VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



**CHARITY FUNDRAISING APPLICATION BUILT ON ETHEREUM BLOCKCHAIN**

By

Ho Minh Huy

A thesis submitted to the School of Computer Science and Engineering

in partial fulfillment of the requirements for the degree of

Bachelor of Computer Science

Ho Chi Minh city, Vietnam

2018

**CHARITY FUNDRAISING APPLICATION BUILT**

**ON ETHEREUM BLOCKCHAIN**

APPROVED BY:

Tran Thanh Tung, Ph.D.

THESIS COMMITTEE

**ACKNOWLEDGMENTS**

I would like to express my very great appreciation to Dr. Tran Thanh Tung for his valuable and constructive suggestions and guidance during the planning and development of this research work. His constant encouragement and support helped me to achieve my goal.

Table of Contents

[LIST OF TABLES 5](#_Toc512869996)

[**LIST OF FIGURES** 6](#_Toc512869997)

[ABSTRACT 7](#_Toc512869998)

**iv**

# LIST OF TABLES

# **LIST OF FIGURES**

# ABSTRACT

Blockchain is a recent technology which provides a alternative type of database, structured as a chain of blocks of data, which is managed autonomously using a peer-to-peer network and distributed timestamping server. By design, a blockchain is unchanged. At the time researching this study, there are 3 major blockchain protocols such as Bitcoin, Ethereum and Ripple Consensus Network.

Current approaches of making payment through applications, especially on websites, is that payers have to trust and provide their private information to the middleman which is a bank or a payment service to valid transactions from their accounts and transfer their money to the final destinations. It is too risky. Not only in making payment but also in funding, the fundraiser owner does not directly receive funds from donators. Both sender and receiver are able to keep track of their transactions. But, it is not the way they expected.

This research approach is to apply Blockchain technology to create a decentralized application running on Ethereum private blockchain. This application has its own custom Token exchange and fund between owners and fundraisers. Besides, it is also required HTML including CSS, Javascript, Jquery, Bootstrap and Web3js for the front-end. The local server is run using Express module in Nodejs and a distributed database, which is MongoDB, to store fundraisers contents.

# CHAPTER I

## INTRODUCTION

The current fundraising web application is very centralized. Kickstarter, one of the largest fundraising platform, is a conspicuous example. Product teams can go to <https://www.kickstarter.com/> to create a project, set a goal and start collecting money from others who interested in their ideas. Kickstarter is essentially a third party that sits between product teams and supporters. If the project successfully funded, the project team expects Kickstarter to give them the money. On the other hand, supporters want their money to go to the project if it was funded or to get a refund when it hasn’t reach its goals. Therefore, both of them need to trust Kickstarter to handle their money correctly.

There is always a middleman sits between the donators and the fundraiser’s owner. It makes the fundraising system becomes centralized and depends too much on the third-party such as a bank or a trusted organization ( like Kickstarter ) to manage and control the fund. Moreover, the donators must provide their personal information to the bank in order to prove their identity each time they want to use their money. Personal information leaking is a constant worry of internet users these days. Actually, it is one of the world’s most valuable commodity. It can be mostly used in business marketing, surveys, or event hacking bank accounts and take control on someone’s property. The intermediary often promises to keep the personal information safe by providing an array of protection strategies. But somehow, they are not totally efficient. The Facebook’s Data Leak, which is a most serious problem of the company in 2018, has been revealed that the data of up to 87 million users was leaked. It made the reputation of Facebook be seriously damaged. It has also aroused controversy over the data branch of the intermediaries. Internet users then have an awareness of providing their information on the public network.

The risk of personal information leaking and middlemen role has been proved. In Vietnam, from 2016 to 2018, there were a dramatically increase of reports of cases that the money was withdrawn from bank accounts without any permissions of the owners. Most of cases reasonable occurred due to the greed of a person or a party, who worked for the bank. Those people were able to steal the money from an account because they could keep track of it using the information that the owner had provided. Even they finally paid for their crime, but it was not sufficient to make up for the loss and the reputation of the bank.

Moreover, the traditional client-server architecture of current web application has some issues. The first issue is that the server is running on a Centralized Hosting Service. As more and more people around the world use the same popular hosting service, it becomes a bigger single point of failure for the entire internet. To take down all the sites that are running their Backend on this popular hosting service, the trouble-maker only need to disrupt this provider’s infrastructure in order to collapse potentially half of the internet. This is not a very resilient architecture for the web. It is also an increasing amount of pressure on a singular organization to expect them to build and maintain all the infrastructure for the entire world. Centralized servers are also easy to censor and control. Taking down a web application is as simple as removing it from the centralized server. This can happen at the state level where government agencies request hosting providers to take down websites they may not like or agree with.

Users are demanding from an application which does not need the role of an intermediary and provide users a method to manage their data or digital possession without revealing personal information. Moreover, that application should run everywhere so that taking it down would not be as straightforward. Is that kind of application possible? Yes. With blockchain, everything becomes possible. A blockchain is a decentralized database which managed by a peer-to-peer network. Each peer in the network has a copy of the blockchain database running on a local computer. A blockchain cannot be controlled by any individual.

The main goal of this research is to build a Decentralized Charity Fundraising Application called D-Fundraiser, which is a combination of smart contracts deployed to Ethereum blockchain testnet- a simulation of real Ethereum blockchain which is managed by a peer-to-peer network - in order to make all the fundraisers and their funding transactions in this system become decentralized, immutable, indelible and transparent. Funding by cryptocurrency, donators are also anonymous. The solution is not as 100% as it is expected because the real world is still mainly using banknote in trading. Hence, it requires a final exchange from cryptocurrency to cash.

# CHAPTER II

## BACKGROUND

Developing applications on the blockchain is certainly something that is new and cutting edge –in fact, called Decentralized Applications (Dapps) is a new word to describe the type of applications that are powered by the blockchain. To have a well prepared, there are some basic knowledge that developer should clearly understand.

### Blockchain technology

Blockchain technology is amazingly trendy nowadays. This technique was authentically described in 1991 by a group of researcher and its purpose was to “timestamp” digital documents in order to prevent them from being tampered or put on a date that was earlier than which they were written. But until 2009, Satoshi Nakamoto adapted this technology to run the digital cryptocurrency called Bitcoin, Blockchain has become widespread.

Hash function and Merkle Tree are two techniques that must be known before going into details with blockchain.

**Hash Function**

00111100...10101

INPUT

e1703fa96e70xa....

OUTPUT

HASH

This is a one-way function. It takes an input, usually a sequence of bits which could be any digital data, and produces a fixed size values,such as an alphanumeric string, as output. However, if just only a single bit of the input is changed, the output would be completely different. Moreover, there is noway to figure out the input if there is only the output.

**Merkle Tree**

Merkle Tree is a data structure where each layer is a combination of hashes.

1

2

12

3

4

34

R

1234

To construct the Merkle tree, each data is hashed and then coupled by hashing their hashes together. For example, from the figure......, [1], [2], [3] and [4] are hashes of data. They are then coupled and produce the output of two new hashes [12] and [34]. To form the Root of the tree, [12] and [34] are hashed together to produce the Top hash [R]. The root will be a representation of this data structure and it is impossible to figure out each individual data from the root.

Into details, a blockchain is a distributed computing architecture. Instead of using a central entity to manage the chain, blockchain uses a peer-to-peer network. Anyone is allowed to join and became a node of the network. When someone runs as a full-node, he or she gets the full copy of the blockchain. In this way, every network node executes and records the same transactions, which keep the blockchain in consensus across the whole network. Individual user interactions with blockchain are called transactions. At a time, only one block can be added and every block contains a mathematical proof that verifies that it follows in sequence from the previous block. The common format of a block consists is illustrated in figure:

BLOCK

HEADER

Previous block hash

Technical data

Timestamp

Merkel Root

Difficulty target

Nonce

Transaction count

Block content

**Header**: The block header is hashed twice to create the fingerprint which is referred to in the next block.

* *Technical data*: Included a Magic ID, a version number ( to specify which set of protocol rules a block conforms to), the size of a block.
* *Merkel Root*: Distills all the transactions in the block into a single hash (the root of transactions Merkel Tree).
* *Timestamp*: Approximate timestamp of when a block was created. Use to figure out mining difficulty re-target i.e if the network is making blocks too quickly or too slowly
* *Previous block hash*: 2x SHA256 has of previous block header (excluding magic ID & block size). This is the link that creates the chain of blocks.
* *Difficulty target*: Related to mining and how hard it is successfully mine the block.
* *Nonce*: A random number. One of the things that miner can change when mining to create different hashes, while search for a suitable hash.

**Transaction count**: How many transactions are in the block.

**Block content**: Holding a list of transactions.

To create a new block and link it to a chain, blockchain network has its workers called miners. A miner start constructing a candidate block by gathering the transactions in the “*transaction pool*”-a pool of new or unconfirmed transactions. This candidate block arrangement is same to a normal block structure. Next, he or she spends resources (dedicate hardware and electricity) to compute the hash of the block’s header. If the output is fulfill the *Difficulty target,* new block is created and linked to the chain*.* Otherwise, that miner has to change the *Nonce* until an appropriated output is found. In this way, miner is “mining” for a new block. Since the computational process of solving for the solution to create a new block is costly and time-consuming, the output is literally called a Proof-of-work.

After a mining is done, that new block is sent to every nodes on the network. Each node then verifies the block to make sure it is valid. All the node in this network create consensus. They make agreement about which blocks are valid and which are not by following those criterion:

* Block header hash is appropriated to the block target.
* Block size is within acceptable limits.
* Block timestamp is less than a *T* time in the future.
* All transactions within the blocks are valid (also have a checklist on their own).

After successfully mining a new block, miner get a total of reward for mining block and transactions fees.

### Cryptographic Proof of identity

Cryptographic Proof of identity in simply means proving one’s identity without revealing it. It use the cryptography techniques to generate a digital signature. This will make a person become anonymous because he or she does not need to provide personal information but still can prove the privileged possession of a property.

Here is what a cryptographic system can do. It generate something called a public key-private key pair. They are a set of long characters that are mathematically connected. A public key is public like a username, and the private key is a secret like a password. Once a property of a person is encrypted with his or her public-private key pair by the cryptographic algorithm, only that person can prove the ownership of that property. Since, it would take 1000’s of years to break the cryptographic algorithm because of the computational limits nowadays. Quantum computers in the future might challenge this. But, there would be a solution to handle this.

### Smart Contract

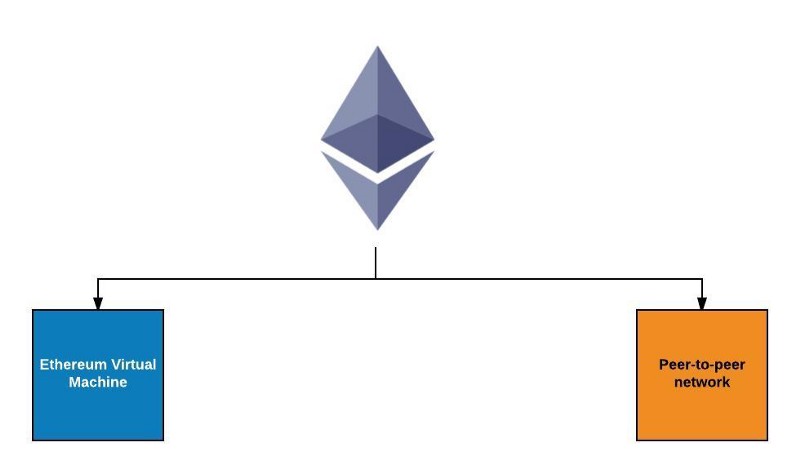
The born of Smart Contracts is one of the recent developments in this field. They are very popular nowadays. The term “smart contract” was first used by Nick Szabo in 1997. He wanted to use a distributed ledger to store contracts.

Nowadays, smart contracts are similar to contracts in real world. The only difference is that they are completely digital. Two parties can agree on a set of rules, create and code a contract. As soon as the criterion/rules are met, it automatically authorizes the validation between two parties. Smart contracts not only store conditions but also data. However, people should admit the limitations of the smart contract. Some systems need human support, computers cannot handle everything. In the real world implementing smart contracts is not easy. Smart contracts once published cannot be altered, silly mistakes can be costly.

### Ethereum

A next-generation blockchain that had the ambitions to implement a general, fully trustless smart contract platform called Ethereum, which is a combination of Blockchain, Cryptography and Smart Contract technologies.

In 2014, Vitalik Buterin, Gavin Wood and Jeffrey Wilcke became the founders of Ethereum. It comprises a Ethereum Virtual Machine and a peer-to-peer network protocol. The Ethereum blockchain database is maintained and update by many nodes connected to the network. Each and every node of the networks runs the Ethereum Virtual Machine and executes the same instructions. For this reason, Ethereum can be described as a “World Computer”.



Ethereum Virtual Machine

Peer-to-peer Network

**Ethereum Virtual Machine**

As has been said, Ethereum is a programmable blockchain. Instead of providing users a set of predefined operations (like Bitcoins did with its transactions), Ethereum furnish them a friendly language running on a “Virtual Machine”-called an Ethereum Virtual Machine (EVM). At the heart of Ethereum, Ethereum Virtual Machine (EVM) can execute code of arbitrary algorithmic complexity, which makes it to be “Turing complete” – a system that can understand a set of instructions and execute them in some logical order, just like a computer does. Using Solidity as its programming languages, Ethereum serves as a platform for various types of decentralized blockchain applications, including but not limited to cryptocurrencies.

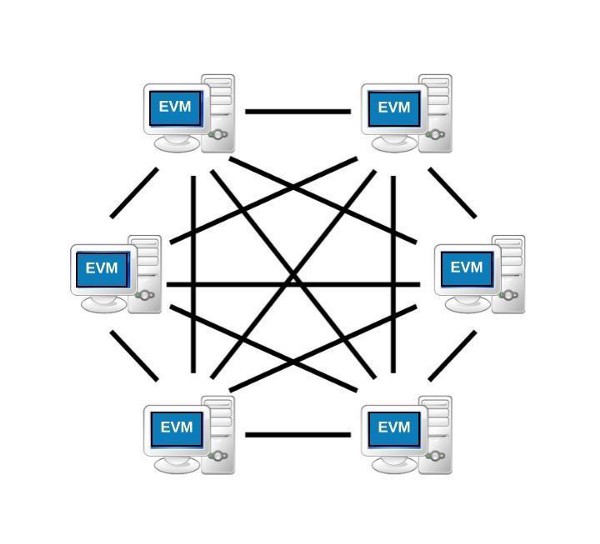
**Solidity**

Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++. Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM)

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

**Peer-to-Peer Network Protocol**

The fundamental network consists of nodes (or computers) connected in a decentralized, peer to peer network. Each node runs an EVM, and process the same instructions to make sure that consensus across the network is achieved on any particular transaction.



**Ether**

Ether is the cryptocurrency of the Ethereum network which can be used in trading with other supported cryptocurrencies. In the other hand, it can be seen as the fuel, which is needed to run a transaction on Ethereum blockchain. Ether is also a reward to miners each time they succeed mining a new block on Ethereum blockchain. Developers who intend to use Ethereum as a platform to build apps need Ether, as it is used to pay for computation within the EVM. Shortly, users who want to interact with Ethereum blockchain applications must use Ether.

Ether is to the Ethereum network, what bitcoin is to the Bitcoin Blockchain network. Hence, users should become Ethereum miners, or trade with other currencies using centralized or trustless services in order to gain some Ethers.

The base unit of Ether is called Wei.

**Ethereum Accounts**

Comparing to Bitcoins, one of the first and biggest blockchain application which is purely a list of transactions, Ethereum’s basic unit is the account. There are two type of accounts:

|  |  |
| --- | --- |
| **Externally Owned Accounts (EOAs)** | **Contract Accounts** |
| Controlled by an External party or person  Accessed through private keys  Contain Ether Balance  Can send transactions as well as ‘trigger’ contract acounts | Have code that executes when being triggered  Also contain Ether Balance  Can trigger other contract accounts  Live on the Ethereum Blockchain |

If a user wants to participate in the Ethereum ecosystem, he or she should have a user account and the keys to operate that account. These types of accounts are called Externally Owned Accounts (EOAs) in Ethereum. Once an Externally Owned Account (EOA) is created, transactions can be made from that account to the others as well as to other types of account – Contract Accounts - on the Ethereum.

A Contract Account contains code to execute some designed functions, and this code is put on the Ethereum Blockchain. When a smart contract is triggered – by an Externally Owned Account or by another contract account – the code inside is executed by the EVM on each participating nde.

**Smart Contract in Ethereum**

There is nothing different from the meanings mentioned before except that Smart Contract is now put on Ethereum blockchain. There are steps which a smart contract is created in Ethereum:

1. A smart contract is coded in Ethereum programming language (such as Solidity) following some conditions (e.g. “If A is true, then do B”) and then deployed on Ethereum Blockchain by an EOA (using some Wei as fees).
2. Once deployed, the smart contract gets a public key address, which can be used to reach the contract and trigger its code execution. This address is the contract account of that smart contract.
3. A deployed smart contract cannot be changed, even by the EOA that created it.

The bottom line is that Smart Contracts are code that is deployed on the Ethereum blockchain, and this code runs on every single node connected to the Ethereum network.

**The workflow of Ethereum**

Ethereum has the same Hashing data and Proof-of-work mechanism as a blockchain, but its hashing function is keccak256, sometimes (erroneously) called sha3. On Ethereum, a user could trigger a transaction from his or her EOA. A transaction is a validated packet that contain:

. The address of the recipient.

.A signature that proves the possession of the sender’s account.

.Value field – The amount of ether to transfer from the sender to the recipient. It can be empty, in some cases.

.An optional data field, which could be an arbitrary message or function call to a contract. As an example, accidental insurance smart contract would require to input the proportion of damage in order to release the respective compensation.

.Transactions on ethereum also need something called “gas” to run. Gas is Ethereum’s metering scheme and it accounts for bandwidth used, cost of data storage and cost of computation on the Ethereum blockchain. Every computational operation in the EVM consumes gas and different computations consumes different amount of gas. Those additional attributes must be generated before triggering new transactions:

-startgas: the quantity of “gas” that a transaction is willing to consume to cover its user of the EVM’s computation and any storage bandwidth used. It is really difficult to define exactly how much is the startgas. Therefore, there are certain APIs pretending the transaction was actually being included in the transaction, and then returning the estimated amount that would have been charged if that pretend operation was real.

- gasprice: mentioned by the transaction sender. This is the amount of ether that a sender is willing to pay per unit gas.

- gas\_rem: If a transaction execution is done and consuming less gas then its specified limit, the transaction sender receives a refund of *gas\_rem\*gasprice*.

Once a transaction is sent, the destination can be another EOA or a Contract Account. The transaction to an EOA is simply an Ether transfer, which means ether balances from both sender and receiver will be adjusted. On the other hand, destination of the transaction is a Contract Account, the smart contract’s code will be executed automatically. In some cases, a Contract Account needs to run a function from another Smart Contract (or Contract Account), it will send a message to that Contract Account which contains:

+ the address of the contract sending the message

+ the address of recipient contract

+ the amount of ether to transfer alongside the message

+ an optional data field

+ a startgas value

The message results in the recipient contract running its code.

EOA

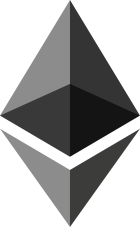
sender

EOA

receiver

Contract

Account



Update blockchain

Update blockchain

Run code

Transaction

message

Ethereum

Another

Contract Account

Transaction

After a transaction is done, the “state” of an account is updated in the Ethereum blockchain. At the first time an account (either EOA or Contract Account) is created, it is at the genesis state. When a transactions on a Ethereum is successfully mined by a miner (a blockchain worker who groups transactions – which included many updates to the ”state” of accounts- into block and compete with one another for his or her block to be the next one to be added to Ethereum blockchain), the current state of the recipient accounts is updated in the Ethereum blockchain by the whole nodes on the network.

Miners are rewarded the amount of Ethers, which is equal to *startgas\*gasprice* of a trasaction. At the start of a transaction execution, this amount of ethers is removed from the transaction sender’s account to ensure the miner receives the fee even if the sender account is bankrupted midway during execution. If there is gas refund *gas\_rem* to the sender, miner of that transaction receives a refund of *(startgas – gas\_rem)\* gasprice.*

**Ethereum Decentrailizaed Application**

A dApp consists of back-end code that runs on a decentralized peer-to-peer network.

For an application to be considered a Dapp, it must meet the following criteria:

* The application must be completely open-source, it must operate autonomously, and with no entity controlling the majority of its tokens. The application may adapt its protocol in response to proposed improvements and market feedback but all changes must be decided by consensus of its users.
* The application's data and records of operation must be cryptographically stored in a public, decentralized blockchain in order to avoid any central points of failure.
* The application must use a cryptographic token which is necessary for access to the application and any contribution of value from (miners) should be rewarded in the application’s tokens.
* The application must generate tokens according to a standard crytptographic algorithm acting as a proof of the value nodes are contributing to the application.

A Ethereum decentralized Web application is designed following the structure in figure.

FRONT END UI

HTML/CSS/JS

ETHEREUM BACKEND

Smart Contracts

json

The Frontend is programmed using HTM, CSS, Javascript and Nodejs. It is use to display the responses from the Ethereum blockchain (server-side) and handle input from the client-side.

The Backend code is the combination of programmed smart contracts stored on Ethereum blockchain.

There are several characteristics according to which decentralized applications can be classified. For the purposes of this paper, we will classify Dapps based on whether they have their own block chain or they use the block chain of another Dapp. Based on this criterion, there are three types of Dapps.

**Type I** decentralized applications have their own block chain. Bitcoin is the most famous example of a type I decentralized application but Litecoin and other “alt-coins” are of the same type.

**Type II** decentralized applications use the block chain of a type I decentralized application. Type II decentralized applications are protocols and have tokens that are necessary for their function.

**Type III** decentralized applications use the protocol of a type II decentralized application. Type III decentralized applications are protocols and have tokens that are necessary for their function.

# CHAPTER III

## METHODOLOGY

### Overview

Besides the implementation of smart contracts, Ethereum ecosystem also supports building *decentralized applications* (dApps).

**Detail Of Methodology:**

In this research and implementation, A decentralized charity fundraising application is built to be used in a university. It is a combination of 4 layers: Front-end, Transport, Services, Back-end.

The Backend has programmed smart contracts including functions and methods. They are initially deployed to Ethereum blockchain. The application backend contains

+Admin contract: approve and close fundraisers.

+Fundraiser contract: receive Charity token as fund. When the fundraiser is expired, this contract will automatically sending the amount of Funding Token to the owner.

+Charity Token contract, Funding Token contract, Reward Token contract: ERC20-based Tokens which are programmed as cryptocurrencies for funding, exchanging and rewarding in this decentralized application.

The Web3 is the javascript library included in this project to help the client talk to the contract and make calls to and from the Ethereum blockchain. The web3 library is what the interfaces between the Frontend and the server side code that is written on the blockchain via smart contracts.

With this system, fundraiser’s owners have right permission to their funds without an involvement of any third-party. On the other hand, donators are anonymous when they donate a fund by using cryptocurrency. Hence, there is no middleman stands between donators and the fundraiser owners. All the money is transferred from donators’ wallet to the fundraiser. And, the owners directly withdraw all the funds from their fundraisers.

Each time a student or an organization in the school wants to form a new fundraiser to raise fund online, they deploy a fundraiser contract to the Ethereum blockchain. This contract is programmed to automatically open when reaching the start date and close after having expired. The address of the fundraiser contract will be used to receive Charity Coins (a customized cryptocurrecy token).

The owner of the fundraiser is able to withdraw all funds after their fundraisers have expired.

There is a EOA called administrator managed by the Charity Office. This EOA will have to read and check the valid of a fundraiser. If it is valid, administrator will allow that fundraiser to start by activating it. Otherwise, It is refused and the owner must contract the office to find out what kind of mistake he or she has made. Of course, the fundraiser owner could not be anonymous.

The donator is the anonymous one. Donators could

Những thành phần của của Dapp và những công nghệ áp dụng vào nó

**Front-end**

**Transport**

**Services**

**Back-end**

# CHAPTER IV

## IMPLEMENTATION

**System Workflow ( sơ đồ ) giải thích vai trò của từng tier**

**Environment settings:**

Tạo genesis block

Chạy Ethereum private node bằng geth: giải thích các thong số

cài metamask

Giải thích Các Function trong smart contract

Caai2 các engine của nodejs cần thiết

Setup ABI và Address của các smart contract trong file json

**Implement step**

\_ tạo account ( geth và metamask )

- Mining some ether using Geth

Deployed smart contract to Ethereum ( bằng geth hoặc remix )

# CHAPTER V

## RESULT / DISCUSSION

Demo:

Tương tác với Blockchain ( các comment web3 của web )

Thử các flow của system

Kết quả:

Viết về so sánh kết quả với mục tiêu đề ra

# CHAPTER VI

## CONCLUSION

**1. Summary**

### The combination ( viết lại )

So, why blockchain technology along with cryptographic identity proof is so promising? The first combination of these two techniques is Bitcoin, a Peer-to-Peer Electronic Cash System (or shortly called Bitcoin Blockchain), that completely removes the middleman and becomes an anonymous but safe system. Satoshi Nakamoto built a blockchain as a shared ledger since it stores digital money in a decentralized way because no single entity will have control over it. Moreover, with cryptography, people could register to the system by generating a wallet (public key-private key pair) without providing any personal information such as email address, full name, etc. When a person A sends Bitcoins to person B, transactions is cryptographically signed and throw into the transaction pool. The miners of the blockchain then put it in to a block and start mining and broadcast to the entire nodes in the Bitcoin blockchain network. They update their copy of the ledger, hence everyone in the network knows who owns what. Although Bitcoin is open for everyone, people want to do more than just sending and receiving money. Bitcoin’s system was difficult for people to code autonomous systems. If there is any system, like Bitcoin, but allows users to program and host code in the form of smart contracts then push to the blockchain and execute them, it could be a promising future for trust-less and decentralized applications becoming widespread.

There is a question about the advantages if this kind of system becomes possible. Definitely, it is decentralized, immutable, indelible, transparent but private.

**Decentralized**

The purpose of creating smart contracts is to rebuild a community exchange system that does not require a third-party such as bank, middle organizations, etc. Programmers can write a smart contract which contains conditions and put into a blockchain to make them completely distributed. This technique could remove the role of the middlemen or a third-party, which means no one is in control of others.

Are smart contracts trustful? Because they are stored on a blockchain they inherit some sophisticated properties. Smart contracts are immutable and distributed. Once a smart contract is created, it can never be changed. Hence, no one can go behind the back and edit the code of a smart contract. The output of a smart contract in a blockchain is validated by everyone on the network, which means nobody could force one smart contract to release a wanted output because other nodes on the network will spot this attempt and mark it as invalid. This ability is related to smart contract distributed property. Tampering with smart contracts becomes almost impossible. Smart contracts can be applied to many different things: on crowdfunding, automatic payments, processing claims of insurances, delivery payments and so on.

**Immutable and indelible**

Once transactions are committed to the blockchain, there is no going back to edit or delete it. The hash of a block in a blockchain can be comparable to a fingerprint. It identifies a block and all of its contents. Just as a fingerprint, a hash is always unique. Once a block is created, its hash is calculated. If there is any change inside the block, it will be easily detected because the hash of that block is also changed. Therefore, it no longer is the same block. In a blockchain, if any block is tampered, the hash of that block will be recalculated. In turn, that will make its adjacent block and all the following blocks invalid since they no longer store a valid hash of the previous one. Hence, changing a single block will cause all the hashes of other blocks to be recalculated in order to make the blockchain valid again.

Computers these days are so sophisticated that can produce hundreds of thousands of hashes per second. The recalculation of all hashes of other blocks as a consequence of tampering with a block can be done easily. Now, it is time to check for the Proof-of-work. A blockchain has a mechanism to automatically adjust the Difficulty Target in a block in order to remain the time required to calculate the appropriated block header hash. The valid proof-of-work is based on the correct header hash within a time period. For example, in Bitcoin blockchain, it takes at least 10 minutes to mine a new block. A Bitcoin mining software running on a node computer would automatically detect the performance of the hardware and adjust the Difficulty Target of the block. So that, the miner of that node could not mine a new block less than 10 minutes. Even if the miner successfully mine the block less than the time, that block is finally broadcast to the whole network for checking validity. If the Proof-of-work is cheated in mining process, it could not reach the Consensus across the whole network because there is no proof that the miner has worked on that block for a sufficient time. So if a person want to tamper with block N in a blockchain requiring 5 minutes to solve for the proof-of-work, he or she has to recalculate the hash of ( N-1) previous blocks for at least T = ( N-1)\*5 minutes, take control of more than 50% number of nodes on the network. By doing that, the tampered block could have a little chance to be validated. Besides, this is the parts where a dishonest miner can be found out. If there is any invalidation, the block will be unaccepted and the miner would have wasted his time and computing power. Messing with blockchain seems to be impossible!

**Transparent but private.**

All transactions on the blockchain are traceable, back to the very first genesis block. As a shared ledger, it is open everyone to view or audit. But how privacy and transparency can effectively coexist. The transaction can only show the sender and receiver public key, the data which is hashed and related financial information. There is nothing personal here. Therefore, it is total private but still transparent.

**The benefits of Ethereum mechanism**

Computing across the entire Ethereum network is not done to make everything more efficient. Reversely, this process makes computation on Ethereum far slower and more costly than on a computer. Every node runs the EVM in order to keep the consensus across the blockchain. This decentralized consensus provides Ethereum extreme levels of fault tolerance, no service interruption, and makes data stored on the blockchain forever unchangeable.

Ethereum, also a platform, is suited for applications that automate direct interaction between peers or facilitate coordinated group action across a network such as applications for coordinating peer-to-peer marketplaces, or the automation of complex financial contracts. While Bitcoin allows individuals to exchange cash without involving any intermediary, Ethereum could do more. It helps financial interactions or exchanges any complexity being carried out automatically and reliably using code running on Ethereum. Furthermore, any systems, which trust, security and permanence are important such as asset-registries, voting, governance, etc could be massively impacted by the Ethereum platform.

**2.Future work:**

Some promising improvements.

# LIST OF REFERENCES

# APPENDICES