Independent Project Final Report

Blockchain-based Forum

|  |  |
| --- | --- |
| Name | LING Liyang |
| Student ID | 20527456 |
| ITSC | llingab |

# Introduction

At the beginning of my first semester in UST, I chose independent project as one of all my courses. The basic idea is to gain knowledges from practical projects so that the information obtained could be more concrete and this experience might be helpful for job hunting.

From the projects list, I finally pick Blockchain-based Forum.

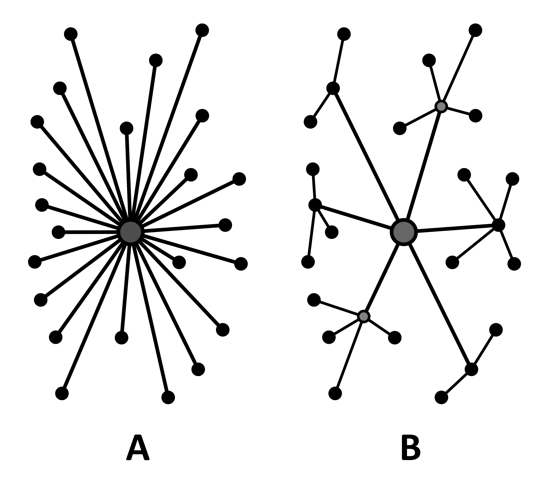
Basically, this project is based on the blockchain technique to implement a forum system. Taking the advantages on blockchain, the target is to empower this forum with high security so that the contents on this forum cannot be tampered and deleted, in this way, users’ freedom of speech and authenticity of messages can be guaranteed.

Why this project?

Because I have heard the word blockchain many times before, and still have no idea about that. For this project, my purpose was to unveil the mystery and learn the theory and mechanism of blockchain. On the other hand, I am familiar with python programming which just meet the project language requirement, implementing the forum in python would not be too hard for me to get start and time-consuming.

# Preliminaries

A blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across many computers, so that any involved record cannot be altered retroactively, without the alteration of all subsequent blocks

To understand the idea of Blockchain, firstly we should get to know the idea of decentralization.

From the left picture, figure A shows a typical centralized structure. All nodes connect to the same central node, and the central node takes responsibility to control and transfer information among those connected nodes. That means all sub-nodes should fully trust the central nodes. If central node make mistakes, this mistake can only be checked and corrected by itself. If central node crashed down, the whole system would get crashed. So, the central node should be very robust and powerful. For example, nowadays many common bank systems follow this structure.

Figure B shows a kind of model of decentralization. So-called decentralization has the purpose of breaking the rule of centralization. In this way, each node in this system could have a copy of all information of all nodes, using specified cryptography methodology to guarantee the validity of transactions respectively. Considering the performance and storage ability of each node, these nodes are divided into different levels, like server nodes and customer nodes. Server nodes are in charge of storing all blocks, while customer nodes only keep blocks that they actually need. Once new message or block is delivered, all nodes will verify the information and choose to synchronize or not. Thus, nodes in different levels construct a big complex network, as if some nodes crashed down, other nodes would not be influenced.

We apply this mechanism onto forum system.

For blockchain system, there two kinds of nodes, we call them full node and light node. As its name, full node has full functions of blockchain server node and stores all blocks, light node exists on mobile devices and browsers, only has partial abilities and store its needed blocks.

In this way, once some users post their articles and comments on something onto this forum, this message will be stored into all effective nodes. Even someone modify or delete this message, the cryptography mechanism would not let this kind of change pass and make sure the authenticity of each message.

Of course, to implement those functions, there are many detailed works to do, and there are also many problems remaining to be solved. I’ll explain them combining my works.

# Work Description

Basically, because of the change of concrete progress of infnote project, my project work arrangement had some adjustments. I divided them in to three parts.

## Stage 1 – Blockchain backend in python

In the first week, my task is to implement a simple version of blockchain storage system in python while using MongoDB as the database system as a result of its high performance on dealing with high redundancy and less related information.

According to the given demo and blockchain backend class diagram, the simple version was implemented with two main parts, database and blockchain. Among them, database module controls the I/O interfaces for blockchain part, blockchain module including keys, blocks, users and payloads manager to support the full backend functions.

Because of the modifications on cryptography methodology and the changes of backend functional needs, this part of work cannot fully satisfy the requirements of expected blockchain forum, this part of work was abandoned in the third time project meeting. The overlapped parts I will explain in the second stage.

## Stage 2 – Blockchain backend in JavaScript

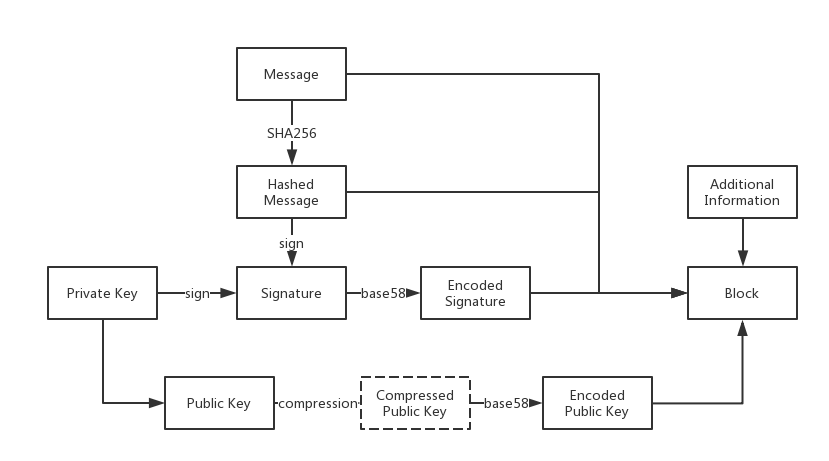
After simply evaluation of first week work and discussion on future plan, my work center was transferred to implement independent blockchain system on browser using JavaScript. Similar to the structure of python version, there are three parts Keys, Blocks and Storage on browser backend.

### Keys

Basically, the crypto key method is NIST P-256 from ecdsa which is popular in common bit coin and blockchain encryption methodology. In JavaScript platform, after comparing several encryption packages in npm, I picked up ec-key with P-256 encrypt curve.

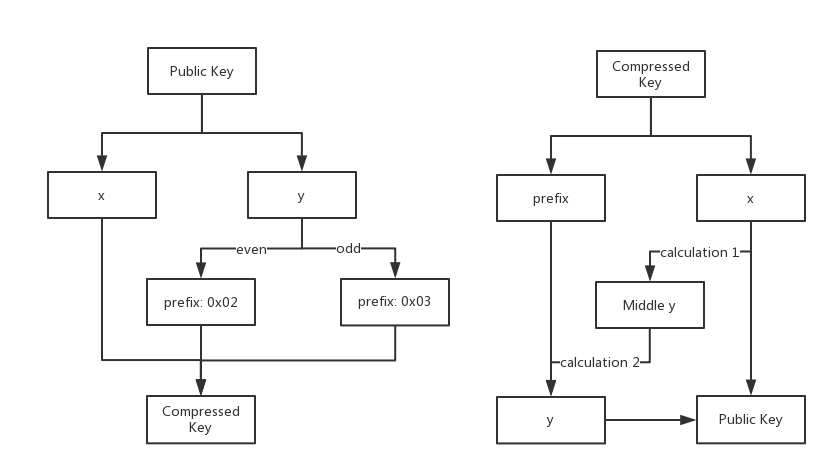
Each key object has two class members: the private key and the public key, and the public key comes from private key. Which means if one has the private key, he or she is the valid user of this forum. If they want to post something, they could use their private key to sign the message and send the message, signature and the public key generated from their private key, so that other nodes can use the public key to verify the authenticity of this block.

The detail procedure is showed in the flow chart below.



Except for the signing and key generating procedures I mentioned, there are some additional procedure.

To make sure the security of content and computational complexity, the signature and public key should be encoded in base58 format.

Also, there is a compression procedure for public key. After systematically evaluation on the system performance, we found the redundancy of storing keys and price of make key hash are too high. So, the proper key compression is required.

What should be mentioned before is that there are three parts x, y and d of ec-key generally. In these three, d is unique for private key, while x and y are sharing part of public key and private key. However, x and y are not exactly two independent parts, y could be generated from x. So, we have the above compression and decompression procedure.

For compression, the public key is separated into x and y two part. According to the parity of y, we get the relative prefix in hex and then join it with x, we could get compressed public key. The public key can be shortened from 64bits to 33 bits. That means the cost on storing and processing public key can be reduced to around a half.

For decompression, we split the compressed key back to prefix and x. Then through a time of big integer calculation, we could get a middle value. If the middle value’s parity conforms to the prefix, the middle value is y, or another big integer calculation is needed to get y.

Here are the two big integer calculation formulas.

(1)

(2)

Where *b*, *prime* and *pIdent* are all constant big integer.

### Blocks & Chains

For this part, based on the basic idea of blockchain, the specified data structures of blocks and chains are implemented.

During this design, a block is smallest storage unit while each valid block contains publish timestamps, encoded compressed public key, encoded signature, it belonged chain id, its position in this chain (height), the message will be posted in forum (payload), and self-content hash value and the hash value of previous block. This design is according to the implementation of blockchain in Bitcoin.

The chain is a logical idea that distinguish different blocks by the chain id, also the signing key object of the chain owner.

### Storage

Different from using MongoDB in python platform, browser cannot call any local resources directly for security issues. To implement the complete functions on browser, the storage tool is the local storage embedded in most popular browser software. The restrictions of local storage also cause the most difference between light nodes and full nodes.

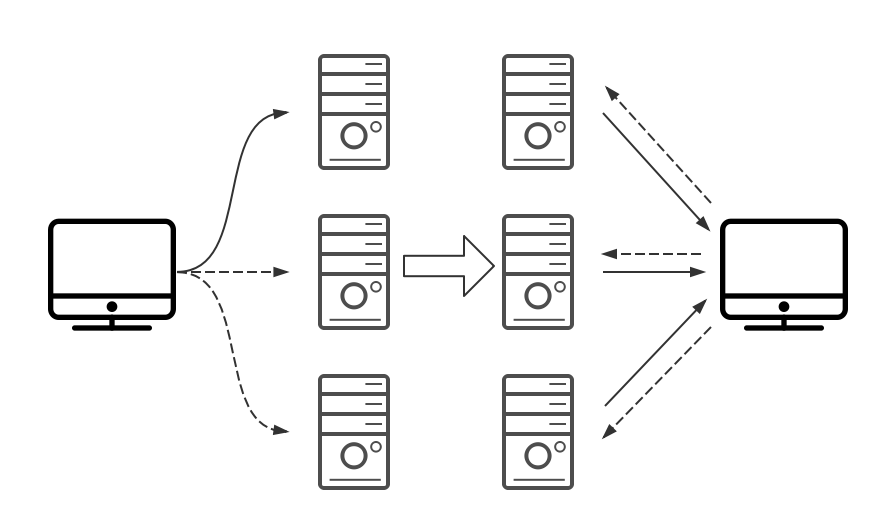
The first one is the storage format. Local storage only supports very simple key-value pair format. No tables, no collections, no supports for different kinds of queries, would make it quite hard for many common database operations. So, it’s needed to implement the needed database operation functions on our own. To solve this problem, I designed the specified naming rule for different kinds of content and used an extra python dictionary to improve its performance on searching and adding.

The second one is the storage upper bound of local storage is 5M while if we want to store most blocks, its size could not lower than 5M. So, the system should make it possible that unimportant blocks could be removed from light node in time and consistency of blockchain could be maintained well at the same time. To solve this problem, Merkle tree algorithm are planned to be applied, my group mates will finish that.

## Stage 3 – P2P network backend in JavaScript

Another very important part of this project is the mechanism of communication.

### Communication mechanism

Basically, we defined the forum specified communication protocol and implement it based on web-socket protocol on different platforms.

However, although we called it P2P network, the actual working mechanism is like the mixture of BS framework and peer to peer network. Because the light nodes like mobile devices and browsers cannot be connected initiatively.

Thus, there are several solutions for this problem. Firstly, we hard code the IP and port addresses for well-prepared servers. Secondly, user can input the target peer address manually on the web page to connect to specified peer. Third, we build our DNS to help find peer through domain name.

For my work, I just solve this problem by the first two methods temporarily.

### Message format

To make the message and request can be delivered properly, we defined the message format in two layers.

The first layer we call it Message layer, it’s used for identifying each message. There are only three fields: IDENTIFIER, TYPE and CONTENT.

The IDENTIFIER is 10 length random character and digits, which mark the one-time conversation. There are three types in TYPE fields: question, answer, broadcast and error which represent the information request, response, broadcast signal and error happening signal respectively. The last field CONTENT contains the content of the next layer and also the concrete message in this message package.

The second layer is called Sentence layer, which contains eight types of message, they are INFO, ERROR, WANT\_PEERS, PEERS, WANT\_BLOCKS, BLOCKS, NEW\_BLOCK and BAD\_CHAIN. In the next part, I will explain what these types are combining how these types work.

### Action dispatching

In the forum system, one of the most important parts is implementing the mechanism that make nodes could synchronizing through p2p network automatically and effectively. So, there is needed a series of methods on message package construction, message dispatching, response information collecting and delivering.

The detailed fields are easy to be seen in code, so here I am going to introduce my implementation on dispatching.

Firstly, when a node receives a QUESTION, it should according to this request give back the related response in ANSWER type. When the node gets the answer for the question it asked from the target peer, that means this conversation is finished. Because web-socket is a high-level protocol, there is no need to do several times of handshakes as TCP and this mechanism has already implemented in the lower level.

When the connection is successfully built between two peers, they will exchange their info and according to difference between their info to decide what kinds of message should be sent the next. The messages about peers and blocks are similar. If the status of blocks’ or peers’ is inconsistent with these two nodes, question want\_blocks or want\_peers would be sent to synchronize the different part. When receiving these questions, receiver would collect blocks or peers information according to the parameter passed in question and wrap all collected content to response in answer.

The another one is BROADCAST message. When a node receives a message in type BROADCAST, it should pass this message to all its connected peers to achieve the idea of broadcasting. However, if no constrains, the broadcasted message would be echoing in the network endlessly. So, each node would record the message it has broadcasted, when receiving the same message for the second time, it would not broadcast it again.

Currently the BROADCAST message only contains NewBlock, as BadChain is not fully implemented because of lacking complete strategy to define bad chain.

# Results and Discussions

## Results and contributions

Partial Final Codes is on: https://github.com/Infnote/Infnote\_SPA

In this course, I spent around six week-time on this project, effective codes achieve 1800 lines, and help finishing performance testing before infnote paper submission.

## What I have learnt

* Basic idea on how blockchain works
* Basic cryptography techniques
* JavaScript programming skills

## Discussions and Thinking in some problems

Of course, there is a pity that this project is not fully finished because of problems from multi-aspects.

Firstly, the new adjustment on data transferring format I have not applied onto light node part. The main problem is the message encoded on python platform using protobuf in python could not be decode on the protobuf package in JavaScript. Maybe protobuf.js is not the proper package, and I have found another more popular package called protobuffer on github, but there is no time for me to do further researching and testing because of the pressure from other courses in the end of this semester.

The next is that there still many inefficient codes in JavaScript part which could be improved.

What’s more, some more functional modules and functions should be redesigned and refined to make this product be better.

Thanks for this project and this semester’s work!