BLOCKCHAIN – THE GATEWAY TO TRUST-FREE CRYPTOGRAPHIC TRANSACTIONS

Complete Research

Beck, Roman, IT University of Copenhagen, DK, beck@itu.dk[1]

Stenum Czepluch, Jacob, IT University of Copenhagen, DK, jstc@itu.dk

Lollike, Nikolaj, IT University of Copenhagen, DK, nlol@itu.dk

Malone, Simon, IT University of Copenhagen, DK, soma@itu.dk

Abstract

Recently, the Bitcoin-underlying blockchain technology gained prominence as a solution that offers the realization of distributed trust-free systems, where economic transactions are guaranteed by the underlying blockchain. We are still at an early stage and thus require a deeper understanding of how the blockchain potentials can be realized, and what are the opportunities and challenges in so doing. Following a design science approach, we developed a proof of concept prototype that has the potential to replace a trust-based coffee shop payment solution that is based on an analogue, pre-paid punch card solution. The demonstrator provides a starting point to evaluate the strengths and weaknesses of the blockchain technology when replacing a trust-based by a trust-free transaction system. We conclude that the secure and trust-free blockchain-based transaction has the potential to change many existing trust-based transaction systems, but that scalability issues, costs, and volatility in the transaction currency are hindrances.

Keywords: Blockchain, Design Science Research, Transactions, Cryptographic Economic System

1 Introduction

Even though the blockchain gained prominence with the emergence of Bitcoin in 2009 and thus exists now for about six years, we are still at the beginning to fully understand it's potential. Innovative solutions that go beyond crypto-currencies such as Bitcoin may have the potential for fundamentally changing society and we might witness right now the dawn of cryptographically secured trust-free transactions economy. The Economist just coined the blockchain "the trust machine", thereby indicating that the blockchain takes care of trust issues, thereby freeing us from the necessity of implementing mechanisms to signal or convey trust (Economist, 2015). In other words, the created system is running without trust concerns, making a transaction "trust free", once it is settled as an agreement in the blockchain. While the Bitcoin blockchain was just the beginning, with the increasing availability of generic, self-programmable blockchains, as offered by foundations such as Ethereum, blockchains are now used in other areas beyond crypto-currencies as well. This paves the way of utilizing the features of the blockchain such as its trust-free, transparent, and highly secure nature in other application areas. For example, IBM and Samsung announced to experiment with the Ethereum-based blockchain to power Internet of Things (IoT) solutions.

[1] All authors contributed equally to this paper and are mentioned in lexicographic order. We would like to thank the Ethereum Foundation for its support.

In this paper, we will investigate the potential of the blockchain technology by developing, implementing and evaluating the proof of concept prototype of a blockchain-based, digitized punch card solution. In so doing, we illustrate how a trust-based centralized system can be replaced by distributed and trust-free transaction systems. Thus, our research is concerned about how decentralization through blockchain technology can be applied in economic systems and what are the strengths and weaknesses of the technology in doing so.

2 Cryptographic economic systems and the "Internet of Things"

A cryptographic economic system can be understood as a system that organizes transactions completely reliable, without any human interaction, following intractable rules set in the computer protocol. The operating economic system can be compared with an unstaffed, automatically navigating vessel or a driverless steering car, bringing safely passengers from A to B, completely controlled by a cryptographic protocol that is minimizing any malicious and accidental exceptions because no humans are involved.

Cryptographic economic systems can also support distributed autonomous organizations (DAO) such as IoT to operate trust-free by completing transactions on basis of self-enforcing rules. This enables to strip off any transaction costs with regards to the lack of trust between economic agents within the transaction phase, which formerly would have been only interested to interact with each other if an entrusted third party steps in. In other words, a DAO might enable to curb transaction costs involved to setup, maintain, regulate and supervise the operation of an entrusted third party organization.

Thus, not surprisingly, the blockchain became very popular in finance where transparency, trust, and security in transactions are vital (Economist, 2015) and where we have witnessed the development of different cryptocurrencies such as Bitcoin, Litecoin, and Dogecoin, among others, in the past. Cryptocurrencies do not require any intermediary such as a central bank to ensure trust and security in transactions (Buterin, 2013, Economist, 2015, Nakamoto, 2008) and are also more cost-efficient in microtransactions compared to fiat money (Buterin, 2014b). Radiating trustworthiness through third-parties is replaced by understanding the blockchain technology and seeing what the status of the transaction is. In other words, instead of trusting that a transaction will be conducted as agreed upon, now one can see the status of the transaction and knows what is going on. However, the blockchain does not only support financial transactions, but can support all kinds of agreements or smart contracts: Digital assets such as shares, contracts, and stock options have been traded as smart contracts on the blockchain as well. Thus, all kinds of economic systems, or more specifically, trading of property rights, benefit from such a trust-free, secure, and transparent transaction system. Recently, the NASDAQ started to use the blockchain to automate transaction processes that have been handled by lawyers so far (Price, 2015).

For such a broad range of applications beyond Bitcoin, a blockchain is needed that is designed to support transactions on a generic level through the use of smart contracts. For our research, we decided to use Ethereum's blockchain that supports a Turing-complete programming language to write smart contracts into the blockchain. It is thus a completely programmable blockchain that distributes logic that would normally be executed on a centralized server. For example, with the introduction of smart devices in the late 2000s, computers have found themselves being implemented in virtually everything. The next step in development is that these smart objects can communicate with each other over the Internet and essentially maintain themselves. To organize such an IoT (Panikkar, 2015, Pureswaran and Brody, 2015), companies such as IBM and Samsung build their IoT version based on the Ethereum blockchain (Higgins, 2015, Panikkar, 2015) so that devices can interact with each other through smart contracts. It is expected that these transactions between smart devices will develop an "Economy of Things" where all these smart devices will be "a point of transaction and economic value creation for owners and users" (Panikkar, 2015).

3 Applied design science approach

We deemed a design science research (DSR) approach as an appropriate research lens for our blockchain proof of concept (March and Smith, 1995) and the design of a prototype project (Simon, 1969, Walls et al., 1992). In this context, our prototype depicts an instantiation of a blockchain-based IT artefact. Considering that DSR encompasses the creation of something that has not existed previously and that it serves a meaningful human purpose (March and Smith, 1995) to solve a problem, its historical origins tend to emerge from engineering (Au, 2001, March and Smith, 1995), though it also relates to many other academic disciplines. In engineering, architecture, or IS, the science of the artificial is of interest for both research and practice (Baskerville, 2008). When a rigorous DSR approach meets the requirements for generating design theories (e.g., (Gregor and Jones, 2007)), it leads to more generic and richer contributions, though theorization is an additional element of IS design theories. In principle, a problem solution and contributions to the knowledge base derive from either existing kernel theories and IT artefacts or the development of new IT artefacts (Holmström et al., 2009). In this research, we are applying a DSR approach that enables us to generate theoretical insights (Beck et al., 2013) rather than starting with a kernel theory. In our case, we started with the problem of replacing a trust-based manual system by a trust-free, blockchain-based system to theorize about implications and challenges for established theories such as transaction cost theory. Transaction costs mainly arise from mitigating uncertainty in an economic exchange. Such costs can be interpreted as the economic equivalent of friction and lack of trust. If blockchain technology is enabling friction-free, trust-free transactions, then fundamental economic theories need to be revisited. However, there has not been published a lot about blockchain and its applicability which is the reason why we decided to emphasis in this paper on development of the IT artefact before we theorize about it as described by (Beck et al., 2013).

Typically, the DSR process begins with a search for a problem with practical relevance (von Alan et al., 2004). In our case, we started with a technological solution that has the capability to solve elementary economic exchange problems, while we still do not fully understand the potential of blockchain technology. The DSR outcome, however, is always embedded in some place, time and community and must be theorized about to meet innovative and progressive demands (Orlikowski and Iacono, 2001). Therefore, we decided to study a rather simplistic problem system that we turned into a blockchain-supported solution.

In general, the DSR approach comprises two basic processes, building and evaluating (March and Smith, 1995). The building process refers to the sequence of activities required to produce ‘something new’, that is, the blockchain instantiation, to a problem. The evaluation process instead involves evaluation of the created IT artefact to provide feedback and generate new knowledge about the problem at hand. The newly generated insights improve the quality of both the IT artefact and the design process (von Alan et al., 2004). Hence, generalizable outcomes for more generic design problems can be drawn from the build and evaluate process (Gregor and Jones, 2007). However, building and evaluating the IT artefact occurs partly in parallel, with multiple iterations (Beck et al., 2013). The same happened in our DSR project, where the research process has been iteratively going forth and back as well. After we had acquired enough knowledge about the Ethereum-based blockchain, we identified a managerial problem that allowed us to study the strengths and weaknesses of the technology. In our case, we have chosen a trust-based punch card solution that is been used in our university’s students cafeteria for self-service transactions. In our design science proof of concept approach, we can illustrate now how that system can be replaced by a trust-free blockchain based transaction system. Subsequently, we theorize and propose how such trust-free transaction system in general should look like, based upon our research as well as literature in the field.

4 The blockchain based trust-free economic system

The proof of concept in this DSR approach builds upon the blockchain developed by Ethereum in late 2013 (Buterin, 2013). While the platform was released in the second half of 2015, we were granted access to a pre-market environment for our research back in 2014 already, as it still was in a development stage. Thus, we are among the first to be able to test and develop a smart contract-based prototype as a 'new-to-the-world' information systems artefact (von Alan et al., 2004).

The Ethereum blockchain allows user-created digital smart contracts to be executed by implementing a generically programmable code. It is not solely a network for exchanging cryptocurrencies but a network where transactions can be used to execute an unlimited number of self-developed smart contracts. This blockchain serves as the back-end of decentralized applications, which can be as different as voting systems, domain name registries, financial exchanges, crowdfunding platforms, company governance, or intellectual property such as music songs. The Ethereum blockchain solution allows everyone to write smart contracts resulting in decentralized applications where it is possible to build your own arbitrary rules for ownership, transaction formats, and state transition functions (Buterin, 2013). Thus, it is relatively easy to create a smart contract and subsequently adding it to the chain of blocks. In so doing, one can create an environment to potentially solve some issues related to lack of trust and reputation or incomplete information about the counterparty one is trading with, which traditionally required a central trusted party such as an insurance company, a central bank, or the government. In contrast, the blockchain is a "trust-free", distributed solution that can be understood as techno-social system, where the technical part assures the transactions of the social part, which cannot be altered as long as the transaction takes place under the conditions of the agreed upon smart contract. If that is changed or compromised, no transaction will take place, which makes the system inherently secure.

The blockchain is essentially just what the name says: a chain of blocks. A block contains the data of all transactions within a period of time, and a reference to the block before it. The cryptography that goes into creating a block differs depending on which blockchain protocol is used, but essentially one can traverse through the entire blockchain and find every single transaction ever made, all the way back to the first one, so-called genesis block. Hashing algorithms are used to make sure that all blocks are well formed and not tampered with, and thus the blockchain keeps itself secure and virtually unbreakable. The blockchain is not provided from a single server, but is run on a widespread network of computers as a distributed ledger. All network participants hold all data in the blockchain, and all work together on expanding it. These computers are often called miners. Depending on the blockchain protocol, these will compete to form new blocks that are then added to the blockchain when selected through consensus schemes.

The combination of these features makes the blockchain desirable: It is secure through the immutability it gains from the hashing algorithms, when applied on the decentralized network, and transparent because anyone can look through all blocks. The combination of security and transparency is what makes the blockchain a trust-free technology. In the past, blockchains have been compared to bank ledgers (Blockchain, 2015, Graham, 2013). However, this definition somewhat limits the blockchain artificially, as the general idea behind it can be used in a much broader way. Transactions are not limited to monetary transferals, but can also be used to transfer any form of data. How this data is then used again depends on what blockchain protocol it is sent on and to which smart contract it has to adhere to.

A smart contract is a piece of code that the Ethereum Virtual Machine (EVM) is able to execute on the blockchain. Once this piece of code has been added to the blockchain, the smart contract itself cannot be altered, only the storage of the smart contract can. This means that there now exists some piece of code that acts as a contract and that is available for anyone to use. The execution of these smart contracts are made possible by the Turing-complete programming languages that are compiled into EVM bytecode. These languages are, as of July 2015, Solidity (Java like), Serpent (Python like), Mutan (C like), and LLL (Lisp like). A smart contract has an address, just like an external account (essentially a user), but instead of acting like a wallet, it will execute code based on the data it receives. Smart contracts can call other smart contracts in almost the same way through messages. In order to avoid malicious and infinitely looping code or distributed denial of service attacks to be executed on the blockchain, each execution and creation of a smart contract is powered by gas or ether (yet, the name Ethereum). Powered means that the amount of gas needed to run the contract is determined by the total amount of computations and storage entries of the byte code that the EVM compiles the smart contract into. Even though it is estimated that the total amount of computing power available directly on the blockchain is equivalent to that of a 1999 smartphone (Buterin, 2014a), smart contracts are extremely powerful in the way they function as the building blocks of worldwide decentralized trust-free transaction systems. Fuelling is understood as a way to pay for the execution of EVM bytecode and storage of data on the blockchain. How much a specific computation will cost is defined by the complexity of the computation. The most basic computations such as addition, subtraction, and multiplication is intended to cost 1 gas. Contracts and transactions also have a start price that is fixed in order to pay the miner for the provided computational power. And finally, there are messages, which are essentially transactions between two smart contracts. They carry almost the same fields as a regular transaction, except there is no gas price. The gas used by the all computations done by all smart contracts activated by a transaction must be covered in the gas field of said transaction (Buterin, 2013).

There are some differences in the way the Ethereum blockchain and the Bitcoin blockchain work in terms of mining. Each block on the Ethereum blockchain contains the entire state of the Ethereum system, stored in a “Patricia tree”, which is an evolved “Merkle tree” as known from Bitcoin. It only stores the data of new transactions, whilst unchanged data is stored as pointers to the original block location (Buterin, 2013). Also the way a miner is selected to create a block on the blockchain is different to prevent some of the centralization tendencies that have been found in Bitcoin’s blockchain. Bitcoin requires the miner to do a specific hash function on some strings, which means that it is possible to use application-specific integrated circuits, which is already being done. This makes it almost impossible for a user to have a chance to mine. The block validation is now done in mining pools relying on this technology. This is even worse, since at the time of writing, the three biggest mining pools held over 50% of the computing power (Hashrate, 2015). Crucial for commercial applications is also the time required to settle a transaction using the blockchain. The block time is the amount of time it takes for the network to assign a new block to the blockchain, thus confirming pending transactions. Ethereum has a block time of 12 seconds, but there is no guarantee of the amount of time it actually takes for the network to assign a new block. It is just a value the network aims to hit, and it does this by adjusting the so called block difficulty (Buterin, 2014d). The block difficulty is regulated each time a new block is found. If a block is found “too quickly”, that being less than 12 seconds, the difficulty of finding the next block goes up. If it takes too long time to find a block, the block time consequently goes down. The difficulty is defined by how many hashes on average it takes to create a new block. Thus, if the difficulty is high, the harder it is to find a hash within the certain goal that satisfies the proof of work algorithm. The reason Ethereum has a shorter block time of 12 seconds compared to Bitcoin’s 10 minutes is that contrary to the monetary transactions of Bitcoin, the Ethereum blockchain is supposed to be used in a wide variety of applications where transactions need to be more responsive (Buterin, 2014d).

5 Developing a blockchain based proof of concept prototype

5.1 The coffee self-service case

To test the potential of a cryptographic trust-free transaction system based on blockchain, we developed a proof of concept solution for an existing trust-based self-service solution at the coffee shop place operated by students at our University. Essentially, we wanted to implement a digitized version of the currently used punch card approach for coffee self-dispensing, using now smart contracts. The currently used punch card system, where each coffee drinker has to clip off a piece of a pre-purchased, 10 clips comprising punch card him- or herself when having a cup of coffee, can be described as a trust-based system with inexplicit transactions rules. Essentially, coffee could be stolen or people could just forget to clip off a piece of their punch card. While replacing this system by a blockchain based approach might seem a bit over-engineered, the example comprises all the essential benefits and challenges of implementing such an IT solution for other commercial transactions which would benefit similarly from trust-free transaction processes through enforcing the transaction rules by smart contracts within the blockchain.

At this point, the way the coffee buying works is that one buys a card that is worth 10 cups of coffee. Each time one wants a cup, he or she has to deduct one clip using a scissor. Then you fill up your cup from a coffee pot. The punch card represents an artefact that has been given a certain value in the real world. The punch card in itself is a worthless piece of cardboard. It only has a value when used in a transaction in a specific place. Thus, the idea of the punch card is that you can help yourself without the need for a service person. It is up to the user of the punch card to deduct the correct amount of clips when buying one or more cups of coffee. This way, you can get coffee at all times through self-service.

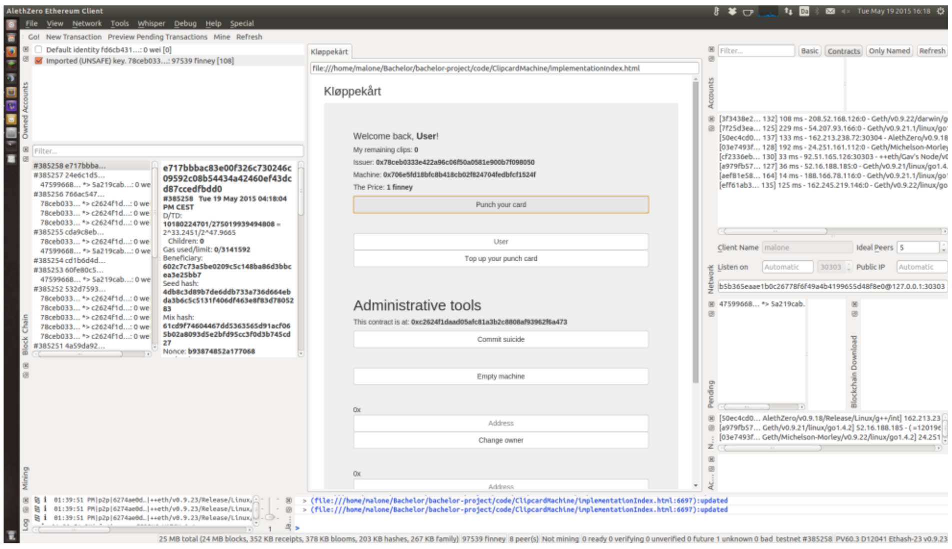
There are, however some problems using this method, since it only works through trusting the users. And to trust the users, they have to be well-informed about how the system works, e.g., which kind of beverages a punch card gains access to and how long the punch card is valid and so on. They have to acknowledge the rules of the “real life contract”. In the current system, there is no way of actually enforcing that the users deduct the correct amount of clips when purchasing coffee – or even pay at all, unless there is somebody enforcing the rule through physical presence, which, however, contradicts the idea of having a self-service system. The reason users deduct the wrong amount of clips can both be malicious behaviour or misunderstanding of how the system works. Using a blockchain-based smart contract solution could prevent these problems.

5.2 Implementing a decentralized trust free transaction system

In order to obtain the desired functionality, a smart contract needs to be implemented to dictate the rules of the system. Based on the rules and requirements for a digital punch card based system to drink coffee, all functionalities are needed to issue punch cards for users to purchase. To distribute this smart property, a function called *buyClipcard* is implemented, which allows for a user to purchase a punch card at a certain price. The price can be adjusted by the issuer of the smart contract to whatever amount through a function called *setPrice*. When purchasing a punch card, the smart contract will make sure that one has a sufficient balance of ether before being able to commit the transaction. To keep track of users and their balances, a map of user addresses on the network and their punch card property, called *clipCards*, is implemented. This way, it is possible to check if people own a punch card or has sufficient clips. Once those functionalities had been created, it was necessary to formulate the rules of buying a cup of coffee, using the punch card token. This part of the contract is basically a “vending machine” in the sense that a product has a fixed price. A function that deducts one clip from your punch card is therefore added as *useClip*. Next, in order for this to be fully automated, a digital lock on the coffee dispensers is needed. In our proof of concept approach, we did not implement a hardware solution since it is the aim of this DSR approach to investigate the blockchain mechanism for creating cryptographic, trust-free transaction systems. However, if a system like the suggested one should be implemented, a commercial coffee vending machine could be connected to the network. A smart contract could then be implemented to activate the coffee dispense, as this smart contract would be invoked by the system. Essentially, this would be an IoT application, where a fully-fledged automated system could be implemented with two smart contracts in conjunction with each other. In order to issue these smart contracts onto the network you make a transaction with the smart contract logic while the price of fuelling such a contract computation is determined by how many operations it includes. If enough ether to fuel the smart contract is available, the transaction will be submitted to the blockchain and be accessible through an address.

In order to access the client of our decentralized application, a browser is needed to support Ethereum based applications. Currently, there are two clients which are called AlethZero and Mist. We used for our research AlethZero as illustrated in Figure 1 to showcase our application. Once connected to the

Ethereum network through a supported browser, the local blockchain will be synchronized. Upon data synchronization, one can navigate to the website of the digitalized punch card as illustrated in Figure 2.



```
Default identity fd6cb431...: 0 wei [0]
Imported (UNSAFE) key. 78ceb033...: 97539 finney [108]

#385258 e717bbba...
#385257 24e6c1d5...
   47599668... *> 5a219cab...: 0 we
#385256 766ac547...
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
#385255 cda9c8eb...
   78ceb033... *> c2624f1d...: 0 we
   47599668... *> 5a219cab...: 0 we
#385254 cd1b6d4d...
#385253 60fe80c5...
   47599668... *> 5a219cab...: 0 we
#385252 532d7593...
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
   78ceb033... *> c2624f1d...: 0 we
#385251 4a59da92...

e717bbbac83e00f326c730246c09592c08b54434a42460ef43dcd87ccedfbdd0
#385258 Tue 19 May 2015 04:18:04 PM CEST
D/TD: 10180224701/275019939494808 = 2^33.2451/2^47.9665
  Children: 0
Gas used/limit: 0/3141592
Beneficiary: 602c7c73a5be0209c5c148ba86d3bbcea3e25bb7
Seed hash: 4db8c3d89b7de6ddb733a736d664ebda3b6c5c5131f406df463e8f83d7805283
Mix hash: 61cd9f74604467dd5363565d91acf065b02a8093d5e2bfd95cc3f0d3b745cd27
Nonce: b93874852a177068
```

file:///home/malone/Bachelor/bachelor-project/code/ClipcardMachine/implementationIndex.html

Kløppekårt

Welcome back, **User**!
My remaining clips: **0**
Issuer: **0x78ceb0333e422a96c06f50a0581e900b7f098050**
Machine: **0x706e5fd18bfc8b418cb02f824704fedbfcf1524f**
The Price: **1 finney**

Administrative tools
This contract is at: **0xc2624f1daad05afc81a3b2c8808af93962f6a473**
0x
0x

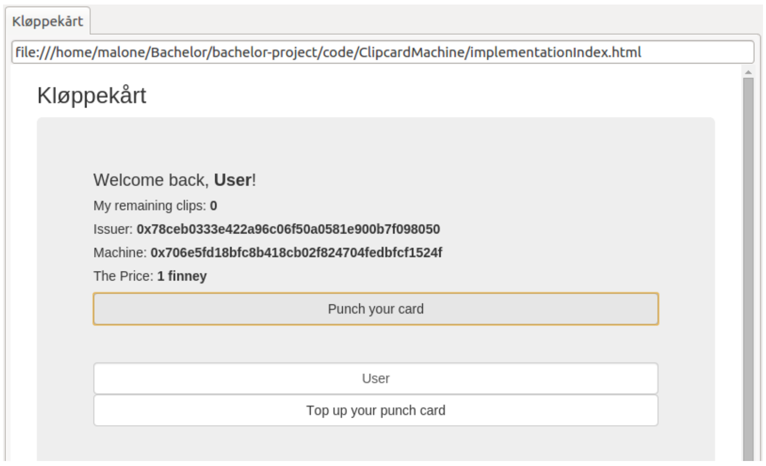
```
[3f3438e2... 132] 108 ms - 208.52.168.126:0 - Geth/v0.9.22/darwin/g
[7f25d3ea... 125] 229 ms - 54.207.93.166:0 - Geth/v0.9.21.1/linux/go
[50ec4cd0... 137] 133 ms - 162.213.238.72:30304 - AlethZero/v0.9.18
[03e7493f... 128] 192 ms - 24.251.161.112:0 - Geth/Michelson-Morley
[cf2336eb... 130] 33 ms - 92.51.165.126:30303 - ++eth/Gav's Node/v0
[a979fb57... 127] 36 ms - 52.16.188.185:0 - Geth/v0.9.21/linux/go1.4.
[aef81e58... 164] 14 ms - 188.166.78.116:0 - Geth/v0.9.21.1/linux/go
[eff61ab3... 135] 125 ms - 162.245.219.146:0 - Geth/v0.9.22/linux/go
```

Client Name: malone Ideal Peers: 5
Listen on: Automatic 30303 Public IP: Automatic
b5b365eaae1b0c26778f6f49a4b4199655d48f8e0@127.0.0.1:30303

```
47599668... *> 5a219cab.

[50ec4cd0... AlethZero/v0.9.18/Release/Linux/g++/int] 162.213.23
[a979fb57... Geth/v0.9.21/linux/go1.4.2] 52.16.188.185 - ( =12019€
[03e7493f... Geth/Michelson-Morley/v0.9.22/linux/go1.4.2] 24.251

01:39:51 PM|p2p|6274ae0d..|++eth/v0.9.23/Release/Linux,
01:39:51 PM|p2p|6274ae0d..|++eth/v0.9.23/Release/Linux,
01:39:51 PM|p2p|6274ae0d..|++eth/v0.9.23/Release/Linux,

> (file:///home/malone/Bachelor/bachelor-project/code/ClipcardMachine/implementationIndex.html:6697):updated
> (file:///home/malone/Bachelor/bachelor-project/code/ClipcardMachine/implementationIndex.html:6697):updated
>

25 MB total (24 MB blocks, 352 KB receipts, 378 KB blooms, 203 KB hashes, 267 KB family) 97539 finney 8 peer(s) Not mining 0 ready 0 verifying 0 unverified 0 future 1 unknown 0 bad testnet #385258 PV60.3 D12041 Ethash-23 v0.9.23
```

Figure 1. AlethZero browser view

Now that the implementation of the backend functionality has been covered, it is desirable to use the smart contracts in an easier way than through command line transactions. This is possible by using the Ethereum JavaScript API to build an HTML client. The role of the client is basically to make the smart contract easier to use, displaying stored information and making functions accessible through a graphical user interface (GUI) that interacts with the contract itself. Thus, instead of making specific transactions with data to the smart contract, input fields and buttons have been implemented to take care of the tasks, as illustrated in Figure 2. This is done by using a JavaScript API that uses web3.js (Ethereum, 2014). The web3 object makes it possible to communicate with the network because it possesses the users private address information. The JavaScript API offers a lot of functionality that can be used to communicate with a smart contract and the Ethereum blockchain. In our punch card proof of concept, we used the function of getting stored data out of the contract to display it in our client, as well as calling functions to manipulate this data.



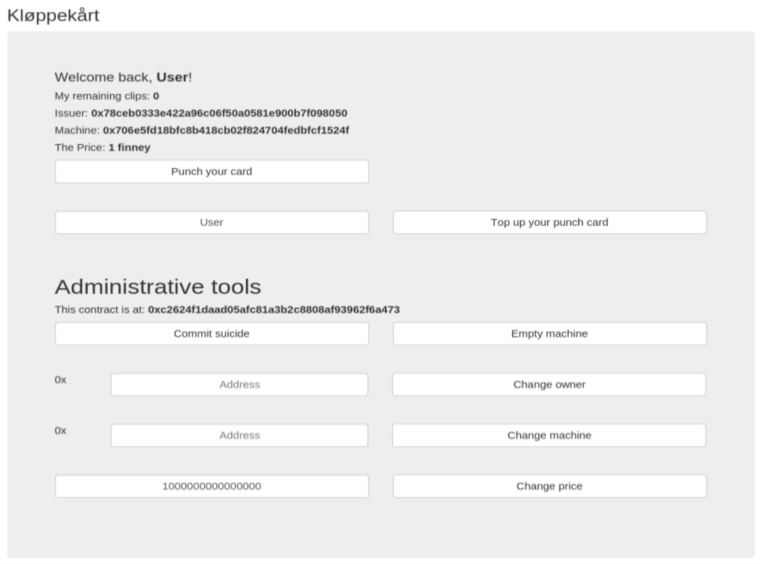
file:///home/malone/Bachelor/bachelor-project/code/ClipcardMachine/implementationIndex.html

Kløppekårt

Welcome back, **User**!
My remaining clips: **0**
Issuer: **0x78ceb0333e422a96c06f50a0581e900b7f098050**
Machine: **0x706e5fd18bfc8b418cb02f824704fedbfcf1524f**
The Price: **1 finney**

Figure 2. The graphical user interface of the HTML/Java client

All the data displayed in bold in Figure 2 is information taken directly from the blockchain, as it is contained in the storage of the smart contract of the implementation. Since the client is connected to the Ethereum network through authenticating with a private key, there is no need for an additional logging in, since the network already recognizes each user individually.

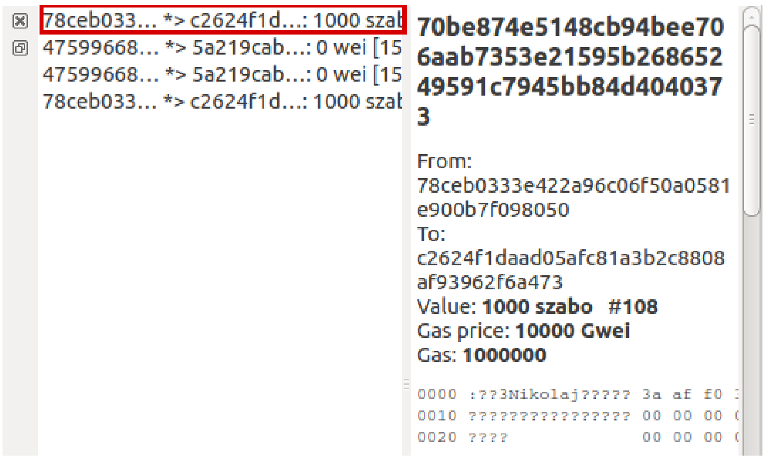

Kløppekårt

Welcome back, **User**!
My remaining clips: **0**
Issuer: **0x78ceb0333e422a96c06f50a0581e900b7f098050**
Machine: **0x706e5fd18bfc8b418cb02f824704fedbfcf1524f**
The Price: **1 finney**

Administrative tools
This contract is at: **0xc2624f1daad05afc81a3b2c8808af93962f6a473**

Figure 3. *The administration view of the client*

In the depicted case, the client does not know the user yet since that is the first time a person uses the system. Therefore, one is identified as “User”, and has no balance. In addition, the client displays the current price of a punch card, the network address of the issuer of the contract, and the network address of the smart contract that controls the lock on the coffee dispensers. All these values can be updated through an admin panel as seen in Figure 3 that is accessible to users with admin rights. Consequently, there is no admin login either since the data of user rights too is stored on the blockchain.


```
78ceb033... *> c2624f1d...: 1000 szab
47599668... *> 5a219cab...: 0 wei [15
47599668... *> 5a219cab...: 0 wei [15
78ceb033... *> c2624f1d...: 1000 szab
```

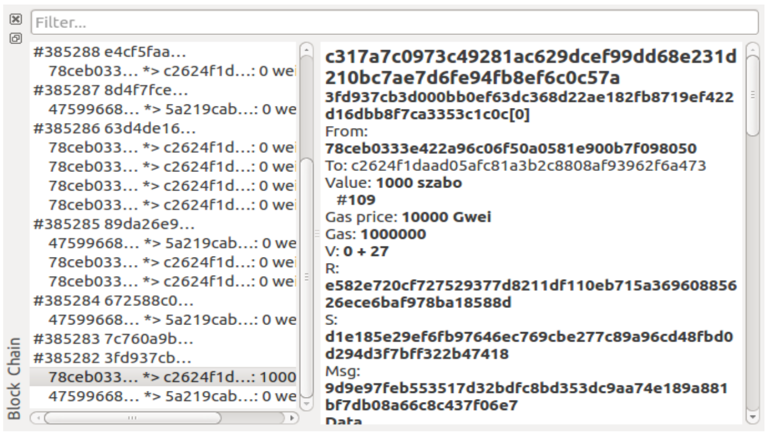
70be874e5148cb94bee70
6aab7353e21595b268652
49591c7945bb84d404037
3

From:
78ceb0333e422a96c06f50a0581
e900b7f098050
To:
c2624f1daad05afc81a3b2c8808
af93962f6a473
Value: **1000 szabo** **#108**
Gas price: **10000 Gwei**
Gas: **1000000**

```
0000 :??3Nikolaj????? 3a af f0 :
0010 ???????????????? 00 00 00 (
0020 ????             00 00 00 (
```

Figure 4. *A pending transaction*

In order to purchase a punch card, some amount of ether is needed. In the current implementation the price is set to 1 finney which is 0.001 ether. The price is set low for testing purposes but as stated it can be set as you wish, e.g., to match the price of the physical punch card. Upon the release of Ethereum, ether can be acquired through exchanges or through mining on the network. If one owns an adequate amount of ether you can purchase a punch card by typing your name in the name input field and click on the “Top up your punch card” button. This will initiate a transaction to the smart contract, powering the system, consisting of the payment of 1 finney, your name and that you want to invoke the purchase punch card function on the smart contract. This is all done through the JavaScript of the client. When using AlethZero, one can see the transaction in the pending window as seen in Figure 4. The transaction is completely transparent as all its data is publicly visible, in the sense that the computational operations a user wants to be carried out in addition to the data sent with it are listed. Whilst waiting for the transaction to go through, a loading screen appears. Once a new block has been discovered, the transaction will be submitted to the blockchain. The transaction is thus carried through once it is submitted to the blockchain.

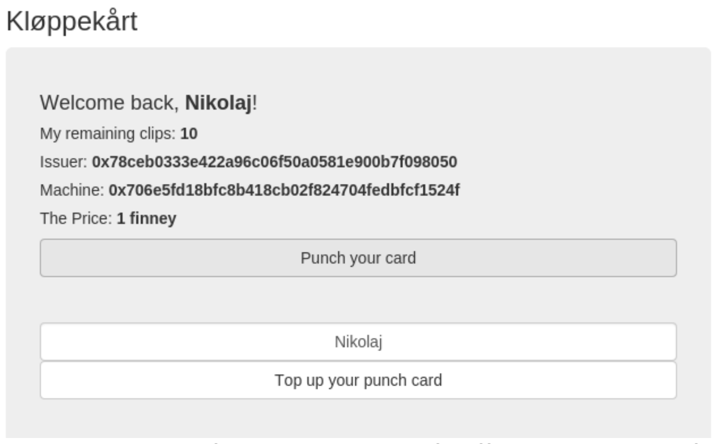


```
#385288 e4cf5faa...
  78ceb033... *> c2624f1d...: 0 wei
#385287 8d4f7fce...
  47599668... *> 5a219cab...: 0 we
#385286 63d4de16...
  78ceb033... *> c2624f1d...: 0 wei
  78ceb033... *> c2624f1d...: 0 wei
  78ceb033... *> c2624f1d...: 0 wei
  78ceb033... *> c2624f1d...: 0 wei
#385285 89da26e9...
  47599668... *> 5a219cab...: 0 we
  78ceb033... *> c2624f1d...: 0 wei
  78ceb033... *> c2624f1d...: 0 wei
#385284 672588c0...
  47599668... *> 5a219cab...: 0 we
#385283 7c760a9b...
#385282 3fd937cb...
  78ceb033... *> c2624f1d...: 1000
  47599668... *> 5a219cab...: 0 we
```

```
c317a7c0973c49281ac629dcef99dd68e231d
210bc7ae7d6fe94fb8ef6c0c57a
3fd937cb3d000bb0ef63dc368d22ae182fb8719ef422
d16dbb8f7ca3353c1c0c[0]
From:
78ceb0333e422a96c06f50a0581e900b7f098050
To: c2624f1daad05afc81a3b2c8808af93962f6a473
Value: 1000 szabo
  #109
Gas price: 10000 Gwei
Gas: 1000000
V: 0 + 27
R:
e582e720cf727529377d8211df110eb715a369608856
26ece6baf978ba18588d
S:
d1e185e29ef6fb97646ec769cbe277c89a96cd48fbd0
d294d3f7bff322b47418
Msg:
9d9e97feb553517d32bdfc8bd353dc9aa74e189a881
bf7db08a66c8c437f06e7
Data
```

Figure 5. The transaction is stored on the blockchain

In Figure 5, one can see that the transaction and its data are now stored publicly visible in a block. The system will now recognize the user through his or her address being mapped in the smart contract. Consequently, your name and a balance of 10 clips will be displayed as seen in Figure 6. This transaction is now stored forever on the blockchain for everyone to see, but unable to be tampered with.



Welcome back, **Nikolaj**!

My remaining clips: **10**

Issuer: **0x78ceb0333e422a96c06f50a0581e900b7f098050**

Machine: **0x706e5fd18bfc8b418cb02f824704fedbfcf1524f**

The Price: **1 finney**

Nikolaj

Figure 6. After buying a cup of coffee, the balance of the punch card has been updated

6 Analysis and evaluation of the blockchain prototype

For a meaningful analysis of the prototype, it must first be documented that it is in fact working as intended. To test the proof of concept of the smart contract implementation, an extensive unit testing would typically be applied to make sure that every method works correctly, especially in boundary cases. This poses a problem in the current state of Ethereum development, as there is no testing library available so far. There is one under development, but it is not ready to use yet.

Due to the lack of an automated test environment, a simple functionality test of the punch card system in form of a black box testing seemed to be acceptable. Black box tests are designed to test boundary cases for all the functions used by the client, except for the *setIssuer* function because that would mean to hand over the client to someone else, and the functionality is equal to that of *setMachine*, *commit-Suicide* and *emptyMachine* are not tested either, as they are trivial in their functionality. All tests but one passed. The one test that failed is in relation to the price of a punch card. The test is designed to test that negative numbers cannot be pushed, as it would result in a negative price. However, the function has been implemented using unsigned integers that does not allow for negative numbers, instead converting it to a very high number instead. Thus, the test failed but had no further consequence on the applicability of the prototype.

The testing of the client would normally entail some form of usability tests as well as tests of the functionality. There are a few different functionalities present in the client, however they all link up directly to functionality in the contract, and due to the pending window in AlethZero it is possible to check that the correct transactions are sent at each click. The remainder of the functionality links up to showing information on the screen, which is easily tested by manual checks in AlethZero as both the original issuer and a customer and a check of the contract in a non-Ethereum client (to check that it states the correct information).

As the implementation of the smart contract dictates the rules of the transaction processes, it removes the issue of trust from these. It is now possible to buy a punch card, given one has enough money. The blockchain based trust-free transaction system is then able to identify if a user has the right amount of clips for a cup of coffee. It will deduct the right amount of clips and update a user's balance automatically.

Consequently, the users do not need to know the transaction rules as they are instead dictated by the smart contract. This way, malicious behaviour and misunderstandings are ruled out by the system since they are carried out automatically according to the defined protocol. Additionally, due to the transparency of the operations conducted on the smart contract, users will also be able to verify that the actions are in fact carried out correctly, and that there is no invisible bias or malicious computing happening during transactions. Therefore, the system operates trust-free as the problem of trusting the users to carry out the transaction according to the rules has been circumvent.

7 Discussion and conclusions for blockchain based trust-free transaction systems

One of the downsides of the blockchain implementation is that transactions can take up to 12 seconds, due to the block time of the blockchain. When purchasing a punch card or a cup of coffee, the transaction has to go through and be submitted to the blockchain before any balances are updated. This means waiting for a new block to be discovered. In general, this would average out to around 6 seconds of waiting for your transaction to go through. The block time issue is hard to circumvent, e.g., by introducing a centralized service of some sort in conjunction with the blockchain to bring down transaction time. However, using the blockchain would then become pointless since such a solution would no longer be a purely decentralized system and thus compromising transparency, trust, and security.

The prototype system we designed relies on two smart contracts where one of them has all the transaction logic. If we would have to change inner workings of this specific smart contract, we had to issue the entire thing again which is costly in ether, and all current punch cards would be lost, meaning that ether would have to be manually transferred back to the customers, which is costly too. Also, the client needs to be updated with a new smart contract address for connectivity. To address this problem, "getters and setters" have to be implemented in order to give the issuer the opportunity to change a lot of the data stored in the smart contract without having to issue an entirely new one, for example, in order to change the address of the coffee machine's smart contract and the address of the issuer (essentially the administrator of the contract). The only reason to issue a new smart contract is if the logic of some of the transaction logics needs to be updated.

Another lesson learnt from the installation of the prototype is that the nature of smart contracts would make it easy to extend the system with new features, e.g., for automating other processes of the coffee shop. In transaction systems in general, the use of a blockchain can prove useful in regards to get statistics of sales and revenue as you will have a complete dataset of all transactions. Consequently, additional smart contracts can be implemented in order to handle the purchase of supplies through interacting with the record of sales. In this way, by automating processes through smart contracts in conjunction with each other, a shop can be turned into a decentralized autonomous organization.

7.1 When does a cryptographic economic system make sense?

There are many other ways to implement a digital punch card, than using blockchain technology. Digital assets are commonly known, so why not just implement a system where the punch card balance is maintained by a database running on a webserver?

When implementing a webserver based system, one is introducing a different aspect of trust, where users have to trust that the system works properly. Thus, the system will not be verified by anyone but the issuer. A system like this is completely opaque and the user will not be able to know the inner workings of the transactions. The user has no way of verifying that the transaction is processed correctly. The fact that such a solution is centralized introduces other problems as well. When the system runs on a centralized unit, it can be tampered with by someone hacking the system. Some level of security is needed. Also, one introduces reliability on hardware. One server crash can have catastrophic consequences if proper backups are not implemented. And no matter what, a crash will mean downtime and lost profit. These problems are inherently solved by a blockchain technology such as Ethereum. The lack of a central webserver means that there is no server that can crash and bring downtime. Even if a node crashes, the system will still be up. As you cannot tamper with the data stored on the blockchain, the system is much more resilient to hackers. Since the system is run transparently, one has not to necessarily trust the issuer either. Therefore, blockchain technology can be effectively used in transaction systems due its perks in the areas of security, transparency and trust – aspects that are all of utmost importance when conducting a transaction.

Naturally, blockchain technology has also some shortcomings, which need to be addressed when trying to establish a blockchain based trust free cryptographically secured transaction system. There are some issues that have not yet been solved and the implementation of a blockchain solution is still rather demanding. One of the main issues of blockchain technology is scalability. The problem is that for assuring the theoretically achievable security of the blockchain, a large number of full nodes are required (Buterin, 2014b, Buterin, 2014c, Buterin, 2014.). Otherwise, one might end up in a less decentralized system, like Bitcoin has experienced (Hashrate, 2015). The problem here is that this nullifies the security measure that decentralization is a big part of. Before the blockchain technology can be applied for a wider range of economic transactions and operations, it is a very important that this security aspect is addressed at a satisfactorily level. Ethereum has taken steps toward managing scalability better than Bitcoin. For instance, by using accounts rather than being based on unspent transaction outputs (Buterin, 2014b, Buterin, 2014c, Buterin, 2014., Buterin, 2015). The final solution presented to the scalability problem is to have many blockchains (Buterin, 2014c), e.g., some might only be used for some specific purposes while others are used for more generalized tasks (Au, 2001). The idea behind this is that the blockchains can use each other to provide security for one another, no matter what the separate blockchains are used for. This way, a miner can mine on a blockchain of a suitable size, and yet security would still be very high, albeit not as high as if only one blockchain existed. This solution also relies on a proof of stake model but because of the interconnectivity of the different chains, a high percentage of the combined stakes of the blockchain ecosystem would be needed.

7.2 How ready are blockchain-based cryptographic transaction systems?

An obstacle that is important for the more widespread use of blockchain technology is how the system copes with heavy transaction loads. A payment technology such as VISA handles 4,000 transactions per second on average and has been stress-tested in 2013 to handle 47,000 transactions per second (Manny Trillo, 2013). In comparison, Bitcoin can only handle 7 transactions per second, due to the fact that block sizes are restricted to have a maximum size of 1 MB (Bitcoin, 2015). This is not so much a problem if the blockchain is only used to buy coffee in a small coffee shop. However, if the same blockchain is to be used on a wide variety of applications, more performance is needed. If for instance the customers in the coffee shop cannot buy coffee because an election is being held somewhere else in the world and for electronic voting the same blockchain is used, it is going to be hard to sell the idea of the system to the coffee shop owner. It is obvious that this is not a viable amount of transactions, should this technology be implemented on a global basis. Changes to the way different firms use the blockchain are being made, to assure that a transactions per second level alike or even higher than VISA's can be handled.

Another disadvantage of running applications on the blockchain is the time factor. The blockchain we used in our prototyping has a block time of 12 seconds, meaning this is the potential waiting time for an action to be carried out. This is very unlike typical transaction time nowadays, where processing and Internet speed are at a level, where it is expected that requests are processed almost immediately. Therefore, this might be an obstacle in regards to the universal acceptance of the technology as an alternative to centralized services. For instance, in our proof of concept, we had to wait up to 12 seconds from "punching" our "cryptographic punch card" until we could get coffee, whereas today with the manual punch card it is an instant process.

And finally, the blockchain use is not free of cost, which is one of the more obvious drawbacks of decentralization and the blockchain technology. In the centralized world of today, someone pays for a server to run applications on it, or the user downloads a program that then runs on their computers. Most likely, the cost of running a server will somehow be paid by consumers since companies need to generate profit, and the costs of running a server will be priced into the products and services offered. However, this means that those cost are implicit and hidden from the user. On the blockchain, it is very blatant that transactions and computational power come at a price. This is definitely going to be one of the challenges of the IoT, as users are not used to pay in this way for services online, and that fact might hinder the spread of the technology. The fact that users will be constantly reminded that an action has a fee will certainly make some users choose centralized solutions where the prices are more hidden. For instance, when looking at the proof of concept presented in our prototype, there will be fees involved when buying a punch card, and every time it is punched (the gas or ether). These fees do not exist (or are not visible) in the current transaction process.

Obviously, we are at the beginning of understanding the potential of the blockchain and thus it is too early to theorize about all its aspects and full potential. However, we strongly believe that information systems research with its economics and information systems orientation on the one hand and computer science on the other is the discipline that can contribute significant insights by not only studying the acceptance and diffusion of blockchain solutions such as Bitcoin but also construct and design those solutions by applying a DSR approach. Thus, we hope that the potentials but also limitations we were able to illustrate by designing our proof of concept punch card solution is a starting point for more research to come on blockchain, the gateway to trust-free transactions.

References

Au, Y. A. (2001). "Design science I: The role of design science in electronic commerce research". *Communications of the Association for Information Systems* 7(1) 1.

Baskerville, R. (2008). "What design science is not". *European Journal of Information Systems* 17(5) 441-443.

Beck, R., S. Weber, and R. Gregory (2013) "Theory-Generating Design Science Research". *Information Systems Frontiers* 15(4) 637-651.

Bitcoin. (2015). *Scalability*. Available at: https://en.bitcoin.it/wiki/Scalability [Accessed 21st November 2015].

Blockchain (2015). *Blockchain Documentation*. Available at: https://bitcoin.org/en/ - block-chain [Accessed 21st November 2015].

Buterin, V. (2013). *Ethereum White Paper*. Available at: https://github.com/ethereum/wiki/wiki/ White-Paper [Accessed 21st November 2015].

Buterin, V. (2014a). *Ethereum Development Tutorial*. Available at: https://github.com/ethereum/wiki/ wiki/ Ethereum- Development-Tutorial [Accessed 21st November 2015].

Buterin, V. (2014b). *Scalability, Part 1: Building on Top*. Available at: https://blog.ethereum.org/ 2014/09/17/scalability-part-1-building-top/ [Accessed 21st November 2015].

Buterin, V. (2014c). *Scalability, Part 3: On Metacoin History and Multichain*. Available at: https://blog.ethereum.org/2014/11/13/scalability-part-3-metacoin-history-multichain/ [Accessed 21st November 2015].

Buterin, V. (2014d). *Toward a 12-second Block Time*. Available at: https://blog.ethereum.org/ 2014/07/11/toward-a-12-second-block-time/ [Accessed 21st November 2015].

Buterin, V. (2015). *Notes on Scalable Blockchain Protocols (verson 0.3)*. Available at: https://github.com/vbuterin/scalability_paper/blob/master/scalability.pdf [Accessed 21st November 2015].

Economist (2015). *Blockchain - The next big thing*. Available at: http://www.economist.com/news/ special-report/21650295-orit-next-big-thing [Accessed 21st November 2015].

Ethereum (2014). *JavaScript API*. Available at: https://github.com/ethereum/wiki/wiki/JavaScript-API [Accessed 21st November 2015].

Ethereum and Foundation. *Ether Unit Converter*. Available at: http://ether.fund/tool/converter [Accessed 21st November 2015].

Graham, R. (2013). *BitCoin is a public ledger*. Available at: http://blog.erratasec.com/2013/05/bitcoin-is-public-ledger.html - .VlArLMp4iKI [Accessed 21st November 2015].

Gregor, S. and D. Jones (2007). "The anatomy of a design theory". *Journal of the Association for Information Systems* 8(5) 312-335.

Hashrate *Bitcoin Hashrate Distribution*. Available at: https://blockchain.info/pools [Accessed 21st November 2015].

Higgins, S. (2015). *IBM Reveals Proof of Concept for Blockchain-Powered Internet of Things*. Available at: http://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet-things/ [Accessed 21st November 2015].

Holmström, J., M. Ketokivi and A. P. Hameri (2009). "Bridging practice and theory: a design science approach". *Decision Sciences* 40(1) 65-87.

March, S. T. and G. F. Smith (1995). "Design and natural science research on information technology". *Decision support systems* 15(4) 251-266.

Nakamoto, S. (2008). "Bitcoin: A peer-to-peer electronic cash system". *Consulted* 1(2012) 28.

Neal (2014). *Ghash.io promises not to break the Bitcoin marke.* Available at: http://www.theinquirer.net/inquirer/news/2322534/ghashio-promises-not-to-break-the-bitcoin-marke [Accessed 21st November 2015].

Orlikowski, W. J. and C. S. Iacono (2001). "Research commentary: Desperately seeking the “IT” in IT research—A call to theorizing the IT artifact". *Information Systems Research* 12(2) 121-134.

Panikkar, S. (2015). *ADEPT: An IoT Practitioner Perspective*. Available at: http://www.scribd.com/doc/252917347/IBM-ADEPTPractictioner-Perspective-Pre-Publication-Draft [Accessed 21st November 2015].

Price, R. (2015). *Nasdaq is experimenting with the revolutionary technology behind bitcoin*. Available at: http://uk.businessinsider.com/nasdaq-private-market-blockchain-bitcoin-experiment-currency-ledger-2015-5 [Accessed 21st November 2015].

Pureswaran, V. and P. Brody. (2015). *Device democracy: Saving the future of the Internet of Things*. Available at: http://public.dhe.ibm.com/common/ssi/ecm/gb/en/gbe03620usen/GBE03620USEN.PDF [Accessed 21st November 2015].

Simon, H. (1969). "The sciences of the artificial". Cambridge, MA, Cambridge: MIT Press. p. 252.

Trillo, M. (2013). *Stress Test Prepares VisaNet for the Most Wonderful Time of the Year*. Available at: http://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html [Accessed 21st November 2015].

von Alan, R. H., et al. (2004). "Design science in information systems research". *MIS quarterly* 28(1) 75-105.

Walls, J. G., G. R. Widmeyer and O. A. El Sawy (1992). "Building an information system design theory for vigilant EIS". *Information systems research* 3(1) 36-59.