PARTIALLY CENTRALIZED SIMPLE PEER TO PEER DIGITAL CURRENCY

by
Erkan Erol

Submitted to the Department of Computer Engineering in partial fulfilment of the requirements for the degree of

Bachelor of Science
in
Computer Engineering

Boğaziçi University
April 2015

TABLE OF CONTENTS

Project Name:	Partially Centralized Simple Peer to Peer Digital Currency
Project Team:	Erkan Erol , 2010400129
Advisor:	Can Özturan
Term:	2014-2015/2
Keywords:	Digital Currency, Centralized, Peer to Peer, Simplified, Bitcoin,Cryptocurrency

Abstract

After the increase in the use of Bitcoin[1], digital curriencies are very popular as research topics nowadays. A lot of new digital currencies have been released for different needs. In this project, we propose a new digital currency that is a kind of simplified and partially centralized form of Bitcoin. The creation of coins are done in a well-known server and the initial users got their keys from the server. Then,users can transact the coins without permission of the server and everybody can create a new account and participate the usage of the coin. The server can create new coins whenever it needs. The architecture can be use for many different purpose such as ticket systems, documents of limited usage permissions etc.

Chapter 1

1.1 Introduction

The Internet has been changing the conventional methods in many fields. The Internet has an anarchic spirit that allows people to do whatever you want without limiting by an authority. Although many laws,rules,methods have been being developed to regulate it,another methods for getting rid of them have developed.

Cryptocurrencies are good example of these. They have all features of money and they don’t need an authority like a central bank. Bitcoin is the most common cryptocurrency and its value and usability has been increasing day by day. It inspired many researches. Our project is one of them.

In this project, we use the flexibility of the Bitcoin and add some new features to develop a new system such that there is a server and unlimited peers. The server can be able to create new coins for initial users whenever it needs and doesn’t need to control the transactions. People can be attend the system by creating new ids for themselves and purchase coins from initial users like in Bitcoin. Of course, it is against the Bitcoin spirit but we think it is usable for different purposes such as ticket system.

The Github repository of the project:

https://github.com/erkanerol92/P2PDigitalCurrency

Chapter 2

2.1 General Informations about Bitcoin Architectures

Bitcoin is a well-designed cryptocurrency that obeys hashing and signing standards and has own working principles that makes it most popular.

In Bitcoin, the history of transactions are stored in blocks and blocks are attached to others in block chain. The transactions have inputs and outputs. The inputs are links of outputs in the previous transactions. The transactions also contain signatures of users who has the ownership of the inputs. The ownership of the money is to own corresponding private keys to the public keys in the outputs. Briefly, the amount of money that a user own is the total of unspent transaction outputs in the block chain whose public ids are the user’s id.

The mining process is the innovation of Bitcoin[2]. Users in Bitcoin ecosystem forward all transactions in their transaction pools to the other peers. The transactions are gathered in blocks after validity checking. However, the system has a need a consensus and it must define a protocol about the consensus. The Bitcoin solves the problem such that the users have to generate block hash in a certain range and they can do this by consuming computation power. The process is called as mining and the miners gain new coins if they succeed being first miners who can generate the hash. It is an incentive to the users and it sets a consensus mechanism.

2.2 Definitions of Problems and Overview of Solutions

The architecture of the Bitcoin was designed such that every user has equal and all peers keep the all information about the system as shown in Figure 2.2.1. In classical banking systems, there is an authority that keeps the all informations and there are users who have to interact with the authority to do a operation as shown in Figure 2.2.2. We mixed the two architecture in this project.

The first problem is how to change the architecture of the system to adapt it to the new paradigm. At first, we create a server running on a well-known IPv4 and then put the information into the software. All peers will know the machine and act according to its rules. The new architecture is shown in Figure 2.2.3

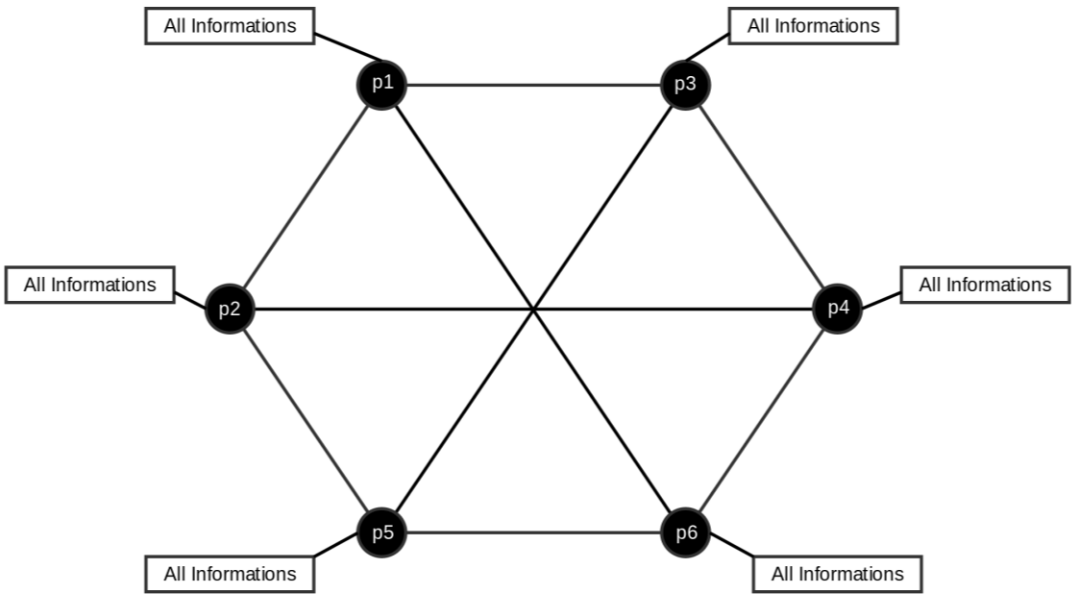

Figure 2.2.1 Bitcoin architecture

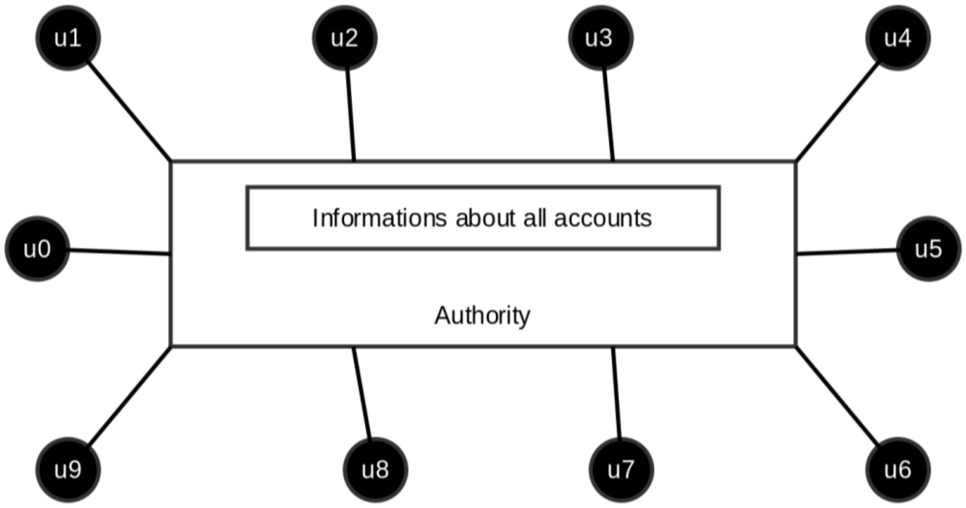

Figure 2.2.2 Classical banking system architecture

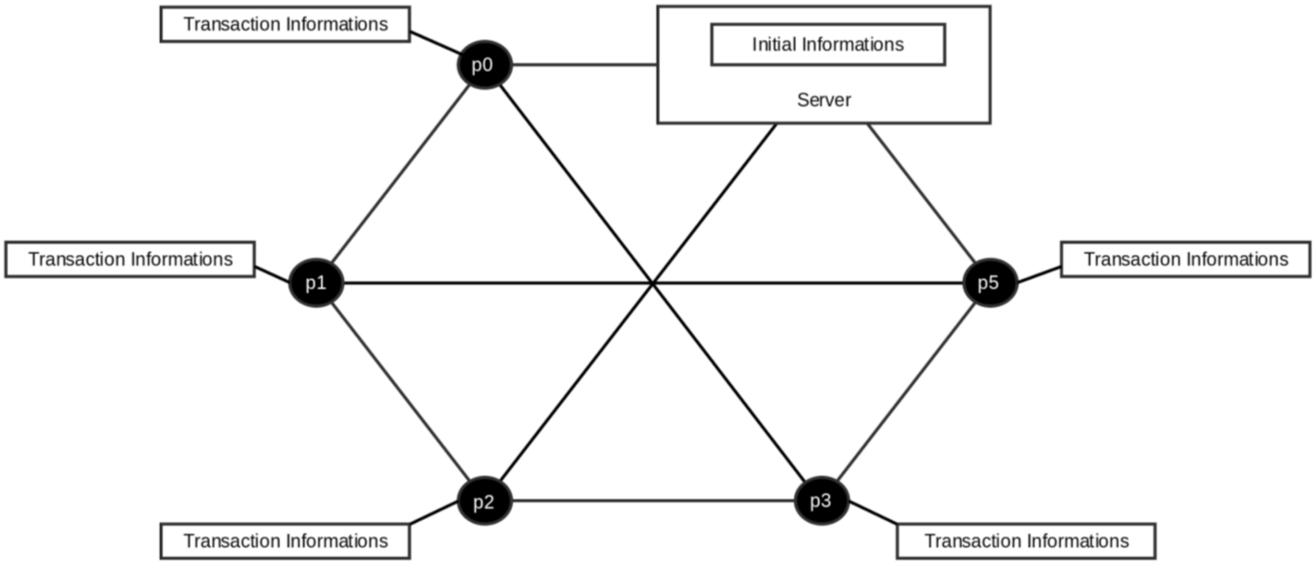

Figure 2.2.3 Our design

The second problem is how to change the block chain for the new peer architecture. We solved this problem by dividing the block chain into two parts: the genesis blocks (created by the server) and normal blocks (created by users). The normal blocks have positive block id’s and the genesis blocks have negative block id’s. The transactions in the genesis blocks have no inputs and the outputs are new coins. The inputs in the transactions in normal blocks can refer to outputs in both genesis and normal blocks. The Bitcoin block chain is shown in Figure 2.2.4 and the block chain architecture used in this project is shown in Figure 2.2.5.

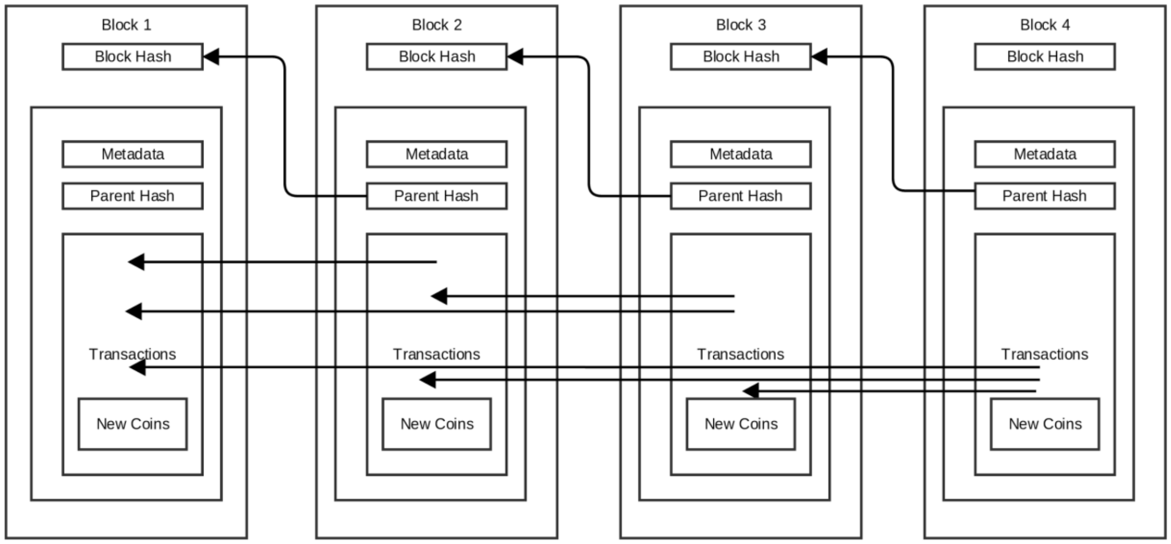

Figure 2.2.4 The block chain in the Bitcoin system

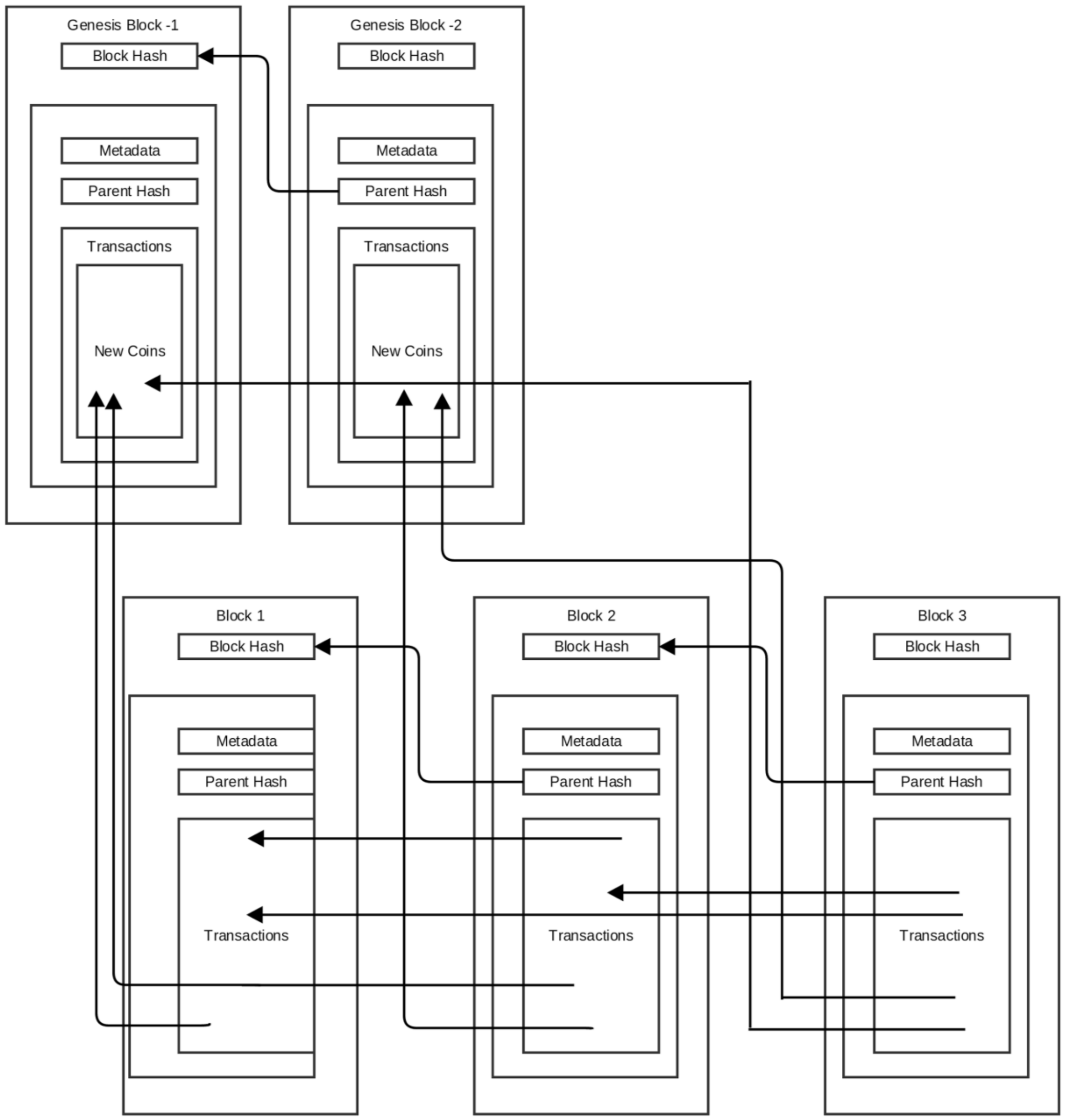

Figure 2.2.5 The block chain architecture in this project

The third problem is motivating users to keep system reliable.The security of the Bitcoin depends on the number of users who run the software and the Bitcoin solves the problem by giving mining reward to the user. However, we cancel the reward because we want to create all coins ourselves and we still need to motivate users to keep the system safe. Also, when removing the mining process, the block creation cannot be managed. Therefore,we found a solution like this: the mining process will be same in Bitcoin and the authority will keep track of the miners. The authority may pay money to the miners or give some coins in the next block creation etc. It depends on the system that will be built with this architectures.

Chapter 3

In this chapter, we will explain the details of the project implementation.

3.1 Project Features

The project is written in Java as an Eclipse project. In the project, there are more than one main classes. From the project, two executable jar file can be exported: Server.jar and Client.jar. We only used one external library (Gson[3]) in the project and while exporting the jar[4] files, the dependencies are included to the jar files. Therefore, the output of the project is just a jar file.

The project software is licensed GPL v3[5].

3.2 Menus

The program has not a graphical interface. It is runned and managed from the terminal. There is simple menu in both server and client software. In server software, new coins can be created. In client software, new keys and transactions can be created.

3.3 Storage

At the beginning of the program, necessary directories are created if they don’t exist. All data is stored in txt files with simple text encoding.

The keys and blocks are stored in different folders under “./resources/” directory. The user blocks are stored in “./resources/chain/” and genesis blocks are stored in “./resources/genesis/”. Also, in server side, the keys corresponding to the genesis blocks are kept in “./resources/genesis/keys/public” and

“./resources/genesis/keys/private/”. Some informations about program history are stored in ./resources/properties/. The block files contains JSON[6] of the blocks and their names are “block_id.txt”.

3.4 Data Structures

3.4.1 Keys

Every user has a public key and a private key in this system. The keys are generated with RSA[7] and encoded with Base58[8]. The private key length is 128 characters and public key length is 32 characters. The signing procedures are done with MD5WithRSA[9] hashing functions in Java.

3.4.2 Transactions

In this system, the inputs of a transaction have to belong to the same user. It means it doesn’t allow that more than one user perform a transaction together (multisig feature). However, the outputs can be belong different users. The fields of a transactions are below.

- Sender Public Key: The public key of the user who wants to perform the transaction
- Transaction Content: The informations of the transaction.
  - Transaction Date: The field is generated automatically.
  - Transaction Inputs: List of Transaction Input
    - Amount: The amount of the transaction input

        - Block ID: The id of the block in which the unspent transaction output is
        - Transaction Order: The order of the transaction in the block (whose block id is above) in which the unspent transaction output is
        - Output Order: The order of the output in the transaction
    - Transaction Outputs: List of Transaction Input
        - Amount: The amount of the output
        - Public Key: The public key of the receiver of the output
- Signature: The signature of the user that is generated for the transaction content

3.4.3 Blocks

We don’t use merkle tree or another special data structures in this system. The transactions are kept in simple list. In addressing, hashing is not used. To address an unspent money, block id,transaction order and output orders are necessary.

In mining, the system requests the hash target from server and it starts to create a hash in desired range by changing mining variable:nonce.

- BlockContent: The content of the block
    - Block ID: The id of the block (negative for genesis blocks)
    - List of Transactions: Transactions in the block
    - Parent Block Hash: The block hash of previous block
    - Miner Public Key: The public key of the miner
    - Nonce: A variable for creating different hashes in mining
    - Date: The mining date
- Block Hash: SHA-256[10] hash of the block content

3.5 Network

We use Java Socket programming to create a peer-to-peer network. The application socket is 9001. All requests are sent and received in JSON format of base request class. This class consists of a request code,shows the request type, and corresponding request data. The request list are below.

- NewPeerEchoRequest: Sends the ip number of the user to the server
- GetPeersRequest: Gets all peers' ip numbers from the server
- GetTargetRequest: Gets the hash range from server to mine new block
- GetNumberOfGenesisRequest: Gets the number of genesis blocks from server
- GetGenesisBlockRequest: Gets the genesis blocks from server
- TransactionForwardRequest: Sends a transaction to a user
- GetNumberOfBlockRequest: Gets the number of block request from a peer
- GetBlockRequest: Gets a block from a user
- BlockForwardRequest: Sends a block to a user

In this system, a peer gets the number of genesis blocks,genesis blocks and ip address of other peers from server. Then, the peer asks the number of normal blocks to all peers and it accept the most common number. Than, it asks the blocks one by one and it select the most common block in each time. In downloading, it checks the block's validity.

Also, each peer forwards coming block and transaction to all peers except the sender. Although the algorithm is not efficient, we use it because of its simplicity. The peer received a forwarded transaction or a forwarded block checks its existence in the pool and its validity before adding into the pool.

3.6 Threads[11]

3.6.1 Main Thread

It is the main program thread and it administers the other threads. At first, it sets initial settings,starts the network listener and displays a menu for user. The user can do his jobs by using this console menu.

3.6.2 Network Listener

It is the server-like thread in each peers and it listens the application port. Whenever a new request is came, it starts a new thread for this request.

3.6.3 Request Handler Thread

It is runned for each request. It checks the request id, calls the necessary functions and answers the request with the result of the functions.

3.6.4 Transaction and Block Forward Threads

The threads are runned to send a transaction or a block to all peers.

3.6.5 Miner Thread

The thread does the mining process. There is an interface named MinerResultListener and it is used to connect the main thread and a miner thread. A miner thread calls the interface functions after the mining process and gives the result to the main thread.

Chapter 4

4.1 Tests

We tested the software in Azure Virtual Machines[12]. The tested features are below.

In Server Side:

1) Running the server software
2) Creating new genesis blocks
3) Tracking the client connections

In Client Side:

1) Running the client software
2) Connecting to server
3) Downloading the number of genesis blocks from server
4) Downloading genesis block from server
5) Creating transaction
6) Checking a transaction validity
    a) Sign validity
    b) Input existence
    c) Input is spent or not
    d) Input ownerships
    e) Input-output matching
    f) Key validities
7) Forwarding a transaction

8) Getting, checking and forwarding a forwarded transaction
9) Getting the number of blocks from peers
10) Getting the normal blocks from other peers

Chapter 5

5.1 Conclusion

The project is not completely finished yet. Research and design phases were done. Bitcoin and classical banking systems are learned and analyzed and the new architecture is designed. Necessary data structures and core functions are implemented properly. However, some functions (will be listed in 5.2) have to be written. The testing of the software was done on a few machines and only simple situations (listed in 4.1) were tested. It needs a wide scale testing to be sure about the system security. Also, the menus are so primitive and a graphical user interface is necessary.

5.2 Future Works

5.2.1 Necessary works for this project

As we mentioned before, there are lack of implementations and tests in the project. The list is below.

1) The software needs a GUI. It can be implemented by Swing[13] libraries.
2) The options that are presented in the menu must be increased. Key management, monitoring system, wallet settings are candidates.
3) Peer-to-peer mechanism used in the project is so primitive. Each peer interacts with the all other peers. Improvements are essential.

4) Forwarding mechanism needs a smart algorithm to decrease the workload on the network.
5) Block forwarding must be implemented.
6) Block pool must be implemented.
7) Transaction pool management must be improved.
8) Mining thread is written but its usage in the software must be implemented.
9) Thread synchronization must be implemented.
10) The software needs a wide scale testing. Network latencies, thread synchronizations have to be tested.

5.2.2 Future Applications

In this project, we built the system for a new cryptocurrency however the model can be applied for other areas. The model has 2 main features: The transferable units of the system are created by an authority and the units can be transferred without the authority permission. In cases that the features are hold, the model can be used. Ticket systems are one of good example. The organizers can create tickets and sell them. Then, the users can sell their tickets whenever they want. It increases the flexibility and decreases providers' workload.

The car licensing system are another good example. Now, people have to go notary to certificate sales. However, if the system is used, after manufacturers creates licences at first, people can certificate sales by themselves.

Briefly, the model can be applied to many real systems to make life easier.

5.3 References

[1] "Bitcoin - Open Source P2P Money." *Bitcoin - Open Source P2P Money*. Web. 12 Apr. 2015. <https://bitcoin.org/en/>.

[2] Antonopoulos, Andreas M.. *Mastering Bitcoin*. Sebastopol, CA: O'Reilly Media, pp 177, 2015.

[3] “Google-Gson - A Java Library to Convert JSON to Java Objects and Vice-Versa - Google Project Hosting.” google-gson - A Java library to convert JSON to Java objects and vice-versa - Google Project Hosting. N.p., n.d. Web. 31 May 2015. <https://code.google.com/p/google-gson/>

[4] “Using JAR Files: The Basics.” (The Java™ Tutorials > Deployment > Packaging Programs in JAR Files). N.p., n.d. Web. 31 May 2015. https://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html

[5] “The GNU General Public License v3.0 - GNU Project - Free Software Foundation.” The GNU General Public License v3.0 - GNU Project - Free Software Foundation. N.p., n.d. Web. 31 May 2015. <http://www.gnu.org/licenses/gpl.html>

[6] "Introducing JSON." *JSON*. Web. 12 Apr. 2015. <http://json.org/>.

[7] "RSAPrivateKeySpec (Java Platform SE 7)." *RSAPrivateKeySpec (Java Platform SE 7)*. Web. 12 Apr. 2015. <https://docs.oracle.com/javase/7/docs/api/java/security/spec/RSAPrivateKeySpec.html>.

[8] "Base58." *Wikipedia*. Wikimedia Foundation. Web. 12 Apr. 2015. <http://en.wikipedia.org/wiki/Base58>.

[9] "Signature (Java Platform SE 7)." *Signature (Java Platform SE 7)*. Web. 12 Apr.

2015. <https://docs.oracle.com/javase/7/docs/api/java/security/Signature.html>.

[10] “SHA-2.” Wikipedia. Wikimedia Foundation, n.d. Web. 31 May 2015. <http://en.wikipedia.org/wiki/sha-2>

[11] “Thread (Java Platform SE 7).” Thread (Java Platform SE 7). N.p., n.d. Web. 31 May 2015. <https://docs.oracle.com/javase/7/docs/api/java/lang/thread.html>

[12] “Virtual Machines.” - Linux & Windows VMs. N.p., n.d. Web. 31 May 2015. <http://azure.microsoft.com/en-us/services/virtual-machines/>

[13] “Trail: Creating a GUI With JFC/Swing.” (The Java™ Tutorials). N.p., n.d. Web. 31 May 2015. <http://docs.oracle.com/javase/tutorial/uiswing/>