Seminar Report

on

Blockchain Technology and Applications

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award

BACHELOR’S DEGREE IN COMPUTER SCIENCE & SYSTEM ENGINEERING

By

DEEPANSHU JAYSWAL (1728263)

KIIT

KALINGA INSTITUTE
OF INDUSTRIAL TECHNOLOGY
Deemed to be University U/S 3 of the UGC Act, 1956

Contents

Abstract

This report focuses on explaining the blockchain technology and its application fields. We distinguish between multiple types of blockchains and explain the two biggest platforms, namely Bitcoin and Ethereum. While introducing those two platforms we explain the most important technology and algorithms used such as proof of work concept. Some of the security issues and solutions are also covered. We conclude with some concrete Ethereum based applications that demonstrate the usage of blockchain technology beyond cryptocurrency and illustrate current developments in this field.

1. Introduction

Throughout history there were few points in time where the emergence of new technology changed the way people lived. Personal computers, internet, cellphones and smartphones just to name a few. According to some researchers the world is on the verge of another big change and this change is being brought by cryptocurrency and blockchain technology in a broader sense. Since the dawn of capitalism, banks have played the most important role in shaping the world's economy. They represent central authority and regulate the flow of money and also sometimes the value of the same.

In this report we present new technology, namely blockchain, which has the potential to seriously alter the economy we know today. The key principle behind blockchain is its distributed nature and lack of central authority. In the world where trusted parties represent an overall weakens of any system, but it's being justified as necessity by governments and other ”trusted authorities”, blockchain technology brings new concept which embraces new technological developments and shifts the power from one to many, or from the central concept to distribute one.

The concept where it is possible to perform transactions without having to trust any central party has been unimaginable for a long time until Bitcoin emerged with its underlined blockchain technology. Bitcoin showed for the first time that it is possible for two parties, who do not necessarily trust each other, to perform transactions without any central authority as intermediary. Both sender and receiver trust only the underlying architecture which guarantees the security of the system.

The application of blockchain technology has been until recently primarily in the cryptocurrency sector, but new potential areas emerged such as distributed storage, smart contracts etc. which extend the functionality of the blockchain and open doors for the new breakthroughs.

This report is structured as follows: We first explain the blockchain technology in general and its first application field, namely cryptocurrency.

2. Blockchain Overview

In this chapter we explore the basics of the blockchain technology and mainly its role in cryptocurrencies. Even though the idea of a blockchain as an underlying architecture is relatively new, there are recent developments which suggest applications of the blockchain in other domains apart from cryptocurrencies. According to the literature [2] there are three main categories of blockchain applications:

- Blockchain 1.0: Currency
- Blockchain 2.0: Smart contracts
- Blockchain 3.0: Areas in government, health, science etc.

Because the last application area is still relatively new and only ideas rather then actual solutions exist, in this report we focus on the first two areas where the blockchain is used.

2.1 Blockchain 1.0

Because cryptocurrencies are the first real application of blockchain technology, it is often referred as Blockchain 1.0 [2]. Since the blockchain had its first big breakthrough with Bitcoin protocol and cryptocurrency, it is often difficult to distinguish those three main components shown in Figure 1: blockchain, protocol and currency. In this chapter we look separately into each of these three parts of Blockchain 1.0 and then in the next chapter by using Bitcoin as an example we demonstrate the practical application of these technologies in the real world.

2.1.1 Blockchain

The blockchain in its basic form can be seen as the distributed, decentralized, transparent and chronological database of transactions, sometimes also called the ledger. The data in the blockchain (e.g transactions) is divided into blocks. Each block is dependent on the previous one. The system in which a blockchain serves as the database of nodes or workers. These workers are responsible for appending new blocks to the blockchain. A new block can only be appended after all nodes in the system reach a consensus, i.e all agree that this block is legit and contains only valid transactions. How the validity of transactions is determined and how the nodes compute new blocks, is regulated by the protocol. We will explain Bitcoin protocol in later sections. Blockchain is shared among all nodes in the system, it is monitored by every node and at the same time controlled by none. The protocol itself is responsible to keep the blockchain valid. An illustration of the blockchain can be seen in Figure 2.

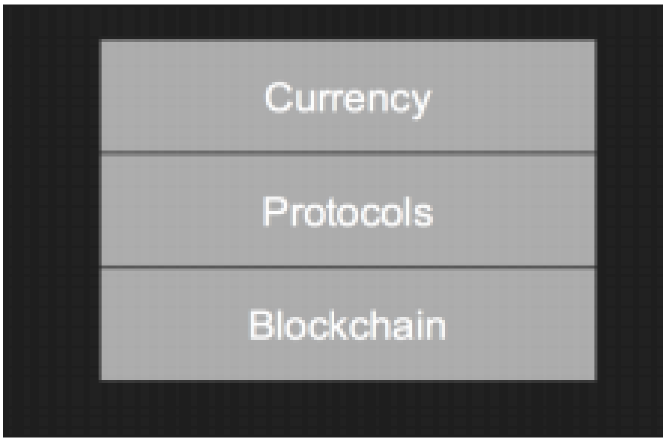


Figure 1: Success rates

2.1.2 Protocol

The protocol, as briefly mentioned previously, regulates how the blockchain is used for a specific purpose. In the cryptocurrency sense, it represents the software which transfers the money between two parties in the system. Protocol is also the one thing to whom the users trust in the system. Since the protocol is completely transparent to all users, everybody can analyse it and check if it actually performs the intended task.

2.1.3 Currency

Represents the currency itself, e.g Bitcoin, Litecoin, Peercoin etc. In order to use these currencies (which are just hashes in the system), the protocol has to be followed and blockchain architecture is used as an underlying database where every transaction between two parties is stored. So, every change of ownership is marked in the blockchain and it can be traced back until its creation (Coin creation, i.e introduction of new coins is also determined by the protocol). This implies: To check whether a person possesses a specific coin, one has to check the blockchain and see if a specific person got a specific coin from somebody or not. Also the blockchain technology makes it impossible to double-spend the coin, because every change is written in the block. Much like regular currency Bitcoins are decomposable into smaller units. The smallest unit in Bitcoin is 1 Satoshi which is 10^{-8} Bitcoins (or abbreviated BTC).

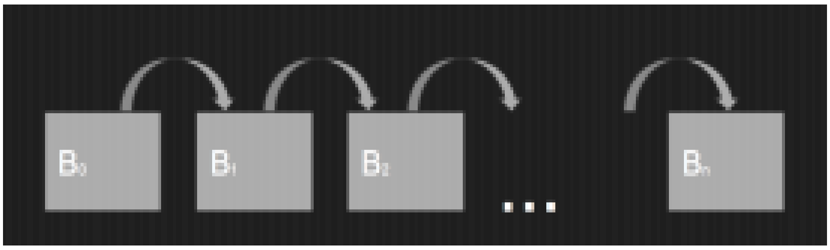


Figure 2: Blockchain

2.2 Beyond Cryptocurrency: Blockchain 2.0

Almost since the introduction of Bitcoin and its underlying blockchain ledger, researchers began to explore other fields where blockchain technology might be of great use. In Chapter 4 we explore Blockchain 2.0 in detail but here, as an introduction, we introduce additional types of blockchains and reason about their potential in other fields beyond cryptocurrency. Some of those potential applications are:

- General (bonded contracts, multiple signature transactions)
- Financial transactions (pensions, stocks ...)
- Public records (land titles, vehicle registrations ...)
- Identification (drivers licence, ids ...)
- Private records (loans, contracts ...)
- Physical asset keys (home, hotel rooms, rental cars)
- Intangible assets (patents, trademarks, ...)
- In literature [3] two main categories of blockchain are distinguished:
- Permissionless Blockchain
- Permissioned Blockchain

2.2.1 Permissionless Blockchain

Permissionless blockchains are the ones where anybody can join the network to be a verifier without obtaining any prior permission to perform such network tasks. Since anybody can join, special types of incentive mechanisms are necessary in order for verifiers to participate. It has the advantage that it can accommodate both anonymous and pseudo anonymous actors. Bitcoin and Ehtereum are examples of permissionless blockchains.

2.2.2 Permissioned Blockchain

The other type is the so-called permissioned blockchain where special permission is needed from an authority to become a verifier in the system. Permission blockchains are intended to be purpose-built, and can thus be created to maintain compatibility with existing applications (financial or otherwise). An advantage of a permissioned blockchain is scalability. In a permissionless blockchain, the data is stored on every computer in the network, and all nodes verify all transactions. In a permissioned blockchain, only a smaller number of preselected participants will need to operate . However, because of the smaller number of participants, it is much easier for a group of users to collaborate and alter the rules, or revert transactions and that is why only trusted parties should be given permission to act as verifiers. Examples of permissioned blockchains include Eris, Hyperledger, and Ripple.

3 Bitcoin

In this chapter we explore the concrete usage of the blockchain with the most prominent example, namely Bitcoin. Bitcoin itself presented a revolution in 2009, when it was introduced by Satoshi Nakamoto [8] because it introduced peer-to-peer transactions without the need of an intermediary. The users involved in transactions do not need to trust each other but they trust the system (more precisely the protocol) which comprises decentralised nodes which verify and validate the transactions.

Bitcoin represents the first decentralized cryptocurrency in the world and it is the largest of its kind in terms of market value. In this chapter we explore the basic principle of Bitcoin, how the transactions look like, how they are validated and included in the block. At the end we also present a brief overview of security and demonstrate why the problems such as double-spending are highly improbable.

3.1 Overview and key technology

The main parties in the Bitcoin protocol are:

- Sender: the one who initiates the transfer of currency.
- Receiver: the recipient of the transfer
- Miners: Independent nodes which verify and confirm the transactions, also sometimes called as workers. The security of the system is guaranteed by these nodes.
- Blockchain: Decentralised ledger shared among all miners where all transactions are stored (the complete history since the Bitcoin creation)

The system has following characteristics:

- Decentralized: The work is done by miners which are located around the world. Moreover any individual who possesses a device capable of performing computation can participate.
- Pseudo-Anonymous: Users are identified via public keys in the system and there is no way to know to whom the public key belongs unless it is made public by the owner of the key.

As mentioned above the public-key cryptography is used to identify and verify the users, that is for signing the transactions. Also cryptographic hash functions play a crucial role. Their unique properties of one-wayness (there is no way of knowing what is the string that produced a given hash) and collision resistance (it is highly improbable to find two messages m1 and m2 which have the same hash or for a given m1 to find m2 with the same hash value) are used in the so-called proof of work concept.

3.2 Bitcoin transaction

A graph of a Bitcoin transaction can be seen in Figure 3. To create a transaction we need two parties who are involved. Let us name them Alice and Bob and let us say Alice wants to send 50 BTC (Bitcoins) to Bob. Both parties, Alice and Bob, use public-key cryptography and have generated public-private key pairs. In this context public key is known as ”validation key” and represented with abbreviation VK and private key is known as ”signing key” and written as SK. Public keys are known to all parties in the system, while private keys must be kept secret.

Alice, the initiator of the transaction has to specify a few key points which make up a transaction. Bob on the other hand does nothing and waits on the other side to receive the Bitcoins. To send 50 BTC to Bob, Alice, of course, has to possess those coins. While the Bitcoins are nothing but hashes of transactions in which they are spent, Alice has to specify where she received the Bitcoins she wants to give to Bob. So she specifies 3 transactions in which she received Bitcoins. Let us say she received 25 BTC from Carol, 20 BTC from David and 20 BTC from Ted. She computes the hash of these transactions and creates the digests Dc, Dd and Dt. Important thing here is that Alice can not ”break” the transaction to take, let us say, only 5 coins from the last transaction which are sufficient for this transaction (25+20+5 would equal 50). Later during verification, nodes in the system check these digests to verify Alice has indeed received those Bitcoins. In the second part of the transaction (on the right side in Figure 3), Alice specifies how many BTC she wants to send and to whom. In this case she writes 50 BTC and she also includes Bob’s public key to identify the intended recipient. Because of the fact that she can not break the transaction in which she received the BTC, she specified how many BTC she will receive as a return (or change). Here she writes her own public key as a recipient and 14 BTC as the amount she would like to get in return. The last part of the transaction is Alice’s signature. So she signed the transaction with her private key and includes that signature in the transaction, so the nodes in the system can verify Alice is indeed the initiator of the transaction. After all parts are written, this transaction is sent (broadcasted) to all nodes in the system, to be included in the next block. The transaction only becomes valid after it is included in the blockchain. Note here that Alice specified 50 BTC for Bob and 14 BTC back, but she included 25+20+20 = 65 BTC in incoming transactions. The one remaining BTC will be given to the node in the system, which includes this transaction in a block i.e which does the work necessary for this transaction to be valid and be included in the blockchain. This is called a transaction fee.

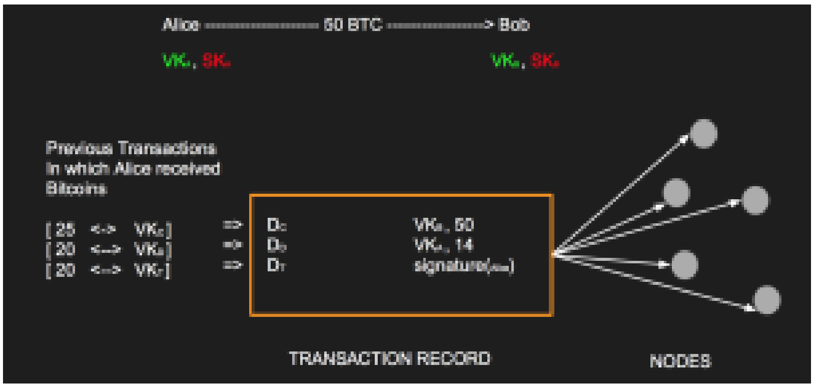


Figure 3: Bitcoin Transaction

3.3 Block Creation

In this section we examine how transactions are incorporated in a block and how all nodes agree on the next block which should be included in the blockchain. After transactions are initiated (like transactions from Alice, explained in the previous section), they are broadcasted to all nodes in the system. These nodes now need to incorporate the transactions into a block and append this block to the blockchain. For simplicity let us assume every node in the system does following work (in reality multiple computation devices work together to create one block but w.l.o.g we can assume every node in the system does the same work).

Each node takes a subset of transactions which it has received (or all of them) and computes hashes of these transactions. The transactions which the node took, are candidates to be in the next block provided that the node succeeds in publishing the block first. More about this consensus later. After the node computes the hash of each transaction individually, it hashes them again pairwise thus creating a Merkle tree. Merkel trees are the trees in which the children are hashes of their parents. This can be seen on the left side of Figure 4. After this (one) hash of all transactions has been computed, the node combines it with the hash of the previous block in the blockchain (the last published block). The combination of those two hashes creates the so-called challenge. Challenge is a string to which a proof needs to be found.

The work done so far (computation of Merkle tree and combining this hash with the hash of the previous block) is not computationally intensive. It can be done really quickly by every node in the system. That means every node now has a block which they want to append to the blockchain. Without some sort of consensus mechanism it would be impossible to determine which block should be included next. Only one node succeeds to include the block, the rest have to start over with new transactions. To determine which block (a block computed by which node) is going to be included in the blockchain next, proof of-work puzzle has to be solved, and this proof is computationally intensive and whichever node solves it first, the block created by that node is going to be accepted as the next block in the blockchain. In the next section we look into proof-of-work concepts a little bit deeper.

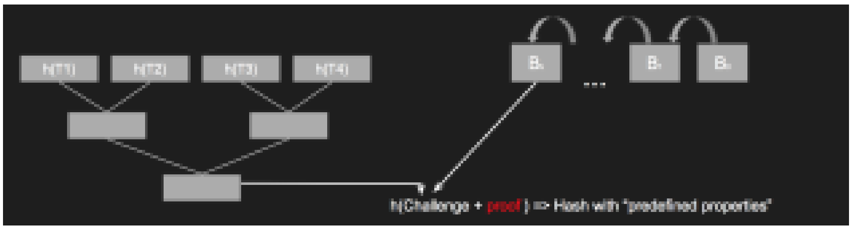


Figure 4: Block Generation

3.4 Proof-of-Work

Proof-of-Work is essentially a puzzle which consists of a challenge to which a proof has to be found. This concept has been known before Bitcoin and virtual cryptocurrencies and it has been used in spam and DDoS prevention. The characteristics of a proof-of-work puzzle are:

- Computationally difficult to compute
- Easy to verify

This concept for example can be used to prevent spam by forcing the e-mail client to solve some puzzle which can only be solved by brute force and takes up a few CPU cycles. For a legitimate user who writes only a small number of emails, the computation time necessary to solve this puzzle is negligible, but for a spammer who tries to send millions of emails in a short period of time this constraint makes it impossible to send so many emails that fast. This concept of finding a solution by brute force is essentially the core property of proof-of-work puzzles. The solution to a puzzle is only possible to find by using brute force (no other deterministic way should exist). To apply this concept in protocols such as Bitcoin, we need to find a puzzle which is a lot more difficult than the one for an email client, because here we are dealing with much more computation power and much more nodes in the system. The Bitcoin uses the following proof-of-work concept (shown in Figure 5):

After the nodes compute the challenge (Merkle tree + hash of the previous block) they need to find a proof (string) which when concatenated with a challenge and hashed using SHA-256 gives an output which has a specific amount of leading 0s. E.g when we combine challenge and proof and hash it, the output should have for example 40 leading 0s. This is very challenging problem be cause SHA-256, like other hash functions, has the property that it is impossible

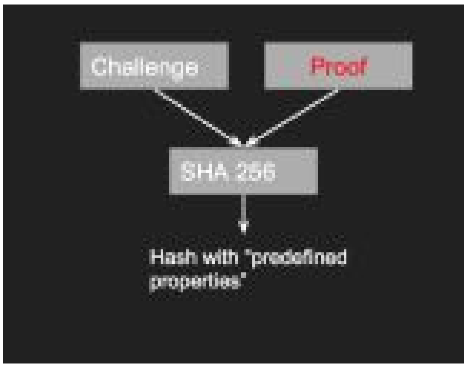


Figure 5: Bitcoin Proof-Of-Work Concept

For example:

- SHA-256 of cat = 77af778b51abd4a3c51c5ddd97204a9c3ae614ebccb - 75a606c3b6865aed6744e
- SHA-256 of Cat = 48735c4fae42d1501164976afec76730b9e5fe467f68 - 0bdd8daff4bb77674045

So the only way to find the output with 40 leading 0s is to try as many inputs as possible until one hash has desired properties. Also this problem is easy to make more difficult as the Bitcoin network grows, because the protocol only has to state that after some time the output has to have e.g 41 leading 0s. This makes the problem exponentially more difficult.

3.5 Consensus

After a node finds the proof, it broadcasts the proof together with the block. All other nodes now only have to compute one hash to verify that the proof provided is indeed correct. After verification the block is included in the blockchain and every other node can now abandon the work they have been doing (they have been trying to incorporate the same transactions in a block which is now published) and start incorporating new transactions. As previously explained, finding the proof for a challenge is computationally intensive, and it is highly unlikely for the two nodes to find it at the same time, so with high probability only one proof will be published at a give time. We explain in subsequent sec tions what happens if we have more than one blockchain (divergent versions).

3.6 Incentive For Miners

As seen from the previous three sections, to incorporate a block in a blockchain is not an easy task. Finding a proof is challenging and computationally intensive. The question arises: Why would someone provide computing power and do all the work necessary? In this section we explain the payout to the miners. In order to verify the transaction and create a block miners use computation power which has its cost. Only the miner which comes up with the proof for the challenge gets the award for the current block. The rest gets nothing and have to start working on a new block and so on. So for every block in the blockchain only one miner gets awarded.

There are two kinds of awards miners get when they successfully publish a new block:

- Transaction fees from all transactions
- Coinbase transaction

When we explained the creation of a Bitcoin transaction in Section 3.2 we had an example where Alice specified 3 incoming transactions with a total worth of 65 BTC. She sent 50 BTC to Bob and assigned 14 BTC back to herself as change. But there is 1 BTC left, and this 1 BTC actually goes to the miner who incorporates the transaction in a block. This is similar for every transaction in the Bitcoin system. We also said in Section 3.3 that miners can incorporate any number of transactions in a block from the pool of transactions which have not been verified yet. Here we see the incentive for the miners to actually take all available transactions instead of just a subset of them. The more transactions are in the block, the more transaction fees are collected if the block is published successfully. The additional overhead of incorporating a large number of transactions instead of the small one is negligible because that overhead only happens in the first part where the Merkle tree is computed and that is not computationally intensive. Proof-of-Work complexity remains the same.

The second source of the payout to the miners is through the so-called coin base transactions. Up until now we have seen that the ownership of Bitcoins changes through transactions much like the ownership of traditional currency today. Still we need to look into how new Bitcoins get introduced in the system. An answer to that question is the coinbase transactions. In every block the miners are allowed to include an additional transaction which does not come from anyone (there are no BTC that enter the transaction like in the normal transactions) and sends the BTC to themselves. The protocol regulates how many BTC miner is allowed to assign to himself in a coinbase transaction and that is how additional BTC are introduced in the system. The amount of BTC the miners assign to themselves is decreasing over time. After every 210000 blocks the amount of coinbase transactions decreases by 50%. That means there are a limited number of BTC that can be produced and that is around 21 million. After all BTC are mined, there will be no coinbase transaction anymore but the system is expected to grow until then, and the amount in transaction fees alone should compensate for lack of this transaction.

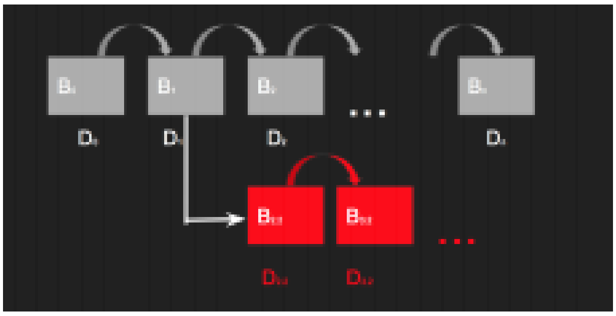


Figure 6: Fork in the Chain

3.7 Blockchain Security

In this section we explore how Blockchain makes it highly improbable for some one to cheat the system or to do fraudulent actions. One of the biggest potential problems is the so-called double spending problem. This means that a user can try to spend coins twice or more times. That is: initiate multiple transactions with the same Bitcoins as an input. Of course this should not be allowed and only one transaction should go through and others should fail. From section 3.2 we know that when a transaction is created it gets broadcasted to all nodes in the system which, before incorporating the transaction in the blockchain, actually check whether the user who initiated the transaction is the owner of the Bitcoins and whether he tried to double spend them. If a user would attempt to do this, the second transaction would fail and everybody in the system would know that the user, Alice, is not honest and would not trust her anymore.

But there is one other problem and that is the possibility of multiple blockchains to exist, or multiple versions of the history. We know that everybody should work on one chain, but it is possible that some users did not receive notification that a new block is created or they want to create their own version of the chain with bad transactions in it etc. and then multiple versions could exist at some point. This is called a fork in the chain and it is illustrated in Figure 6.

Before we look into a scenario where a user might want to double spend Bitcoins let us look at the notion of chain length. According to the protocol all miners should work on a chain that is the longest. This does not mean the chain that has the highest number of blocks but the chain that has the most work put into. So for every block there is a number which signifies how hard it was to create the particular block, let us call this number Dn for a block n. The sum of all these Dn's gives us the total chain length. Now this number is proportional to the amount of computation power users put into creating the block.

In the example from previous sections Alice had sent 50 BTC to Bob. Let us assume Alice is dishonest and wants to double spend those 50 BTC. She needs to create a second transaction that gives those 50 BTC to someone else or to another account which belongs to her. She can not publish this transaction like she did with the original transaction because it would be rejected by the miners. Now she, on her own, has to create a second block containing her second transaction but not the first transaction in which she sent 50 BTC to Bob and convince everyone to start building on top of that block. In order to do that she needs to solve proof-of-work puzzles which are very difficult, and even if she succeeds other nodes may have created several other blocks on the original chain in the meantime. She now has to create a longer chain by herself. In Figure 6 this means Dns of the red chain which Alice is trying to create needs to be larger than Dns of the white (legitimate) chain.

In order for Alice to succeed in her intention she would have to possess more computation power then everybody else combined in the system. That power would cost millions of dollars and it is highly unlikely that a single entity is in possession of such power. So the system is safe as long as there are no entities which possess more computation power then all other ”honest” nodes in the system.

And even if someone would theoretically possess all that power it is economically much better to use it for legitimate Bitcoin mining/verification because with every new block this user would receive all transaction fees + BTC from coinbase transactions which would outweigh the possible income received by fraudulent transactions.

4. Blockchain 2.0: Applications [1, 2, 3]

After the rise of the Bitcoin protocol people started to think about application domains other than cryptocurrency. The nature of the blockchain network has the potential to enable the development of a wide range of different applications that are decentralized. Decentralized applications are becoming more and more important in recent years. In this context not only the architectural decentralization is important but the political one as well. Blockchain based applications can not be stopped, censored or controlled and they ensure transparency and trust between all parties involved in the interaction. The discussion about whether these are arguments for or against the use of decentralized application often depends on the context of the usage too. Because every information on the blockchain is public, one will not find many people thrilled to use a medical blockchain application and feed it with their personal medical history for example.

The focus of a new generation of the blockchain applications is not on the transfer of money via transactions on the blockchain but on carrying out serious computation on a decentralized network of computers. Despite the fact that the use of the blockchain as a ledger for decentralized applications offers a seemingly unlimited amount of potential, many concerns regarding the use of the blockchain exist. One of the largest problems with blockchains is the issue of scalability. As for now the use of the blockchain for applications requires every full network node to perform every calculation to reach consensus. Before blockchain technology is able to become mainstream this is a problem that certainly needs to be solved. This is a problem among many others. In section 5.5 we will try to shed light on a few of them.

Blockchain shows potential to be used in many different fields and some of them are:

- Domain registration (Namecoin)
- Trading Assets (Colored Coin)
- Cloud Storage
- Voting
- Crowdfunding
- Car sharing
- Gambling and prediction markets
- Internet of Things

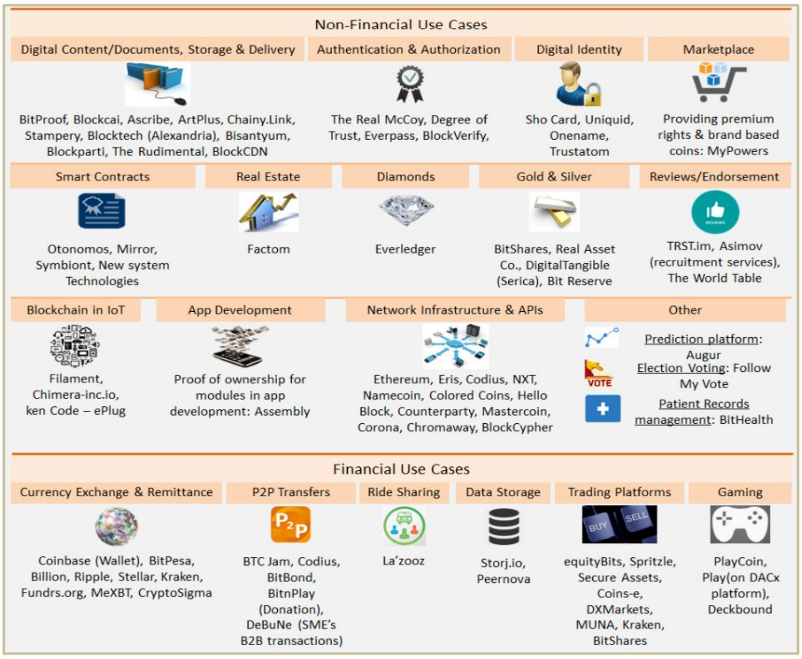

Non-Financial Use Cases
Digital Content/Documents, Storage & Delivery
BitProof, Blockcai, Ascribe, ArtPlus, Chainy.Link, Stampery, Blocktech (Alexandria), Bisantyum, Blockparti, The Rudimental, BlockCDN
Authentication & Authorization
The Real McCoy, Degree of Trust, Everpass, BlockVerify,
Digital Identity
Sho Card, Uniquid, Onename, Trustatom
Marketplace
Providing premium rights & brand based coins: MyPowers
Smart Contracts
Otonomos, Mirror, Symbiont, New system Technologies
Real Estate
Factom
Diamonds
Everledger
Gold & Silver
BitShares, Real Asset Co., DigitalTangible (Serica), Bit Reserve
Reviews/Endorsement
TRST.im, Asimov (recruitment services), The World Table
Blockchain in IoT
Filament, Chimera-inc.io, ken Code – ePlug
App Development
Proof of ownership for modules in app development: Assembly
Network Infrastructure & APIs
Ethereum, Eris, Codius, NXT, Namecoin, Colored Coins, Hello Block, Counterparty, Mastercoin, Corona, Chromaway, BlockCypher
Other
VOTE
Prediction platform: Augur
Election Voting: Follow My Vote
Patient Records management: BitHealth
Financial Use Cases
Currency Exchange & Remittance
Coinbase (Wallet), BitPesa, Billion, Ripple, Stellar, Kraken, Fundrs.org, MeXBT, CryptoSigma
P2P Transfers
P2P
BTC Jam, Codius, BitBond, BitnPlay (Donation), DeBuNe (SME's B2B transactions)
Ride Sharing
La'zooz
Data Storage
Storj.io, Peernova
Trading Platforms
BUY
SELL
equityBits, Spritzle, Secure Assets, Coins-e, DXMarkets, MUNA, Kraken, BitShares
Gaming
PlayCoin, Play(on DACx platform), Deckbound

Figure 7: A few block chain applications with their respective domains

Maybe the most prominent blockchain application that has a purpose outside of sending money from one party to another is Namecoin. Namecoin is the first fork of the Bitcoin protocol ever published and it aims to work as a decentralized domain name registration service and database. Without such a system (centralized or not) services like Tor use pseudo random hashes to identify accounts. Technically there is no problem with this approach but users would prefer to use more meaningful names to identify the accounts they interact with. Of course systems like Tor would not work if the names identifying the users were not unique. Namecoin ensures the uniqueness of the names chosen by the users. The consensus protocol used by Bitcoin is perfectly suited to ensure that the first user that gives himself a certain name gets this name and that all users trying to register a name that is already taken will fail to do so.

A closer look at all those application domains is outside the scope of this report. Furthermore a lot of these concrete protocols lack documentation. Therefore we decided to focus on a single relatively new protocol called Ethereum (5). This decision is based not only on the fact that Ethereum is well documented but also because Ethereum, unlike other blockchain protocols, can be used as a tool to easily implement a wide range of different applications that make use of the blockchain.

5. Ethereum

5.1 What is Ethereum?

Ethereum is a project which attempts to build the generalised technology; technology on which all transaction-based state machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto unexplored compute paradigm in the mainstream: a trust ful object messaging compute framework

Our research on Ethereum suggests that there are as many opinions on what Ethereum actually is, as there are people trying to answer that question:

- public blockchain-based distributed computing platform, featuring smart contract functionality
- multipurpose protocol built for decentralised applications on the blockchain • operating system enabling the development of decentralised application • first (slow and expensive to use) decentralised computer
- Bitcoin 2.0 and many more

Unlike the question of how Ethereum works, a definition of Ethereum seems to be really subjective and highly dependent on the role people have in the Ethereum community (Scientists, Investors, Developers etc). For a developer the most practical view of Ethereum is likely that it is a platform that enables developers to develop decentralized applications that run on the blockchain. This section aims to give a short introduction into Ethereum while highlighting the differences to the Bitcoin protocol described in chapter 3.

5.2 Accounts and Transactions [1, 4]

Ethereum is based on the concept of so called smart contracts. A contract is usually a piece of code that is stored on the blockchain. It is executed by a user via sending a transaction to the contract, the code is controlling. The terminology contract suggests that an Ethereum contract is the same as a legal contract which is clearly not the case. Contracts in Ethereum are not limited to financial workflows and are much more generic. Because of that, contracts are also often referred to as agents or objects. Conceptually there exist two different types of accounts in Ethereum:

- Externally owned accounts
- Contract accounts

An externally owned account is controlled by a private key and owned by a real human being. This is the equivalent to a Bitcoin account. Contract accounts however are controlled entirely by code. Accounts consist of four main fields:

- Nonce
- Ether balance
- Contract code
- Account storage

These four fields represent the accounts status. The status of all accounts combined will be called the status of Ethereum. The so-called nonce is a scalar value that is equal to the number of transactions sent from the account's address. It is used to make sure that each transaction can only be processed once. Ether balance reflects the amount of Ether an account has accumulated. Ether is the cryptocurrency of Ethereum and is also indirectly used to pay transaction fees. The contract code is empty if the account is controlled externally otherwise the contracts controlling code is stored in here. Last but not least every account gets its own long term (not reseted after computation) storage, where mainly contracts save their data. The account storage field contains a hash referencing the location of the accounts status in the blockchain. As we will see later the states of the accounts are stored on the blockchain and are altered by transactions submitted to the blockchain.

If we view Ethereum as a state machine in which all possible combinations of variable assignments to the account fields over all accounts represent the states the Ethereum system can operate in, transactions would be the transitions between those states. Another way of seeing transactions is to say that transactions enable the interaction between accounts. Independent of the approach transactions are made up of the following fields:

- Recipient of the message
- Signature identifying the sender
- Amount of Ether to transfer from the sender to the recipient
- Optional data field
- GasLimit
- GasPrice

As in Bitcoin every transaction: has a recipient, has to be signed by the sender and contains a field where the user can specify the amount of money he wants to transfer. The fields GasLimit and GasPrice are unique to Ethereum and will be discussed in Section 5.4. If an externally controlled account submits a transaction with a contract account as the recipient the transaction triggers the execution of the code of the targeted contract account that usually alters the state of the accounts. The figure below illustrates such a transaction. In the illustrated example the external account wants the contract to store the word CHARLIE at position 2 in a tupel the contract holds in its permanent storage. Furthermore the user transfers 10 Ether to the contract which is reflected in the new Ether balances. Note that the fields GasLimit and GasPrice were left out intentionally in this example.

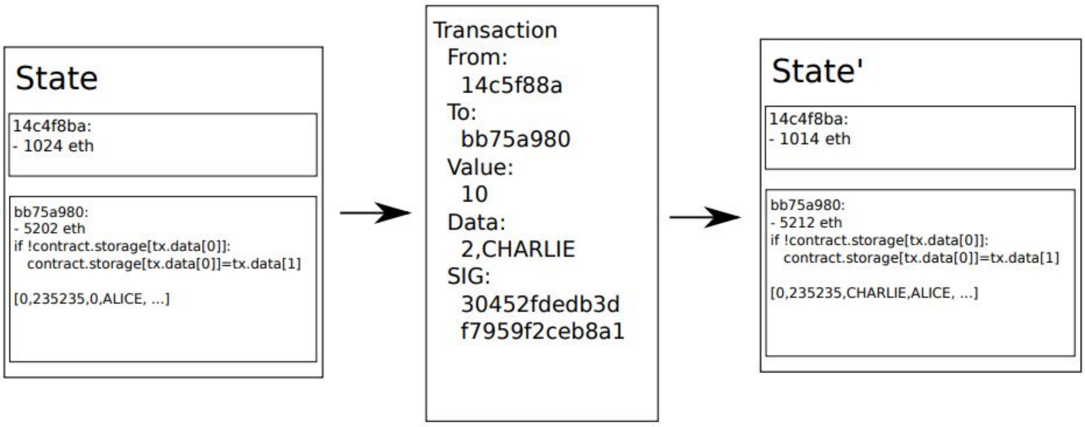


Figure 8: Execution of a contract (bb75a980) triggered by a transaction submitted by an external account (14c4f8ba)

If a transaction is sent from an externally controlled account to another externally controlled account, this transaction is no different from a transaction made in Bitcoin and is used to transfer Ether from one external account to another external account. If a contract account sends a transaction to another contract account, the contract code of the recipient is executed. Such a transaction is called a message and can be seen as a simple function call between objects. The ability of contracts to send messages to other accounts not only allows the creator of a decentralized application to split up its application into multiple contracts but furthermore makes interaction between different applications possible. As seen, the basic structure of Ethereum is not that different from the Bitcoin protocol. Although not discussed in Chapter 3, Bitcoin is also able to support contracts to a certain extent. The UTXO field of a transaction can not only contain a public key but also a small script expressed in a simple stack-based programming language. There are a few limitations of the scripting language of Bitcoin. Two important ones are that Bitcoin contracts are stateless and the scripting language is not Turing complete. The UTXO field in Bitcoin is either spent or unspent. There is no way to store some sort of contract state. Besides that, the scripting language used is purposefully not turing complete, meaning that among other things loops are not allowed.

5.3 Blockchain and Mining [1, 4, 5]

If we think of Ethereum as a state machine it is obvious that values of the fields of all Ethereum accounts describe the state of the Ethereum system in general, while transactions (if valid) lead to a transition from one state into another. The system's state is stored in blocks on the blockchain. Unlike Bitcoin every block of the Ethereum system contains the entire state of the system. The picture below shows the architecture of Ethereum's blockchain.

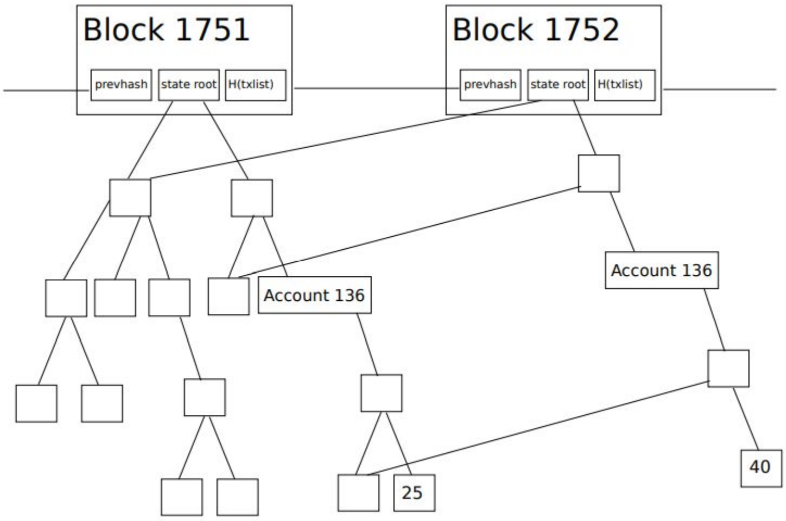


Figure 9: An illustration of two neighboring blocks of the Ethereum blockchain. Unchanged parts of the status (saved in a patricia tree) are referenced. H(txlist) contains the transactions added to the block.

As mentioned before every block contains the entire state of Ethereum. This means that every block contains every account with its respective fields. Because the difference of Ethereum's status between two blocks is relatively small (only a few accounts status are changed) in order to reduce storage, unchanged parts of the tree are referenced. To accomplish this, a tree structure called Patricia tree is used. Detailed information about this special tree structure can be found in the official Ethereum wiki at [9].

Both Bitcoin and Ethereum use the proof of work consensus algorithm in their mining system. In order to incorporate a block into the blockchain, in a proof of work based system a miner has to provide a proof to a specific challenge. This proof should be hard to find but easy to validate. A good example for proof of work is a simple lock. It is hard to find out the correct combination just by guessing, but once found, the solution is easy to validate by

everyone (by just trying out the combination and see if the lock opens).
The proof of work concept as used in Bitcoin comes along with quite a few downsides as well. The two biggest concerns are:

- Waste of energy
- Not resistant to Application Specific Integrated Circuits (ASIC)

A big topic in Bitcoin is the lack of mining decentralization. In recent years miners of Bitcoin switched from PCs to ASICs. These are specialised pieces of computer hardware that exist only to do a single task. In Bitcoin's case the task is the SHA256 hash function. Because it is not profitable for the average miner to go and buy such hardware, more and more mining power shifted to fewer but much bigger mining pools. Currently three mining pools namely F2Pool (~20%), AntPool(~20%) and BTCC Pool (~15%)[10] are in control of more than 50% of the total hashpower of the Bitcoin network. This means that together those three pools mine more than 50% of the blockchains blocks. It's important to say that we are talking about mining pools. Such pools do not mine Bitcoin themselves but only produce and sell the specialized hardware for mining efficiently. In his article on mining [5], Vitalik Buterin (chief scientist of Ethereum) goes into detail about ASICs and how they affect the Bitcoin network. Despite all this the sole fact that these three pools have the power to take over the blockchain, is enough to worry a lot of people in the Bitcoin community because unlike manufacturers of processors like Intel and AMD those entities determine exactly what runs on the ASICs. With a modified proof of work algorithm calle Ethash, Ethereum tries to combat this undesired distribution of the hash rate. In the end mining should be a way of distributing the money over the network. At least this is the egalitarian pursuit. A description of the rather complex algorithm would be out of scope of this article.
Instead of a constant block size like in Bitcoin, Ethereum blocks possess a certain Gas limit that grows over time. The sum of the Gas limits of the transactions contained in the block is not allowed to be higher than the Gas limit of the block. The next section will talk about the Gas unit of Ethereum more closely. At this point it is enough to say that if an account wants to submit a transaction it has to pay for this transaction by using a certain amount of Gas (which can be bought with Ether). The miner of the block that contains this transactions receives that Gas value in Ether. If a block was mined the block validation algorithm of Ethereum works as follows:

- Check if the previous block referenced exists and is valid
- Check that the timestamp of the block is greater than that of the refer enced previous block and less than 15 minutes into the future

- Check that the block number, difficulty, transaction root, uncle root and gas limit are valid
- Check that the proof of work on the block is valid
-
- Let S[0] be the state at the end of the previous block
- Let TX be the block's transaction list, with n transactions. For all i in 0...n − 1 set S[i + 1] = AP P LY (S[i], T X[i]).
- Let S F INAL be S[n], but adding the block reward to the miner
- Check if the root of S F INAL is equal to the final state root provided in the block header. If it is, the block is valid; otherwise, it is not valid

Every miner is obligated to validate every new block that is added to the blockchain in such a way. Such nodes that enforce the ruleset of Ethereum are the backbone of the system and are also called full nodes. Every miner has to be a full node but not ever full node has to mine blocks. On the other hand there are a lot of people that do not want to mine or just cannot run a full node but want to use the Ethereum system anyway. This will be especially true if Ethereum wants to become mainstream. The so-called Light client protocol, which is still in development, aims to allow users to use Ethereum in low-capacity environments such as smartphones, by not enforcing block validation.

5.4 Code execution and Gas [1, 4]

If a user sends a transaction to a contract stored in a block of the blockchain, the contract is executed. The transaction (contract's code) is physically executed on every client that either downloads and/or validates the block, where the mentioned transaction is incorporated in. Because of the turing complete nature of the contracts (loops and messages), this is obviously a problem. It is impossible to determine for every given input whether a turing complete application will finish or not. In Bitcoin this undesired behaviour is avoided by prohibiting loops altogether. It is needless to say, that this can not be the solution for a platform, whose entire purpose is, to be a tool for developing useful decentralized applications.

In order to deal with the halting problem, Ethereum introduced the concept of Gas. Each computational step uses up a certain amount of Gas. As seen before, each transaction has a field called GasLimit. This field defines how much Gas the transaction is allowed to burn, before the code execution stops. Gas only exists during code execution. It is bought prior to the code execution at a certain price (GasPrice), used during code execution and is refunded in Ether to the sender of the transaction, if the code execution did not use all of it. Besides that the user is free in choosing how much he wants to pay for a single unit of Gas. On the other hand the higher the GasPrice the more likely miners will incorporate the specific transaction into the block. As mentioned above there is a fee for every transaction submitted to the blockchain. The fee the miner receives for evaluating a transaction, depends on the amount of Gas used by this transaction and the worth of a Gas unit inside this transaction. Of course miners will only incorporate transactions, that are worth incorporating e.g the fee for validating the transaction outweighs the validations cost.

The smart contracts can be written in a Java-like language called Serpent, which is later compiled down into a low-level stack-based bytecode language to be executed in the Ethereum virtual machine. Data can be stored on the stack, in memory space or directly on the contracts storage. Data stored in the stack or in memory is reseted after the contract finished its execution. In order to store data permanently, the contract accesses its storage in the blockchain. The computational state of the EVM can be described by the following tuple:

(block state, transaction, message, code, memory, stack, pc, gas)

The field block state contains the entire Ethereum state meaning all accounts with their current Ethereum balance and storage. The field pc (program counter) always points to the current instruction. Each instruction affects the computational state in a certain way. For example the instructions ADD and STORE work as follows:

- ADD: pop two items off the stack, push sum of items onto the stack,reduce gas by one, increment pc by one
- STORE: pop two items off the stack, insert the second item into contracts storage at index specified by the first item; used to store data permanently

A complete description of the instruction set of Ethereum and the virtual machine can be found in the yellow paper[4]. Of course a deeper understanding of the EVM or the instruction set is not required to develop Ethereum contracts. As most modern programming languages, Serpent does its best to hide such things completely from the user.

5.5 Problems and future directions

Scalability turns out to be a huge issue in all blockchain applications. The throughput of mainstream payment networks exceeds the current throughput of Bitcoin by a factor of roughly 300. In order to increase the number of trans actions per second, a simple change of the block size limit would be enough. A bigger block size (beside multiple other negative effects) would lead to the situation that it would be hard for an average user to run a full node. At the time of writing this report the size of the Bitcoin blockchain is about 75GB with a growth of 6 MB per hour. A bigger block size could lead to an faster growth of the blockchain. Depending on the development of storage cost it is quite pos sible that the blockchain outpaces the average HDD capacity and only a small set of businesses would be able to manage this load, which would defy the entire idea of Blockchain. The problem of scalability is still subject of ongoing discussions in the Bitcoin community.

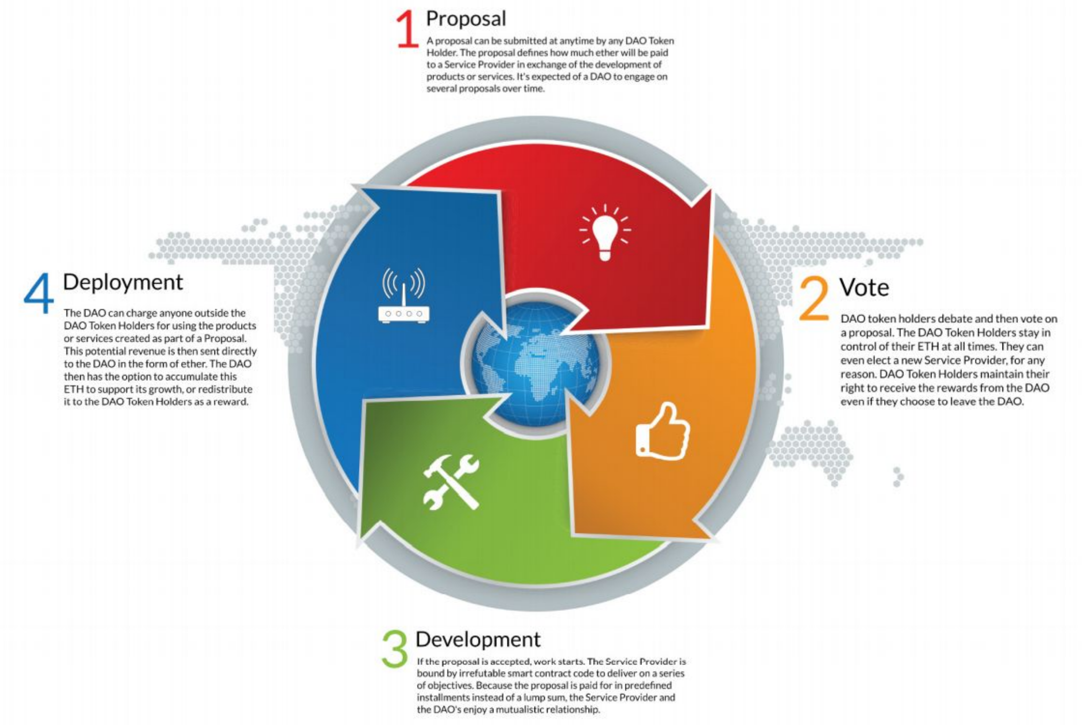
Another main concern is the proof of work approach, which is taken by Ethereum consensus algorithm Ehtash. Although Ethash is ASICs resistant and solves the problem of centralized mining, it does not address the fact that proof of work based mining can be seen as a competition about who can waste the most energy in a certain amount of time. Ethereum tries to develop a consensus algorithm which is fuelled by coins and not real energy. This approach is often called proof of stake. Every single account has a chance of being the creator (owner) of the next block. The chance of an account being chosen to be the owner of the next block is proportional to the amount of coins this account owns. If the total amount in a system is a 1000 coins and an account A is in possession of 10 coins.

Smart contracts by themselves have no way of determining the state of external variables. For example, an agricultural insurance policy which pays out conditionally based on the quantity of rainfall in a given month. Suppose that the contract receives this information via a web service. It is not guaranteed that all nodes that execute the contract will receive the same result from the web service. But the blockchain concept only works if every node reaches an identical status after the execution of all the transactions of a newly generated block. As we will see in the section that introduces an Ethereum application called ”TheDAO" 5.6.1, mechanisms to overcome this problem exist.

Certain applications require privacy. As we saw in previous chapters, data on the blockchain is public by nature. The obvious approach to tackle this problem is the use of cryptography. The Ethereum blockchain aims to support this privacy feature in the future.

5.6 Ethereum Applications: DAO

The DAO is a Decentralized Autonomous Organization that lives as a smart contract on the Ethereum blockchain. There are actually many DAO’s on Ethereum's blockchain. This section is going to focus on a DAO called ”theDAO”. In its first stage (initial creation phase) the DAO sells tokens for Ether. After a certain amount of time the DAO stops collecting money and starts to operate. People in possession of tokens can now make proposals on how to spend the money. Needless to say that the DAO itself does not realise projects but transfers Ether to an accepted (by the token holders) contractor. If the voted projects end up becoming profitable, people will get their fair share of the profit. The lifecycle of DAO is illustrated below.



1 Proposal

A proposal can be submitted at anytime by any DAO Token Holder. The proposal defines how much ether will be paid to a Service Provider in exchange of the development of products or services. It's expected of a DAO to engage on several proposals over time.

2 Vote

DAO token holders debate and then vote on a proposal. The DAO Token Holders stay in control of their ETH at all times. They can even elect a new Service Provider, for any reason. DAO Token Holders maintain their right to receive the rewards from the DAO even if they choose to leave the DAO.

3 Development

If the proposal is accepted, work starts. The Service Provider is bound by irrefutable smart contract code to deliver on a series of objectives. Because the proposal is paid for in predefined installments instead of a lump sum, the Service Provider and the DAO's enjoy a mutualistic relationship.

4 Deployment

The DAO can charge anyone outside the DAO Token Holders for using the products or services created as part of a Proposal. This potential revenue is then sent directly to the DAO in the form of ether. The DAO then has the option to accumulate this ETH to support its growth, or redistribute it to the DAO Token Holders as a reward.

Figure 10: The DAO Proposal Voting Process

A pretty obvious attack vector on this protocol would be the so-called 51% attack. A holder of more than 50% of the DAOs token could make a proposal to transfer all the money to him. This proposal would always succeed and the invested money of the minority would be gone. The naive approach to tackle this potential threat, is to give investors the opportunity to remove their investment at any given time. Obviously this would mean that an investor had to check all proposals in order to not miss the malignant one. For this purpose the token holders vote for a so-called curator.

The concept worked quite well and the DAO managed to collect over 150 Million Dollars in its first funding period. Because of that and the fact that DAO was actually hacked right after its first funding period we chose to write about this particular Ethereum application. The hack of DAO shows that even if we make the assumption that there are no attack vectors against the Ethereum system itself (which of course nobody can do), the contracts will always be the weak spot of Ethereum. The hacker managed to exploit a software bug in the contract and stole about a third of the money the DAO had managed to collect (60 Million Dollars). A detailed explanation of the attack can be found at [12]. The attacker exploited the fact that when a contract sends Ether to another contract it also executes the code in the destination contract. The destination contract is constructed by the attacker in such a way that it immediately asks the original contract to send the Ether again. If the original contract does not update the balance before sending the Ether, the attacker is able to withdraw the Ether several times. Even more interesting than the attack itself is the way the Ethereum community dealt with the attack. Despite the fact that a key promise of the blockchain is that all that is on the blockchain, remains on the blockchain, the Ethereum community decided to do a hard fork. This was done by convincing enough miners to stop working on the current blockchain and fork the blockchain at the block right before the block that contained the malicious transaction. Even though the fork was a success, it is questionable how this behaviour will affect the reputation of Ethereum as a whole.

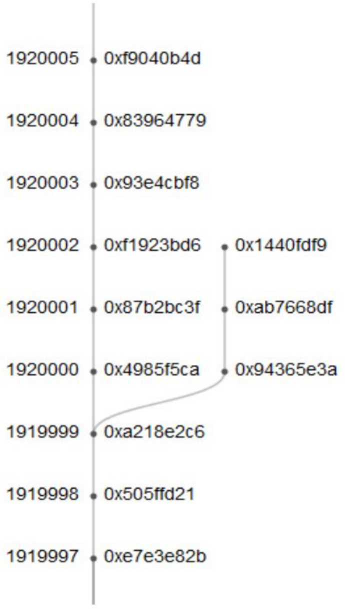


Figure 11: Ethereum: hard fork

6. Conclusion

This report tried to give an introduction into the wide spectrum of use cases of the blockchain. With Bitcoin and Ethereum we discussed probably the two most prominent state of the art representers of blockchain technology. We saw that in recent years, people started to move away from building only economic systems on top of the blockchain. In times of a quite centralized Word Wide Web (Facebook, Google, Amazon etc.) this development seems quite refreshing. If successful, decentralized block chain applications could play a major role in the democratisation of the internet, by shifting the power from the big players back to the users. On the other hand this field is still pretty young, prone to problems and seemingly quite far away from being able to replace the big players. Only time will tell how this battle turns out.

References

[1] V. Buterin, “Ethereum white paper.”

[2] M. Swan, Blockchain: Blueprint for a New Economy, 1st ed. O’Reilly, February 2015.

[3] E. P. Gareth W. Peters, “Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart con tracts on the internet of money,” November 2015.

[4] E. Y. P. Gavin Wood, “Ethereum: A secure decentralised generalised trans action ledger.”

[5] V. Buterin. On mining. [Online]. Available: https://blog.ethereum.org/2014/06/19/mining/

[6] C. Jentzsch, “Dezentralized autonomous organization to automate gover nance,” 2016.

[7] J. Peterson and J. Krug, “Augur: a decentralized, open-source platform for prediction markets,” 2015.

[8] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 20008.

[9] Ethereum wiki, patricia tree. [Online]. Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree

[10] H. Distribution. [Online]. Available: https://blockchain.info/pools

[11] Ethereum wiki, problems. [Online]. Available: https://github.com/ethereum/wiki/wiki/Problems

[12] A. M. Zikai Alex Wen. Scanning live ethereum contracts for the ”unchecked-send” bug. [Online]. Available: http://hackingdistributed.com/2016/06/16/scanning-live-ethereum-contracts-for-bugs/