Replacing Paper Contracts With Ethereum Smart Contracts

Contract Innovation with Ethereum

Wesley Egbertsen, Gerdinand Hardeman, Maarten van den Hoven, Gert van der Kolk and Arthur van Rijsewijk

June 10, 2016

Abstract

This research finds out what criteria Ethereum needs to fulfil to replace paper contracts and if it fulfils them. It dives into aspects such as privacy and security of the blockchain and its contracts, and if it is even possible at all to place a contract on the blockchain. However, due to the variety of contract clauses and a large privacy setback, it is not recommended to place paper contracts on the Ethereum blockchain.

1 Introduction

In spring 2016, impactful decisions or even some of the fundamental life decisions take a lot of time to be fully integrated with a city's municipality database. For instance, when one is buying a house, a notary will need to form a contract between the seller and the buyer, meaning there is a third party involved in making the transfer legal which takes a lot of time and costs a lot of money. The cause of this is that contracts need to be signed by authorised people and checked by authorised and trusted third parties. Signing these has to be done according to the law. Not only is this often required via notaries but all jobs that have the authority to make a contract stating something happened at a particular point in time such as lawyers, officers and prosecutors. The complete process takes a lot of money and time and can be done faster if it is done another way.[1] Ethereum is a blockchain based technology that comes with the option to store code in the blocks on its blockchain. Bitcoin on the other side is merely a cryptocurrency. The Bitcoin blockchain does not support smart contracts yet, making Ethereum a possible and viable candidate for replacing paper contracts with smart contracts.[2]

To prove this theory, several subquestions must be answered. The first three subquestions are an introduction to what Ethereum is and how it works. The last two subquestions will return a list of criteria that will together answer whether or not Ethereum can replace paper contracts in a safe and practical way.

"What is Ethereum?" explains the idea behind Ethereum and how it came to be. It also explains what makes Ethereum unique and what its actual uses are. This subquestion gives a basic understanding of Ethereum needed for the following subquestions.

"How does Ethereum work?" shows the technical side of Ethereum. This subquestion dives into the technology that drives Ethereum, such as the Blockchain, Ether, Accounts and Smart Contracts. A phase of experimentation with Ethereum software was needed to complete this subquestion.

"What defines a paper contract?" is a small subquestion that gives background information on paper contracts, its bureaucracy around it and a list of properties all paper contracts have which is needed for the following subquestion.

"What kind of paper contracts are best suited to be replaced by Ethereum contracts?" explains the possibilities and limitations of smart contracts. It also shows what paper contracts might be capable of converting to a smart contracts, which is necessary to be able to pick a contract for the proof of concept in the last subquestion.

"What are the risks of Ethereum?" is a subquestion that is imperative to this research. It shows tense parts of Ethereum like the volatile price and the 'centralised' foundation, but also more of the technical side such as wallet security and the privacy and security of the blockchain.

"Proof of concept" is a chapter that contains information about a decentralised application that has been made as a proof of concept. This DApp shows the ability that a paper contract can be converted to a smart contract on the Ethereum blockchain. The goal of this DApp will be explained as well as the technology used to create this Dapp. Finally the result of this proof of concept will be explained.

2 What is Ethereum?

This subquestion explains the concept that defines Ethereum, how it was invented, why it was invented and how it influences the modern world.

2.1 History

The history of the World Wide Web consists mostly of centralised applications. Even though the load is often distributed to several servers, there is always a third party involved. For instance, payments and other data went from a client to a centralised server and the other way around.[3] Depending on third party servers and companies used to be a volatile business due to privacy and centralisation. In the early 1980's protocols arose based on the so-called 'Chaumian Blinding', coming from the founder of the digital electronic currency 'ecash' David Chaum. The Chaumian Blinding provided a substantial amount of privacy, but due to their centralised underlying protocols they mostly failed to stay alive.[4] In the late 1990's a proposal from Wei Dai called 'b-money' introduced the idea of generating money by solving computational puzzles as well as an idea for a decentralised protocol.[5] A few years later the computer scientist Hal Finney introduced a concept of a system which uses parts of Wei Dai's b-money to create a concept for a cryptocurrency.[6] This idea, however, failed as well due to relying on trusted computing on the back end. By 2009, a decentralised currency was successfully implemented for the first time in practice by an anonymous person or organisation who call themselves "Satoshi Nakamoto". For this reason, Satoshi Nakamoto's development of Bitcoin in 2009 has been labelled as

a game-changing development in the economic and technological world.[7] More interesting however was the underlying decentralised blockchain technology that relies on a peer-to-peer network of nodes.[8] The mechanism behind this was a breakthrough because it solved two major problems. It set a straightforward and practical set of rules that allowed nodes in the network to agree unanimously on updates to the blockchain, but it also provided free and easy entry into the process, solving the mostly political problem of deciding who gets to influence the consensus and preventing attacks at the same time.

Over the following four years, there was a massive increase in bitcoin-based cryptocurrencies which received the term 'altcoins' (alternative coins). Most of these altcoins are direct copies of Bitcoin's source code that have minor changes made to them as the name, coin supply and confirmation time.[9] In late 2013 Vitalik Buterin initially described Ethereum with the goal of creating a platform on a custom blockchain that offers users to build decentralised applications with a Turing complete language, which will be elaborated in later subquestions. Unlike most of the altcoins, it was described as being part of a group of projects with the potential to extend blockchain use beyond Bitcoin's peer-to-peer money system.[10]

2.2 Why is it unique?

Ethereum is a project that tries to build a technology on which all transaction based state machine concepts may be developed. It intends to provide the end-developer with an integrated end-to-end system for developing software on an unexplored compute paradigm in the mainstream. Ethereum provides an alternative protocol for building decentralised applications. It provides a different set of trade-offs that will be very useful for a large class of decentralised applications.[2] Ethereum emphasises situations where rapid development time, security for small and rarely used applications, and the ability for different applications to very efficiently interact, are necessary.[7] One can achieve this by creating what the ultimate abstract foundational layer is: a blockchain including a built-in Turing-Complete programming language, enabling anyone to write a smart contract and decentralised applications where they can create their arbitrary rules for ownership, transaction formats and state transition functions. Smart contracts are boxes that can hold multiple values and will be unlocked when certain conditions are completed. These can be built on top of the platform and with more power that that offered by Bitcoin, because of the added power of Turing-completeness, value-awareness, block-chain awareness and state.[7] According to Vitalik Buterin, a inventor and co-creator of Ethereum, the advantages of Ethereum are the following:[11]

1. Smart contracts - Funds can be stored into a contract. If the owner of the contract dies, as defined by becoming inactive for six months. The funds become available for the next-of-kin. An agreement between the company and shareholder can be written in self-executing code. The votes and proposals are being done on the blockchain and executed automatically if and only if a proposal gets a sufficient amount of support.

2. Computational resource marketplaces - Ethereum contracts can use Merkle trees; to check if the resource you are requesting is still being stored on the blockchain and send them a payment if it is.

3. Decentralized DNS - Ethereum contracts can store database mapping names to public keys or addresses. This way the contracts can be used to register websites, decentralized applications, contracts, pieces of static code, etc. Essentially a decentralized phonebook for anything.

4. Financial applications - the Ethereum repository for the programming language Serpent contains a contract that implements Kickstarter-like crowdfunding in 40 lines of code. There can be also be the opportunity to do prediction markets.

5. "Smart property - imagine self-driving cars that are tied to contracts, so anyone can send a transaction to make a payment and then automatically receive the right to use the car (ie. the car will recognize messages signed by the user's private key, controlled by some specially designed app on their smartphone) for some short time."

2.3 Actual use

By searching through the Google search engine, not much information on companies using Ethereum is found at fist, while the Ethereum YouTube channel shows that more than ten small start-up companies are pitching their projects and products built around Ethereum. However, on big multinationals using Ethereum is not much to find. In January 2015 IBM and Samsung have created a functional demo product. It uses the Ethereum smart contracts to operate. IBM states the reason they chose Ethereum as underlying blockchain protocol is that Ethereum blockchain helps achieve coordination and transactions on the blockchain. Ethereum manages the contracting between devices. Devices can create and set their responsibilities and permissions. This technology makes the devices almost entirely autonomous. This is a big thing for the future of the Internet of Things.[12]

In March 2016, there are a few companies that already have integrated Ethereum into their businesses. One of those companies is called *Slock.it*. *Slock.it* is a German company that uses the Ethereum blockchain to be able to rent, sell or share anything without the need of a middleman. *Slock.it* also created usable locks called Slocks that connect to the Ethereum blockchain. "When someone purchases a Slock, it will be linked to the Slock smart contract in the Ethereum blockchain and controlled by it," says *Slock.it* co-founder Christoph Jentzsch. "The owner of a Slock can set a deposit amount and a price for renting his property, and the user will pay that deposit through a transaction to the Ethereum blockchain (without us), thereby getting permission to open and close that smart lock through their smartphone."[13]

"The deposit will be locked in the Ethereum blockchain until the user decides to return the virtual key by sending another transaction to the Ethereum blockchain," continues Jentzsch. "Then the contract will be automatically enforced. The deposit will be returned to the user minus the price for the rental, which will be automatically sent to the owner of the Slock. All of this happens without any assistance from a third party!"[13]

One of the big projects the company *Slock.it* is working on involves electrical car charging stations. Slock.it has a partnership with RWE, a large German power company, to change the way electric cars are charged. "Cars with digital wallets will be able to "talk" to autonomous electric charging stations which

use smart contracts to allow users to rent the station, put up a deposit, charge their car, then get their deposit back."[14]

Another company that uses the Ethereum blockchain is Ujo. Ujo is bringing the music industry to the blockchain. Phil Barry, one of the founders of Ujo, explains in his talk on Ujo, that songs are often owned by multiple artists.[15] Moreover, each of these artists has the right to a different share of the earnings. For instance, the singer gets 20%, the music directors gets 5%, and some other artists get 1.2%. The odd thing is that there is no definitive record of this information. So every record company and every rights society have their version of this information, and they have a hard time agreeing which version is correct. Ujo claims to bring the solution to this problem. By adding the music rights on the blockchain, a decentralised database, the problem of different versions of music rights documents is solved. Also, anybody can use the registered content provided that he or she meet the terms of the policy. The right to do so is transferred automatically through a smart contract. Payments are delivered to individual stakeholders instantly and automatically using digital currency, eliminating the need for intermediaries. Ujo's open platform provides a shared infrastructure upon which infinite potential services, applications and business models can be built.[16]

2.4 Summary

Ethereum is a blockchain based technology. This technology comes forth from Bitcoin's blockchain. This technology is not necessarily focused on building a safe and trusted cryptocurrency, but on creating the possibility to write decentralised applications. Ethereum is a blockchain with a Turing-Complete programming language, allowing anyone to write smart contracts.

From our research on companies working with Ethereum, we can conclude that there are no big multinational corporations that have integrated with Ethereum yet. However, the Internet of Things sector is interested in the many possibilities it brings. As explained in section 2.3 IBM in cooperation with Samsung have made a functional demo product. Start-up tech companies are still trying to find the right use for the Ethereum blockchain technology. Those companies observe real life problems and solve them by using the Ethereum blockchain.

3 How does Ethereum work?

In this subquestion, we will explain how Ethereum works on a slightly more technical level.

3.1 Blockchain

A blockchain is a series of data (transactions) chained together. The blockchain technology is decentralised, which means that there is no single party which holds control. This decentralisation has the advantage that it is not necessary to trust a single party, as everyone on the blockchain collectively decides what happens.

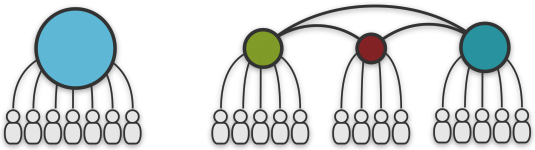

Figure 1: Centralization vs Decentralization

Vitalik Buterin, the founder of Ethereum, says the following about blockchains: "They allow for a large number of interactions to be codified and carried out in a way that greatly increases reliability, removes business and political risks associated with the process being managed by a central entity, and reduces the need for trust. They create a platform on which applications from different companies and even of different types can run together, allowing for extremely efficient and seamless interaction, and leave an audit trail that anyone can check to make sure that everything is being processed correctly."[17] It reduces the need for trust, because all transactions can be viewed publicly.[18] The security of the blockchain does denote that the performance of a decentralised application is not as good as a centralised application. The performance drop is caused by the need to validate transactions on the blockchain, which is done by miners.[19] There are plans to increase the performance of the blockchain. On a short term this would mean migrating from a Proof of Work to a Proof of Stake algorithm. The Proof of Stake algorithm will result in significantly faster block chains, and it is also expected to result in more transactions per second.[20] However, a decentralised application will never outperform a centralised application, based on the nature of the blockchain. On the blockchain, every transaction has to be processed by every node, in contrast to centralised applications, where a transaction only has to be executed once. Also, because this is a peer to peer connection, every transaction has to be verified by a signature. This is computationally complex and thus costs time.[21]

3.1.1 Blocks

The blockchain consists of blocks; these blocks are like a journal by recording a series of transactions. A block can be seen as a container, to which only the owner has the key, however everyone can see the metadata. In contrast to Bitcoin, Ethereum has no fixed limit on the size of one block.[22] The Ethereum block size is limited to a gas limit instead. This gas limit is not set, but it is expanding based on the long-term exponential moving average of the gas used in recent blocks.[23] The concept of gas will be explained in section 3.2.

3.1.2 Mining

Mining is the process of adding block to, and verifying computation on the Ethereum blockchain. Miners produce new blocks, which other miners check for validity. Currently blocks are validated by checking whether or not they contain a Proof of Work of a given difficulty. In Ethereum 1.1 this Proof of Work system will likely change to a Proof of Stake system. Anyone can participate in the mining process, but the chance of finding a valid block increases with the power of the computer performing the calculations. At the publication date of this

document the block time is set to 12 seconds, so the difficulty of the block will adjust based on the time of the previous block.

Sometimes a miner will find an uncle block; an uncle block is a block which is initially valid but is surpassed by another, faster block. However, this uncle block will still be rewarded with $\frac{7}{8}$ of a full block value and their hashes will be added to the valid block. A maximum of two uncle blocks can be added to a valid block. The miner of the valid block also receives $\frac{1}{32}$ extra ether per uncle block included. This ensures that these blocks still contribute to the security.[24, 25]

The rewards for successfully mining a block can be found in section 3.2.

3.1.3 Mining pools

Mining can be done individually or in a pool of miners. Miners in a pool can mine together, and the reward will be split between all the members in the mining pool. A mining pool has a much better chance of solving a block and winning the reward compared to mining alone. Mining pools are a way to encourage small-scale miners to stay involved. There are multiple mining pools for Ethereum. Each with their own set of rules regarding fees and rewards. Mining pools pose a risk as they have a high hashrate, however, they are not allowed to go over 51% of the hashing power of the network. If a single pool controls more than 50% of the network's computing power, it could cause a disturbance.[26] The concept of the 51% attack will be explained in section 6.4.

3.2 Ether

Ether is used as fuel for the distributed application framework Ethereum as well as being a form of payment. The purpose of Ether is to give developers an incentive to develop applications of a higher quality as wasteful code costs more.

The base reward for successfully mining a block is five Ether. If another miner finds a solution as well, but is not fast enough to be included in the blockchain, it will turn into an uncle block. Miners of uncle blocks receive 4.375 Ether, while also raising the reward for the block in which the uncle block is included by $\frac{1}{32}$ per uncle block included. Because there is a limit of two uncles per block, if another miner also finds a solution, but this block can not be included in the blockchain, this miner will usually receive 2-3 Ether. There is currently a maximum growth of 18 Million Ether per year, but it is unlikely to be kept at this limit, as there is a new algorithm called Casper under construction, expected to be released in 2017.[25, 27, 20]

Transactions on the Ethereum platform need fuel to execute, as further explained in section 3.3.1. In Ethereum this fuel is called gas, this gas is purely used internal and is paid for with Ether. The gas price can be changed by developers, so the cost of execution of the code will not necessarily change with the fluctuation of the Ether price.

To enable easier calculations, Ether also has some subdenominations:

- Wei - 10^0
- Szabo - 10^{12}
- Finney - 10^{15}

- Ether - 10^{18}

When discussing costs, Wei is the most commonly used term.[2]

3.3 Accounts

There are two types of accounts in Ethereum, namely the following:

- Normal or externally controlled
  This is an account controlled by a private key. If a person owns the private key associated with the account, they have the ability to send Ether and messages from it.

- Contracts
  This is an account that has its own code and is controlled by code. A contract is thus basically a regular account but with the extra option of containing code. Though it can only fire a transaction in response to other transactions that they have received. This means that all actions on the Ethereum blockchain start by transactions fired from normal accounts.

In paragraph 3.4 we will discuss contracts, but for this section, the general concept of accounts will be explained. The state in Ethereum is made up of these accounts, and each account has a 20-byte address which is unique.[28] Because an address is made up of 20 bytes, its uniqueness is limited. 20 bytes is equal to 160 bits, so there is a 2^{160} chance that a newly generated address is already used. Thus the likelihood that a newly generated address is already used, is about 1 in 1,461,501,637,330,902,918,203,684,832,716,283,019,655,932,542,976. This limit should not be a problem for now. Such an address is part of an Ethereum account. An account consist of following fields:

- Nonce, a counter so that transactions can only be processed once.
- The account's Ether balance.
- Contract code, if existing.
- Storage, which is empty by default.

As told before in this section, the state is made up of accounts. Each block in the Ethereum blockchain contains the state information from that moment. It could be seen as inefficient at first glance because the entire state stores itself in each block, but the efficiency should be comparable with Bitcoin. It is similar because the state gets stored in the tree structure, and with every block, only a small part of the tree is changed. Because of this, the tree between two blocks is mostly the same. Therefore, the data can be stored once and referenced twice with pointers, such as the hashes of the subtrees. Because all of the state information is part of the last block, the entire blockchain history does not need to be stored. For example, if Bitcoin would also use this strategy, it is calculated that when this procedure gets applied to Bitcoin it can provide 5-20x savings in space.[7]

3.3.1 Transactions

In Ethereum, the term transaction is used to refer to a signed data package that stores a message to be sent from a regular account. A transaction contains the following information:

- The recipient to which the message gets sent.
- A signature that identifies the sender.
- Amount of Ether to transfer.
- An optional data field.
- `STARTGAS` value, this defines the maximum number of computational steps the transaction is allowed to do.
- `GASPRICE` value, this determines the fee the sender must pay per computational step.

The data field can contain data that a contract can access. As an example, if a contract is providing a way for people to vote, the data field could include some identifier for that person and the party for whom they are voting. The `STARTGAS` and `GASPRICE` fields are essential for the anti-denial of service model from Ethereum. Meaning that accidental or hostile infinite loops are to be avoided. That is why each transaction is required to set a limit to how many computational steps it can do. Usually computational step costs one gas, but it can be higher if it is computationally more expensive or if it increases the amount of data that must get stored as part of the state. Also, for each byte of the transaction data, there is a fee of 5 gas. So the intent of this fee system is to require an attacker to pay respectively for every resource they use. Therefore, an attack with much computing power would cost a large quantity of Ether.[7]

3.3.2 Messages

As explained in subsection 3.3, contracts can only fire a transaction when they receive a transaction. When a contract sends a transaction, it's called a message. A message is like a transaction; the only difference is that a contract fires it. A message contains the same attributes as a transaction as seen in subsection 3.3.1, except it does not have the `GASPRICE` value as this is defined in the first transaction that is fired that caused a message to fire. For example, if a normal account, account 1, sends a transaction to account 2 with 500 gas, and account 2 only used 100 gas before sending a message to account 3, and the execution of account 3 only uses 300 gas, then account 2 can use another 100 gas.[7]

3.4 Contracts

As explained in subsection 3.3, a contract is an account that also has its own code and is controlled by code. A contract can only send a transaction in response to a transaction it has received.[28] The code from a contract activates whenever it receives a message, allowing the contract to read and write to its storage and send other messages or create contracts. Contracts in Ethereum should not be seen as something that should be fulfilled or complied with. Contracts should

be perceived as autonomous agents, meaning that they can carry out some set of operation on behalf of a user with some independence. Moreover, with that represent some of the user's goals that are programmed in the contract. So contracts are like autonomous agents that live inside of the Ethereum execution environment, always executing its code when it receives a message or transaction and having control over their own Ether balance and their own key/value store to keep track of persistent variables. The key/value store is the contract's long-term storage, unlike the stack and memory that is used when the contract's code starts running when triggered by a transaction or message.[7]

Appendix A explains how to build a contract that returns "Hello world". It also shows how to deploy an established contract and how to invoke a function from the deployed contract.

3.5 Summary

Ethereum is a technology based on the blockchain. Blockchain technology enables decentralisation because everyone has a copy of all the information stored on the blockchain. Every time a transaction executes, it is broadcasted between users, and will be confirmed by a process called mining. This mining process ensures that no individuals can control or change what will be saved in the blockchain.

Because this process requires a certain amount of computational power Ethereum has its own cryptocurrency called Ether. Ether is used to pay for transactions and is used to compensate the people doing the computations necessary to mine the blocks.

There are two types of accounts in Ethereum, a normal account, and a contract account. A normal account allows a person to send Ether or messages if a person owns the associated private key from the account. A contract account is an account with its own code and is controlled by code. It can only fire a transaction in response to a transaction that the contract has received. The state in Ethereum is made up of these accounts. Each account has its own unique 20-byte address. For the contract code to function properly without denial of service attacks. Each transaction must have a `STARTGAS` and `GASPRICE` field. The `GASPRICE` is the fee the sender must pay per computational step and the `STARTGAS` is the maximum number of computational steps the transaction can do. A message is basically the same as a transaction, but it is sent by a contract and does not have the `GASPRICE` value, as this was already defined by the first transaction that caused all the events to fire.

The code in a contract activates whenever it receives a message or transaction, allowing the contract to read and write to its storage, and in turn send messages and create contracts. These contracts should not be seen as something that should be fulfilled or complied with. Contracts should be seen as autonomous agents, meaning that they can fulfil a set operations on behalf of a user with some independence. All in all, contracts can never execute their own code by themselves, only when they receive a transaction or message. A message can only get sent if the contract that is sending the message received a transaction, thus meaning that every execution of code from contracts in the blockchain is started by one transaction from a normal account.

4 What defines a Paper Contract?

There are several different kinds of paper contracts, each with their own set of properties. This subquestion explains what defines a paper contract and list the properties that apply to each kind of paper contract. It is important to know what a paper contract consists of before moving to the next subquestion. Once a list of properties is defined, they are compared to Ethereum smart contracts, and are checked if it is even possible to replace paper contracts properly.

4.1 Background Information and Bureaucracy

A contract is an agreement between two parties: one that delivers, and one that pays. However, traditionally a paper contract takes a long path through several middlemen and third parties such as notaries, lawyers, officers and prosecutors. This process takes a lot of money and time that can be spent on other important matters.[29, 30]

4.2 List of properties of paper contracts

Every existing contract is based on two or more entities reaching an agreement. These agreements could be all sorts of deals. Some of these agreements are written down in a paper contract. Every contract has its individual properties, also called clauses, and these may differ from other contracts. Some features exist in every contract. All common contract properties are listed and explained below.

Parties
Anyone can participate in a paper contract. However, there are some exceptions. For example, some parties like minors, felons or people of unsound mind can not enter certain types of contracts. Contracts must identify who the parties are. Some contracts just use names. Others are more comprehensive.[30]

Consent
A valid contract requires each parties' consent. The consent must be free, mutual and communicated to each other. Thus, if one of the parties forces the other party to sign a contract, that party has not signed out of consent and can rescind the contract.[30]

Object
The object is the matter being agreed upon. This matter is sometimes called the subject. Objects must be legally enforceable terms and conditions.[30]

Consideration
All contracts require consideration, which means each party must gain something. The benefit can be an effort or a product. Some companies will never sign stating they will have to make some effort. Other companies only sign contracts stating they have to deliver a product that lives up to the requirements. A contract can also state restrictions such as to not sell their house to anyone else for thirty days.[30]

These four properties are mandatory in a contract. There are however more properties; that depend on the different kind of contracts. Of the well over 50 properties, a few are listed below.

Notice
Some contracts contain a period of notice. The notice is usually the term after one of the parties informs the other that the contract will end. In a contract, the notice shows how long the period will be before the contract ends. Notices are usually seen in contract for house rental, employment, etc.[31]

Confidentiality
Other contracts contain a confidentiality clause. The confidentiality clause is a clause in which certain information is labeled private and prohibited from being disclosed or distributed to anyone other than specifically identified individuals or organisations.[32]

Indemnification
Certain contracts contain an indemnification clause. In an indemnification clause, one party agrees to be financially responsible for specified types of damages, claims or losses. The other party signs to reimburse the other party, may the event present itself.[32]

The law states that state courts may intervene in disputes between conflicting parties. When there is a dispute concerning a contract, the party that suffered a loss may bring a claim for compensation. This is governed by Authorities of law of the country which carries out performance of the contract unless specified in the contract.[33]

4.3 Conclusion

There is a clear definition of a contract: An agreement between two or more parties for the doing or not doing of something specified. Every contract must contain a description and some information about both parties and the object. Both parties must gain something from signing a contract and both parties must give consent. These are the basis of a contract. Contracts could also contain more than those basic properties. A contract can have clauses, which are distinct articles or provisions in a contract, treaty, will, or other formal or legal written document.

Any contracts concerning the law must go through several middleman and third parties. Because of this, most contracts take a lot of time and cost a lot of money to become legally enforceable.

5 What kind of paper contracts are best suited to be replaced by Ethereum contracts?

In this subquestion, the possibilities and limitations of smart contracts will be explained. Besides that, some examples from paper contracts as smart contracts are stated.

5.1 Possibilities and Limitations

Storing a contract on the blockchain could be a wise choice as one will probably need contracts, especially business-, trade- or other necessary contracts, to be safe and trusted.[34] Blockchain technology provides secure storage as explained in section 3.1 and section 6.1. As a matter of fact, the NXT-foundation already owns a start-up company in Singapore storing commercial contracts on the blockchain.[34] This means there is a possibility to store commercial contracts on the blockchain.

In the previous chapter, the different kind of contract properties have been discussed. These features come with lots of possibilities and limitations when it comes to replacing them with an Ethereum smart contract. In other words, these properties either can or can not be translated into code for a smart contract. Therefore, if a contract has to be automated by Ethereum smart contracts, the contract's clauses need to be able to be translated into code. Otherwise, this contract can not be fully automated by smart contracts.

For instance, when one has a purchase agreement and wants to automate the contract for that purchase agreement, there is a clause stating the transfer of the purchased item, or multiple items. This is a physical act and therefore cannot go through the blockchain. A smart contract can trigger a method in the smart contract changing the ownership of the purchased item or its items. This action can be triggered by a DApp (Decentralized application). Both parties can use the user interface of the DApp to trigger actions in the smart contract.

5.2 Example of paper contract to smart contract

In this subsection, examples of paper contracts will be given and explained how these would fit as smart contracts on the Ethereum blockchain.

5.2.1 Purchase Agreement

A purchase agreement is the event where one person is selling the other person goods, and when the buyer agrees to buy those goods. This example explains how the purchase could be done without an intermediary by using Ethereum. The problem with not having an intermediary is that the buyer or seller could be a scammer. Typically, both parties send the money or items to an escrow intermediary. An escrow is a third party that receives the goods and verifies the purchase agreement. If both sides have fulfilled their part, the purchase agreement will be met. But this can also go wrong if the escrow is a scammer.

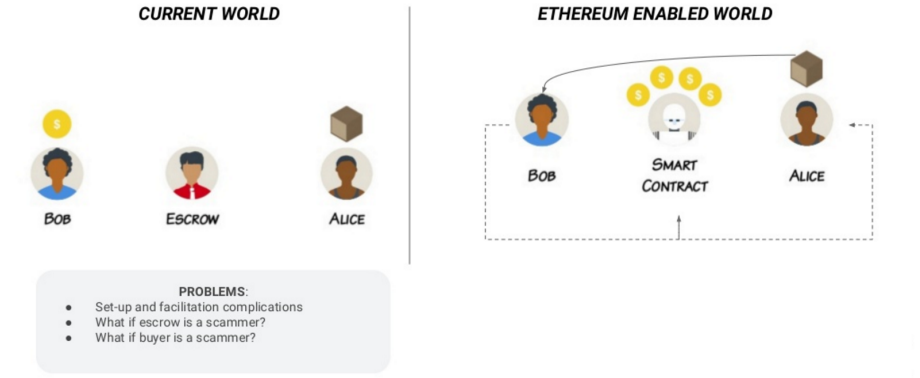

Figure 2: Regular purchase compared to an Ethereum purchase

With Ethereum, one can construct a contract that acts as an intermediary between the seller and buyer. A smart contract does exactly what it is programmed to do. It is a bit more complicated because it has to interface with the real world as one can not send goods to the contract. An idea by Oleg Andreev[35] is that the seller puts a deposit of two times the value of the items. The buyer also puts the same deposit in the contract. At this point, the buyer and seller can not withdraw their money from the contract. Then the seller can send the items to the purchaser, who in response can verify the goods and notify the contract that the goods have arrived. Once the buyer has verified this, the contract sends the value of the goods onefold back to the purchaser, and sends the value of the goods twofold back to the seller. The contract will also send the price of the goods the buyer paid to the seller. So in short the buyer has received their items and one time the value of the goods, and the seller has received three times the value of the goods. The seller receives three times the value of the goods because the seller made a deposit to the contract, with two times the value of the goods and the money the buyer is paying. If the purchaser verifies that the received goods are not the right goods, the buyer can send back the goods to the seller. Then the only option for the seller is to

refund the buyer. This is the only option as the seller also wants the deposit back that was made by them earlier.[36]

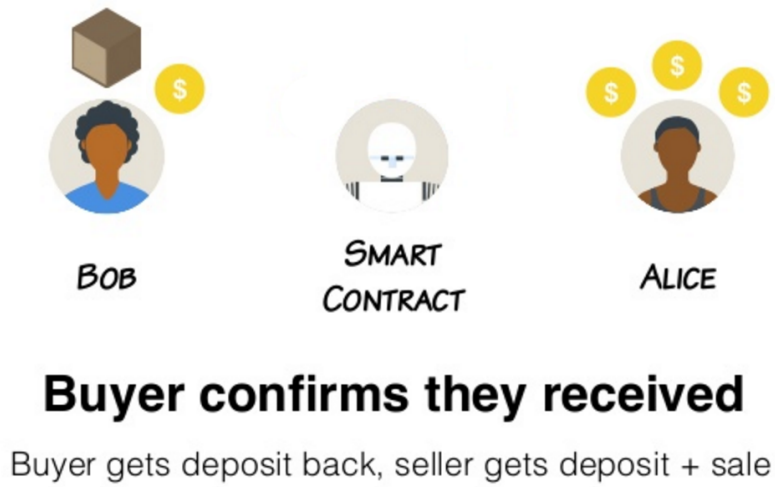

Figure 3: Purchase confirmed with smart contract

5.2.2 Diploma

Fake diplomas and degrees are a global problem. For example, these days people can fake diplomas with Photoshop or people can buy a diploma from a diploma mill. A diploma mill is a company or organisation which offers illegitimate degrees and diplomas for a fee. The diplomas from a diploma mill are hard to differentiate from a real diploma. Thus allowing people to buy these diplomas, and act like they have graduated in that field of work.[37]

For example, schools can use the blockchain for certifications of diplomas. The Leonardo da Vinci Engineering School, a renowned academic institution in Paris, has already thought of this idea. They want to issue and certify diplomas using Bitcoin's blockchain. The school has not yet decided if it will develop their own tools and applications with the blockchain or use established platforms. Cyril Grunspan, educational director of the Financial Engineering and Mathematics department, thinks that the easiest way would be a link to http://blockchain.info. Also noting that employers could access their school's website to verify diplomas. The school would still issue diplomas through paper, alongside certifying them on the blockchain.[38]

The immutability of blockchains makes the idea of issuing and certification of diplomas on the blockchain an interesting idea, because once the diploma is issued on the blockchain, it is immutable. Allowing potential employers to verify diplomas of schools using blockchain technology.[39]

5.2.3 Voting

With voting in a democracy, there are problems such as corruption, voter intimidation and fraud. With Ethereum, this issue could be tackled by giving transparency to the voting process.

In Ukraine, a group of officials signed a memorandum in February 2016. In this memorandum, they wanted to move multiple levels of elections to the Ethereum blockchain using E-vox.[40] E-vox is a platform developed by a group of companies including Ambisafe, Distributed Lab and Kitsoft. With this memorandum, the officials want to create a decentralised, transparent and accessible system for group decisions making via blockchain-based mechanisms. They want to use this system for political primaries, elections, online petitions or referenda.[41]

With Ukrainian officials taking Ethereum and the blockchain seriously as an instrument for voting, it is an idea that is to be taken seriously. Ethereum smart contracts can store the vote of a person on the blockchain and make it permanent, as the blockchain is immutable.[39] This will allow the voting to be transparent which decreased the chance of fraud. A smart contract in this example can save the information of the voting person along with his or her vote.

5.2.4 Residential Lease Agreement

Residential lease agreements are agreements that can be altered. When the tenant alters his copy of the Residential lease agreement. There will be a conflict because the landlord sees that the Residential lease agreement does not match with his version of the contract. If the Residential lease agreement is stored on the blockchain as a contract, then the contract cannot be altered. Companies are already starting their own Ethereum blockchain on which they can store the Residential lease agreements. The Smart Tenancy Contract is an online software tool created by Midasium. The Smart Tenancy Contract is for independent landlords or property managers. These Smart Residential Lease agreements are stored on the blockchain and can be digitally signed by both parties, unlike a traditional contract. The contract can receive a bond payment or a direct debit for a rent payment from the tenant.[42]

5.3 Conclusion

There is a company already storing contracts onto the blockchain. This company is doing so for safety and to make sure contracts are not forgeable. However, it is done without the use of smart contracts and purely for the reasons as stated in the previous sentence, such as safety. If someone has the intention to automate a contract, the first question to be answered is whether the clauses of the contract can be translated into code for these smart contracts. If all the clauses are translatable, smart contracts based on that contract can be made and put onto the Ethereum blockchain.

The examples from paper contracts to smart contracts in this subquestion are all ideas that are taken seriously by professionals. Thus, should be seen as potential future implementations, if the blockchain technology will get more attention from professionals.

Referring to the subquestion: "What kind of paper contracts are best suited to be replaced by Ethereum contracts?": There is no 'best' paper contract to be replaced by smart contracts. As explained earlier, if all the clauses from a paper contract are translatable to smart contract code, the paper contract is suited to be replaced by a smart contract. The only question that remains is

whether or not people are willing to enter the digital age with paper contracts, and maybe the smart contracts of Ethereum offer the solution.

6 What are the risks of Ethereum?

Ethereum poses several risks due to its young age, head company and the vague legal area it finds itself in. This subquestion looks at several of these aspects, as well as diving into the technology that runs Ethereum, to check its level of security and privacy.

6.1 Risks of Ethereum in general

New cryptocurrency-like concepts like Ethereum are often vulnerable to internal as well as external factors.[43] These risks and vulnerabilities range from internal company issues to larger legal and fraudulent cases.[44]

A notary can help prevent fraud and identity theft. It is a powerful risk management tool if a notary publically witnesses the signing of contracts.[29] Using Ethereum this risk management is reduced, since all records enter the blockchain and can not be deleted. Contract fraud comes in several forms, like Fraud in the Inducement, where the fraud exists with regards to the entire contract; the person is deceived into signing due to the fraudulent circumstances (for instance, they sign because they though the person was a real estate agent, when in fact they were not), and Fraud in the Factum where the fraud exists as to a certain fact or description within the contract. For instance, if one party signs because they thought they would be purchasing 50 items, when in fact the person intends to sell them 100 items. However, these are mainly due to human error and careless or inaccurate reading of the contract.[45] A more compelling type of fraud with regards to Ethereum is the forgery of signatures. Forgery is considered a crime when a person creates a false document or alters a genuine one with the intent to defraud. Some criminal statutes also require the person making the forgery to benefit from it in some way. For example, someone may change a check worth $100 to $1,000.[46] In this situation, Ethereum could come to great use. By using the Ethereum blockchain, everyone would own a copy of the created contract. Altering the contract would not be possible, since one person would have the modified contract, and the rest of the Ethereum users would have the original copy. Due to the contract not being alterable and since the author is always known it prevents forgery in a great way.

Another possible problem is Ethereum moving to a Proof-of-Stake system.[47, 48] One of the problems with this system is called the "nothing at stake" problem, where block-generators have nothing to lose by voting for multiple blockchain-histories, which prevents the consensus from ever resolving. Anyone can abuse this issue to attempt to double-spend "for free" because there is little cost in working on several chains.[49] To solve this, Ethereum suggested a Slasher protocol that allows users to punish the cheater, who mines on the top of multiple blockchain branches.[50] However, Slasher was never implemented. The Ethereum developers concluded proof-of-stake was non-trivial and instead designed a proof-of-work algorithm called Ethash.[51, 2]

However, since day one of Ethereum it has always been the plan to implement a Proof of Stake algorithm. In order to make sure the transition between a proof

of work and an proof of stake algorithm goes smoothly, a difficulty bomb has been implemented. This difficulty bomb ensures that when the time comes to transition to the proof of stake algorithm, miners will follow, because the difficulty bomb will make it impossible to make any profit off of the proof of work algorithm. [52] However, because the proof of stake algorithm will utilise the amount of Ether owned by the stakeholders in order to validate a block instead of computing power, there will be less incentive for miners to mine on the Ethereum blockchain. However, as there is less computing power required in order to validate transaction, this should be a negligible problem. The Casper proof of stake algorithm is also a betting scheme; miners will use Ether to bet on their block to be part of the blockchain. If this is the case, they will be rewarded Ether, if this is not the case, they will have wasted computing power without a reward.[20]

6.2 Wallet security

To execute transactions, one needs to have Ether. This Ether has to be stored somewhere. While it is not necessary to use a wallet for this, it is often a more convenient solution because of the graphical interface. Because the wallet contains valuable goods, security is an important factor. Not all wallet clients that are available are trustworthy[53], so a choice should be made carefully.

A powerful technology that can be used to secure the funds stored in the wallet is the multiple signature wallet. The multiple signature wallet can be configured to need the authorization of multiple accounts to send a transaction. This functionality is flexible, because of the amount of accounts that own the wallet, the amount of accounts that have to approve the transaction and the limit of Ether, which can be spent daily before the multiple authorisation is needed. These accounts do not necessarily have to be owned by other people as the accounts can also be in one's possession. However, to improve security, the private keys of the account should not be stored on the same device, because in the case of a hack, multiple keys will still be compromised.

Should a multiple signature wallet be successfully created, with at least two accounts on different devices, it is impossible for hackers and rogue employees to compromise the system by hacking only one of these devices.[54] The amount of accounts needed for authorising a transaction can be adjusted for higher levels of security.

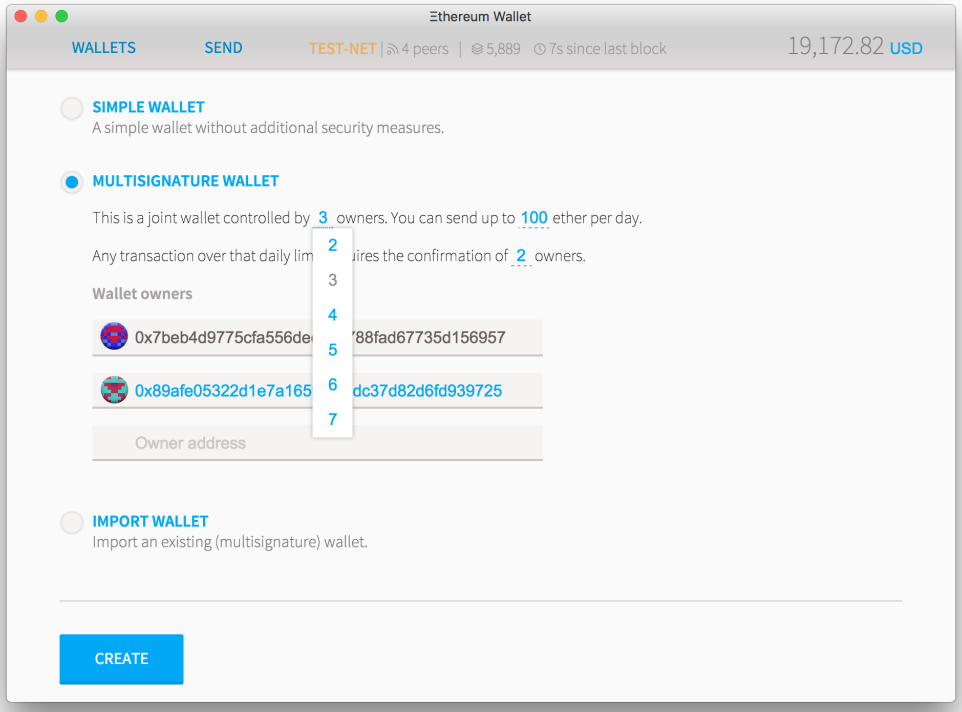

Figure 4: Multi signature wallet

The figure above shows how the Ethereum Mist wallet implements the multiple signature wallet technology. Another option is *Ether.li*, which is a multiple signature web wallet, but lacks the amount of configuration Mist provides. *Ether.li* only works with the standard model of 2-out-of-3 authorization, with the second one always being SMS-authentication. While this does look promising, this does not leave flexibility for companies to implement their own kind of 2-factor authentication technology.[55] This wallet is the best option at the moment, because the Mist wallet implements this technology, has these configuration options and is actively being developed by the Ethereum foundation itself, [56]

6.3 Privacy and Security

When Ethereum is being discussed, besides the scalability, the main issue is the privacy of the blockchain. Privacy in the blockchain is a significant factor in this research, because it is a crucial aspect of contracts. Contracts often contain private information about the people involved, such as finances, which should not be known to the public. Because security is closely matched to privacy, it will be discussed together with privacy.

Ethereum is based on a blockchain technology, which means that everyone owns a copy of all the data stored on the blockchain. This does, however,

implicates that everyone can see all data stored on the blockchain, even someones financial records. All this data can be seen by competitors, governments, family, friends, etc.[57]

Currently, with the Homestead release, a solution for the privacy problem has not yet been implemented. However, with the Serenity release, there will be more possibilities to implement advanced cryptotechnology such as ring signatures. These possibilities will open up because Serenity will introduce a model in which all transactions are valid.[58] To solve the privacy problem of the blockchain, several technologies have already been proposed.[57] These technologies and their viability for privacy for paper contracts will be discussed in the following section.

6.3.1 Ring signatures

One of the most talked about technologies is ring signatures, which looks very promising, as it is already implemented by Monero, which is another blockchain technology.[59] Another reason for why this will be the primary privacy solution being looked at is that there is already a ring signature verification contract created for the second Serenity proof of concept, which enables the use of ring signatures for privacy.[60, 61]

Ring signatures provide privacy by taking a ring of possible signers, including the user sending the transaction. Each of the public keys can be used to compute a mathematical function contained in the ring, but only the private key of the original sender can be used to control the output. There is no way to determine the real sender from this group because the user just has to prove membership to the group.[57] If there is only a certain group of signers which should be able to send certain information, this ring can also, theoretically, be comprised of a group of those signers, instead of all random signers.[62]

However, the fact that each user is anonymous brings one problem. There is no way to check whether or not a user in a ring has already signed with his private key. This is where linkable ring signatures come in. Linked ring signatures can determine when something is signed twice with the same private key. This linkability can prevent users from spending more than once with the same public key.[57]

6.3.2 One time accounts

One time accounts is a technology implemented by Zerocash, another blockchain technology[63]. The way this works is that every sender must use a given address only once for either sending or receiving coins. After receiving coins, a user should immediately make a new address, and pour those coins into that address. Only after this, the money can be spent again. This technology makes sure that there is an everlasting anonymity. However, the creation of new accounts with every transaction gives a substantial overhead, which results in reduced performance.[64] One time accounts is not a technology currently implemented in the Ethereum core, but this is the easiest technology to implement from a user's standpoint.

The usage of the one time account technology can be implemented by users themselves. This can be achieved by creating a new account for each transaction.

However, it requires a lot of time and effort to set up, which is why this is not the ideal solution.

6.3.3 Obfuscation

Obfuscation is the technology of making the process between input and output unintelligible. An example in the blockchain would be that it is recorded that user A spent money and user B received money. If this is obfuscated, there would be no distinguishable relation between the two users, just the fact that one spent money, and one received money. Complete obfuscation is mathematically impossible. However, there is indistinguishable obfuscation[65], which is something that can be used by us to create smart contracts.

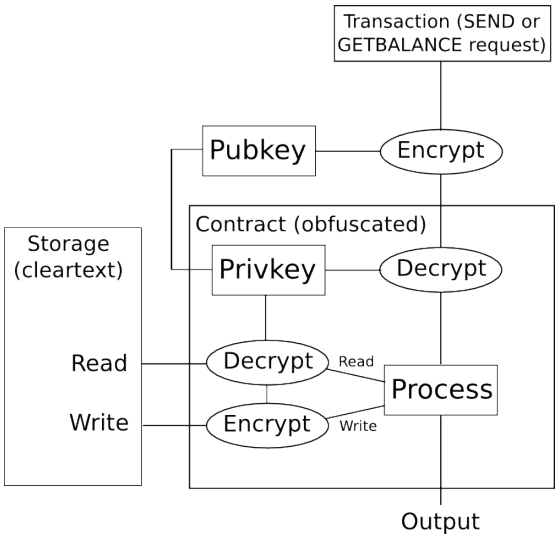

Figure 5: Obfuscating Smart Contract

Indistinguishable obfuscation can be implemented in a smart contract as seen in the figure above. In this case, the obfuscated contract code will check whether or not the user is entitled to read the balance. If they are entitled, the code will return the balance. Otherwise, the code will return an error.[57] However, this implementation is very inefficient, with calculations taking a very long time to complete [66]. On the blockchain, this will result in very expensive contract calls, which is why we can not use this kind of smart contract.

6.4 Blockchain

One of the biggest weaknesses of Bitcoin is the 51% attack. The blockchain contains all the records of the past transactions, which means the information is not stored on any central server. The blockchain is stored in multiple pools 3.1.3. These pools are continuously checked and rechecked. Sometimes a block will be mined that is not part of the conventional blockchain, a so-called orphan block. When the orphan block is mined, it will be validated against the pre-existing blockchain. If the orphan block is not validated, it will be removed.

This means there are multiple blockchains being compared to each other. The bitcoin protocol declares that the valid blockchain is the one that has been worked on the most. This is where things go wrong. If an attacker has 51% of the network's total hashing power, the attacker can create its own blockchain. When the attacker has its own blockchain, the attacker can put more data in their blockchain. If this is faster than the number of times the original blockchain updates, the attacker can create a new blockchain. The attackers blockchain can double-spend coins by removing transactions from its blockchain after spending them. This means the coins are returned to the original user's wallet. When in control of 51% of the network hash rate, certain addresses can be made unspendable by rejecting transactions for those addresses. A mining pool that controls 51% can be devastating towards other mining pools by rejecting their data. That data will become orphan blocks.[67] However, despite widespread concern about the vulnerability of the bitcoin network to large mining pools, there remains no easy solution to the issue. In theory, one might be able to affect the next block or two, launching a double-spending attack by holding on to enough power to confirm the majority of transactions. However, this would take an enormous amount of expense and effort and, should a big double spend be attempted; the data would likely appear on the blockchain for all to see. The attack could destroy the integrity of the system. This will cause the price of the coin to crash and is not something the community wants. Their profits depends on the current state of the bitcoin. If there were an attack, the coin value would crash and everybody loses their money.[68]

6.5 Conclusion

Contract fraud comes in several shapes and sizes, like Fraud in the Inducement and Fraud in the Factum. However, these are mainly due to human error and careless or inaccurate reading of the contract.[45] A more compelling type of fraud with regards to Ethereum is the forgery of signatures. In this situation, Ethereum could come to great use. By using the Ethereum blockchain, everyone would own a copy of the created contract. Altering the contract would not be possible, since one person would have the modified contract, and the rest of the Ethereum users would have the original copy.

Another possible problem is Ethereum moving to a Proof-of-Stake system. Anyone can abuse this issue to attempt to double-spend "for free" because there is little cost in working on several chains.[49] To solve this, Ethereum suggested a Slasher protocol that allows users to punish the cheater, who mines on the top of multiple blockchain branches.[50] However, Slasher was never implemented. The Ethereum developers concluded proof-of-stake was non-trivial and instead designed a proof-of-work algorithm called Ethash.[51, 2] A fundamental flaw of a proof of work algorithm is that the cost of attacking is equal to what is spent running the system. This means that high security can only be achieved at high operating costs. When the Proof of Stake for Ethereum comes, an honest validator is expected to have very low costs, in contrast to an attacker.

As far as safe storage of Ether is concerned, multisig wallets are the way to go. This way, not a single person is responsible, so if a single account gets compromised, it is not a big deal. The Mist wallet for Ethereum should be the one to use for this. *Ether.li* looks promising as well, but it lacks the configuration options the Mist wallet offers.

On the field of privacy and security, a couple of technologies that could implement this have been researched. The ring signature technology looks very promising regarding privacy on the blockchain, but this is not yet implemented in the current Ethereum release. However, with the release of Serenity, this will be a viable option to implement privacy. The second option is one-time accounts. Although this is something that has to be implemented in the blockchain protocol itself to be viable. It could also be implemented manually, but that will cost too much effort and time. The last option is the obfuscation of smart contract code. If this works, this would be a perfect solution. However, it is proven to be very computationally expensive, which is why this is not viable. For now, the ring signature technology looks the most promising, but this is not implemented in the Ethereum blockchain yet. At the moment, privacy is still a very weak point in the Ethereum blockchain.

One of the biggest weaknesses of the blockchain is the 51% attack. This attack can be accomplished by any attacker that gains over 51% of the network's total hashing power. When this happens, the attacker can create his own blockchain and can double-spend coins by removing transaction after spending them. When in control of 51% of the network hash rate, certain addresses can be made unspendable by rejecting transactions for those addresses. However, when this attack happens the market value would crash, and everybody loses their money.

7 Proof of concept

To prove that a smart contract can replace a paper contract, a proof of concept was built. In this chapter, its goal, technology, and the result will be explained. There has been chosen to replace a paper residential lease agreement with a smart contract.

7.1 Goal

The goal of the proof of concept is to support the main research question. The theory is that a smart contract can replace a paper contract, this proof of concept is built to support this theory. In section 5.2.4, a residential lease agreement is explained as an example to be replaced by a smart contract. This example is taken and used in the proof of concept to prove that a smart contract can replace a paper contract.

7.2 Technology

The proof of concept uses Meteor to build a real-time web application. Meteor is a full-stack framework with the ability to create real-time web applications. This framework was chosen because the Lead DApp developer for Ethereum, Fabian Vogelsteller, recommends it.[69]

DApp is an acronym for decentralized application. Ethereum DApps usually interface users with an HTML/Javascript web application using a Javascript API to interact with the blockchain.[70] The proof of concept uses Meteor as explained earlier for the HTML/Javascript web application. As for the Javascript

API to communicate with the blockchain, the proof of concept uses the Web3 Javascript API.[71]

To build the proof of concept the official wiki from Ethereum has been used, about how to create a DApp using Meteor.[72] The information on the wiki suggested a boilerplate that offered a platform to build a DApp using Meteor.[73]

The smart contract is written in Solidity. This has been the standard for smart contracts for a while, and is still actively being improved. The predecessor of Solidity is Serpent. However, development on Serpent has stopped, because it is seen as finished. Besides the fact that Solidity is actively being developed on, it has the advantage of being statically typed and offering many more advanced features such as inheritance, libraries, complex user-defined types and a bytecode optimizer.[74]

7.3 Result

The proof of concept consists of a smart contract and a DApp that allows a landlord and a tenant to make an agreement on a residential property.

The first step in making this possible is to create a smart contract that holds the data of the agreement. In appendix B the full smart contract is shown. This smart contract can contain information that is relevant to a residential lease agreement. The smart contract needs two values when it is deployed; the house and rent for the residential lease agreement. It will make the person that deployed the smart contract on the blockchain the landlord. When the contract is deployed the state of the contract is set to "Created". Only someone else than the landlord can confirm the agreement if the state is "Created". When someone confirms the agreement that person is set as the tenant and the state is set to "Concession". When the state is "Concession" the agreement can not be confirmed again, thus eliminating the possibility of overwriting the current tenant. When the state is "Concession" it allows the tenant to make payments to the contract, the contract will then, in turn, send the payment to the landlord and save the payment information in a payment list in the contract. If the contract needs to be terminated, it is possible for the landlord to do so. When the landlord terminates the contract the state is set to "Terminated" and all Ether left in the contract is sent to the landlord. After the contract is terminated it will continue to exist on the blockchain, but payments can not be made anymore, and the state will continue to be "Terminated".

The second step of making this possible is to create a DApp that uses the smart contract. The DApp allows a person to deploy the contract as a landlord with an inputted house and rent. When the contract is deployed, its information is viewable within the DApp. Another person can confirm the contract as a tenant in the DApp when the person agrees with the shown information from the contract in the DApp. Further implementations for paying the rent and terminating the contract are left out because the main concept is proven with the built smart contract and DApp.

Figure 6: Smart contract information in the DApp

8 Conclusion

The main question "What criteria does Ethereum need to fulfil to replace paper contracts?" has successfully been answered. Ethereum needs to be able to save a paper contract on the blockchain, the privacy of the contract should be sufficient, and the security of the contract needs to be sufficient.

As shown by the proof of concept, it is possible to store paper contracts on the Ethereum blockchain. A user can transfer a paper contract to the blockchain using the proof of concept, as well as being able to read the contract. However, A contract can have a wide variety of clauses. To transform certain contracts to smart contracts written in Solidity code, they need to be translatable. Some clauses can not be translated to smart contracts, making them unautomatable.

The proof of concept created for this project makes it possible to place a paper contract on the blockchain using a smart contract. However, this means that the properties of the contract can be viewed by anyone. Because data such as the tenant, the landlord, the address and the rent is publically viewable, the privacy is a very weak point. With the Serenity release of Ethereum, it will be possible to implement ring signatures. This technology enables one to hide the identity of the tenant. However, at the current state of Ethereum, it is not advised to implement paper contracts on the blockchain, as the privacy is not yet sufficient.

Storing a paper contract on the blockchain ensures that the contract is immutable, thus making the agreement final. However, the biggest risk is a 51% attack. Such an attack would allow the perpetrator to modify the blockchain. With this risk, it would be possible to alter the blockchain, and thus allowing the perpetrator to change the contract or even remove it from the blockchain. Fortunately, the chance of this happening is rare, as the perpetrator has a significant stake in the price of Ether, which would drop massively if the Ethereum community found out that the blockchain had been compromised.

Due to the variety of the contract's clauses and the privacy setback, it is not recommended to place paper contracts on the Ethereum blockchain.

References

[1] mr. B.G.N. (Bart) Gubbels. Contract opstellen.

[2] Gavin Wood. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

[3] David Schuff and Robert St Louis. Centralization vs. decentralization of application software. *Communications of the ACM*, 44(6):88–94, 2001.

[4] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[5] Wei Dai. B-money. *Consulted*, 1:2012, 1998.

[6] Hal Finney. Rpow: Reusable proofs of work, 2004.

[7] Vitalik Buterin et al. Ethereum white paper, 2013.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[9] Bobby Ong, Teik Ming Lee, Guo Li, and David LEE Kuo Chuen. Evaluating the potential of alternative cryptocurrencies. *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*, page 81, 2015.

[10] Olga Kharif. Bitcoin 2.0 Shows Technology Evolving Beyond Use as Money, 2014.

[11] Vitalik Buterin. What is so special about Ethereum, 2016.

[12] Rywalk. An Overlooked Development: Ethereum, IBM ADEPT and the Internet of Things (IoT), 2015.

[13] Dcrypted. Slock.it DAO pre-sale, 2016.

[14] Ian Allison. RWE and Slock.it – Electric cars using Ethereum wallets can recharge by induction at traffic lights, 2016.

[15] Phil Barry. DEVCON1: Ujo Music - Phil Barry, 2015.

[16] Ujo. Rebuilding the music industry on the blockchain, 2015.

[17] Vitalik Buterin. Privacy on the blockchain. *Ethereum Blog*, 2016.

[18] Etherchain. Transactions, 2016.

[19] Aggelos Kiayias and Giorgos Panagiotakos. Speed-Security Tradeoffs in Blockchain Protocols. 2015.

[20] Vlad Zamfir. Introducing Casper "the Friendly Ghost". *Ethereum Blog*, 2015.

[21] Gideon Greenspan. No Title. *Private blockchains*, 2016.

[22] Aaron van Wirdum. The Decentralist Perspective, or Why Bitcoin Might Need Small Blocks. *Bitcoin Magazine*, sep 2015.

[23] Vitalik Buterin. State of Ethereum: August Edition. *Ethereum Blog*, 2014.

[24] Ethereum. Ethereum blockchain, 2015.

[25] Ethereum. Mining, 2015.

[26] Coindesk.

[27] Ethereum. What is Ether?, 2016.

[28] Ethereum. Frontier Guide, 2015.

[29] Tami Kamin-Meyer. What is a Notary and Why do we Notarize Documents?, 2008.

[30] Jefrey Steinberger. Is This Contract Valid?, 2007.

[31] Unknown. Notices, 2016.

[32] TransLegal. Types of contract clause (1).

[33] International Trade. International Law & Contracts.

[34] Fast Moving Targets. Bas Wisselink (NXT Foundation) legt de blockchain uit, 2016.

[35] Oleg Andreev. Joint Escrow, 2015.

[36] Christian Reitwiessner. DEVCON1: Building a DApp: Writing contracts, 2016.

[37] DUO. Diploma mills, 2016.

[38] Groupe Léonard de Vinci. L'ESILV va certifier ses diplômes grâce à la Blockchain et délivrer des enseignements sur ce réseau, 2016.

[39] Antony Lewis. A gentle introduction to immutability of blockchains, 2016.

[40] e Vox. e-Vox: Open e-Democracy Platform.

[41] Nick Abouzeid. Ukraine Government Plans to Trial Ethereum Blockchain-Based Election Platform, 2016.

[42] midasium. Smart Tenancy Contracts.

[43] Joy Marie Virga. International criminals and their virtual currencies: the need for an international effort in regulating virtual currencies and combating cyber crime. *Braz. J. Int'l L.*, 12:512, 2015.

[44] John Fry and Eng-Tuck Cheah. Negative bubbles and shocks in cryptocurrency markets. *International Review of Financial Analysis*, 2016.

[45] Jose Rivera. What is contract fraud?, Dec 2014.

[46] Effect of forgery on a contract.

[47] Andrew Poelstra. On Stake and Consensus, 2015.

[48] Vitalik Buterin. On Stake, 2014.

[49] Community. Problems, 2015.

[50] Vitalik Buterin. Slasher: A Punitive Proof-of-Stake Algorithm, 2014.

[51] Vitalik Buterin. Slasher Ghost, and Other Developments in Proof of Stake, 2014.

[52] Stephan Tual. Ethereum Protocol Update 1. 2015.

[53] EthereumCanada. WARNING to 1000+ users of "Ethereum Wallet" by Freewallet on Android. They hold your keys, website and support emails very sketchy. Many signs of scams we've all seen before., 2016.

[54] Vitalik Buterin. Blockchain and Ethereum Security on the Higher Level, 2016.

[55] Jacob Donnelly. BitGo Engineers Launch Ethereum Wallet Side Project. 2016.

[56] Alex van de Sande. Mist releases, 2016.

[57] Vitalik Buterin. Privacy on the Blockchain. *Ethereum Blog*, 2016.

[58] Vitalik Buterin. Understanding Serenity, Part I: Abstraction. *Ethereum Blog*, 2015.

[59] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015.

[60] Ring signature code, 2016.

[61] Vitalik Buterin. Serenity PoC2. *Ethereum Blog*, 2016.

[62] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. 2005.

[63] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.

[64] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. 2014.

[65] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits, 2013.

[66] Sebastian Banescu, Martin Ochoa, Nils Kunze, and Alexander Pretschner. Idea: Benchmarking Indistinguishability Obfuscation – A candidate implementation, 2015.

[67] Joel Hruska. extremetech.

[68] Daniel Cawrey. Coindesk.

[69] Fabian Vogelsteller. DEVCON1: Building a DApp: What are DApps and why Meteor, 2016.

[70] Ethdocs Dapps.

[71] Web3 JavaScript Dapp API.

[72] Dapp using Meteor.

[73] meteor-dapp-boilerplate.

[74] Solidity. Frequently asked questions. 2016.

[75] Ethereum. Create a digital greeter, 2015.

A Contract creation and deployment

Ethereum created a tutorial on how to create contracts; they showed the following example about a contract that can greet you:[75]

```
contract mortal {
    /* Define variable owner of the type address*/
    address owner;

    /* this function is executed at initialization and sets the owner of
        the contract */
    function mortal() { owner = msg.sender; }

    /* Function to recover the funds on the contract */
    function kill() { if (msg.sender == owner) suicide(owner); }
}

contract greeter is mortal {
    /* define variable greeting of the type string */
    string greeting;

    /* this runs when the contract is executed */
    function greeter(string _greeting) public {
        greeting = _greeting;
    }

    /* main function */
    function greet() constant returns (string) {
        return greeting;
    }
}
```

This is a simple contract with the classic "hello world". As seen in the example code, there are two contracts: mortal and greeter. This is because the contract language Solidity has inheritance, which means that we can reuse contract code so that we do not have to rewrite it every time. The mortal contract has code that allows a contract to kill itself by its owner. This will remove the contract from the blockchain and recovers locked funds in the contract back to its owner. In its default state a contract is immortal and does not have an owner. Therefore, it is needed if you want to be able to remove a contract from the blockchain. Before a contract can be deployed it needs to be compiled. The code must compile through, for example, the Geth console (command line interface for running a full Ethereum node). You need to reformat the contract to compile it by removing the line-breaks so that it fits into a string variable. When this is done you can run the following commands in the Geth console and the contract will compile:[75]

```
var greeterSource = 'contract mortal { address owner; function mortal()
    { owner = msg.sender; } function kill() { if (msg.sender == owner)
    suicide(owner); } } contract greeter is mortal { string greeting;
    function greeter(string _greeting) public { greeting = _greeting; }
    function greet() constant returns (string) { return greeting; } }'
```

```
var greeterCompiled = web3.eth.compile.solidity(greeterSource)
```

After running these commands, the contract is ready to be deployed. For the current example the greeting that returns when the contract executes needs to be defined. By running the following commands in the Geth console, the contract will deploy on the blockchain:[75]

```
var _greeting = "Hello World!"
var greeterContract =
    web3.eth.contract(greeterCompiled.greeter.info.abiDefinition);

var greeter = greeterContract.new(_greeting,{from:web3.eth.accounts[0],
    data: greeterCompiled.greeter.code, gas: 300000}, function(e,
    contract){
   if(!e) {

     if(!contract.address) {
       console.log("Contract transaction send: TransactionHash: " +
           contract.transactionHash + " waiting to be mined...");

     } else {
       console.log("Contract mined! Address: " + contract.address);
       console.log(contract);
     }

   }
})
```

After all these commands the contract is deployed on the blockchain and ready to be used. As programmed in the contract example, it has a function called "greet" that returns a greeting. The greeting was defined in the last command block as "Hello World!". To execute the contract and receive a greeting, the following command must be used:[75]

```
greeter.greet();
```

Now with that command ran, the console will return "Hello World!" as defined in the constructor from the contract in the deployment. Calling the function greet does not cost gas as the call does not change anything on the blockchain.

B Rental contract

```
contract RentalContract {

    /* This declares a new complex type which will hold the paid rents*/
    struct PaidRent {
        uint id; /* The paid rent id*/
        uint value; /* The amount of rent that is paid*/
    }

    PaidRent[] public paidrents;

    uint public createdTimestamp;
    uint public rent;
    /* Combination of zip code and house number*/
    string public house;
    address public landlord;
    address public tenant;
    enum State { Created, Concession, Terminated }
    State public state;

    function RentalContract(uint _rent, string _house) {
        rent = _rent;
        house = _house;
        landlord = msg.sender;
        createdTimestamp = block.timestamp;
    }

    modifier require(bool _condition) {
        if (!_condition) throw;
        _
    }

    modifier onlyLandlord() {
        if (msg.sender != landlord) throw;
        _
    }

    modifier onlyTenant() {
        if (msg.sender != tenant) throw;
        _
    }

    modifier inState(State _state) {
        if (state != _state) throw;
        _
    }

    function getPaidRents() internal returns(PaidRent[]) {
        return paidrents;
    }

    function getHouse() constant returns(string) {
```

```
    return house;
}

function getLandlord() constant returns(address) {
    return landlord;
}

function getTenant() constant returns(address) {
    return tenant;
}

function getRent() constant returns(uint) {
    return rent;
}

function getContractCreated() constant returns(uint) {
    return createdTimestamp;
}

function getContractAddress() constant returns (address) {
  return this;
}

function getState() returns (State) {
  return state;
}

event agreementConfirmed();
event paidRent();
event contractTerminated();

/* Confirm the lease agreement as tenant*/
function confirmAgreement()
    inState(State.Created)
    require(msg.sender != landlord)
{
    agreementConfirmed();
    tenant = msg.sender;
    state = State.Concession;
}

function payRent()
    onlyTenant
    inState(State.Concession)
    require(msg.value == rent)
{
    paidRent();
    landlord.send(msg.value);
    paidrents.push(PaidRent({
            id: paidrents.length + 1,
            value: msg.value
        }));
}
```

```
    /* Terminate the contract so the tenant can't pay rent anymore,
    and the contract is terminated */
    function terminateContract()
        onlyLandlord
    {
        contractTerminated();
        landlord.send(this.balance); /* If there is any value on the
            contract send it to the landlord*/
        state = State.Terminated;
    }

}
```