KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Feasibility of blockchain application as medium for collaborative systems or databases

How to replace a previously thought indispensable middleman with technology

Pelle Jacobs
r0364018

Thesis submitted to obtain the degree of

Masters in Information Systems Engineering
Majoring in Data Science

Promotor: Prof. Dr. Jochen De Weerdt
Assistant: Vytautas Karalevicius

Academic year: 2016-2017

Contents

Preface

Leuven, August 16, 2017.

Chapter 1

Introduction

THIS TEXT SHOULD BE UPDATED WHILE WRITING CHAPTERS

Hitting a market capitalization of $13.8 Billion in December 2013, it was clear that a once fringe cryptocurrency called "Bitcoin" had hit mainstream. By building on decades of research in decentralized currencies and cryptography, Bitcoin is the first successful implementation of a truly decentralized currency. Most of this success has been attributed to an innovation called "Blockchain". However, this success has opened up a deeper concept: the power of technology to replace a previously thought middleman.

Now, the question is: "How can this idea to replace a middleman with technology be applied to collaborative, distributed databases and what is the role of blockchain is this concept?" The next three chapters provide an answer to this question:

In the next chapter, three main concepts that are essential aspects of this question are discussed. First public key encryption is explained, as it is the basis for modern encryption and digital signatures. Next, the concept of the blockchain, its properties, and some well-known implementation are examined. Finally, the concept the thesis delves into distributed databases, the difference between distributed, decentralized and centralized databases and some non-blockchain examples of distributed databases and networks.

The third chapter explains how this thesis tries to solve the research question in the final chapter. Collaborative databases will be split up into two groups: centralized databases, and distributed databases. Both groups have a middleman that could be replaced, but the role of this middleman differs greatly between both groups.

The final chapter investigates the viability of specific proposals trying to remove the middleman in a centralized or a distributed database. Every section discusses advantages and disadvantages, features, and constraints of the proposal.

Chapter 2

Prior research into relevant concepts

2.1 Public key cryptography

Public key cryptography is the backbone of most distributed systems. It provides both a way to encrypt a message and to confirm the source of a message, without the need to agree upon a shared key. The most common implementation of this idea is the RSA encryption algorithm, named after inventors Rivest, Shamir and Adleman [1].

2.1.1 Basics

If two parties wanted to safely exchanges messages before public key cryptography, they had to agree on a common code. This code is referred to as a cipher. One of the more famous of such encryption systems is the Ceasar cipher [2]. With a Ceasar cipher, every character of the message is offset by an agreed upon number. Yet, if a third party manages to intercept the transmission of the cipher itself, all encrypted messages can be decoded. Thus, a trusted third party network is required to safely exchange the cipher.

Public key cryptography uses a pair of hashes instead. These hashes are often referred to as the keys. The keys are algorithmically generated so that they are cryptographically connected. This means that a message encrypted with one key can only be decrypted with the other key and vice versa [3].

One of these keys can be broadcasted. As a result, everyone knows that this key is connected to the owner's identity. Therefore, it is referred to as the public key. The other key will be held privately. It is very important that nobody except the owner

knows the content of this key, as it will be used to prove the owner's identity. It is therefore referred to as the private key.

Because no cipher has to be exchanged, there is no need for the trusted middleman to exchange the cipher. This eliminates a key weakness of cipher encryption.

2.1.2 Uses: encryption and source validation

A first use case of public key cryptography is encryption. If a person called Alice wants to send a message to Bob, she encrypts her message with Bob's public key. As a result, only Bob can decrypt this message as only Bob knows the corresponding private key. The message can now be sent over any insecure network, such as the internet, email or even carrier pigeon, without any danger of leaking its contents.

Besides encryption, public key cryptography can also be used for source validation. If Alice wants to broadcast a message to the world, she can encrypt the message with her private key. As the encrypted message can only be decrypted with Alice's public key, everyone knows that only Alice could have encrypted it. This case is often referred to as a digital signature, as Alice signed the message with her private key.

For full security, a secure message is often first encrypted with the private key of the sender and then with the public key of the recipient. Now only the recipient can decrypt the message and can then confirm the origin of the message. This full process is modeled in figure 2.1.

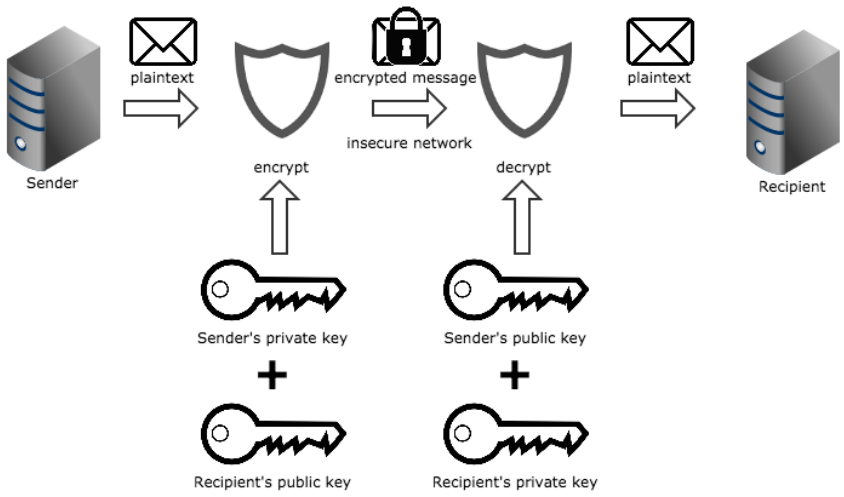

Figure 2.1

2.2 Blockchain

2.2.1 Definition of a blockchain

In the last few years, the blockchain as a concept has been given a wide range of meanings and definitions. Some explain the blockchain as a history of events, some focus on blockchain as a bitcoin transaction ledger and some talk about the open decentralized database aspect of blockchains [4]. Therefore, this section starts by defining this concept, to avoid any misunderstanding or confusion further on.

This thesis will use the blockchain definition as defined by Antonopoulos (2014) [5]: "The blockchain data structure is an ordered, back-linked list of blocks and transactions." (p. 159). There are several aspects in this definition that need further explanation.

Back-linked lists as data structure

First of all, a data structure is a "specialized format for organizing and storing data" (Rouse 2006 [6]). There are several types of data structures such as arrays, lists, graphs, and trees.

Next, a blockchain is a specialized version of an "ordered, back-linked list". A list is a data structure that combines a number of ordered values (Abelson 1996 [7]). A blockchain will use back-linking to preserve the order of the values. This means that all values have a reference to the previous value in the list. For a blockchain, this means that every block in the blockchain contains an identification of the previous block.

Transactions

The second aspect of the definition mentions transactions. Transactions are the data objects that store information on the blockchain. There are two types of transactions: monetary transactions and non-monetary transactions.

In a monetary transaction, there is a transfer of value between participants, eg. an amount of bitcoin. The simplest monetary transaction transfers value from one account to another. However, several blockchains allow custom conditions to be scripted into the transactions. These conditions can be used to imitate for example a crop insurance contract. In this situation, the transaction would only execute if at the recipient experiences a drought. Transactions with such a custom condition are often referred to as "smart contracts" [20].

On the other side, there are non-monetary transactions that store data onto the blockchain.

An example is the storage of the fingerprint of a document on the blockchain.

Transactions are the raison d'être of the blockchain. Blockchains are designed to make sure that transactions can be validated, propagated over the network and added to a distributed ledger.

Blocks

Every block in the blockchain includes several transactions. The blocks are wrappers for the transactions, so these transactions can be included into the blockchain. The blocks are the ordered values of the back-linked list addressed in the beginning of this section. Each block links to the previous block in the list. Blocks are ordered chronologically. Transactions stored in a previous block are executed before the transactions in a later block and all transactions in the same block are executed concurrently.

To conclude, the blockchain is a list of blocks, with each block containing several transactions.

2.2.2 Properties of a blockchain

Securing the immutability of block headers

As the Bitfury Group explains [8], every implementation of a blockchain must secure itself against possible attacks on its immutability. Take for example a blockchain powering a cryptocurrency. If the immutability of this blockchain was not ensured, an attack could spend a coin and afterward transmit a version of the blockchain without this transaction. As a result, the attacker changed the blockchain to a new reality in which this coin has not been spent, allowing the attacker to spend this coin again.

To ensure immutability, blockchains make use of a consensus mechanism. This is an algorithm the different nodes in the network use to agree upon the newest state of the blockchain. Every consensus mechanism is designed in such a way that no malicious entity can break the system unless this entity has some sort of majority in the network. How this majority is defined, depends on the consensus mechanism.

Proof of work To suggest a new block for a blockchain with a proof of work consensus mechanism, an entity needs to solve a random computational problem. This problem is designed as such that this entity has a chance of p% to solve this problem if he controls p% of the computing power currently in the network. Such an entity is called a miner. If the miner solves the problem, the miner will be rewarded with the relevant cryptocurrency.

Miners are incentivized to provide as much computing power as possible until it is no longer economically interesting considering the reward. As the Bitfury Group states [8]: "[the] security of the network is supported by physically scarce resources: specialized hardware needed to run computations, and electricity spent to power the hardware." (p.2)

If a malicious entity wants to manipulate a proof of work blockchain, this entity would need to control 50% of the computing power on the network. Therefore, the security of a proof of work blockchain depends on two main factors. First is the total computing power in the network, also called the hash rate. A higher the hash rate means it is more difficult for a new entrant to suddenly take over the network. Second is the distribution of the computing power in the network. If the total computing power is concentrated between a couple of entities, they could effectively work together to achieve a majority share.

The main advantage of a proof of work system is its security, as it is very costly to attack. However, this comes at a steep ecological cost. A lot of electricity and computing hardware is wasted on useless calculations. Therefore, other consensus mechanisms have been suggested.

Proof of stake The difficulty to create a new valid block for a blockchain with a proof of stake consensus mechanism depends on the balance an entity holds of the relevant cryptocurrency. An entity with a higher balance will more easily find a valid block, without having to spend electricity and computing hardware required by the proof of work algorithm [8].

The main idea is that entities with a higher balance, are more invested in the currency and its success. If one entity would control more than 50% of the tokens available, it would be against his own interest to attack the blockchain as he would be hit the hardest.

The advantage of this system is the absence of wasteful spending. Instead of having to spend $1000 on specialized mining hardware, you can invest this money in the cryptocurrency itself with the same effect. However, there are several problems with the proof of stake mechanism. It is outside the scope of this paper to discuss these issues individually, but all result out of the fact that the "proof of stake consensus is not anchored in the physical world (cf. with hashing equipment in proof of work)." (Bitfury Group, 2015, p22).

The most valuable cryptocurrency using some variation of proof of stake mechanism is Bitshares. On July 20 2017, Bitshares had a market capitalization of $300 million [9]. This makes it the 14th most valuable cryptocurrency at the time of writing. The fact that the top 13 use a proof of work mechanism, shows how the market trusts the proof of work mechanism better. Furthermore, Bitshares' market capitalization was only

$20 million in April 2017 and reached a record of $1.2 billion on June 10, 2017. This illustrates further the extreme volatility of cryptocurrencies.

Other consensus mechanisms Several other consensus mechanisms have been proposed, besides proof of work and proof of stake. An example is proof of storage. Here, an entity can create new blocks depending on how many megabytes of storage he provides to the network [10]. However, none of these other consensus mechanisms have been implemented in successful blockchains. This makes it is very hard to evaluate their viability.

Access to the blockchain data

As explained by the Bitfury Group [11], there are three levels of access to a blockchain. Firstly, there is read access to the data stored on the blockchain. This is the lowest level of access to a blockchain. Secondly, on a higher level, there is access to propose new transactions to the blockchain. Finally, on the highest level, there is access to create new blocks of transactions and add these blocks to the blockchain.

Depending on the design of a blockchain, different entities can have different levels of access to this blockchain.

Read access and transaction submission: private and public blockchains The first two access levels are intertwined and will be discussed together.

As mentioned by the Bitfury Group [11], there are two options on these access levels. The first option is a public blockchain. This means that there are no restrictions a reading the data on the blockchain or on submitting transactions. The other option is a private blockchain. Only a predefined list of entities can directly access the data or submit transactions.

A private blockchain might seem more secure because of the controlled access. Yet, the Bitfury Group [11] shows that several advantages of using a blockchain are lost. First of all, several clients will not have direct access to the blockchain data. They will have to rely on a node with direct access. This works against the decentralized aspect of a blockchain as there are only a limited amount of possible nodes to connect to. Next, as clients need to connect to a node with full access, not only do they have to trust that node, the interaction with that node becomes a vulnerability as well. For example, a 'man in the middle attack' is possible. This means that when a user tries to interact with the blockchain, a malicious party could intercept this communication and reply with false information [12]. Finally, only a limited set of computers has access to the transactions on the blockchain, creating the possibility of a human factor intervening in

the blockchain operation. This circumvents the reliance of the algorithmically enforced rules of a blockchain.

On the other hand, data on public blockchains can be properly encrypted. This will provide enough security and privacy in most use cases.

Access to transaction processing: permissioned and permissionless For the third level, the developer of the blockchain has to chose who is allowed to accept transactions that get incorporated into the blockchain. The first option is a permissionless blockchain. On a permissionless blockchain, there are no restrictions on the identities of the transaction processors. The opposite is a permissioned blockchain. Only a predefined list of entities can process transactions.

The use of a permissioned, public blockchain can be a necessary compromise in situations where compliance is an issue, such as in the financial sector.

2.2.3 Examples of blockchains

Bitcoin

The best-known implementation of blockchain technology is in the bitcoin cryptocurrency. Bitcoin is a cryptocurrency created by Satoshi Nakamoto in 2008 [13]. In June 2017, bitcoin hit a record market capitalization of $48.8 billion [14]. It is by far the most popular and best-established cryptocurrency currently. The drawback of this popularity is the difficulty to change flaws in the system. This results for example in crises such as the current block size crisis [15].

The bitcoin blockchain is a public, permissionless blockchain with a proof of work consensus mechanism. This means that everyone has read access to the data on this blockchain, everyone can propose transactions to be added to the blockchain and everyone can start a node to help to accept proposed transactions.

The bitcoin blockchain uses a basic scripting language to power its transaction processing. This language is only advanced enough to process transactions and store short pieces of data [5]. On one side, the simplicity of this language could be seen as a limitation of its potential. On the other hand, is it a feature of the bitcoin blockchain making it more secure.

The bitcoin blockchain is interesting because of its extreme security. In July 2017, the bitcoin blockchain mining network has a hash rate of around 6.6 exa hashes per second [16]. This means that the bitcoin network is being supported by the equivalent computing power of 1.9 trillion Macbook Pros from 2014. This shows the security of

the bitcoin network. To break the system, one entity would need to control half of this computing power, which is quite unlikely. The only conceivable way is to develop a supercomputer that is as a more powerful than all the computers ever made, such as a 50 qubits quantum computer [17].

Because the bitcoin mining network is so strong, it is often used by other blockchains to help secure their own network. Namecoin is such an example [18].

Ethereum

Ethereum is another cryptocurrency powered by a blockchain. It is the second most valuable cryptocurrency with a market capitalization of $36.2 billion in June 2017. While bitcoin focuses on creating a secure, non-nonsense currency, ethereum tries to leverage the entire potential of the blockchain. This creates endless possibilities but makes ethereum also more complex than bitcoin.

Like the bitcoin blockchain, the ethereum blockchain is currently public and permissionless with a proof of work consensus mechanism. However, there are plans to move to a proof of stake approach in the future [19].

As explained in the ethereum white paper [20], the programming language powering the ethereum blockchain, called Solidity, is Turing complete. Practically, this means that any conceivable program could be written in this language [21]. This creates the possibility for smart contracts. These are transactions that will be executed automatically once a certain condition is met. An example is a crowdfunding campaign that only transmits the raised budget to the beneficiary once the threshold is met before a certain deadline.

Because of its wide potential, the source code of the ethereum blockchain is often copied (called “a fork”), adjusted and redeployed for custom blockchain solutions.

2.3 Distributed databases

2.3.1 Distributed, decentralized and centralized databases

To understand distributed databases, it is important to distinguish them from decentralized and centralized databases. Baran (1964) [22] explains the difference in the context of networks. A centralized network makes all nodes connect to one central node. A distributed network is the opposite, in which any node can communicate with any other node without the need for a central node. Between lies the decentralized network, in which there are a couple of central nodes that communicate with each other. Other nodes have to connect with one of these central nodes. Figure 2.2 visualizes the differences.

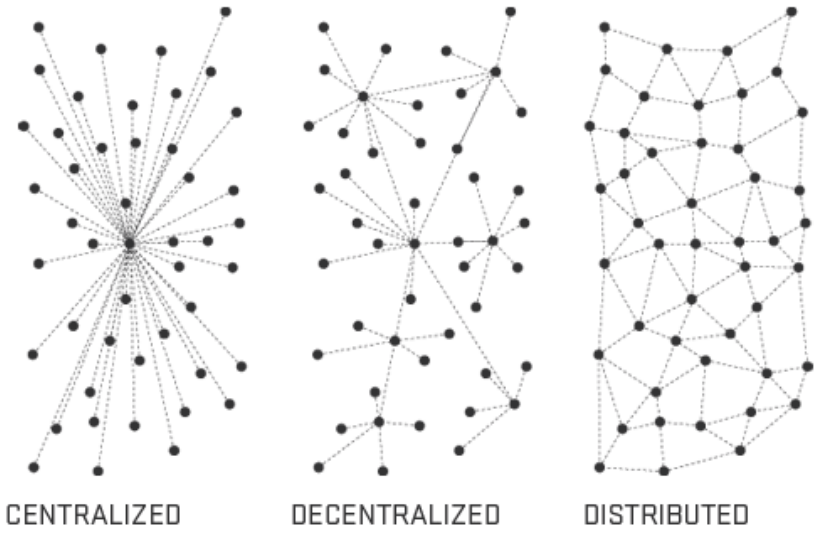

Figure 2.2: Baran, P. (1964) Centralized, decentralized and distributed networks.

This definition can be ported to the context of databases. A centralized database consists of one main data storage node. Users connect to this node to read and write data. A self-hosted website is a perfect example of a centralized database. Any user that wants to connect to your website, has to connect to your server. If your server breaks down, nobody can access the website anymore.

A decentralized database has many storage nodes that communicate with each other. A user can connect to either of these main nodes to access the data. Use of a CDN (Content Delivery Network) is an example of a decentralized database. Websites hosted on a CDN are copied across servers all over the world. To access to this website, a user can connect to any of these servers.

Finally, a distributed database does not have main storage nodes. Instead, every node

can hold the entire database. When a new node wants to access the data, the node asks its peers for the data. Examples are BitTorrent and git, which both will be discussed further one in this chapter.

2.3.2 Consistency, availability, and partition-resilience: the CAP theorem for distributed systems

Clearly, a trade-off is made when deciding between a centralized or distributed database. When a user reads data from a centralized node, the user receives the correct data, as there is only one version of the data. However, when this one centralized node somehow breaks down, nobody will be able to access the database anymore. Fox and Brewer (1999) [23] calls such a database not "partition-resilient". This means that when one node goes offline in the network, the system stops operating.

The opposite is true when working with a distributed database. Even though some nodes are offline, a user can always connect to the data through the other nodes. However, this comes at a drawback. The data the user reads is not necessarily the most updated version, as it could have been updated on another node in the meantime. Fox and Brewer (1999) calls such a database not "consistent".

To avoid the consistency issue, a distributed database can force all nodes to update to the most recent version before responding to anymore read requests. But it could happen that a connection between a not-updated node and the updated node is broken. This hinders the not-updated node in receiving the latest version of the data. The node will refuse any read requests until the connection is restored, to not provide incorrect information. This node is still online and functioning, but not responding to requests. Fox and Brewer (1999) calls such a database not "available".

The CAP theorem (Consistency, Availability, and Partition-resilience) states that any network can choose two out of three characteristics. The theorem was first proposed by Fox and Brewer (1999). It was formally proven by Gilbert and Lynch (2002) [24].

2.3.3 Issues with distributed databases in collaborative systems

In a guided system, there is one authoritative entity controlling every node in the network. But in a collaborative system, separate agents with individual incentives have to work together. This creates a whole new set of challenges, namely consensus and immutability issues.

Consensus issues arise when the database has been updated on two different nodes at the same time. Somehow, these two conflicting versions have to be merged.

Immutability issues emerge when the agreed upon the past can be changed. For example, Alice wants to spend a cryptocurrency coin. The merchant makes sure that the coin is valid and approves the purchase. But when immutability is not properly enforced, Alice can go into the database and manually remove the transaction. This nullifies the transaction, allowing Alice to spend the coin again. It is important to note that this idea of immutability is not related to the immutability representing the C in the CAP theorem.

2.3.4 Examples of distributed databases and networks used in collaborative systems

This final part of the chapter discusses several common implementations of distributed databases and networks in the context of collaborative systems.

Git version control

Git is the most popular distributed version control system. A version control system helps a developer to keep track of the changes he makes to his code. A distributed version control system helps developers collaborate on the same code base by keeping track of everyone's changes. Every developer has a version of the entire code base and all the changes ever made. While developing, a developer makes changes to his local code base. Once finished, he can publish his changes. Often, he publishes his work to a central computer with the latest code. Other developers will pull now and then the code from this central computer to update their local code base. A developer can also pull the code from a colleague, in case he wants to use specific changes that have not yet been added to the central code (Chacon & Straub, 2014 [25]). This entire process is modeled in figure 2.3.

Git includes a specialized algorithm to automatically merge new changes into the code base. Consequentially, the consensus problem is often automatically solved. However, when two developers made changes to the same code, the algorithm fails, resulting in a "merge conflict". It is then up to the developer to manually solve the conflict.

Immutability is not enforced in git. This means that any developer can manipulate the history of changes. This is useful when for example a large file has accidentally been included into the project. This bloats the size of the project and hampers collaboration. Simply removing the file does not help, as the file will still be part of the code history. To completely get rid of this file, the original inclusion has to be nullified. However, this could be used for malicious practices as well. Imagine a problem arises because of a programming mistake made some weeks ago. In case the developer does not want to be held accountable, he could remove his change from the history. Or he could even change

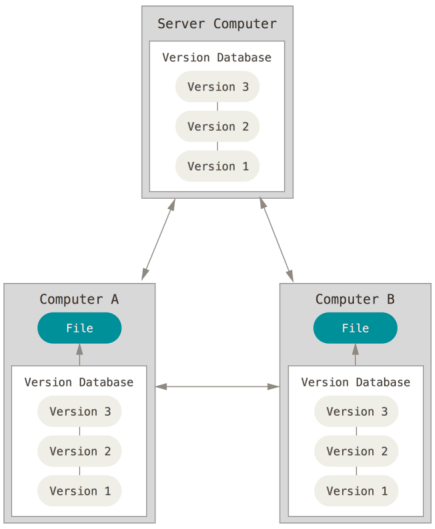

Figure 2.3: Chacon & Straub (2014) Distributed version control.

the author of the change to someone else [26].

It is clear that the git system needs trust between the developers. But in some projects, there is no trust. For example, in open source projects, any developer can propose code changes. This is why on hosting sites for git repositories, such as Github, authorized users need to approve a code change before it can be added to the main code base [27]. But this partially removes the distributed character of git.

Considering the CAP theorem, it is clear git is available and partition-resilient, but not consistent. A node can always download a version of the code base, either from the main server or from one of its colleagues. This makes git available. If a node goes offline, a developer can contact any other developer for the full code base, making git partition-resilient. However, as every developer has a slightly different code base because of his own changes, git will never be consistent.

BitTorrent

BitTorrent is a peer-to-peer file distribution system. Instead of downloading a file from a central server, users download the file from peers that already (partially) have the file. However, once a file is uploaded with BitTorrent, it can no longer be changed. Most databases need some sort of write functionality. This characteristic disqualifies BitTorrent for most database applications. Yet, the single focus makes BitTorrent work really well for file sharing. As a result, BitTorrent was one of the first widely accepted collaborative, distributed networks. It “increasingly becomes the norm for media acquisition

among the general Internet public" (Andersson, J. 2009 [29])

As there are no updates to a BitTorrent file, consensus and immutability are not an issue. Also, the CAP theorem is not relevant in this writeless context. BitTorrent is consistent, available and partition-resilient.

Blockchain

The blockchain technology has already been explained in the previous section. Blockchain solves both the consensus issue as the immutability issue quite well. It is, therefore, a prime candidate for applications that require a collaborative database.

Blockchain handles the consensus issue quite well. As previously explained, once a miner has found a new valid block, he propagates this block through the blockchain network. Other miners will continue mining on the new block, as such appending the blockchain. But if two miners simultaneously find a valid block, they will both propagate this block through the network. Some miners will first receive one block and continue mining on top of that block, others will mine on top of the other block. In this case, there are effectively two different versions of the database. This phenomenon is called a fork. To solve this fork, miners are incentivized to work on top of the longest fork. As it is unlikely that miners will again simultaneously find a new valid block, one fork will quickly become longer than the other. This compels the miners to switch to the longest fork, thus resolving the fork [5].

immutability is built into the blockchain idea. An attacker would first have to create a fork before a certain block he wants to alter. For this fork to be accepted as the proper blockchain, the attacker would need to make the fork longer than the current blockchain. Without more than 50% of the network's computing power, this is mathematically impossible for blocks deep in the blockchain. As Nakamoto (2008) calculates in his paper, there is less than 0.1% an attacker with 25% of total computing power can catch up after 15 blocks have passed. As currently, the largest mining pool holds only 23.8% [31], waiting 15 blocks would be extremely safe. In general, the bitcoin community agrees that after 6 confirmation blocks a transaction is confirmed [32]. For small transactions, best practice states that one confirmation block should suffice. Larger transactions might need three confirmation blocks [33].

The CAP theorem can also be applied to blockchains. As Goland states in his blog [30], blockchains are available and partition-resilient and eventually consistent. As said before concerning the bitcoin blockchain, a user can be virtually sure that a block is immutable after 6 confirmation blocks [32].

However, there are some practical disadvantages with the blockchain. The first disadvantage is the lack of scalability of a blockchain. The main reason is that blockchain

holds all transactions ever executed. As a result, the size of the blockchain constantly increases. This gradually pushes smaller nodes out of the network as they cannot afford to hold the entire blockchain in storage [34]. Another disadvantage is the latency of blockchains. To make sure a transaction has been included in the blockchain, merchants are expected to wait for some confirmation blocks. As the bitcoin blockchain, for example, creates one block every 10 minutes, this could be is an issue for several applications. Imagine going to a coffee shop and having to wait 30 minutes before you can get your coffee. Even though there are blockchains with lower block time (ethereum has a block time of 22 seconds [35]), these blockchains need more confirmations because of the higher chance of a fork. Finally, privacy could be an important concern on public blockchains. Although the data stored to the blockchain can be properly encrypted, everyone with reading access to the blockchain can see when this data is submitted. Again, this could be a problem for certain applications.

Non-persistent P2P networks

The final aspect of this chapter discusses collaborative peer-to-peer (P2P) networks that are not persistent. This means that there the data on the network is not stored. It can only be accessed in real-time.

An example of such a network is Open Bazaar. Open bazaar is a decentralized peer-to-peer marketplace. When a customer buys an item from eBay, the customer searches through listings that have been posted to the centralized servers of eBay. When searching for a specific item on Open Bazaar, a customer searches through the network of nodes that are currently online. But when a node posting a listing goes offline, this listing will no longer show up in search results [36]. This is why we call this a non-persistent network instead of a distributed database.

Chapter 3

Aim of research

The central research question of this paper asks "How can the idea to replace a middleman with technology be applied to collaborative, distributed databases?". Section 2.3 shows collaborative, distributed databases face immutability and consensus issues. These issues arise because there is no central authority or trusted middleman. Sometimes some authority is created, such as in the example of git. However, section 2.3.4 indicates that blockchain solves both the consensus and immutability issue without the need for a trusted authority.

When applying the main idea of replacing a middleman with technology to collaborative databases, two use cases come up. The first case focuses on a centralized database that is stored without a trusted middleman. The second case examines the management of the interactions between the nodes of a distributed database without a central, guiding authority.

The first case studies centralized, collaborative databases. Centralized databases have not been heavily covered in the previous chapters as they are rather trivial. But it is, of course, possible to collaborative on a centralized database as well. An example would be a shared Dropbox folder. Several users work together on the same files, stored at a trusted third party [37]. The final chapter looks into proposals to store this data in a distributed way instead. It is important to note that proposals in this category do not intend to solve either consensus or immutability issue. Because the collaborative databases are centralized to begin with, this would not be possible.

The second case focuses on decentralized, collaborative databases. As often discussed before, the two main issues are consensus between the nodes and immutability between the data stored on the nodes. The final chapter examines several proposals to manage these interactions in a distributed manner without a central authority.

Chapter 4

Proposed solutions

This final chapter discusses both general and application-specific proposals to manage collaborative databases. All proposals can be roughly divided into two groups.

Proposals in the first group (proposals 4.2 and 4.3) focus on centralized, collaborative databases. They focus on the removal of a trusted third party in the storage of data.

Proposals in the second group (proposals 4.1, 4.5 and 4.4) focus on distributed, collaborative databases. They focus on solving the immutability and consensus issues without a trusted middleman.

4.1 Storing centralized data onto a blockchain

An obvious proposal is to store centralized data directly onto a blockchain, for example onto the bitcoin blockchain. A user could encrypt his data and store it onto the bitcoin blockchain using a data transaction. Anyone else with the proper decryption key could access this data and submit an updated version.

However, there are multiple issues with this approach.

First, it is not possible to change or remove data once it is stored onto the blockchain. To update a file, the user would have to upload the entire new file. Another option is to store just the changes to the original file onto the blockchain. In this case, the application used to read the data from the blockchain would create the most recent version of the file based on the original file and the submitted changes.

Secondly, blockchains' characteristics such as low scalability, high latency, and potential privacy concerns make them useless for most centralized database applications. These

issues have been thoroughly discussed in section 2.3.4.

Finally, storing data onto a blockchain is very expensive because of its low scalability. Considering the bitcoin blockchain, every byte of storage costs 0.0000026 BTC[38]. At a current market cap of 2418.09 per BTC, 1 KB costs 6.44. [14].

On the other hand, solving the issues immutability and consensus are the main selling point of blockchains but are not seen as issues in a context of centralized databases. On the contrary, enforced immutability could be a problem for certain applications that require delete functionality.

Because of these reasons, it is clear that blockchains are not a viable instrument to store normal amounts of centralized data directly.

4.2 Distributed cloud storage

Distributed cloud storage network replaces a trusted storage provider such as Amazon or Dropbox with a large network of storage providers. Storage providers can be large companies with data centers to their disposal or individuals with some available space on their hard drive. These storage providers are referred to as the hosts.

When a user, called a renter, wants to store his data on the network, he divides his data into small pieces. The host then sends these pieces to the hosts. For extra redundancy and security, the renter duplicates his pieces several times on the network. The renter then pays the hosts for the time they store their data.

This simple concept faces multiple issues. A host could promise to store a renter's data without actually keeping the data. A renter would have to do some insane redundancy duplication on the network to safely store his data, while still being at risk of losing an essential piece of data. On the other hand, a host storing a renter's data is never guaranteed the renter will actually pay.

Currently, several solutions try to implement this concept. Siacoin, Storj, and SWARM implement focus on cloud storage. Others, such as IPFS and Maidsafe, want to go further and are trying to replace the internet as it currently works. This thesis will focus on cloud storage and compare two current commercial solutions: Siacoin and Storj.

4.2.1 Storj

Storj uses a pay as you go system. From time to time, the renter asks the hosts for a proof of storage that they still hold the data. If the renter receives this proof, he pays them. If the renter does not receive the proof, his assumes this host does no longer store his data and the renter makes sure that piece of the software is reduplicated on the network. If a host does not receive a payment for a proof of storage, he assumes the renter does no longer require him to store the data and that he can delete the renter's data. [40]

Although this solves some of the original incentive issues, there are still other issues.

Hosts are never sure they will be paid when storing data. This means that the network is vulnerable to a denial of service attack. If a malicious agent stores a lot of data on the network without ever paying the first proof of storage, the network's storage space is being wasted, making it unavailable for honest renters.

Furthermore, hosts still have no cost of not providing a proof of storage. This makes the network more vulnerable to a Sybil attack [39], in which a malicious agent pretends to be multiple hosts without any cost. A renter might think his data is secure as it

seems duplicated over several hosts. However, the malicious agent controls these hosts, allowing him to destroy the data at no cost.

4.2.2 Siacoin

Instead of a pay as you go system, Siacoin uses a blockchain to support their network. A host and a renter both sign a smart contract onto the Siacoin blockchain. In this smart contract, the renter puts in his payment for the storage and the host puts in a collateral in case he is not able to produce a proof of storage when the contract expires. Consequentially, the host is assured he will be paid by the blockchain, even when the renter is offline at the time the contract expires. The renter has a better assurance the host will provide the stored data as he would be rewarded the collateral instead.

Because the Siacoin blockchain is publicly available, hosts' performance is publicly visible. Hosts that are always providing a proof of storage when asked will be very reputable, while less available hosts will have a lower reputation or might even be blacklisted.

Finally, Siacoin has an additional protection against a Sybil attack. Although hosts have to pay a collateral if they do not provide a proof of storage, a malicious agent might still spin up an insane amount of hosts. His hosts are very likely to be picked to store data for the same renter. While it is not costless, the data duplication redundancy is not sufficient to prevent the malicious agent from effectively destroying the data of this renter. To prevent this situation, a renter expects a host to prove they are real by providing a proof-of-burn. This means that a host sends coins to an address that cannot spend these coins. [41]

4.3 Applied example: replacing notary testament services

This section will discuss how a notary's testament service can be replaced by technology. In the context of centralized databases, the notary can be seen a trusted storage provider for centralized storage, the testament.

It is not necessary by Belgian law to have a notary witness or store a testament for the testament to be valid [**?**]. However, notaries are often used to witness the signature, store the testament and in general make sure everything is correct. This section will show how a notary's service can be fully replaced by public key encryption, a blockchain and a regular lawyer with no authority concerning witnessing signatures. The section starts with a very simple solution. The proposal will be gradually expanded, so all aspects of a notary's testament service are covered.

4.3.1 Signature

As mentioned before, the main reason for a notary's service is the witnessing of the signature. Because a testament can be a valuable document, several parties might be incentivized to falsify the document and the signature. As the testator will be deceased, he will not be able to testify in court to verify his signature.

With the incorporation of the digital signature into every Belgian ID, every Belgian has the possibility to sign any document digitally (D. De Win, personal email communication, July 19, 2017). A digital signature is more secure than a manual signature, as it cannot be faked or imitated [43].

In the simplest solution, a testator writes his testament with the help of a regular lawyer to make sure all wording is correct. He then signs a digital version with his digital signature. Finally, he distributes the signed document to all potential stakeholders: family, friends, beneficiary institutions and other recipients. He could even publish his signed testament online to make sure every stakeholder could access it.

4.3.2 Extra security with a blockchain

In this simple proposal, a malicious agent could try to steal the testator's ID in the hours after the testator's death. If this agent knows the pin code of the testator's ID, he could forge a fake testament.

To avoid this situation, the testator could store a fingerprint of the signed testament onto a public blockchain, eg. the bitcoin blockchain. In this situation, it is quickly proven that a document had to exist before the fingerprint had been incorporated into

the public blockchain.

Storing a fingerprint onto a public blockchain has another advantage as well. It is now no longer necessary to date the testament, as the earliest fingerprint on the blockchain can be seen as the date of creation.

4.3.3 Secret keeping

The final aspect of a notary's testament service that has not yet been covered by the blockchain alternative is secret keeping. A notary will only reveal the contents of a testament once the testator is deceased. The current blockchain alternative is expecting the testator to distribute his testament to all stakeholders before he dies. This could result in uncomfortable, unwanted conversations.

An ethereum script could solve this final aspect. Instead of distributing the testament, the testator could distribute an encrypted version of the testament instead. He could then store the key to decrypt the testament into a script on the ethereum blockchain. This script would only release the decryption key once the testator has deceased. There are several ways for this script to determine whether the testator is effectively deceased.

There are several approaches to write this script. The testator could distribute keys to family and friends. These key holders can then vote to release the decryption key, with the testator given a veto. If the testator is not yet deceased, he will always be able to block the release the key. The rationale of using a voting system is to prevent immediate publication if the testator is offline for a while and unable to exercise his veto. Another solution could involve a script that routinely asks a public governmental database whether the testator has been declared deceased.

4.4 Storing distributed data onto a blockchain

As discussed in section 4.1, it is hard to find a viable to store centralized data onto a blockchain. But concerning distributed data, there are multiple use cases that benefit from storing their data directly onto a blockchain. These use cases have several characteristics in common. First, consensus and/or immutability are real issues. On the other hand, the downsides of a blockchain are not big problems: the low scalability is not a problem as they do not require large amounts of data to be stored, the high latency is not a problem as they prefer immutability over direct availability, and the application does not require delete functionality.

This section will discuss two use cases. Although they might seem different, at the core they both solve similar issues. Every use case starts out with an incentive issue between peers resulting in immutability and/or consensus issues. Then every case checks if the constraints of blockchain technology limit the potential use of a blockchain.

4.4.1 Managing an intercompany blacklist of unwanted clients

Several companies keep a blacklist of clients they do no longer want to do business with. To prevent shopping between competitors and protect the industry, some industries have moved from an intracompany blacklist to an intercompany blacklist.

However, competitors have incentives to mess with this blacklist. They could decide not to put a bad client on the intercompany blacklist to hurt their competitors. On the other hand, a company could decide to put a good client on the intercompany blacklist. When its competitors refuse this client, the company will be able to negotiate a more favorable contract with the client.

These issues could be solved by retribution system. If it can be proven that a company did not include a bad client after having a bad customer relationship, or if a company cannot provide strong evidence for an inclusion of a client to the blacklist, this company should be punished. This punishment could for example be a monetary fine, or the company could be denied further access to the blacklist.

But now are participating companies incentivized to mess with the data on the blacklist itself. If a company is accused of not including a bad client or of including a good client, this company could go into the database and respectively add or remove the client. In a normal database, there would be no way to prove the rogue company messed with the data. Immutability is essential.

By using a custom blockchain to store the blacklist, immutability and accountability are guaranteed. Scalability is not a problem as the blacklist should never become too large,

as it only includes the identification of the blacklisted entity. Latency is also not an issue, as it is probably sufficient for an entity to be included into the blockchain within the hour of being defined a bad client. Finally, removal of clients is not essential in a standard situation. To indicate a removal of the client from the blacklist, a company can add another data transaction to the blockchain vouching for this client. But, this last point will be expanded upon in section 4.5.

Specific design of such a custom blockchain is rather flexible. Because all data and metadata on the blockchain should be private, a private blockchain is advised. The most suitable consensus mechanism depends on the requirements and constraints of the specific situation. For example on the maximum time before a record can never be changed and the maximum maintenance cost.

4.4.2 Resource allocation

Several organizations deal with some kind of allocation problem of a limited resource. This resource could be anything. It could be a client lead several sales offices want to close but only one sales office is allowed to work on at the same time, it could be the use of a practice studio, or it could be the use of a domain name as in Namecoin [18].

In most of these situations there are incentive issues between the participants to tinker with the database. The examples above deal with immutability issues. A rival sales office might want to steal a lead allocated to another sales office, a band might want to remove any proof of their reservation of the practice studio because they do not want to pay, and someone might want to take over a popular domain name. There are often consensus issues as well. If two bands try to reserve the same practice studio time slot at roughly the same time, which band should be awarded the reservation?

Resource allocation problems clearly deal with immutability and consensus problems, which could be solved by a custom blockchain. Scalability will not often be a problem, as only the resource and the allocator are stored. Latency is not a problem as long as the allocation is made some time before it is actually needed. Finally, deletion is not essential. If necessary, the application could incorporate a transaction type that frees up an allocation.

Design decisions need to be adapted to specific situation.

Finally, besides directly solving the allocation problem, a blockchain also allows for queries on who allocated what. This is very interesting in the domain name example. It allows someone with knowledge of the blockchain to find the owner and the associated location of a certain domain name. But for the other examples this could be useful as well. In the sales office example, general management could easily find which sales office is working on which lead.

4.5 Timestamping a database

Although blockchains have many use cases for storage of distributed data, section 4.4 shows that blockchain technology has some serious constraints as well. This section proposes an alternative to storage directly on the blockchain if the use case needs scalability or needs the functionality to delete records.

Because the blocks in a blockchain are never deleted, a blockchain only grows in size. If the transactions of a blockchain consist of more than a small message, the blockchain could quickly explode in size. As a result, any use case that requires the peers in the network to store larger amounts of data cannot use a blockchain.

A use case could also require deletion functionality. As Jean-Luc Verhelst, blockchain expert at Deloitte Belgium, explains: one of the issues of implementing blockchains inside the European Union is dealing with the European right to be forgotten (J.-L. Verhelst, personal interview, July 10, 2017). The proposed solution in section 4.4.1 would be illegal considering the right to be forgotten, as it is never possible to completely remove a bad client from the records.

To evade these issues, peers in the network can decide to use a database similar to how git operates. Every peer would hold its own version of the data. If a peer changes an entry, he propagates the change through the network. If a peer receiving such a change thinks this change is fine, he might decide to implement the change. If the change conflicts with another change to the same data, this peer might decide not to implement the change. Because not everyone implements the same changes, all versions of the database will gradually start to differ.

To fix these differences, all peers will come together every now and then to manually agree on a consensus concerning the differences between the databases. Once consensus is achieved, all peers agree on a single truthful version of the data. To ensure immutability, participants then take a fingerprint of the database, digitally sign this fingerprint and post it to a public blockchain, eg. the bitcoin blockchain.

Peers can now continue to work from this new version of the database and remove all previous data. If a peer wants to prove the distributed database contained certain data at a certain time, he only has to show that an exact this version of the database has the same fingerprint as the fingerprint stored on the public blockchain at that time. As the fingerprint has been signed by all peers, all peers agreed at the time that was the current version of the truth. This essentially timestamps the database.

This solves the scalability issue as a digital fingerprint is agnostic of the size of its subject. The size of the database does not matter anymore. Deletion functionality is available as well. Once consensus has been achieved, all previous versions of the database can be deleted. Applied to the blacklist example of section 4.4.1, peers can decide to remove

a client from the blacklist essentially removing him from the database used for related decisions.

This solution has some disadvantages as well. First, the consensus process can be very costly. It can take a long time before all peers achieve manual agreement. Furthermore, data cannot be changed in a production environment during the consensus process, as nobody knows what the next version of the database will look like. Finally, this solution involves a very high latency. Only the latest version of the database on which consensus has been achieved can be used to make important decisions. Although the constant updates from the peers can be incorporated in a working version of the database, a peer will never be sure such a change will be incorporated into the next consensus agreement.

Because of these disadvantages, the timestamping solution should only be used if there is no other option.

Chapter 5

Conclusion

Throughout this paper, it has become clear several trusted third parties can be replaced by technology.

In the private environments, central storage providers and central interaction management providers can be replaced by blockchains. In the public environment, privileged professions such as notaries can be replaced by the more secure alternative of public key encryption and blockchains.

I think governments could largely reduce the amount signature disputes by incorporating blockchains into the law. Although this paper specifically focused on testaments, a similar case could be made for other contracts.

If the government does move to blockchains, several aspects have to be taken into account concerning which blockchains should be eligible. I am convinced the law should not limit itself to one single blockchain, but should rather put requirements on which public blockchains could be used: which consensus algorithm should be used, what is the minimum hashing power the blockchain should possess and what is the maximum centralization of the hashing power.

Personally, I am convinced only proof of work has currently proven to be extremely secure. Concerning minimum hashing power, I would recommend making the law time agnostic by defining the minimum hashing power as a percentage of the estimated total computing power currently available in the world. Concerning centralization, I would recommend putting maximum cumulative shares on the largest miners. For example, the largest miner should not own more than 30% of total hashing power, the largest two miners should not own more than 40% combined and the largest three miners should not control more than 50%.

But even without laws specifically mentioning blockchain, I think blockchain verified

testaments (and contracts) are already more secure. As mentioned before, a notary is not required for a Belgian testament to be valid. It is therefore only a matter of time before a case is brought before a court concerning which testament is valid: a digitally signed, blockchain verified document or a notary witnessed one. This will set a precedent for the future, probably forcing the legislative branch to incorporate blockchains into the laws.

Bibliography

[1] Rivest, R.L. and Shamir, A. and Adleman, L.M. US Patent 4,405,829 issued in 1983

[2] Cipher, C., & Cipher, M. (2004). Introduction to cryptography. EEC, 484, 584.

[3] Calderbank, M. (2007). The RSA Cryptosystem: History, Algorithm, Primes.

[4] Bischoff, P. (2017). What is Blockchain? 10 experts attempt to explain it in 150 words or less. Comparitech. Retrieved 20 July 2017, from https://www.comparitech.com/blog/information-security/what-is-blockchain-experts-explain/

[5] Antonopoulos, A. M. (2014). Mastering Bitcoin. Sebastopol, CA.

[6] Rouse, M. (February 2006). What is data structure? - Definition from WhatIs.com. SearchSQLServer. Retrieved 20 July 2017, from http://searchsqlserver.techtarget.com/definition/data-structure

[7] Abelson, H. et al. (1996). Structure and Interpretation of Computer Programs. MIT Press.

[8] Bitfury Group (2015). Proof of Stake vs Proof of Work

[9] CryptoCurrency Market Capitalizations. (2017). Coinmarketcap.com. Retrieved 20 July 2017, from https://coinmarketcap.com/currencies/

[10] Alternatives for Proof of Work, Part 2: Proof of Activity, Proof of Burn, Proof of Capacity, and Byzantine Generals Bytecoin Blog. (2017). Bytecoin: private secure financial system. Retrieved 19 July 2017, from https://bytecoin.org/blog/proof-of-activity-proof-of-burn-proof-of-capacity/

[11] Bitfury Group (2015). Public versus Private blockchains. Part 1: Permissioned Blockchains.

[12] Desmedt, Y. (2011). Man-in-the-middle attack. In Encyclopedia of cryptography and security (pp. 759-759). Springer US.

[13] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[14] Bitcoin (BTC) price, charts, market cap, and other metrics — CoinMarketCap. (n.d.). Retrieved July 19, 2017, from https://coinmarketcap.com/currencies/bitcoin

[15] Chen, L. Y., & Nakamura, Y. (2017, July 10). Bitcoin Is Having a Civil War Right as It Enters a Critical Month. Retrieved July 19, 2017, from https://www.bloomberg.com/news/articles/2017-07-10/bitcoin-risks-splintering-as-civil-war-enters-critical-month

[16] Hash Rate - Blockchain. (n.d.). Retrieved July 19, 2017, from https://blockchain.info/charts/hash-rate

[17] Bauer, M. R. (2017, April 14). Quantum Computing is going commercial with the potential to disrupt everything. Retrieved July 19, 2017, from http://www.newsweek.com/2017/04/21/quantum-computing-ibm-580751.html

[18] Loibl, A. (2014). Namecoin. namecoin. info.

[19] Buterin, V. et al. (2017). Proof of Stake FAQ. URL https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[20] Buterin, V. (2013). Ethereum white paper.

[21] Kepser, S. (2004, August). A Simple Proof for the Turing-Completeness of XSLT and XQuery. In Extreme Markup Languages. Chicago

[22] Baran, P. (1964). On distributed communications networks. IEEE transactions on Communications Systems, 12(1), 1-9.

[23] Fox, A., & Brewer, E. A. (1999). Harvest, yield, and scalable tolerant systems. In Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on (pp. 174-178). IEEE.

[24] Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. Acm Sigact News, 33(2), 51-59.

[25] Chacon, S., & Straub, B. (2014). Pro git. Apress.

[26] How can I change the author (name / email) of a commit? (2017). Git-tower.com. Retrieved 21 July 2017, from https://www.git-tower.com/learn/git/faq/change-author-name-email

[27] About pull requests - User Documentation . (2017). Help.github.com. Retrieved 21 July 2017, from https://help.github.com/articles/about-pull-requests/

[28] Cohen, B. (2003, June). Incentives build robustness in BitTorrent. In Workshop on Economics of Peer-to-Peer systems (Vol. 6, pp. 68-72).

[29] Andersson, J. (2009). For the good of the net: The Pirate Bay as a strategic sovereign. Culture Machine, 10. Chicago

[30] Goland, Y. Y. (March 8, 2017). The block chain and the CAP Theorem. Retrieved 21 July 2017, from http://www.goland.org/blockchain_and_cap

[31] Hashrate Distribution. (2017). Blockchain.info. Retrieved 21 July 2017, from https://blockchain.info/pools

[32] Confirmation - Bitcoin Wiki. (2017). En.bitcoin.it. Retrieved 21 July 2017, from https://en.bitcoin.it/wiki/Confirmation

[33] Gornick, S. (March 13, 2013). How many confirmations do I need to ensure a transaction is successful?. Bitcoin.stackexchange.com. Retrieved 21 July 2017, from https://bitcoin.stackexchange.com/a/8373/47645

[34] James-Lubin, K. (2015). Blockchain scalability. O'Reilly Media. Retrieved 21 July 2017, from https://www.oreilly.com/ideas/blockchain-scalability

[35] Ethereum / Ether (ETH) statistics - Price, Blocks Count, Difficulty, Hashrate, Value. (2017). Bitinfocharts.com. Retrieved 21 July 2017, from https://bitinfocharts.com/ethereum/

[36] FAQ - OpenBazaar Docs. (2017). Docs.openbazaar.org. Retrieved 21 July 2017, from https://docs.openbazaar.org/09.-Frequently-Asked-Questions/

[37] Shared folders: Give people edit access to your files. (2017). Dropbox.com. Retrieved 22 July 2017, from https://www.dropbox.com/help/files-folders/share-with-others

[38] Bitcoin Fees for Transactions — bitcoinfees.21.co. (2017). Bitcoinfees.21.co. Retrieved 22 July 2017, from https://bitcoinfees.21.co/

[39] Douceur, John R. (2002). The Sybil Attack. Springer Berlin Heidelberg

[40] Wilkinson, S., Boshevski, T., Brandoff, J., & Buterin, V. (2014). Storj a peer-to-peer cloud storage network. Chicago

[41] Vorick, D., & Champine, L. (2014). Sia: Simple Decentralized Storage.

[42] Notary Public vs. Lawyer: What's the Difference?. (2015). Downtown Notary Toronto. Retrieved 7 August 2017, from http://downtownnotarytoronto.com/the-notable-blog/2015/6/22/notary-public-vs-lawyer-whats-the-difference

[43] De Cock, D. (April 2017). Belgian eID cards & ePassports.

FACULTY OF BUSINESS AND ECONOMICS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be