

Automated near real-time Cheque Clearing System with Digital Cheque

Weerasinghe A
2015



Automated near real-time Cheque Clearing System with Digital Cheque

**A dissertation submitted for the Degree of Master of
Science in Information Security**

**Weerasinghe A
University of Colombo School of Computing
2015**



Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Students Name: Weerasinghe A

Signature:

Date:

This is to certify that this thesis is based on the work of

Mr. Weerasinghe A

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. T. N. K. De Zoysa

Signature:

Date:

Abstract

Cheque is the most demanded non-cash retail payment instrument in Sri Lanka, which is accounted for more than 90% of total non-cash retail payment transaction.

Total value cheques cleared per day is gradually increasing and it has been surpassed more than 20billion per day. These volumes suggest the continual importance of cheques in Sri Lankan Economy.

Various modes of digital payments are taking the space of this well mature payment mode due to its few weaknesses, such as time to realize a check, signature validation and logistics. To overcome such weaknesses, this research proposes a secure near real time cheque realization process based on blockchain technology. Ethereum platform has been chosen to demonstrate the proof of concept due to its ability to run the programming code of any decentralized application on blockchain technology.

Signature validation of paper cheques are getting more and more difficult due to increasing cheque volumes. To properly validate the signatures, it is required to keep many factors to be accurate in cheque imaging. Signature forgery also is not easy to detect in this manual or semi-automatic process and leading many financial loses. To overcome these issues and instantly validate authenticity of cheques, digital signature and data encryption is used coming with blockchain technology. So entirely electronic based cheques, digital signature verification is instant and delivering cheques to the end recipient can have many different logistical modes.

Participating virtual banks are acting as cheque book issuance authority for its own customer base. When a Customer request a digital Cheque book (reserved cheque number range) via his own online banking interface and then he get a proper cheque image, digital signature and serial number range. Customer affiliation to the bank is established via the validation of bank issued signature, which was previously registered with bank in the initialization of cheque book application, against the Ethereum wallet address of the checkbook application.

Digitally issued cheques are submitted to the respective bank's smart contract server to validate the authenticity of the customer and then forward to the central Smart contract server to forward to the payees bank. The whole process can be automated to digitally validate the signature of drawer of cheques.

Acknowledgements

I would like to express my special appreciation and thanks to the supervisor Dr. T. N. K. De Zoysa for encouraging me to use blockchain for this research. Without His encouragement and support, I would not have come this far.

I would especially like to thank the MIS lecture panel for giving specialty knowledge on information security and the panel members, who were evaluating the project proposal and interim assessment, to give valuable comments to steer the research in to right direction. Thanks to you.

A Special thanks goes to my family, Who went through numerous hardships to find me time to achieve the goal.

Table of Contents

Introduction	1
1.1 Cheque Layout and fields	2
1.1.1 Amount field.....	3
1.1.2 Transaction code field.....	3
1.1.3 Account number field	3
1.1.4 Bank routing transit number field.....	4
1.1.5 Cheque number field.....	5
1.1.6 MICR characters.....	5
1.2 Brief History	5
1.3 problems	6
1.3.1 Highly resource intensive in terms of people and hardware.....	6
1.3.2 Poor Signature validation could lead to cheque frauds	6
1.3.3 Check image have to be image friendly, so that scan copy to be clear	6
1.3.4 Irregular endorsement of order cheques	6
1.3.5 MICR line reading errors.....	6
1.3.6 Inefficient cheque deposit boxes operation	7
1.3.7 Encoding errors of account number and amount.....	7
1.3.8 Logistical issues of cheques.....	7
1.3.9 Time restrictions on clearing process	7
1.3.10 Validity of BlockChain in this Context	8
1.4 Objective of the Project.....	9
1.5 Scope and Limitations.....	9
2. Literature Review	11
2.1 Cryptographic hash functions.....	12
2.2 Turing Completeness for smart contracts.....	12
2.3 Bitcoin – Birth of blockchain	13
2.4 Proof-of-work	14
2.5 Mining new blocks in Ethereum.....	15
2.6 Merkel Tree	16
2.7 E-Cheque	16
2.8 Cheque Digitization by HITACHI Group	17
2.9 Ethereum.....	17
2.10 GAS	18
2.11 Smart Contracts	18
2.11.1 Scripting.....	19
2.11.2 Storage	19
2.11.3 Jump operations.....	19

2.11.4 Contract operations.....	19
2.11.5 Inspection operations.....	20
2.11.6 Logging operations.....	20
2.11.7 Ethereum implementations.....	20
2.11.8 Ether.....	20
2.11.9 Solidity.....	21
2.11.10 Mining.....	23
3. Design and Methodology.....	25
3.1 Methodology.....	25
3.1.1 Cheque Book Creation.....	26
3.1.2 BlockChain Extranet and Transaction Realization.....	26
3.1.3 Digital Signature.....	27
3.1.4 Accounts on Ethereum.....	27
3.1.5 Building a test network.....	28
3.2 Design.....	29
3.2.1 Architecture Diagram.....	29
3.2.2 Ethereum Private network for cheque clearing.....	30
3.2.3 Transaction flow.....	31
3.2.4 Basic component integration of Ethereum implementation.....	33
4. Implementation.....	34
4.1 Ethereum Single Node system Private Network.....	34
4.2 Installation.....	34
4.3 Signature creating and validation in Ethereum.....	36
4.3.1 Web3 Interface (Signature Creation).....	37
4.3.2 Signature Verification.....	38
4.4. Client Side.....	40
5. Results and Evaluation.....	41
5.1 Cheque book issuance.....	45
5.2 Issue a Cheque.....	45
5.3 Withdraw Money.....	45
6. Conclusion and Future Work.....	46
6.1 General Discussion.....	46
6.2 Findings.....	46
6.3 Limitations.....	47

Abbreviations

DSA	Digital Signature Algorithm
Drawee	Person who is withdrawing money from the cheque
Drawer	Person who is writing the cheque
ECDSA	Elliptic curve Digital Signature Algorithm
Ether	Ethereum Currency
EVM	Ethereum Virtual Machine
FIFO	First In First Out
HMAC-MD5	Hash-based Message Authentication Code Digest Algorithm
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MICR	Magnetic Ink Character Recognition
P2P	peer to peer
POW	Proof of Work
RIPE-MD RACE	Integrity Primitives Evaluation Message Digest
RSA	Ron Rivest Adi Shamir and Leonard Adleman
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
UTXO	Unspent transaction output
XBT	Bitcoin Currency

Chapter 1

Introduction

A cheque is defined in section 73 of the Bills of Exchange Ordinance No.25 of 1927 as “a bill of exchange drawn on a banker payable on demand” (“bills of Exchange”, 1927). Hence, a cheque is an unconditional order in writing signed by the person giving it (i.e. drawer) requiring the bank to whom it is addressed (i.e. drawee bank) to pay on demand a sum certain in money, to or to the order of a specified person (i.e. payee) or to bearer”. This is basically a contract between drawer and drawee through reliance on third party Banks.

Paper based checks are continued to be popular and convenient mechanism for payments in Sri Lanka and in the world. Realizing a cheque has taken days and finally it has been expedite to the level that interbank clearing process to achieve T + 1 day (Transaction date + one day) clearing cycle. Most significantly, more than 90 percent of the total values of cashless retail payments are done through cheques. This statistic highlights the importance of cheque based transactions in the national payment system. (“LankaClear”, 2018)

Present Cheque Imaging & Truncation (CIT) is an image-based cheque clearing system, which replaces the physical presentment of cheques with electronic image of cheque, flowing throughout the clearing cycle. This process eliminates the actual paper cheque movement in cheque clearing cycle and reduces the delays associated with the movement of cheques. Please refer the below (Figure 1.1) for stages of cheque movement at present.

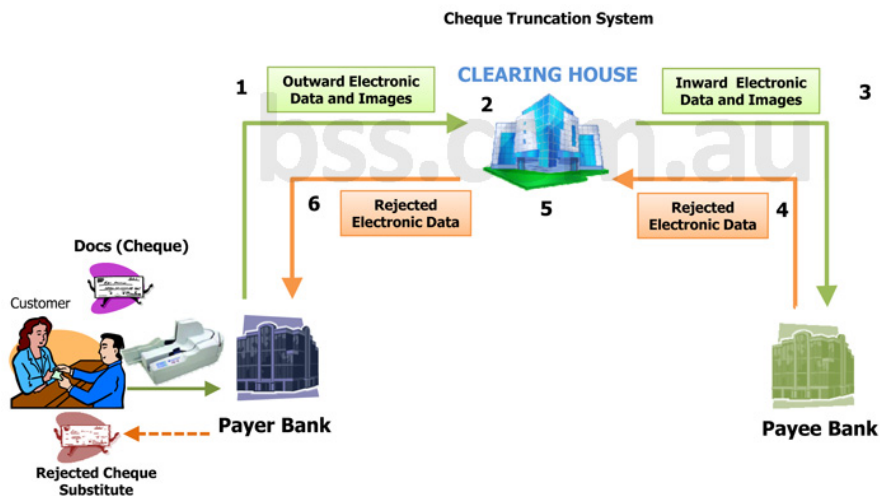


Figure 1.1

However, even to achieve T+1 day cheque clearing cycle, Banks and financial institutions have to deploy lots of resources in terms of people, process and technology. Present automation is just digital imaging of paper cheque to make logistics of cheques among banks

though the electronic media. Cheque validation is done manually with visual matching of the signature of the drawer of the cheque to the, what is on the scanned or paper cheque. With the huge volumes of cheque are being handled every day, this tasks needs lots of manpower. There are systems to validate the signatures, but there is a margin for errors in those systems. Each location, which accepts cheques needs cheque scanners and accurate reading of MICR line is essential to have processing of cheques without any delays.

Entire process is highly resource intensive and stressful and financial institutions needs to invest considerable CAPEX/OPEX to drive the process. Failure to meet stipulated deadline by central processing house, impose penalties for such banks.

It is imperative that all participating banks issue image friendly cheques to their customers for the smooth operation of the present CIT System. It is equally important that customers are informed that they should not issue or accept non-image friendly cheques since it have fair chances of not being accepted by the system due to poor image quality of scanned image.

1.1 Cheque Layout and fields

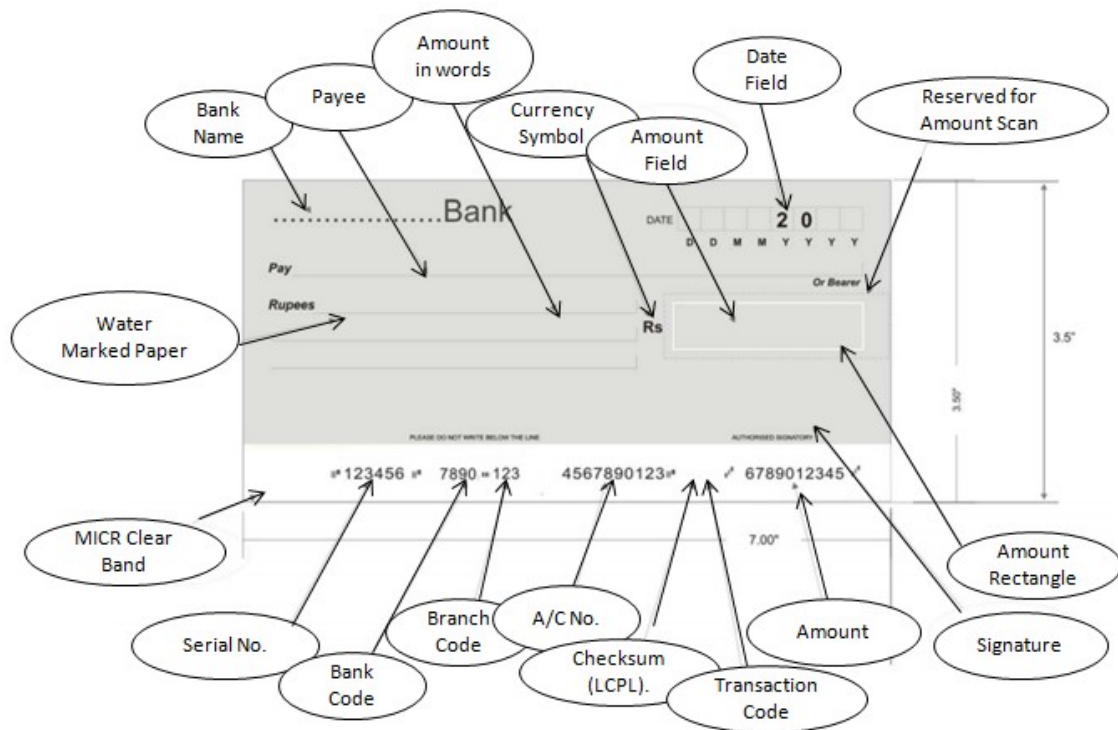


Figure 1.2

1.1.1 MICR CODE LINE

The MICR code line comprise of a series of MICR characters divided into fields, which are encoded at the 5/8" clear band at the bottom edge of a MICR document, and follows a

specific format that has been adopted for a given document processing system.

In Sri Lanka, the MICR code line used in the computerized cheque clearing system consist of five fields, the specifications of which were designed by the Central Bank in close association with Sri Lanka Bank's Association.

The MICR goes through a high-speed document processing reader sorters and therefore, it is imperative that the MICR is accurately printed. An error in the encoding of the adopted code line format and poor quality of the MICR encoded characters may cause rejects and delay in the completion of work. For this reason only reliable MICR cheque printers should be used for sorting/reading of all cheques.

The fields in the MICR code line are read from right to left of a document in their printed positions. Each field is bounded by specific symbols (delimiters) which serve to identify one field from the other while being read by the Reader/Sorter.

All fields are detailed below to provide a clear understanding of their functions, and importance. Please refer the [Figure 1.2](#).

1.1.2 AMOUNT FIELD

This is the first field to be read by the Reader / Sorter at high-speed. It is opened by an amount symbol, followed by ten digits and closed by another amount symbol.

1.1.3 TRANSACTION CODE FIELD

This field follows the amount field. It is opened by the amount field's closing symbol (amount symbol) and followed by two digits and closed by the opening (on-us) symbol of the account number field.

This field is used to identify the type of document to be processed. Clearing documents are assigned specific transaction codes. Encoding the wrong code in this field will cause wrong identification.

If a participating bank finds that the regular MICR cheques used by it does not indicate any transaction code then such bank should leave that data field BLANK.

1.1.4 ACCOUNT NUMBER FIELD

This is the 3rd field on the MICR code line. It follows the transaction code field. The account number field is opened by the transaction code field's closing symbol (on-us symbol), followed by ten digits and closed by a transit number symbol.

On regular MICR cheques, this field is pre-qualified by the MICR cheque printers or

encoded by the issuing bank. Verification of the encoding in this field is a must as this is an important field in the MICR code line affecting the drawee banks.

1.1.5 BANK ROUTING TRANSIT NUMBER FIELD

This field is known as the BRN field. It contains the transit number of bank/branch. The BRN field is opened by a transit number symbol, followed by three digits, a dash symbol, and four digits. It is closed by an (on-us symbol). The next four digits represent the bank number and area code, i.e. XYYD - ABC where: X = classification Code YY = Bank Code D D = check digit ABC = Branch Code All banks/branches participating in clearing operations are assigned BRN numbers and a register is maintained for this purpose.

1.1.6 CHEQUE NUMBER FIELD

The cheque number field is the last field to be read by the Reader / Sorter and it follows immediately after the BRN field. It is opened by the BRN field's closing symbol (on-us symbol), followed by six digits, and closed by another (on-us symbol). This symbol is the last character on MICR code line and also closes the cheque number field reading from right to left.

1.1.7 MICR CHARACTERS

The MICR E13B characters consist of numbers and symbols (delimiters) containing magnetic ink, which are capable of being read by both humans and computers.

1.2 Brief History

In early days cheque clearing were done by individual banks and later on automated clearing house (SLACH) was established by the central bank to process inter-bank clearing.

LankaClear's inception can be traced to the former Sri Lanka Automated Clearing House (SLACH) owned by the Central Bank of Sri Lanka (CBSL). In 2002 the CBSL took a policy decision to divest the activities of the SLACH and together with all licensed commercial banks operating in Sri Lanka, CBSL took shares to launch LankaClear (Pvt) Ltd.

The Company was incorporated on 8th February 2002 and commenced operations on the 1st of April in the same year. The Company introduced the US Dollar Clearing System on 1st October 2002.

The Cheque Imaging and Truncation System (CITS) for cheque clearing was introduced in

May 2006. This has changed the cheque clearing system in Sri Lanka in terms of time taken for a cheque to be cleared irrespective of where it was issued within the country.

1.3 Problems in present system

1.3.1 Highly resource intensive in terms of people and hardware.

Millions of cheques are exchanged in a day and there are specially trained people in each bank to manage the process. Each and every bank branch is having specially built hardware scanners to scan cheques and whole process is time bound with heavy penalties for any delay. There are many failure points in the chain and very close scrutiny is needed to make sure daily process is completed in timely manner.

1.3.2 Poor Signature validation could lead to cheque frauds

Signature forgery and cheque alteration (Altering fields to show a value, payee's name and/or date not originally authorized by the drawer of the cheque) is not so uncommon.

Due to large volumes and manual processing, only a small percentage of the signatures on cheques are actually reviewed on screens by banks. Some banks define safe limits to ignore signature verification just to manage the time and resource.

1.3.3 Check image have to be image friendly, so that scan copy to be clear

Since whole clearing process is based on the quality of the scanned image, it is essential to have image friendly cheque image issued by the banks.

1.3.4 Irregular endorsement of order cheques

Order cheque must be endorsed by the payee. The A/C payee crossing does not dispense the need for the endorsement. Therefore, collecting/presenting banks must first ensure that such cheques are endorsed properly and also ensure that the cheque is collected to the payee's A/C. if the endorsement is unreadable or blurred, cheques get rejected.

1.3.5 MICR line reading errors

The MICR goes through a high-speed document processing reader sorters and therefore, it is imperative that the MICR is accurately printed. An error in the encoding of the adopted code line format and / or poor quality of the MICR encoded characters may cause rejects and delay in the completion of work.

1.3.6 Inefficient cheque deposit boxes operation

Some banks operate cheque deposit boxes but it is only a convenient way of collection of cheque, however still needs to go through the existing clearing process with said weaknesses.

1.3.7 Encoding errors of account number and amount

MICR encoder operator must exercise care and be accurate when encoding this field. An encoding error in this field will cause an item to be miss- cleared or rejected.

1.3.8 Logistical issues of cheques

Submitting cheques for realization is improving but still it has its own weaknesses due cheque being a paper and physical delivery is necessary to the presenting bank.

1.3.9 Time restrictions on clearing process

All Banks must follow the below mentioned cutoff times in Table 1.3.1 irrespective of volumes.

Clearing Leg	Window Open	Window Close	Activity	From	To
Outward Return	09:30 a.m.	11:30 a.m.	Transmit Outward Returns data via VPN of LCPL	Paying Banks	LCPL
Inward Return	01:15 p.m.	Next business day 07.30 a.m.	Issued CRN Data by LCPL & download Inward Return data/Reports via online	LCPL	Collecting Bank
	N/A	1.30 p.m.	Ready to dispatch Inward Return data CDs	LCPL	Collecting Bank
Outward	10:00 a.m.	07:00 p.m.	Submit Outward CDs	Collecting Bank	LCPL
Inward	N/A	12.00 a.m.	Ready to dispatch Inward CDs	LCPL	Paying Bank

Table 1.3.1

Otherwise, heavy penalties would be imposed by the LCPL (CBSL). Therefore, this research and implementation is looking at the present cheque issuance and clearing system to mitigate above mentioned weaknesses by replacing paper check with a digital cheque with smart contract built on block chain technology for cheque authentication and validation without compromising the prevailing legal framework of “Bills of Exchange Ordinance” for cheque presentment and clearing.

1.3.10 Validity of Block chain in this context

blockchain is commonly described as a distributed ledger where every update or movement on the chain is saved on all the components of the ledger.

This new technology can also be defined as a cryptographically secured database shared across thousands of computers.

As mentioned before, this technology is based on a decentralized chain. This chain is mainly formed by servers and customer devices spread all around the bank network and its extranet that are constantly sharing information between them and storing this information in blocks.

These servers sustaining the full blockchain are known as nodes, and as long as one node is running, the entire blockchain will be running. This is a private chain, where only some trusted and known participants can access.

Transactions are the actions of sharing information and blocks are where all this information is stored. These blocks are essential for the blockchain because they also save a record of time when these transactions were added to the blockchain. Furthermore, when one block is formed it is added on top of the other blocks, which means that the chain only can be extended and previous records cannot be changed. This fact, related with the time stamp, makes the blockchain with cheque transaction records immutable. Moreover, as it is a distributed ledger, the blocks are not only formed in one server but a copy of each block is updated in all the servers sharing the ledger among the bank network.

Since the cheque issuance to cheque realization is totally digital on the block chain, integrity of the transaction details is preserved. Therefore, the customer application and bank side applications can accurately retrieve transaction information instantly, unlike in MICR reading in present clearing system. There is no need for and image scanning, since relevant cheque image is generated completely digitally.

Apart from transactions and blocks, there exist two types of keys: public keys and private keys.

public key are given to whom has created an account (cheque book account) on the blockchain. This public key is the personal address of the new account and as its name suggests, it is public for extranet.

On the other hand, private keys are the ones that are only known by the owner of the cheque book account. These ones allow the owner to validate incoming and outgoing cheque transactions.

hash is created every time a block is settled down. Each block has a different hash from the others. The hash is like the serial number to identify blocks along the network. Furthermore,

this hash is what helps with the security of the blockchain as all the nodes are validating hashes from previous blocks ever since a transaction is trying to be executed.

Two customers of the blockchain, drawer and drawee want to exchange cheque transaction between them. Using their public and private keys, the information requested to go from one peer to another is started to be validated by all the nodes of the chain. Once the transaction is validated, a block is formed; and so is the hash of this block. Then, it is stored on top of the chain. Finally, the transaction between drawer and drawee is settled down and stored in the block chain.

Therefore, without going through manual signature validation, digital signature validation can be done with respective keys to establish the authenticity of cheque transaction quickly.

1.4 Objective of the Project

The main goal of the research is to design a digital cheque presentment and processing architecture to handle transactions using blockchain technology to address weaknesses mainly in the signature verification process.

1. Build conceptual design on blockchain analogues to the paper cheque clearing process
2. Built smart contract on Ethereum blockchain to reflect basic features of paper cheques
3. Real time signature verification of drawer and validation of cheque book.
4. Preserve the appearance of the cheque very similar to paper cheque in digital form for easy reference.
5. Archival and retrieval of old cheques in digital form

1.5 Scope and Limitations

To develop a secure architecture to achieve digital cheque issuance and clearing process on blockchain, so that banks and financial institutions could easily integrate and adapt to the digital form of cheque in parallel to paper based system.

Under this project, research is conducted in to using blockchain to create digital cheques, which are tamper free and only a person with the correct cheque number and signature can unlock and cash out the cheque. Cheque instruction will be built in to smart contracts, so that such instructions are immutable and guarantee the integrity.

Limitations:

1. Proof of concept is built on single node Private Blockchain network, so that block mining not resource intensive and source code of EVM is changed to minimize the effort of mining.
2. Present regulations for digitalized paper based cheques, to Sec 73 of Bill of Exchange Ord. - the following will apply.
“It shall include the complete and accurate electronic image of a cheque presented for payment by the Collecting/Presenting Bank by electronic means to the paying bank. (Sec.33(1) Payment & Settlement Act No. 28 of 2005)”
“It shall also include an image return document/cheque return notification (CRN) issued by the Collecting/Presenting Bank to the holder who delivered the cheque for collection. (Sec.34 (1) Payment & Settlement Act No. 28 of 2005.)”
3. Handling Cheque bouncing due to insufficient funds is not considered in to this project.

Chapter 2

2. Literature Review

There are few related projects in to this area. One such is the projects initiated by the Financial Services Technology Consortium (FSTC), which consisted of financial institutions, hardware and software firms, governmental agencies and others of United States of America. They have evaluated several models to securely manage digital certificates and align the clearing process with the existing paper based cheque clearing system. Lots of work has been done to develop sub set of XML framework to successfully use for the eCheques. (Milton M, 1998)

In 2015, Hong Kong Interbank Clearing Limited (“HKICL”), which is a private company jointly owned by the Hong Kong Monetary Authority (“HKMA”) and the Hong Kong Association of Banks (“HKAB”), has developed a eCheque issuance and clearing process for Hong Kong. eCheque is digitally signed and produce in PDF format, so that payee could directly upload to their own online bank portal or to dropBox of HKICL. This process is very similar to the FTSC process, but HKICL has successfully integrated all the commercial banks operated in HongKong very much similar to the paper based cheque clearing system in HongKong. (HKMA, 2017)

In 2008, A purely peer-to-peer version of electronic cash system was introduced by Satoshi Nakamoto to allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but double spending is eliminated without using trusted third party.

Although, the ownership transfer of the cheque is legally prohibited, limited ownership transfer of the chequebook from bank to the chequebook owner and then drawing cheques could be established in similar fashion of bitcoin chain of ownership mechanism.

Here Ethereum, project based on blockchain comes handy , because it allows to create smart contracts, which triggers the instructions built in to special accounts in the blockchain, when a user communicate to the contract account. (Nakamoto, 2008)

Know Your Customer (KYC) is a challenge, in the blockchain technology. As distinct from KYC, Anti Money Laundering (AML) and Countering the Financing of Terrorism (CFT) are about understanding patterns of money flow at the transaction level to avoid helping criminals.

Banks get into trouble when they get caught moving money around if the money has illegal or immoral origins (dirty money) or is used for illegal purposes, such as funding terrorism.

However, there are projects built on Ethereum framework to generate gift card systems, which uses blockchain technology with smart contracts. It is entirely working on individual accounts holding Ether and there is no any banking system connected to it.

2.1 Cryptographic hash functions

A cryptographic hash function is a function that maps any chunk of data on to a fixed length string of characters. A cryptographic hash function is considered practically irreversible, as in it is computationally infeasible to compute the original message given its hashed value. Such a function must adhere to certain features in order to be considered a good and secure cryptographic hash function. First of all it must provide compression in the sense that the output must be relatively small compared to the input. Moreover it has to be easy to compute a hash of any input value in a small amount of time and the time complexity should not grow rapidly as the length of the input grows. Lastly it must be resistant to collisions, meaning that it has to be computationally infeasible to find any two distinct values that produce the same hash function output. Should such a collision be discovered then the function is considered to be broken.

2.2 Turing Completeness for smart contracts

Turing Completeness is a Computational Theory term used to denote that a system that acts on data can compute any function that the Universal Turing Machine can compute. That effectively states that no automaton can ever be built that has more computing capability (as to the number of functions it can compute) than the Turing machine. (Cambridge, 2012)

System, such as a programming language, as Turing complete it is basically stating that this system can be arranged in such a fashion that it can perform any real world computation. Turing completeness does not take into account performance or resource consumption aspects. If a programming code built in to the blockchain to execute certain functions, such code should have above characteristics. Ethereum built on blockchain technology comes with smart contracts with characteristics of Turing complete.

2.3 Bitcoin – Birth of blockchain

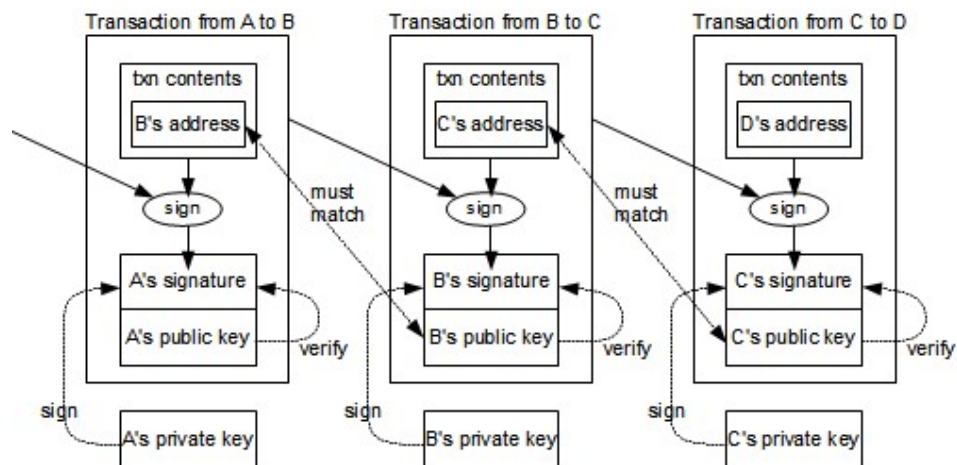


Figure 2.1

The first bitcoin Blockchain technology is derived from the original white paper, (Nakamoto, 2008), which outlines how to build crypto currency Bitcoin. Bitcoin solves a very important problem in the field of electronic money called double-spending, ie use the same electronic coin to pay for many things. It is usually resolved through a central authority, such as a bank or other third-party trusted but Nakamoto proposed a time stamp server, which ensures that all transactions occur chronologically in the database. The author (s) suggested using a Proof-of-Work algorithm for establishing a consensus on which chain is correct one. It establishes an incentive for users to be correct in validating transactions. It's important, making it more expensive in fake transactions than potential gain. There is no appropriate algorithm for establishing a blockchain agreement, there may be reluctant to blockchain-system Bitcoin because anyone with access to the history of transactions (all nodes), can rewrite history and publish it as true. At Bitcoin, users do not have accounts or account balances, but rather signs transactions using their private key. Each bitcoin is linked to a public key through an unspent transaction output (UTXO) and the user who possesses the corresponding private key is the owner and can control the usage of it. UTXO is there because, on Bitcoin, coins sent to an address need to be spent, even if the user is actually does not want to spend the full amount. However, it is possible to divide a transaction. Assume that a specific address contains 3 XBT and the owner of the private key to that address, i.e. the owner of the bitcoins, wants to pay 1 XBT to another address. . The new 1 XBT transaction will use the entire 3 XBT as the input and the 3 XBT are thus spent. Thus, the change in this transaction, the 2 XBT, will be sent back to the same user but as a new input using a new address. One user Participating in payments, whether as a payer or paying, will have a collection addresses, all summed up to

the total balance of his wallet. Because the first one The blockchain to be used is Bitcoin, there is at first no difference between Bitcoin and blockchain.. Many used cases based on blockchain are in the financial sector, however a Bitcoin-like crypto currency Ethereum has added functionality for smart contracts.

Bitcoin address

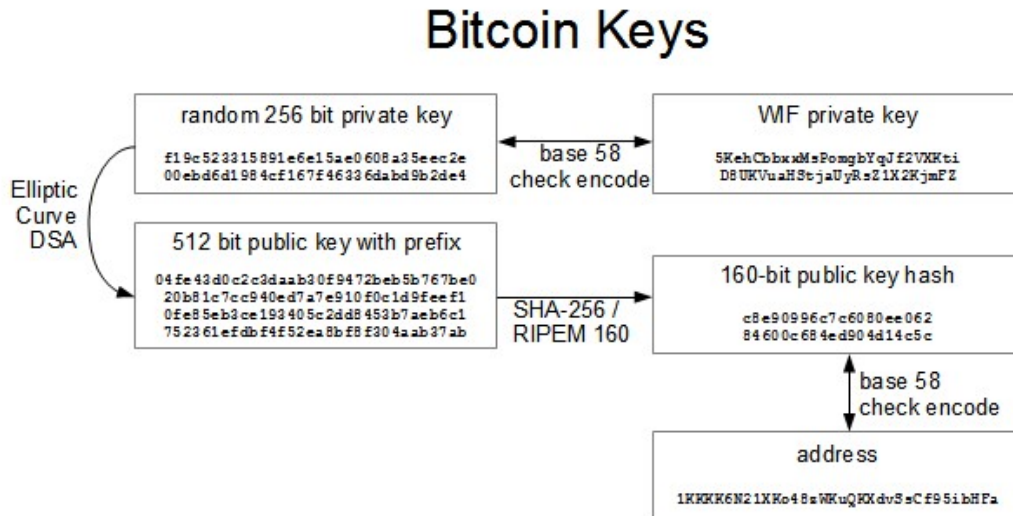


Figure 2.2

2.4 Proof-of-work

Proof-of-work is a mechanism that is used to ensure that a distributed consensus is achieved without either the presence of a central authority or the requirement that a set of participating users are identified in some way. Proof-of-work is implemented as a requirement for each block in the blockchain to have a hash smaller than a given target number. This target number is determined through a calculation that is carried out every 2016 blocks (roughly two weeks), and aims to ensure that, independently of the global amount of computing power, a new block will be generated roughly every 10minutes in average. But for the private usage of blockchain, these parameters can be adjusted or removed to make sure that, new blocks are generated much quicker in trusted distributed network, (Nakamoto, 2008). Proof of work for the Ethereum is very much similar to proof work used in Bitcoin.

2.5 Mining new blocks in Ethereum

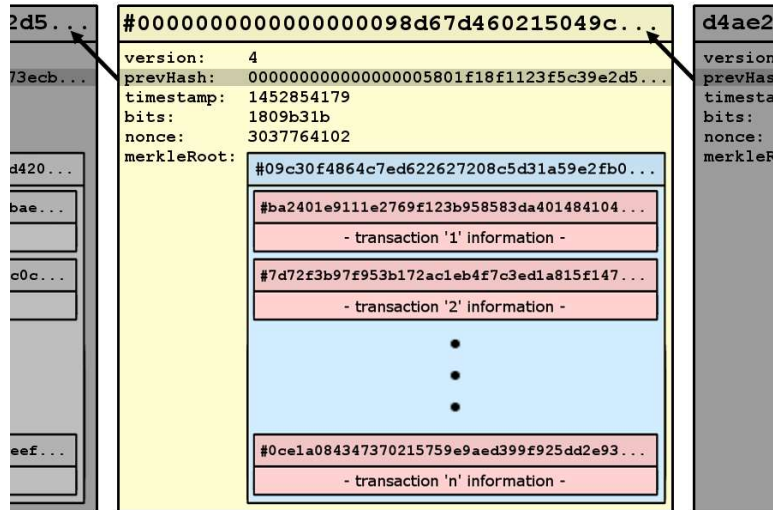


Figure 2.3

Usually, banks are in charge of keeping accurate records of transactions. They ensure that money isn't created out of thin air, and that users don't cheat and spend their money more than once(double spending).

Blockchains, though, introduce an entirely new way of record-keeping, one where the entire network, rather than an intermediary, verifies transactions and adds them to the public ledger. Someone still needs to secure the financial records, ensuring that no one cheats. Mining is one innovation that makes decentralized record-keeping possible. Miners come to consensus about the transaction history while preventing fraud (notably the double spending of ethers) – an interesting problem that hadn't been solved in decentralized currencies before proof-of-work block chains.

If the mining server finds a hash that matches the given current target, the mining server will be awarded ether and broadcast the block across the network for each node to validate and add to their own local copy of the ledger. If server B finds the hash, server A will stop work on the current block and repeat the process for the next block.

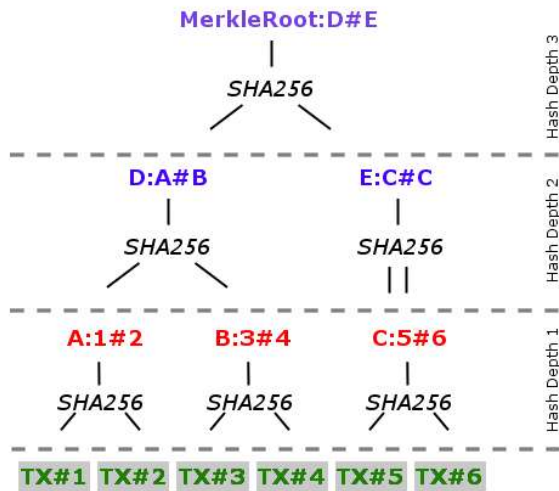
On the other hand, other mining nodes can instantly verify that the hash value is correct, which is exactly what each mining server does.

Approximately every 12–15 seconds, a mining server finds a block. If servers start to solve the puzzles more quickly or slowly than this, the algorithm automatically readjusts the difficulty of the problem so that miners spring back to roughly the 12-second solution time.

But for the private usage, there algorithms can be altered to make instant block creation. Ethereum uses “ethash” algorithm for the proof of work. (blockgeeks, 2018)

2.6 Merkle Tree

When a new block is being mined, the transactions framed in to it, have been locked in. The



transactions are laid out and hashed together in pairs to create a new SHA256 hash. This is recursively done until a single hash remains; the merkle root. Below is a diagram showing how the hashes are made on each level of recursion. On any given level, there may not be an even number of hashes, if this is the case the last hash to be processed is hashed with itself.

Figure 2.4

2.7 E-Cheque:

There was a gift card system created using Ethereum to generate online gift cheques. However that system is no more available due to unknown reasons.



Figure 2.5

2.8 Cheque Digitization by HITACHI Group

Tokyo-Mitsubishi UFJ, one of Japan's largest banks being under the same group to which Bank of Ayudhya belongs, i.e. Mitsubishi UFJ Financial Group (MUFG), collaborated with Hitachi Group, in testing the Blockchain technology in Singapore. This project involves the development of a prototype system for Tokyo-Mitsubishi UFJ, under which electronic cheques are issued and sent to Hitachi Group in Singapore for settlement. Testing is under the Regulatory Sandbox of the Monetary Authority of Singapore (MAS). However still the system is not in live operation. (Hitachi, 2016)

2.9 Ethereum

Ethereum is a crypto currency system built based on blockchain technology to provide decentralized general purposes computing machine. Underlying currency is called Ether and the programs that runs on de-centralized computer are called as smart-contracts. Smart-contracts are automatically enforced through the blockchain validation process carried out by the full nodes in the network.

Full nodes are those that download and validate the whole blockchain, these nodes do not need to trust any other node, since they can validate the whole transaction history. In contrast, because the size of the blockchain is considerable, portable devices often use so called lightweight clients, which only store part of the block chain and rely on full nodes to validate transactions.

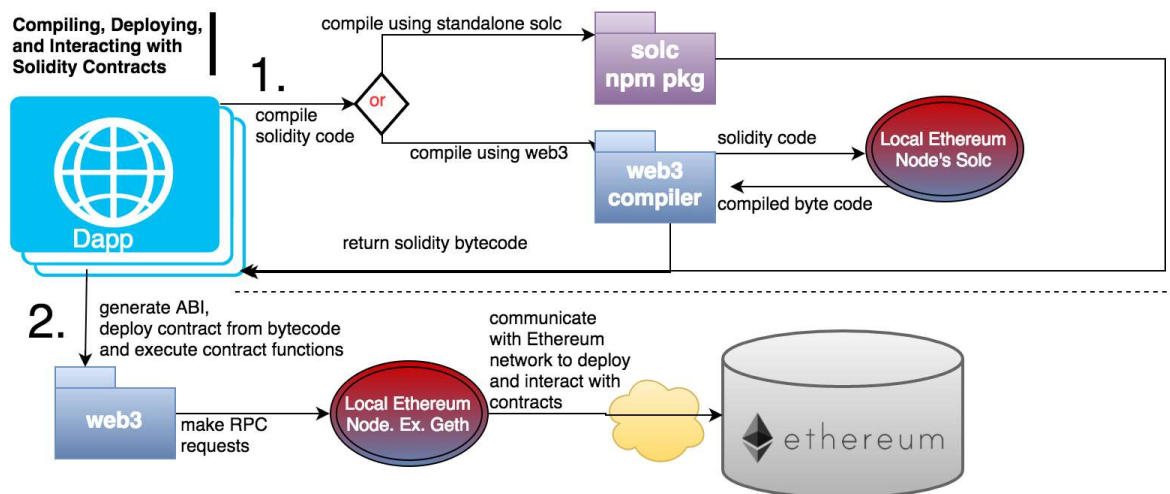


Figure 2.6 Source (Karl F, 2016)

Unlike Bitcoin, the Ethereum scripting mechanism allows for looping behaviour through the use of both jumps and recursive calls. To stop DoS (Denial of Service) attack by sending a

transaction that loops forever, Ethereum also introduces a limit to the execution time of each transaction that is called gas.

2.10 GAS

Gas is an amount of ether, paid in advance when a transaction is issued, that covers the cost of executing the transaction. If a transaction runs out of gas while it is being executed, the transaction will be rolled-back but the gas consumed will not be returned. Because creating transactions requires their creators to specify and allocate the maximum amount of gas that they are willing to pay, the miners have the opportunity of detecting transactions that will take too long to validate without actually having to compute it.

2.11 Smart Contracts

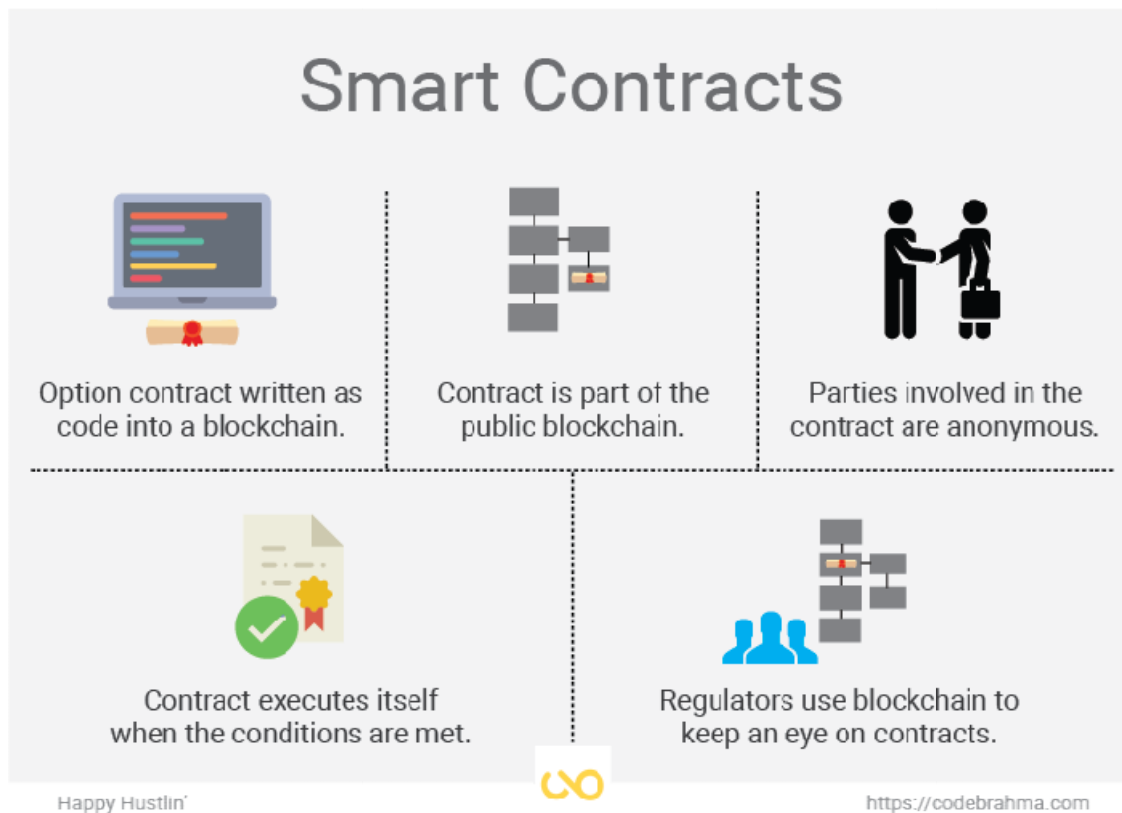


Figure 2.7 Source (“codebrahma”, 2018)

Contracts In addition to carrying out computations and transferring ether, it is also possible for transactions to create standalone contracts that are saved in the block chain and can store ether, data, and executable code, can communicate with other contracts and even create new ones in turn.

Basically contracts act as users, with the difference that they cannot initiate transactions (they

are reactive). This limitation was imposed in order to avoid DoS attacks against existing contracts. With this design, the gas required to execute the code triggered by a transaction must be initially paid by the user that issued the transaction in the first place. But contracts may programmatically choose to refund legitimate users so effectively.

2.11.1 Scripting

The code of Ethereum's smart-contracts is written in bytecode and executed into a virtual machine called EVM. EVM has a fixed word size of 32 bits and is untyped for simplicity.

2.11.2 Storage

Unlike Bitcoin, Ethereum's EVM provides a single stack limited to 1024 elements, but it provides two additional types of storage:

- Temporary storage (memory), which is a byte array and is deleted at the end of the execution of each transaction.
- Permanent storage, which is a word-indexed key-value dictionary, and is preserved on the blockchain between executions, but which can be deallocated explicitly.

2.11.3 Jump operations

Ethereum's EVM language provides both conditional and unconditional jump operations. In order to allow for easier and efficient implementations of JIT-compilers, these jump operations can only target parts of code marked as jump destinations.

2.11.4 Contract operations

Contracts are virtual entities that reside in the blockchain and can store ether and byte code, can send and receive messages and ether, and create other contracts. Unlike external accounts managed by users, contracts cannot initiate transactions: they are reactive.

Creation New contracts can be directly created by users or by other contracts through the CREATE byte code operation.

Calls Contracts can send messages to other contracts or accounts through the use of the call operations. These allow sending ether, to execute the code of another contract, to stipulate a maximum amount of gas for the code executed by the call (which may be smaller than the gas available at the time of the call), and can pass and receive information. Contracts in Ethereum have a single block of byte code that is executed by calls, but high level languages like Solidity automatically define a function selector at the beginning of the block that

redirects calls to the appropriate part of the byte code.

Suicide In order to save storage space, Ethereum allows contracts to delete themselves when they are no longer necessary. In order to promote this, part of the cost of creating contracts is refunded when the suicide operation is called.

2.11.5 Inspection operations

Ethereum's EVM allows contracts to access several kinds of meta information about the blockchain, about the contracts themselves, and even about the code of other contracts, which can be copied to memory.

2.11.6 Logging operations

Another functionality provided by the EVM is logging. There is a set of byte codes that allow smart-contracts to log values. These logs are returned as the "receipt" that results from processing the transaction, but they are not explicitly stored in the blockchain.

2.11.7 Ethereum implementations

Ethereum comes with three different officially maintained implementations written in C++, Go and Python:

- **Eth:** The C++ implementation; it is faster than the other implementations and it also is the basis for the Mix IDE contract development toolkit. It comes with a number of network analysis tools like Alethzero and it is suggested as a development platform for Internet of Things projects. It is also the only client that supports GPU mining .
- **Geth:** The Golang implementation; Geth is basis for the future Mist web browser. And proper development platform for Web-based applications.
- **Pyethapp:** The Python implementation; Pyethapp is indented mostly for educational purposes. To understand the inner workings of Ethereum to contribute to its expansion are invited to use Pyethapp due to its code readability and clarity.

2.11.8 Ether

Ether is a type of crypto currency created to serve as the 'fuel' of the Ethereum network. Its purpose is to be used to pay the network nodes for the amount of computational resources

they provide in order to secure the blockchain or execute the contracts. Ether can be obtained either by participating in the mining process, where each mined block is rewarded with 5 Ether, or by purchasing it from a third party.

Ether has a number of denominations the smallest of which is called 'Wei' which equals to 10^{-18} Ether. Bellow you can find a list of the Ether denominations as they relate to Wei:

Unit	Wei Value	Weis
Wei	1 Wei	1
Babbage	1e3 Wei	1,000
Lovelace	1e6 Wei	1,000,000
Shannon	1e9 Wei	1,000,000,000
Szabo	1e12 Wei	1,000,000,000,000
Finney	1e15 Wei	1,000,000,000,000,000
Ether	1e18 Wei	1,000,000,000,000,000,000

Table 2.11.1 Source (Imran B, 2018)

It is obvious that the use of all the above names is impractical, thus anything other than Ether and Wei is rarely used.

2.11.9 Solidity

Solidity is one of the high level programming language used to describe smart contracts. Solidity is the language that is officially maintained by the Ethereum project and suggested in the guides as the main contract language. Solidity is Object Oriented and it resembles JavaScript. Below we will present some of its main features.

Solidity supports a number of different data types. Like any traditional language it supports Booleans, integers and strings. There is no support for floating point variables as of yet.

A very interesting data type is that of an Ethereum address. It holds the 20 byte representation of an Ethereum account address, be it an external account or a contract. The address data type has internal predefined members to check the balance of an account or transfer Ether via a contract, as well as to call functions from other contracts.

Solidity also supports structs and enumerations as well as byte arrays that can hold data of any type. Moreover, solidity support mappings which are in essence key-value stores that map keys of any data type to values of any data type as well. An accessor function is created for each declared variable.

Events are the way Solidity provides in order for accessing the transaction logs, a special data structure in the Blockchain. Functions can emit these events populated with return values, and event messages will be broadcast and stored on the blockchain. Each application has to specify certain JavaScript callback functions that listen for these event messages and print them to the user interface. Event messages are not accessible from within contracts, not even the contracts that have created them.

Solidity distinguishes two types of functions, constant and transactional. Constant functions are those functions whose purpose is to return a value and cannot update the state of the contract. They can be called directly and do not consume gas since they do not modify the blockchain. Transactional functions are used to modify the state of the contract. Whenever called an amount of gas has to be supplied to cover the transactional costs. Constant functions need to be declared as such by using the keyword 'constant'.

There are four levels of visibility for Solidity functions. These are:

- External: External functions are part of the contract specification and they can be called by other contracts, but they are not accessible by the contract itself.
- Public: Public functions can be called by the contract itself or by any external contract or entity.
- Internal: Internal functions can only be accessed by the contract itself and its derivative (inherited) contracts.
- Private: Private functions are visible only to the contract itself and cannot be called by any external entity or derivative contract.

Function Modifiers are constructs used to change the behavior of a function. They are mainly used to check if a condition is satisfied before a function can be executed. Modifiers are inheritable properties of functions and each function can belong to multiple modifiers. (“readthedocs”, 2017)

2.11.10 Mining

Ethereum much like most blockchain technologies uses a mining process to secure the network. The process involves the production of blocks whose validity will be verified by the network. Just as the Bitcoin protocol, a block is only valid if it contains a Proof of Work of a certain difficulty.

The PoW algorithm used by Ethereum is called Ethash and it involves finding a nonce input to the algorithm so that the result is below a certain threshold depending on the difficulty. The mining difficulty is automatically adjusted so that a block is produced every 12 seconds. Ethash is designed to be memory hard in order to be impossible to implement on ASIC(Application-Specific Integrated Circuit) machines. This is because in order for the PoW to be calculated, a subset of a fixed resource depending on the block header and the nonce. This resource, a Directed Acyclic Graph (DAG), is 1GB in size and it is different every 30000 blocks. With 12s block time 30000 blocks amount to a 100 hours period, called an 'epoch'. This DAG depends solely on the height of the blockchain and it can be, thus, pre-generated. Its generation takes a few minutes and no block can be produced before the DAG is generated. However, the full graph is not needed for block verification since the appropriate subsets can be calculated from a 16MB cache, requiring low CPU power and memory.

The miners receive a reward for the computational effort they invest on the network. The amount they receive includes a static reward of 5 Ether for the discovered block and all the gas expended within the block, i.e. the gas consumed by all the transactions in that block. The static reward is issues as a transaction from the miners themselves while the transaction costs are paid by each transaction sender. It is expected that as the network grows, the transaction gas within each block will surpass the value of the static rewards and will be the main incentive for mining.

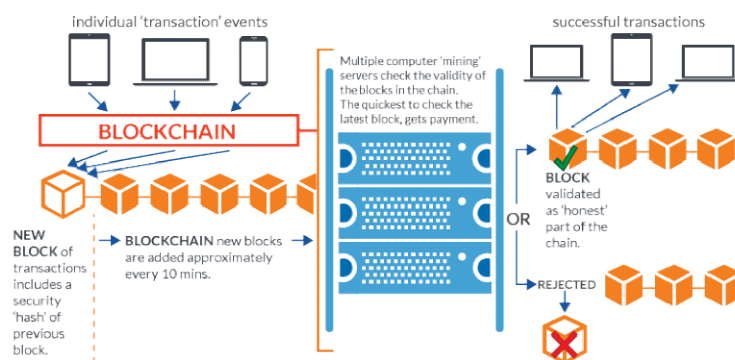


Figure 2.11.1

Interestingly enough, Ethereum also rewards blocks in a forked chain called 'Uncles'. An uncle is a block that is an ancestor of the current block but it is not part of the main (longest)

chain. For example if the current block is number 30 its parent is block 29 whose parent is block 28. If there is another block that points to block 28, that block is an uncle of block 30. A block is considered an uncle only if it is up to 6 blocks back from the current block. The reward for a valid uncle is 7/8 of the static block rewards. A maximum of 2 uncles per block is allowed.

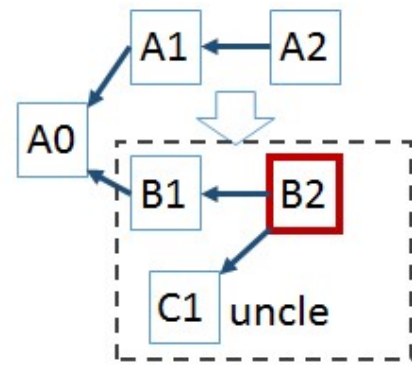


Figure 2.11.2

Ethereum chooses to reward uncles in order to neutralize the effect of mining rewards being collected by one central institution, due to network lag. Moreover, this increases security by including more work in the Blockchain.

At the moment, the mining difficulty on the live Ethereum network is such that only GPU hardware can perform this action. CPU mining is possible but the probability that any real Ether can be mined that way is minimal. (“bitdegree”, 2018)

Chapter 3

Design and Methodology

3.1 Methodology

The proposed digital cheque clearing framework is motivated by real financial sector demands and designed according to existing cheque clearing process. The framework consists of a private blockchain network comprising participating banks, cheque book application and the bank account integration. Each bank is having a full. Participation in the distributed ledger in banks node is based on the customers involved in the execution of the corresponding cheque transaction. The extranet ledger is open to all legitimate participants and contains tracking information for all cheque transactions. Private ledgers with cheque transactions are distributed and each participating node keeps a copy of the ledger of interest. This chapter describes the data model used for the digital cheque clearing mechanism.

Focus is to develop a secure architecture to achieve digital cheque issuance and clearing process on blockchain technology, so that banks and financial institutions could easily integrate and adapt to the digital form of cheque in parallel to paper based system.

Methodology is focus in to using blockchain to create digital cheques, which are tamper free and only a person with the correct cheque number and cheque digital signature can unlock and cash out the cheque. Cheque processing instruction is built in to smart contracts, so that such instructions are immutable and guarantee the integrity.

Digital-Cheque can be generated by drawers and sent to drawee. Drawer's bank would take charge of drawer's account and generate authorized software based Digital-Cheque book for drawers. Here the drawers account created on blockchain is linked to drawer's bank account with public key to account mapping. Drawee's bank would help drawee to deposit Digital-Cheque into his/her account. Any bank in the closed blockchain network could validate the transaction with its blockchain interface. In order to finish the complete transaction, Drawer's bank and Drawee's bank and blockchain application need to interact with each other.

Cheque is defined as “an order to a bank to pay a stated sum from the drawer's account, written on a specially printed form.” (“Oxford Dictionary”, 2018). This is a contractual agreement between drawer and the bank to pay to a third party. Therefore, selected blockchain technology should have the capability to implement such kind of contracts in to the blockchain to simulate cheque clearing operation.

Ethereum was the first public blockchain to allow the programming of Smart Contracts (programs using Turing-complete languages) in the block chain itself. This was a

revolutionary development, as it allows the creation of algorithms or adding logic to the transfers.

Two other solutions have been developed based on Ethereum and these add features to Ethereum to complement it in its applications. They are Monax and Quorum. There are ways of creating private Ethereum networks where access to only certain nodes is allowed, but this means the network will have fewer mining nodes, validators that will look for the hash to validate the blocks. Therefore the Ethereum technology has been selected for the proof of concept implementation of cheque clearing system.

3.1.1 Cheque book creation

Drawers software application generates a chequebook through his/her bank interface by submitting the public key of the blockchain account created before and associate the cheque book application to the bank account, in return, bank sends a new serial number range to used as the cheque numbers. Bank any time can finds the bank account association to blockchain account using the login credentials of the drawers application and finding mapping in between public key and the bank account. Public key act as blockchain wallet identification number as well. Drawers application provided by the drawers bank can create the appearance of the digital cheque very similar to the appearance of the paper cheque with a QR code to insert essential transaction information. When a cheque is issued, it can be delivered to the drawee by email or some other electronic means, so that he/she can present it to the drawee's bank for the cheque realization.

If the drawer issue a post dated cheque, transaction will remain in the cheque book application until the date is reached to post to the blockchain. Meanwhile drawer can make sure find is available in the blockchain wallet account to honour such transactions.

3.1.2 Block Chain Extranet and Transaction Realization

Blockchain extranet is created on Ethereum platform, so that cheque transaction routing logic can be built in the immutable smart contract. Each bank affiliated to the blockchain extranet has its own wallet account to handle interbank transactions. When a cheque is issued, it is send to the central smart contract, which is handling the cheque clearing logic. It will validate the date of the check, fund availability and the bank code and if the cheque date is matched with system date, then the cheque transaction is routed to the drawee's bank blockchain account. When a drawee presents the cheque image to the drawee's bank, bank can read the information from QR code and very transaction details with the transactions in its blockchain

wallet account. This verification is very fast and Drawee either can cash it or deposit the amount to normal bank account.

3.1.3 Digital Signature:

Most of the issues of present paper based cheque system are due the difficulty of verifying authenticity of drawer. All the control points and expensive hardware is in place to makesure signature of the drawer and cheque routing information are visually readable and MICR friendly.

Therefore, digital signature is used to establish the authenticity of the drawer of the cheque. This mechanism is built in the blockchain technology and each and every transaction is digitally signed by the drawer and encrypted using public key of the recipient.

Using Ethereum built on blockchain technology any user can have access global platform that provides a set of features, they could solve the weaknesses in cheque authentication:

- Drawer authentication is verified by the use of cryptographic signatures.
- Payment system logic can be setup on Ethereum EVM quickly with no third party reliance.
- DDoS resistance. Each application on Ethereum is executed on each and every node on the system. As long as there is one node maintaining the blockchain the application will run perpetually and will be able to be interfaced by any joining node.
- Limitless interoperability. Each Ethereum contract can seamlessly interact with any other contract instance via the provided interfaces in the Ethereum network.
- Before Ethereum is completely built on top of a Peer-to-Peer network with no central server infrastructure involved. Thus, the deployment of an application on the blockchain does not require the setup and the costs of setting and maintaining servers.

Unlike digital currencies such as bitcoin, it is required to make connection between bank current account and the account create on the Ethereum eco system to parallely run the digital cheque system with present paper based one. Smart contracts can be used to establish such connections with outside parties.

3.1.4 Accounts on Ethereum

In Ethereum any entity that holds an internal state is associated with an account, i.e. a private/public key pair. The public key is also considered the address of the account.

Ethereum distinguishes two types of accounts, Externally owned accounts and contracts. An externally owned account is a personal account controlled by a private key and that is used as the cheque book. The owner of the private key can send cheque to other external accounts. A contract is basically an account that holds logic of controlling cheque transactions mapped into the code that controls it.

Proof of concept needs an Ethereum test network to build and test the logic.

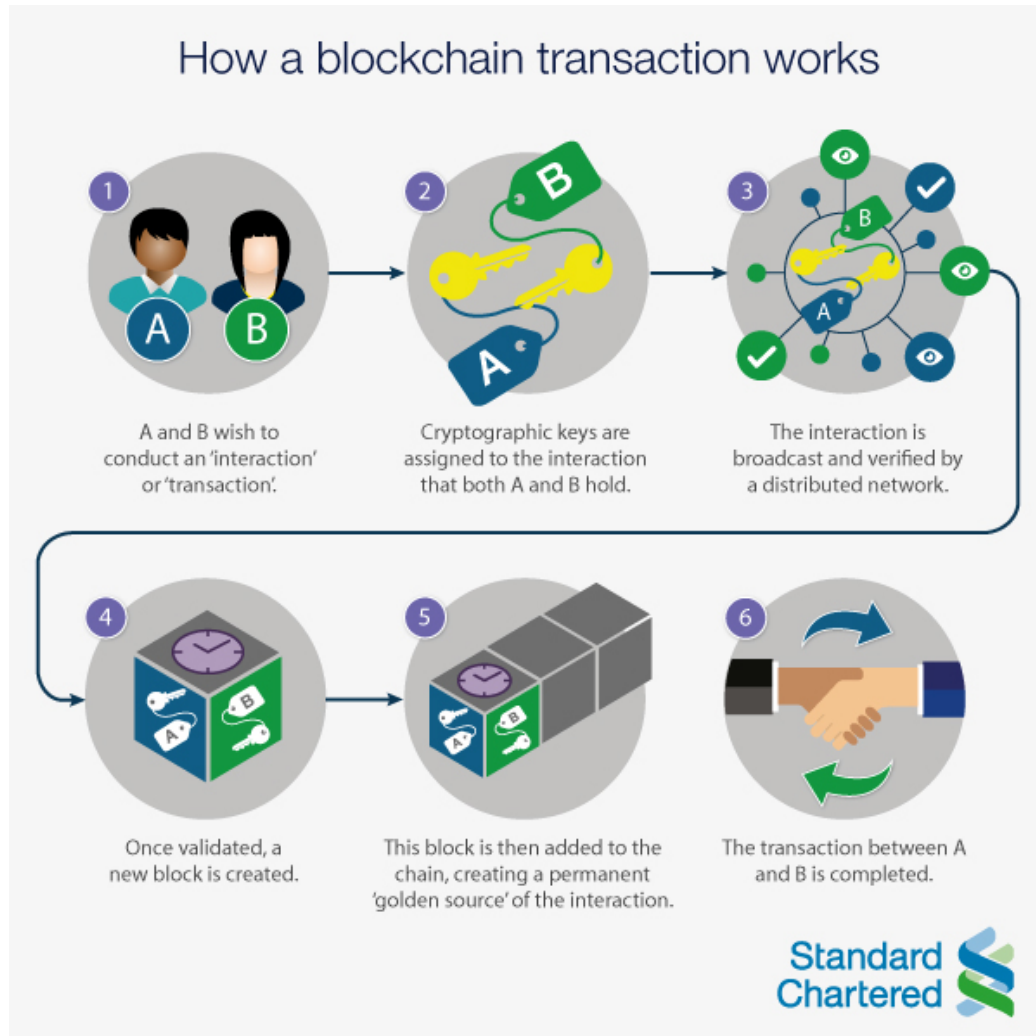


Figure 3.1.1 Source (John C, 2018)

3.1.5 Building a test network

On a private network the initial block difficulty can be adjusted in such a way that CPU mining can produce blocks in small time. Source code of the Ethereum can be edited to remove the mining difficulty algorithm from it.

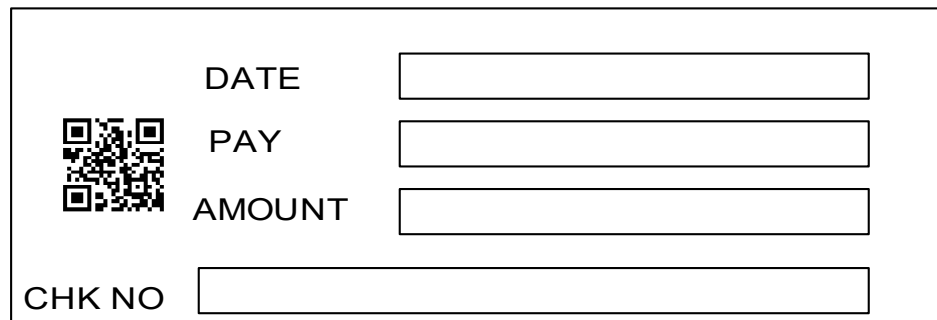
Bank function is implemented on the simple Mysql database to map customer public key with account information and store cheque serial numbers and image information.

3.2 Design

3.2.1 Architecture Diagram

Ethereum BlockChain technology is used to store and validate cheques. The cheque validation is built entirely upon Ethereum's smart contract, which built on top of the blockchain technology. This contract runs on decentralized private network, so no one, even the miners, can tamper with its content. Only a person with the correct cheque number and signature can unlock and cash out a cheque.

Appearance of the cheque



The diagram shows a digital cheque form enclosed in a rectangular border. On the left side, there is a square QR code. To the right of the QR code, the form contains four input fields. The first field is labeled 'DATE' and is a horizontal rectangle. The second field is labeled 'PAY' and is a horizontal rectangle. The third field is labeled 'AMOUNT' and is a horizontal rectangle. The fourth field is labeled 'CHK NO' and is a horizontal rectangle. The labels are positioned to the left of their respective input fields.

Figure 3.2.1

Appearance of the cheque is preserved with the branding of the bank, where end customer belongs. Cheque is generated with a client application, which has already generated key pair using Ethereum to sign the cheque. Cheque number format is preserved as of the cheque number format of the respective bank.

When the initial key pair is generated, public key is send to the client's bank to register it against the cheque account number. So that anytime account verification could be done by the bank for any realization of cheque of its customers.

QR Code includes the public key, date, amount, cheque number and payee information. Since cheque book accounts are mapped to the core-banking system account with the public key (wallet ID), bank can verify the status of the cheque drawer.

3.2.2 Ethereum Private network for cheque clearing system

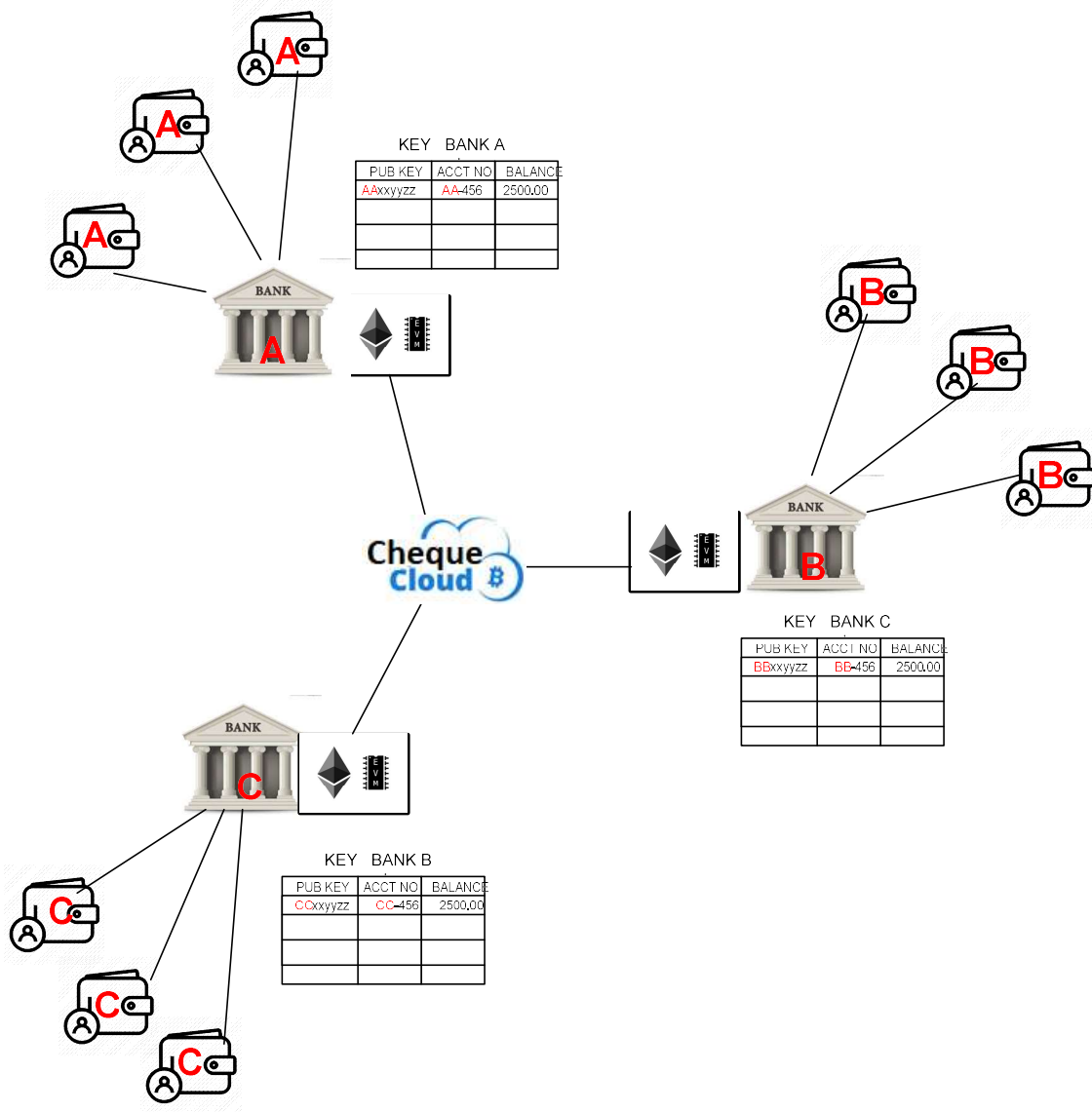


Figure 3.2.2

Ethereum private network is built among the participating banks. Each bank is having online banking interface to request cheque books for its clients. Initially, client is registered to the Ethereum network with account creation and then the wallet-ID (public Key) is passed to the bank to map it with the account number of the customer. In this way, de-centralized Ethereum network and the bank systems has correlation for account reconciliation purposes. Check recipient doesn't need an account in the Ethereum network to receive a digital cheque and when they want to realize a cheque, it should be submitted to online dropbox, which is attached to a central smart contract. This central smart contract gets input from the drawee and then redirect the transaction to the drawee's bank Ethereum account which sending account

credit information to the core-banking system.

3.2.3 Transaction flow:

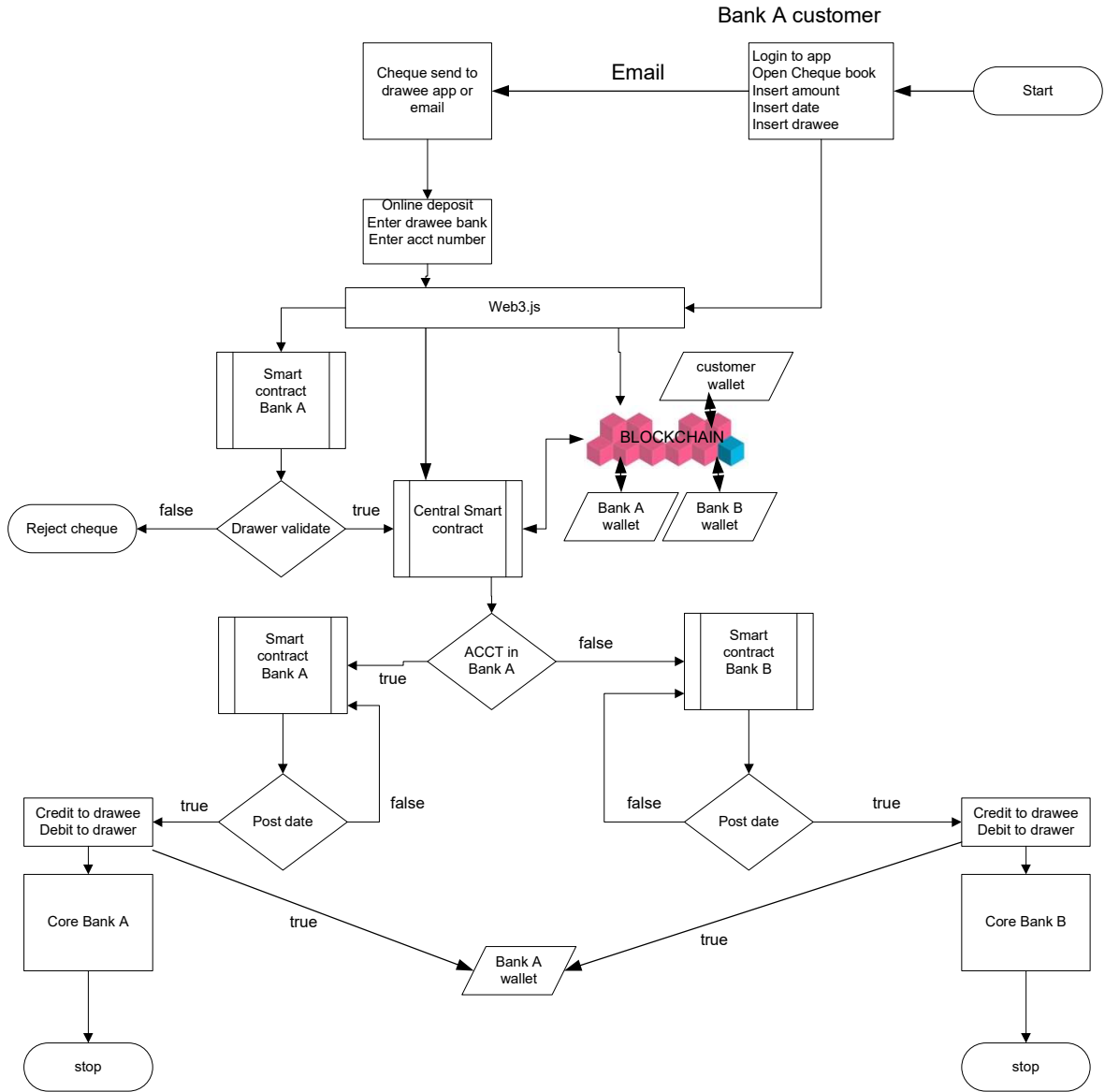
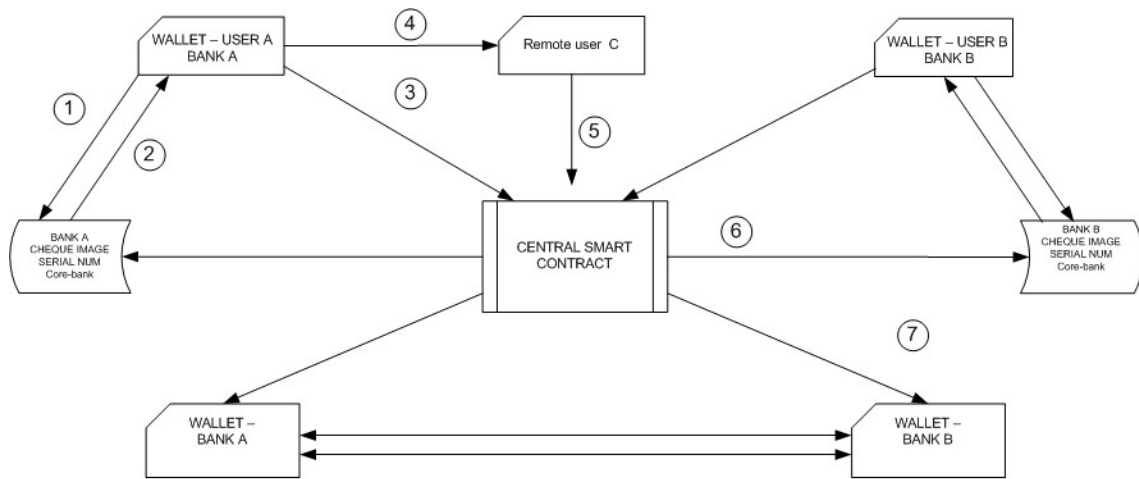


Figure 3.2.3

Above schematic diagram (Figure 3.2.3) show the digital cheque transaction has done by a bank A customer.



- ① Login to bank interface and submit pub-key
- ② Retrieve cheque information and image
- ③ Write a cheque and submit to central smart contract
- ④ Write a cheque and submit to payee via electronic means
- ⑤ Go to web based cheque box attached to central contract and submit cheque image with QR code. Interface request payee bank information with account number
- ⑥ Send transaction to core-bank system of the selected bank with account information
- ⑦ Send transaction to wallet of the selected bank

Figure 3.2.4

Following table 3.2.3 shows the different set of customers with account status to test cheque transaction.

BANK A			Core-bank system				Cheque Creating system Bank A	
Name	ID-num	Address	acct-num	balance	active	branch	acct-num	public-key/hash
siripala	551493204	godagama	1234567890	10000	yes	1	1.235E+09	
soma	652343255	colombo	1134567890	15000	yes	2	1.135E+09	
nadun	801553145	moratuwa	1123567890	25000	no	3	1.124E+09	
BANK B			Core-bank system				Cheque Creating system Bank B	
Name	ID-num	Address	acct-num	balance	active	branch	acct-num	public-key/hash
gunapala	551493205	mahagam	2234567890	20000	yes	4	2.235E+09	
siriya	652343256	nugegoda	2134567890	25000	yes	5	2.135E+09	
dasun	801553146	pitigala	2123567890	15000	no	6	2.124E+09	
Siripala - Bank A			Scenarios	drawer	drawee	amount	Actions	Cheque present to Bank A
Pvt Key	Pub Key	Cheque book app	1	Siripala	Soma	2000	validate drawer check acct status debit drawer credit drawee	send to smart contract check for bank code
Soma - Bank A								
Pvt Key	Pub Key	Cheque book app						
nadun - Bank A								
Pvt Key	Pub Key	Cheque book app	2	siripala	nadun	1000	validate drawer check acct status debit drawer credit drawee	Cheque present to Bank A send to smart contract check for bank code
gunapala - Bank B								
Pvt Key	Pub Key	Cheque book app						
siriya - Bank B								
Pvt Key	Pub Key	Cheque book app	3	gunapala	soma	3000	validate drawer check acct status debit drawer credit drawee	Cheque present to Bank A Inter bank fund transfer send to smart contract check for bank code

Table 3.2.3

To execute the proof of concept, initially all the bank central cheque accounts are loaded with fixed amount of cash so that interbank transaction will bring some equilibrium within the eco system.

3.2.4 Basic component integration of Ethereum implementation.

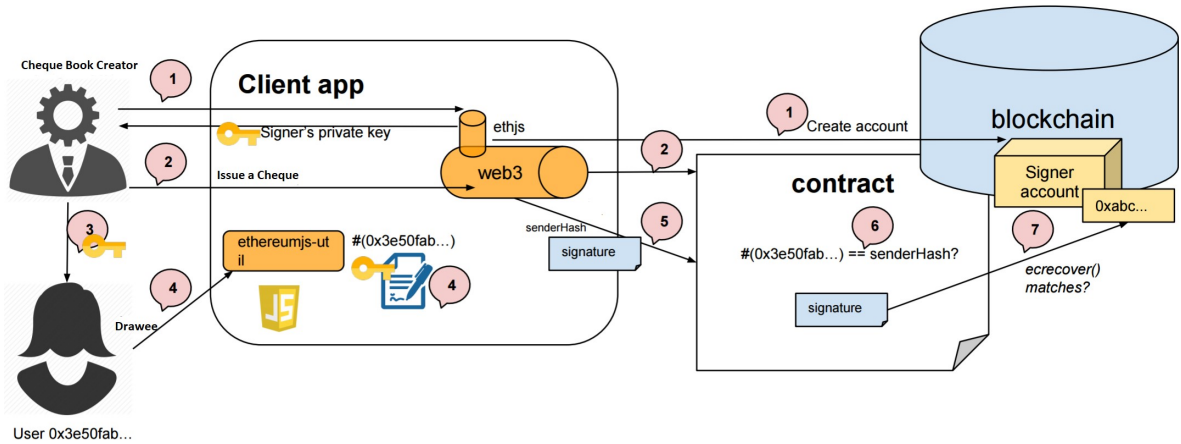


Figure 3.2.4

Above figure 3.2.4 shows the basic integration components of the test system.

Chapter 4

Implementation

There are three major components in this system: Ethereum Private Network, customer application and Core Banking interface.

4.1 Ethereum Single Node system Private Network

Running a private network is requiring lesser resources and it is ideal platform for-the testing applications.

“An Ethereum network is a private network if the nodes are not connected to the main network nodes. In this context private only means reserved or isolated, rather than protected or secure.” Ganache CLI is installed on Ubuntu Linux server as the personal blockchain for Ethereum development.

Ganache CLI uses ethereumjs to simulate full client behavior and It also includes all popular RPC functions and features (like events) and can be run deterministically to make development fast. Former TestRPC is now come under the Ganache-CLI.

4.2 Installation

ganache-cli is written in Javascript and distributed as a Node package via npm. Therefore before installing the application, following is installed.

```
#Apt install npm  
#npm install web3  
#npm install abi-decoder  
#npm install -g ganache-cli
```

Following command creates 3 accounts with 1000 Ether load on to each account and start the network on network ID 15. This avoids mixing with live Ethereum network.

```
root@ubuntu:~/ucscnet/data/node_modules# ganache-cli -a 3 -e 1000 -i 15  
Ganache CLI v6.1.0 (ganache-core: 2.1.0)
```

Available Accounts

-
- (0) 0x5e1faf4984821d199794e2b7acf6f8373eccdf86
 - (1) 0x45f376313359ac676bb00da98febc0e37a3eaa43
 - (2) 0xd7e321073ebc4d048ab6fc1f3b640d699087eb11

Private Keys

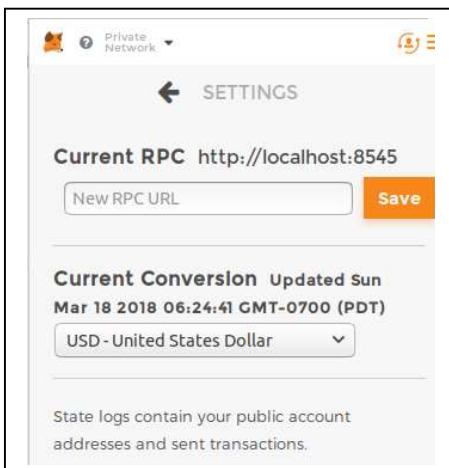
-
- (0) b1991d882f6a0da505571ed4e828bc9d42e8151e9b830247ffb818ddf2b3393b
 - (1) b319125bbb1fa078a2d6e07c8206d74bbf63cb9a183ae2fde35f081e4489b033
 - (2) 70861e1726d4b3ae01b9a64299d5cc22b0b442d3fda1eefc65ac0f591123e32e

HD Wallet

Mnemonic: resemble fancy scissors rather ostrich rude render universe coral pudding
crucial powder

Base HD Path: m/44'/60'/0'/0/{account_index}

Listening on localhost:8545



Metamask is used as the client to access and test the transaction with ganache-CLI

Figure 4.2.1

Those three accounts, which created using Ganache-CLI, are accessed via the Meta Mask client as follows.



Figure 4.2.2

Both the command line interface and the graphical interface were used for Smart contract development and testing (Ethereum Remix environment / Geth)

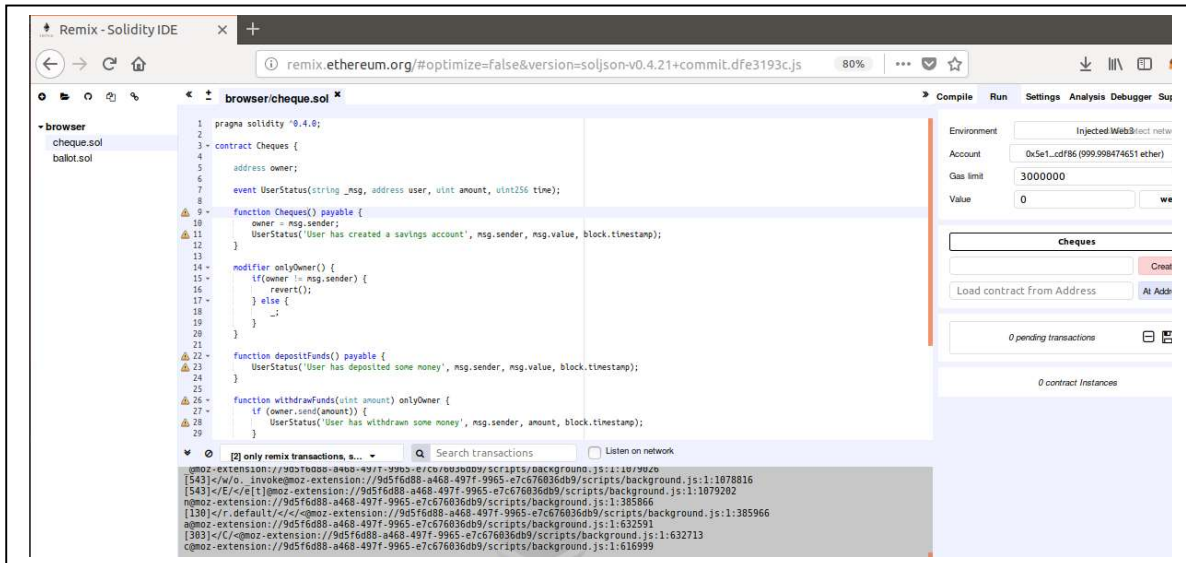


Figure 4.2.3

4.3 Signature creating and validation in Ethereum

Verify user signature Each transaction message carries the signature of its data. Once receives the transaction each server node verifies the signature and continues to other steps if and only if signature is successfully verified. If it finds that the signature is invalid, send an error response to payer and winds up the transaction processing task.

Solidity provides a globally available method ecrecover that returns an address. If the return address is the same as the signer, then the signature is real.

Pragma
solidity^0.4.8;

```
contract Verifier {
    function recoverAddr(bytes32 msgHash, uint8 v, bytes32 r, bytes32 s) returns
(address) {
        return ecrecover(msgHash, v, r, s);
    }

    function isSigned(address _addr, bytes32 msgHash, uint8 v, bytes32 r, bytes32
s) returns (bool) {
        return ecrecover(msgHash, v, r, s) == _addr;
    }
}
```

4.3.1 Web3 Interface (Signature Creation)

```
const Web3 =
require('web3')

const provider = new Web3.providers.HttpProvider('http://localhost:8545')
const web3 = new Web3(provider)
function toHex(str) {
    var hex = ''
    for(var i=0;i<str.length;i++) {
        hex += ''+str.charCodeAt(i).toString(16)
    }
    return hex
}
let addr = web3.eth.accounts[0]
let msg = 'Message to be Signed'
let signature = web3.eth.sign(addr, '0x' + toHex(msg))
console.log(signature)
```

4.3.2 Signature Verification

```
const SignVerifyArtifact = require('./contracts/SignAndVerifyExample')
const SignVerify = contract(SignVerifyArtifact)
SignVerify.setProvider(provider)
//...
SignVerify
    .deployed()
    .then(instance => {
        let fixed_msg = `\x19Ethereum Signed Message:\n${msg.length}${msg}`
        let fixed_msg_sha = web3.sha3(fixed_msg)
        return instance.verify.call(
            fixed_msg_sha,
            v_decimal,
```

```

        r,
        s
    )
})
.then(data => {
    console.log('-----data-----')
    console.log(`input addr ==> ${addr}`)
    console.log(`output addr => ${data}`)
})
.catch(e => {
    console.error(e)
})

```

4.3.3 Solidity smart contract for transaction routing between banks

```
pragma solidity ^0.4.18;
```

```

contract chequearraynew {
    address[100] public drawers;
    uint[100] public chequeval;
    address public checkdr;
    struct chequedata {
        uint value;
        uint bankcode;
        uint time;
    }
    uint public count = 0;
    address public dfcc=0x8e5373fed970410254259427daad01b9c27b27b0;
    address public sampath=0x3dbe54d1b6b9c3b20a2d44f42aac7240905f7008;
    address public boc=0xbe772543451ebd3f6241b735389062ba06e8ca16;

    // mapping (address => chequedata) alldata;
    // address[] public draweracct;

    function storedrawer() payable public returns (address) {

        drawers[count] = msg.sender;
        chequeval[count]=msg.value;
    }
}

```

```

// return drawers[count];
    count++;
}

function checkDrawerExists(address checkdrawer, uint chequevalue) public constant
returns(bool){
    for(uint256 i = 0; i < drawers.length; i++){
        if(drawers[i] == checkdrawer && chequeval[i]==chequevalue) return (true);
    }
    return false;
}

function checkAmount(uint storevalue) public constant returns(bool){
    for(uint256 i = 0; i < drawers.length; i++){
        if(drawers[i] == checkdrawer) return (true);
        bankSend(checkdrawer)
    }
    return false;
}

function getdrawer(address checkdr, uint bankcode, uint amount) payable public
returns(address, address,uint) {
    return (alldata[_address].value, alldata[_address].bankcode, alldata[_address].time);

    if(bankcode==1){
        bankSend(dfcc,checkdr,amount);
        return(dfcc,checkdr, amount);
    }
    else if(bankcode==2){
        bankSend(sampath, checkdr);
        return(sampath,checkdr,amount);
    }

    else
        if(bankcode==3){

```

```

    bankSend(boc, checkdr);
    return(boc,checkdr,amount);
}
// return (checkdrawer);
}
function bankSend(address receiver, address checkdraw, uint amt) payable public
returns(address,address,uint)
{
    require(checkDrawerExists(checkdraw,amt));
    for(uint256 i = 0; i < drawers.length; i++){
        if(chequeval[i] == amt) receiver.transfer(amt);
        bankSend(checkdrawer)
    }
    receiver.transfer(amt);
    remove(checkdraw,amt);
    return(receiver,checkdraw,amt)
}
function remove(address chequenum,uint cheval) returns(address) {
    for(uint256 i = 0; i < drawers.length; i++){
        if(drawers[i] == chequenum && chequeval[i]==cheval)
            drawers[i] = 0;
    }
}
}

```

4.4. Client Side

In this system client side is implemented using Python on Linux and available Ethereum wallet clients are used to view transactions.

Chapter 5

Results and Evaluation

Initially, three external accounts and three smart contract accounts were created on the Ethereum private blockchain.

Two accounts were named as Bank A and Bank B and the other account is used as cheque client user. Two smart contracts were named as Bank-SC-A and Bank-SC-B and third contract is used as central contract, where third party drawees interact to deposit cheques.

```
athula@ubuntu:~/chequesrv$ ganache-cli -a 5 -e 1000 -i 15
Ganache CLI v6.1.0 (ganache-core: 2.1.0)

Available Accounts
=====
(0) 0x8fca378df14b8f9bad9556098aad92fd011da9e0
(1) 0x25c8d71d0d254fae06acde365625df9c305cedd9
(2) 0x5c901856d7005e3b4f11d1378f45acd47b081fa3
(3) 0x2076d637265de9a3fbc18d802172cdcc80dd378
(4) 0xebea34d96d3f1ff6b3ff880b3cd26d55f33535f8

Private Keys
=====
(0) 6726166dbba5e04650ed09986ad1c2b6bfa9e79c517ac21f8f5ccaa99850ea
(1) 54c1a21ce7f1df10b2d9715805d197308b61ea4295fb35565ffbbd38ca780cae
(2) a0b2d4033c146eb12561d6aa6e7b79348d38571d5a4dce6a7f6fe92411db5b4d
(3) 78cb08430a5b0c99428de8533721b120a99571a83442997b96ef9f9cd01e589b
(4) 900a181a7ac0c214f9b0987841703ca68744eec4aa1fa02fc3eb56a6b4cff8c9

HD Wallet
=====
Mnemonic:      host hidden tissue because intact prepare kit topic quote frequ
nt address trust
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
[Show Applications]
```

Figure 5.0.1

Using Ganache CLI code developed by Ethereum was used to generate initial customer and bank wallet accounts. Ethereum Virtual Machine (EVM) is active and listening on port 8545 of a Ubuntu Linux Virtual environment.

To access the EVM and execute and deploy smart contracts, Ethereum Remix online environment and MetaMask browser plug-in were used. Remix online environment was configured to access the EVM open on port 8545 as illustrated in the above figure.

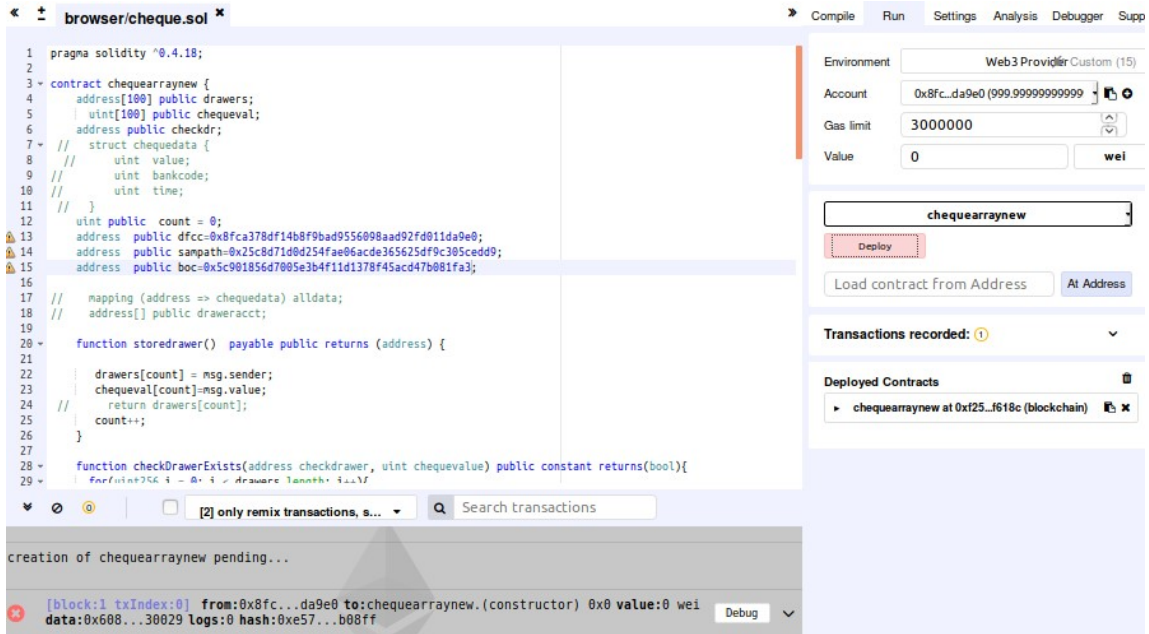


Figure 5.0.2

As shown in the above Figure 5.0.2, central smart contract was named as “chequearraynew” and deployed to the chain. Then the basic environment is ready to do the transaction to test transaction routing.

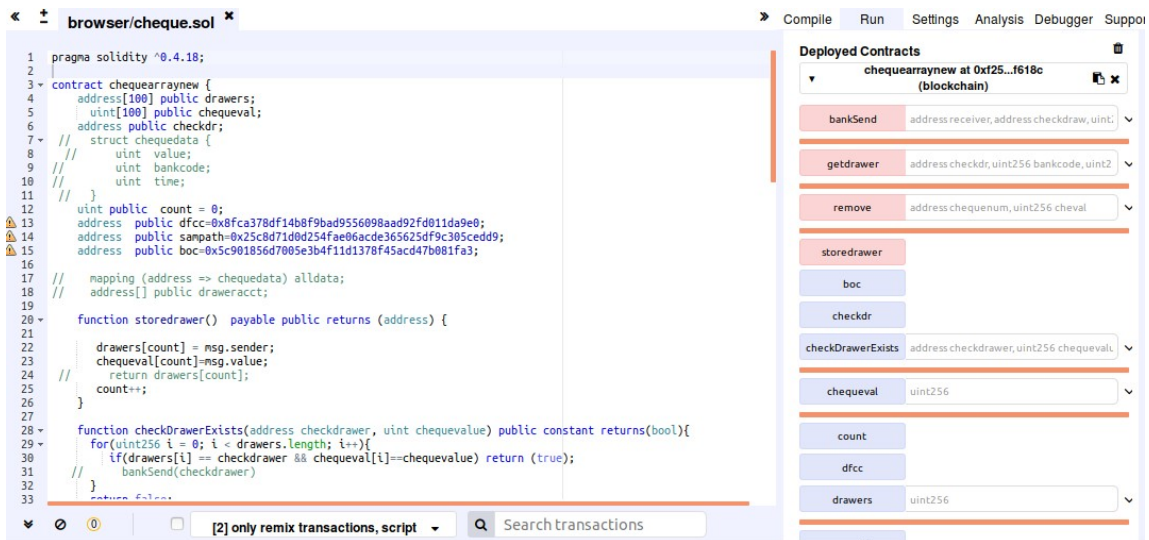


Figure 5.0.3

In the right hand side of the remix panel, all the getter and setters can be seen and each function can be separately tested to passing appropriate values to it. So, several tests were carried out to demonstrate the cheque issuance functionality and cheque realization functionality.

When the cheque book owner draws a cheque, it will store the transaction finally in the central smart contract. Below figure shows the storing of transaction in the array of central

smart contract.

```
transact to chequearraynew.storedrawer pending ...
```

[block:3 txIndex:0] from:0x207...dd378 to:chequearraynew.storedrawer() 0xf25...f618c
value:5000000000000000000 wei data:0x03d...2357b logs:0 hash:0x396...da2e8

status	0x01
transaction hash	0x396452235dfaab7153262e31077332c7345cbe6637610d8ef62ea2c050bda2e8
from	0x2076d637265de9a3fbcdd18d802172cdcc80dd378
to	chequearraynew.storedrawer() 0xf258a0d9050a78df7cbb78dd8e9e9510225f618c
gas	67440 gas
transaction cost	67440 gas
hash	0x396452235dfaab7153262e31077332c7345cbe6637610d8ef62ea2c050bda2e8
input	0x03d...2357b
decoded input	{}
decoded output	-
logs	[]
value	5000000000000000000 wei

Figure 5.0.4

The customer, who is having a cheque book referred to the Public Key(Wallet ID) of 0x207***378 has raised a cheque with a value of 5 Ether. That transaction is stored in the smart contract, until the Drawee withdraws it. Drawee get a cheque image with QR code, comprises of Drawer Wallet ID, amount and the timestamp.



Figure 5.0.5

Stored values in the smart contract can be queried using the remix environment as follows



Figure 5.0.6

Once the 5 Ether is transferred , it can be seen in the account status window of the Remix Environment.

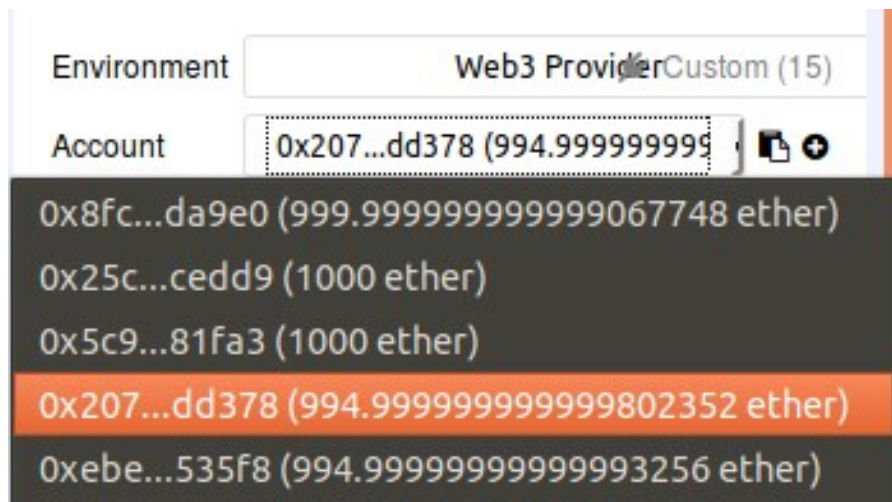


Figure 5.0.7

Stored value in the smart contract can be further reviewed using the transaction details Jason file in the Remix environment [Figure 5.0.8](#).



Figure 5.0.8

When the Drawee submits his cheque to withdraw money, QR code is read by the system and Drawee is asked to select the preferred bank to deposit the Cheque. When the Drawer Wallet ID, amount, timestamp and the bank code is sent to the central smart contract, it will authenticate and route the transaction the respective bank. Below [Figure 5.0.9](#) shows the transaction routed to the respective bank wallet ID.

The screenshot shows a transaction interface with the following details:

- status:** 0x1 Transaction mined and execution succeed
- transaction hash:** 0xcfaad2098cde4c8883703c25e2b7c6b73c7a930cde7eb3a87cd111bf6eb386c
- from:** 0xdd870fa1b7c4700f2bd7f44238821c26f7392148
- to:** chequearray.bankSend(address,address,uint256) 0xbbf289d846208c16edc8474705c748aff07732db
- gas:** 3000000 gas
- transaction cost:** 32295 gas
- execution cost:** 8015 gas
- hash:** 0xcfaad2098cde4c8883703c25e2b7c6b73c7a930cde7eb3a87cd111bf6eb386c
- input:** 0xdc8...000fa
- decoded input:**

```
{
  "address receiver": "0x14723A09ACff602A60cdF7aA4Af308FDDC160C",
  "address checkdraw": "0xd0870fa1b7c4700f2807f44238821c26f7392148",
  "uint256 amt": "250"
}
```
- decoded output:**

```
{
  "0": "address: 0x14723A09ACff602A60cdF7aA4Af308FDDC160C",
  "1": "address: 0xd0870fa1b7c4700f2807f44238821c26f7392148",
  "2": "uint256: 250"
}
```
- logs:** [1]
- value:** 0 wei

Figure 5.0.9

Recipient bank can authenticate the Drawee and Drawer to credit the transaction to core banking system of Drawee’s bank.

This is a basic prototyped tested to created a smart contract to simulate store and forward transactions to facilitate dated cheques as well. If both Drawee and drawer are having blockchain wallet ID, then the transaction can be executed directly to wallet to wallet.

To make this solution practically to work, mapping system with core banking and audit and tracking interface should be in place; however those components were taken as out of scope for this project.

Testing was carried out as mentioned in the [Table 3.2.1](#). However in the testing, existing applications were used as client application to demonstrate the basic functionality. Focus was mainly in to smart contract, which routes cheque transactions based on value, date and dreawee’s bank code.

5.1 Cheque book issuance

Customer login to the online bank interface and request a cheque book. He gets a cheque image and interact with Ethereum Private network to create his wallet, while send the public key back to the core banking application.

Core-bank app registers the public key against the account number of the cheque book client.

5.2 Issue a Cheque

When issue a check, drawer put details of drawee with Name, amount and date and sends to the drawers bank Smart Contract, which validate the drawer and forward the transaction to the central smart contract. Central contract functionality has been demonstrated in section 5.0.

Generated image embedded with the QR code is then sent to the Drawee to withdraw money from his/her respective bank.

5.3 Withdraw Money

Recipient of the cheque uses the Cheque Box application integrated to the central smart contract to submit the cheque image with QR code. In addition to the cheque submission, drawee has selected the bank to deposit the check. Central smart contract gets the transaction ID and other details from the QR code and the selected bank smart contract ID to forward the transaction to the Drawee's bank Ethereum wallet. At the same time transaction details are forwarded to the Drawees bank core system to update to credit the drawees account.

Chapter 6

Conclusion and Future Work

This chapter contains the general discussion of the study. The discussion, findings and the conclusion are briefly outlined in this chapter. It includes the limitations and further suggestions in order to improve the research.

6.1 General Discussion

Cryptographic innovations can be used to address various security issues related cheque payment system. Digital signing and data encryption are two key areas, which can be used to enhance the verification of authenticity of cheque transactions.

One of the main issues in digital distributed payment system is the possibility of double spending. However, using blockchain based protocols; this issue can be easily solved. Digital Validation of authenticity of the transaction generator is built in to the blockchain based Ethereum protocol and such features make sure the security and speedy processing of transaction within the Ethereum network.

Private Key is used to signing and protects the Ethereum wallet. These protocols have been used when the customer issue a cheque and send it to the smart contract of his own bank. Image copy, which is similar to the paper cheque, but with embedded QR code is send to the drawee of the cheque, so that drawee doesn't need an ethereum wallet account to clear the cheque. Once the customer withdraw money successfully it will be stored in the paying banks digital wallet as well as in the core-bank application.

Banks needs to maintain Wallet to core-system mapping ledger system for reconciliation and establishing monetary value mapping.

This system can be run in parallel to the current Cheque issuance and clearing system and since the Cheque recipient doesn't need a Wallet Application, there is a high chance of people embrace the technology for the ease of cheque operation.

6.2 Findings

Ethereum Blockchain application can be successfully used to build financial payment applications to provide security and integrity, anonymity and high availability. Smart contracts are very useful to run conditional codes with the blockchain eco system.

6.3 Limitations

There is no direct monetary value chain established between Ethereum wallet of the bank and its core banking system. They are two separate systems but only linked with transaction references.

Bouncing of cheque should be handled via a separate channel and it is not discussed in this thesis.

Bibliography

- [1] Dr. David Wood (2017 August, 07). ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf> [Accessed 26 Oct. 2017].
- [2] Nakamoto, S. 31 October 2008. "Bitcoin: A Peer-to-Peer Electronic Cash System". Also known as the Bitcoin whitepaper. <http://nakamotoinstitute.org/bitcoin/>. <http://bitcoin.org/bitcoin.pdf>. <https://github.com/saivann/bitcoinwhitepaper>. Accessed 25 July 2017.
- [3]. *Ethereum White Paper* [online] github.com. Available at: <https://github.com/ethereum/wiki/wiki/White-Paper> [Accessed 27 Oct. 2017].
- [4] M M. Anderson. 31 October 2008. " The Electronic Check Architecture[Online]. Available at: <http://echeck.org/files/ArchitectualOverview.pdf>[Accessed 26 Oct. 2017].
- [5] A. Lewis. 15 December 2015. " On KYC and blockchains [Online]. Available: <https://bitsonblocks.net/2015/12/10/on-kyc-and-blockchains/> [Accessed 28 Oct. 2017].
- [6] *Micropayments*, (2011) : A viable business model. [Online]. Available at: cs.stanford.edu [Accessed 28 Oct. 2017].
- [7] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better|how to make bitcoin a better currency. In *Financial Cryptography and Data Security*, pages 399{414. Springer, 2012.
- [8] Scott Driscoll. *How bitcoin works under the hood*, 2013.
- [9] Sunny King and Scott Nadal. *Ppcoin: peer-to-peer crypto-currency with proof-of-stake*, 2012.
- [10] Ronald L Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. In *Security Protocols*, pages 69{87. Springer, 1997.
- [12] *Bill of Exchange* [Online]: Available at :http://www.commonlii.org/lk/legis/consol_act/boe92191.pdf [Accessed 20 July 2018].
- [13] Milton M. Anderson September 29, 1998. [Online]: Available at: <http://echeck.org/files/ArchitectualOverview.pdf> [Accessed 20 July 2018].
- [14] *HongKong Clearing House* [Online]: Available at: http://www.hkma.gov.hk/media/eng/doc/key-functions/finanical-infrastructure/infrastructure/retail-payment-initiatives/e-Cheque_e-brochure_Plain_Text_eng.pdf[Accessed 21 july 2018].
- [15] *Turing Machine* [Online]: Available at: <https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html> [Accessed 22 July 2018]
- [16] *Proof of Work* [Online]: Available at: <https://blockgeeks.com/guides/proof-of-work-vs->

proof-of-stake/ [Accessed 23 July 2018]

[17] [Online]: Available at: <http://www.hitachi.com/New/cnews/month/2016/08/160822.pdf> [Accessed 23 July 2018]

[18] [Online]: Available at: <https://codebrahma.com/brief-intro-smart-contracts-endless-possibilities/> [Accessed 23 July 2018]

[19] *Ethereum Mining* [Online]: Available at: <http://site.ssct.edu.ph/send-miner/computer-requirements-for-bitcoin-ethereum-shannon/> [Accessed 23 July 2018]

[20] Imran, B. (2017). Mastering BlockChain. [online] Available at: <https://www.safaribooksonline.com/library/view/mastering-blockchain/9781787125445/ch07s07.html> [Accessed 23 July 2018] [21] <http://solidity.readthedocs.io/en/v0.4.21/> 2017

[22] Ray, K. (2018). Ethereum Mining. [online] Available at: <https://www.bitdegree.org/tutorials/etherem-mining/> [Accessed 23 July 2018]

[23] John, C. (2017). Fighting Financial Crimes. [online] Available at: https://www.sc.com/fightingfinancialcrime/av/SCB_Fighting_Financial_Crime_Deep_dive_Blockchain_August_2017.pdf [Accessed 24 July 2018]

[24] *Definition of Cheque*. [online] Available at: <https://en.oxforddictionaries.com/definition/cheque> [Accessed 24 July 2018]

Appendices

Appendix A Cheque Image generation in client software

```
from PIL import Image, ImageDraw, ImageFont
# get an image
#import qrcode
import pyqrcode
import sys
base = Image.open('cheque.jpg').convert('RGBA')

# make a blank image for the text, initialized to transparent text color
txt = Image.new('RGBA', base.size, (255,255,255,0))

# get a font
fnt = ImageFont.truetype('Pillow/Tests/fonts/FreeMono.ttf', 17)
print("=====")
print("Please enter cheque details")
print("=====")
date = input ("Date:")
amount = input("Amount:")
name = input("Payee Name:")
#awords= input("Amount in words:")
# get a drawing context
d = ImageDraw.Draw(txt)
#Date
d.text((480,37), str(date), font=fnt, fill=(0,0,0,255))
# ount
d.text((480,140), str(amount), font=fnt, fill=(0,0,0,255))

# payee
d.text((80,85), name, font=fnt, fill=(0,0,0,255))
# Amount
```

```

d.text((70,120), "hundrad thousand", font=fnt, fill=(0,0,0,255))
#cheque routing
d.text((70,300), ":123456789: 0750099: 01: 10000000:", font=fnt, fill=(0,0,0,255))
out = Image.alpha_composite(base, txt)
#outimg = out
out.save('bkgcheque.png')
#-----

big_code = pyqrcode.create('Athula Weerasinghe 10000
1q2w3e4r5t6y7u8i9o0pqawsedrftgyh', error='L', version=4, mode='binary')
big_code.png('code.png', scale=1, module_color=[0, 0, 0, 128], background=[0xff, 0xff,
0xcc])
#big_code.show()
##data = "Athula Weerasinghe|10000|1q2w3e4r5t6y7u8i9o0pqawsedrftgyh"

# Add data
##qr.add_data(data)
##qr.make(fit=True)

# Create an image from the QR Code instance
#img = qr.make_image()
bg_w, bg_h = out.size
#img_w, img_h = big_code.size
#offset = ((bg_w - 40)/2, (bg_h - 40)/2)
#out.paste(big_code, offset)
#-----
img = Image.open('code.png', 'r')
img_w, img_h = img.size
#background = Image.new('RGBA', (1440, 900), (255, 255, 255, 255))
background = Image.open('bkgcheque.png', 'r')
bg_w, bg_h = background.size
offset = ((bg_w - img_w) / 2 + 170, (bg_h - img_h) / 2 + 90)
background.paste(img, offset)
#background.save('out.png')
background.show()

```

```
#out.show()
```

```
Smart contract of Bank A,B
```

```
pragma solidity ^0.4.18;
```

```
/**
```

```
 * Contract that will forward any incoming cheque to its central SC
```

```
*/
```

```
contract Forwarder {
```

```
 // Address to which any funds sent to this contract will be forwarded
```

```
 address public destinationAddress;
```

```
 function Forwarder() public {
```

```
     destinationAddress = msg.sender;
```

```
 }
```

```
 function() payable public {
```

```
     destinationAddress.transfer(msg.value);
```

```
 }
```

```
 function flush() public {
```

```
     destinationAddress.transfer(this.balance);
```

```
 }
```

```
}
```