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, July 24, 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.

The final chapter investigates the viability of specific proposals to answer the research question: how every proposal tries to solve its specific problem, the advantages and the disadvantages of its approach.

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.

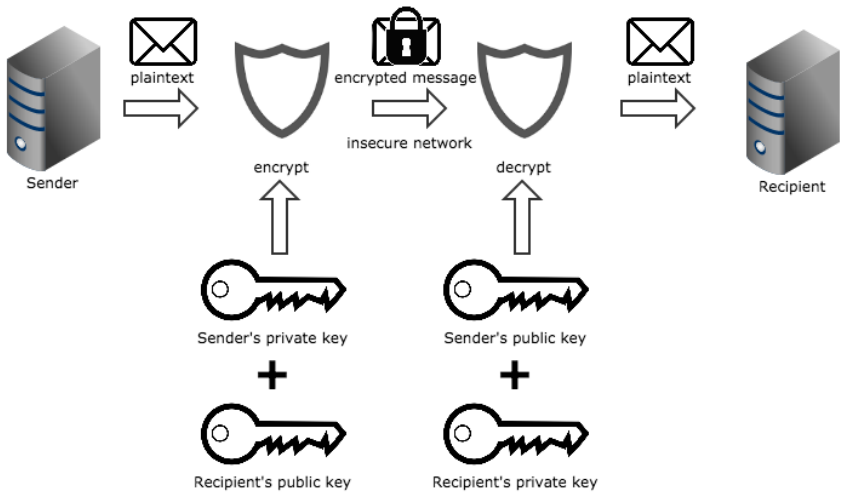

Figure 2.1

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.

Antonopoulos (2014) [5] defines the blockchain as follows: "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, we are talking about a data structure: "A specialized format for organizing and storing data" [6]. There are several types of data structures such as arrays, lists, graphs, trees, etc.

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]). Incidentally, the "ordered" in the definition provided by Antonopoulos is superfluous, as by definition every list is ordered. An unordered list would no longer be a list but would be a set instead. 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

Antonopoulos (2014) [5] defines transactions as: "data structures that encode the transfer of value between participants in the bitcoin system" (p. 109). Although it gives a good idea of what transactions are, even in the context of the bitcoin blockchain this definition is incomplete. Despite most transactions being value transactions between participants, a transaction can also store data on a blockchain. Even the bitcoin blockchain allows non-monetary transactions to be included.

Hence, we can conclude that there are two types of transactions: monetary transactions and non-monetary transactions. In a monetary transaction, there is a transfer of value

(eg. an amount of bitcoin) between participants. It must be noted that this does not have to be an instant transfer. Several blockchain protocols allow for a custom condition to be scripted into the transaction.

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. Everything discussed in this chapter exists to make sure that transactions can be validated, propagated over the network and added to a distributed ledger.

Blocks

Antonopoulos (2014) [5] defines a block as: "a container data structure that aggregates transactions for inclusion in the public ledger, the blockchain." (p. 160). A block is a wrapper for transactions to make it possible for them to be included into the blockchain. These blocks form the ordered values of the back-linked list addressed in the beginning of this section, with each block linking to the previous block in the list.

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 entities 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 not 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 money 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 out of 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 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 an 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 of the 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 an 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 accepting proposed transactions.

The bitcoin blockchain uses a basic scripting language to power its transaction processing. This language only advanced enough to process a 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 Pro's 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.

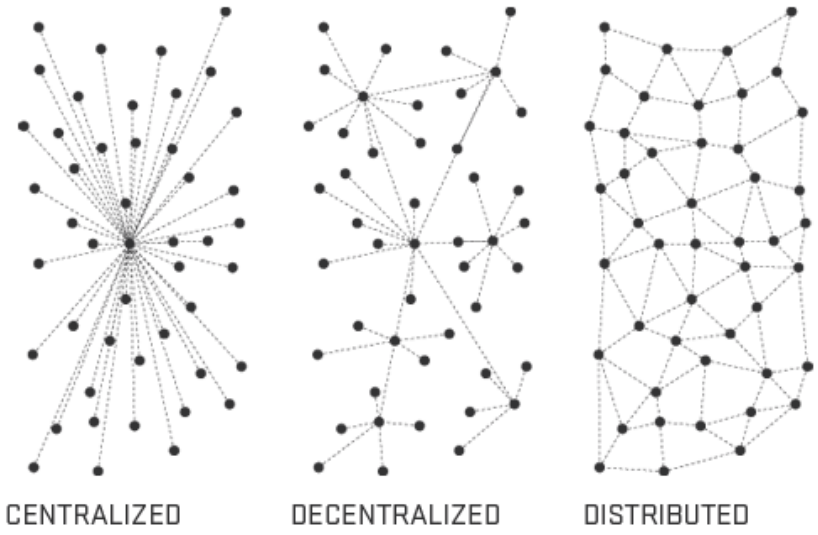

Figure 2.2: Baran, P. (1964) Centralized, decentralized and distributed networks.

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

It is clear 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 consistency 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.

Consistency 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 consistency 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 consistency is not related to the consistency 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.

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

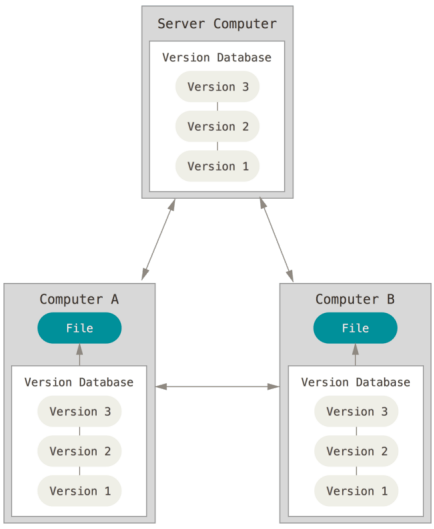

Figure 2.3: Chacon & Straub (2014) Distributed version control.

It is clear that the git system needs trust between the developers. But in certain 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 consistency 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 consistency 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].

Consistency 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 read 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 this idea to replace a middleman with technology be applied to collaborative, distributed databases?". Section 2.3 shows collaborative, distributed databases face consistency 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 consistency 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 intent to solve either consensus or consistency 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 consistency 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. The first group (proposals 4.1, 4.2 and 4.3) focuses on centralized, collaborative databases. The second group (proposals 4.4 and 4.5) focuses on distributed, collaborative databases.

4.1 Storing onto the bitcoin blockchain

An obvious proposal is to store collaborative 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 key could access this data, change it and submit an updated version.

However, there are multiple issues with this approach.

4.2 Blockchain distributed storage

4.3 Applied example: replacing notary will services

4.4 Timestamping a database

4.5 Resource Allocation

Chapter 5

Conclusion

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] What is data structure? - Definition from WhatIs.com. (February 2006). 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/

Appendix A

A.1 Appendix

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