόλαια, Διαδίκτυο των Πραγμάτων, Τεχνολογία κατανεμημένου καθολικού, Μηχανισμοί Ομοφωνίας, Αξιολόγηση ασφάλειας, Αξιολόγηση Πλατφόρμων Τεχνολογιών αλυσίδας συστοιχιών

Contents

List of Figures

List of Tables

1 Introduction

Over the past few years, after introducing Bitcoin in 2008, several cryptocurrencies appeared in the world market. This was the opportunity for a variety of businesses to familiarize with their underlined technology named blockchain and its possible applications. Particularly, in the field of IoT there has been a lot of research and applications, emerging or applied such as smart contracts. In the next paragraphs, we are going to present the research questions, the methodology and the outline of this thesis followed by some important definitions.

1.1 Research Questions

In this master's thesis we are going to make an approach, in order to answer the following research questions (RQs):

- **RQ1 –What is the current status and the predictions concerning blockchain technology, smart contracts and IoT?** Technologies evolve, new models appear and more applications for smart contracts come up. However, businesses and clients are still questioning if it's worth switching to applications using blockchain.
- **RQ2 – What are the main similarities and differences among blockchain platforms that enable smart contracts?** Every year new platforms make their appearance, promising new features and a different technological approach. We will evaluate five of the most well-known and influential platforms enabling smart contracts, presenting their main features of the chosen platforms and making a cost-benefit analysis of smart contracts.
- **RQ3 – What is the level of security in smart contracts and what can we do to improve it?** The use of smart contracts is growing in time and their global market value is exponentially increasing. Projections for the future value of blockchain, smart contracts and IoT are promising. However, security is a major issue that has to be taken into serious consideration since these breaches are costly.

1.2 Research Methodology and Outline

The research methodology for this thesis consists of the following:

The evaluation of blockchain platforms supporting smart contracts is based on the current literature. This information is well documented in some platforms whereas in the majority of others, it is dispersed in different sources or it is insufficient due to their recent appearance. Their key features are placed in a matrix in order to be highlighted allowing thus, a comparison, which highlights both their similarities and differences. Additionally, we are going to provide an insight of a cost-benefit analysis for smart contracts and supporting technology, plus a security assessment of ethereum-based smart contracts, discussing the risks, the vulnerabilities and the related countermeasures.

Apart from the study of the literature, we will deploy a smart contract in an IoT environment describing the steps of the process.

An outline of the contents of this thesis is given below:

Chapter 2: In this chapter we provide a description of the blockchain technology.

Chapter 3: This chapter provides an overview of smart contracts.

Chapter 4: Forecast, market data and other related figures for blockchain, smart contracts and IoT.

Chapter 5: A study and a comparative evaluation of blockchain platforms supporting smart contracts.

Chapter 6: A brief description of security issues related to blockchain and smart contracts.

Chapter 7: An effort to make a security assessment in ethereum-based smart contracts.

Chapter 8: A set of smart contracts applications in IoT is described.

Chapter 9: A deployment of a smart contract system in an IoT environment.

Chapter 10: Answers to research questions and conclusion.

1.3 Important Definitions

In order to make the distinction among blockchain and related terms clearer, we are providing below some definitions and their relation.

Distributed Ledger Technology (DLT): It is a family of technologies that includes blockchain, where a ledger is maintained by a group of peers rather than a single central authority.

Blockchain: A type of distributed ledger, which enables records to be stored and

sorted into blocks.

Cryptocurrency: An electronic currency application using blockchain technology. This category is synonymous to Bitcoin, the first and foremost known cryptocurrency that uses this technology. Other applications exist too, but they are not fully functional or widely accepted by the public. Cryptocurrencies attracted most of the users' attention, due to their financial impact.

Miner: A peer in the blockchain network that confirms or verifies a transaction by solving a challenging cryptographic problem and adds the transaction or "blocks" the blockchain, thus updating the ledger.

The figure below shows the relationship between the basic technologies and their applications.

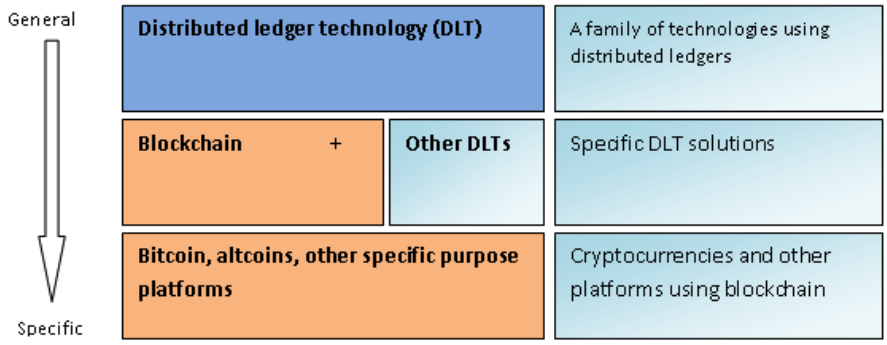

Figure 1: Relational Scheme

2 An Overview of Blockchain Technology

This chapter discusses the different types of blockchain technology and their connected technologies. The main difference among blockchain platforms, the consensus mechanism, is discusses at the end of this chapter.

2.1 About Blockchain

Blockchain technology is a distributed database, which maintains an immutable public ledger of all the transactions. Blockchain allows for the time stamped recording of all the transactions. Every node in the network is responsible for the maintenance and continuous verification of transactions. The blockchain technology involves creation of digital tokens for digital files, such as documents or transactions. These digital tokens can be considered as digital fingerprints of the files. These fingerprints are saved in groups called "block". The individual blocks are then linked in a chain of blocks and each subsequent block has a digital token from the previous block. Thus, it becomes impossible to modify the information in an old block in the chain without modifying the subsequent blocks. The main idea behind the blockchain technology is to register, confirm and transfer all kinds of contracts and properties without the need of any intermediary [1].

The ability of blockchain to secure the data and history of transactions led it to be called as "The Trust Machine" by the Economist [2]. The World Economic Forum conducted an expert survey in 2015 and reported that the majority (57%) of the respondents estimated that by the year 2025 10% of the world's Gross Domestic Product (GDP) will be registered in blockchain [3]. Furthermore, in Dec 2015, Goldman Sachs, an influential international investment bank stated that "....Silicon Valley and Wall Street are betting that the underlying technology; the blockchain can change..... Well everything" [4].

2.2 A Short History of Blockchain

In 1991, Stuart Haber and W. Scott Stornetta were the first ones to present their work on a cryptographically secured chain of blocks [5]. Later, Bayer, Haber and Stornetta

incorporated Merkle trees to the blockchain in 1992; this was to improve the efficiency so as to collect several documents into one block [6]. The concept of distributed blockchain was first introduced by an anonymous person or group known as Satoshi Nakamoto in 2008 by publishing a whitepaper title ”Bitcoin: A Peer-to-Peer Electronic Cash System” [7]. Blockchain (bitcoin) was born when Satoshi Nakamoto solved a complex Game Theory conundrum called Byzantine Generals Problem, which ensured that at a particular time, a block of asset could be transferred to only one other person, without the need for a third-party check.

In 2009, the concept of distributed blockchain was implemented and released as an open-source software as a core component of the digital currency bitcoin. The use of the blockchain for bitcoin made it the first digital currency to solve the double spending problem without requiring a trusted administrator [8]. The words block and chain were used separately in the original paper presented by Satoshi Nakamoto in October 2008 [7], and when the term moved into wider use it was originally *block chain* [8] before becoming a single word, *blockchain*, by 2016.

2.3 Byzantine Generals Problem

Blockchain technology answers the “Byzantine Generals Problem” or how do individual users secure their data from non-trusted actors. The Byzantine Generals problem is the computer-world’s practical take on a thought-experiment called the Two Generals Problem. The problem is illustrated by two or more generals that siege a city from opposite sides, trying to coordinate an attack. If General A sends a message that says “attack at noon tomorrow,” he has no idea whether or not General B will actually receive the message, and could potentially be marching toward death if he attacks without the other general. Upon receipt, General B has no idea if the message is authentic or has been sent from the enemy to draw him into a trap. Nevertheless, he will assume authenticity and send a response confirming the attack, but without knowing whether General A received his response, he may fear that the other general will hold off attacking, meaning that General B will be the one attacking alone at noon tomorrow and facing certain death. General A could, of course, send a message confirming receipt of General B’s acknowledgment, but will never actually know if it reached its destination, or even if the message was authentic in the first place. This puts him in the same spot General B was just in. This problem bounces

back and forth into perpetuity, with neither general ever able to be sure whether their message went through, let alone is authentic. [9]

To relate this to the blockchain we can describe it as follows: A person can store just about anything of value into a 'digital lock box'. The content inside of the box can only be opened and changed with a unique private key. The information inside of this box can then be shared on demand without the possibility of it being altered, changed, or replicated from its original form. [10]

2.4 Types of Blockchain

Buterin [11] categorized blockchains into three categories: Public Blockchain; Private Blockchain; and Hybrid Blockchain.

2.4.1 Public Blockchain

A fully open public ledger has no limitations with regards to reading and writing permissions. Anyone can connect to the network obtain access to information and has the possibility to add information. Anyone connected to the network has the right to participate in the consensus protocol, to verify the newly added blocks and ensure that it is not conflict with previous blocks in the chain. The consensus protocol is forced to be based on a cryptoeconomic mechanism, because of the open nature of the system and due to lack of trust between the nodes. A public ledger blockchain system operates without the requirement of trust between users; hence, considered to be fully decentralized.

Some of the state of the art open source public Blockchain protocols based on a Proof of Work (PoW) consensus algorithms are Bitcoin, Ethereum and Monero.

The main characteristics of public blockchain are as follows:

- Anyone can participate, without permission.
- Anyone can download the code and start running a public node on their local device, validating transactions in the network, and thus participating in the consensus process. This is the process for determining what blocks get added to the chain and what the current state is.
- Anyone in the world can send transactions through the network and expect to see them included in the blockchain, if they are valid.
- Anyone can read transactions on the public block explorer. Transactions are

transparent, but anonymous/pseudonymous.

2.4.2 Private Blockchain

A private blockchain has certain limitations on the reading and writing permissions and is more tightly controlled than a public ledger. The right to modify, add or read information is restricted and kept centralized to a group of participants, e.g. an organization. In a private blockchain system, a consensus protocol is usually not required because of the trusted nodes. Private ledgers have the ability to fast access information, make transactions cheaper, and possibility to control the level of privacy. Example applications include database management, auditing, etc. which are internal to a single company, and so public readability may in many cases not be necessary at all. In other cases public audit ability is desired. Private blockchains (such as Monax and Multichain) are a way of taking advantage of blockchain technology by setting up groups and participants who can verify transactions internally. This has the risk of security breaches just like in a centralized system but has advantages when it comes to scalability and state compliance of data privacy rules and other regulatory issues.

2.4.3 Hybrid (or Federated or Consortium) Blockchains

There is a hybrid blockchain system consisting of some features of a public and private ledger, called consortium ledger. In a consortium ledger, the consensus protocol is usually predetermined and managed by a predefined group of institutions [11]. A consortium blockchain system could e.g. have 20 institutions controlling one node, and every newly added block must be signed by at least 13 institutions to be considered valid. A hybrid blockchain system is considered to be partially decentralized. In a consortium ledger, the reading permissions could be open to the public or restricted to a group of participants. There is a hybrid solution to this as well, such that the parts of the information are public and other parts are not. Federated Blockchains (such as R3 (Banks), EWF (Energy), B3i (Insurance), Corda), operate under the leadership of a group without letting any person with access to the Internet to participate in the process of verifying transactions. Federated Blockchains are faster providing more transaction privacy.

Consortium blockchains are mostly used in the banking sector. The consensus process is controlled by a preselected set of nodes. The right to read the blockchain may be public, or restricted to the participants.

2.4.4 Adoption Stages of Blockchain

Based on Accenture's maturity model [11A] for the adoption of blockchain, by 2025 blockchain technology will reach the stage of maturity. This means that today we are in the early adoption of this technology and it will take years for its complete acceptance.

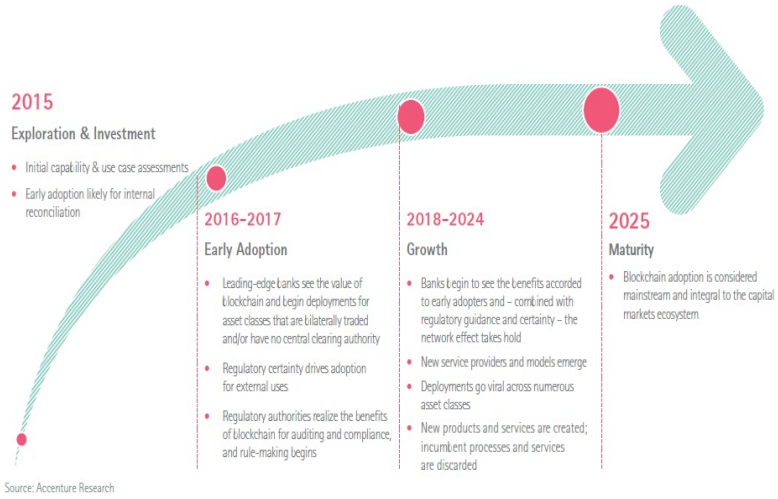

Figure 2: Adoption Stages of Blockchain

2.5 Main Technologies Connected to Blockchain

Bitcoin is considered to have created new functions by combining existing technologies. In order to operate a system like the one for electronic money, without any central authority, it is indispensable to put in place measures to prevent falsification of data and duplicate payments, as well as a mechanism to maintain the system against any attacks by malicious users. Major technologies connected to blockchain to make applications such as Bitcoin function in a proper manner (P2P, hash, public-key cryptography and digital signature, Proof of Work, Proof of Stake) are outlined below.

2.5.1 Peer-to-Peer (P2P) Network

In a client-server network, a server takes charge of preservation and provision of data while a client requests the server for data and gains access to them, and their roles are thus fixed. In contrast, in P2P networks, all participating nodes hold data respectively and create an autonomous network wherein data are requested and provided among these nodes. In such a network, the role (server or client) of respective nodes is not fixed. It is necessary for P2P networks to consider search methods and data

transmission methods. Search methods are the means to manage locations of nodes as well as data and data transmission methods are means of transmitting data between nodes (direct or relayed).

2.5.2 Hash

Blockchain technology relies extensively on hashes and hash functions. A hash function is a mathematical algorithm that takes an input and transforms it into an output. A hash is the result of a transformation of the original information that serves as input. The main hash function characteristic is collision resistance meaning that it is extremely difficulty to recreate the input data from its output (hash value) alone. This mechanism is characterized by the fact that the same hash value is obtained from the same data and even a slight difference in the original data will result in a completely different hash value. Taking advantage of such characteristics, this mechanism is used for the detection of falsification of data.

2.5.3 Public Key Cryptography and Digital Signature

Public-key cryptography is a cryptographic method using different keys for encryption and decryption. The problem of handing over keys was solved by dividing the key into one for private use (private key) and one available for anyone (public key). In the case of symmetric-key cryptography using the same key for encryption and decryption, requires various safety measures for delivering the key only to the relevant counterparty. In contrast, public-key cryptography enables safe delivery and receipt of files only if a receiver prepares a pair of a private key and a public key and delivers the public key to the sender in advance. Safety can be maintained even though other persons use the public key as long as the receiver properly manages his/her private key. Blockchain uses an asymmetric cryptography mechanism to validate the authentication of transactions.

A digital signature refers to a mechanism to prove the authenticity of the data sent via a network using a pair of keys like in public-key cryptography. A digital signature, sent to the receiver together with the relative file, is made by encrypting the hash value of the file to be sent with the sender's private key. The receiver uses the same hash function as the sender to create the hash value of the file by himself and crosschecks the created hash value with the hash value obtained through decrypting the sender's digital signature with the sender's public key. This results to the

confirmation that the sender's digital signature is authentic. In blockchain the typical digital signature is involved with two phases: the signing phase and the verification phase as described above. Among other algorithms, the elliptic curve digital signature algorithm (ECDSA) is used in blockchain.

2.5.4 Consensus Mechanism

A major difficulty with blockchain technology with public ledgers is to make sure that the consensus protocol is reached by the entire peer-to-peer network participants [12]. A consensus protocol is used to make sure that the participants in the network follow the network's rules and to make sure the transactions are validated in the right order. It is also used to make sure that the information within a block is correct, that the nodes (miners) get a fair compensation and to avoid issues like the double spending problem.

Algorithms for achieving consensus with arbitrary faults require a type of voting among a known set of peers. Two main approaches that exist are the "Nakamoto consensus" and the Byzantine Fault Tolerance (BFT). The first approach elects the leader, through some form of "lottery", who then proposes a block that can be added to a chain of previously committed blocks. The second approach is based on Byzantine Fault Tolerance (BFT) algorithms and uses multiple rounds of explicit votes to achieve the necessary consensus. In this part of the section, we present some of the most popular consensus mechanisms with Proof of Work and Proof of Stake being the most popular and tested mainly through cryptocurrencies.

2.5.4.1 Proof of Work (PoW)

Proof of Work (PoW) generally refers to a mechanism to confirm a person's innocence by giving him to do a certain work, which is simple but troublesome and can be easily verified that he did it. Backlund [13] argues that one way of ensuring authenticity is to let each user within the network get one vote and let all users vote which transaction should be included in the next block. The number of votes decides which set of transactions should be included. This kind of consensus-process is vulnerable to Sybil attacks, where one user could create multiple accounts and get a higher influence within the network.

Nakamoto, the creator of Bitcoin, solved this issue of influence by adding a cost to the vote. Each user's amount of influence is based on the computing power of that user.

The more computing power, the higher the needed energy and the higher the hardware costs. This is the concept of proof-of-work consensus protocol. In the case of bitcoins (which use a proof-of-work consensus protocol) the network collects all the transactions made during a set period into a block. The nodes task is to confirm those transactions and write them into a blockchain hashing the information as well, to protect it from intruders. The nodes get economic incentives to keep mining and hashing, the more blocks created, the more bitcoins received.

Carlsson and Huang [14] argue that when a node creates a block, it gets distributed to neighbouring nodes. The neighboring nodes independently verify that the information is correct within the block and that the rules have been followed. In a bitcoin network, it is recommended to wait at least six blocks to make sure that the transaction is final. Nodes compete against each other to be the first one to produce a block and a couple of nodes could be working on the same transaction simultaneously, a blockchain fork is created. The block that is created first, with the longest blockchain behind it, wins and that node gets rewarded. Despite this being hardware and energy intensive and thus raising mining costs, proof-of-work has been empirically proven to be safe and robust [13].

Buterin [15] argues that there are some downturns with a proof-of-work consensus protocol, e.g. the risk of a 51% attack and there are high-energy costs of producing one block. Courtious [16] further argues that the proof-of-work protocol is heading towards self-destruction. The mining community is getting smaller and more specialized, where big companies with great resources could outwork the individual miner. This specialization of mining is making the system more centralized to a few big companies and the risk of a 51% attack increases.

2.5.4.2 Proof of Stake (PoS)

To reduce the risk of a 51% attack and to reduce energy consumption, a new consensus protocol was introduced within the blockchain community, called proof-of-stake. Instead of proving that a node solved a computational hard task, like one does in the proof-of-work protocol, the node could instead proof it has a certain amount of coins [17]. In the case of proof-of-stake it takes coins to create a new block, not computational power and the node with the most coins, gets the most influence [18]. The bitcoin community and Manning [19] argue that a proof-of-stake protocol will reduce the risk of a 51% attack. He argues that the likelihood of a 51% attack is

reduced due to the coins invested by the miner within the network. If someone has 51% of the computational power within a proof-of-stake protocol, one needs to own 51% of the total bitcoins as well. According to game theory, it is thus in the interest of the majority owner to have a stable and secure network, and will therefore not attack it. If there is an attack, it will only destabilize the digital currency and decrease its value. One issue with the proof-of-stake protocol is the issue of forking. When one node starts mining on a transaction, another node could start mining on it simultaneously, without the cost of computational power.

Backlund [13] argues that the risk of villainous nodes that fork the blockchain is increased compared with a proof-of-work protocol. This increases the risk of double-spending attacks and greedy behavior, where the nodes start to mine on all forks to not miss out on block rewards. This issue can be solved by using check-point blocks, where blocks before a check-point cannot be revised and the issue of double spending attacks are solved. There still remains a risk of a 51% attack in a proof-of-stake protocol and Houy [20] argues that the points made by the bitcoin community and Manning is not valid. He argues that it will cost nothing for a miner to buy 50% of a proof-of-work cryptocurrency monetary base and thus take over the platform.

2.5.4.3 Delegated Proof of Stake (DPoS)

DPOS (Delegated proof-of-stake). Similar to POS, miners get their priority to generate the blocks according to their stake. The major difference between POS and DPOS is that POS is a direct democratic while DPOS is representative democratic. Stakeholders elect their delegates to generate and validate a block. With significantly fewer nodes to validate the block, the block could be confirmed quickly, making the transactions confirmed quickly. Meanwhile, the parameters of the network such as block size and block intervals could be tuned. Additionally, users do not to need worry about the dishonest delegates because the delegates could be voted out easily. DPOS has already been implemented, and is the backbone of Bitshares.

2.5.4.4 Practical Byzantine Fault Tolerance (PBFT)

PBFT (Practical byzantine fault tolerance) is a replication algorithm to tolerate byzantine faults (Miguel and Barbara, 1999). Hyperledger Fabric (Hyperledger, 2015) utilizes the PBFT as its consensus algorithm since PBFT could handle up to 1/3 malicious byzantine replicas. A new block is determined in a round. In each round, a

primary would be selected according to some rules. And it is responsible for ordering the transaction. The whole process could be divided into three phase: pre-prepared, prepared and commit. In each phase, a node would enter next phase if it has received votes from over 2/3 of all nodes. So PBFT requires that every node is known to the network. Like PBFT, Stellar Consensus Protocol (SCP) (Mazieres, 2015) is also a Byzantine agreement protocol. There is no hashing procedure in PBFT. In PBFT, each node has to query other nodes while SCP gives participants the right to choose which set of other participants to believe. Based on PBFT, Antshares (antshares, 2016) has implemented their dBFT (delegated byzantine fault tolerance). In dBFT, some professional nodes are voted to record the transactions instead of all nodes. [21]

2.5.4.5 Proof of Elapsed Time

The Proof of Elapsed Time (PoET) Consensus was originally released to Hyperledger utilizing an abstract TEE (trusted execution environment). In terms of functionality, PoET stochastically elects individual peers to execute requests at a given target rate. Individual peers sample an exponentially distributed random variable and wait for an amount of time dictated by the sample. The peer with the smallest sample wins the election. Cheating is prevented through the use of a trusted execution environment, identity verification and blacklisting based on asymmetric key cryptography, and an additional set of election policies. [21A]

2.5.4.6 Ripple Protocol

The Ripple Transaction Protocol *(RTXP)*, was issued in 2012, and its purpose is to facilitate financial transactions by defining a set of rules that allow users to make online transactions with any currency, cryptocurrency or any other means. Ripple Schwartz et al., 2014) is a consensus algorithm that utilizes trusted sub-networks within a larger network. In the network, nodes are divided into two types: server for participating consensus process and client for transferring funds. [21]

2.5.4.7 Tendermint

Tendermint (Kwon, 2014) is a byzantine consensus algorithm where a block is determined in a round. All nodes need to be known and among them a proposer is selected to broadcast an unconfirmed block. The process is divided into three steps:

1) Prevote: Validators choose whether to broadcast a prevote for the proposed block.
2) Precommit: If the node has received more than 2/3 of prevotes on the proposed block, it broadcasts a precommit for that block. If the node has received over 2/3 of precommits, it enters the commit step.
3) Commit: The node validates the block and broadcasts a commit for that block. If the node has received 2/3 of the commits, it accepts the block.

The process is quite similar to PBFT, but Tendermint nodes have to lock their coins to become validators. [21]

The following table is a comparison of properties among the different consensus algorithms:

Property	**PoW**	**PoS**	**PBFT**	**DPoS**	**PoET**	**Ripple**	**Tendermint**
Node identity management	open	open	permissioned	open	-	open	permissioned
Tolerated power of the adversary	<25,0% computing power	<51,0% stake	<33,3% fault replicas	<51,0% validators	-	<20,0% faulty nodes in UNL	<33,3% byzantine voting power
Energy saving	No	Partial	Yes	Partial	Yes	Yes	Yes
Known apps or platforms	Bitcoin	Ethereum (next version)	Hyperledger Fabric	Bitshares	Sawtooth Lake	Ripple	Tendermint

Table 1: Consensus algorithms

2.6 How Blockchain Works

The next figures will provide a high-level explanation on how blockchain works. The blockchain as a meaning, involves blocks consisting of several data (i.e. transactions) including some other important pieces of information to make the chain work with its special features. Each block points to the immediate previous block via a reference that is an essential hash value of the previous block called parent block. It is worth noting that uncle block hashes (children of the block's ancestors) would also be stored in ethereum blockchain (Buterin, 2014). The first block which has no parent block is called the genesis block of a blockchain,

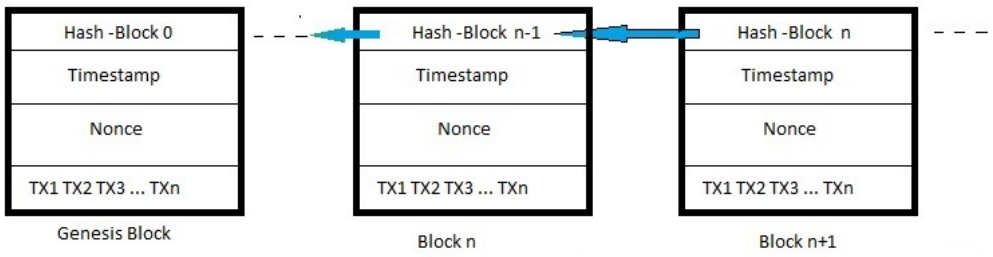

Figure 3: Block Structure

A block consists of the block header and the block body where the block header includes the following information:

- Block version: indicates which set of block validation rules to follow.
- Parent block hash: a 256-bit hash value that points to the previous block.
- Merkle tree root hash: the hash value of all the transactions in the block.
- Timestamp: current timestamp as seconds since 1970-01-01T00:00 UTC.
- Nonce: a 4-byte field, which usually starts with 0 and increases in every hash calculation.
- nBits: current hashing target in compact format.

The block body is consisted of a transaction counter and transactions. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. [21]

Header	**Block version**	0000000000010
	Parent block hash	fffff00015421256fffffffffff00000000
	Merkle tree root	dd008121256ddddddddffff11111111
	Timestamp	12e5d1985y
	Nonce	0efdef12
	Nbits	30x30x301845
Body	**Transaction counter**	TX1 TX2 TX3 TX4….. TXn

Table 2: Block Content

3 An Overview of Smart Contracts

3.1 About Smart Contracts

An American cryptographer Nick Szabo is deemed to be the person who conceptualized smart contracts in 1994. He often mentioned about an example of a rented car with a smart contract such that the control to the car is returned back to the car owner when a car renter forgives the payments. Smart Contracts can be defined as autonomous computer programs (self-executing codes) that, once started, execute automatically and in a mandatory manner the underlying conditions, such as the facilitation, verification or enforcement of the negotiation or performance of a contract, executing a payment transaction, and so on.

The main benefits of deploying Smart Contracts over a blockchain are that the blockchain guarantees that the contract terms cannot be modified. Blockchain makes it impossible to tamper or hack the contract terms. Thus, smart contracts deployed over a blockchain are expected to bring reduction in costs of verification, execution, arbitration and fraud prevention. Furthermore, smart contracts can be useful in overcoming the moral hazard problem.

3.2 Definition and Characteristics

A smart contract is a digitally signed, computable agreement between two or more parties. A virtual third party, a software agent, can execute and enforce (at least some of) the terms of such agreements. In the context of the blockchain, where it truly takes it sense, a smart-contract is an event-driven program, with state, that runs on a replicated, shared ledger and which can take custody over assets on that ledger.

Smart contracts on the blockchain, created by computer programmers, are entirely digital and written using programming code languages. This code defines the rules and consequences in the same way that a traditional legal document would, stating the obligations, benefits and penalties, which may be due to either party in various different circumstances. The big difference is that this code is automatically executed by a distributed ledger system, in a non-repudiable and unbreakable way.

Smart Contract code has some unique characteristics:

- **Deterministic**: Since a smart contract code is executed on multiple distributed

nodes simultaneously, it needs to be deterministic i.e. given an input; all nodes should produce the same output. That implies the smart contract code should not have any randomness; it should be independent of time (within a small time window because the code might get executed a slightly different time in each of the nodes); and it should be possible to execute the code multiple times.

- **Immutable**: Smart contract code is immutable. This means that once deployed, it cannot be changed. This of course is beneficial from the trust perspective but it also raises some challenges (e.g. how to fix a code bug) and implies that smart contract code requires additional due diligence/governance.
- **Verifiable**: Once deployed, smart contract code gets a unique address. Before using the smart contract, interested parties can and should view or verify the code.

3.3 Main Components

A smart contract can be broken down into two separate components:

- **Smart Contract Code:** The code that is stored verified and executed on a blockchain.
- **Smart Legal Contracts:** The use of the smart contract code that can be used as a complement, or substitute, for legal contracts.

Following is a short description of how smart contracts work divided into three parts, the input, the process and the output [22].

- **Coding** (what goes into a smart contract): Because smart contracts work like computer programs, it is very important that they do exactly what the parties want them to do. This is achieved by inputting the proper logic when writing a smart contract. The code behaves in predefined ways and doesn't have the linguistic nuances of human languages, thus, it has now automated the "if this happens then do that" part of traditional contracts.
- **Distributed Ledgers** (how the smart contract is sent out): The code is encrypted and sent out to other computers via a distributed network of nodes running a distributed ledger.
- **Execution** (how it is processed): Once the computers in the network of distributed ledgers receive the code, they each come to the same agreement or consensus on the results of the code execution. The network would then update the distributed ledgers to record the execution of the contract, and then monitor for compliance

with the terms of the smart contract. In this type of system, single party manipulation is averted because control over the execution of the smart contract is no longer possible since that execution is not in the hands of a single party.

3.4 Oracles

Oracles are trusted entities which sign claims about the state of the world. Since the verification of signatures can be deterministic, it allows deterministic smart contracts to react to the (non-deterministic) outside world. Oracles are required to connect smart contracts to critical data feeds, any web API or various accepted payment methods. As mentioned previously, Smart Contracts are executed by examining all the conditions of execution, which have been defined in advance in the contract code, and the problem arises from the validation of these conditions of execution.

Two scenarios are then possible:

- The execution conditions of the contract are linked to other entries in the blockchain or are simple time markers. In this case, checking these execution conditions is very easy: the contract is programmed to verify that these entries exist or that the execution time is passed, and it executes when this is the case.
- The conditions of execution of the contract are outside the blockchain. In this case, the execution of the contract requires the use of a trusted third party, an oracle.

An oracle is instructed to enter the blockchain information reliably so that the contract can run properly and can be constituted in several ways [22]:

- Prior designation of a trusted third party known to both parties.
- Reference to a database considered trustworthy.(i.e. in the case of a sports betting, possibility of referring to the result recorded on the site of a sports newspaper).
- Using a decentralized oracle service. It is an existing service on the blockchain involving many participants. Each participant votes for the result he considers to be accurate and it is the consensus among the participants that determines the final result sent to the contract. Decentralized oracle projects already exist, notably the Oraclize project.

The Oracle stands in between of the External world data or API and the Smart Contract.

3.5 Taxonomy of Smart Contracts

We will categorize smart contracts by application domain as far as concerning Bitcoin and Ethereum contracts, for the first ones based on the research in their web pages and the related discussion forums and on manual inspection of the Solidity source code for the Ethereum contracts as it is described in the related paper [23]. The result of this research is the distribution of smarts contracts into five categories as described below:

i. **Financial**

The main feature of this type of contracts is to manage certain amount of money, or verify the ownership of goods. Other contracts are used in crowdfunding money gathering in order to fund specific projects. High-yield investment programs are contracts witch include a high risk collecting money from users promising them a high interest rate if new investors will join the scheme. Some contracts provide insurance on setbacks, which are digitally provable, and others deal with advertisements.

ii. **Notary**

In this category contracts exploit the immutability of the blockchain to store data, and in some cases to certify their ownership and provenance. Some contracts allow users to write the hash of a document on the blockchain, so that they can prove document existence and integrity. Others allow declaring copyrights on digital media such as photos and music and some allow users to write down on the blockchain messages that everyone can read. In this category there are contracts that associate users to addresses, in order to certify their identity.

iii. **Game**

Contracts in this category include games of chance and games of skill that users want to participate.

iv. **Wallet**

These contracts handle keys, manage money complete transactions and deploy contracts, making the interaction with the blockchain simple. Wallets can be managed by one or many owners and the latter is possible using multiple authorizations.

v. **Library**

These contracts are implemented for common and general-purpose operations, to be used by other contracts.

Next, we present a table including the quantitative results of the research in smart contracts giving information for the number of contracts (Bitcoin and Ethereum) and the transactions in each category concerning a sample of 834 smart contracts [23]. The table consists of four columns including the number of detected contracts (third column), and the total number of transactions (fourth column). Overall, we have 1.673.271 transactions. Even though, Bitcoin contracts are fewer than the ones in Ethereum, they have a larger amount of transactions:

Category	Platform	Number of Detected Contracts	Total Number of Transactions
Financial	Bitcoin	6	470.391
	Ethereum	373	624.046
Notary	Bitcoin	17	443.269
	Ethereum	79	35.253
Game	Bitcoin	0	0
	Ethereum	158	58.257
Wallet	Bitcoin	0	0
	Ethereum	17	1.342
Library	Bitcoin	0	0
	Ethereum	29	37.034
Unclassified	Bitcoin	0	0
	Ethereum	155	3.679
Total	**Bitcoin**	**23**	**913.66**
	Ethereum	**811**	**759.611**
Overall	**Overall**	**834**	**1.673.271**

Table 3: Smart contracts by category

3.6 Benefits of Smart Contracts

For a wide range of potential applications, blockchain-based smart contracts offer a number of benefits:

- **Speed and real-time updates**: because smart contracts use software code to automate tasks that are otherwise typically accomplished through manual means, they can increase the speed of a wide variety of business processes.
- **Accuracy:** automated transactions are not only faster but less prone to manual

error.

- **Lower execution risk:** The decentralized process of execution virtually eliminates the risk of manipulation, nonperformance, or errors, since execution is managed automatically by the network rather than an individual party.
- **Fewer intermediaries:** smart contracts can reduce or eliminate reliance on third-party intermediaries that provide "trust" services such as escrow between counter parties.
- **Lower cost:** new processes enabled by smart contracts require less human intervention and fewer intermediaries and will therefore reduce costs.
- **New business or operational models:** because smart contracts provide a low-cost way of ensuring that the transactions are reliably performed as agreed upon, they will enable new kinds of businesses, from peer-to-peer renewable energy trading to automated access to vehicles and storage units.

3.7 Challenges

Smart contract technology is still in its early stages in both technology and business developments surrounding smart contracts. On the technology side, certain advances will help broaden the applications and adoption of smart contracts. Current challenges in the blockchain are the following [22]:

- **Scalability:** Smart contract platforms are still considered unproven in terms of scalability. The community is aware of this problem and is thinking of several approaches. It remains to be seen whether or not any of the approaches can preserve the benefits of a public blockchain.
- **Access to real world information:** As discussed above, because smart contracts can reference only information on the blockchain, oracles that can push information to the blockchain will be needed. While some initiatives are promising, approaches for creating oracles are still emerging.
- **Privacy:** The code within smart contracts is visible to all parties within the network, which may not be acceptable for some applications. For instance, some retailers may not want their deals with their suppliers to be public.
- **Latency and performance:** Blockchains suffer from high latency, given that time passes for each verified block of transactions to be added to the ledger. For Ethereum blockchain this occurs approximately every 17 seconds too far from the

milliseconds to which we are accustomed while using non-blockchain databases.

- **Permissioning:** While excitement for smart contracts is growing in the realm of both permissionless and permissioned blockchains, the latter is likely to see faster adoption in industry, given that complexities around trust, privacy, and scalability are more easily resolved within a consortium of known parties.
- **Limits of application:** There are often good reasons for providing options while writing contracts. In many of them, clauses are written into things on purpose to create a channel for arbitration. In addition some business may simply not be modeled in a way that would enable it to benefit from Smart Contracts or other blockchains.
- **Governance:** If blockchains are to be sustainable in the long run, serious consideration of appropriate governance mechanisms is needed. The distribution of mining power and crypto-currency holdings combined with pseudonymity of account holders and a strong incentive to game the system makes it prone to deception, unaccountability and fraud.

3.8 Cost-Benefit Analysis of Smart Contracts

3.8.1 Cost Analysis

The attempted cost analysis includes costs for smart contract and blockchain adoption costs too. Because of the uniqueness and the dependence of these costs to a variety of factors, we will make an effort to describe them in most cases and where possible, we will provide a quantitative explanation.

3.8.1.1 Smart Contract Costs

From a general point of view, smart contracts aim to reduce the cost of trust. This fact however, does not mean that smart contracts come for free even though anyone can write one. The main costs related to smart contracts can be divided into the following categories:

- **The cost of writing a contract.**

This cost depends on the developer and any other person's expertise needed to complete the contract or deploy new ones.

- **The cost of the purchase of the contract**

The cost of purchasing a contract depends on the seller's policy but the cost of writing one includes among others:

- The complexity of the code of smart contract.
- The control needed for the security of a contract
- The cost to link to external resources such as Oracles.

- **The cost of the transaction.**

In paper [23A] there is a detailed analysis of the costs related to the transaction of a simple ethereum smart contract. Breaking down the costs of each stage of the contract can be done in practice too, by deploying an instance of a simple escrow contract [24] onto a testnet, and the gas spent at each stage can be viewed in Etherscan. We will try to simplify these costs and categorize the most important ones between fixed and variable costs.

The costs of the different steps of a transaction in a simple escrow contract are presented below:

Action	Description	Fixed Costs	Variable Costs	Additional Costs
First transaction	contract initialization and transaction data	21000 gas baseline transaction fee	-	-
Second transaction	code execution - the sender calls confirm (for each call)	-	200 gas/byte	-
Third transaction	code execution - the arbitrator calls confirm (for each call)	-	200 gas/byte + a fee solely for arbitration rather than all of the book keeping	-
Gas Transfer	Transferring gas to recipient	-	-	25000 gas in case the recipient has an empty account. In case of a non-empty account, the gas costs would be smaller.

Table 4: Cost Analysis of a Smart Contract Transaction

3.8.1.2 Blockchain Adoption Costs

Apart from the costs related to the contract itself and the transaction process, there are also costs concerning the adoption of the blockchain in an organization. These involve:

- **Hardware cost**

  These costs are related to the extra hardware that is possibly needed on top of the existing one in the company.
- **Software cost**

  These costs are related to the extra software that is possibly needed on top of the existing one in the company and the cost of the contract itself.
- **System implementation cost**

  This category includes the modification costs and the costs of deployment of the blockchain in order to put on top of it, the smart contract. Security parameters as well as possibly software and hardware, need to be added here.
- **Operational cost**

  The operational cost deals with the cost of the resources dedicated to the smart contract process.
- **Maintenance cost**

  The maintenance cost is the cost needed to maintain the whole blockchain system.

Critical factors to determine these costs are:

- The size of the organization and
- The scope of smart contracts in relation to the organization's mission.

Because of the complexity of the costs involved in smart contracts, before adopting it, a thorough total-cost-of-ownership analysis of a blockchain platform for smart contracts is important to be done in order to identify the related costs.

3.8.2 Benefit Analysis

The benefit analysis for applications varies from business to business. That is because until now, there are no extensive business cases in order to conduct an actual data research and as a result, we cannot provide a detailed benefit analysis. Our benefit analysis points out the following:

- **The mediator**

As smart contracts eliminate mediators, the benefit is considered to be equal to the mediation cost of the real world mediator charged in different types of businesses. As far as this approach is concerned, we need to point out that the mediator still exists in smart contracts with a smaller fee though and it intends to compensate the mediation itself and not other related services that might be offered in the real world.

- **The transaction itself**

Reduction of time and fewer mistakes during the participants is a key benefit.

We will provide a brief analysis of the potential economic benefits, as explained in the report [24A], from the use of smart contracts in the financial sector. These benefits for both clients and businesses, are illustrated in the next use cases. We need to point out that more and more real world cases arise and there will be a more accurate benefit analysis.

- **Investment banking:** In trading and settlement of syndicated loans, corporate clients could benefit from shorter settlement cycles. Instead of the current 20 days or more, smart contracts bring this down to 6 to 10 days. This could lead to an additional 5% to 6% growth in demand in the future, leading to additional income of between US$2 billion and $7 billion annually. Investment banks in the US and Europe can see lower operational costs too.
- **Retail banking:** There will be a significant benefit for the mortgage loan industry by adopting smart contracts. Consumers could potentially expect savings of US$480 to US$960 per loan and banks would be able to cut costs in the range of US$3 billion to $11 billion annually by lowering processing costs in the origination process in the US and European markets.
- **Insurance:** Usage of smart contracts in the personal motor insurance industry alone, could result in US$21 billion annual cost savings globally through automation and reduced processing overheads in claims handling. Consumers could also expect lower premiums as insurers potentially pass on a portion of their annual savings to them.

These potential cost savings are shown in the figure below taken from CapGemini's report [24A] .

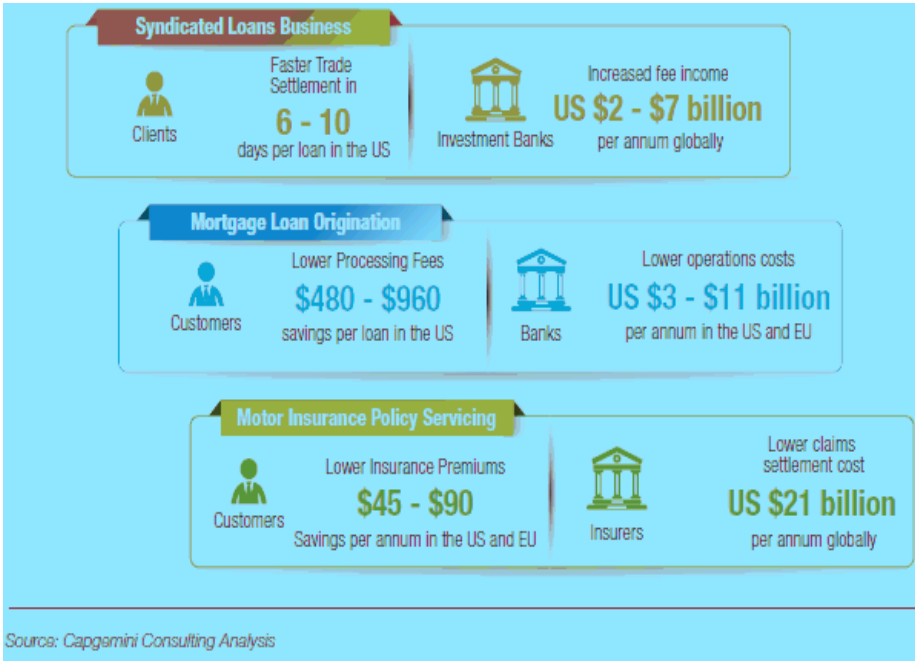

Figure 4: Smart Contracts Potential Benefits

3.8.3 Outcome

The results from the above analysis indicate that:

- A careful financial assessment of the proposed blockchain system should take place.
- The cost of the transactions can be reduced adopting smart contracts.
- There are costs related to the adoption of smart contracts and blockchain that should be taken into consideration by organizations.
- Not every business is suitable for smart contracts in terms of financial benefits.

4 Market Data and Forecast

This section deals with the current status and prospects that the market has about blockchain technology, smart contracts and IoT. We will focus on market data, meaning market size, regional spread and projections concerning the future financial aspects of these technologies.

4.1 Blockchain technology data

Businesswire [28] and MarketsandMarkets [29] conducted the two main researches published in 2017 that we used to gather data for the future value and market evaluation of blockchain technology. Both researches show similar high growth rates and similar increase in terms of usage in various sectors and regions.

4.1.1 Market size

The Businesswire research [28] indicates that the global blockchain market is projected to witness a significant CAGR of 71.46 % during the forecast period to reach a total market size of US$4.401 billion by 2022, increasing from US$297 million in 2017. It is expected that the usage of blockchain technology is going to increase owing to investments in blockchain technology start-ups, ties between financial organization and blockchain technology providers.

According to the market research [29] conducted by MarketsandMarkets, blockchain market will worth 7,683.7 Million USD by 2022. The blockchain market size is expected to grow from USD 411.5 Million in 2017 to USD 7,683.7 Million by 2022, at a Compound Annual Growth Rate (CAGR) of 79.6%.

4.1.2 Regional spread

The global market is segmented in regions which include North America, Europe, Asia Pacific (APAC), Latin America, Middle East and Africa (MEA). North America is home for the majority of industries with a large operation base, and has witnessed a prominent implementation of smart contracts, documentation, and payment applications in most of its industries especially BFSI, healthcare and life sciences.

North America is dominating the global blockchain market and contributes to the leading shares in terms of revenue on account of high acceptance of blockchain

technology due to its transparency and immutability. Europe has also significant shares in the global blockchain market and is anticipated to register the healthy growth. Asia Pacific is the most lucrative market due to the growing demand for the blockchain technology from various industries such as banking, finance, insurance, media, entertainment, retail and e-commerce sectors. South America, Middle East and Africa, are at an early stage in the global blockchain market due to low awareness and adaptability in the regions and are anticipated to register decent market growth. [29]

Tractica, a market research firm conducting a research in 2016 expects the worldwide market for enterprise blockchain applications to reach $19.9 billion by 2025 from $2.5 billion in 2016. North America will primarily drive demand during the forecast period, followed by Europe. [30]

According to a report from Tractica, the annual revenue for enterprise applications of blockchain will increase from $2,5Bn worldwide in 2016 to $19,9Bn by 2025. The firm's predictions are described in the following figure:

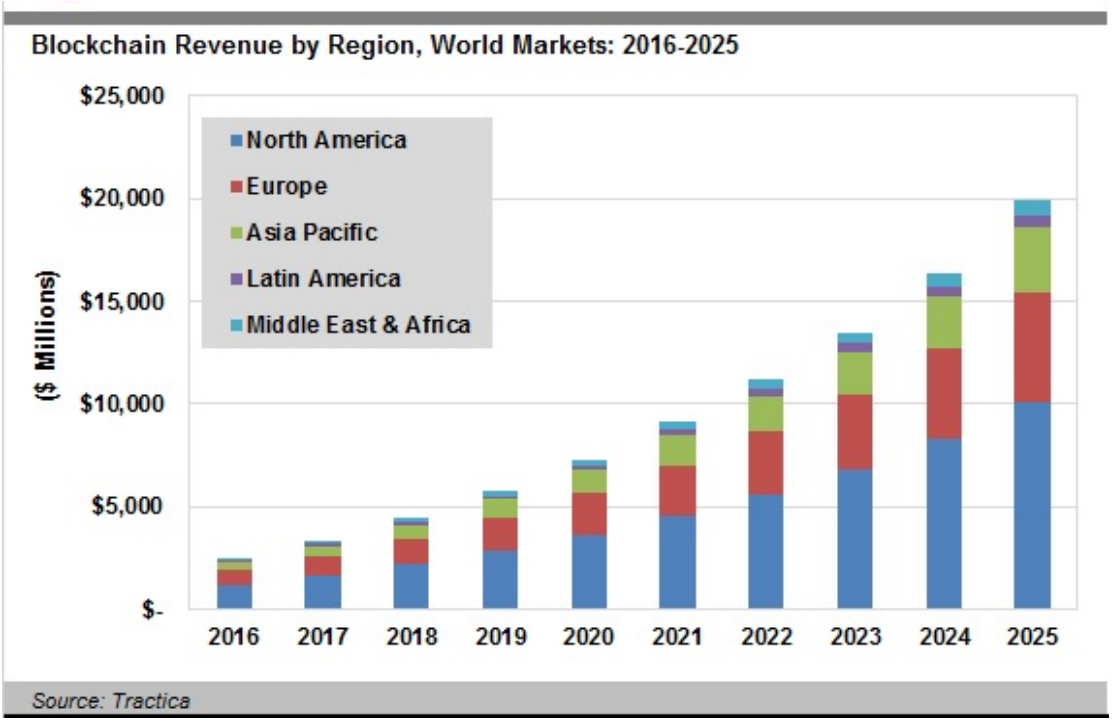

Figure 5: Blockchain Revenue by Region

4.1.3 Development drivers

Increasing penetration rate in multiple industries including transportation and logistics, retail and eCommerce, media and entertainment, real estate and IT and Telecommunication verticals, are expected to drive the market growth. Moreover, financial institutions in the APAC region are heavily investing in designing a

permissioned blockchain network to streamline their internal operations and minimize costs. The integration of the technology in these sectors is changing the way businesses are conducted across varied industry verticals.

The major leading factor of the blockchain use is the transparency of a transaction along with the ability to be incorruptible, which has resulted in increased acceptance among the wider audience. The adoption of DLT among the various applications such as payments, smart contracts, exchanges, digital identities, and documentation has also supplemented its growth.[28]

Key factors including reduced total cost of ownership, faster transactions, simplified business process with transparency and immutability and rising cryptocurrencies market cap and ICO, are expected to drive the overall growth of the market, too.[29]

Blockchain technology companies could experience a revenue pool of $6 billion by 2020 and $20 billion by 2030. These figures are based on the impact of digital ledger technology on payments: (1) Business Cross Border, (2) Remittance, as well as impact on (3) Capital Markets and (4) Title Insurance. Findings show that Cross Border B2B will make up most of the Blockchain revenue pool at $3.5 billion (56%) in 2020 and $12.2 billion (60%) in 2030. We also expect the increasing adoption of digital currency to put downward pressure on remittance payments shifting 20% of revenue to Blockchain companies. We project this to account for $1.5 billion (24%) of Blockchain revenue in 2020 and $3.8 billion (19%) in 2030. Remaining revenue is captured by the reduction of infrastructure and counter party risk for capital markets, as well as savings in Title Insurance commissions and maintenance cost. In the period 2020-2030, Autonomous Research estimates a CAGR of 12.6% for the Blockchain technology market presented in the next figure. [31]

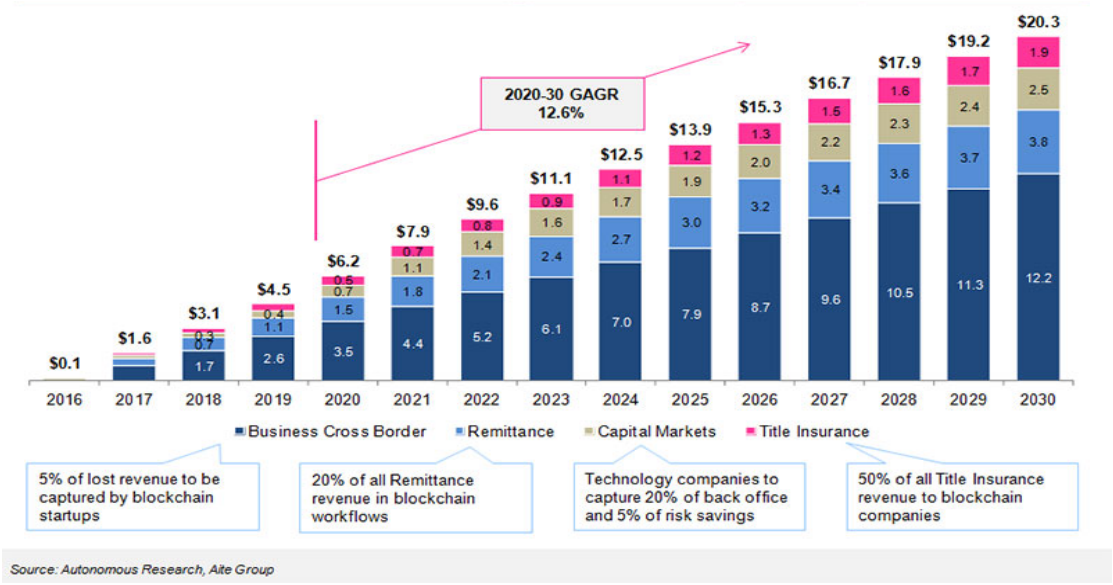

Figure 6: Blockchain Revenue Pool

4.1.3.1 Application and solution providers

Application and solution providers in the blockchain market, deliver significant value to the businesses by reducing duplication in transactions data, providing periodic reconciliation and authentication for commercial and regulatory reasons. These vendors provide an international online money transfer network and cloud-based services using blockchain that allows people to do transactions more easily. Application and solution providers have the potential to deliver disruptive outcomes and reshape digital businesses by providing distributed ledger technology to multiple industry verticals.

4.1.3.2 Digital identity

The digital identity management is said to be the fastest-growing application in the blockchain market, as it eliminates the need for central authority and third-party, thereby making it easier for the individuals to manage and take control over personal information and access. The vendors operating in the global market are focusing on developing commercially feasible solutions. Moreover, the focus of dominant players in the market is towards the development of blockchain-based identity management solutions for financial transactions and personal use cases.

4.1.3.3 BFSI

BFSI (Banking, Financial Services and Insurance) industry has realized the significance of distributed ledger technologies, which help secure the transaction for the customers. In addition, the technology shift from centralized infrastructure management to the distributed ecosystem, is paving the way for new business models in payments, internet banking, and financial transaction technologies. However, the real estate majorly works on traditional paper records to register land and property ownership which makes the process slow, time-consuming, and prone to fraud. Blockchain integration constantly records and shares information to address traditional process inefficiencies in the commercial real estate industry. The distributed ledger technology providers, offer a system in which anyone can access and record information, eliminate the middlemen and provide overall transparency to the buyer, seller, and government bodies involved in the process. Companies operating in the market provide blockchain platforms which are powered by tokens to provide lifelong access to office facilities.

4.2 Smart contracts data

According to a research report from Market Research Future the global smart contracts market is expected to reach approximately $300 Million by the end of 2023 with 32% CAGR during the forecast period from 2017-2023. The next figure presents this rate of growth in the coming years. [32]

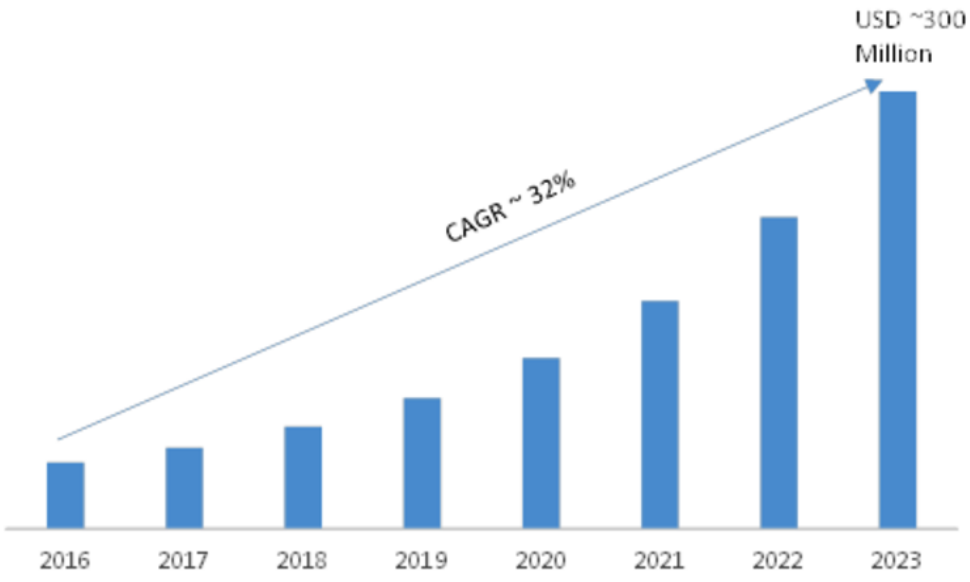

Figure 7: CAGR of Smart Contracts

4.2.1 Drivers

The main driver and at the same time the main factor that holds the growth of smart contracts, is their tendency to define the rules and regulations of an agreement but also automatically enforce any of the obligations. The smart contract can be used for situations such as financial, insurance premiums, contract breaches and property law. Smart contracts are expected to grow significantly in the forecast period catering the end users like banking, government, insurance, real estate and supply chain among others. For governmental purposes, smart contracts can prove to be the next advanced step towards voting, legal agreements or tenders.

4.2.2 Market segmentation

In the research of MarketsandMarkets, the market for smart contracts is segmented on the basis of blockchain platform, technology, end user and region. All the data and categorization presented below come from this research.

4.2.2.1 Blockchain platforms

On the basis of blockchain platform, the segmentation is divided into Bitcoin, Sidechains, NXT and Ethereum. Ethereum is the most advanced smart contract for coding and processing and accounts for the large share followed by Bitcoin and NXT.

4.2.2.2 Technology

In terms of technology, the segmentation is done on the basis of Ethereum, Rootstock, Namecoin, Ripple among others. The market holds many of the end users such as banking, government, management, supply chain, automobile, real estate, insurance and healthcare.

4.2.3 Regional Spread

In regional terms, the market is segmented in four regions: North America, Europe, Asia Pacific and the rest of the world. The market for smart contracts, is led by Europe while North America shows a significant growth. Asia Pacific and the rest of the world are far behind from these two competing markets.

4.3 IoT Data

The growth of the Internet of Things and the number of connected devices, is driven by emerging applications and business models, and supported by standardization and falling device costs. [34]

4.3.1 Key Data

- 70% of wide-area IoT devices will use cellular technology in 2022
- In 2018, mobile phones are expected to be surpassed in numbers by IoT devices
- There will be around 400 million IoT devices with cellular connections at the end of 2016
- Around 29 billion connected devices are forecasted by 2022, of which around 18 billion will be related to IoT.
- Between 2016 and 2022, IoT devices are expected to increase at a CAGR of 21%, driven by new use cases.

4.3.2 Connected IoT devices

In the figure below illustrating all connected devices, IoT is divided into short-range and wide-area segments. [34]

The short-range segment consists of devices connected by unlicensed radio with a typical range of up to around 100 meters, such as Wi-Fi, Bluetooth and ZigBee. This category also includes devices connected over fixed line local area connections. The wide-area IoT category consists of devices using cellular connections (3GPP-based with some CDMA), as well as, unlicensed low-power technologies, such as Sigfox, LoRa and Ingenu.

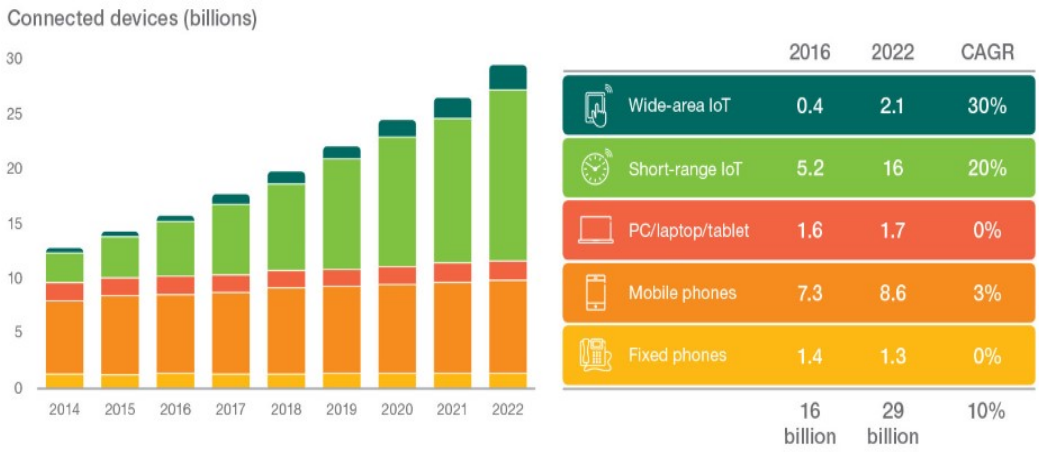

Figure 8: Forecast of Connected Devices

There will be around 400 million IoT devices with cellular connections at the end of 2016 and that number is projected to reach 1.5 billion in 2022, or around 70 percent of the wide-area category. This growth is due to increased industry focus and 3GPP standardization of cellular IoT technologies.

Today, LTE's share of cellular IoT devices is around 5 percent. Declining modem costs, evolving LTE functionality and 5G capabilities are all expected to extend the range of applications for critical IoT deployments.

5 Study and Comparative evaluation of Blockchain Platforms supporting Smart Contracts

5.1 Platforms under evaluation

Hyperledger, Ethereum, Stellar, Monax and Lisk are some major open source implementations of distributed ledgers with usability for Internet of Things. They all have common fundamental characteristics such as: linking among blocks, being cryptographically secure for hashing of multiple transactions in single blocks, asymmetric cryptography, digital signatures, consensus mechanism and smart contracts support. However, their implementations vary significantly as well as their extent of applicability to IoT considering how different aspects and particularly smart contracts are applied to these platforms.

5.1.1 Hyperledger Fabric

Hyperledger is a permissioned blockchain which uniquely applies access control, smart contracts based on chaincode, variable consensus with a current implementation of practical byzantine fault tolerance (PBFT) and includes trust anchors to root certificate authorities as an enhancement to the asymmetric cryptography and digital signature features with SHA3 and ECDSA [35]. The permissioned nature of Hyperledger enhances security of the network by means of preventing attacks like Sybil attacks- an attack in which consensus could potentially be threatened by a malicious entity creating and enrolling illegitimate (Sybil) peers to affect the network adversely [36]. Furthermore, the implementation of smart contracts in Hyperledger involves the chaincode, which can self-execute conditions such as asset, or resource transfers among peers in hundreds of milliseconds [37], [38]. This latency is low among comparative blockchain systems. Hyperledger's adoption of PBFT prevents the probabilistic and computationally expensive mining of hashes. However, there is a trade off between the immediate computational overheads with network utilization. In the context of Internet-of-Things, the scale at which network utilization increases in

comparison with the increase in the number of devices on a network, must be investigated and measured further. Overall, between applying smart contracts based on chaincode and a unique PBFT implementation which offsets computational overhead for increased networking among peers, Hyperledger offers robust platform of applications for Internet-of-Things.

5.1.2 Ethereum

Ethereum began as an alternative cryptocurrency solution to compete Bitcoin but further on things have changed. It has some special characteristics, as it is an adaptable blockchain implementation with an implementation of smart contracts and a derivative of proof-of-work consensus known as Ethash. This also applies to directed acyclic graphs to manage probabilistic hash generation in matters that will prevent potential abuse from specialized hardware where other proof-of-work algorithms are vulnerable to [39], [40]. In addition to implementing smart contracts, Ethereum transactions can also store custom data. This increases the potential for auditability and immutability of IoT data beyond cryptocurrency transactions and allows robust extensibility for IoT applications that involves performance tradeoffs. Due to Ethash being based upon proof-of-work, Ethereum is very fast (compared to Bitcoin's proof-of-stake) and may require between 10 to 20 seconds to produce a block. Still high frequency and time-sensitive IoT device operations may not support such delays [37], [41]. While Ethash prevents abuses from potential specialized hardware, it does not necessarily enhance fault tolerance. At scale, IoT devices would need to rely on trusted and computationally powerful peers to ensure fault handling.

Storage also presents another problem, as Ethereum requires all peers to store a blockchain that is tens of gigabytes larger. IoT devices, that normally don't have such storage capacity, will either need to intercommunicate with a proxy server that will act as a peer in the Ethereum network or accommodate large storage. Ethereum, as it is used longer than most distributed ledger implementations, has IoT prototypes, such as handling tokens and contracts for electronic lock sharing and supply chain assurance prototypes [41].

5.1.3 Monax

Monax supports the execution of Ethereum contracts, without having its own currency. Monax allows users to create private blockchains, and define authorization

policies for accessing them. Its consensus protocol is organised in rounds, where a participant proposes a new block of transactions and the others vote for it. When a block fails to be approved, the protocol moves to the next round, where another participant will be in charge of proposing blocks. A block is confirmed when it is approved by at least 2/3 of the total voting power. [23]

5.1.4 Stellar

Stellar features a public blockchain with its own cryptocurrency, governed by a consensus algorithm inspired by federated Byzantine agreement. Basically, a node agrees on a transaction if the nodes in its neighborhood (that are considered more trusted than the others) agree as well. When the transaction has been accepted by enough nodes of the network, it becomes infeasible for an attacker to roll it back, and it can be considered as confirmed. Compared to proof-of-work, this protocol consumes far less computing power, since it does not involve solve cryptographic puzzles. Unlike Ethereum, there is no specific language for smart contracts; it is still possible to gather together some transactions (possibly ordered in a chain) and write them atomically in the blockchain. Since transactions in a chain can involve different addresses, this feature can be used to implement basic smart contracts. For instance, assume that a participant A wants to pay B only if B promises to pay C after receiving the payment from A. This behavior can be enforced by putting these transactions at the same chain. While this specific example can be implemented on Bitcoin as well, Stellar also allows to batch operations different from payments, as creating a new account. Stellar features special accounts, called multisignature that can be handled by several owners. To perform operations from these accounts, a threshold of consensus must be reached among the owners. Transaction chaining and multisignature accounts can be combined to create more complex contracts. [23]

5.1.5 Lisk

Lisk has its own currency, and a public blockchain with a delegated proof-of-stake consensus mechanism. More specifically, 101 active delegates, each one elected by the stakeholders, have the authority to generate blocks. Stakeholders can take part in the electoral process, by placing votes for delegates in their favor, or by becoming candidates themselves. Lisk supports the execution of Turing-complete smart contracts, written either in JavaScript or in Node.js. Unlike Ethereum, determinism of

executions is not ensured by the language: thus, programmers must take care of it e.g. by not using functions like Math.random. Although Lisk has a main blockchain, each smart contract is executed on a separated one. Users can deposit or withdraw currency from a contract to the main chain, while avoiding double spending. Contract owners can customize their blockchain before deploying their contracts, e.g. choosing which nodes can participate to the consensus mechanism.

5.2 Method and criteria of evaluation

The chosen platforms for the purpose of this master thesis are: Hyperledger Fabric, Ethereum, Monax, Stellar, Lisk. They satisfy the following criteria:

(i) they have already been launched.

(ii) they are publicly accessible.

(iii) they are popular or they seem to make a breakthrough in the field.

The features that we consider for the platforms under evaluation are:

- **General**
  - (i) Permission restrictions
  - (ii) Privacy/access to data
  - (iii) Ownership/Governance
  - (iv) Blockchain Type
  - (v) Release Year
- **Financial (31/12/2017)**
  - (vi) Currency
  - (vii) Market Cap
  - (viii) Average Transaction fee
  - (ix) Cost Model
- **Technical**
  - (x) Scalability
  - (xi) Blockchain size
  - (xii) Consensus Mechanism
  - (xiii) Turing-Complete
  - (xiv) Contract Language
  - (xv) Blockchain Interval
  - (xvi) Anonimity

5.3 Comparison Table

Following, the comparison matrix presents the main characteristics of the platforms included in the test revealing differences and similarities. Wherever there is no information found or the feature does not exist, N/A (not applicable) appears on the table.

	Platform Name				
Features/Characteristics	**Ethereum**	**Hyperledger Fabric**	**Monax**	**Lisk**	**Stellar**
General					
Permission restrictions	Permissionless	Permissioned	Permissionless	Permissionless	Permissionless
Privacy/access to data	Public or private	Private	Public	Public	Public
Ownership/Governance	Ethereum Foundation	Linux Foundation	Monax Industries Limited	MIT	Stellar Development Foundation
Blockchain Type	Public	Public	Private	Private	Public
Release Year	2015	2015	2014	2016	2014
Financial (31/12/2017)					
Currency	Ether	No	No	LSK	XLM
Market Cap	$69.767.510.695	N/A	N/A	$2.221.966.516	$ 5.676.400.000
Average Transaction fee	$2,50	N/A	N/A	flat 0.1 LSK	fixed fee of 0.00001 XLM
Cost Model	N/A	N/A	N/A	N/A	The Stellar network is free to use.
Technical					
Scalability	High node scalability, Low performance scalability	Low node scalability, High performance scalability	N/A	N/A	N/A
Blockchain size	17-60GB	N/A	N/A	N/A	N/A
Consensus Mechanism	PoW	PBFT	Tendermint	DPoS	Stellar Consensus

					Protocol
Turing-Complete	Yes	Yes	N/A	Yes	No
Contract Language	Solidity, Serpent, LLL	Go	Solidity	JavaScript	Javascript, Go, Ruby, Python, C
Blockchain Interval	12sec	Custom	Custom	Custom	3sec
Anonimity	Pseudonymity, no encryption of transaction data	Pseudonymity, encryption of transaction data	N/A	N/A	N/A

Table 5: Comparison of Blockchain Platforms

6 Security Issues in Blockchain and Smart Contracts

6.1 Known Security Issues in Blockchain

Despite the attention that the blockchain technology has received in the community of researchers, practitioners, and developers, it still faces several challenges. Some of the known security issues in blockchains are presented below:

6.1.1 The Majority Attack

In a Proof of Work based consensus mechanism, the probability of mining a block depends on the work done by the miner (e.g. CPU/GPU cycles spent checking hashes). Thus, providing an incentive to the miners to join together in order to mine more blocks, and become "mining pools". Mining pools can be defined as miners with huge computing power. Once it holds 51% computing power in a particular blockchain network, it can then take control of the blockchain. Apparently, it causes security issues because if someone has more than 51% computing power, then he/she can find Nonce value quicker than others, means he/she has authority to decide which block is permissible. What it can do is:

1. Modify the transaction data; it may cause double spending attack.
2. To stop the block verifying transaction.
3. To stop miner mining any available block.

A majority attack was more feasible in the past when most transactions were worth significantly more than the block reward and when the network hash rate was much lower and prone to reorganization with the advent of new mining technologies.

6.1.2 Fork Problems

Another known security issue is fork problem. Fork problem is related to decentralized node version, agreement when the software upgrade. It is a critical issue as it affects a wide range of blockchain. When a new version of blockchain software is published, new agreement in consensus rule is also changed for the nodes. Therefore, the nodes in blockchain network can be divided into two types: New

Nodes; and Old Nodes. This may lead to following four situations:

1. The new nodes agree with the transaction of blocks sent by the old nodes.
2. The new nodes don't agree with the transaction of blocks sent by the old nodes.
3. The old nodes agree with the transaction of blocks sent by the new nodes.
4. The old nodes don't agree with the transaction of blocks sent by the new nodes.

It is because of the above-mentioned different cases in getting consensus, the fork problem arises.

Based on the type of consensus issue, the fork problems can be categorized as: Hard Fork; and Soft Fork. In addition to distinguish between a new node and an old node, we also need to compare the computing power of new nodes with old nodes, and assume that the computing power of new nodes is more than 50.

6.1.2.1 Hard Fork

Hard Fork implies that a system is not compatible with the new version or new agreement, and there are compatibility issues with the previous version such that the old nodes do not agree with the mining of new nodes; thus, creating two forks of the blockchain. On the other hand, new nodes may have higher computing power than the old nodes, yet, the old nodes will continue to maintain the chain which they think is right.

When a Hard Fork situation occurs then all nodes in the network to should be upgraded the latest version, agreement. Nodes, which are not upgraded, will not work as usual. Thus, if there are a large number of old nodes that did not upgrade to the latest agreement, then they will continue to work on a completely different chain. This implies that the ordinary chain will fork into two chains.

6.1.2.2 Soft Fork

Soft Fork occurs when a system comes to a new version or new agreement, but is not compatible with the previous version. In such a scenario, the new nodes couldn't agree with the mining of old nodes. Since the computing power of new nodes is stronger than the old nodes, the blocks mined by the old nodes will never be approved by the new nodes, but new nodes and old nodes will still continue to work on the same chain. When Soft Fork happens, nodes in the network don't have to upgrade to the new agreement at the same time, it allows them to upgrade gradually. Unlike Hard Fork, Soft Fork will only have one chain, it won't affect the stability and effectiveness

of the system when nodes upgrade. However, Soft Fork makes the old nodes unaware that the consensus rule has changed, contrary to the principle of every node can verify correctly to some extent.

6.1.3 Scale of Blockchain

As the blockchain grows and data size swells, the loading and computing will become harder and harder; it will take a lot more time to synchronize data, in the same time, data still continue to increase, thus, creating a huge problem for the client running the system. A potential solution to this problem is Simplified Payment Verification (SPV) technology. SPV is a payment verification technology, it does not require full blockchain information and only the block header message is used for transaction. This technology can greatly reduce users' storage problem and lower the pressure on users' when the number of transactions increase in future.

6.1.4 Time Required to Confirm Blockchain Transactions

Blockchain came to existence in the form of a "cryptocurrency" called Bitcoin as an alternative to fiat currency and electronic transactions. In comparison to traditional online credit card transaction that usually takes 2 to 3 days to confirm the transaction, Bitcoin transactions usually take an hour to verify. Thus, Bitcoin clearly is better than traditional transaction mechanisms however; it is still not good enough to what we want it to. A potential solution is Lightning Network. Lightning Network is a proposed implementation of Hashed Timelock Contracts (HTLCs) with bi-directional payment channels, which allows payments to be securely routed across multiple peer-to-peer payment channels. This allows the formation of a network where any peer on the network can pay any other peer even if they don't directly have a channel open between each other.

6.2 Known Security Issues in Smart Contracts

Given that the smart contracts are computer codes that define contractual agreement between two or more parties, it is crucial to look at some of the well-known security issues in the smart contracts mechanism. Some of the issues are as follows:

- **Reentrancy:** Reentrancy should not be used in smart contracts. If a coder performs external calls in contracts then it should be ensured that it is absolutely necessary.
- **Send can fail:** When executing the instructions such as sending money, the code

should always be prepared for the send function to fail.

- **Loops can trigger gas limit:** It is important to be careful when looping over state variables, which can grow in size and make gas consumption (in Ethereum based smart contracts) hit the limits.
- **Call stack depth limit:** One should not use recursion in smart contracts, and should be aware that any call can fail if stack depth limit is reached.
- **Timestamp dependency:** Smart contract coders should avoid using timestamps in critical parts of the code as the miners can manipulate the timestamps.

7 Security Assessment in Ethereum Smart Contracts

In this section we will identify the known vulnerabilities, the attacks, their explanation and the countermeasures concerning smart contracts.

7.1 Vulnerabilities

According to the authors of the paper [43], there is a number of discovered vulnerabilities that take advantage of issues found in different components that can be exploited to perform a certain attack. These components are listed in three categories based on the component they affect: Solidity (the language), the EVM bytecode (the "operating system") and the blockchain itself. These vulnerabilities are described below:

Call to the unknown: Some of the primitives used at Solidity to invoke functions and transfer ether, may have the side effect of invoking the fallback function of the callee/recipient.

Gasless send: When using the function send to transfer ether to a contract, it is possible to incur in an out-of-gas exception. This may be quite unexpected by programmers, because transferring ether is not generally associated to executing code.

Exception disorder: In Solidity there are several situations where an exception may be raised, e.g. if (i) the execution runs out of gas; (ii) the call stack reaches its limit; (iii) the command throw is executed. However, Solidity is not uniform in the way it handles exceptions: there are two different behaviors, which depend on how contracts call each other.

Reentrancy: The atomicity and sequentiality of transactions may induce programmers to believe that when a non-recursive function is invoked, it cannot be re-entered before its termination. However, this is not always the case, because the fallback mechanism may allow an attacker to re-enter the caller function. This may result in unexpected behaviors, and possibly also in loops of invocations, which eventually consume all the gas.

Keeping secrets: Fields in contracts can be public, i.e. directly readable by everyone, or private, i.e. not directly readable by other users/contracts. Still, declaring a field as

private, does not guarantee its secrecy. This is because, to set the value of a field, users must send a suitable transaction to miners, who will then publish it on the blockchain. Since the blockchain is public, everyone can inspect the contents of the transaction, and infer the new value of the field.

Ether lost in transfer: When sending ether, one has to specify the recipient address, which takes the form of a sequence of 160 bits. However, many of these addresses are orphan, i.e. they are not associated to any user or contract. If some ether is sent to an orphan address, it is lost forever (note that there is no way to detect whether an address is orphan). Since lost ether cannot be recovered, programmers have to ensure manually the correctness of the recipient addresses.

Stack size limit: Each time a contract invokes another contract or even itself, the call stack associated with the transaction grows by one frame. The call stack is bounded to 1024 frames and when this limit is reached, a further invocation throws an exception.

Unpredictable state: The state of a contract is determined by the value of its fields and balance. In general, when a user sends a transaction to the network in order to invoke some contract, he cannot be sure that the transaction will run at the same state as the contract was at the time of sending that transaction. This may happen because, in the meanwhile, other transactions have changed the contract state. Even if the user was fast enough to be the first one to send a transaction, it is not guaranteed that such transaction will be the first to run. Indeed, when miners group transactions into blocks, they are not required to preserve any order; they could also choose not to include some transactions. There is another circumstance where a user may not know the actual state wherein his transaction will run. This happens in case the blockchain forks

Generating randomness: The execution of EVM bytecode is deterministic: in the absence of misbehavior, all miners executing a transaction will have the same results. Hence, to simulate non-deterministic choices, many contracts (e.g. lotteries, games, etc.) generate pseudo-random numbers, where the initialization seed is chosen uniquely for all miners. A common choice is to take for this seed (or for the random number itself) the hash or the timestamp of some block that will appear in the blockchain at a given time in the future. Since all the miners have the same view of the blockchain, at run-time this value will be the same for everyone. Apparently, this is a secure way to generate random numbers, as the content of future blocks is unpredictable. However, since miners control which transactions are put in a block

and in which order, a malicious miner could attempt to craft his block in order to bias the outcome of the pseudo-random generator.

Time constraints: A wide range of applications use time constraints in order to determine the permitted actions in the current state. Typically, time constraints are implemented by using block timestamps, which are agreed upon by all miners.

The next table summarizes and categorizes the vulnerabilities and the components they affect.

Component	Vulnerability
Solidity	Call to the unknown
	Gasless send
	Exception disorders
	Type casts
	Reentrancy
	Keeping secrets
EVM bytecode	Immutable bugs
	Ether lost in transfer
	Stack size limit
Blockchain	Unpredictable state
	Generating randomness
	Time constraints

Table 6: Known Vulnerabilities in Ethereum smart contracts

Alharby and Moorsel [45] provide another taxonomy based on a systematic mapping study of current research topics related to smart contracts. In their study, they discovered four key smart contract issues;

Codifying issues: they refer to the challenges/mistakes that are related with the development of smart contracts.

Security issues: they include bugs or vulnerabilities

Privacy issues: they refer to the issues related to unintentional information disclosure to the public.

Performance issues: they are related to the challenges that affect the ability of blockchain to scale.

7.2 Attacks and Description

In the next paragraphs, we will provide a general description of some known attacks that have taken place or can be used from a malicious actor against smart contracts supported by Ethereum. The attacks are divided according to the level they are introduced, meaning Solidity, EVM bytecode and blockchain as in paper [45].

7.2.1 The DAO attack

The DAO was a contract implementing a crowd-funding platform, which raised around $150M before being attacked on June 18th, 2016. An attacker managed to put approximately $60M under her control, until the hard-fork of the blockchain nullified the effects of the transactions involved in the attack.

7.2.2 King of the Ether Throne

The “King of the Ether Throne" is a game where players compete for acquiring the title of king. If someone wishes to be the king, he must pay some ether to the current king, plus a small fee to the contract.

7.2.3 Multi-player games

Consider a contract, which implements a simple “odds and evens” game between two players. Each player chooses a number: if the sum is even, the first player wins, otherwise the second one does.

7.2.4 Rubixi

Rubixi is a contract which implements a Ponzi scheme, a fraudulent high-yield investment program where participants gain money from the investments made by newcomers. Further, the contract owner can collect some fees, paid to the contract upon investments. There is an attack that allows an adversary to steal some ether from the contract, exploiting the “immutable bugs” vulnerability. At some point during the development of the contract, its name changed from DynamicPyramid into Rubixi. However, programmers forgot accordingly to change the name of the constructor, which then became a function that could be invoked by anyone. After this bug became public, users started to invoke DynamicPyramid in order to become the owner, and as a result to withdraw the fees.

7.2.5 GovernMental

GovernMental is another known Ponzi scheme. To join the scheme, a participant must send a certain amount of ether to the contract. If no one joins the scheme for 12 hours, the last participant gets all the ether in the contract (except for a fee kept by the owner). The list of participants and their credit are stored in two arrays. When the 12 hours are expired, the last participant can claim the money, and the arrays are cleared The EVM code obtained from this snippet of Solidity code clears one-by-one each location of the arrays. At a certain point, the list of participants of GovernMental grew so long, that clearing the arrays would have required more gas than the maximum allowed for a single transaction. From that point, any attempt to clear the arrays has failed. The contract did not check if the operations were successful, leaving it vulnerable to attack.

7.2.6 Dynamic libraries

We now consider a contract, which can dynamically update one of its components, which is a library of operation on sets. Therefore, if a more efficient implementation of these operations is developed or if a bug is fixed, the contract can use the new version of the library.

7.2.7 Summary and Evaluation of Attacks

Based on the components they affect, we attempt a qualitative evaluation in the scale of low-high with low being the least effective attack and high being the most dangerous attack. The severity is related to the amount of money claimed by the hackers or how fundamental to the structure the attack is (i.e GovernmMental attack was a code mistake). The results are shown in the next table.

Attack	**Severity**
The DAO attack	High
King of the Ether Throne	Medium
Multi-player games	Low
Rubixi	Low
GovernMental	High
Dynamic libraries	High

Table 7: Known Attacks in Ethereum smart contracts

7.3 Countermeasures

In the papers [43], [45] authors propose as countermeasures, different types of tools able to prevent the vulnerabilities in smart contracts. Some solutions are explained below:

7.3.1 Methods

ZeppelinOS: is an operating system for smart contract applications developed by Zeppelin Solutions [40]. As referred to by Zeppelin Solutions, ZeppelinOS is "an open-source, distributed platform of tools and services on top of the EVM to develop and manage smart contract applications securely" [40]. Zeppelin introduces a novel approach in developing smart contracts by using already developed and secure smart contracts (i.e. libraries).

SolCover: provides code coverage for Solidity testing and by relying on that, it measures and describes the degree of overall testing in a smart contract.

HackThisContract: It is a crowdsourcing experimental website that encourages developers to test their smart contracts before deployment by uploading it on their website. Other developers, with their own techniques, will try and exploit possible vulnerabilities. Additionally, they provide a list of vulnerable smart contract examples, which the developers should not follow. Overall, with the sole purpose of deploying secure smart contracts and mitigate (eliminate) severe issues in a pre-deployment phase.

Hard Fork: Among other suggestions, the authors of the paper [42] suggest an upgrade in the Ethereum platform adding functionalities that can improve operational semantics and face security issues such as: guarded transactions to deal with transaction ordering dependence (TOD), deterministic timestamp and exception handling.

7.3.2 Software Tools:

a. **Oyente:** The tool Oyente extracts the control flow graph from the EVM bytecode of a contract, and symbolically executes it in order to detect some vulnerability patterns. In particular, the tool considers the patterns leading to vulnerabilities of kind exception disorder (e.g. not checking the return code of call, send and delegate call), time

constraints (e.g. using block timestamps in conditional expressions), unpredictable state, and reentrancy.

b. **Distributed proof market and program verification in an interactive theorem prover (ITP).** In this approach we translate smart contracts, either Solidity or EVM bytecode, into the functional language F*. Various properties are then verified on the resulting F* code. In particular, code obtained from Solidity contracts, is checked against exception confusion and reentrancy vulnerabilities, by looking for specific patterns. Code obtained from EVM supports low-level analyses, like e.g. computing bounds on the gas consumption of contract functions. Furthermore, given a Solidity program and an alleged compilation of it into EVM bytecode, the tool verifies that the two pieces of code have equivalent behaviors. Both tools have been experimented on the contracts published in blockchain of Ethereum.

c. **SmartPool:** In pooled mining, miners combine their power to solve the PoW puzzles together and split the reward according to each's contribution. This approach is called pooled mining in which miners are asked to solve pool-puzzles much easier. SmartPool is a decentralized pooled mining protocol for cryptocurrencies that leverages smart contracts in existing cryptocurrencies, coupling with its data structures and verification mechanism, providing security and efficiency to miners.

d. **Hawk:** Hawk is a decentralized smart contract system that does not store financial transactions in the clear on the blockchain, resulting in retaining transactional privacy from the public's view. The Hawk compiler automatically generates a cryptographic protocol where contractual parties interact with the blockchain, using cryptographic primitives such as zero-knowledge proofs.

e. **Town Crier:** Town Crier (TC) is a system that addresses this challenge by providing an authenticated data feed (ADF) for smart contracts. TC acts as a high-trust bridge between existing https-enabled data websites and the Ethereum blockchain. It retrieves website data and serves it to relying contracts on the blockchain as concise pieces of data (e.g. stock quotes) called datagrams. TC uses a novel combination

of Software Guard Extensions (SGX), Intel's recently released trusted hardware capability, and a smart-contract front end. It executes its core functionality as a trusted piece of code in an SGX enclave, which protects against malicious processes and the OS and can attest (prove) to a remote client that the client is interacting with a legitimate, SGX-backed instance of the TC code.

f. **Remix:** is a web-based IDE that allows users to write, deploy and run Solidity smart contracts. In Remix we find integrated a debugger and a testing environment (test-blockchain network). It serves as a security tool by analyzing the Solidity code only (a setback for the tool), to reduce coding mistakes and identify potential vulnerable coding patterns. Its security analysis relies on formal verification that is a deductive program verification and theorem provers.

7.4 Relational Table

Component	Vulnerabilities	Attacks	Countermeasures
Solidity	Call to the unknown	The DAO attack	Methods or Software tools
	Gasless send	King of the Ether Throne	
	Exception disorders	King of the Ether Throne, GovernMental	
	Type casts	-	
	Reentrancy	The DAO attack	
	Keeping secrets	Multi-player games	
EVM bytecode	Immutable bugs	Rubixi, GovernMental	
	Ether lost in transfer	-	
	Stack size limit	GovernMental	
Blockchain	Unpredictable state	GovernMental, Dynamic libraries	
	Generating randomness	-	
	Time constraints	GovernMental	

Table 8: Smart Contracts Vulnerabilities-Attacks and Countermeasures

8 Blockchain and Smart Contracts Applications in IoT

8.1 About Internet of Things (IoT)

As defined by ITU [25], the Internet of Things (IoT) refers to the network of numerous physical objects (20 billion by 2020, according to Gartner, [26] which are provided with Internet connection. Such devices acquire information about the surrounding environment, and they communicate with each other and with other systems through Internet. As a consequence of that rich interaction, they also produce a large amount of data, usable in turn to enable dependent services. Possible use cases for smart contracts and IoT are presented in this section.

8.2 Use Cases for Smart Contracts and IoT

Further on are the 12 smart contract business use cases as explained and presented in the paper *"Smart Contracts: 12 Use Cases for Business & Beyond A Technology, Legal & Regulatory Introduction"* [27] by the Chamber of Digital Commerce with the support of the Smart Contracts Alliance. In the next paragraphs we will explain the cases illustrated in this paper including the explanatory figures (9-20) from the above paper.

8.2.1 Digital Identity

Smart contracts can allow individuals to own and control their digital identity containing data, reputation and digital assets increasing compliance, resiliency and interoperability. They permit individuals to decide what data to disclose to counterparties, providing companies the opportunity to get to know with their customers. The different parties will not have to hold sensitive data to verify transactions reducing liability while facilitating know-your-customer requirements. The figure below describes how the current state will transform with the use of blockchain. [27]

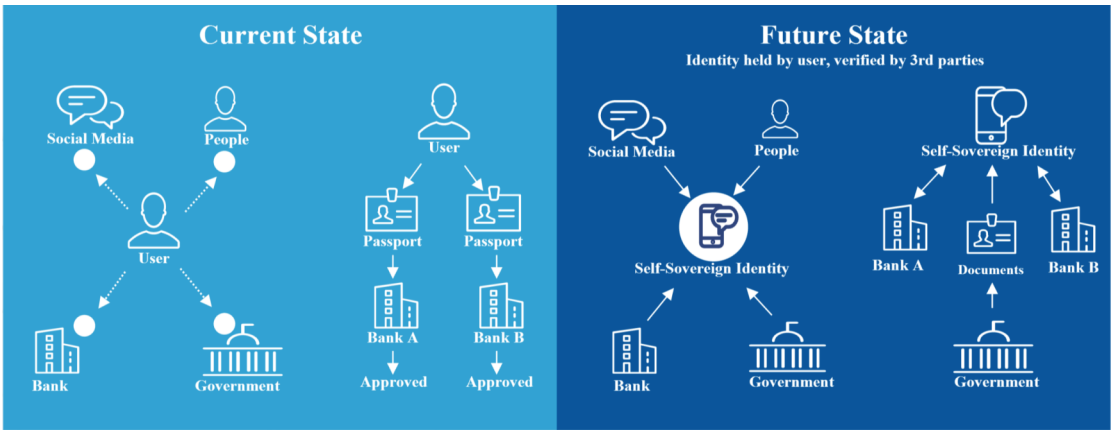

Figure 9: Digital Identity

8.2.2 Records

In the United States smart contracts can reduce legal costs as they can potentially digitize the Uniform Commercial Code (UCC). This can be achieved by filing and automating the renewal and release processes. They can automate compliance with the rules that require destroying records at a future date. They also make possible a UCC feature that allows a lender to establish priority in repayment in case of debtor default or bankruptcy by automatically releasing and renewing or automatically requesting collateral as shown in the figure below.[27]

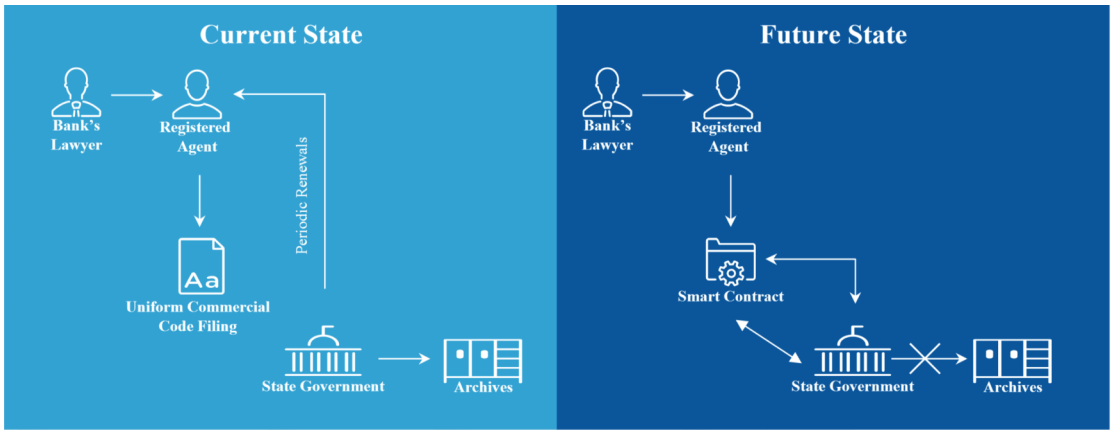

Figure 10: Records

8.2.3 Securities

Smart contracts can exclude intermediaries in the chain of securities custody and facilitate the automatic payment of dividends, stock splits and liability management, while reducing operational risks and digitizing work flows. Visibility into who owns their securities is an issues because it is accepted and welcomed by some issuers of

securities while others want to protect this information as described in the next figure. [27]

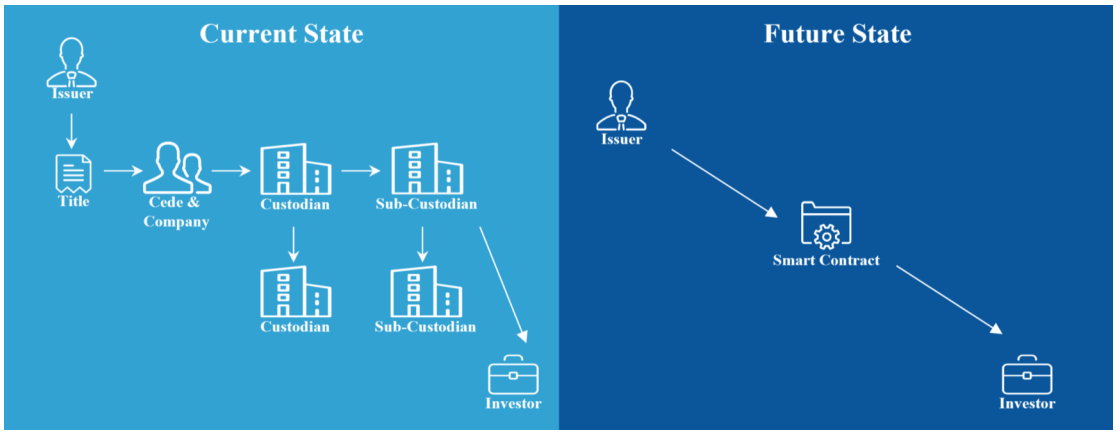

Figure 11: Securities

8.2.4 Trade Finance

Smart contracts can improve international transfers of goods reducing time through fast letter of credit and trade payment initiation, while enabling a greater liquidity of financial assets. They can also improve financing efficiencies for buyers, suppliers and institutions. Because of a complex international trading system it is necessary to integrate its parameters and procedures into smart contracts in order to benefit the most out of it. The process reform in described next. [27]

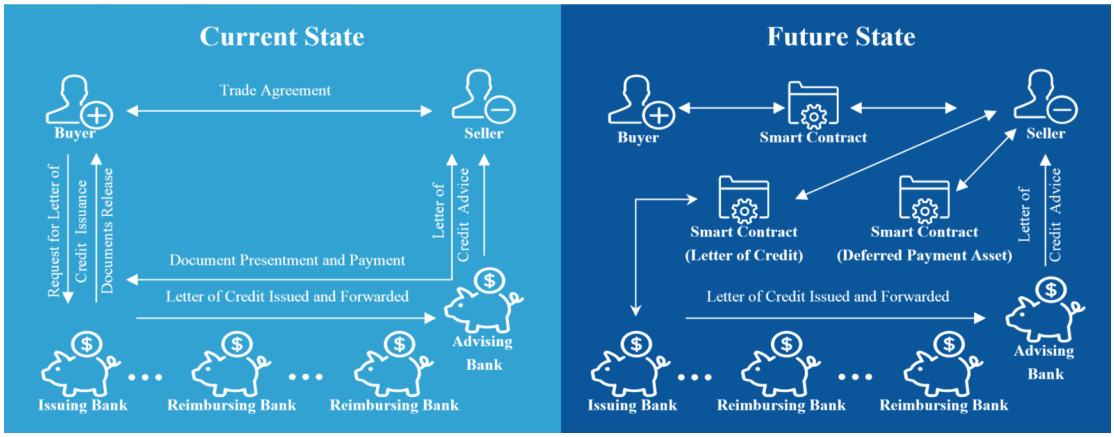

Figure 12: Trade Finance

8.2.5 Derivatives

It is important to address protocol changes related to regulatory reform for derivative smart contracts to improve the post-trade processes by removing duplicate processes executed by the participating counterparties. They can enable a standard set of

contract conditions and a real-time valuation of positions for monitoring, preventing and reducing mistakes. [27]

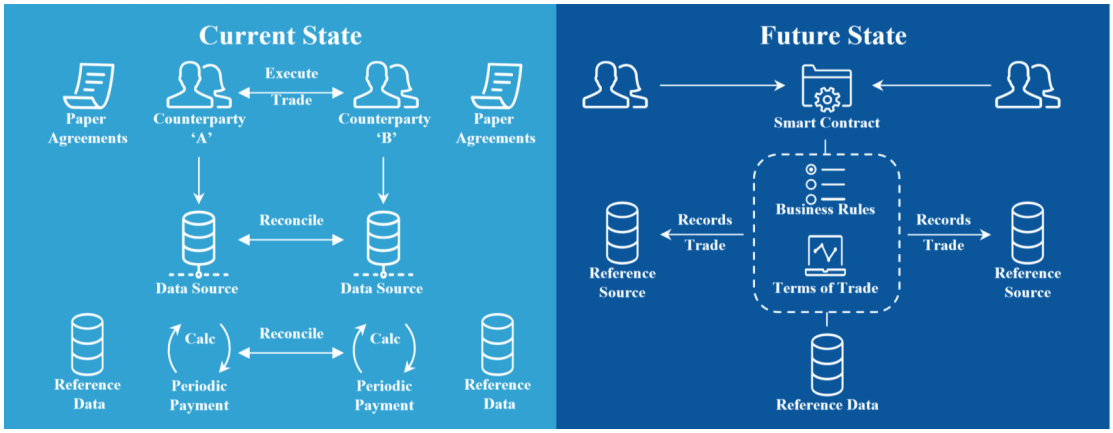

Figure 13: Derivatives

8.2.6 Financial Data Recording

Financial organizations can utilize smart contracts for accurate, transparent financial data recording. Smart contracts allow for uniform financial data across organizations, improving financial reporting and reducing accounting and auditing costs. Data integrity through smart contracts increases market stability. [27]

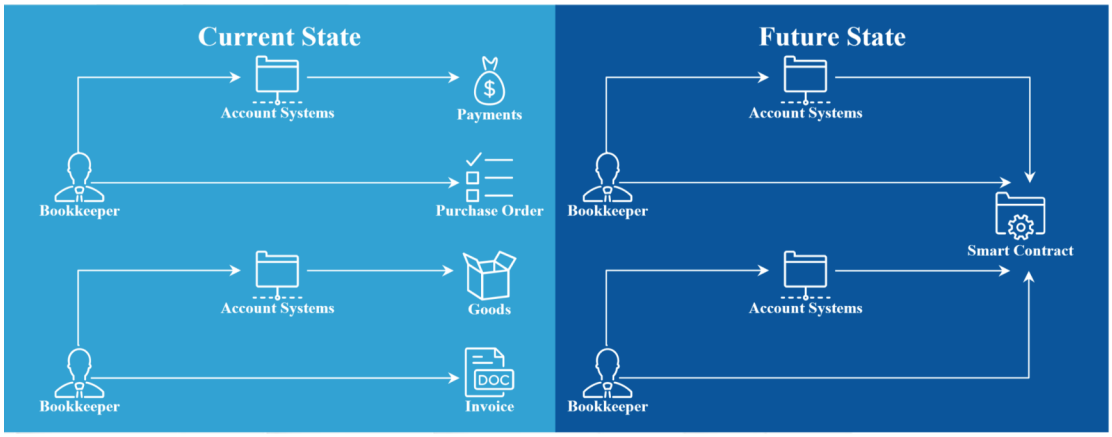

Figure 14: Financial Data Recording

8.2.7 Mortgages

Smart contracts using digital identity as a prerequisite can automate mortgage contracts by automatically connecting the parties, providing a frictionless and less error-prone process. The payment process can be automated and release liens from land records when the loan is paid. They can also improve record visibility for all parties and reduce errors and costs associated with manual processes. [27]

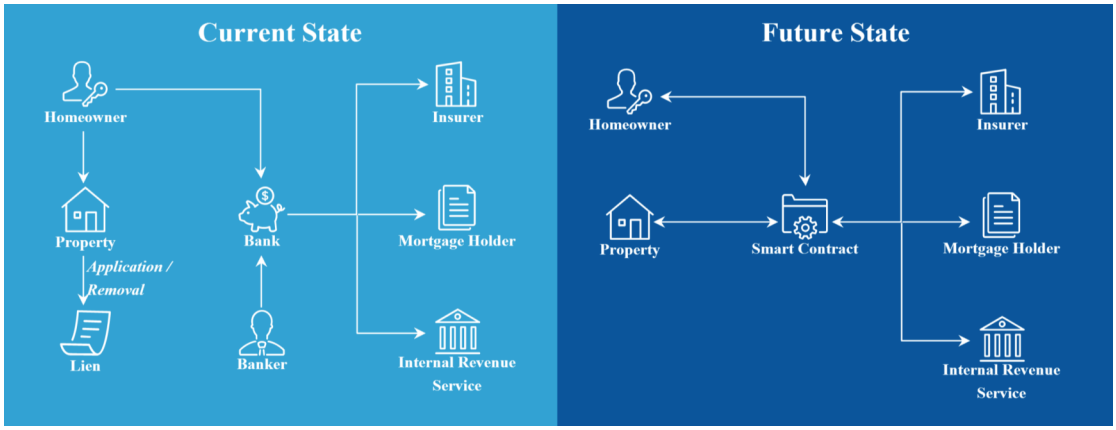

Figure 15: Mortgages

8.2.8 Land Title Recording

Smart contracts that facilitate property transfers, can improve transaction transparency and efficiency and strengthen confidence in identity, reducing auditing costs. In order to succeed electronic record filing, common protocols need to be developed.

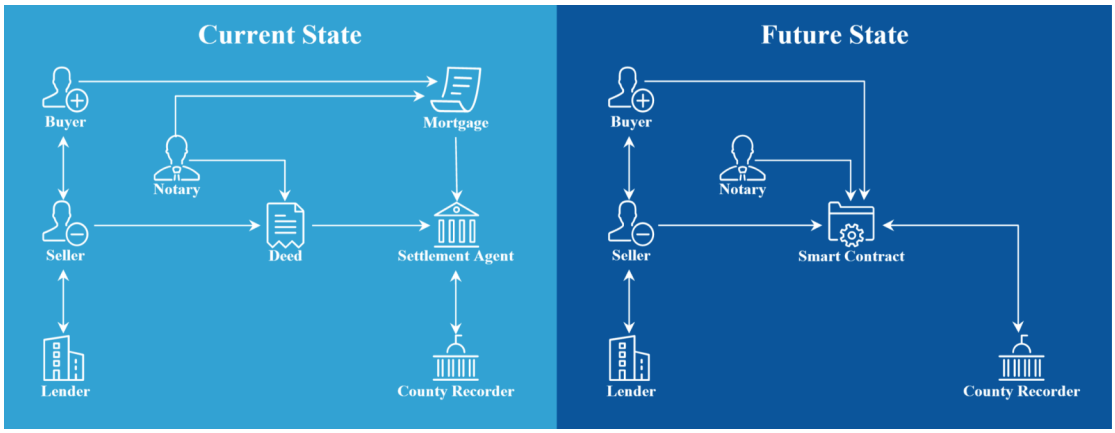

Figure 16: Land Title Recording

8.2.9 Supply Chain

Smart contracts can provide visibility for all stages in the supply chain and using IoT devices can keep record of a product moving from the production to the retail store. Consequently, inventory tracking is facilitated, benefiting supply chain financing, insurance and risk. [27]

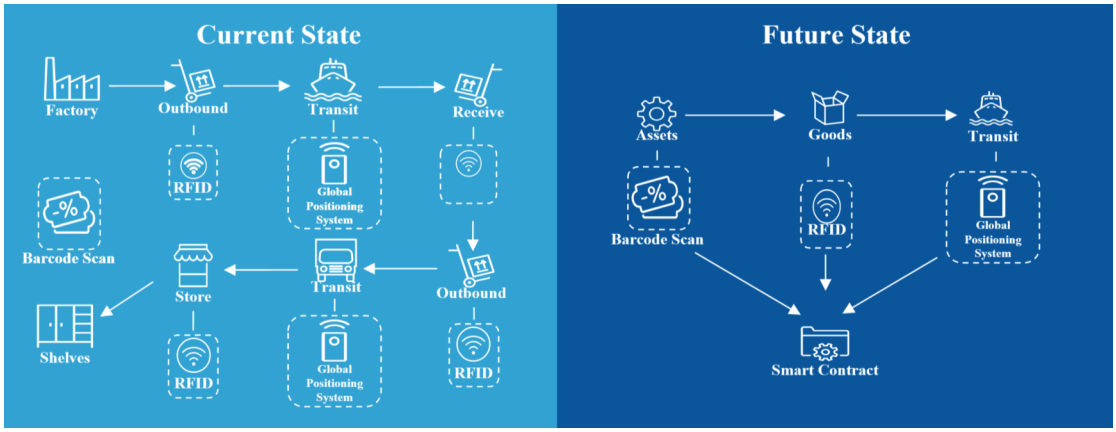

Figure 17: Supply Chain

8.2.10 Auto Insurance

Smart contracts can improve the auto insurance process through cross-industry collaboration in order to cope with the technological, regulatory and financial issues.. A smart contract can record the policy, driving record and driver reports, allowing future vehicles equipped with IoT devices to execute claims shortly after an accident automating claims processing, verification and payment. [27]

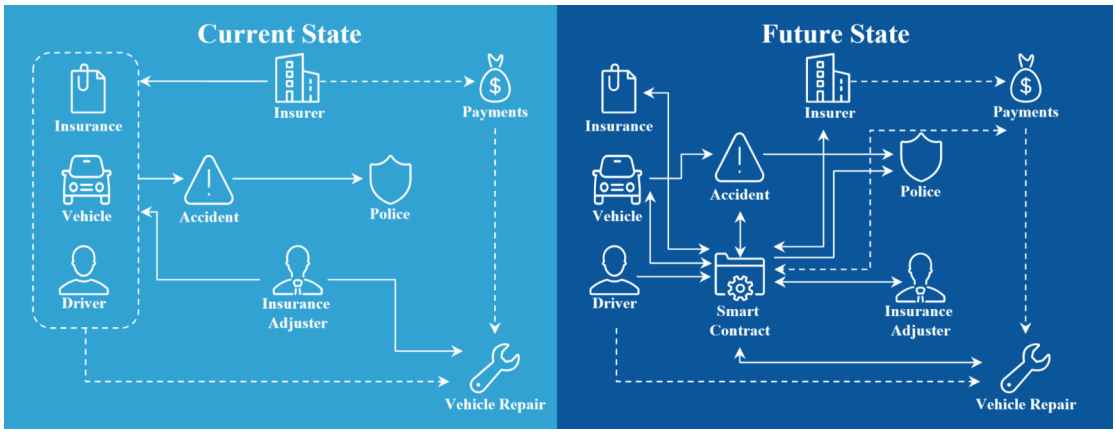

Figure 18: Auto Insurance

8.2.11 Clinical Trials

Smart contracts can improve clinical trials through increased cross-institutional visibility considering privacy issues. They can improve processes for trials, access to cross-institution data, and increase confidence in patient privacy. [27]

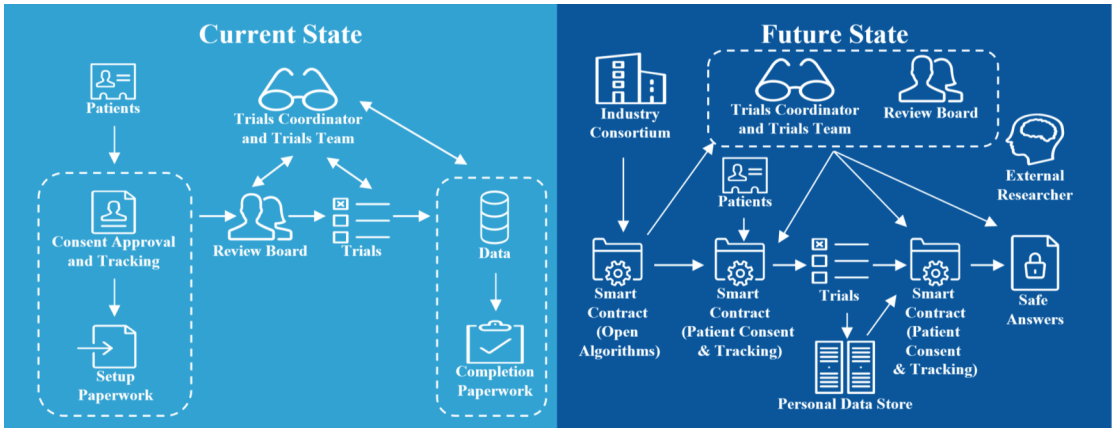

Figure 19: Clinical Trials

8.2.12 Research

Smart contracts can facilitate the sharing of important research data. They can facilitate the patient consent management process and aggregate data contribution and data sharing while protecting patient privacy. As stated previously, protection of data confidentiality is an important issue in this case too. [27]

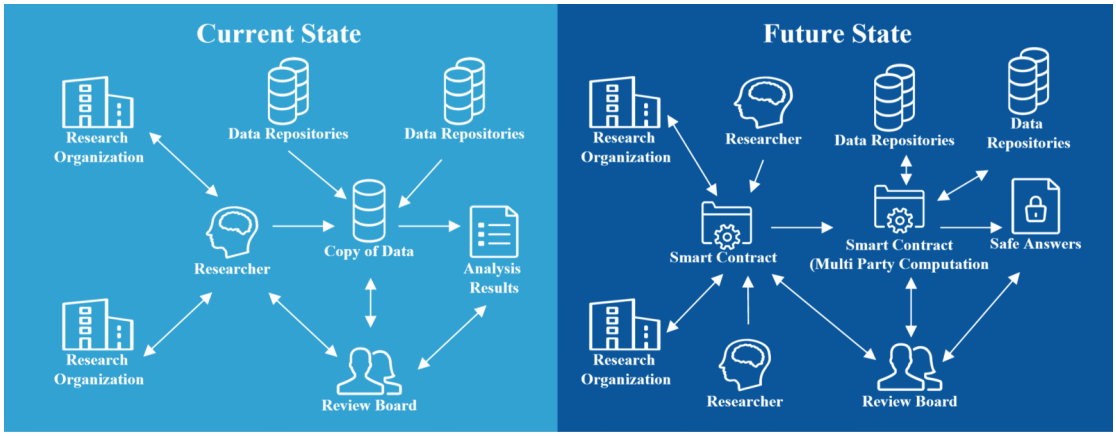

Figure 20: Research

9 Deployment of a smart contract system in an IoT environment

9.1 Environment of the Application

9.1.1 Hardware

The hardware used to develop the system is:

- a laptop and
- a Raspberry PI 3 (RPi).

Two separate miners will run on the same laptop and the RPi is intended to act as a node connected to our private ethereum blockchain. Due to hardware limitations (CPU, memory), the RPi will not be able to mine ethers (gas in Ethereum) or to build new blocks of transactions.

9.1.2 Software

The operating system of the laptop is Windows 10 but we use Ubuntu Linux 16.04.01 LTS (VM in Virtual Box) and the steps to build the ethereum blockchain and the miners are described below. As for the RPi we will install an image of Raspbian provided by the project EthRaspbian. The image includes both geth and parity but we will use the first one deactivating the other.

9.1.3 Goal

The goal of this process is to set up a private Ethereum blockchain composed of a computer (miners) and one Raspberry PI 3 device (node). The objective is to develop and test the use of a simple smart contract. We will integrate the RPi with a smart contract that will be used to check if a user has enough tokens to use a service. The necessary steps to successfully deploy this system are described below:

9.2 Configuration and setup

9.2.1 Installing an Ethereum node on the Raspberry PI (RPi)

We will describe the steps to transform the Raspberry Pi 3 (RPi) into an Ethereum node. The RPi is intended to act as a node connected to our private Ethereum blockchain.

Step 1- Installing image on RPi

- We download the image from http://www.ethraspbian.com/
- We burn the image using related software (in this case Etcher) in the SD card.

Step 2- Preparing the RPi to become a node

- We connect the RPi to a screen or we use **SSH** from another computer in order to interact with the device.
- We connect the RPi to the Internet

9.2.2 Installing an Ethereum node on the Computer

Step 1- Updating Ubuntu installation with the latest packages

sudo apt-get update

sudo apt-get -y upgrade

Step 2- Installing geth from PPA

The installation is done using a PPA:

sudo apt-get install software-properties-common

sudo add-apt-repository -y ppa:ethereum/ethereum

sudo apt-get update

sudo apt-get install ethereum

Note:

At this stage, Ethereum is installed on the computer and it is able to synchronize with the live chain (mainnet) if we run the geth command.

The blockchain (chaindata) is located in: **~/.ethereum**

The accounts will be stored in a wallet located in this folder for Linux: **~/.ethereum/keystore**

9.2.3 Setting up a private chain and the miners

Our private chain needs miners in order to validate and propagate blocks of transactions within the blockchain. Miners will also be used to generate ether to pay

```
{
"nonce": "0x0000000000000042",
"mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000"
,
"difficulty": "0x400",
"alloc": {},
"coinbase": "0x0000000000000000000000000000000000000000",
"timestamp": "0x00",
"parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000"
,
"extraData": "0x436861696e536b696c6c732047656e6573697320426c6f636b",
"gasLimit": "0xffffffff",
"config": {
"chainId": 4253647586,
"homesteadBlock": 0,
"eip155Block": 0,
"eip158Block": 0
}
}
```

for the gas required to process transactions on the Ethereum blockchain. After completing this step, each miner should be running its own version of the private blockchain and the transactions will not be distributed within the same private blockchain.

The requirements for each node to join the same private blockchain are the following:

- Each node will use a separate data directory to store the database and the wallet.
- Each node must initialize a blockchain using the same genesis file.
- Each node must join the same network id different from the one reserved by Ethereum.
- The port numbers must be different for different nodes installed on the same computer.

Step 1 – Creating the datadir folders

mkdir -p ~/blockchain/miner1

mkdir -p ~/blockchain/miner2

Step 2 – Creating the Genesis file

We create the genesis file copied in both folders (miner1 and miner2)

Among the parameters, we have:

- **difficulty:** if the value is low, the transactions will be quickly processed within our private blockchain.
- **gasLimit:** define the limit of Gas expenditure per block. The gasLimit is set to the maximum to avoid being limited to our tests.

Step 3 – Initializing the private blockchain

Step 3.1 – Initialize miners

We initialize the blockchain by executing the following command inside the blockchain folder

geth --datadir miner1 init genesis.json

geth --datadir miner2 init genesis.json

After that, **miner1** and **miner2** folders, include the subfolders:

- **geth**: contains the database of the private blockchain (chaindata).
- **keystore**: location of the wallet used to store

Step 4 – Creating accounts for miners

We create the default account that will be used to run the node. This account will receive all ethers created by the miner in the private blockchain. These ethers will serve to test by paying the gas required to process each transaction. To create the default account for the miner 1, we type the following command.

geth --datadir miner1 account new

The system will prompt for password

For miner 2 we type:

geth --datadir miner2 account new

Step 5 – Preparing the miners

To prepare the miners we will create two files for each miner in order to make things easier.

Step 5.1 – Miner 1: setup

Password file

For miner 1 we create a file named password.sec in the miner 1 folder where we type in plaintext the password we entered when we created the account in the previous step.

Initialization script

We create a script to initialize each miner due to the length of the needed parameters and we will save it in a file named startminer1.sh in the miner 1 folder.

The content of startminer1.sh file:

geth --identity "miner1" --networkid 4253647586 --datadir "~/blockchain/miner1" --nodiscover --mine --rpc --rpcport "8042" --port "30303" --unlock 0 --password ~/blockchain /miner1/password.sec --ipcpath "~/blockchain/miner1/geth.ipc"

The meaning of the main parameters is the following:

- **identity**: name of our node
- **networkid**: this network identifier is an arbitrary value that will be used to pair all nodes of the same network. This value must be different from 0 to 3 (already used by the live chains)
- **datadir**: folder where our private blockchain stores its data
- **rpc and rpcport**: enabling HTTP-RPC server and giving its listening port number
- **port**: network listening port number, on which nodes connect to one another to spread new transactions and blocks
- **nodiscover**: disable the discovery mechanism (we will pair our nodes later)
- **mine**: mine ethers and transactions
- **unlock**: id of the default account
- **password**: path to the file containing the password of the default account
- **ipcpath**: path where to store the file for ipc socket/pipe

Step 5.2 – Miner 2: setup

Password file

For miner 2 we create a file named password.sec in the miner 2 folder where we type in plaintext the password we entered when we created the account in the previous step.

Initialization script

Because the geth command is long we will save it in a file named startminer2.sh in the miner 2 folder. Additionally we will change the port from 30303 to 30304 and the rpc port from 8042 to 8043 as the miners must have different ports.

The content of startminer1.sh file:

geth --identity "miner2" --networkid 4253647586 --datadir "~/blockchain/miner2" --nodiscover --mine --rpc --rpcport "8043" --port "30304" --unlock 0 --password ~/blockchain/miner2/password.sec --ipcpath "~/blockchain/miner2/geth.ipc"

Step 5.3 – Miner 1: start

We make the script from the previous step executable and then we execute it.

chmod +x startminer1.sh

./startminer1

Step 5.4 – Miner 2: start

We make the script from the previous step executable and then we execute it.

chmod +x startminer2.sh

./startminer2

9.2.4 Pairing the miners

As a blockchain is a peer-to-peer network, our private blockchain miners must communicate with each for the execution of transactions. Furthermore, the discovery protocol will not work on a private blockchain. Therefore, we must configure each node to specify the identity and the location of its peers.

Step 1 Cleaning the miners

Step 1.1 – Stopping miners

press Control - C on the miner's console

Step 1.2 – Deleting the chaindata

rm -rf ~/blockchain/miner1/geth

rm -rf ~/ blockchain/miner2/geth

rm -rf ~/blockchain/miner1/keystore

rm -rf ~/blockchain/miner2/keystore

Step 1.3 – Initializing miners

geth --datadir ~/ blockchain/miner1 init genesis.json

geth --datadir ~/ blockchain/miner2 init genesis.json

Step 2 Getting IP address

we type the command: ifconfig

Step 3 Getting Node info from miners

Step 3.1 Getting Node info for miner 1

cd ~/blockchain/miner1

./startminer1.sh

Open a second terminal and start the Geth console:

geth attach --datadir miner1

> admin

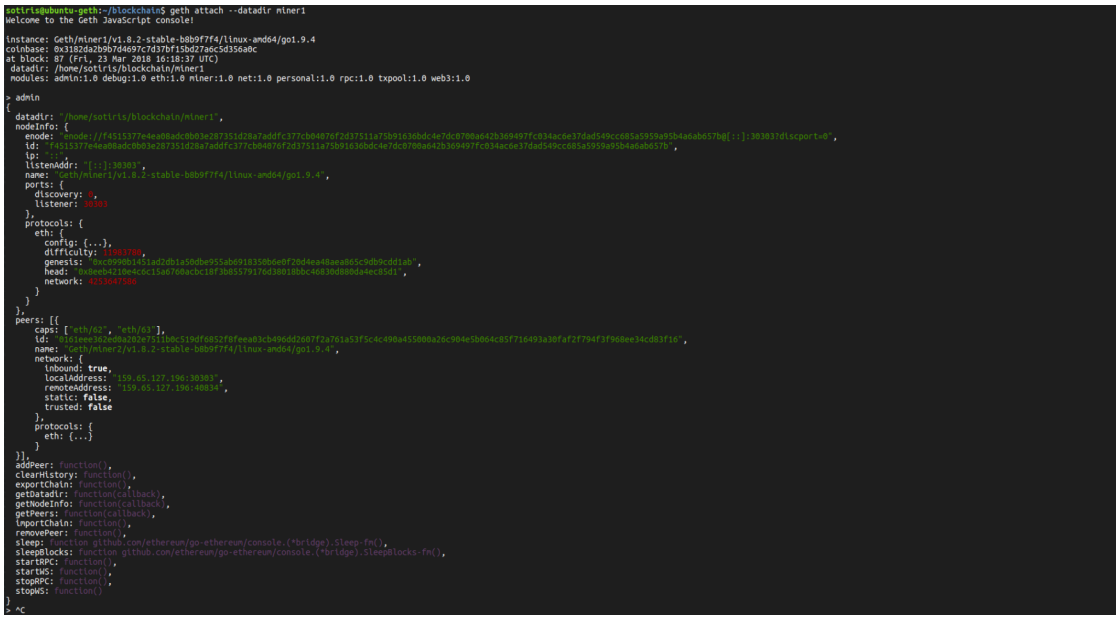

Figure 21: Node Info

We copy the enode address without discport

"enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@[::]:30303"

Step 3.2 Getting Node info from miner 2

cd ~/blockchain/miner2

./startminer2.sh

Open a second terminal and start the Geth console:

geth attach --datadir miner2

> admin

"enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@[::]:30304"

Step 4 – Pairing the nodes

We will define the permanent static nodes stored in a file called “static-nodes.json“. This file will contain the node information of our miners. Based on our environment, we will have the following content:

We replace [::] with the ip address we got in paragraph 9.2.4 Step 2

["enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@192.168.1:30303",

"enode://41be9d79ebe23b59f21cbaf5b584bec5760d448ff6f57ca65ada89c36e7a05f20d9cfdd091b81464b9c2f0601555c29c3d2d88c9b8ab39b05c0e505dc297ebb7@192.168.1:30304"]

Note: The file "static-nodes.json" must be stored under the following locations:

- ~/ blockchain/miner1
- ~/ blockchain/miner2

Step 5 – Restarting the miners

We stop and start the miners to ensure that they will properly reload the "static-nodes.json" file.

If we check the console of the miners, we should see a line mentioning the synchronization process ("Block synchronization started" and "Block became a side fork"):

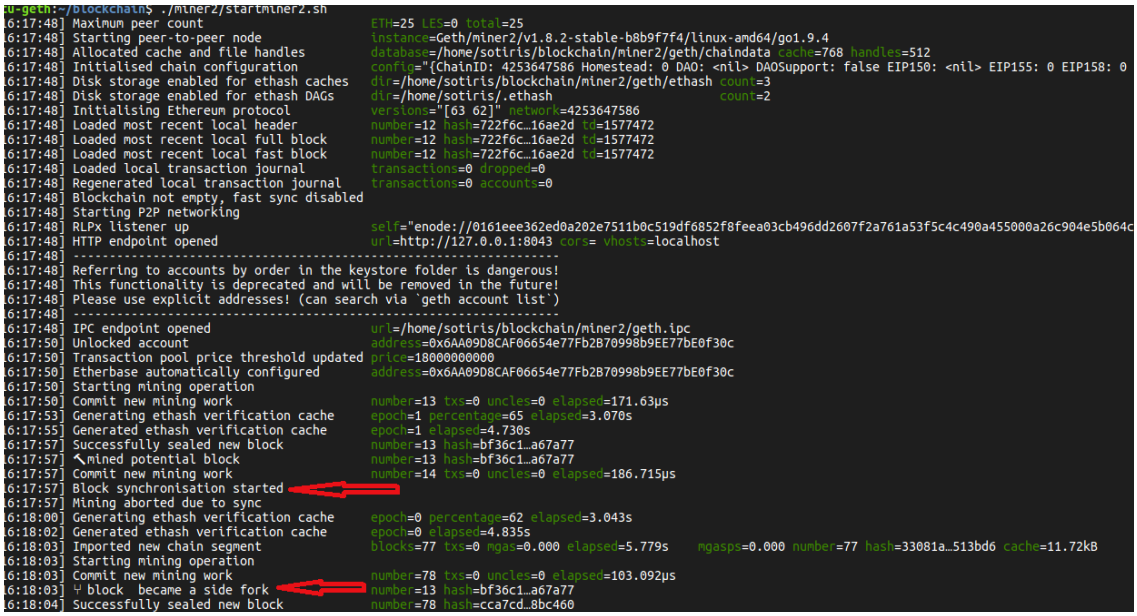

Figure 22: Block Synchronization

Step 6 – Checking the synchronization process

Step 6.1 – Checking from miner 1

geth attach --datadir miner1

> admin

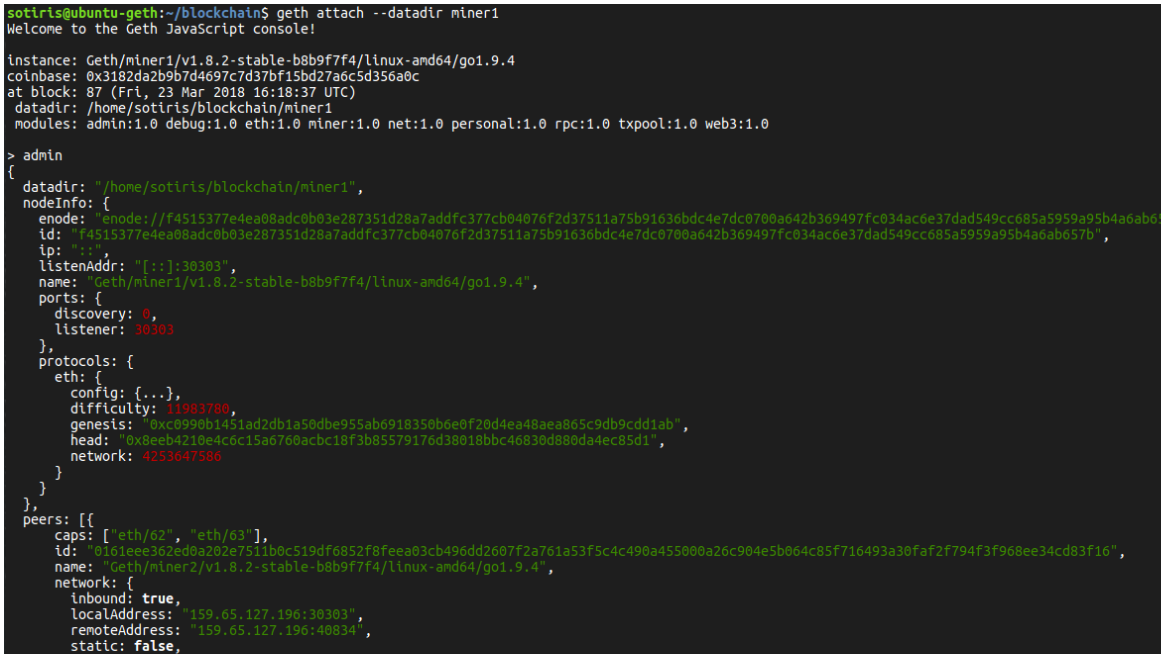

Figure 23: Peers Info

Step 6.2 – Checking from miner 2

geth --datadir miner2

> admin

Step 7 -Validating the synchronization

We can see from the onscreen information that our node is paired to miner 1 identified by its IP address and its port number.

9.2.5 Synchronizing the RPi node with miners

In the previous step we paired the miners and ensured that the private blockchain is properly synchronized. In this one, we will synchronize the RPi node with the miners and the process is the same as we did to setup and synchronize the miners before with minor changes.

Step 1 – Creating the datadir folder

mkdir -p ~/blockchain/node

Step 2 – Transfering the genesis file

In the computer console we type:

cd ~/blockchain

sftp pi@192.168.1.XX

sftp> cd blockchain

sftp> put genesis.json

sftp> exit

Step 3 – Initializing the node

cd ~/blockchain

geth --datadir ~/blockchain/node init genesis.json

Step 4 – Creating accounts

geth --datadir ~/blockchain/node account new (password requested)

Step 5 – Preparing the node

Initialization script

Storing the Geth command into an executable script called **startnode.sh** located here:~/blockchain/node, containing:

#!/bin/bash

geth --identity "node1" --fast --networkid 4253647586 --datadir ~/blockchain/node --nodiscover --rpc --rpcport "8042" --port "30303" --unlock 0 --password "~/blockchain/node/password.sec" –ipcpath ~/.ethereum/geth.ipc

Password file

The password file must be in the datadir folder of the node, and this file should just contain the password of the default account in plaintext. We name the file **password.sec** and should be located at: ~/blockchain/node

Step 6– Starting the node

In Rpi‘s console we type the following:

cd ~/blockchain/node

chmod +x startnode.sh

./startnode.sh

Step 7 – JavaScript console

In Rpi‘s console we type the following:

geth attach --datadir node

Step 9 – Synchronizing the blockchain

Step 9.1 – Getting Node info

we continue in the js console typing:

> admin.nodeInfo.enode

["enode://c52b3349d899e1f8ea67c2d01df35c3a40532dec41174460b777bab020079e1a546313552b91d5f61adb86ed4e74dd80c0ced70c91d658ef0e4f05969a1cf78e@192.168.1.31:30303",

Step 9.2 – Updating the file “static-nodes.json”

Based on our environment, we will have the following content (adjust the values according to the environment):

["enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@192.168.1.39:30303",
"enode://41be9d79ebe23b59f21cbaf5b584bec5760d448ff6f57ca65ada89c36e7a05f20d9cfdd091b81464b9c2f0601555c29c3d2d88c9b8ab39b05c0e505dc297ebb7@192.168.1.39:30304",
"enode://c52b3349d899e1f8ea67c2d01df35c3a40532dec41174460b777bab020079e1a546313552b91d5f61adb86ed4e74dd80c0ced70c91d658ef0e4f05969a1cf78e@192.168.1.31:30303"]

The first two entries are related to miner 1 and miner 2. The last row identifies the node deployed on the RPi (with its IP address and port number).

Note:

The IP addresses should be up-to-date as they tend to change on a local network. The new version of "static-nodes.json" must be stored under the following locations:

- [miner 1] ~/blockchain/miner1
- [miner 2] ~/ blockchain /miner2
- [RPi] ~/ blockchain/node

Step 9.3 – Restarting the blockchain

Stop and start each node of the blockchain:

- miner 1
- miner 2
- RPi

Step 10 – Checking the synchronization process

In the computer console:

geth attach --datadir miner1

> admin.peers

(we must see two peers miner2 and the RPi)

9.2.6 Deployment of the Smart Contract

Step 1 - Installing Mist

We type https://github.com/ethereum/mist/releases and we choose the appropriate Ethereum Wallet version.

Step 2 - Installing Ethereum Wallet (Mist)

After downloading the file and unziping it, we launch the application.

Step 3- Installing nmp

cd ~

curl -sL https://deb.nodesource.com/setup_8.x -o nodesource_setup.sh

sudo bash nodesource_setup.sh

sudo apt-get install nodejs

Step 4- Installing Truffle (Ethereum Development Framework)

The first step is to install a development framework for Ethereum, such as Truffle, for development and deployment of smart contracts.

npm install -g truffle

Step 4.1 – Creating the project

On the laptop we create a folder to host the project. Then, we initiate the Ethereum development framework (Truffle) for the specified project.

Step 4.2 – Creating the contract

In the "contracts" directory, we create a file named "SmartToken.sol" and paste the following code:

```
pragma solidity ^0.4.0;

contract SmartToken {
mapping(address => uint) tokens;
event OnValueChanged(address indexed _from, uint _value);

function depositToken(address recipient, uint value) returns (bool success) {
tokens[recipient] += value;
OnValueChanged(recipient, tokens[recipient]);
return true;
}
function withdrawToken(address recipient, uint value) returns (bool success) {
if (int(tokens[recipient] - value) < 0) {
tokens[recipient] = 0;
} else {
tokens[recipient] -= value;
}
OnValueChanged(recipient, tokens[recipient]);
return true;
}

function getTokens(address recipient) constant returns (uint value) {
return tokens[recipient];
}
```

This contract has three functions analyzed below:

- **depositToken**: add some tokens to a specific address
- **withdrawToken**: withdraw some token from a specific address
- **getTokens**: retrieve the number of tokens available for a specific address

Step 5 – Preparing for deployment

Certain adjustments will be required in the deployment file and network settings before the deployment of a smart contract as descpibed next:

Step 5.1 – Adapting deployment file

We replace the content of "migrations/1_initial_migration.js" with the following content in order to deploy our "SmartToken" Smart Contract:

```
var SmartToken = artifacts.require("./SmartToken.sol");
module.exports = function(deployer) {
   deployer.deploy(SmartToken);
};
```

Step 5.2 – Adapting network settings

The file named "truffle.js" contains network settings used to identify the deployment platform. We will deploy the smart contract onto one of the miners. We change the file to adjust the port number to fit our eonvironment:

```
module.exports = { networks: { development: { host: "localhost", port: 8042, network_id: "*" // Match any network id } } };
```

Step 6 – Deploying the contract

The smart contract's deployment process starts with the initialization of miners to ensure that the smart contract will be mined and deployed on the private blockchain. Before proceeding, start the miners to ensure that the smart contract will be properly mined and deployed on the private blockchain.

Step 6.1 – Starting the miners in 2 different tabs

```
~/blockchain/miner1/startminer1.sh
~/ blockchain /miner2/startminer2.sh
```

Step 6.2 – Compiling and deploying the contract

```
cd ~/blockchain/Projects/SmartToken
truffle compile
truffle migrate --reset
```

The SmartToken contract is deployed onto our private Ethereum chain and it is ready to receive calls.

Step 7 – Interacting with the contract

We can interact with the previously deployed smart contract through the Geth console, the Mist browser or a client application. Furthermore, for the testing of the contract, the default address created on the RPi should be used.

Step 7.1 – Identifying the contract

Before using Mist, we need two elements about the deployed contract:

- its address
- its ABI (Application Binary Interface)

This information can be retrieved through the Truffle console in this way:

truffle console

truffle(development)> **SmartToken.address**

'0x33…'

truffle(development)> JSON.stringify(SmartToken.abi)

Step 7.2 – Watching the contract on Mist

Starting Mist, it detects an IPC file in the default location is being used, it connects to the private network.

We use the following command to start Mist from a different path than the default:

Ethereumwallet network 2435465768 --rpc ~/blockchain/miner1/geth.ipc

Once the user interface appears, we click the Ethereum Wallet tab on the left, and click the CONTRACTS button:

Select “WATCH CONTRACT” and fill the form:

- Give a name for the contract, like “SmartToken”
- Enter the contract address retrieved from the Truffle console (SmartToken.address)
- Enter the ABI (without enclosing quotes) retrieved from the Truffle console (JSON.stringify(SmartToken.abi))

Step 7.3 – Testing the contract

To test the contract we pick the default address created on the RPi.

(eth.coinbase) is:“0x33…..”

To check the contract,get the number of tokens of this contract:

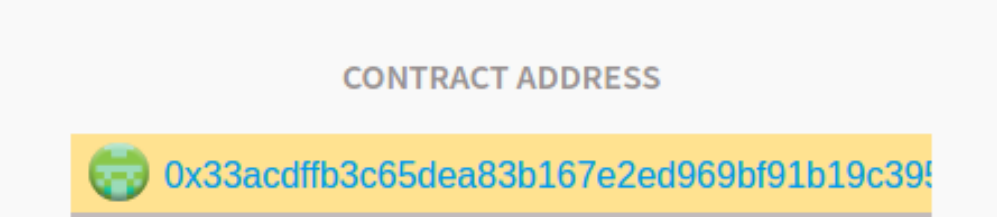

Figure 24: Contract Address

The function "getTokens" is a constant. We can use it without paying any fees. Now, we deposit some tokens to this address. We choose the account that will execute the function and pay the fees:

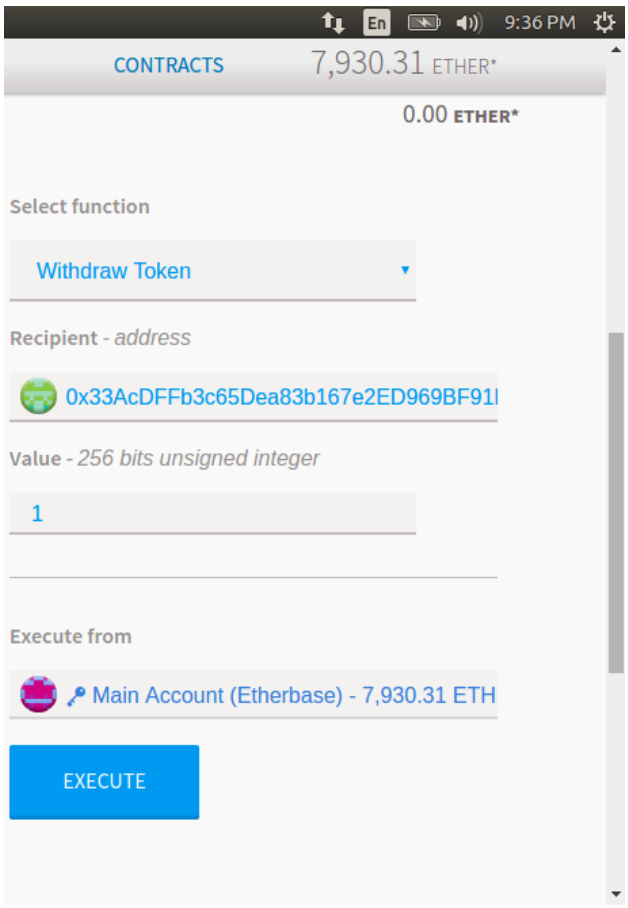

Figure 25: Withdrawing Tokens

When the block is mined, we should see that the value has changed and then, we proceed with the withdraw function to ensure that the contract works as expected.

We can also watch the events triggered by the contract:

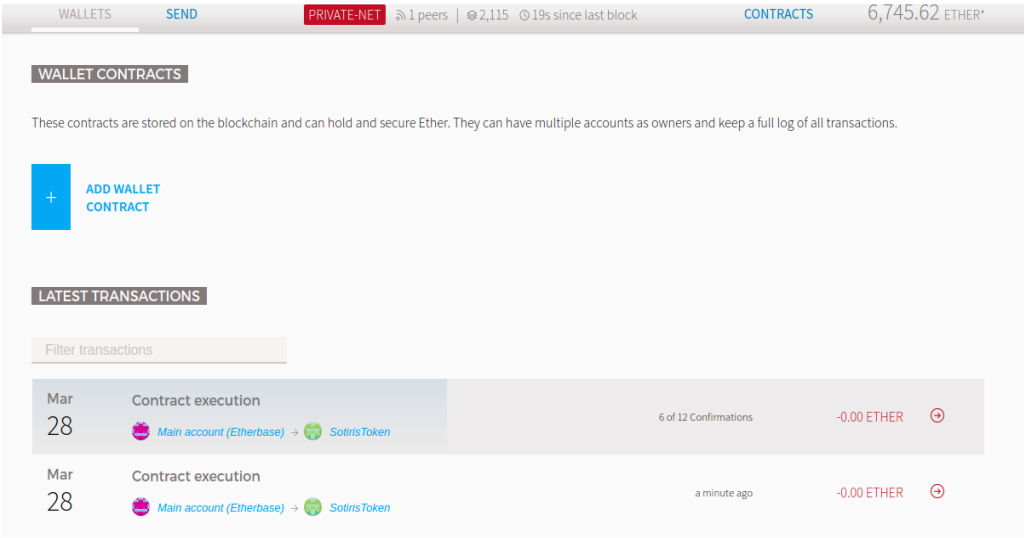

Figure 26: Event viewing

10 Concluding Remarks

10.1 Answers to Research Questions

In the introduction we presented our research questions and in this section we provide a synopsis of the answers given through the analysis of the thesis:

- **RQ1 –What is the current status and the predictions concerning blockchain technology, smart contracts and IoT?**

  We investigated the history of blockchain and the main issue that tries to solve, explaining its different types (public, private and hybrid). We presented the technologies enabling blockchain and the way it works. As far as smart contracts are concerned, we provided an overview of the benefits, challenges and their main components including a high level cost-benefit analysis for developing and deploying a smart contract. In both, blockchain and smart contracts, we tried to identify the major security issues and present twelve business cases of how smart contracts and blockchain can disrupt traditional business models and transform them due to their special characteristics. Furthermore, we presented information from market researches which point out that there will be an increase in the use of IoT, the following years.

- **RQ2 – What are the similarities and the differences among blockchain platforms that enable smart contracts?**

  We identified the platforms we intended to evaluate and we provided a description of their main characteristics. These characteristics were defined based on the information available from a variety of sources. However, a detailed quantitative evaluation was not possible due to the lack of research material and examination of real-life applications. There are a few frameworks that are developed on a theoretical level, because of the lack of applications and lack of standardization in technologies.

- **RQ3 – What is the level of security in smart contracts and what can we do to improve it?**

  We carried out a high-level security assessment based on research papers that investigate the vulnerabilities, the attacks and the countermeasures and how these are related. The assessment revealed several security flaws in the different

components of ethereum-based smart contracts. The tools to improve security are at a primary stage and require improvement considering the amount of money involved in most cases. Finally, we deployed a smart contract system based on Ethereum using IoT in order to explore the process, to investigate the difficulties and demonstrate the potential of the applications.

10.2 Conclusion

The current status of blockchain, smart contracts and IoT reveals that this technology and its applications have a long way to go until they reach a maturity level and become a daily tool. The lack of standardization in blockchain platforms and the several security and legal issues concerning smart contracts that remain to be solved, are going to be critical for the evolution of their adoption cycle.

Throughout the literature, there are only a few frameworks that can be used to assess blockchain platforms especially for the public ones. The results of the comparison among the blockchain platforms have pointed out the main differences and similarities of these platforms. Consensus mechanism is a key element of the blockchain structure that changes properties of the blockchain. By combining blockchain, smart contracts and IoT we can have several possible applications but it remains to be seen which ones will be adopted from the business world and the end-users. The security assessment of smart contracts presented the vulnerabilities and the attacks against them, proposing countermeasures. Finally, the application we built proved the ease and flexibility of smart contracts in an IoT environment.

References

[1] M. Swan, *Blockchain: blueprint for a new economy*. Beijing: O'Reilly Media, Inc., 1st ed., 2015.

[2] "*The trust machine - the technology behind bitcoin could transform how the economy works*," Oct. 2015.https://www.economist.com/news/leaders/21677198-technology-behind-bitcoin-could-transform-how-economy-works-trust-machine.

[3] G. A. C. on the Future of Software & Society, "*Deep shift: Technology tipping points and societal impact,*" Sept. 2015. Survey Report.

[4] O. Williams-Grut, "Goldman Sachs: 'the blockchain can change...well everything'," Dec. 2015. http://www.businessinsider.in/GOLDMAN-SACHS-The-Blockchain-can-change-well-everything/articleshow/50018480.cms.

[5] S. Haber and W. S. Stornetta, "*How to time-stamp a digital document,*" *Journal of Cryptology*, vol.3, pp. 99–111, Jan 1991.

[6] D. Bayer, S. Haber, andW. S. Stornetta, *Improving the Efficiency and Reliability of Digital Time-Stamping*, pp. 329–334. New York, NY: Springer New York, 1993.

[7] S. Nakamoto, "*Bitcoin: A peer-to-peer electronic cash system*." Electronic, 2008. https://bitcoin.org/bitcoin.pdf.

[8] J. BRITO and A. CASTILLO, "*Bitcoin - a primer for policymakers*." Electronic, Mercatus Center, George Mason University, USA, 2013.

[9] A. Heikkila, "*The blockchain and the byzantine generals problem*," Mar. 2017. http://techblog.cosmobc.com/2017/03/16/blockchain-byzantine-generals-problem/.

[10] P. Francis, "*Blockchain, the byzantine generals problem, and the future of identity management*," Aug. 2016. https://medium.com/@philfrancis77/blockchain-the-byzantine-general-problem-and-the-future-of-identity-management-6b50a2eb815d.

[11] V. Buterin, "*Slasher: A punitive proof-of-stake algorithm.*" Etherum Blog, Jan. 2014. https://blog.ethereum.org.

[11A] "*04/Blockchain Technology: preparing for change*", Accenture 2015, https://www.accenture.com/t20160608T052656Z__w__/us-en/_acnmedia/PDF-5/Accenture-2016-Top-10-Challenges-04-Blockchain-Technology.pdfla=en

[12] D. Kraft, "*Difficulty control for blockchain-based consensus systems*," *Springer Peer-to-Peer*
Networking and Applications, vol. 9, pp. 397–413, Mar. 2016.

[13] L. Backlund, *A technical overview of distributed ledger technologies in the Nordic capital market*. PhD thesis, Uppsala University, 2016.

[14] S. Carlsson, J. & Huang, *Blockchain Technology in the Swedish Fund Market*. PhD thesis, Royal Institute of Technology, Stockholm, 2016.

[15] V. Buterin, "Slasher: A punitive proof-of-stake algorithm." Blog, Jan. 2014. https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/.

[16] T. N. Courtious, "*On the longest chain rule and programmed self-destruction of crypto currencies*," tech. rep., Cornell University Library, 2014.

[17] P. Vasin, "*Blackcoins proof-of-stake protocol v2*," 2014. https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

[18] V. Buterin, "*On public and private blockchains.*" Etherum Blog, Sept. 2015. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/.

[19] J. Manning, "*Proof-of-work vs. proof-of-stake explained,*" 2016. https://www.ethnews.com/proof-of-work-vs-proof-of-stake-explained.

[20] N. Houy, "*It will cost you nothing to "kill" a proof-of-stake crypto-currency*," 2014. https://halshs.archives-ouvertes.fr/file/index/docid/945053/filename/1404.pdf.

[21] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, Huaimin Wang, "Blockchain Challenges and Opportunities: A Survey" 2016. https://www.researchgate.net/profile/Hong-Ning_Dai/publication/319058582_Blockchain_Challenges_and_Opportunities_A_Survey/links/59d86d50a6fdcc2aad0a2f2a/Blockchain-Challenges-and-Opportunities-A-Survey.pdf .

[21A] https://sawtooth.hyperledger.org/docs/core/nightly/0-8/introduction.html#proof-of-elapsed-time-poet

[21B] Chen L., Xu L., Shah N., Gao Z., Lu Y., Shi W. (2017) "On Security Analysis of Proof-of-Elapsed-Time (PoET)". In: Spirakis P., Tsigas P. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2017. Lecture Notes in Computer Science, vol 10616. Springer, Cham

[22] J. Kehrli, "*Blockchain 2.0 - from bitcoin transactions to smart contract applications*," Nov. 2016.

[23] M. Bartoletti and L. Pompianu, "*An empirical analysis of smart contracts: platforms, applications,*
and design patterns," 2017. http://fc17.ifca.ai/wtsc/An%20empirical%20analysis%20of%20smart%20contracts%20-%20platforms,%20applications,%20and%20design%20patterns.pdf.

[23A] Gavin Wood, "*Ethereum:a secure decentralised generalised transaction ledger*" Eip-150 revision

[24]https://hackernoon.com/costs-of-a-real-world-ethereum-contract-2033511b3214

[24A] "*Getting from Hype to Reality*", CapGemini, 2017. https://www.capgemini.com/consulting-de/wp-content/uploads/sites/32/2017/08/smart_contracts_paper_long_0.pdf

[25] International Telecommunication Union, "*Measuring the Information Society Report*," International Telecommunication Union (ITU), Report, 2015.

[26] "Gartner Says 6.4 Billion Connected "Things" *Will Be in Use in 2016, Up 30 Percent From 2015*," http://www.gartner.com/newsroom/id/3165317.

[27] "*Smart Contracts: 12 Use Cases for Business & Beyond A Technology, Legal & Regulatory Introduction*", Smart Contracts Alliance - In collaboration with Deloitte, http://www.the-blockchain.com/docs/Smart%20Contracts%20-%2012%20Use%20Cases%20for%20Business%20and%20Beyond%20-%20Chamber%20of%20Digital%20Commerce.pdf

[28] MarketsandMarkets, *"Blockchain Market by Provider, Application (Payments, Exchanges, Smart Contracts, Documentation, Digital Identity, Supply Chain Management, and GRC Management), Organization Size, Industry Vertical, and Region - Global Forecast to 2022",* https://www.marketsandmarkets.com/PressReleases/blockchain-technology.asp

[29]https://www.businesswire.com/news/home/20171117005320/en/Global-Blockchain-Market-Forecasts-2017-2022--

[30] Tractica, *"Blockchain for Enterprise Applications Market to Reach $19.9 Billion by 2025",* 2016. https://www.tractica.com/newsroom/press-releases/blockchain-for-enterprise-applications-market-to-reach-19-9-billion-by-2025/

[31] Autonomous Research, *"2030 Projection of Blockchain Technology Market"* https://next.autonomous.com/insights/2030-projection-of-blockchain-technology-market

[32] Market Research Future, "Smart Contracts Market Research Report – Global Forecast to 2023"https://www.marketresearchfuture.com/reports/smart-contracts-market-4588

[33] Market Research Future*, "Smart Contracts Market Research Report – Global Forecast to 2023"*, Feb. 2018. https://www.marketresearchfuture.com/reports/smart-contracts-market-4588

[34] Ericsson, Internet of Things forecast, https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

[35] C. Cachin, "*Architecture of the hyperledger blockchain fabric*," in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, pp. 1–4, 2016.

[36] J. R. Douceur, "*The sybil attack*," in International Workshop on Peer-to-Peer Systems, pp. 251–260, 2002.

[37] M. Samaniego and R. Deters, "*Hosting virtual iot resources on edge-hosts with blockchain*," in *IEEE CIT*, pp. 116–119, 2016.

[38] B. Smith and K. Christidis, "*IBM blockchain: An enterprise deployment of a distributed consensus based transaction log*," in Fourth International IBM Cloud Academy Conference, pp. 140–143, 2016.

[39] G. Wood, "*Ethereum: A secure decentralised generalised transaction ledger*," 2014. http://gavwood.com/paper.pdf.

[40] C. Natoli and V. Gramoli, "*The blockchain anomaly*," in *IEEE* 15th International Symposium on Network Computing and Applications (NCA), pp. 310–317, 2016.

[41] K. Christidis and M. Devetsikiotis, "*Blockchains and smart contracts for the internet of*

things," IEEE Access, vol. 4, no. 1, pp. 2292–2303, 2016.

[42] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor, "*Making Smart Contracts Smarter*", https://www.comp.nus.edu.sg/~loiluu/papers/oyente.pdf

[43] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli, "*A survey of attacks on Ethereum smart contracts*".

[44] Information Economy Division Commerce and Information Policy Bureau, "*Evaluation Forms for BlockchainBased System ver. 1.0"*, 12 April 2017. http://www.meti.go.jp/english/press/2017/pdf/0329_004a.pdf

[45] lharby, M. & van Moorsel, A. 2017. *"Blockchain-based smart contracts: A systematic mapping study"*. arXiv preprint arXiv:1710.06372 https://arxiv.org/ftp/arxiv/papers/1710/1710.06372.pdf

.