The Final Report for A Blockchain Explorer

MSBD6000D Group Project, 2020 Spring, HKUST

YANG Rongfeng
20644943
ryangag@connect.ust.hk

LI Yunli
20659584
ylijd@connect.ust.hk

Abstract

As Bitcoin and other cryptocurrencies have been picking up steam, focus has been turned to blockchain – the underlying distributed ledger technology (DLT) that empowers these digital currencies. In this paper, we aim to build a decentralized application - a blockchain explorer that connects the Blockchain File System (BFS) and Blockchain File Coin (BFC) provided by the course. It offers a user-friendly front-end page to support various fundamental functions in the process of digital currency transaction.

Key Words: Blockchain, Explorer, BFS, BFC

1 Introduction

1.1 Background

A blockchain,[1][2][3] is a growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block,[4] a timestamp, and transaction data (generally represented as a Merkle tree).

A blockchain is a decentralized, distributed, and oftentimes 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.[1][5] This allows the participants to verify and audit transactions independently and relatively inexpensively.[6] A blockchain database, namely Blockchain File System, is managed autonomously using a peer-to-peer network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests.[7] Such a design facilitates robust workflow where participants' uncertainty regarding data security is marginal.

Blochain File Coin(BFC) is a kind of cryptocurrency and digital payment system intended to be a blockchain-based cooperative digital storage and data retrieval method. Miners can elect to provide storage capacity for the network, and thereby earn units of the Filecoin cryptocurrency (FIL) by periodically producing cryptographic proofs that certify that they are providing the capacity specified.

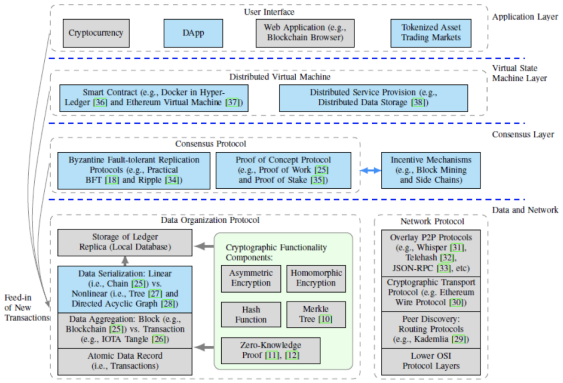

Figure 1: Blockchain Network Implementation Stacks[9]

From Figuresection 1.1 we can see the whole blockchain network stack. There are four layers in this network. And the application layer, i.e. user interface, is the bridge that connects the user and the blockchain. Additionally, in the virtual state machine layer, smart contracts are computer protocols intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.

1.2 Project Description

Our team proposes to build a useful front-end explorer, a decentralized web tool that provides detailed information about blocks, addresses, and transactions, and supports basic query operations, including searching for the transaction ID (*txid*), account, and smart contract address (*ctAddress*).

The article is organized as follows: we firstly introduce the background of project and describe the project goal under individual contributions of each group member (§1). Then at the (§2), we discuss the related work to our project. Furthermore, we describe our application architecture in (§3) and their corresponding functionalities. Additionally, we will show the layout of our explorer in (§4), and reveal detailed experimental process - how to utilize this useful web tool to fulfill the users needs.

1.3 Contribution

`YANG Rongfeng:` Project managing, framework building, coding, report.

`LI Yunli:` Web interface improving, coding, presentation, report.

2 Related Work

There are several ecosystems in this blockchain network: [10]

- BOS (Blockchain Operating System): a mini operating system for digital assets management.
- BFS (Blockchain File System): a decentralized cloud storage system that reduces cost and boosts efficiency by crowd-sourcing resources with the underlying distributed ledger.
- BFC (Blockchain File Coin): the underlying distributed ledger powering BFS. BFC distributes rewards to and charges fees from different BFS stakeholders.
- BDC (Blockchain Distributed Computing): a decentralized platform for buying and selling computation power. Task organizer can acquire computing power for general tasks at the cost of BDC.
- BCA(Blockchain Certificate Authority): A decentralized platform for certificate issuing and identification. Official records can be shared and verified for a lifetime.

Since we aim to link the BFS and BFC, short introductions will be given concerning the two systems.

2.1 Blockchain File System (BFS)

The traditional cloud storage method vastly used nowadays are centralised services that access data on distributed servers with load balancing algorithms. It has various drawbacks:

- Expensive.
- Centralised bottlenecks.
- Data transparent to service provider.
- Incapable of permanent storage.

BFS offers a better solution to cope with these problems. It is a P2P decentralization network with compressed sensing, where encrypted data are distributed throughout the network to increase reliability and avoid bottom necks from centralization. What's more, by leveraging blockchain for authenticity and incentive, records contracts of file storage sessions are difficult to tamper with and rewards contributors will be rewarded accordingly, which in turn, stimulates the development and security of this network.

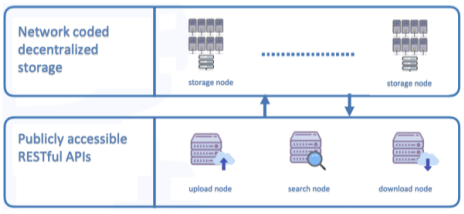

Figure 2: Comparison between Cloud Storage and BFS

The system mainly consists of four parts:

1. Client
2. Operation Nodes, such as Upload Node, Download Node, Search Node, and Annotation Node
3. Retrieval Node, Proxy Node (Optional)
4. File Node

This is an encryption system, where all files stored on it must be cryptographically protected when uploaded. A client firstly accesses the system, choosing the operation nodes as they want. The chosen node will then sent the request to the retrieval node (there might be a proxy node prior to the retrieval node), and the rn node will then operate in the corresponding file node.

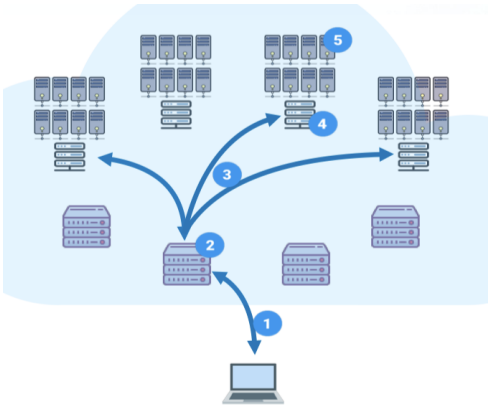

Figure 3: BFS Architecture

2.1.1 API

We implement two APIs of BFS in our website: uploading files and downloading files. A friendly interface was built to guide users to upload and download their contract and data.

Uploading Contract and Data

Type	post
Storage	file copies number
Days	days the files store
Field	storage field
File	the path of file

Table 1: File Upload API

Upload contract.py, var.json, input.json or any kinds of files such as videos, documents, music and etc. If files are uploaded successfully, the afid of the files will return.

Downloading Contract and Data

Type	get
FileID	the afid of file
FileName	the name of file

Table 2: File Download API

Files can be download by providing the afid and the name of files.

2.2 Blockchain File Coin (BFC)

Blockchain File Coin (BFC) is a decentralized network-coded storage system. What makes it different from other storage system lies in its features of dynamic storage and credential levels for different services and needs. The credentiality requires on-chain deposit, with Delegated Proof of Stake (DPoS) consensus algorithm to achieve dynamic credential levels. It allows file search and sharing, which is different from the traditional filecoin.

2.2.1 API

All the APIs of BFC can be categorized into five parts: Query, Fetch, Apply, Sign and Submit.

Query

Type	get
Query parameter	value

Table 3: BFC Query API

Query APIs include account balance query, account statues query, contract query and transaction execution status query. We design a search bar to collect the necessary parameters (Transaction ID, Account Address or Contract Address) to complete query.

Fetch

Except for querying certain data, the blockchain explorer also need to display the detailed information of the blocks and accounts. Therefore, we fetch data from the `get blocks` API and `get accounts` API to display in the front end.

Apply

Type	post
From	your address
To	receiver's address
Amount	the fee you want to transfer
Auxdata	custom data to store
CarryFee	reward for miner
FunctionName	Need in contract-call TX
InputDataAFid	Need in contract-call TX

Table 4: BFC Apply API

If user wants to make a transaction, applying a TX is the first step of the whole process. There are two kinds of TX, basic one and a contract-call one. The first one is only transferring some coins from A to B and another one will call a contract during a transaction.

Sign

After applying a transfer transaction, an unsigned TX will return. We deploy an executable binary SignSDK in the Server to sign this TX, generate its signature and append behind.

Submit

The signed TX is been placed in the body of POST request and the server will submit it to the BFC.

Type	post
Unsigned TX	the raw data of unsigned TX
Signature	generated by SignSDK

Table 5: BFC Submit API

3 Explorer Design

3.1 Explorer Architecture

The architecture of the explorer are shown as Figuresection 3.1 below. We utilize HTML and javascript to encode our front-end and PHP for the back-end. The framework is based on JQuery and Vue. JQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. Vue.js is an open-source Model–view–viewmodel JavaScript framework for building user interfaces and single-page applications. Combined the two, our web provides a well-made layout and useful interfaces to serve the users.

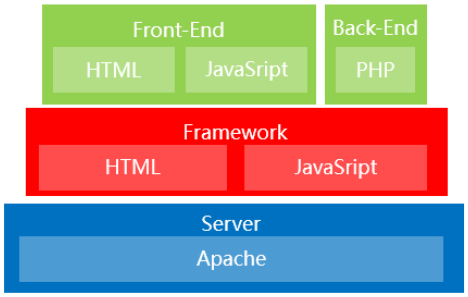

Figure 4: Explorer Architecture

It is worth mentioning that we adopt Responsive web design (RWD) approach to our explorer, which makes our web pages render well on a variety of devices and window or screen sizes. Content, design and performance are shown across all devices to ensure usability and satisfaction. The practice consists of a mix of flexible grids and layouts, images and an intelligent use of CSS media queries. As the user switches from their laptop to iPad, the website should automatically switch to accommodate for resolution, image size and scripting abilities.

For example, figure 5 and figure 6 shown below represent the responsive designs for iPhone X and iPad Pro respectively. The information in different devices are presented well-adjustedly.

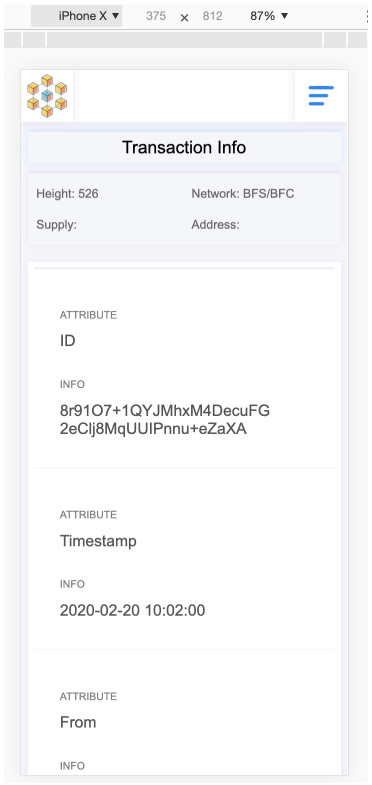

Figure 5: iPhone X Display

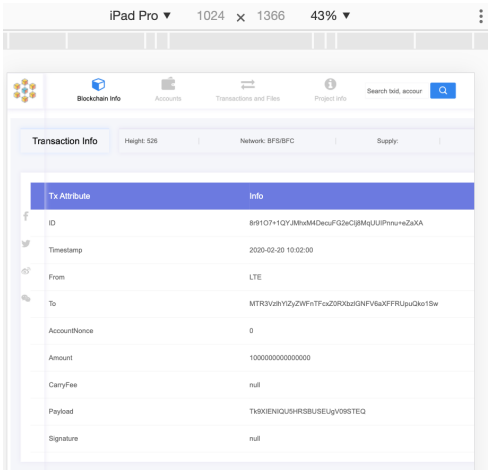

Figure 6: iPad Pro Display

3.2 Explorer Webpages

The design of our explorer consists of three main pages: 1) Blockchain Information 2) Accounts 3) Transactions and Files. They are shown in the top of our web as a fixed menu bar. Additionally, there is a search box besides the menu, where users can search for the transaction ID (txid), account, and smart contract address (ctAddress).

We will elaborate each page, introducing their layouts and functionalities sequentially.

3.2.1 Blockchain Information Page

This is the home of our web, where all the transactions and blocks are shown. There is a sub tool bar with two buttons: 1) Transactions 2) Blocks. After clicking different buttons, different information will render to users.

Transactions

If we click on the "Transaction" button, which is also the default showing option, the following info will be printed out on users' screen:

Info	Description
ID	the hash of the transaction ID
Timestamp	transaction timestamp
Sender	the public key of the sender
Recipient	the public key of the recipient
Amount	transaction amount
Fee	transaction fee
Height	block height

Table 6: Transactions Page Info

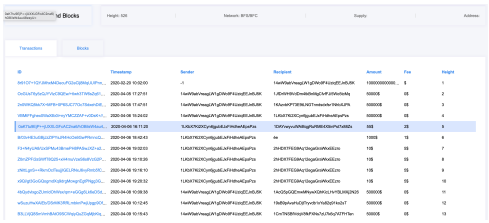

Figure 7: Transactions Subpage

Besides the listing of transactions, users can click on each transaction ID, which is a hyperlink, and detailed info of the transaction will be shown as below:

Info	Description
ID	the hash of the transaction ID
Timestamp	transaction timestamp
From	the sender
To	the public key of the recipient
AccountNonce	account nonce
CarryFee	the reward for the miner
Payload	contract information
Signature	the signature of transaction

Table 7: Transaction Attributes

Blocks

If we click on the "Blocks" button, the following info will be printed out sequentially on users' screen:

Info	Description
BlockHash	the hash of the block
Height	block height
Timestamp	block timestamp
TxNumber	the number of TX
Generated by	the miner
Nonce	nonce

Table 8: Blocks Page Info

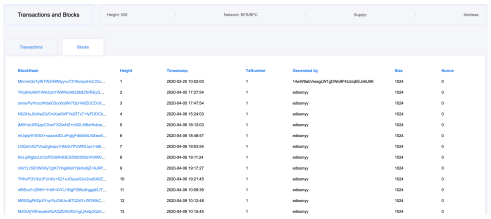

Figure 8: Blocks Subpage

Besides the listing of blocks, users can click on each BlockHash, which is also a hyperlink, and detailed info of the block will be shown as the table below:

Info	Description
Hash	the hash of the block
Version	current chain version
Height	block height
ParentBlockHash	hash of parent block
MinerAddr	miner address
Size	size of the block
Timestamp	block timestamp
Nonce	nonce
TxNumber	the number of Tx
Afid	afid of transactions in the block
Extra	extra information
StateRoot	world state
TxRoot	root of txs in current block
ReceiptRoot	root of receipts

Table 9: Block Attributes

3.2.2 Accounts

By selecting the "Account" button in the menu, a list of accounts in the network and their detailed info will be shown on the screen. The info and their descriptions are listed in table 10 below:

Info	Description
#	the info sequence
ID	the name of the account
Address	the address of users
Balance	account balance
AccountNonce	account nonce
StorageRoot	storage root
CodeHash	code hash

Table 10: Account Attributes

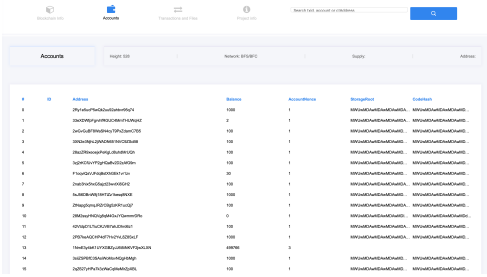

Figure 9: Account Page

3.2.3 Transactions And Files

The layout of this page is shown as figure 11 below.

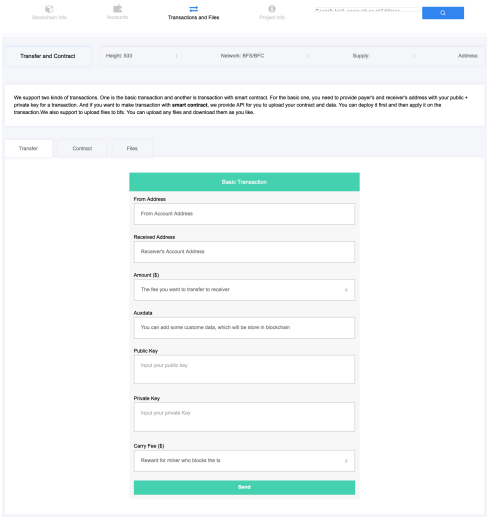

Figure 10: Transactions And Files Page

There are three subpages under Transactions and File page, which support two kinds of transactions as well as an extra function of file operation. The two types of transaction are:

1. **Basic transaction:** users need to provide their public and private key, together with the addresses of payer and recipient.

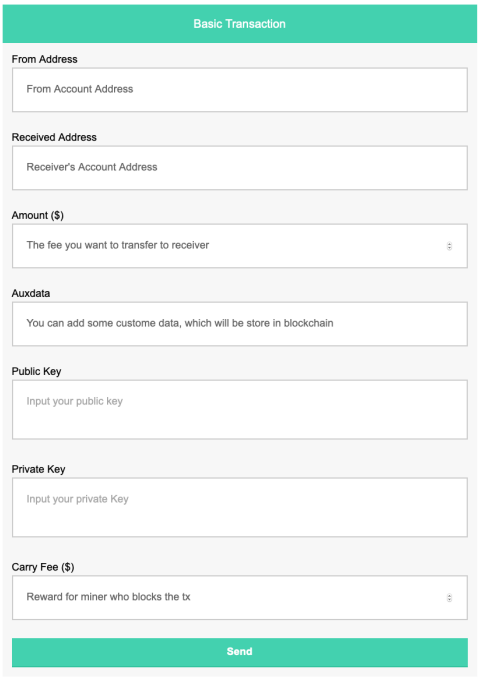

Figure 11: Basic Transaction

2. **Transaction with smart contract:** the API for uploading contract and data is provided. Users need to deploy their contract in Step 1. In this process, users need to utilize the file upload function, which we will elaborate shortly after, to upload their contract, and get the contract address, *ctAddress*. After the deployment of the contract, users can choose to apply it during the transaction process.

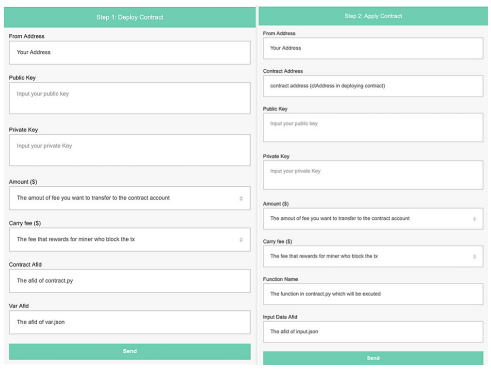

Figure 12: Transaction with Smart Contract

Additionally, we support file uploading and downloading in BFS, which is the necessary step during the deployment of contracts. Users can click on "Files" button and the corresponding page with upload, download APIs will be shown as below. Users can upload any files, getting file addresses for further operation. And they can also

and download them by searching the corresponding file address.

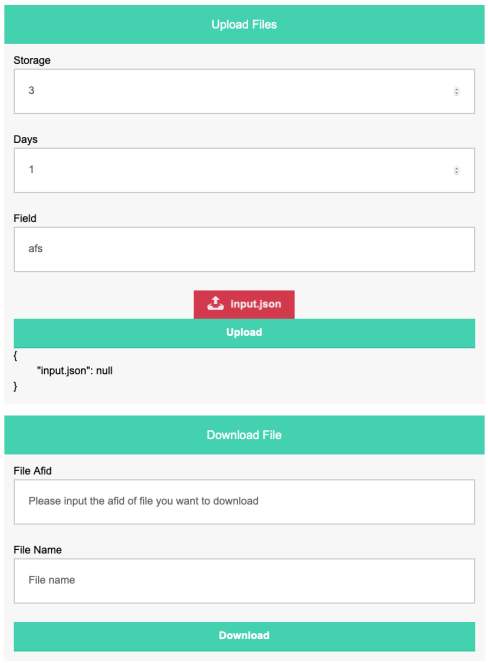

Figure 13: File Subpage

4 Experiments and Results

4.1 Account Status Query

For the querying box besides the main menu, for example, if a user wants to search for his or her own account to check the balance, he or she can type in their address in the box, and hit the search button, the web will then, returns the status information of the account as a small pop-up message shown in figure 15. The background of the webpage will be cast on shadow, which emphasizes the pop-up message, where users can easily find the information they need. The effect is shown in the figure 15. We can see that the balance of the account we have searched is 500957, account nonce is 4, with the return message, which states that this query has been successfully executed.

After getting the info, the user can click anywhere outsides the message window, the web will return to the normal state, waiting further requests from users.

4.2 Transaction and *Txid* Query

Now let's say I want to transfer 100 to my teammate for the basic transaction without a smart contract. Under the basic transaction page, firstly, I type in my address, my teammate 's address, and

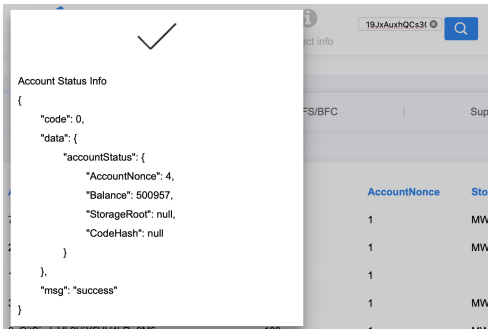

Figure 14: Address Query

the amount of value I want to transfer. After filling out these info, I can add in some customized data under the *Auxdata* box, which will also be store in the blockchain. Additionally, my public and private key are required for verification. And carry fee is the reward for the miner. After I have done all of these, click the **Send** button, and I get a message from the server, stating that the transaction has been successful and the *txid* will be printed out together.

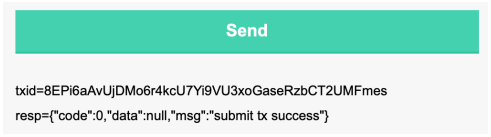

Figure 15: Basic Transaction Return Message

Now I can use the *txid* I got from the transaction and perform query at the top. The query returns the transaction info as shown in figure 17. The transaction status, type, carry fee, block number will be printed out to users. The same effect as the account query.

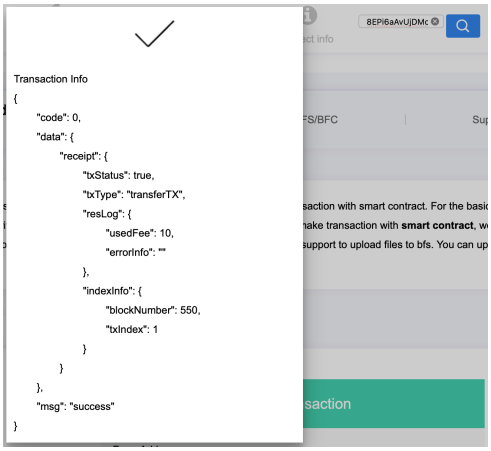

Figure 16: Txid Query

4.3 Transaction with Smart Contract and *ctAddress* Query

Let's say I want to transfer another 100 to my teammate, but this time, I want to apply my own smart contract. There are two parts to finish this process: 1) deployment of the contract and 2) Application of the contract.

Deployment of the Contract

The steps to deploy contract are as follows:

1. Upload the contract in the Files subpage, get the *ctAddress* of the contract.
2. Upload the var file, get the *Afid* of the var.
3. Upload the input file, get the *Afid* of the input.
4. Switch back to Contract subpage, fill out the Address, Public Key, Private Key, Amount, Carry fee, Contract Afid and Var Afid, press the "Send" button.

If the contract has been successfully deployed, a piece of information should be appearing under the "Sent" button, in which transaction id (*txid*) and contract address (*ctAddress*) are returned to users.

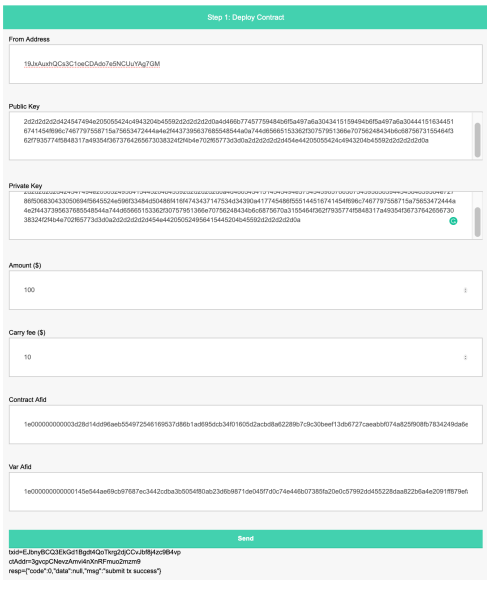

Figure 17: Contract Deployment

Application of the Contract

In the Step 2, users need to fill out their Address, Contract address (*ctAddress*), Public Key, Private Key, Amount, Carry fee, Input data address (*Afid*) inside the text-boxes. What's more, users must add the function they want to achieve in the contract manually in the Function Name box. In this demo, we use the function "add" the demo contract provides.

From the figure 18 we can see that the server renders a *txid* and a message to indicate the transaction has been successful and completed.

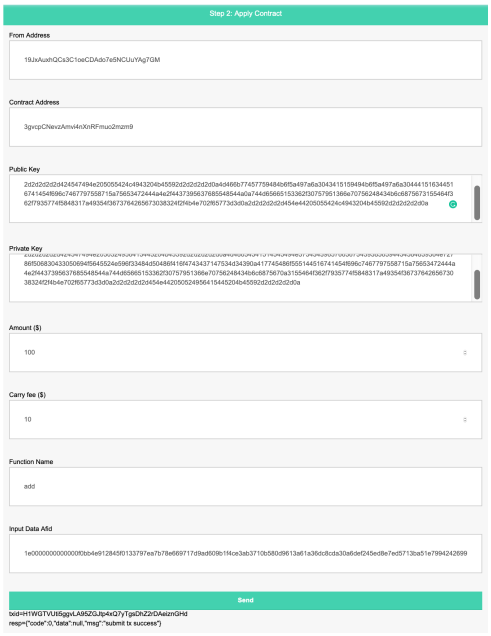

Figure 18: Contract Application

Now we can use the transaction id we just got and perform query by searching the *ctAddress*. The pop-up window shows the transaction info just as the basic transaction info search, however, the *txType* clearly shows that this is a transaction with a smart contract.

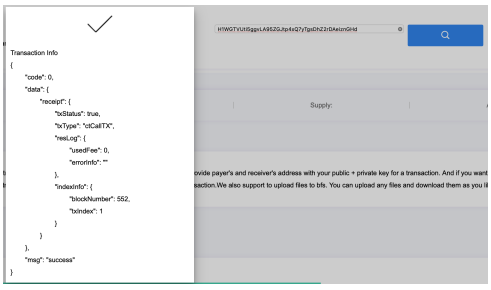

Figure 19: ctAddress Query

4.4 File Upload and Download

Just like any other file uploading process, users press the red upload button in the bottom, choose the file they want to upload to the server. Batch upload is supported, which means that users can

choose multiple files at once. The storage needs to be specified by choosing the number between 3 to 7. The Days should also be addressed, meaning how many days the file will be perserved on the server. Lastly, users have to fill out Field, whose value is chosen from $|afs|, |arfs|$, and $|afs|arfs|$.

After finshing filling out, click upload and the server will then return the file's address.

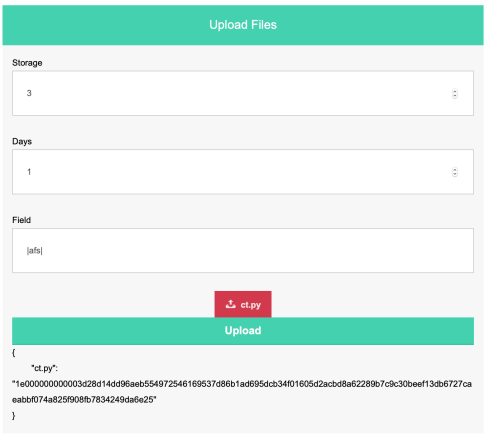

Figure 20: Upload File

With the file name and address, users can fetch the file and download it again by typing the File Afid and the File Name inside the Download function.

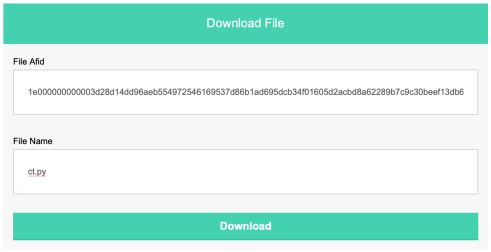

Figure 21: Download File

5 Conclusion & Future Work

In conclusion, in this project, we have built a well-designed explorer to present the information on the blockchain, such as transaction info, block info. Moreover, it provides several useful APIs for users to perform query as well as file operations. It fulfills the basic needs for most cases.

However, due to the time scope of the project, there are some drawbacks we are aware of, yet have not managed to fix. And we list them as our future work:

1. Pagination display
   In the home page, where transactions and blocks are listed, there is only one widget called "Next" for users to browse. In the early developing stage it should not be a problem. However, as users and transactions are adding up, this pagination display will present as a big problem, impeding the user experience greatly.

   What's more, such design slows down the query process. Because after every operation, the server will return all the blocks, which is a considerable amount of data flow that slows down the loading speed.

   Therefore, in the future work, the pagination display will be changed to an active and hoverable pagination, where all pages are listed at the bottoms and users are able to jump to any page as they want. Such a modification greatly improve the user experience and in the same time, improve the fluency of page loading.

2. Addition of the storage structure
   The query function is still primitive and limited. Powerful storage structures such as MySQL database or Radius Cache can be added into the server to support complex queries.

3. Server architecture replacement
   The server we use now is built on Apache, an open-source HTTP server for modern operating systems. It is one of the most popular server ever in existence. But in the same time, it's also one of the oldest web servers, with its first release all the way back in 1995. As the blockchain network grows larger and maturer, there will be numerous users access this website every day or even every second. In the situation of high concurrency and traffic, Apache doesn't perform well at scale. Thus, a new server architecture called NGINX can be considered as the next server architecture to cope with the problems.

References

[1] "Blockchains: The great chain of being sure about things"
https://www.economist.com/news/briefing/21677228-technology-

behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable

[2] Morris, David Z. (15 May 2016). "Leaderless, Blockchain-Based Venture Capital Fund Raises $100 Million, And Counting" http://fortune.com/2016/05/15/leaderless-blockchain-vc-fund/

[3] Popper, Nathan (21 May 2016). "A Venture Fund With Plenty of Virtual Capital, but No Capitalist" https://www.nytimes.com/2016/05/22/business/dealbook/crypto-ether-bitcoin-currency.html

[4] Narayanan, Arvind; Bonneau, Joseph; Felten, Edward; Miller, Andrew; Goldfeder, Steven (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction* Princeton: Princeton University Press. ISBN 978-0-691-17169-2.

[5] Armstrong, Stephen (7 November 2016). "Move over Bitcoin, the blockchain is only just getting started" https://www.wired.co.uk/article/unlock-the-blockchain

[6] Catalini, Christian; Gans, Joshua S. (23 November 2016). "Some Simple Economics of the Blockchain" http://www.nber.org/papers/w22952.pdf

[7] Tapscott, Don; Tapscott, Alex (8 May 2016). "Here's Why Blockchains Will Change the World" http://fortune.com/2016/05/08/why-blockchains-will-change-the-world/

[8] Bheemaiah, Kariappa (January 2015). "Block Chain 2.0: The Renaissance of Money" https://www.wired.com/insights/2015/01/block-chain-2-0/

[9] Wang and Hoang. "A survey on consensus mechanisms and mining management in blockchain networks"

[10] "Intro-Blockchain-System-For-Project.pdf"