Master's Thesis

# Designing a cryptocurrency and a detailed analysis on anonymous schemes

Chair of Economic Theory
Universität Basel

Supervised by:
Professor Dr. Aleksander Berentsen

Author:
Tobias Petri
Submission Date: January 11, 2018

## Abstract

The goal of the following thesis will be to design a digital currency with preferably a private ledger without losing the effciency of a decentralized approach and the implied blockchain technology. Furthermore, an analysis of various anonymity and privacy procedures from an economical perspective within decentralized schemes will be shown. To preceed this, there will be a presentation of the vast bulk of promised anonymous structures from existing cryptocurrencies. After that there will be one possible implementation of the presented anonymous schemes for the newly designed cryptocurrency.

# Contents

# List of Figures

# Plagiatserklärung

Ich bezeuge mit meiner Unterschrift, dass meine Angaben über die bei der Abfassung meiner Arbeit benutzten Hilfsmittel sowie über die mir zuteil gewordene Hilfe in jeder Hinsicht der Wahrheit entsprechen und vollständig sind.
Ich habe das Merkblatt zu Plagiat und Betrug vom 22. Februar 2011 gelesen und bin mir der Konsequenzen eines solchen Handelns bewusst.

Basel, January 11, 2018

Tobias Petri

# 1 Introduction

A currency is a form of money that includes coins and paper notes. Generally speaking each country has its own official currency which are typically issued by each country's central bank, but there are several notable exceptions to this model. One exception is the Euro which represents the legal tender for multiple European countries. Moreover, the currency can be used as a medium of exchange for goods and services, as a unit of account and as a store of value. The definition of a cryptocurrency is somewhat different to a fiat currency. A cryptocurrency is a type of digital or a virtual currency that does not possess any physical form and it is only available in the underlying network. Furthermore, cryptography is used in order to validate secure transactions. Cryptocurrencies do not need to be issued, regulated, and controlled by sovereigns and its responsible authorities, meaning the value of cryptocurrencies is set by the market. In addition, cryptocurrencies have no intrinsic value, can be seen as a medium of exchange, and most often have entirely decentralized networks with independent verification procedures.

The world of digital currencies is embedded in a rapidly evolving environment. The most famous digital currency is Bitcoin. It was introduced in 2009 by Satoshi Nakamoto and since then has captured the public's attention. Bitcoin does not only face competition by traditional existing fiat currencies, backed by central banks, but also from other cryptocurrencies. Additionally, Bitcoin has a first-mover advantage in the cryptocurrency market. It could be an advantage or might be a drawback in which new digital currencies could profit from a second-mover advantage and suppress Bitcoin's market dominance in the future. Currently, Bitcoin is still the digital currency with the highest price and the biggest market capitalization. As of January 11, 2018 its market price is trading at $13,600 with a market capitalization of $ 228 billion. The other cryptocurrencies in the top five are: Ethereum with a price of $ 1,200 and a market capitalization of $ 117 billion, Ripple with $ 1.70 and market capitalization of $ 67 billion, BitcoinCash with $ 2,500 and market capitalization of $ 42 billion, and finally Cardano with a price of $ 0.69 and the market capitalization adds up to $ 18 billion. Ethereum, Ripple, BitcoinCash and Cardano have a combined market capitalizaition of 36 percent, while Bitcoin has 34 percentage points and is dominating the cryptocurrency market.[1]

As some digital currencies have been accepted as a means of payments predominantly online, central banks are paying increased attention to digital currency issues. For instance, the European Central Bank is following their "impact on monetary policy and price stability"(European Central Bank, Feb. 2015), the US Federal Reserve

[1] www.coinmarketcap.com

is implementing their ability to deliver a “faster, more secure and more efficient payment system”(Bernanke, Nov. 2013), and the UK Treasury is inclined to “support innovation”(Johnson et al., 2014)in this area. The Bank for International settlements (BIS) is also reviewing the intersection between cryptocurrency and central banks more thoroughly (Quarterly Review, Sep. 2017). These developments do not come as a surprise due to the current financial systematic settlements in charge of central banks. For example, the Clearing House Automated System (CHAPS), Target 2 and Fedwire are cost intensive and have a reduced possibility to innovate.

The motivation to design a new cryptocurrency draws from the existing ones. Numerous digital currencies are created in a short period of time. However, there is no detailed plan what has to be considered when building a new cryptocurrency from scratch. Furthermore, the invented currency will have specific properties from different available coins and alternative details that could provide new ways of thinking and give additional inputs to the cryptocurrency environment.

A further aspect in this paper will be based on the anonymity of cryptocurrencies. According to Back et. al (2001) the anonymity of a system as a whole is determined by an efficient, reliable and usable approach including the costs for security issues. All these factors influence the size of the user network and the possible degree of an anonymous achievement. Anonymity and privacy have been a central goal of cryptography for a long time. David L. Chaum’s paper from 1985 called “Security without identification: Transaction Systems to Make Big Brother Obsolete” discusses anonymous digital cash using blind signatures and the double spend problem. The title of Chaum’s paper indicates what current blockchains should enable: "a path of regulation without sacrificing privacy" (Diedrich (2016), p.254). As an example with a decentralized structure, Bitcoin does not possess true anonymity. Their transactions have pseudonymous addresses with hash keys in which there can be easy linkages between users if a user reuses a Bitcoin address and the transactions are stored on a public blockchain. In practice, Bitcoin offers only weak anonymous schemes. For these reasons, the approach of the following thesis will introduce further anonymity structures from already existing cryptocurrencies.

To summarize, the thesis will include the following points. Firstly, there will be an introduction of cryptocurrencies and a provision of motivation for a newly designed one. After that, an explanation of the new invented digital currency will be provided including the blockchain with code credentials and a consensus mechanism. In the next section, the economics of anonymity will be highlighted, followed by the anonymous structure of various cryptocurrencies and their implementations, and a possible recommendation of anonymity for the self-invented digital currency. In the

concluding remarks, a brief summary of the findings in the thesis will be presented.

# 2 Designing a Cryptocurrency

The cryptocurrency Bitcoin solved one big issue, the double-spending problem that could not be solved with digital cash. Before that, there had to be an intermediary which confirmed the transactions. However, the blockchain cryptography inhibits people from spending the same digital cash multiple times by having a decentralized ledger whereon transactions are confirmed. This ledger removes the necessity of having a central authority in place. The blockchain of Bitcoin can approach the double-spending problem with a combination of a peer-to-peer structure(P2P) technology and the private-to-public key cryptography making a new piece of digital cash. The coin owners and the transactions are stored in the public ledger. The confirmation of those transactions and the coin ownership is determined by cryptographic protocols and the miners.

The above example of Bitcoin as a cryptocurrency demonstrates that basically each cryptocurrency consists of three main components: a blockchain, a currency and a protocol. Every form of digital cash normally has both a currency and a protocol. In addition to having a currency and a protocol, cryptocurrencies utilize blockchains, sometimes their own unique blockchain or others modelled off of existing blockchains from other ones, such as Bitcoin or Ethereum. Naming an example, Litecoin has its own currency and protocol, and also relies on the Litecoin blockchain.

Figure 1: Verifying a Transaction Procedure

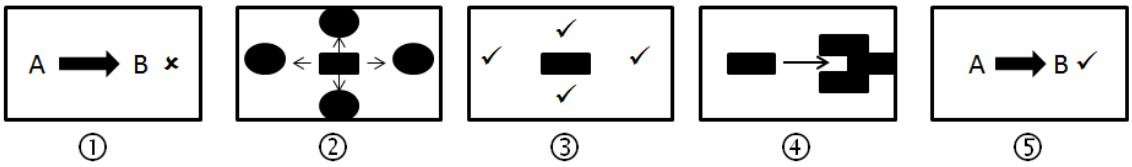



Figure 1 displays how a transaction in the cryptocurrency sphere can be realized in general. First, person A gives a notification that she wants to make a transaction from her account to the account of person B. In two and three, the requested transaction is distributed to a decentralized peer to peer network that consists of nodes mostly in a computing environment. The nodes in this ledger are trying to validate the transaction by using different consensus algorithms such as proof-of-work or proof-of-stake approaches. In the fourth step, when the algoritm consensus has been solved successfully, the transaction can be seen as verified and it can be combined with transactions from other members. The combination of all these

transactions will form a new block of data for the ledger. After that the block can be attached to the existing blockchain. This blockchain, either visible or restricted to the public, can hardly be changed and it will stay permanently on the ledger. Finally, person A and B can be informed that the transaction has been executed succesfully.

## 2.1 Reasons for a New Cryptocurrency

Nevertheless, existing digital currencies such as Bitcoin also have drawbacks. Kosba et al. (2016) are having issues with Bitcoins' privacy and are suggesting a new approach with its Hawk model. Limited privacy can only be ensured with pseudonymous public key implications and the reduced privacy issues are discussed among others by Androulaki et al. (2013) and Bonneau et al. (2014). Thus, Green et al. (2013) also suggest improving the anonymity of a user by presenting a new cryptocurrency named Zerocoin. Moreover, Barber et al. (2012) are showing the loss of Bitcoin coins for instance through malware incidents as an additional disadvantage of Bitcoin. One further weakness involves the reduced transaction scalability. Whereas PayPal can transfer over 100 transactions per second, Bitcoin is only able to handle at maximum seven transactions per second and is having difficulties in increasing this rate to higher levels.

Since 2010, there was a one megabyte restriction for the block size to limit the amount of transactions to seven per second. However, the scalability issue has been a highly controversial topic for several years in the Bitcoin community itself. Due to that on the 1st of August 2017, a hard fork took place in which a new Bitcoin version called BitcoinCash emerged. The original Bitcoin version wanted to increase its block size to two megabytes (MB) in November 2017. However, there were some controversies and on December 28, 2017 another hard fork took place introducing an increase of block size to four MB by Segwit2x when the entire wallets and users change to Segregated Witness addresses and transactions. BitcoinCash does not support Segregated Witness. Nevertheless, its block size rises to eight MB meaning the transactions will increase by eight times per second over today.

The expense of computational energy by dealing with proof-of-work mechanisms and the confidence on broadcast are one component for the limited scalability. The computational energy to handle the transaction ledger and mitigate double spending will have the capacity of large power plants or more by some estimation providers. For instance, in March 2016 the daily energy consumption for the entire mining network was around 350 Kilowatt (KW) that is approximately the energy consump-

tion of 280,000 US households. Deetman (2016) estimates that, at actual growth rates in computing technology, the Bitcoin ecosystem could consume around 15 Gigawatt (GW) by 2020. This amount is similar to the consumption usage of Denmark in 2014. There are some existing alternatives such as Permacoin or Litecoin that can limit the computational energy aspects; however, it cannot reduce the costs efficiently.

Another key limitation of Bitcoin refers to the monetary supply. This involves no control mechanism over monetary supply after its creation. For instance, Bitcoin will provide around 21 million coins in total in 2140 while it is not possible to manipulate the supply which means there is full to almost complete rigid macroeconomic policy and no flexibility. Moreover, the predefined total value of 21 million coins in the algorithm bears the risk of a deflation within the Bitcoin system.

Additionally, the value of Bitcoin and other digital currencies can be extremely volatile as it has been experienced for Bitcoin at the most recent development of its currency performance with respect to the dollar. For instance the highest price of Bitcoin was reached so far in December 2017 with a price above $ 20,000. Just a year ago the price was only around $ 780.[2] The daily market volatility is also quite high meaning that some users might not swap their coins for goods and services and the means of payment is thereby questionable.

The reasons aforementioned imply that existing cryptocurrencies such as Bitcoin as an explicit show case do not satisfy the demand in terms of fast transaction scalability and anonymity properly. However, it is also worth mentioning that one cryptocurrency cannot satisfy all desired key properties, yet. The newly invented cryptocurrency should possess a fast transaction scalability and a flexible network system. Thus, the presented cryptocurrency will have the following key properties: it will use a private blockchain approach, will be highly scalable, provide an optional higher degree of anonymity and a central authorithy will be responsible for an adequate implementation of the new coins, the sound control of the process, and validation of its transaction system.

## 2.2 The Blockchain

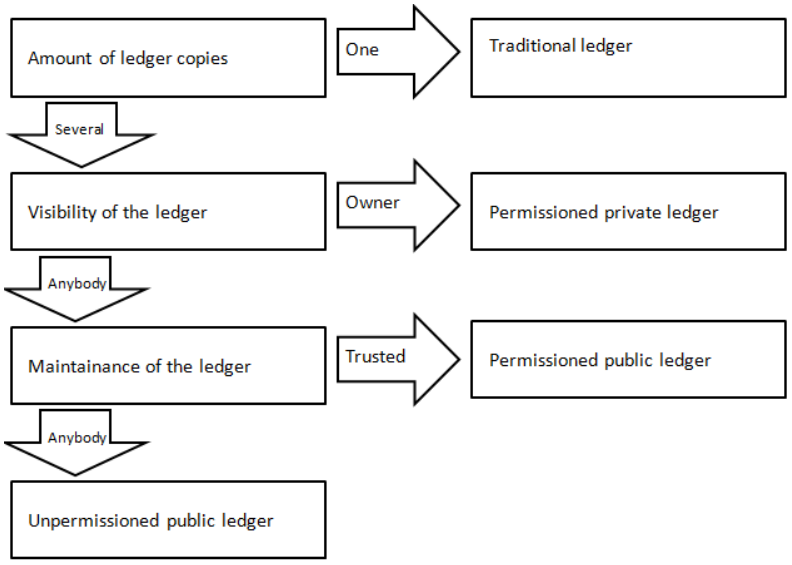
Classically, the verification of transaction processes is dominated by intermediaries. But according to Gomber et. al (2017) dependence on those intermediaries means that the whole procedure takes a long time and is expensive. It also involves a credit -and reputation risk if an intermediary is not able to maintain the verification

[2] www.coinmarketcap.com

process. With the invention of the blockchain technology a user does not have to rely on intermediation for the verification of transactions any longer. The blockchain solves all the aforementioned issues with a change "from trusting people to trusting math" (Antonopoulos, (2014) p.1) because human interference is no longer needed for the verification process. In other words, a blockchain is trustless and the user only has to trust the system and its technology but no other party or intermediary.

A blockchain consists of four basic elements: a replicated ledger, cryptography, consensus, and business logic. The replicated ledger contains a complete history of carried out transactions, it has unalterable past modifications, it is distributed and it can be replicated. The integrity of the ledger is one characteristic of the cryptographic element. Further points include the authenticity of transactions, a possible degree of anonymity and privacy of transactions and the identity of participants. The consensus refers to the decentralized protocol, validated transactions and a shared control that respects possible disruptions. Finally, the business logic presents the logic of the defined ledger and gives out the execution together with the transaction. The logic itself can vary broadly from coins to smart contracts. Based on these four elements, a blockchain can differ immensely due to the implemented specifications and the inherent variability. For these reasons, only blockchains regarding to the cryptocurrency environment, will be considered and their different forms of ledger possibilities will be presented. Following Brennan et al. (2016) three ledger properties can determine the degree of a blockchain. The first property is the amount of copies, the second characteristic is the accessibility of the reader and finally the write access.

Figure 2: Different Forms of Ledger Levels (based on Baran(1962) and Brennan et al. (2016)



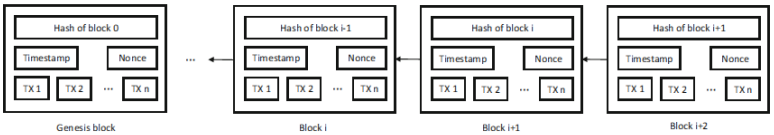
In the cryptocurrency sphere, two basic blockchain types can be distinguished: the private blockchain and the public blockchain.

### 2.2.1 Public Blockchain

Brennan et al. (2016) differentiate the public ledger into two parts. For the permissionless public ledger any person can participate in the network and view the transaction history and the consensus mechanism. Furthermore, anyone can also join the consensus process in which the transactions get verified and to find out the actual state of the chains. On the other side, in the permissioned public ledger, only permissioned entities are allowed to participate in the consensus mechanism controlling a pre-selected set of nodes. But anyone of them can view and make transactions upon the blockchain. By comparing both ledgers, the permissionless public ledger results in increased transparency and accountability.

The public ledger in general is completely distributed and it has a complete trustless integrity. Swanson (2015) mentions an obvious advantage of a public blockchain is that someone does not have to prove her identity to the blockchain network. Furthermore, there are no entry barriers for the users and miners except the needed technology. Morevover, any miner who respects the rules can participate in solving the consensus mechanism and verify the block for getting the mining reward. A further positive argument involves the openness of public ledgers. The open environment permits many participants to use the system, resulting in gaining network effects, for instance in the improvement of the blockchain system. Finally, the public ledger gives users a protection from the developers because one developer does not have the authorithy to change anything within the system. Two points encourage this statement. Firstly, it will strengthen trust and ensure increased interactions within the network. Secondly, the pressure of other entities on oneself will be diminished because nobody can force a user to change anything because one user does not have the authority to proceed. In fewer words, a public blockchain is censorship resistant. Bitcoin -and Ethereum blockchains are representative examples of a permissionless public ledger. Ripple is also an example of a public blockchain, but only permissioned users can take part in the consensus mechanism, while anyone is able to make and see transactions on its blockhain.

Figure 3: Example of a Blockchain: Bitcoin provided by Zheng et al. (2016)



The typical Bitcoin blockchain example can be seen in Figure 3. In this case the blockchain has a chain of data blocks in which one block consists of several transactions (TX1-TX$n$). The respective blockchain is increased by every additional block and shows an entire list of the transaction history. In the specific case of Bitcoin, the miners create the validated blocks that are rewarded by bitcoin coins and transfer fees. The blocks' validation is done by the network via cryptography. Moreover, each block has a timestamp, the hash value of the last verified block and a nonce. The first block of the blockchain is the so called "genesis block" and the following procedure assures the integrity of the complete blockchain including the genesis block. The hash values only exist once, meaning fraud can be prevented. The reason for the prevention of fraud is that the changes within the blocks would immediately change the hash value in the chain. The proof-of-work is a consensus mechanism in the Bitcoin blockchain which ensures that when the majority of nodes agree to the validation of transactions within a block, the block can be added to the existing chain. According to Swanson (2015) the consensus algorithm consists of a set of rules and processes that enables the maintenance between several nodes. Newly added transactions are not a part of the ledger automatically. The consensus procedure gives the permission that the transactions can be part of the block after a specific time. With Bitcoin, transactions can be conveyed to the ledger after ten minutes. After the transfer of the block, altering the information in the blockchain is exceedingly difficult, if not, near impossible.

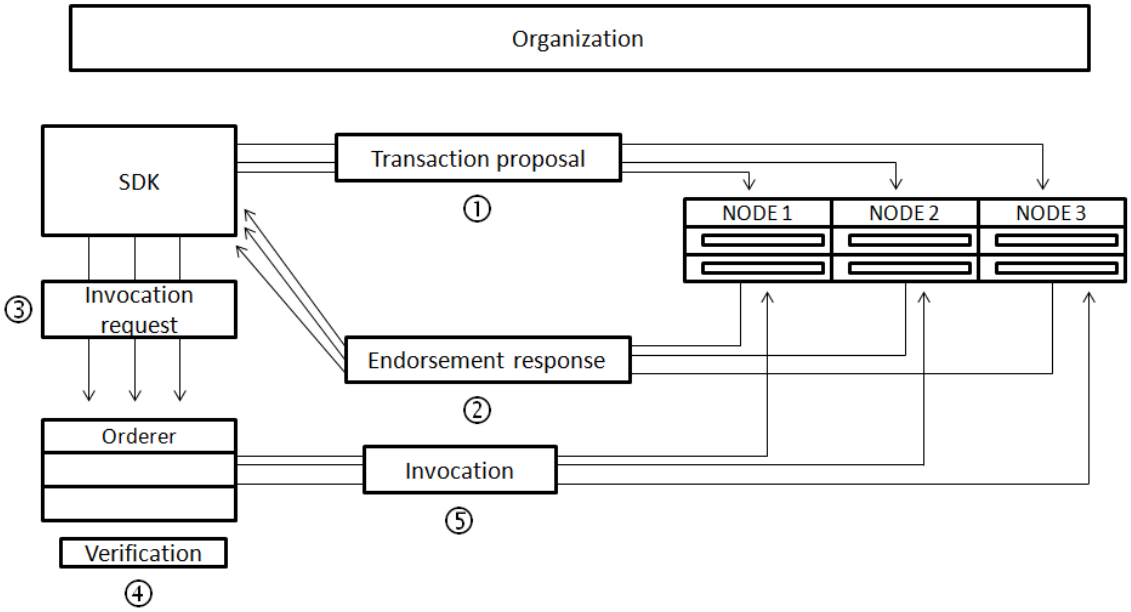
### 2.2.2 Private Blockchain

The private ledger consists of one central authority or organization that can give instructions to the ledger and check the consensus meaning only this authority has write permissions. The responsible authority also cares about the identity of the ledger. Read permissions can be public or restricted depending on the desired characteristics and needs of the blockchain.

In general, the private ledger has multiple advantages compared to public legders. One advantage is that a user has to be an approved entity for participating in the consensus mechanism in the network. Following the consensus mechanism, public ledgers use predominantly proof-of-work applications to reach consensus. Private blockhains do not need to use computer power related mining applications to acquire consensus because all entities and thereby nodes are known. In this case you can use algorithms such as Raft or Paxos and more algorithms without any proof-of-work mining. A detailed analysis of different consensus algorithms will be presented in the next paragraph. As there is no requirement to use any of the mining procedures,

the 51% attack that is needed to manipulate the blockchain within the system from possible miner collusion is not possible. Since the verification process will need only a few nodes, the transactions are cheaper and more scalable referring to public blockchains. A further obvious advantage involves the privacy element of private blockchains. Within private ledgers solely the responsible organization or authority can change the rights or rules of the blockchain, or see the transactions. In addition to that, as long as read permissions will be under restriction, a greater form of privacy will be enhanced which can allow restricted access to the transactions within the blockchain. Existing examples of cryptocurrencies based on a private blockchain are Hyperledger, Interledger or Multichain.

Figure 4: The Concept of Hyperledger




In the following, Hyperldeger has one organization and 3 nodes. The preparation of a transaction is done by the Software Development Kit (SDK). This preparation is called transaction proposal because this proposal will be forwarded to the nodes. It can be sent to one or several nodes depending on policy details. All nodes are executing a simulation on its respective ledgers. Meaning the key from the transaction proposal will be updated to a new value set. The nodes sign it with cryptography and send it as a so called endorsement response to the SDK. The endorsement response consists of the cryptographic details of the node and the transaction, and the updated value of the new set. The SDK collects all endorsement responses, signs them with a key and transmits them as an invocation request to an orderer. The orderer has the task to verify the chaincode and its implemented policy. A possible policy could be that all nodes have to agree on the transaction proposal. However, if for example only node one got the proposal and sent the endorsement response to the SDK, the orderer cannot verify the transaction due to unfulfilled policy re-

quirements. Additionally, the orderer verifies every endorsement response from all nodes and the updated set of the new values. Those endorsement requests from all participating nodes must be the same because the result will consist of nodes that have the same ledger, the same data and the same chaincodes. If the policy and the endorsement requests are valid, the orderer sends the updated set of data to all the nodes as an invocation command with a valid transaction. The participating nodes accept the new data set and update its respective ledger on the same stage to keep all nodes in synchronization. Hyperledger is not restricted to only one organization. It can also have several organizations in parallel executing transactions with a respective orderer and multiple nodes.

## 2.3 The Consensus Mechanism

The blockchain's main operation is that the network should agree on the ledger properties collectively. However, the authority for maintaining the accounts is a decentralized structure. In order to achieve decentralization on the blockchain, the network keeps a consensus mechanism around the recorded information. Consensus mechanisms enable a secure update of distributed states by using a fault tolerance within the system. The updating process of the replicas in the system follows a programmed transition rule in the network with the help of a state machine executing it on every replica. This process ensures that the state will still be in the system if some nodes crash. The state machine rule allows for the same execution of outputs of every node, giving the same inputs and resulting in an agreement in the consensus protocol. In other words, the state transition rules are the rules of the blockchain protocol.

According to Baliga (2017), a consensus protocol consists of three key characteristics. The first property is safety which is when all nodes give out the same output and the nodes are in line with the protocol. It also refers to the consistency of the shared states. The second point involves the guarantee of liveness of the consensus protocol when all valid nodes participate in the consensus producing a final value. Fault tolerance is the third property and a consensus protocol can be maintained if it can live with a few faulty nodes that participate in the consensus process. Fischer et al. (1985) show that for asynchronous systems all three properties cannot be satisfied simultaneously by a deterministic consensus protocol. Fault tolerance can elaborate in two different ways within distributed networks. The first category, called fail-stop faults, causes nodes to stop participate in the consensus protocol by having hardware or software issues. The second form of faults are Byzantine faults. Lamport et al. (1980) come across the "Byzantine General's problem". A Byzantine

node is able to lie, can give out ambiguous results or mislead other nodes which are part of the consensus protocol. With a limited amount of Byzantine nodes in the distributed system, the consensus protocol must reach consensus and the consensus protocol is not allowed to operate incorrectly. As with different forms of blockchains, the consensus protocol also works differently in a permissionless and a permissioned setup.

### 2.3.1 Permissionless Consenus Mechanism

In a permissionless network, the number of nodes is large and unknown. Any node can join the network meaning the nodes are anonymous and trustless. For this system, the consensus mechanism has to be responsible for malicious behavior, especially Sybil attacks. Bitcoin solved the problem with a proof-of-work (PoW) approach. Other early adopters of cryptocurrencies such as LiteCoin or Monero also rely on the PoW consensus.

#### The PoW mechanism

In the Bitcoin PoW mechanism each node has to prove it has done some work in order to add blocks to the blockchain. Following Diedrich (2016) the work is a non-ending puzzle challenge in which all nodes participate against each other in the network. The node looks for a hash value given a correct nonce input which is not greater than a specific number attached with a difficulty level set by the system. The difficulty level depends on the Bitcoin protocol and the currently existing hash power-the hash rate- of the nodes. On average, one block is produced in a ten minute interval. To solve the PoW task, the node has to find a matching hash value and this process is called mining. The node that finds the wining hash value first gets the mining reward. Currently, this mining reward consists of 12.5 newly minted bitcoins and an optionable transaction fee. The mining reward of the bitcoins halves every four years and in the future there will be only a transaction fee left. However, in the race to the wining hash value more than one node can find a matching hash value at the same time. This means every winning node informs the network about its new added block to the blockchain which can result in a temporary fork in the system. In this case by adding more blocks, the branch with the maximum size will be part of the blockchain, eventually and the other branch will die out.

According to Baliga (2017) the PoW mechanisms have several weak elements. The first includes the possibility of 51% attacks in which the attacker can double-spend his or her funds or can actively dictate which transactions to include on the blockhain. Eyal et al. (2013) prove a further weakness of the PoW approach. It is

called selfish mining, in which honest mining entities are persuaded to join a 51% attack. Additionally, other factors that influence the Bitcoin PoW negatively are a long duration of transaction confirmation resulting in a poor match of immediate transaction finality and a high transaction rate. The waste of energy computation for getting the hash values in the mining process is an additional drawback of Bitcoin's PoW system. On the other hand, the scalability of the nodes participating in the network is very good and the system is completely decentralized with open-end participation.

The Bitcoin PoW is not the only PoW in the cryptocurrency environment. Ethereum has a separate consensus model known as EthHash. EthHash is confirmed faster and created on ASIC resistance to fight the 51% attacks that Bitcoin is vulnerable to. Mining centralization is an additional weak point of Bitcoin and a further reason that EthHash has been designed. The PoW of Ethereum uses two properties for tackling mining centralization. The first one is called memory hardness in which the computer is able to shift data around in memory unlike pursuing calculations. The second technique refers to the GHOST protocol that is a revised version of the Bitcoin PoW. The headers of the recently discarded blocks are contained in the technique of GHOST. The discarded blocks, called uncle blocks, were part of the temporary forks and not on the main blockchain. The node that creates the uncle block and the node that includes the created uncle blocks on the blockchain, receive a smaller reward to nudge them to work on the current block in the Ethereum blockchain again. As with the Bitcoin PoW, the EthHash is also looking to find a correct nonce input that can give out a hash value below a specific difficulty threshold. Ehereum's PoW also cares about a possible 51% attack. But with the ASIC design, EthHash can develop a better resistance level in its network than the Bitcoin PoW. In addition, Ethereum plans to move to a proof-of stake (PoS) algorithm in the future.

**The PoS mechanism**

The main disadvantage of PoW consensus protocols involves the high waste of electricity consumption in the mining process. The PoS mechanism is trying to overcome this problem by using a different approach. Instead of using a mining operation, a user's stake or ownership of the cryptocurrency will play a part in the blockchain system. The amount of coins of each participant will be used as a stake to acquire a part in the cryptocurrency's validation process. This stake is proportional to the transformation of a user to a validator and the chance of creating blocks in the blockchain. For instance a user with 200 coins in the system will be two times more likely to be selected as someone with only 100 coins.

Furthermore, the PoS procedure does not need any specialist hardware to mine the blocks. The PoS algorithm cannot predict the next validator of block creation as the selection is pseudo-randomized. Some weak PoS algorithms are confronted with an issue known as "Nothing-at-Stake". In those cases, the nodes have no incentive to vote on the correct block meaning nodes can vote on several blocks with several forks in order to enhance their chances of getting remuneration. In this circumstance a user with nothing to lose has no incentive to behave honestly. One possible solution is to instruct a validator to store the coins in a form of digital vault. In case the validator tries to double fork or vote in the network, his or her coins will be destroyed. Peercoin implemented the PoS mechanism firstly. Other cryptocurrencies that followed the approach are BitShares, NXT or Tendermint. Ethereum will release an own PoS algorithm called Casper in the Serenity version of Ethereum in the future. Casper uses the procedure of digital vaults and hopes to achieve consensus in this way.

**Further "proof-of" mechanisms**

Beside the PoW and PoS mechanisms there are numerous consensus mechanisms around today and it is still an ongoing process in the field of permissionless setups. An alternative consensus protocol is the proof of elapsed time (PoET). This algorithm works almost as the PoW approach but with less waste of electricity. Furthermore, no cryptographic puzzle needs to be solved because the consensus mechanism relies on a trusted execution environment to assure that the creation of blocks is done in a randomized lottery and without any work. The proof-of-activity mechanism is another one that uses a combination of both PoW and PoS. In the first stage, there is a mining process until the winning block only includes a header and the address of the miner's remuneration. In the second stage, the system enables PoS and it is based on the headers' information while some validators are selected to sign the newly created block. The ownership of coins of the validator determines the probability of the validator's selection. The fees are divided between the miners and the validators. The cryptocurrency Decred uses a variation of the proof-of-activity algorithm. By sending coins to a non-reachable address in spite of investing money for the computer hardware, the amount of coins permits the user to mine in the system with a randomized selection process. This consensus mechanism is called proof-of burn. The more coins the user "burns" by sending coins to a non-reachable address, the greater their chances are to mine the new block. The cryptocurrency Slimcoin uses a combination of PoW, PoS and the proof-of-burn mechanism. In the proof-of-capacity environment the user will provide hard drive space. The more a user will supply, the greater is the possibility to mine the newly created block and receive the mining reward. Burstcoin implemented the proof-of-capacity approach in

its consensus mechanism. Additional variations of the proof-of-capacity algorithms are proof-of-storage and proof-of-space mechanisms.

### 2.3.2 Permissioned Consensus Mechanism

In the permissioned case, the amount of nodes is restrictedly small and the identity is known, meaning they can be semi-trusted in general. For these reasons, compared to the permissionless network an alternative consensus mechanism without any computer power based mining, such as the PoW, can be implemented. Permissioned platforms can rely on and adopt existing algorithms such as Paxos, Raft or numerous "Byzantine Fault Tolerance" algorithms. Within distributed systems, those consensus approaches have placed their focus on creating fault tolerance facing unreliable systems that provided fail-stop faults. Lamport (2001) presents one form of such an algorithm to implement it in a fault-tolerant distributed environment called Paxos. This consensus algorithm consists of three roles executed by three classes namely proposers, acceptors and learners. On the one hand, Paxos is created to provide a fault-tolerant and consistent approach regarding completely or temporary failed nodes. On the other hand, it is built to ensure a reliable network in case of unreliably delivered messages. Paxos' consensus algorithm provides a progressive and consistent procedure which can be reached with a state machine process within a distributed system.

Ongaro et al. (2014) present a consensus algorithm called Raft which is based on Paxos. Raft focuses on a replicated log and enhances the element of understandability of its consensus algorithm compared to Paxos. In general, the efficiency in Paxos and Raft are the same unlike the structure to improve understandability and building of practical systems. Furthermore, there is a separation of the leader election, log replication, providing a safe system, and membership changes. The servers in Raft can also be divided in one of the three states similar to Paxos: a leader, the followers and the candidates. First, the client goes to all the followers, after that, the log gets to the selected leader. Finally, all the candidates are used to choose a new leader. The greatest difference between Paxos and Raft is the leader selection. In Raft, the leader selection plays an important part while it is not the case in Paxos' algorithm. Paxos, Raft and further modifications of them can order transactions in distributed systems. These networks can organize generated requests from clients and respective state transformations in a distributed environment with the use of replicated state machines. In such systems $2f + 1$ numbers of nodes are needed, where $f$ is the tolerance level with the amount of fail-stop failures.

**Practical Byzantine Fault Tolerance (PBFT) algorithm**

With the tolerance of Byzantine faults, the consensus protocol got more complex with additional layers of messaging within the system. Castro et al. (1999) show a "Practical Byzantine Fault Tolerance" (PBFT) algorithm that can provide consensus with Byzantine faults. The PBFT approach uses the concept of a replicated state machine and voting. The replicas can be distinguished into primary and secondary replicas in which the secondary ones review the proper implementation of decisions from the primary replicas and can be substituted with a new primary in case the used one is compromised. The replicas can also be used to optimize the signature and encryption of messages that have been swapped between replicas and clients. A further optimization involves the reduction of the size and the amount of messages exchanged for a smooth implementation in the system amid Byzantine faults. The algorithm needs $3f+1$ replicas to tolerate $f$ faulty nodes. Regarding the performance of the replicated service the PBFT algorithm institutes a low overhead. Following experiments in the paper of Castro et al. (1999), a 3% overhead can be reported for a reproduced network file system (NFS) service. Nevertheless, the PBFT algorithm is limited to around 20 nodes in terms of scalability and current academic research results. Above the 20 nodes threshold, the messaging overhead goes up tremendously due to an increase of the number of replicas.

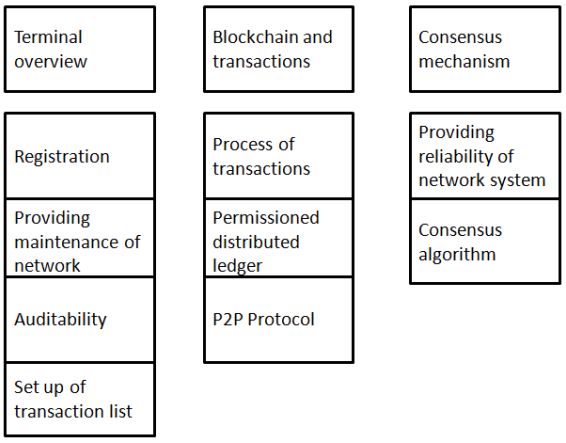
**Ripple consensus algorithm**

Following Schwartz et al. (2014) the consensus algorithm of Ripple is inspired by the Byzantine Fault tolerance with modifications of unlimited participation from users, gateways and also market-makers. Each node needs a definition for a Unique Node List (UNL). The UNL consists of other Ripple nodes with trust of the existing node and no collusion against it. A consensus can be reached on the Ripple network when each node calls other nodes in its UNL. Every UNL needs an interference of 40% with other nodes in the system. The consensus can be achieved in several rounds in which each node gathers transactions in a specific data form. The data structure is known as “candidate set” and a node relays its candidate sets to other ones. The validation of the transactions is ensured by the nodes and a voting system. In the voting system, each node screens its candidate set and sends transactions to the next round that are getting the largest amount of voting shares. If a candidate set gets a supermajority, in Ripple’s case more than 80% of the accumulated votes from all the nodes in the UNL, the candidate set can be seen as a valid block or a ledger. Finally, the ledger is called the “Last Closed Ledger” and it will be included on the Ripple blockchain. The appended consensus round begins with new transactions and transactions that have not been proceeded in the previous round of consensus. Once each sub-network achieved consensus, the process of consensus in the Ripple network is completed.

## 2.4 Creating the New Cryptocurrency

As stated before, the key properties of a cryptocurrency are a blockchain, a currency and a protocol. The newly invented digital currency will be based on a protocol to send transactions from different participants within the network. The peer-to-peer structure in the validation process of the transactions is an additional protocol that will be used as it ensures resilience and decentralization aspects for the consensus mechanism. The currency will be called ChoiceCoin (CC) and its tokens are called Choice. Furthermore, ChociceCoin's main characteristics are a private blockchain backed by a central authority, cryptographic usage and optional anonymity which will be described in the next paragraph.

Figure 5: Architecture of the Cryptocurrency




The architecture of ChoiceCoin will be based on three key modules: the terminal overview, the blockchain and transaction module, and the consensus mechanism.

### 2.4.1 The Terminal Overview

As the new cryptocurrency will be based on a restricted validator network, the central authority will be in charge of the so called terminal overview. One service component of the terminal includes the registration procedure of the nodes in the system. Those pre-defined nodes can determine the verification process of the transactions and identify the transaction authority. Furthermore, the central entity will be responsible for controlling and managing the authorizations of the participants in the new network. In addition to that, the terminal is also authorized to provide the disclosure of identities and roles of the participants of the cryptocurrency meaning it will provide the overall maintenance of the network. In this perspective, the central entity can support a fast and flexible adoption of new rules and orders and facilitate their access of it to control the environment.

To implement or change the existing rules in the network, the pre-defined nodes can participate in a voting system and if a majority is reached to change anything on the set of rules, the nodes can propose it to the central authority. However, it should also be noticed that the exercising power of the central authority has to be limited and it cannot play god in the system without any approval by the participating nodes. One option of the central entity could be a veto right for inaugurating new rules voted by the majority of nodes. The auditability property signifies the terminal can offer the provision to ensure authorized participants to handle and allow the transactions of each user using the network. Furthermore, the central authority can also screen the complete process of the network system in order to improve the efficiency and scalability. Eventually, the terminal overview will set up a deterministically ordered transaction list which will be submitted to the node leader in the transaction process.

### 2.4.2 The Blockchain and Transactions

#### The transaction process

The transaction procedure in the system works as follows:

1. Any user can propose a transaction to one of the defined nodes that is authorized in the network and in the acceptance of transactions. This node will save all its receiving transaction details such as the hashed values and the amount on a separate transaction list as well.

2. All transactions from the chosen nodes will be collected. After that the central authority will take all transactions from the nodes that sent them to the authority and order them numerically with a timestamp to a transaction list. The central authority is also responsible to delete any double entries or wrong sets of transactions and hands over the transaction list to the node leader.

3. The node leader will be determined with a random function in the beginning of each new transaction round. More details will be provided in the code representation of the blockchain implementation.

4. This node leader is the leader of the verification process and takes back the transaction list from the central authority to submit it to all validator nodes in the network.

5. The nodes verify the transactions with a consensus algorithm that will be similar to Ripple's consensus algorithm with a voting system, an "avalanche process" and the need of a supermajority that is more than 80% of the votes.

6. After several verification rounds and reaching the supermajority the transaction list can be seen as valid and the list will be transformed to a new block called "theAddedBlock" to the existing blockchain.

7. Following the confirmation of the blockchain by all subnodes, the transaction can be seen as accomplished and a new transaction round can begin.

The mentioned transaction process only shows a proper execution of the network system without any misbehavior of nodes so far. The treatment of malicious nodes and possible attacks will be covered in the choice of a stable and reliable consensus algorithm. Moreover, it should be mentioned that any transactions which did not get into the first round of verification for "theAddedBlock" might be a part of the transaction list for the next verification round and the next block implementation.

**The choice of an accurate blockchain type**

A major reason why the private blockchain approach will be chosen relates to the scalability issues of public blockchains. McConaghy et al. (2016) differentiate the scalability into three parts: latency, throughput, and capacity and network bandwidth. In all mentioned parts, most public blockchain types such as Bitcoin's have worse properties in those three parts than the permissioned counterparts. As the blockchain should consist of a high throughput, a low latency and a smaller capacity and network bandwidth, the private blockcchain is superior to the public ledger possibility in transferring transactions. Even though the security versus cost trade-off equips the private case with a lower security than public blockchains, the coming costs have a bigger impact in choosing the blockchain technology in this case. The permissioned ledger opportunities do not have to rely on high cost -and unneeded energy intensive consensus mechanisms.

The consensus mechanisms of public blockchains also affect the transaction scalability negatively. In addition to that, only permissioned entities will be allowed in the participation of the consensus mechanism in the blockchain. The consensus mechanism will be presented in the next sub-section. Furthermore, the pre-selected nodes can be semi-trusted while one central authority can organize and change the blockchain instructions. This will result in a more flexible and faster implementation of new rules and upcoming changes. Since the anonymous structure will be introduced in the next section, until now anyone of the pre-defined nodes can view and make transactions upon the blockchain.

**General process of the blockchain**

The created blockchain is based on code lines running on the Python programming

language. [3] In the following a few key properties will be highlighted to enhance the understanding of the chosen blockchain approach.

Figure 6: Code makeTransaction

```
import random
import time
random.seed(0)

def makeTransaction(maxValue=5):
    # It will create valid random transactions in the range of 1 to with a timestamp
    sign = int(random.getrandbits(1))*2 - 1   # This will randomly choose -1 or 1
    amount = random.randint(1,maxValue)
    aPays = sign * amount
    bPays = -1 * aPays
    # timestamp based on milliseconds
    timestamp1 = time.time() * 1000
    print(timestamp1)
    time.sleep(0.1)
    amount = random.randint(1, maxValue)
    cPays =  sign * amount
    dPays = -1 * cPays
    # timestamp based on milliseconds
    timestamp2 = time.time() * 1000
    print(timestamp2)
    # It will always return transactions that respect the conservation of tokens.
    # However, since now no checks pursued whether these overdraft an account
    return {u'A':aPays,u'B':bPays,u'C':cPays,u'D':dPays}
```

The code snippet "makeTransaction()" displays two randomly produced transaction pairs from users A to B and C to D. Deposits will be indicated with positive numbers, while withdrawals will have a negative sign. Each transaction will have a timestamp, here shown as "timestamp1" and "timestamp2" based on milliseconds. Additionally, there will be the amount of the transaction for instance "aPays" for the user A and an individual hash value. However, each node should also save its received transactions on an extra transaction list in case the transaction details will have to be reviewed at a later point to prevent fraudulent use of the system.

Figure 7: Code txnRawList

```
txnRawList = [makeTransaction() for i in range(20)]
# The txnRawList will create 20 transactions
```

The "txnRawList()" will generate several transactions for each pair of transactions. In this case 20 transactions will be simulated. This list will be transferred to the central authority afterwards. There, it will check the transactions in general and rank it into a deterministically ordered list based on the respective timestamps on a first-in first-out approach. To implement a smooth block creation, there will be a few checks and set of rules for the users and its transactions. Firstly, coins cannot be created or destroyed by default. In other words the sum of deposits and withdrawals of each transaction must be zero. The second condition is that a user's account must be covered with enough funds for withdrawals and mitigate overdraft issues.[4]In case this set of rules cannot be maintained, the transaction will be rejected.

[3]The complete code of the simulated blockchain can be found in the appendix section.

[4]Those validity checks are defined in the functions "updateState()" and "isValidTxn()" in the code section in the appendix

Figure 8: Code nodeLeader

```
# Determine the node leader
random.seed(0)
master_node = random.randint(1,24);
print("The leader node will be the following node:" + str(master_node))
```

The node leader is based on a randomized function. In this case there are 24 pre-defined nodes in the network. Thus, the node leader will be a random number between one and 24. The node leader will be the leader of the following verification round and relays the transaction list to all other pre-defined nodes. By applying this working process, it will be ensured that no node will manipulate any transaction such as omitting transactions which the node does not prefer or pursuing malicious behavior. In the next round a new node leader will be determined with the help of the random function generator. After that the verification process begins. The consensus mechanism will be explained in the next sub-section including the consensus algorithm without any code.

Figure 9: Code block creation

```
def makeBlock(txns, chain):
    time.sleep(0.1)
    timeStamp = time.time() * 1000
    time.sleep(0.1)
    parentBlock = chain[-1]
    previousHash = parentBlock[u'hash']
    blockNumber = parentBlock[u'contents'][u'blockNumber'] + 1
    txnCount = len(txns)
    blockContents = {u'timestamp': timeStamp,u'blockNumber': blockNumber, u'previousHash': previousHash,
                     u'txnCount': len(txns), 'txns': txns}
    blockHash = newHash(blockContents)
    block = {u'hash': blockHash, u'contents': blockContents}
```

After successfully executing the consensus mechanism, blocks will be built. The blocks have the following components: a timestamp of the block creation, a block number, the previous hash of the parent's block, the amount of transactions ("txns") and the new block hash. The initial states of users can be found in the definition of the genesis block.[5] In this demonstration, the network assumes each user starts with 60 coins. This assumption will simplify the simulation process with the creation of the blocks and the blockchain in the following case.

Figure 10: Code block size limit

```
blockSizeLimit = 4  # Randomly chosen number of transactions per block
```

In this demonstration the amount of transaction to form a block is set to four to show the process in theory. However, in a more practical approach as can be seen with Ripple, the number of transaction could be 1000 to 1500 transactions per block. This number is also feasible in the shown approach. The number can be set by the central authority in the beginning and the defined nodes can also vote to change this number at a later point in time if desired.

[5] see Appendix for source code: "# generating the states and the genesis block"

Figure 11: Code create final block

```
## Create final block
theAddedBlock = makeBlock(txnList, chain)
chain.append(theAddedBlock)
```

If a supermajority in the consensus algorithm for all the transactions within the transaction list is achieved, the list will be transformed into a new block called "theAddedBlock" and will be on the existing blockchain (see Figure 11).

Figure 12: Code output first block

```
Output of first block:
{'contents': {'txns': [{'A': -5, 'B': 5, 'C': -2, 'D': 2}, {'A': -3, 'B': 3, 'C': -1, 'D': 1}, {'A': 4, 'B': -4, 'C': 3, 'D': -
3}, {'A': 4, 'B': -4, 'C': 1, 'D': -1}], 'txnCount': 4, 'blockNumber': 1, 'parentHash':
'314f0aedf64e33e80823bf07420c939731908ea10b8b82f61a7475d4eff071dd', 'timestamp':
1508314219898.487}, 'hash':
'faea5f4b6eaf30d782b796e88145b0140954023382fad0fd28b5a78a632f1f7d'}
```

This code snippet shows the output of chain [1]. The block does have the number one because the genesis block started with zero. The hash of the genesis block is shown in the previous hash. Furthermore, all transactions of the users are shown under the "txns" section. In total, there are four transactions ("txnCount") as we defined it in the "blockSizeLimit" section before. Finally, the new block also has a timestamp and a new hash value: "faea5f4b6eaf30d782b796e88145b0140954023382fad0fd28b5a78a632f1f7d".

Figure 13: Code state

```
state
print (state)
Output of state:
{'A': 63, 'B': 57, 'C': 68, 'D': 52}
```

It is also worth mentioning that with this approach and creating the first block, the account balances that are denoted, as "state" in the code section for each user, get updated immediately with all four transactions for the respective user.

Before the defined nodes can send the block to the blockchain, several checks will be made to verify the chain validity. The checks involve the hash value of the blocks, the validity of the blocks, and the state of the chain.[6] The hash value check should return that the block component matches the hash. Verifying the validity of the blocks makes sure each block is based on the previous block and the current state. It should return an updated state given the block is valid or otherwise send an error message. Finally, checking the chains ensures the complete chain is valid including the genesis block. If this is true it will give back the system state and if not an error message will be drawn. Those checks are necessary in a blockchain environment due to the nodes. In this approach only full nodes will be accepted in the system.

[6] see Appendix for source codes of checks: "def checkingBlockHash(block):", "def checkingBlockValidity(block, parent, state):" and "def checkingChain(chain):"

This property is similar to the Bitcoin system. A full node means the complete blockchain will be maintained including all transactions on the pre-selected nodes. Those nodes verify and create the blocks independently starting with the genesis block until the most recently added block in the network. By checking the validity of the complete chain, a new node can be protected against misbehavior by others including invalid transactions. Additionally, the participating nodes in the system have to check the validity of the newly added blocks to keep an updated blockchain structure.

The complete blockchain structure can be compressed to the following code lines:

Figure 14: Code complete Blockchain: Part1

```
import copy
nodeBlockchain = copy.copy(chain)
nodeBlockTxns = [makeTransaction() for i in range(4)]
newAddedBlock = makeBlock(nodeBlockTxns,nodeBlockchain)
```

The expression "nodeBlockchain" returns the respective chain and the "nodeBlock-Txns" includes all necessary transactions. In this case the four transactions and the chain represents a new block entry for a defined node in the network.

Figure 15: Code complete Blockchain: Part2

```
print("Amount of blocks on the blockchain on Node 1 is currently:"+ str(len(chain)))
print("Amount of blocks on the blockchain on Node 2 is currently:"+ str(len(chain)))

try:
    print("A new block is in the system and in the verification process!")
    state = checkingBlockValidity(newAddedBlock,chain[-1],state) # Updating the current state
    chain.append(newAddedBlock)
except:
    print("No valid block! Waiting for the next block...")

print("The Blockchain has been updated succesfully! On Node 1 it is %s blocks long"%len(chain))
print("The Blockchain has been updated succesfully! On Node 2 it is %s blocks long"%len(chain))
```

In the following the "newAddedBlock" is transferred to two nodes, node one and node two. Those nodes check the block and will update the current state if the validity of the "newAddedBlock" was successful.

Figure 16: Code output general

```
Output:
Amount of blocks on the blockchain on Node A is currently:6
Amount of blocks on the blockchain on Node B is currently:6
A new block is in the system and in the verification process!
The Blockchain has been updated successfully! On Node A it is 7 blocks long
The Blockchain has been updated successfully! On Node B it is 7 blocks long
```

In this example, in the beginning the blockchain consists of six blocks on both nodes. Following that node one and node two check the state of the "newAddedBlock". If the check was successful, the result will be a new valid block. The new created block is added to the blockchain thereby the total amount of the blockchain increased to seven blocks on both nodes eventually.

### 2.4.3 The Consensus Mechanism

To select a reasonable consensus mechanism several properties are crucial. One indicator is the blockchain type. As aforementioned it will be based on a private blockchain structure. Thus, the PoW model and with some limitiations the PoS model, that could also be used in a permissioned setup, will not be ideal for the setting. The transaction finality is another key issue. A transaction can be seen as final as soon as it is added to a block on a blockchain. This procedure is pursued faster on the consensus algorithms of private blockchains than on public ones. Especially, the PoW and PoET take a longer time based on the model construction with the danger of several blocks being mined simultaneously. In other words those consensus models have probabilistic transaction finality.

Consecutively on the transaction finality argument is the transaction rate. The following consensus should execute and confirm transactions fast. This can be achieved on a consensus mechanism based on PBFT including the Ripple and Stellar protocol and PoS approaches. The cost versus security tradeoff also plays a significant role to determine a reliable consensus model. Whereas PoW and PoS have a high cost for the participation process in the consensus with high energy costs in the PoW approach and high coin expenses for the PoS model, the costs are low for the others based on a permissioned setup. The characteristic of scalability of the peer network is higher in the unpermissioned consensus types. An increasing number of nodes in the consensus system in the permissioned case increases the amount of messages sent between the nodes exponentially. The consequence is a greater amount of overhead in the system. Therefore, for a fast system the number of peers should be around 20 optimally. The level of trust plays no role in the proof of models in the unpermissioned case. There, the nodes can be untrusted. On the other hand, in a permissioned environment nodes have to be known. Furthermore, based on current research at least two thirds of the validating nodes should behave correctly to maintain a valid consensus process and avoid Byzantine failures. Based on all these characteristics, the Ripple protocol with slight modifications provides the best option to implement it as a meaningful consensus mechanism for the new cryptocurrency.

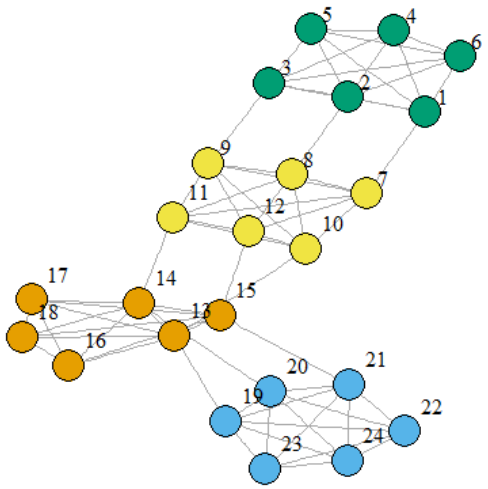
#### The consensus algorithm

The Ripple protocol is based on the usage of semi-trusted sub-networks in a wider system. The factor of trust is almost negligible and can be made smaller with an appropriate selection of member nodes. A consensus mechanism with a low latency is the consequence of the Ripple protocol with a robust setup in terms of Byzantine failures and standard failures. Schwartz et al. (2014) define three main goals that

should be satisfied with the Ripple algorithm: agreement, correctness and utility. The components to reach consensus of the Ripple protocol are a server, a ledger, the last-closed ledger, the open ledger, a Unique Node List (UNL) and a proposer. The server is responsible to run the Ripple Server Software while only registered user nodes can participate in the consensus procedure. The updated blockchain with all valid transactions verified in the consensus process is called ledger in the Ripple protocol. The last-closed ledger is the most current verified block and reflects the recent state of the system. The open ledger represents the current working block of the nodes reaching consensus. A key specific characteristic of the protocol involves the UNL. Every server $s$ has a UNL consisting of servers that $s$ asks when verifying the consensus mechanism. Solely the nodes of $s$ are respected meaning the UNL can be seen as a sub-network that is trusted by $s$ on a collective perspective. However, not any participants of the UNL have to be trusted. The proposer element guarantees only suggestions from servers on the UNL of a server $s$ can be considered by $s$ itself.

A nonfaulty node behaves honestly in the system, while a faulty node can get an error either though being honest and getting standard failures based on data issues or the faulty node performs a malicious behavior with Byzantine errors. To formalize a correct transaction in the protocol, an individual validating node makes a decision given the information on the binary value of zero, implying no success or, one meaning success. According to Attiya et al. (1984) consensus can be defined based on three axioms: First, each nonfaulty node decides finitely ($C1$). Second, all nonfaulty nodes get the identical decision value ($C2$). The third axiom embodies for every nonfaulty nodes, zero and one are both feasible values. ($C3$).

Figure 17: 24 pre-defined Nodes and 4 UNLs in a Network System




**ChoiceCoin's consensus algorithm**

The following algorithm of ChoiceCoin is based on the Ripple Protocol consensus algorithm (RPCA) with slight differences. The RPCA runs every few seconds by all registered nodes. Unlike in the Ripple algorithm, the validating nodes in ChoiceCoin cannot be run by the central authority in order to boost the decentralized argument in the verification process. In figure 17, the new cryptocurrency will start with 24 pre-defined nodes and four UNLs with six nodes each in total. It can be seen that not all nodes need to be linked to each other to reach consensus consistently. There will be several rounds to achieve consensus. At first, every server prepares a list with all unconfirmed valid transactions. After checking the transaction list with the help of the terminal overview, the node leader gets the transaction list back. The node leader distributes the transaction list to each server on its UNL and starts the verification procedure based on a voting mechanism. If a transaction gets more than a minimum of "yes" votes, the transaction will be forwarded to the next round if one is available. Transactions that do not receive sufficient votes will not be processed in this round or delayed for the next consensus process on the next block.

Figure 18: Transaction Process of 24 pre-defined Nodes and four UNLs in a Network System

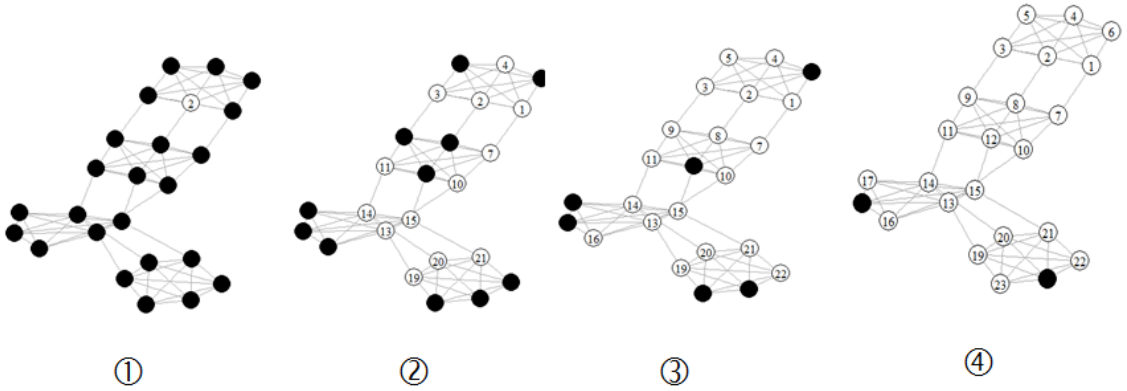



Figure 18 shows the verification process of one transaction in a setup of 24 nodes and four UNLs. The black color signifies unconfirmed notifications of the nodes, while the white colored nodes verified the transaction as valid. At first the node leader, here number two, gets the transaction, verifies it, and relays it to other pre-defined nodes within the UNL environment. In the first round, it requires 50% of accepted votes to send the transaction to the next round. In the second consensus round, the third number in the figure, the bound increments by 10%, thus 60% of positive votes are required. In the final round, consensus needs at least 80% of agreement of a server's UNL meaning the transaction can be added to the "newAddedBlock".

Correctness implies that a distributed system should make a difference between a valid and a fraudulent transaction. Based on the algorithm a valid transaction

can be achieved as long as a UNL of $n$ nodes within the system has the following property:

$$f \leq (n-1)/5. \tag{1}$$

The parameter $f$ defines the number of Byzantine failures. As long as this property holds, the consensus protocol can be maintained. Another main goal is the agreement argument. In case there is only one possible solution in a decentralized system, this can be referred to the agreement requirement. To reach the requirement, every nonfaulty node has to achieve consensus on the same list of transactions irrespective of its UNLs. In the original white paper of Ripple, a fork is an option as long as the UNL is smaller than $0.2 * n\_total$. The parameter $n\_total$ is the number of all nodes in the system. To prove the agreement requirement, there is an upper bound:

$$\mid UNL_i \cap UNL_j \mid \geq 0.2\, max(\mid UNL_i \mid, \mid UNL_j \mid) \vee i, j \tag{2}$$

The upper bound makes sure two sets of UNLs cannot reach consensus on conflicting transactions because the 80% threshold cannot be passed. However, Armknecht et al. (2015) investigate a possible fork scenario in the underlying Ripple algorithm and proved that forks are not possible in the system if and only if:

$$\mid UNL_i \cap UNL_j \mid > 0.4\, max(\mid UNL_i \mid, \mid UNL_j \mid) \vee i, j \tag{3}$$

This means instead of the 20% upper bound as presented by Ripple developers, two intersected nodes of UNLs needs to have more than 40% of shared connections such that forks are not possible. The developers of Ripple agreed to the proved statement of Armknecht et. al (2015) and in the demonstrated example of figure 17, the intersection amounts to 50% such that a fork is impossible in the algorithm protocol of ChoiceCoin as well.

In this context utility can be defined as the latency of the network. The Ripple algorithm differentiates utility into convergence, and heuristics and procedures. The convergence argument verifies the consensus will terminate on a finite time schedule. Convergence is the point when strong correctness is achieved. In other words, equation (1) and (3) have to be fulfilled meaning consensus is satisfied in finite time. The crucial element for the algorithm termination is the communication latency between peers. The time to reply between nodes is screened and nodes that have a larger response latency than a preset threshold $b$ are deleted from all UNLs. Adding to the convergence component, there are more heuristics and procedures supporting

the utility of the Ripple algorithm. To build the initial transaction list on each pre-defined node, there is a two second window to make a proposal in each round of consensus. This will guarantee participation of nodes with solid latency. In the beginning UNLs will be determined by the central entity. This default list can be changed by the users at a later point in time if favored based on a voting decision.

Schwartz et al. (2014) mention that consensus can also be reached with solely one round. However, it might lack utility improvements. Several rounds with an increase of the needed minimum threshold of agreement in percent provide utility gains because high-latency nodes can be identified more easily. By processing only one round of consensus, it could be only some transactions satisfy the 80% bound and also slow-acting nodes can survive leading to lower transaction rates for the whole system.

The RPCA can handle transactions in a few seconds and it depends on the required time to accomplish a round of consensus. The algorithm is not the strongest result for Asynchronous Byzantine failures with only tolerating a default rate of 20%. Nevertheless, it can provide a quick and cheap network with sound safety and reliability properties.

**Critique on RPCA**

Nevertheless, the Ripple algorithm faces some weaker points. Todd (2015) mentions the optimal choice of UNLs is not answered properly and the first best option of a node should be to stick permanently with the default starter UNL provided by the central entity. Additionally, there is no stated incentive why a node should process a validation service at all. The Ripple algorithm does not possess a compensation mechanism with transaction fees or a mining process of new coins. Thus, to reduce potential risk for legal issues or financial crime, Todd (2015) proposes making the validation process on the Ripple ledger private. Armknecht et al. (2015) address the privacy and anonymity issue of the Ripple algorithm. Similar to Bitcoin, Ripple relies on pseudonymous signatures of the users to provide a limited degree of privacy. In addition, the open payment system with publicly announced transactions limits the privacy argument in the network and privacy could be improved in the Ripple system. The centralized element of the Ripple system is a further issue. While there has to be a central authority to provide a permissioned network and maintainenance, a lot of validating servers are also managed by the central authority called Ripple Labs and diminish the decentralized approach in the verification process tremendously.

### 2.4.4 Further Properties of the Created Cryptocurrency

In the following a few key properties of ChoiceCoin will be scrutinized in greater detail.

#### Cryptography

ChoiceCoin will use private-to-public key cryptography. First, Chaum (1983) comes up with an early draft of cryptography in 1983. With this approach, cryptographic elements can only be calculated in one way and are irreversible. Cryptography ensures the creation of digital secrets and fraud resistant digital signatures. Any user of the network will have a key pair. The key pair has a private key for each user to sign spending transactions of its account balance. Based on the private key, a unique public key will be generated which is used to receive the funds. However, ChoiceCoin only uses signatures. The ingredients on the blockchain are not fully encrypted, yet. The property of signature hashes is smilar to Bitcoin or Ethereum. So far, this approach ensures only pseudonymous transaction operations but no full anonymity of the transactions and its participants. The public key can be recognized in the network and all participants are able to see it, while the real world identity of the users cannot be traced back.[7] The algorithm used for the private key hash creation is based on the SHA256 hash algorithm. While for the public key creation elliptic curve multiplication is used for cryptographic public keys and it is based on the secp256k1 approach which can be found in Bitcoin as well.

#### Coin issuance and money supply

A further key property of ChoiceCoin consists of the handling of the coin issuance and the money supply. Optimally, the new issued coin should satisfy two goals: support inclusiveness and maintain a stable store of value. Even under ideal circumstances, most cryptocurrencies face two kinds of inflation. The first form of inflation refers to price inflation meaning an increase of the general price of level of goods and services in an economy. The second kind of inflation can be related to the money supply due to the implemented issuance process. Ali et al.(2014) emphasize that many digital currency forms have pre-set money supply paths regulated by protocols and a fixed total supply in the end. They argue that this procedure can have some problems from a macroeconomic viewpoint. Some examples following the rigid money supply are deflation of goods and services or increased volatility in prices with possible welfare destruction.

For instance Bitcoin limits its coins to a 21 million fixed value covered by its algo-

[7] see Appendix for source code based on Python programming language of creating private and public key hashes

rithm until around the year 2100 and reducing the coin mining reward approximately every four years by half. This mechanism fosters deflationary tendencies. Litecoin will issue 84 million coins while pursuing the same approach of increasing coins in the system as Bitcoin. In the beginning of cryptocurrency launches, many new coins were pre-minted. For instance Ixcoin had 580,000 pre-minted coins, or SolidCoin 30,000 pre-minted coins meaning the developers minted the coins themselves for a period of time before the coin went public for all to mint. However, the pre-mining process cannot be seen as a fair procedure for all participants in the network and new issued cryptocurrencies abandoned the approach quickly.

The total supply and the issuance properties of Ethereum's coins, Ether, is not finite. The creation of its tokens is restricted to 18 million per year. While the total issuance is fixed, the monetary inflation component will decrease every year. Theoretically, with this approach at some point in the future, the rate of new created Ether will equalize the calculated loss of misuse such as lost keys, tokens sent to wrong addresses or also the death of users. But as aforementioned, the Ethereum network is going to change its consensus algorithm from a PoW to a PoS method called Casper. The switch signifies more efficiency and a need for less mining capacities. With this change the issuance may be altered, but it is still under investigation for future research.

Ripple followed a different method for the coin issuance and supply compared to other cryptocurrencies. The token called XRP has a total supply of 100 billion coins and was created instantly at its inception in 2013. There were a lot of discussions regarding how to approach the issuance and supply issue. Ultimately, Ripple did not issue all coins to the public at once to avoid market flooding and uncertainty regarding the price volatility. In May 2017, Ripple still owned around 60% of XRP tokens and announced that it will put more than half of its XRP supply, 55 billion, into escrow by the end of the year. Furthermore, around 1 billion will be given out to the users each month. This communication approach should restrict uncertainty and provide long-term stability for the immediate future.

Referring to other cryptocurrencies' examples, there is no universal concept dictating how to give out tokens the best way. Only the respective market capitalization of the total amount of coins is known. ChoiceCoin will have 42 million coins in total at the beginning. Out of these 42 million coins one third meaning around 14 million will be given out in an initial coin offering to all registered participants and the pre-defined nodes. The remaining 28 million coins will be given out such that the inflation can be assessed reasonably. ChoiceCoin will orientate at the procedure of Peercoin. Peercoin gives out approximately 1% of coins per year to satisfy the

inflation argument and uses the PoW meachnism to reward the miners and increase thereby the coin supply. In ChoiceCoin this increasing supply procedure with the help of miners is not possible since no PoW method and no miners are needed. Nevertheless the approach will be based on a lottery system. On the one hand, 50% of the new issued coins will be handed over to the pre-defined nodes, while the other remaining 50% will be given out to all registered users of the cryptocurrency network. By using this method the pre-defined nodes also have an incentive to process validation services honestly.

Those users in the network must have done a transaction before and should be in the network for at least half a year, since the coin issuance will be based on a 6 month interval. In other words, the coin issuance will take place twice per year and the users must represent a reliable constituent to avoid inflationary account creation on the platform and possible misuse of the coins. The total money supply will be reached in the year 2128 if no modifications will be pursued. The presented idea with the cap of 42 million coins is only hypothetical so far and it can be changed anytime because the coin issuance and the money supply are not hard encoded in the algorithm and it can be modified if desired by the pre-defined nodes in a voting proposal to the central authority to provide a maximum degree of flexibility.

**Transaction fees**

Following the original transaction procedure in the blockchain explanation, there is no transaction fee introduced in the transaction process for ChoiceCoin. However, a transaction fee will be included. The main reason is to incentivize the pre-defined nodes to validate transactions properly. Many digital currencies have an advantage of lower transaction fees compared to existing electronic payment services with credit cards or international transfer payment systems. Especially, cryptocurrencies with a mining mechanism, which often times have a subsidy for the miners included. Bitcoin is a representative example of that process. The subsidy of the process results in new minted coins for the miner. The degree of the subsidy can have two sources. The first one is the actual price of the currency. The other reason depends on the belief of the mining entity about the currencies' price in the future. The additional revenue gives miners the possibility for accepting smaller transaction fees- below the marginal cost of confirming a valid block of transactions. On a short-term basis, the subsidy with the new created currency enables an incentive for the miners to support and help making the currency more established. In the long term the supply of money is fixed and the mining process with the subsidy mechanism will disappear. Those currencies will face competition with other payment networks on the cost issue. In this case, higher marginal costs of cryptocurrencies could cause

a competitive disadvantage to centralized systems which can rely on economies of scale.

In Solidcoin, the fee amounted to always one coin irrespective whether the transaction was only worth one coin or several coins in total. In Bitcoin, the transaction fee is calculated on the magnitude of the transaction in kilobytes, whereas in ChoiceCoin the transaction fee will depend proportionally on the value of the transaction. For instance a bigger transaction will have a bigger transaction fee. The standard fee will amount to 0.5 - 1% of the total transaction size at the beginning and can be changed anytime if desired by the pre selected nodes. Currently, the average credit card processing fees are between 1.5 to 2%, while Paypal charges around 2% on average for its service. The reason for offering such a small fee is that ChoiceCoin can handle from 1000 to 1500 transactions per block in a short period of time. With that amount of transaction handling, the nodes can rely on the economies of scale generating revenue that is higher than the marginal costs for the nodes. Nevertheless, the pre-defined nodes should not prioritize transactions with greater fees. Thus, they should treat each transaction with the same degree of importance without the influence of the fees itself. The receiving node which gets the transaction order will collect the fee before adding the transaction to the transaction list. Amid pre-defined nodes, a second transaction fee method is possible such that each pre-defined node will get a pre-determined fee for each consensus round. However, this method can be introduced at a later point in time if some nodes want to switch to this approach in the network.

# 3 Anonymity Schemes from an Economic Point of View

Following Acquisti et al. (2003) an anonymous structure cannot be built by the sender or recipient in a general system. A user is not able to decision the sending of anonymous messages. The anonymity depends on the infrastructure and the distribution of trust to the underlying network. A node will be running in a shared network if the associated incentives are greater than the marginal costs of supporting the reliability of the network. Nevertheless, the associated costs are immense for the running nodes. Besides the bandwidth and processing power, the costs also include the reputation risk of the nodes and the right choice of selecting valid transactions from fraudulent ones. Hiding messages is a crucial part in anonymous networks. The senders consume anonymity and provide the cover traffic that builds the anonymous structures for other participants. Users have an advantage on broad systems due to

the noise provided by others.

A strong anonymity system relies on high traffic in the network, while the high traffic can imply a better performance. The better performance is explained by a smooth process of messages without any delays that is required on a light traffic network to provide anonymous schemes. However, networks with high traffic do not create the best option to hide at all, because if the degree of trust is too centralized an extensive system can be targeted by insiders and attackers. The attacks can be various. For instance attackers can chase for the efficiency or reliability of nodes or raise the cost of the running nodes. Back et al. (2001) make clear that an anonymous network has to take into account the security objectives that consist of an efficient, reliable and usable approach with the costs included. The security aim influences the size of the users in the network and it makes a reference to the possible degree of anonymity. For decentralized systems similar to existing cryptocurrencies' approaches, Acquisti et al.(2003) find out that providing a solid level of trust, those systems can have unbearable coordination costs. Those costs could only be faced if a central authority can be implemented. Nevertheless, a central authority could be vulnerable to attacks exploiting the trust argument.

Privacy and anonymity are also two different components in this environment. Ishai et al. (2006) make clear that anonymity specializes on hiding the responsible users that performed the action. The privacy argument focuses on what action has been performed in the network.

Cryptography does not enable anonymity and privacy automatically. Privacy has been one goal of cryptography for a long time. Chaum (1985) discusses anonymous digital cash using blind signatures. While many cryptocurrencies with a decentralized structure, take Bitcoin for instance, do not possess true anonymity. Their transactions are recorded on the public blockchain. However, the true identity of the users is not available because only the pseudonymous addresses with hash keys are used. Additionally, if a user transaction has been identified all of that user's transactions could be unveiled if they do not use a new pseudonymous address every time they make a transaction. In practice, Bitcoin offers only weak anonymous schemes. In general, Blockchain anonymity and privacy are not easy to reach especially for public blockchains because those ledgers should make all transactions visible and should generate the verification of the supply of coins. Anonymity and privacy mechanisms have to present a solution to both issues that can end up in conflicts between the protection of privacy and the maintenance of public verifiability. For these reasons, this thesis will include a discussion regarding additional anonymity and privacy structures from already existing cryptocurrencies.

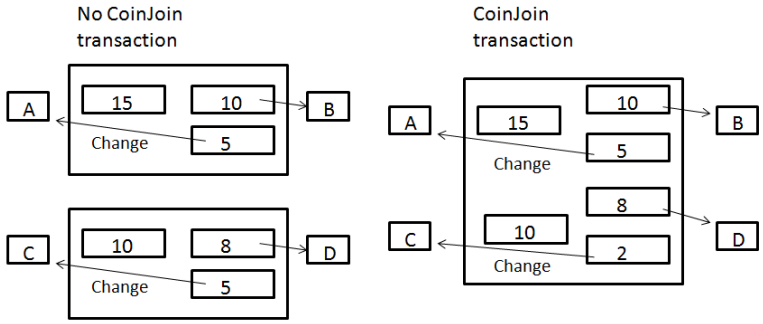
## 3.1 Mixing the Funds

One possibility that can be used is to mix the funds with other participants. Imagine there is one group of users and each user will give the same amount of tokens to a bucket. After that the tokens in the bucket will be mixed and each user will get back the same starting value of tokens. The incentive behind this approach is that mixing tokens will make it more complex to assign the original ownership of the tokens. Following that logic, this procedure will generate some level of privacy. Existing cryptocurrencies relying on the mixing approach are Dash and PIVX. However, PIVX announced it will change to the Zerocoin privacy protocol in the future.

A clear advantage of this method is in its implementation. The method can be applied on top of various cryptocurrencies while there is no change of the consensus mechanism required. In addition to that the method is easy to implement into existing cryptocurrency ecosystems. On the other hand, one disadvantage involves the permanent accessibility of the mixing entities meaning the mixers have to be online all the time. Furthermore the first implementations of this method needed to rely on trust to the third party doing the mixing and not stealing the coins.

CoinJoin is a procedure to mix transactions and improve the anonymity. In Bitcoin, a normal transaction consists of inputs and outputs and those transactions can be seen on the public blockchain. Thus, this concept only provides a low degree of anonymity(see Figure 19). With the help of the CoinJoin mechanism, it is possible to mix all inputs and outputs and add them as one transaction to the block. There are still inputs and outputs available from the sender to the receiver but there is not really a transaction because in this case all participants made payments.

Figure 19: Process of CoinJoin




The concept of CoinJoin is an advanced form of the mixing approach because stealing the tokens is much harder. Böhme et al. (2017) study the implementation of CoinJoin into the Bitcoin system. According to them, money laundering can be pursued on this platform using bitcoins and the authors claim it is the cheapest form of money laundering. CoinJoin transactions consist of secure transactions for

honest users but lack the trust element regarding market hierarchies simultaneously. The trust element conflicts with the anonymous execution of transactions as these are getting a design target next to security. But recent research by Goldfeder et al. (2017) states that a users's wallet can be made transparent even though several rounds of CoinJoin mixing have been performed. The reason is the browser cookies of the users. The mixing procedure can disguise the transaction path between addresses but it cannot break the link entirely.

Important to notice is that in Bitcoin the CoinJoin procedure is not part of the protocol. In Dash, the CoinJoin element is a component of the underlying protocol and it is built on the Bitcoin software. Its main function is to provide instantaneous transactions with a high degree of anonymity. Furthermore, the digital currency has a two tier network with so called "Masternodes". Dash implemented "Masternodes" which are responsible to offer a decentralized governance system. Every node can become a "Masternode" as soon as the node proves to be the owner of 1000 DASH coins. The "Masternode" can vote on essential issues and new proposals and if passed the new issues and proposals are introduced on the blockchain immediately. A further feature of Dash is the “InstantX” function. This function assures that “Masternodes” will send transactions in real time because they can reach consensus internally whose inputs is reserved for the respective output. Moreover, the Dash system is governed de-centrally by the blockchain via "Masternodes" and its voting system introducing new proposals.

The core feature providing anonymity in Dash is known as the “Darksend” function. "Darksend" is based on the CoinJoin concept with additional improvements. This network does not need a trusted third party because the mixing service will be provided within the currency system. "Darksend" needs at least three users to start the mixing process.

A stronger level of anonymity can be assured if the transactions use denominations of 0.1 DASH, 1 DASH, 10 DASH and 100 DASH. All users should state their inputs and outputs in the proposed denominations while the collection of transaction fees should be charged in a separate way. The mixing protocol has a limit of 1000 DASH tokens per session. The intervals are pre-determined and a user will give out requests to join other clients with the help of "Masternodes". Before the user's transaction can be a part of the "Masternode", a queue object is published to the system. This object will include all necessary details including the user's desired denomination. Moreover, stronger anonymity can be achieved if multiple sessions are pursued. The concept of providing multiple sessions can be seen as a chain approach in which the transactions are transmitted to several "Masternodes" resulting in a relaying

network to increase the protection of the user's identity.

"Darksend" improves the privacy element of the Dash ecosystem and the fungibility of its coins. On the other hand, the mixing protocol needs available users who are willing to mix for the required denominations otherwise there will be no mix. In addition to that, the anonymity argument has a limit to the amount of participants in the mixing process. A typical mixing session needs at least three users with repeatable processes to increase the anonymity.

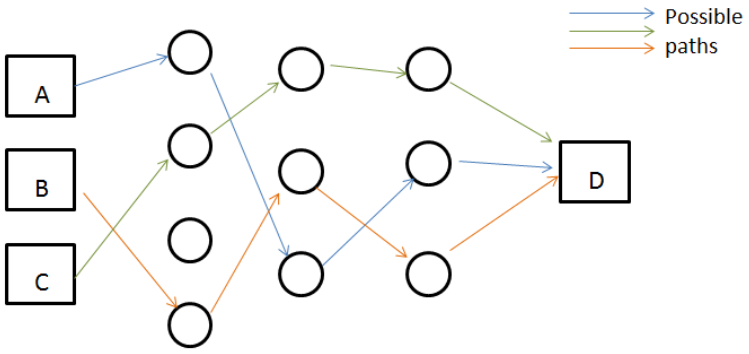
## 3.2 CryptoNote and Ring Signatures

Another opportunity to provide anonymity is through CryptoNote and ring signatures. A ring signature depends on a group of users. Inside this group someone proved that they signed the transaction but it cannot be stated which user was the signor. Anyone can verify the signature but it is only a random guess to acknowledge which user really proved the signature initially. Van Saberhagen (2012) emphasizes two main properties inspired by Okamoto and Ohta's (1991) description of ideal electronic cash that the implementation of CryptoNote has to satisfy. It should be untraceable and unlinkable regarding the transactions. Untraceability means that for every incoming transaction all senders have an equal probability. Unlinkability refers to outgoing transactions and the proof of sending it to the same user is not possible meaning if there are two transactions with receivers $A$ and $B$, it is not possible to identify them if $A = B$.

The process of CryptoNote is also based on ring signatures. A user can make a transaction and take the outputs of available similar transactions on the blockchain to create the inputs to a transaction based on ring signatures. With the ring signature, the input verification of one user can be made non-linkable to the user which is performing the transaction. This process is done automatically in the protocol without any extra notifications to the users. The prevention of double spends can be achieved through a "traceable ring signature". It does not allow the owner of a token to give a signature on two ring signatures with an equal public key and no notifications on the blockchain.

Existing cryptocurrencies that base their anonymity argument on CryptoNote or ring signatures are Aeon, ByteCoin or Monero. ByteCoin was the first one that introduced the CryptoNote protocol in its implementation in 2012. Since the launch, the protocol has gotten updated many times, for example a multiple signature transaction feature has been added. A further approach and the most famous digital currency with CryptoNote technology can be investigated with Monero that im-

Figure 20: Process of CryptoNote




plemented the Ring Confidential Transactions (RingCT). Aeon was introduced in 2014 as a revised version based on Monero and uses the so called CryptoNight-Lite algorithm in its protocol.

On the positive side, the technology does not require a mixing entity. An additional advantage is the well established privacy of the technology with proven records. Only inappropriate implementations have lead to attacks to unveil the anonymity within the system. A further advantage is the increase of anonymity with more time elapsed in the ecosystem. The reason is that the outputs will result in the new inputs of the next mixing section. A higher transaction size results in a clear disadvantage of using the CryptoNote technology due to lower transaction scalability on the blockchain. There is also an increased danger of deanonymization with incorrect implementations of the technology. For instance, a digital currency known as Shadowcash did not implement the technology properly and its blockchain could be de-anonymized completely. A further risk factor is the fact that those approaches cannot be integrated directly on current cryptocurrency networks and need a separate ecosystem to run. Furthermore, the ring signatures have a limit in their ring size in most cases. Thus, the anonymity is curbed by the number of users in the ring.

Monero, implemented in 2014, is a decentralized cryptocurrency and is not based on the Bitcoin protocol but on CryptoNote protocol. Privacy, fungibility and untraceability are additional properties of Monero. Following Noether (2017), the CryptoNote process is vulnerable in particular to two forms of attacks which can unveil anonymity. The specific amount on a given transaction can increase the possibility of de-anonymizing the sender of that transaction because it will be easier to find out which user paid that amount at a given time. The second attack involves the needed transaction properties. A pair $(P,A)$ consisting of a public key $P$ and an amount $A$ is required in ring signatures, while other public keys can have the same

amount. Therefore, regarding privacy, Monero relies on ring signatures, RingCT and stealth addresses.

Those three technologies should ensure that the sending – and receiving part and the amount within transactions are hidden. The transaction is private by default on the Monero ecosystem. The RingCT is based on so called “Confidential transactions" (CT) and it is a mandatory feature for all transactions since September 2017. The prevention of double spending is assured via a "Multilayered Linkable Spontaneous Anonymous Group Signature" (MLSAG). The combination of CT and MLSAG with ring signatures ensures multiple inputs and outputs, anonymity, and the prevention of double spending. Untraceable amounts, origins, and destinations are provided for transactions on the RingCT protocol. Moreover, the coin creation can be pursued on a PoW mechanism and it is thereby trustless with a verifiable security. On the other side, the ring size of Monero is limited to four to curtail the size of the transaction. Thus, the anonymity depends on the number of users in the ring and it might be possible to calculate the probability of the linked transactions. Furthermore, hiding transactions can have negative effects on the verification of new coin creation and the knowledge of the existing coin supply. If anyone can get access to the discrete logarithm of the RingCT protocol that person would be able to create as many coins as desired. However, as of yet no such attack has been noticed.

## 3.3 Zero Knowledge Proofs

Goldwasser et al. (1985) are some of the first to come across zero knowledge proofs. Originally, these proofs can be seen as a convention of two computer programs. One constituent is known as the “Prover”, while the other part is called the “Verifier”. The "Prover" tries to convince the "Verifier" that a statement is true. Furthermore, there are three properties a zero knowledge proof has to satisfy. The first property is completeness meaning that an honest "Prover" will convince a "Verifier". Soundness is the second property and only given a true statement, a "Prover" is able to convince a "Verifier". Finally, zero-knowledge is required. In other words a "Verifier" does not get any additional information than that the statement is true. In this context a zero knowledge proof is a proof that a user has knowledge about something without giving out extra information regarding how the knowledge has been acquired. The cryptocurrency Zerocoin and its token Zcoin is built on the concept of zero knowledge proofs. As stated before, PIVX will also apply the Zerocoin method in the future.

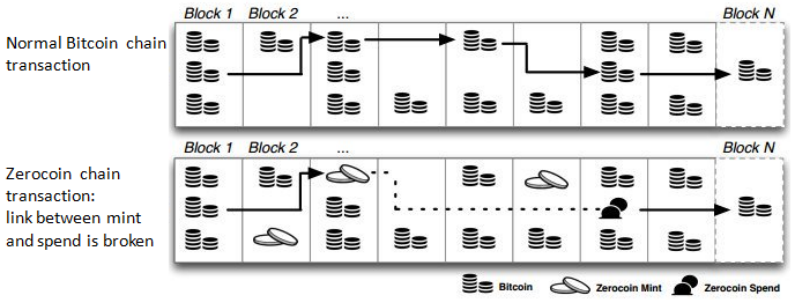
An obvious advantage of the following approach is that the procedure does not need any mixing entity. Moreover, a high level of anonymity can be guaranteed. The

anonymity does not depend on the number of users or the ring size. Every user can use the anonymity set of a used denomination and apply it achieving much higher scalability properties regarding transactions. All that is needed is just one mining –and spending process, while the transaction linkage between addresses breaks. Additionally, the concept of zero knowledge proofs can rely on a well established research history. Maintaining supply auditability is a further positive property.

On the contrary, the proof sizes are quite large and can increase the verification period by several seconds affecting the transaction scalability negatively. Besides the transaction scalibilty, the question is also where to store the proofs. The zero knowledge proof approach needs a trusted setup to implement initial parameters. If those parameters are unveiled, attackers can infiltrate and leak into the system performing actions such as creating extra coins. Another drawback has to do with the deployment of the zero knowledge method. The support of the new mining -and spending functionality requires adjustments on the existing protocols in the cryptocurrency ecosystem.

Green et al. (2013) describe Zerocoin as a distributed e-cash system that extends cryptographic techniques to existing coin protocols, such as Bitcoin's. Zerocoins can be exchanged one-to-one with bitcoins. Any user is able to buy a Zcoin in exchange for the correct amount of bitcoins. This is done via a new Zerocoin mining transaction on the blockchain. It is important to note that the link of a Zerocoin mining transaction and a Zerocoin spending transaction is completely broken through zero knowledge proofs. When re-acquiring a Zcoin, one will get a totally different bitcoin than the one that was used to purchase the Zcoin initially. It is difficult to determine at which place the user took out the conversion due to the mixing and creation of Zcoins with all other users in the system.

Figure 21: Process of Zerocoin by Green et al. (2013)




Zerocoin uses three concepts: digital commitments, a one-way accumulator and zero knowledge proofs. The structure includes four core randomized algorithms. Those algorithms can be grouped into "Setup", "Mint", "Spend" and "Verify".

In the following $\lambda$ indicates an adjustable security parameter and $C$ represents a set of allowable token values. For the "Setup" algorithm: Setup$(1^\lambda) \Rightarrow params$. For the input of a security parameter, the output consists of global public parameters $params$ and an explanation of the set $C$. Zerocoin's setup routine is a one-time strong RSA accumulator by Bernaloh and de Mare (1994) and Camenisch and Lysyanskaya (2001) with $N = p*q, u \in QR_N(u \neq 1)$. Its parameters $p$ and $q$ consist of two large prime numbers and are based on the RSA-2048 parameters. They were created and destroyed immediately and are at present resistant even to quantum computing power.

A core algorithm is the "Mint" algorithm: Mint$(params) \Rightarrow (c, skc)$. The input parameter is $params$, while the output is a coin $c \in C$ and a trapdoor value $skc$. For mining one Zerocoin, the user has to spend a base coin. Each Zerocoin is committed to a serial number $S$. The commitment can be seen almost as an encryption process because the newly created coin does not show the serial number, it is secret, and the coin refers to the chosen amount simultaneously. Furthermore, there will be a randomness factor which will be kept secret all the time. The serial number and the randomness factor form a hash value of the new Zerocoin and those tokens will get some value as soon as the hash value can be found on the blockchain.

The next key algorithm is the "Spend" algorithm: Spend$(params, c, skc, R, C) \Rightarrow (\pi, S)$. For the spending algorithm the inputs are $params$, a coin $c$, the trapdoor $skc$, some transaction string $R \in (0, 1)$ and an arbitrary amount of coins $C$. The output is a spending transaction including a proof $\pi$ and a serial number $S$ if $c \in C \subseteq C$. Otherwise the output is nothing. In a spend transaction, the user has to include the created serial number. The spending transaction's main message is its zero knowledge proof. The proof includes the following statements: the mentioned and posted valid Zcoin is on the blockchain and secondly this Zcoin has the serial number $S$ in accordance with the users' transaction.

Finally, the "Verify" algorithm: Verify$(params, \pi, S, R, C) \Rightarrow (0, 1)$. The input properties include the $params$, the proof $\pi$, a serial number $S$, the transaction information $R$, and a setting of coins $C$. The output will be 1 if $C \subseteq C$ and $(\pi, S, R)$ is valid. If it is invalid the output will be 0. The zero knowledge proof is crucial. Anyone in the system can verify a zero knowledge proof because any user is convinced that the respective person minted a Zcoin and it included the respective serial number $S$. The verifier can also check the existing blockchain if that serial number has been spent before. In addition to that zero knowledge proofs allow that the verifiers have no knowledge about the spending amount of Zcoins. It only finds one other Zcoin and converts it back to the base coin on the blockchain, it must not

be the same as it was created. If only one user would apply this approach, there would be no anonymity. The anonymity comes from the zero knowledge proofs based on Zerocoins and pursuing multiple transactions from numerous users resulting in sufficient noise for the system.

Zerocoin provides strong anonymity with no transaction linkages but it needs a trusted setup, more resources to store on the blockchain, and more computational resources for the verification process.

## 3.4 Zero Knowledge Succinct Non-interactive Argument of Knowledge (Zk snarks)

This form builds also on the knowledge of zero knowledge proof schemes with some additional features. The attribute succinct refers to the property that the proof can be verified in a few milliseconds and has a transaction size of only a few hundred bytes. With the structure of non-interactive components, the proof only has one message sent from the proving element to the verifying element. To date, the only way of creating zk snark proofs is achieved through an initial setup that creates a common reference string. The common reference string, also called the public parameters in most networks, is the sharing point of the prover and the verifier. The cryptocurrency Zerocash considers the zk snark technology in its implementation.

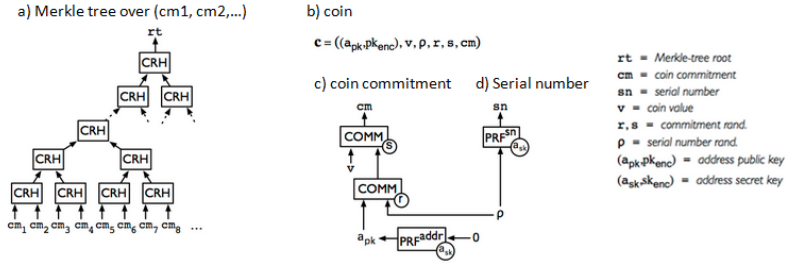
The idea of Zerocash is to improve the ground work of Zerocoin's protocol. Zerocoin can only use pre-determined denominations, while the transaction sizes with around 25kb are quite large. In ZCash and its usage of zk snark technology, the proof sizes are small and allow a fast verification. Another advantage is that it is a very good anonymity set to break the transactional linkages of addresses. Furthermore, the transaction amount cannot be shown and there is no requirement to change the tokens from a base coin to an anonymous one because the coins are exchanged directly.

On the other hand, a trusted setup has to be used as was the case with Zerocoin. However, this setup is structured more complexly. An incorrect implementation of the setup can result in a forgery of coins. A consecutive argument on that point is the difficult detection of the forged coins in the system because of a non-visible supply auditability. An additional shortcoming is the generation of private transactions. It can take almost a minute or even more, affecting the transaction scalability negatively. Finally, the zk snark technology is a relatively new method and lacks a well established academic research background.

Ben-Sasson et al. (2014) present Zerocash to build a separate anonymous currency next to a base currency. In Zerocash's system any user can exchange a base coin into a Zerocash token called Zerocoins and make transactions by sending those coins to other users. The Zerocash algorithm has the following main components: "Setup", "CreateAddress", "Mint", "Pour", "VerifyTransaction" and "Receive".

Two forms of transactions are used in Zerocash's implementation. The first transaction is called "Mint" and gives the user the right to exchange a pre-determined number of base coins for an equal amount of Zcash and a Zerocash address. The transaction includes a cryptographic commitment to a new token and determines its coin value, owner address and a serial number. The SHA-256 hash function is used for the commitment procedure and can make the coin's value and owner address invisible. The nodes in the Zerocash environment remain a Merkle tree for the available token commitments. Because of the decommited values and being a limited witness as a member in the tree itself, every participant can show that he or she is the owner through a coin commitment. However, this ownership proof is not private meaning there is a second transaction to maintain privacy of the user and that they know the information in zero knowledge.

Figure 22: Process of Zerocash by Ben-Sasson et al. (2014)




A private transaction taking some coins from the user to create new coins is done via a "Pour" transaction. The pour transaction is based on the zero knowledge approach, and for instance, for two input tokens and two output tokens it contains that the user has ownership of the two input coins. Furthermore, every input coin was a member of a previous mint transaction or each output coin was part of a previous pour transaction. Additionally, the total sum of input coins has to be the same as the total sum of output coins. This type of transaction ensures that input tokens only show the serial numbers but not any information about the value of input tokens or output tokens, or ownership addresses. The verification process for the mint transaction can be pursued by any user. As stated before, the pour transaction can be verified via zero knowledge proofs, in Zerocash's case it is done by a zk snark-approach.

Definition: An arithmetic circuit satisfiability issue of an $\mathbb{F}$-arithmetic circuit $C$:

$\mathbb{F}^n$x $\mathbb{F}^h \rightarrow \mathbb{F}^l$ is caught by the relation $\mathcal{R}_\mathcal{C} = \{(x, a) \in \mathbb{F}^n \text{ x } \mathbb{F}^h : C(x, a) = 0^l\}$; while its language is $\mathcal{L}_\mathcal{C} = \{x \in \mathbb{F}^n : \exists a \in \mathbb{F}^h \text{ s.t. } C(x, a) = 0^l\}$.

The algorithm of the zk-snark technology for a given field $\mathbb{F}$-arithmetic circuit satisfiability in Zerocash can be distinguished into "KeyGen", "Prove" and "Verify" polynomial timed algorithms. The "KeyGen": KeyGen$(1^\lambda, C) \rightarrow (pv, vk)$. As input components remain a security parameter $\lambda$ and an $\mathbb{F}$-arithmetic circuit $C$. The KeyGen algorithm provides a proving key called $pk$ and a verification key $vk$. Both keys can be used infinitely for the verification and proof process. Moreover those keys are defined as public parameters in $\mathcal{L}_\mathcal{C}$.

The second polynomial algorithm is called "Prove": Prove $(pk, x, a) \rightarrow \pi$. As input parameters there is a proving key $pk$ and any $(x, a) \in \mathcal{R}_\mathcal{C}$. As output, it will give out a non-interactive proof $\pi$ for $x \in \mathcal{L}_\mathcal{C}$. Finally the last component is the "Verify" algorithm: Verify $(vk, x, \pi) \rightarrow b$. For the input attributes verification key $vk$, $x$ and a proof $\pi$, the verifier results in $b = 1$ if the verifier got a conviction that $x \in \mathcal{L}_\mathcal{C}$.

Additionally, the zk snark technology should fulfill the properties of completeness, succinctness, proof of knowledge, and perfect zero knowledge.

## 3.5 Further anonymity schemes

As there is a constant development of new anonymity structures for digital currencies a few more schemes will be presented that do not fit into one of the aforementioned concepts. NAV Coin, created in 2014, consists of a mixture of the Bitcoin blockchain and its own sub-chain. The transactions are encrypted making the sending amount of coins untraceable. The technology of NAV Coin uses splitting transactions generated on a random number. Additionally, the divided transactions are based on several blocks with time delays to provide a maximum level of anonymity. Finally, the transaction parts will be reconnected and separated one more time to create numerous identical transactions on the system.

The cryptocurrency Verge has its origin in the digital currency named DodgeCoinDark. It changed its name to Verge in 2016. Verge builds upon the Bitcoin blockchain and wants to improve the privacy by adding central anonymity networks with Tor and i2P technologies. Thereby, the user's transactions are routed through several global servers. Every server in the ecosystem deletes the information of a previous server such that the last exit node server does not know from which destination the information originally came. Following this procedure, the users can continue with their transactions as normal while Verge makes the currency trail

untraceable.

Showing the fast development in the cryptocurrency sphere regarding privacy and anonymity, Zencash can be presented as a dignified example. In May 2017 Zencash was established by a hard fork from ZClassic, while ZClassic originated from Zerocash. Zencash's main feature regarding anonymity and privacy is built on the zk snark technology as in Zerocash. However, the developers are also considering a few properties from the cryptocurrency Dash such as a decentralized governance model and an integrated voting system. Similar to the "Masternodes" in Dash, Zencash will implement "Secure Nodes" which are responsible to run the system and will get a reward of 3-5% from the mining process.

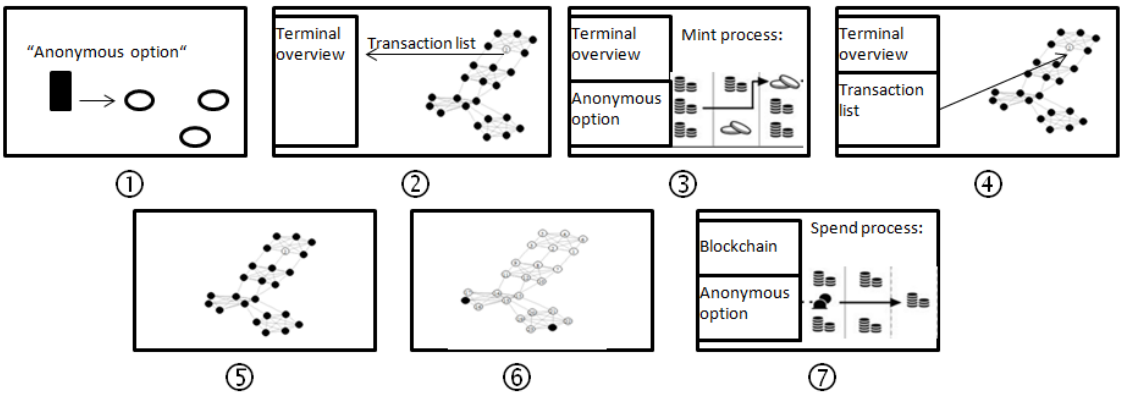
## 3.6 ChoiceCoin's Approach to Anonymity

As aforementioned the anonymity is built mainly on the infrastructure and the distribution of trust to the underlying network. But anonymousness depends also on the user's behavior and motivation. The different examples of anonymity schemes show that its way of implementation has numerous variants and each structure presents advantages and drawbacks depending on the environment and on its main goal. Referring to ChoiceCoin and its modeled structure, several forms of anonymity schemes are thinkable. Mixing the funds is one possibility, whereby the transaction linkage cannot be broken entirely. The ring signature approach is also an opportunity, however, only the different pre-defined nodes could be part of the process limiting the anonymity argument. The zk snark method is also feasible, but the technology is quite new and lacks a well established research history, meaning it could have undiscovered flaws which could de-anonymize the environment. Thus, the anonymity scheme with the zero knowledge approach as used in Zerocoin will be implemented.

This method can be implemented on the existing protocol, the security parameters can be created by the terminal and the central entity can pursue the mining transaction. Furthermore, the zero knowledge technology has a proven research record while generating a high level of anonymity with unlinkable transactions. Additionally, even if pre-defined nodes want to send anonymous transactions to each other it is possible with the untraceable properties.

The process of an anonymous transaction works as follows: at first, a user will approach a pre-defined node and will allow all transaction details to be known including the anonymity option. Next, the node gives the transaction list including the desired anonymous transactions to the terminal overview. On the third point,

Figure 23: Process of optional Anonymity




the central entity prepares all transactions and performs a mint transaction for the anonymous option to transform it to a Zerocoin token. After the mining transaction, the terminal overview transmits the anonymous transaction back to the current node leader. However, the mining process might take longer than the normal process of checking transactions by the central entity and the anonymous transaction might be part of the transaction list after several rounds. In the next step, the node leader verifies the transaction and relays it to the other nodes including all the other transactions for the following transaction round. After the verification procedure, the anonymous transaction will be appended on the blockchain and stays there. Finally, in the spending process of the anonymous transaction, one ChoiceCoin token can be redeemed on the blockchain with a serial number made public, performing a zero knowledge proof, and a random Zerocoin.

A user can make an anonymous transaction with Zerocoins any time, for instance a few moments before an actual transaction to obfuscate the transaction link even more. For the spending transaction all that is needed is the serial number and all members of the network. The recipient of the base coin only knows it can be from one of the many participants who commissioned a Zerocoin mint process without unveiling the true identity. Furthermore, there will be a fixed denomination rate of 0.25 Zerocoins in a one-to-one exchange with the existing coins in the ecosystem. For instance, a transaction of five Choicecoins can be made anonymous, then the user will give a normal transaction to one of the pre-defined nodes. After that, the terminal splits up the transactions in this case to 20 individual transactions with 0.25 Zerocoins each. The user can activate the spend process as soon as the transactions are on the blockchain.

To be more precise in a mint process, a hash value $H$ will be computed by the central entity. The hash value $H(S, r)$ consists of a serial number $S$ and a random secret $r$. The serial number will be made public in the spending process later on.

However, the random secret $r$ will never be made public to ensure unlinkability. After the mint transaction and the verification process of the pre-selected nodes, the hash value will stay on the blockchain. On the blockchain, only the hash value $H$ will be visible. The input values with the serial number $S$ and the random secret $r$ including the recipient address are invisible.

The user who wants to execute an anonymous transaction can wait as long as desired because its Zerocoin is on the blockchain with the respective hash value $H$ and can contact the pre-defined nodes or the terminal to spend it. To redeem a transaction into ChoiceCoins two components are required: First, the serial number $S$ has to be included which has been generated and not been spent before. The second requirement is to create the zero knowledge proof which includes that the users know a number $r$ such that $H(S, r)$ is one of the Zerocoins on the blockchain. The randomness $r$ combined with $S$ picks an arbitrary Zerocoin on the blockchain and uses it as the input in a new ChoiceCoin transaction.

The mint -and spend transactions break the links of the base coin completely and the process can be seen as a huge laundry network. Selecting an arbitrary Zerocoin in the spending procedure determines the anonymity in the system as soon as more than one user relies on the anonymous option regarding transactions. Since $r$ is a secret no one knows which Zerocoin corresponds to the serial numbers. Even after serial numbers have been revealed and it was inside the hash value $H$, nobody knows which hash value it really was. Moreover, the presented approach should be in accordance with anonymity and privacy mechanisms that have to present a solution to both issues between the protection of privacy and the maintenance of public verifiability. The reason is the public verifiability is reached through a permissioned blockchain and the privacy is not mandatory but it is a possible option for each participant in the system. The users can weigh the timing and anonymity argument individually and it allows them to make a choice while providing a flexible approach for the presented ecosystem of ChoiceCoin. It is thinkable to change to a complete anonymous transaction scheme as soon as the transaction scalability can reach the same level as without the anonymous option through a voting decision by the pre-defined nodes.

# 4 Conclusion

One of the main goals of this thesis was to build a cryptocurrency from the ground up with specific properties. Three key components are required for the creation: a blockchain, a currency and a protocol. Within those boundaries the design of a

cryptocurrency can have various spectrums. The core properties as presented in ChoiceCoin are built on a private blockchain with 24 pre-defined nodes to generate fast transactions, a consensus mechanism based in principle on the Ripple algorithm and an optional anonymous transaction possibility created within the protocol. Furthermore, the terminal overview providing a flexible adaptation to change the set of rules via a voting system by the nodes, and the node leader component configure ChoiceCoin with unique details that other cryptocurrencies do not offer directly.

A centralized approach as seen in ChoiceCoin has a key vulnerability: if a central authority, in this case the terminal overview, is not available, the transaction process cannot be executed. That is a risk factor that all participating users have to reconsider. Furthermore, one aspect remains whether it makes sense to introduce a blockchain for ChoiceCoin or if a distributed databank is more usable. For Bitcoin, a blockchain is a reasonable design for a group of un-trusted nodes, no central authority and almost no possibility to shut it down by one or several attacks on its network. The cost of maintenance and support, the limited performance and the fault tolerance factor affecting the robustness of the ecosystem are clear properties that influence the decision making of choosing a blockchain. Blockchains are superior in the provision of a robust and fault tolerant system, while its shortcomings can be found in a limited performance compared to a databank. For ChoiceCoin, it is legitimate to think of introducing a SQL databank. However, as the terminal overview already presents a centralized property and to provide a maximum degree of decentralized approach for the underlying network, a blockchain is more reasonable than relying on a databank. One reason is that the 24 pre-defined nodes should verify the transactions and update its blockhain themselves without any dependency on a central authority. Furthermore, if one node cannot be reached, there are sufficient nodes available and up-to-date to execute the verification process. The robustness aspect is favored over the performance component as the blockchain's development and research is still at an early stage and it is highly probable that improvements regarding performance are possible in the future.

The analysis of anonymous schemes emphasizes that from an economic perspective the network is responsible for providing anonymity and privacy to the participants. The users have to rely on the system and its operating entities. But participants can enhance the anonymity aspect indirectly by encouraging numerous transactions resulting in noise for the system. In the cryptocurrency sphere, each anonymity and privacy structure has its individual set of benefits and drawbacks including respective trade-offs. ChoiceCoin includes an anonymous option for users' transactions based on the zero knowledge proof method pursued by Zerocoin. The reason to provide an optional approach only is the transaction scalability which takes longer if a user

wants to make an anonymous transaction. Furthermore, anonymousness depends also on the user's preferences and offering an option between choosing anonymity and fast transactions will result in a more flexible usage.

A faster anonymous transaction is also possible, for instance with the zk snark procedure. But the technology is not well established, yet and as anonymity can only be provided by the implemented network, the reliability aspect cannot be guaranteed as a core property for the underlying ecosystem. Thus, pure zero knowledge proofs have a detailed research history and a proven record as a resilient and stable technology among others for providing anonymity and privacy.

# A Appendices

## A.1 Glossary

**51% attack:** A mining pool is able to control 51% of the mining power (hash rate) meaning it can include its own blocks into the blockhain. Additionally, it can pursue a fork to make an independent branch that can merge with the main blockchain part.

**ASIC:** It is called Application Specific Integrated Circuit (ASIC) and it is a chip that is custom-designed for just one application for instance Bitcoin or Ethereum mining.

**Asynchromous distributed system:** In those systems messages can be delayed for an unlimited amount of time. The period of message transmission is unknown between nodes. Furthermore, there are no strong assumptions on time and ordered events.

**Byzantine generals' problem:** The main idea of this problem is that there are several generals and every general has the command of its Byzantine army to attack an enemy city. However, to execute a successful attack, all generals need to reach agreement on a common battle plan. The generals can only communicate with messengers. It is possible that those messengers might get captured by the enemy and the original message does not arrive the others. The main difficulty of that problem is that some generals could be traitors and are interested in sabotaging the battle plan. Within this environment some generals might send inaccurate messages but the loyal generals should be unaffected tolerating a small fraction of traitors and not adopting the manipulated battle plan.

**Confidential Transaction (CT):** Originally, CT is a cryptographic tool that should improve the privacy and security of Bitcoin. The transferred amounts will be visible only to the users in the transaction and to those that verify it. The cryptographic technology used is called additive homomorphic commitments. A by product is the possibility to transfer private invoice numbers or refund addresses data without an increase of transaction size. The idea according to Back (2013) is to use a Pedersen Commitment.

**GHOST:** Greedy Heaviest Observed Sub Tree. It is a revised version of Bitcoin's PoW and used in Ethereum. The GHOST-protocol was introduced in 2013 to combat the way that quick block time in blockchains suffer from a big amount of orphaned blocks. This protocol includes orphaned blocks, called uncle blocks, and its rewards are 87.5% for normal valid blocks, while the nephew, the child of the unlce

blocks, gets 12.5% of the rewarded block.

**Hash rate:** The total amount of existing gear in the network to solve puzzles/ tasks in a PoW mechanism. In Bitcoin, the hash level changes over time as miners can join or leave the network. A higher hash rate results in a better network protecting against attacks.

**Invisible Internet Project(I2P):** It is used in network systems. I2P provides increased anonymity with an overlay network. The i2P network is a network within a network. The main goal is to protect the communication within the system from drag net surveillance by additional parties such as ISPs. The users' traffic is encrypted and runs through several thousand computers worldwide. The nearly unlimited possibility of traffic paths makes it harder to follow for monitoring and surveillance processes by third parties.

**Latency:** It is a way of delay publication in the computing -and network system. Latency announces the delay time it takes from an input to a desired outcome within a system. This term affects the usage especially of communication of network system.

**Log replication:** A log is a strictly ordered append-only sequence of numerous operations. In computing, a replication handles storing information to enable consistency between different resources to improve reliability, fault tolerance or accessibility.

**Memory hardness:** It is the ability of a computer to move data around in memory and not with calculations. Furthermore, it is a property that general purpose computer hardware is already designed to perform well but it cannot really lead to efficient results on ASICs. With a resistant algorithm to ASIC, it can prevent large powerful firms from out taking control of mining power in the Ethereum system.

**Multilayered Linkable Spontaneous Anonymous Group Signature (MLSAG):** The MLSAG following Noether (2017) is a generalized form of Back's Linkable Spontaneous Anonymous Group Signature (LSAG). This LSAG is not covered by keys but by key vectors.

Definition: A key vector is just a collection $\bar{y} = (y_1, ..., y_r)$ of public keys with respective private keys $\bar{x} = (x_1, ..., x_r)$.

In a MLSAG approach, it is supposed that each signer of a generalized ring of $n$ members has exactly $m$ keys:"$\{P_i^j\}_{j=1,...m}^{i=1,...n}$.

The first intention behind the MLSAG is to provide a proof that one of the $n$ signers knows the secret keys to the complete key vector. In addition to that another component sketches the MLSAG signature. If a signer uses one of the $m$ signing

keys in a different MLSAG signature this means both rings are linked and the second MLSAG signature will be discarded. The MLSAG is introduced by an algorithm and satisfies three key properties: Unforgeability, Linkability, and Signer Ambiguity.

**Network File System (NFS):** It consists of a distributed file system and gives users access to a client computer to demand files over a computer system. The NFS has an open standard form and anyone can implement the protocol.

**Nonce:** It is a crypto graphical term and an arbitrary number that is in principle only used once.

**Peer to peer (P2P):** Originally, it is a computer that participates in the network peers and does not have any privileges but equal treatment.

**Protocol:** It describes how communication should work and provides several ways of implementation by using different programming languages and so on.

**RSA-accumulator:** An accumulator is used in cryptography and it is a one way membership application. It will respond to whether a participant is a member of a set while not publishing the individual members of the set entirely. The RSA argument is a public-key cryptosystem by Rivest, Shamir and Adleman (RSA) for data transactions. In this system a user builds and shows a public key based on two large prime numbers while those prime numbers must be kept secret to mitigate fraudulent use of the system.

**secp256k1:** It is a cryptographic component and part to the parameters of the Elliptic Curve Digital Signature Algorithm (ECDSA) curve. The secp256k1 is used among others in Bitcoin and its definition can be found in Standards for Efficient Cryptography (SEC).

**Segregated Witness:** It is an upgrade to the Bitcoin protocol. Its technology separates signature data from bitcoin transactions. It is a soft fork and should make Bitcoins' protocol rules more restrictive.

**Segwit 2x:** It is a software upgrade and improves the capacity of the Bitcoin protocol. The deployment of Segwit2x is signaled by the majority of Bitcoin miners. The expected date of the hard fork was November 16th and it should be done when Bitcoin block 497,784 was mined. However, the date was postponed and the hard fork took place on December 28, 2017 on the block number 501,451.

**Selfish mining:** In this form of attack, the attacker gains a lot of mining power at the cost of his/her short term revenue with the maintenance of an own private blockchain to the existing blockchain. He/She publishes a lot of blocks at once and forces the rest of the network to give away their blocks and revenue. This is an

incentive to honest miners to follow the attacker and to increase its revenue that finally can get the 51% of the network mining power. It is another way of a 51% attack.

**Serenity:** It is a name used in Ethereum ecosystem for the last phase to switch from the PoW mechanism to the PoS mechanism called Casper. The Serenity pahse will be implemented via a hard fork from the official Ethereum blockchain.

**Software Development Kit (SDK):** It is a collection of Software and used for development issues for a device or an operating system. An SDK consists of an integrated development environment for a central programming hub and interface. In most cases, SDKs have a sample code that supports developers with example programs and libraries. The developers can create basic programs via SDKs and it eventually helps them to build more complex applications. Furthermore, SDKs can also include technical documentation or sample graphics.

**State:** All or a part of the data that a program deals with.

**Throughput:** It is a way of measurement in the computing -and network system. Throughput tells the amount of units of information a system can process in a fixed period of time. The response time between single users to request and receive the response, and the speed of workload time are related measurements to the throughput component.

**The Onion Router(Tor):** It is a free software and enables anonymous communication. Tor is an overlay network system and provides internet traffic with the help of several thousand relays. The relays are based globally and try to help making the location and usage of the participants' traffic and network surveillance invisible.

**Virtual machine:** It is a device that actually executes a program or code.

## A.2 Source Codes

The following Python code has been executed on Python version 3.4

- **Python code for the created blockchain:**

```
import hashlib, json, sys

def newHash(msg=""):
# This function helps to facilitate the encoding of the hashing_
algorithm with an "utf-8" approach
```

```
if type(msg) != str:
msg = json.dumps(msg, sort_keys=True)  # the keys will be sorted_
to guarantee repeatability

if sys.version_info.major == 2:
return(hashlib.sha256(msg).hexdigest(), 'utf-8')
else:
return hashlib.sha256(str(msg).encode('utf-8')).hexdigest()

import random
import time
random.seed(0)

def makeTransaction(maxValue=5):
# It will create valid random transactions in the range of 1 to with_
a timestamp
sign = int(random.getrandbits(1))*2 - 1   # This will randomly choose_
-1 or 1
amount = random.randint(1,maxValue)
aPays = sign * amount
bPays = -1 * aPays
# timestamp based on milliseconds
timestamp1 = time.time() * 1000
print(timestamp1)
time.sleep(0.1)
amount = random.randint(1, maxValue)
cPays =  sign * amount
dPays = -1 * cPays
# timestamp based on milliseconds
timestamp2 = time.time() * 1000
print(timestamp2)
# It will always return transactions that respect the conservation_
of tokens.
# However, since now no checks pursued whether these overdraft_
an account
return {u'A':aPays,u'B':bPays,u'C':cPays,u'D':dPays}

txnRawList = [makeTransaction() for i in range(20)]
# The txnRawList will create 20 transactions
```

```
# Send the txnList to the central authority
# The central authority will check the defined set of rules and_
rank the transactions in an order regarding the timestamps

def updateState(txn, state):
# Inputs: transfer amount (txn) and account balance (state)
# Returns: Updated state but no validation of transaction only_
update the state

# If the transaction == valid -> update the state
state = state.copy()  # Creates a working copy of the data.
for key in txn:
if key in state.keys():
state[key] += txn[key]
else:
state[key] = txn[key]
return state


def isValidTxn(txn, state):
# Assumption: Transaction is a dictionary keyed by account names

# Checking sum of the deposits and withdrawals is 0
if sum(txn.values()) is not 0:
return False

# Checking overdraft of transactions
for key in txn.keys():
if key in state.keys():
acctBalance = state[key]
else:
acctBalance = 0
if (acctBalance + txn[key]) < 0:
return False

return True

# Determine the node leader
```

```
random.seed(0)
master_node = random.randint(1,24);
print("The leader node will be the following node:" + str(master_node))
# The node leader will relay the transaction list to all other_
pre defined nodes

# Consensus algorithm
#The consensus mechanism will be explained in the section_
"the consensus algorithm"

# Block creation

# Generating the states and the genesis block
state = {u'A':60, u'B':60, u'C':60, u'D':60}
# Definition of initial states
genesisBlockTxns = [state]
genesisBlockContents = {u'blockNumber':0,u'previousHash':None,_
u'txnCount':1,u'txns':genesisBlockTxns,'timestamp': time.time() * 1000}
genesisHash = newHash( genesisBlockContents )
genesisBlock = {u'hash':genesisHash,u'contents':genesisBlockContents}
genesisBlockStr = json.dumps(genesisBlock, sort_keys=True)

chain = [genesisBlock]
print(chain)

def makeBlock(txns, chain):
time.sleep(0.1)
timeStamp = time.time() * 1000
time.sleep(0.1)
parentBlock = chain[-1]
previousHash = parentBlock[u'hash']
blockNumber = parentBlock[u'contents'][u'blockNumber'] + 1
txnCount = len(txns)
blockContents = {u'timestamp': timeStamp,u'blockNumber': blockNumber,_
u'previousHash': previousHash,
u'txnCount': len(txns), 'txns': txns}
blockHash = newHash(blockContents)
block = {u'hash': blockHash, u'contents': blockContents}
```

```
return block

blockSizeLimit = 4  # Randomly chosen number of transactions per block

while len(txnRawList) > 0:
bufferStartSize = len(txnRawList)

## Bring together a set of valid transactions for gathering
txnList = []
while (len(txnRawList) > 0) & (len(txnList) < blockSizeLimit):
newTxn = txnRawList.pop()
validTxn = isValidTxn(newTxn, state)  # If txn is invalid ->False

if validTxn:
txnList.append(newTxn)
state = updateState(newTxn, state)
else:
print("Transaction is invalid")
sys.stdout.flush()
continue  # If invalid transaction -> ignore it and continue

## Create final block
theAddedBlock = makeBlock(txnList, chain)
chain.append(theAddedBlock)

chain[0]
chain[1]
print(chain[1])

state
print(state)

def checkingBlockHash(block):
# Exception raised if the hash is no match to the block contents
expectedHash = newHash(block['contents'] )
if block['hash']!=expectedHash:
raise Exception('Hash does not match contents of block %s'%
block['contents']['blockNumber'])
return
```

```
def checkingBlockValidity(block, parent, state):
# Checking following conditions:
# Is each transaction a valid update to the system state?
# Is Block hash valid for the block contents?
# Does block number go up by one compared to the parent block number?
# Is the parent block's hash referenced properly?
parentNumber = parent['contents']['blockNumber']
previousHash = parent['hash']
blockNumber = block['contents']['blockNumber']

# Checking transaction validity; if an invalid transaction -> error.
for txn in block['contents']['txns']:
if isValidTxn(txn, state):
state = updateState(txn, state)
else:
raise Exception('Invalid transaction in block %s: %s' %
(blockNumber, txn))

checkingBlockHash(block)  # Checks hashes -> error if not accurate

if blockNumber != (parentNumber + 1):
raise Exception('Hash does not match contents of block %s' %
blockNumber)

if block['contents']['previousHash'] != previousHash:
raise Exception('Previous hash inaccurate at block %s'
% blockNumber)

return state

def checkingChain(chain):
# Checks the chain from the genesis block
# Checks if all transactions are valid, no overdrafts found, and
accurate linked hashes of blocks
# This returns the state as a dictionary of accounts and balances
or False if error
```

```
## Data input processing: Verifying that the chain is a list_
of dictionaries
if type(chain) == str:
try:
chain = json.loads(chain)
assert (type(chain) == list)
except:  # A catch all exception
return False
elif type(chain) != list:
return False

state = {}
## Preparing all by checking the genesis block
# Checking the following conditions:
# Is each of the transactions a valid update to the system state?
# Is the block hash valid for the block contents?

for txn in chain[0]['contents']['txns']:
state = updateState(txn, state)
checkingBlockHash(chain[0])
parent = chain[0]

## Checking subsequent blocks: Additionally checks are needed for:
# The reference to the previous block's hash
# The validity of the block number
for block in chain[1:]:
state = checkingBlockValidity(block, parent, state)
parent = block

return state

checkingChain(chain)

chainAsText = json.dumps(chain,sort_keys=True)
checkingChain(chainAsText)

#conclusion of all together
import copy
nodeBlockchain = copy.copy(chain)
```

```
nodeBlockTxns = [makeTransaction() for i in range(4)]
newAddedBlock = makeBlock(nodeBlockTxns,nodeBlockchain)

print("Amount of blocks on the blockchain on Node 1 is currently:
"+ str(len(chain)))
print("Amount of blocks on the blockchain on Node 2 is currently:
"+ str(len(chain)))

try:
print("A new block is in the system and in the verification process!")
state = checkingBlockValidity(newAddedBlock,chain[-1],state)
# Updating the current state
chain.append(newAddedBlock)
except:
print("No valid block! Waiting for the next block...")

print("The Blockchain has been updated succesfully!
On Node 1 it is %s blocks long"%len(chain))
print("The Blockchain has been updated succesfully!
On Node 2 it is %s blocks long"%len(chain))
```

**• Python code for cryptography property inspired by the bitcoin package:**

The output will be a randomely created private key in hex and decimal form and a subsequent public key in hex form and compressed hex form.

```
import bitcoin

# make a random private key
private_key = False
while not private_key:
key = bitcoin.random_key()
decoded_key = bitcoin.decode_privkey(key, 'hex')
private_key = 0 < decoded_key < bitcoin.N

print("Private key in hex form is:", key)
print("Private key in decimal form is", decoded_key)

# mulitply the EC generator point G with the private key_
```

```
to get a public key point

public_key = bitcoin.fast_multiply(bitcoin.G, decoded_key)
print("Coordinates (X,Y) of public key is:" , public_key)

#encode as a hex and prefix 04
hex_encoded_public_key = bitcoin.encode_pubkey(public_key, 'hex')
print("Public key in hex form is:", hex_encoded_public_key)

#compress public key, adjust te prefix depending on whether y_
is even or odd

(public_key_x, public_key_y) = public_key
if (public_key_y % 2) ==0:
compressing_prefix ='02'
else:
compressing_prefix ='03'

hex_compressing_public_key = compressing_prefix +_
 bitcoin.encode(public_key_x, 16)
print("The compressed public key in hex form is:_
", hex_compressing_public_key)
```

**• The following R source code has bee used for the representation of figures 17 and 18:**

```
# subplots implemented (optional)
par(mfrow=c(1,4))
#representation of intial network setup
g <- make_full_graph(6) %du% make_full_graph(6) %du%
make_full_graph(6)_
%du% make_full_graph(6)
g <- add_edges(g, c(1,7, 2,8, 3,9, 10,13, 11,14, 12,15,_
 13,19, 14,20, 15,21 ))
com <- cluster_spinglass(g, spins=6)
V(g)$color <- com$membership
g <- set_graph_attr(g, "layout", layout_with_kk(g))
plot(g, vertex.label.dist=1.5, vertex.label.color="black")
#node leader gets transaction list
```

```
g <- make_full_graph(6) %du%_
make_full_graph(6) %du%
make_full_graph(6)%du% make_full_graph(6)
g <- add_edges(g, c(1,7, 2,8, 3,9, 10,13, 11,14, 12,15,_
13,19, 14,20, 15,21 ))
com <- cluster_spinglass(g, spins=6)
V(g)$color <- "black"
V(g)$color[2] <- "white"
g <- set_graph_attr(g, "layout", layout_with_kk(g))
plot(g, vertex.label.color="black", xlab ="1")

# 50% threshold
V(g)$color[1] <- "white"
V(g)$color[3] <- "white"
V(g)$color[4] <- "white"
V(g)$color[7] <- "white"
V(g)$color[10] <- "white"
V(g)$color[11] <- "white"
V(g)$color[13] <- "white"
V(g)$color[14] <- "white"
V(g)$color[15] <- "white"
V(g)$color[19] <- "white"
V(g)$color[20] <- "white"
V(g)$color[21] <- "white"
plot(g, vertex.label.color="black", xlab ="2")
# 60% threshold
V(g)$color[5] <- "white"
V(g)$color[8] <- "white"
V(g)$color[9] <- "white"
V(g)$color[16] <- "white"
V(g)$color[22] <- "white"
plot(g, vertex.label.color="black", xlab ="3")
# 80% threshold
V(g)$color[6] <- "white"
V(g)$color[12] <- "white"
V(g)$color[17] <- "white"
V(g)$color[23] <- "white"
plot(g, vertex.label.color="black", xlab ="4")
```

# References


Acquisti, A., Dingledine, R., Syverson, P. (2003), 'On the Economics of Anonymity', Volume 2742 of the series Lecture Notes in Computer Science pp. 84-102.

Alwen, J., Fuchshuber, G., Gazi, P., Park, S., Pietrzak, K. (2015), 'Spacecoin: A Cryptocurrency Based on Proofs of Space '.

Androulaki, E., Capkun, S., Karame, G. O., Roeschlin, M., Scherer, T. (2013),'Evaluating User Privacy in Bitcoin' Financial Cryptography and Data Security pp. 34-51.

Antonopoulos, M. A. (2014), 'Mastering Bitcoin. Programming The Open Blockchain', O'Reilly.

Antonopoulos, M. A. (2014), 'Bitcoin security model: trust by computation', http://radar.oreilly.com/2014/02/bitcoin-security-model-trust-by-computation.html, 20.02.2014, online accessed: 11.01.2018.

Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., Zenner, E. (2015), 'Ripple: Overview and Outlook'.

Back, A., Möller, U., Stiglic, A. (2001), 'Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems', Proceedings of Information Hiding Workshop (IH 2001), April 2001, pp. 245-257.

Baliga, B. (2017), 'Understanding Blockchain Consensus Models. Whitepaper', Persistent Systems Ltd.

Barber, S., Boyen, X., Shi, E., Uzun, E. (2012), 'Bitter to Better- How to Make Bitcoin a Better Currency', Financial Cryptography and Data Security pp. 399-414.

Baran, P., (1962), 'On Distributed Communication Networks', The RAND Corporation, Santa Monica, California.

Bech, M., Garatt, R., (2017), 'Central bank cryptocurrencies', BIS Quartely Review, September 2017.

Benaloh, J., de Mare, M. (1994), "One-way accumulators: a decentralized alternative to digital signatures," EUROCRYPT '93, vol. 765 of LNCS, 1994, pp. 274–285.

Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M. (2014),' Zerocash: Decentralized Anonymous Payments from Bitcoin', IEEE Computer Society, pp. 459-474.

Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M . (2015), 'Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture'.

Berentsen, A., Schär, F. (2017), 'Bitcoin, Blockchain und Kryptoassets', BoD - Books on Demand, Norderstedt.

Bernanke, B. (2013), 'Bitcoin may hold long term promise', qz.com/148399/ben-bernanke-bitcoin-may-hold-long-term-promise/, online accessed: 11.01.2018.

BitcoinBlog (2017), 'Dash: Der neue Überflieger-Altcoin, der Bitcoins Probleme löst?', https://bitcoinblog.de/2017/03/02/dash-der-neue-ueberflieger-altcoin -der -bitcoins-probleme-loest/,02.03.2017, online accessed: 11.01.2018.

Blockgeeks (2016), 'What is Cryptocurrency: Everything You Need To Know [Ultimate Guide]',https://blockgeeks.com/guides/what-is-cryptocurrency/, online accessed: 11.01.2018.

Blog.Ethereum (2014), 'The Issuance Model in Ethereum', https://blog.ethereum.org /2014/04/10/the-issuance-model-in-ethereum/,10.04.2014, online accessed: 11.01.2018.

Bonneau, J., Clark, J., Felten, E. W., Kroll, J.A., Miller, A., Narayanan A. (2014), 'Mixcoin: Anonymity for Bitcoin with Accountable Mixes',

Brennan, C., Lunn, W. (2016), 'Blockchain, Connection Series-The trust disrupter'.

Brown, A., Godsiff, P., Kewell, B., Maull, R., Mulligan, C. (2017), 'Distributed ledger technology: Applications and implications*', 2017 John Wiley & Sons, Ltd Strategic Change. 2017;26(5) pp. 481–489.

Brown, R. G., Carlyle, J., Grigg, I., Hearn, M., (2016), 'Corda: An introduction'.

Böhme, R., Möser, M. (2017), 'The price of anonymity: empirical evidence from a market for Bitcoin anonymization', Journal of Cybersecurity, 2017, pp. 1–9.

Cachin, C. (2001), 'Distributing trust on the internet', IBM Research Zurich Research Laboratory.

Cachin, C., Liu, S., Viotti, P., Vukolic, M., (2016), 'XFT: Practical Fault Tolerance beyond Crashes', 12th USENIX Symposium on Operating Systems Design and Implementation.

Cachin, C., Schubert, S., Vukolic, M. (2016),' Non-determinism in Byzantine Fault-Tolerant Replication', IBM Research – Zurich.

Camenisch, J., Lysyanskaya, A. (2001), "Dynamic accumulators and application to efficient revocation of anonymous credentials',CRYPTO '02, 2002, pp. 61–76.

Castro, M., Liskov, B. (1999),'Practical Byzantine Fault Tolerance', Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, February 1999.

Chaum, D. L. (1981), 'Untraceable electronic mail, return addresses, and digital pseudonyms', Communications of the ACM CACM Homepage archive, Volume 24 Issue 2, Feb. 1981, pp. 84-90.

Chaum, D. L. (1983), 'Security Blind Signatures for Untraceable Payments', Advances in Cryptology pp. 199-203.

Chaum, D. L. (1985), 'Security without Identification: Transaction Systems to Make Big Brother Obsolete', Communications of the ACM, Volume 28 Issue 10, pp. 1030-1044.

Coinchoose (2017), 'Litecoin', https://www.coinchoose.com/coins/litecoin/, online acceseed: 11.01.2018.

Coindesk (2017), 'A (Short) Guide to Blockchain Consensus Protocols', https://www.coindesk.com/short-guide-blockchain-consensus-protocols/,04.03.2017, online accessed: 11.01.2018.

Coinmarketcap (2017),'Cryptocurrency Market Capitalizations', https:// coinmarketcap.com/, online accessed: 11.01.2018.

Cryptographyengineering (2013), 'Zerocoin: making Bitcoin anonymous', https://blog.cryptographyengineering.com/2013/04/11/zerocoin-making-bitcoin-anonymous/, 11.04.2013, online accessed: 11.01.2018.

Danezis, G., Meiklejohn, S. (2015), 'Centrally Banked Cryptocurrencies', NDSS '16, 21-24 February 2016, San Diego, CA, USA

DashMasternode (2017), 'What is Dash?', http://dashmasternode.org/what-is-dash/, online accessed: 11.01.2018.

Decker, C., Wattenhofer, R. (2013),'Information Propagation in the Bitcoin Network', 13-th IEEE International Conference on Peer-to-Peer Computing.

Diaz, D., Duffield, E. (2014), 'Dash: A Privacy-Centric Crypto-Currency'.

Diedrich, H. (2016), 'ethereum', Wildfire Publishing.

Dolev, D., Lamport, L., Pease, M., Shostak, R. (1987), 'The Byzantine Generals', Concurrency Control and Reliability in Distributed Systems, Bharat K. Bhargava, editor, Van Nostrand Reinhold, pp. 349-369.

Druschel, P., Rowstron, A. (2001), 'Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems', R. Guerraoui (Ed.): Middleware 2001, LNCS 2218, pp. 329–350, 2001.

Ecomusing (2017), 'Build Your Own Blockchain: A Python Tutorial', http://ecomunsing.com/build-your-own-blockchain, 30.08.2017, online accessed:11.01.2018.

Emmadi, N., Narumanchi, H., (2017), 'Reinforcing Immutability of Permissioned Blockchains with Keyless Signatures' Infrastructure', ICDCN '17, January 04-07, 2017, Hyderabad, India.

European Central Bank (2015), 'Virtual currency schemes - a further analysis', www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemesen.pdf, online accessed: 11.01.2018.

Eyal, T., Gencer, A. E., Sirer, E. G., van Renesse R. (2016), 'Bitcoin-NG: A Scalable Blockchain Protocol', 13th USENIX Symposium on Networked Systems Design and Implementation

Eyal, T., Sirer, E. G. (2013), 'Majority is not enough: Bitcoin mining is vulnerable', Department of Computer Science, Cornell University.

Fischer, M. J., Lynch N. A., Paterson, M. S. (1985), 'Impossibility of Distributed Consensus with One Faulty Process', Journal of the Association for Computing Machinery, Vol. 32, No.2, April 1985, pp. 372-384.

Gernandt, A., Gipp, B., Meuschke, N., (2015), 'Decentralized Trusted Timestamping using the Crypto Currency Bitcoin, iConference 2015.

Github (2017), 'snakecoin-server-full code.py, https://gist.github.com/aunyks/47d157f8bc7d1829a729c2a6a919c173, 23.07.2017, online accessed: 11.01.2018.

Goldreich, O., Micali, S., Wigderson A. (1991), 'Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proofs Systems', Journal of the Association for Computer Machinery, Vol. 38, No. 1, July 1991, pp.691-729.

Goldwasser, S., Micali, S., Rackoff, C. (1985), 'The Knowledge Complexity of Interactive Proof-Systems', Journal of the Association for Computer Machinery, pp.291-304.

Gomber, P., Hinz, O., Nofer, N., Schiereck, D. (2017), 'Blockchain', Springer Fachmedien Wiesbaden.

Green, M., Garman, C., Miers, I., Rubin, A. D. (2013),' Zerocoin: Anonymous Distributed E-Cash from Bitcoin'.

Greenspan, G. (2015), 'MultiChain Private Blockchain-White Paper'.

Hyperledger (2015), 'Hyperledger Whitepaper', https://docs.google.com/document/d/1Z4M_qwILLRehPbVRUsJ3OF8Iir-gqS-ZYe7W-LE9gnE/edit#heading=h.m6iml6hqrnm2, online accessed: 11.01.2018.

Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A. (2006),'Cryptography from Anonymity'.

Johnson, B., Grossklags, J., Laszka, A., Moore, T., Vasek, M. (2014), 'Game-theoretic analysis of DDoS attacks against Bitcoin mining pools', Workshop on Bitcoin Research

Kosba, A., Miller, A., Papamanthou, C., Shi, E., Wen, Z. (2016),' Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts', University of Maryland and Cornell University.

Lamport, L., Pease, M., Shostak, R. (1981), 'The Byzantine Generals Problem', ACM Transactions on Programming Languages and Systems, Vol. 4, No.3, July 1982, Pages 382-401.

Lamport, L. (2001), 'Paxos Made Simple'.

Lippencott, J., Versluis, R., Viglione, R. (2017), 'Zen White Paper'.

Liskov, B., Oki, B. M. (1988), 'Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems', Massachusetts Institute of Technology.

Mazieres, D. (2015),' The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus', Stellar Development Foundation.

Medium (2017), 'Let's Build the Tiniest Blockchain-In Less Than 50 Lines of Python', https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b, 16.07.2017, online accessed: 11.01.2018.

Nakamoto, S. (2008), 'Bitcoin: A peer-to-peer electronic cash system', bitcoin.org, 2008.

NavCoin (2016),'The Unbreakable Code NAVTECH WHITEPAPER 2016 Beta Release v0.9', https://www.navcoin.org/files/navtech-whitepaper-beta-v0.9.pdf, online accessed: 11.01.2018.

Noether, S. (2017),'Ring Confidential Transactions' Monero Research Labs.

Okamoto, T., Ohta, K. (1991), 'Universal Electronic Cash', Advances in Cryptology

CRYPTO '91, LNCS 576, pp. 324-337.

Ongaro, D., Ousterhout, J. (2014), 'In Search of an Understandable Consensus Algorithm (Extended Version), Stanford University'.

Schnorr, C. P. (1991), "Efficient signature generation for smart cards," Journal of Cryptology, vol. 4, no. 3, pp. 239–252, 1991.

Schwartz, D., Britto, A., Youngs, N. (2014), 'The Ripple Protocol Consensus Algorithm',Ripple Labs Inc, 2014.

Schwartz, E., Thomas, S. (2015), 'A Protocol for Interledger Payments'.

Steemit (2017), 'An overview of blockchain privacy mechanisms and how Zerocoin in Zcoin $XZC (not Zcash) stacks up', https://steemit.com/zcoin/@zcoinofficial/an-overview-of-blockchain-privacy-mechanisms-and-how-zerocoin-in-zcoin-usdxzc-not-zcash-stacks-up, online accessed: 11.01.2018.

Swan, M. (2015), 'Blockchain-A blueprint for a New Economy', O'Reilly Media Inc.

Swanson, T. (2015), 'Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems'.

Van Saberhagen, N. (2012), 'CryptoNote v 1.0'.

Verge (2016), 'Verge Blackpaper', https://vergecurrency.com/assets/Verge-Anonymity-Centric-CryptoCurrency.pdf, online accessed: 11.01.2018.

Waves Platform (2017), 'Review of blockchain consensus mechanisms', https://blog.wavesplatform.com/review-of-blockchain-consensus-mechanisms-f575afae38f2, 31.07.2017, online accessed: 11.01.2018.

Wood, G. (2017), 'Ethereum: A Secure Decentralised Generalised Transaction Ledger' EIP-150 Revision (1e18248 - 2017-04-12).

ZCoin (2017), 'Understanding how Zerocoin in Zcoin works and how it compares to other anonymity solutions Part 1', http://zcoin.io/understanding-how-zerocoin-in-zcoin-works-and-how-it-compares-to-other-anonymity-solutions-part-1/, 20.03.2017, online accessed: 11.01.2018.

Zerocash-project (2017),'How Zerocash works', http://zerocash-project.org/how_zerocash_works, online accessed: 11.01.2018.