

國立臺灣大學電機資訊學院資訊網路與多媒體研究所



碩士論文

Department or Graduate Institute of Networking and Multimedia

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於比特幣區塊鏈之分散式自治治理架構

Decentralized Autonomous Governance on Bitcoin

Blockchain

陳柏安

Po-An Chen

指導教授：廖世偉 博士

Advisor : Shih-Wei Liao, Ph.D.

中華民國 105 年 7 月

July 2016

中文摘要



在我的論文中描述了一分散式自治治理架構 (Decentralized Autonomous Governance(DAG)) 基於比特幣區塊鏈之設計與實作。一 DAG 會由一群人提供資金 (bitcoin) 共同發起，根據任何目標，並共同直接管理大家提供得資金 (bitcoin)。DAG 實際上是一個軟體，並在程式碼中詳細描述所有的治理規則。當一 DAG 被發起，所有的 bitcoins 會被保存在 DAG 中之一帳戶，每一位提供 bitcoins 的發起人會收到相對應得 Tokens 代表他們對此 DAG 擁有權之分額和投票權。該 Tokens 的擁有者可以直接和 DAG 互動，透過投票支持或否決一 bitcoins 使用申請。投票權之權重根據擁有該 Tokens 的數量。每個 Tokens 的擁有者都可以申請使用 bitcoins。只有在一使用申請被足夠多的 Tokens 擁有者同意支持後，bitcoins 才會轉移至申請者之帳戶。以上的過程和結果均會被記錄在 Bitcoin 區塊鏈 (blockchain)。我會在“GCOIN Layer”ⁱ上逐步得實作我的設計。

關鍵字：比特幣、區塊鏈、虛擬資產管理、治理架構、分散式自治組織

(Decentralized Autonomous Organization(DAO))、智能合約 (Smart Contract)

Abstract

In my thesis, I describe the design concepts and implementation for a Decentralized Autonomous Governance(DAG) on Bitcoin blockchain. A DAG, described in here, will be launched by a group of people who want to co-manage their bitcoins directly for any reason. DAG is actually a software and all governance rules are specified in codes.

When a DAG is launched, the whole bitcoins are saved in DAG and the people who providing bitcoins will receive another tokens for representing their shares and voting powers. The token-holders interact directly with DAG by voting for approving an application of bitcoins or not. The voting powers are corresponding to their amount of tokens. Every token-holders can apply for bitcoins. The bitcoins will transfer only when an application is approved by enough token-holders. All the processes above will be recorded on Bitcoin blockchain. In my thesis, I will implement my design on “GCOIN Layer” gradually.

Key words: Bitcoin, Blockchain, Virtual Asset Management, Governance Structure, Decentralized Autonomous Organization(DAO), Smart Contract

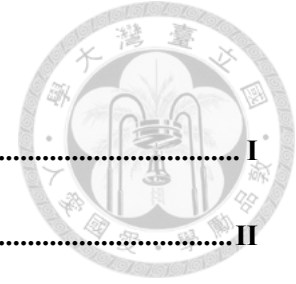
List of Figures

<i>fig 1. The structure of transaction</i>	4
<i>fig 2. Simple illustration of transaction's Input and Output</i>	4
<i>fig 3. Flow of applying and voting for a "License" which applying for bitcoins</i>	10
<i>fig 4. Flow of applying bitcoins through "License" and voting for it. Saving the bitcoins in a 2-of-3 MultiSig address</i>	11
<i>fig 5. An Alliance can illegally broadcast the HEX and sending bitcoins successfully (Alliances with unbalanced voting powers which are inconsistent with LICENSE_APPROVE_PERCENTAGE)</i>	12
<i>fig 6. Flow of applying bitcoins through "License" and voting for it. Saving the bitcoins in a 2-of-2 MultiSig address created by Oracle Alliances(Alliance 1 and 2)</i>	14
<i>fig 7. The structure of "License"</i>	20

List of Tables

<i>Table 1. Comparing of 4 models</i>	15
<i>Table 2. Steps for applying and voting for a new "License" and the corresponding bitcoin transaction's fee for 4 models</i>	17

Outline



中文摘要.....	I
ABSTRACT.....	II
LIST OF FIGURES	III
LIST OF TABLES.....	III
1 INTRODUCTION	1
1.1 RESEARCH BACKGROUND	1
1.2 RESEARCH MOTIVATION	1
2 BACKGROUND AND TECHNOLOGIES	3
2.1 BITCOIN AND BITCOIN BLOCKCHAIN	3
2.1.1 Bitcoin Transaction	3
2.1.2 Transaction Outputs and Inputs	4
2.1.3 Types of Bitcoin Transactions	5
2.2 GCOIN LAYER.....	5
2.3 DAO.....	6
2.4 SMART CONTRACT	6
3 DESIGN AND IMPLEMENTATION	7
3.1 ENHANCING GCOIN LAYER TO WORK AS A EXISTED HUMAN ORGANIZATION.....	7
3.2 MANAGING FUND(BITCOIN) INSIDE GCOIN LAYER	8
3.3 SAVING FUND(BITCOIN) IN A MULTI-SIGNATURE ADDRESS CREATED BY ALLIANCE ADDRESSES	10
3.4 “ORACLE ALLIANCE”	13
4 EXPERIMENT	16
4.1 EXPERIMENT ENVIRONMENT AND SETTINGS	16
4.2 EXPERIMENT RESULT	16

5 CONCLUSION AND FUTURE WORKS	18
--	-----------

REFERENCES.....	21
------------------------	-----------



1 Introduction



1.1 Research Background

When Satoshi Nakamoto first set the Bitcoin blockchain into motion in January 2009, he was simultaneously introducing two radical and untested concepts. The first is the bitcoin, a decentralized peer-to-peer online currency that maintains a value without any backing, intrinsic value or central issuer. So far, the bitcoin as a currency unit has taken up the bulk of the public attention, both in terms of the political aspects of a currency without a central bank and its extreme upward and downward volatility in price. However, there is also another, equally important, part to Satoshi's grand experiment: the concept of a proof of work-based blockchain to allow for public agreement on the order of transactions.ⁱⁱ

According to leverage the Bitcoin blockchain, there comes technologies that saving data in Bitcoin transactions and interpreting by their own nodes or wallets. Most of them are using for issuing and managing new virtual asset(or smart property) on top of Bitcoin blockchain. GCOIN Layer is one of the technologies and with defining governance structure. GCOIN Layer is governed under “Alliance” who have voting powers to co-decide things like whether to approve a new applying asset or a new “Alliance” joining in.

1.2 Research Motivation

Most of existed human organizations are formed by a group of people who providing funds(e.g. US Dollars) and governing according to pre-defined rules(e.g. laws). Historically, the main and fundamental problem is people do not always follow the rules. In the recent years, crowdfunding and peer-to-peer lending make it easier for common people to invest in new projects but also amplifying this problem when

platform or no-name entrepreneurial team are responsible for keeping and managing the funds. In addition, people can easily make mistakes according to complex and hierarchical governance rules. Bitcoin, as a new and easy-to-lose virtual currency, causing it more possible for making mistakes and more difficult to recover from the damages.

Our Goal is designing and implementing a governance model: (1) The governance rules are specified in codes. (2) People can co-manage their funds(bitcoins) through directly interacting with software. (3) The processes and results are recorded on Bitcoin blockchain in real-time transparently. We describe this model as a Decentralized Autonomous Governance(DAG) on Bitcoin blockchain.

We implement by extending GCOIN Layer to managing existed asset - bitcoin. In case, we will launch GCOIN Layer with multiple “Alliance” who acting like organization founder or investor that providing bitcoins and receiving corresponding tokens representing its shares and voting powers. The bitcoins are saved in GCOIN Layer and every members of the Layer can apply for the bitcoins. “Alliance” will vote for approving the application or not, and GCOIN Layer transferring bitcoins only according to the voting result. The rules and processes are enforced by GCOIN Layer and recoded on Bitcoin blockchain.

2 Background and Technologies



2.1 Bitcoin and Bitcoin blockchain

Bitcoin is a collection of concepts and technologies that form the basis of a digital money ecosystem. Units of currency called bitcoins are used to store and transmit value among participants in the Bitcoin network. Bitcoin users communicate with each other using the Bitcoin protocol primarily via the Internet, although other transport networks can also be used. The bitcoin protocol stack, available as open source software, can be run on a wide range of computing devices, including laptops and smartphones, making the technology easily accessible.ⁱⁱⁱ

Bitcoin was first described in 2008 by Satoshi Nakamoto's white paper and the first reference implementation was open source in 2009. The implementation is developing continuously by Bitcoin developer community and is the most commonly used Bitcoin software with full Bitcoin functionalities right now. GCOIN Layer and our implementation are all developed based on the codebase.

2.1.1 Bitcoin Transaction

A transaction is a data structure that encodes a transfer of value from a source of fund, called an input, to a destination, called an output. Transaction inputs and outputs are not related to accounts or identities. Instead you should think of them as bitcoin amounts, chunks of bitcoin, being locked with a specific secret which only the owner, or person who knows the secret, can unlock.

A transaction contains a number of fields, as follows:



Table 5-1. The structure of a transaction

Size	Field	Description
4 bytes	Version	Specifies which rules this transaction follows
1-9 bytes (VarInt)	Input Counter	How many inputs are included
Variable	Inputs	One or more Transaction Inputs
1-9 bytes (VarInt)	Output Counter	How many outputs are included
Variable	Outputs	One or more Transaction Outputs
4 bytes	Locktime	A unix timestamp or block number

fig 1. The structure of transaction^{iv}

2.1.2 Transaction Outputs and Inputs

The fundamental building block of a bitcoin transaction is an unspent transaction output or UTXO. UTXO are indivisible chunks of bitcoin currency locked to a specific owner, recorded on the blockchain, and recognized as currency units by the entire network.^v

The UTXO consumed by a transaction are called transaction inputs, while the UTXO created by a transaction are called transaction outputs. This way, chunks of bitcoin value move forward from owner to owner in a chain of transactions consuming and creating UTXO. Transactions consume UTXO unlocking it with the signature of the current owner and create UTXO locking it to the bitcoin address of the new owner.^{vi} (There existed exception called the coinbase transaction.)

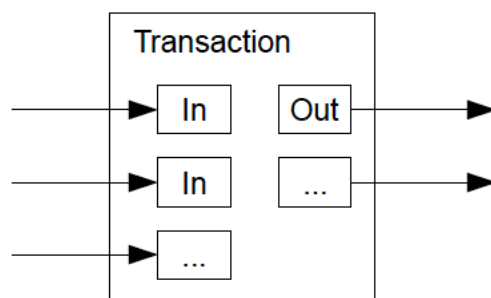


fig 2. Simple illustration of transaction's Input and Output^{vii}

2.1.3 Types of Bitcoin Transactions

There are five types of standard Bitcoin transactions. (1) P2PKH. Using to pay bitcoin to a public key hash, commonly known as a “Bitcoin address”. These contain a locking script that encumbers the output with a “Bitcoin address”. An output locked by a P2PKH script can be unlocked (spent) by presenting a public key and a digital signature created by the corresponding private key.^{viii} (2) P2SH. It’s same usage as P2PKH but with the public key itself is store in the output. Not commonly using at present. (3) Multi-signature. Where N public keys are recorded in the script and at least M of those must provide signatures to unlock the output and spend the bitcoin. (4) OP_RETURN. It contains 80 bytes for arbitrary data in the output but this output can’t be spent. (5) P2SH. It is able to contain arbitrary script(“Redeem Script”) which is replaced by a hash(“Redeem Script Hash”) in the output and providing a more convenient way to using Multi-signature.

2.2 GCOIN Layer

GCOIN Layer is building based on a virtual asset issuing and managing technology on top of Bitcoin blockchain, called “Omni Layer”, and extending it with defining governance structure. The main features including: (1) A “License” which defines a permission to issue one kind of virtual asset(smart property) and specifying the information and attributes of the asset including a unique “License ID”. Member of GCOIN Layer can apply for a “License” through an “applying transaction”, which is a type of transaction defined in GCOIN Layer, and managing by other type of transactions(e.g. “granting transaction” for issuing some amount of assets). Before granting a “License”, the application should be approved by Alliances. (2) An “Alliance” has voting power in GCOIN Layer and voting through a “voting transaction”, which specifying the “License ID” that is voting for and

intention(approved or not) of the “Alliance”. An “Alliance” is responsible for running GCOIN Layer software, and continuously receiving new transactions and executing according to transaction’s logic. (3) The voting system for making consensus between “Alliance” is implemented and recording though Bitcoin blockchain. Things like deciding a new “Alliance” can also making use of this voting system.

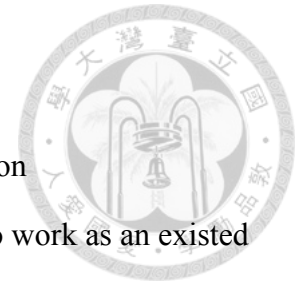
2.3 DAO

“DAO” providing a method that for the first time allows the creation of organizations in which (1) participants maintain direct real-time control of contributed fund and (2) governance rules are formalized, automated and enforced using software.^{ix} Furthermore, providing a solution to the “Majority Robbing the Minority Attack” problem, the “DAO split”, which the minority owner can design to spilt from the “DAO” to a “child DAO” that taking back his or her fund and will not disturb by the decisions of the original “DAO” . It using Ethereum a blockchain technology which integrates a Turing complete programming language with “Smart Contract” processing functionality.^x The concept of “DAO” is somewhat similar to our goal and it is already launched, so we take it as an important reference for our design and implementation.

2.4 Smart Contract

“Smart Contract” is firstly described by Nick Szabo in 1999. For a short answer, it is a program or machine automatically enforce the rules and move value securely. A real life example is a vending machine. In addition, there need a “Oracle” which is responsible for providing data as input that triggering “Smart Contract” to execute the rules and transferring value. For the example of vending machine, people who throwing coins and deciding the good to buy is the “Oracle”. For clarifying, “Smart Contract” is a feature that not every blockchain technologies will provide.

3 Design and Implementation



3.1 Enhancing GCOIN Layer to work as a existed human organization

For the first goal, we would like to enhance GCOIN Layer to work as an existed human organization with two main features: (1) Alliances with weighted voting shares and (2) Launching GCOIN Layer with multiple Alliances.

At first, we define a new virtual asset called “GCOIN Token”(“License id” = 1) to represent the shares of Alliances. The amounts of “GCOIN Token” owned by Alliances represent the corresponding voting powers.

In a “License”, “Approve Threshold” represents how many votes from Alliances this “License” needing to be approved, and “Approve Counter” records the amount of votes a “License” currently received. “Approve Threshold” is decided in applying “License” stage and according to the formula in codes:

```
approveThreshold =  
(round(TOTAL_ALLIANCE_NUMBER*LICENSE_APPROVE_PERCENTAGE)) ;
```

For letting “Approve Threshold” reflect the true threshold when Alliances have weighted voting powers. We modify the formula to:

```
approveThreshold =  
(round(TOTAL_REWARD_TOKENS_OWNED_BY_ALLIANCES*LICENSE_APPROVE_PERCENTAGE)) ;
```

When GCOIN Layer receives new “voting transaction” from an Alliance, it will add the Alliance’s “GCOIN Token” amount to “Approve Counter” and checking whether “Approve Counter” exceed or equal to “Approve Threshold”. If the result is yes, the “License” is “approved” and can be granted.

When launching GCOIN Layer, the total “GCOIN Token” initial amount is needed to decide. After that, we can define multiple Alliances in the code which including Alliance information and most important the Alliance’s address and its “GCOIN Token” amount. The sum of every Alliance’s “GCOIN Token” can’t exceed the total amount that is firstly decided.

3.2 Managing fund(bitcoin) inside GCOIN Layer

Our second goal is to managing fund, which is existed asset on Bitcoin blockchain – bitcoin, inside GCOIN Layer. Just like an existed human organization which composed of two main parts: (1) pre-defined rules for a group of people and (2) a set of assets, e.g. company, foundation or institute. We would like to regard the initial Alliance as organization’s founder and investor who providing funds(bitcoins) and receiving corresponding amount of “GCOIN Token” representing its shares and voting powers. Saving the bitcoins in an account and according to the applying and voting result, GCOIN Layer will transfer designated amount of bitcoins from the account.

At first, one of Alliance’s address should be chosen to keep bitcoins and recording it in GCOIN Layer before launching. The flow to using the saving bitcoins will be: (1) Member with an address can apply for bitcoins according to specified usage. (2) Alliances voting to the application. (3) If the voting result is “Approved”, GCOIN Layer automatically transfers the amount of bitcoins for application from the bitcoin saving address to applicant’s address. (4) At the same time of step (3), keep tracking the bitcoin transferring transaction. (5) When the transaction is confirmed, GCOIN Layer will notice and update the “License” with the confirmed transaction id(txid) for transferring bitcoins.

For step (1), we let applicant applying for bitcoins through applying “License”, so we extend “License” with new two fields: (i) The amount of bitcoins for application and (ii) A transaction id(txid) for the transaction successfully transferring the application fund.

For step (3) and (4), GCOIN Layer is responsible for making a bitcoin transaction that one output(P2PKH) for sending the bitcoins to applicant’s address and another output(OP_RETURN) including a “payload” which containing information to point this transaction as the one sending bitcoins for the applying “License”. In addition, mostly there will need an output(P2PKH) for giving change. After the transaction is made, broadcasting it and recording it in local database for preventing redundantly resending bitcoins due to any reason that causes GCOIN Layer re-scan previous bitcoin transactions.

The “payload” described above actually contains a transaction defining in GCOIN Layer. We called it a “Recording Transaction” and it including three messages: (i) The applicant’s address (ii) The amount of applying bitcoins and (iii) the applying “License ID”. When the transferring bitcoin transaction is confirmed, GCOIN Layer will parse it and finding the “Recording Transaction” which containing in one of the outputs and then comparing the information(applicant’s address and amount of applying bitcoins) saving in the “Recording Transaction” with the output actually sending bitcoins. If matched, updating the “License”, which “License ID” specified in “Recording Transaction”, with this “txid”. The whole flow of applying and voting for a “License” which applying for bitcoins is illustrated in Figure 3.

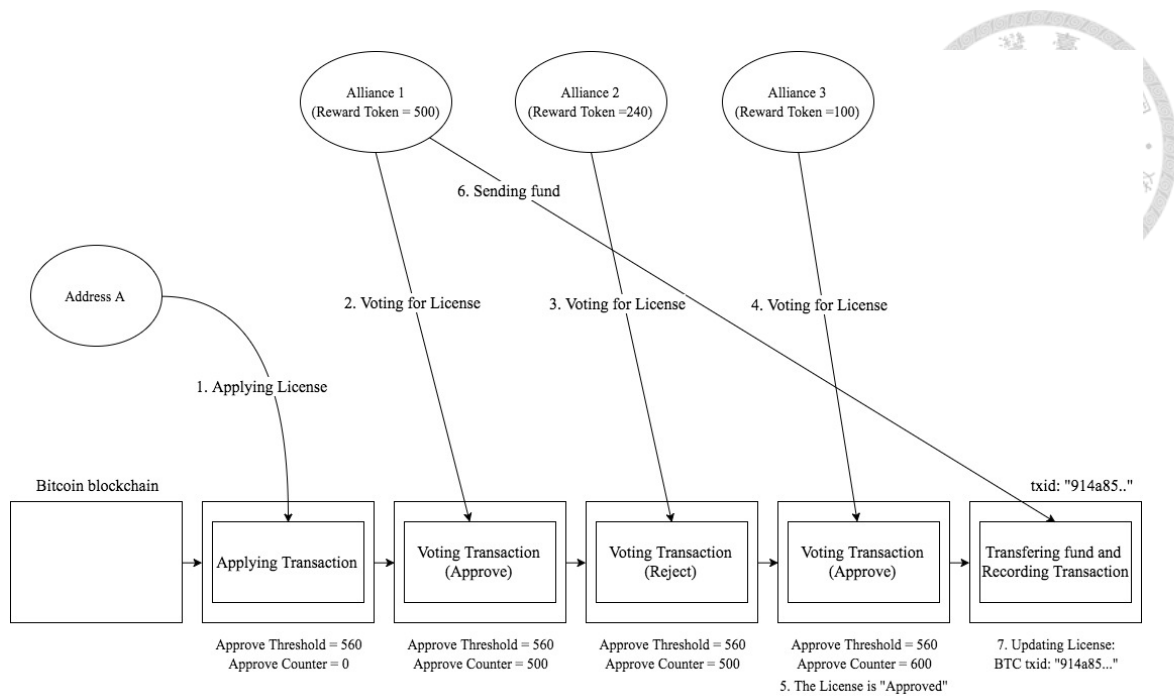


fig 3. Flow of applying and voting for a "License" which applying for bitcoins

3.3 Saving fund(bitcoin) in a Multi-Signature address created by Alliance addresses

However, in previous model that saving bitcoins in one address is actually not secure enough according to: (1) Unfaithful of the chosen Alliance and (2) It's a single point of failure. (This may can be diminished by more secure mechanism for saving bitcoins.) Therefore, we are thinking of improving this situation and make it more decentralized. If we saving bitcoins in a Multi-Signature(MultiSig) address created by Alliance addresses, we will need enough signatures from Alliance's private keys to transfer the bitcoins. For example, 3 Alliances decide the LICENSE_APPROVE_PERCENTAGE to be 2/3 so they create a 2-of-3 MultiSig address by their addresses which will need any 2 of their signatures including in a transaction to spend the bitcoins in these MultiSig address. This is much better than saving bitcoins in arbitrary one Alliance. Therefore, our third goal is managing fund(bitcoin), saving in a MultiSig address created by Alliance addresses, inside GCOIN Layer.

First of all, creating a MultiSig address can be easily done by bitcoin core RPC. Just specifying the “M” (in M-of-N) and the addresses or public keys for the RPC and then it will return a MultiSig address. After that, Alliances saving their funds(bitcoins) in the MultiSig address. To transfer the bitcoins, someone needs to create a transaction spending UTXO from MultiSig address and sending bitcoins to applicant’s address. After that, letting this transaction signed by enough Alliances and including signatures in the transaction. In the end, broadcasting to Bitcoin network.

We let “License” applicant creates the transaction and following the processes:

- (1) Selecting UTXOs of the MultiSig address that have enough bitcoins.
- (2) Creating an output sending bitcoins to applicant’s address.
- (3) Deciding fee, basing on the expected transaction size, and creating an output for giving change to MultiSig address.
- (4) Creating an output that including a “Recording Transaction”. Finally, it will produce a string called HEX. After that, including the HEX in the applying “License”.

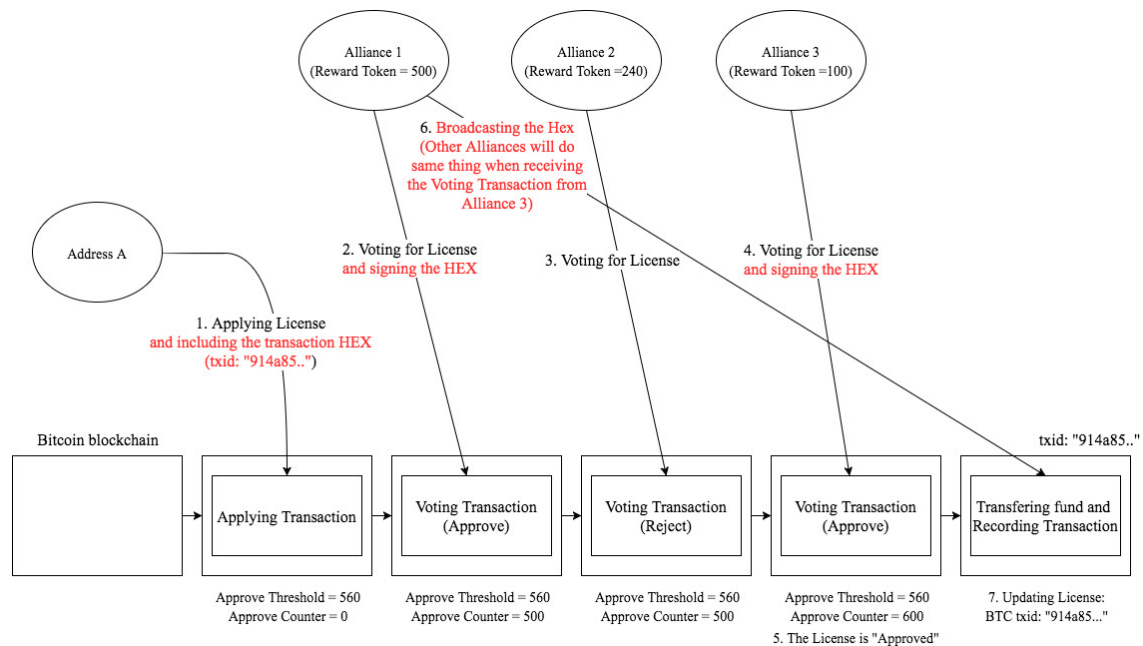


fig 4. Flow of applying bitcoins through “License” and voting for it. Saving the bitcoins in a 2-of-3 MultiSig address

However, there existed limitation of capacity for MultiSig transaction. The “Redeem Script” of P2SH type transaction has the maximum size of 520 bytes. This lead to the limitation of only 15 compressed public key can be included and so the maximum number of “N” is 15. On the other hand, the resulting transaction HEX should be able to fit in GCOIN Layer’s transaction payload but after calculating it will work fine for a 15-of-15 P2SH Bitcoin transaction. (An example of 15-of-15 P2SH bitcoin transaction is 1699 bytes^{xi}. However, GCOIN Layer has total 7,650 bytes maximum actual transaction data storage per Omni Class B transaction.) Therefore, under this mechanism the maximum Alliance number is 15 if every Alliances have same voting powers.

There is also existed an issue for the voting powers of Alliances are unbalanced and is inconsistent of LICENSE_APPROVE_PERCENTAGE as illustrated below:

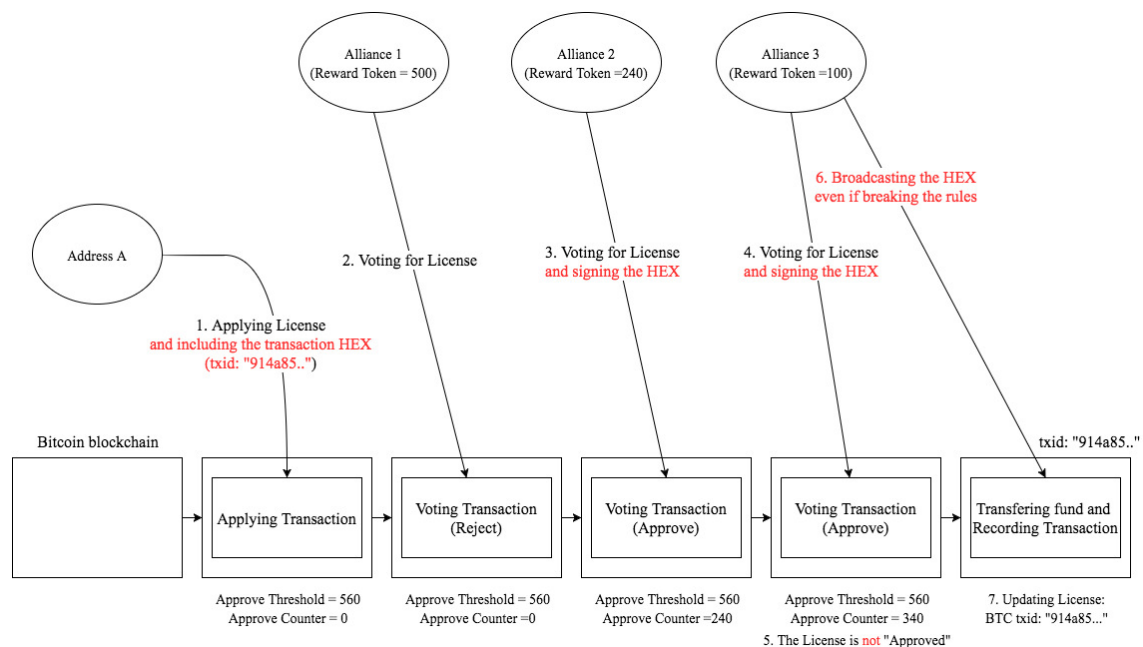


fig 5. An Alliance can illegally broadcast the HEX and sending bitcoins successfully (Alliances with unbalanced voting powers which are inconsistent with LICENSE_APPROVE_PERCENTAGE)

For common case, the “License” is not “Approved” so the HEX will not be broadcast even if it has enough signatures. However, if Alliance 2 or 3 break the rules and

broadcasting the HEX, the fund will be transferred successfully. This become a flaw that may caused huge damage. One solution is an Alliance can hold several public keys that are corresponding to its voting powers and creating a MultiSig address by several public keys owned by multiple Alliances, but with the limitation of “N” up to 15, this solution is impractical.

3.4 “Oracle Alliance”

According to deal with the above issue and mitigating limitation of Alliance numbers. We design new policy that choosing some of the Alliances for keeping fund(bitcoin) and responsible for signing the transaction HEX. The chosen Alliances are acting like “Oracle”. They signed the HEX and including in the “voting transaction” when they making sure the voting result for a “License” is “Approved”. (If The “GCOIN Token” owned by “Oracle Alliance” exceed “Approve Threshold”, they can sign whenever they are “Approved”). Therefore, we called them “Oracle Alliance” and our fourth goal is to manage fund(bitcoin), saving in a MultiSig address created by “Oracle Alliance”, inside GCOIN Layer.

As we mentioned above, the different is in policy. Before the GCOIN Layer launch, the initial Alliances need to co-decide who to be “Oracle Alliance”. The maximum number of “Oracle Alliance” can be arbitrary number that under 15. After that, creating MultiSig address by “Oracle Alliance” addresses and saving the bitcoins in it. The “M” of MultiSig address can also be arbitrary under the number of “Oracle Alliance” and might deciding according to the trade off between security and efficiency. (The more secure, the larger “M”.)

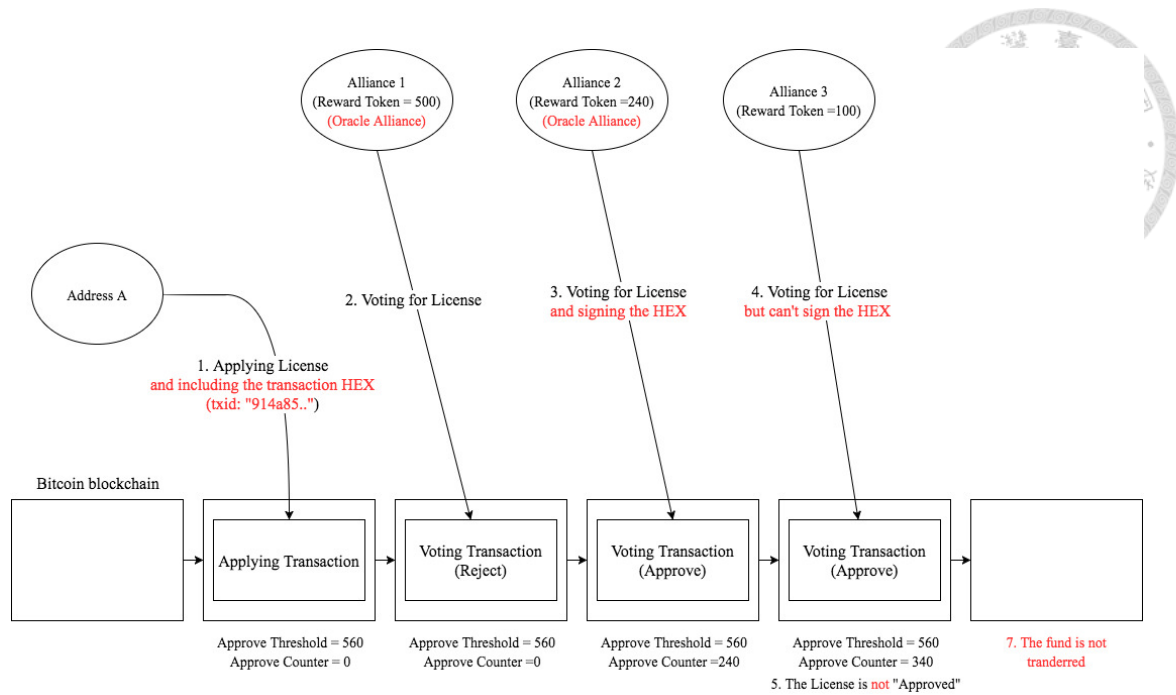
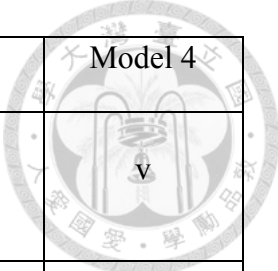


fig 6. Flow of applying bitcoins through "License" and voting for it. Saving the bitcoins in a 2-of-2 MultiSig address created by Oracle Alliances(Alliance 1 and 2)

Figure 6 illustrating how this policy change can prevent the issue happening in Figure 5. The fund(bitcoin) will transfer successfully only when all "Oracle Alliance" sign the HEX. On the other hand, this model rely on the trust of "Oracle Alliance". About the capacity, although the maximum "Oracle Alliance" can only be 15, the total number of Alliance can be arbitrary. In addition, when a new Alliance joins in, there is no need to change MultiSig address and moving bitcoins to a new address. This will happen only when deciding new "Oracle Alliance" or kicking out old "Oracle Alliance".

Our 4 design goals lead to 4 models: (1) Model 1: Enhancing GCOIN Layer to work more like existed human organization (2) Model 2: Managing fund(existed property like bitcoin) inside the organization (3) Model 3: Managing fund(bitcoin), saving in a MultiSig address created by all Alliances, inside the organization (4) Model 4: Managing fund(bitcoin), saving in a MultiSig address created by "Oracle Alliance", inside the organization.



Features	Model 1	Model 2	Model 3	Model 4
Launching Layer with Multiple Alliances	v	v	v	v
Weighted voting powers	v	v	v	v
Managing fund(bitcoin) inside	x	v	v	v
The fund(bitcoin) saving account	x	One Alliance address	MultiSig address created by all Alliances	MultiSig address created by “Oracle Alliances”
Oracle Alliance	x	x	x	v
Alliance Capacity	No	No	15	No (Oracle Alliance up to 15)

Table 1. Comparing of 4 models

4 Experiment



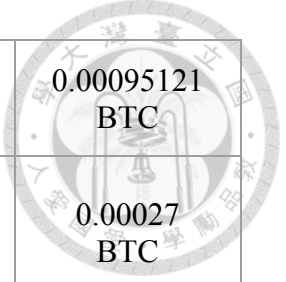
4.1 Experiment environment and settings

- Bitcoin core version 0.10.04
- Omni Layer version 0.0.10
- Running on Bitcoin “testnet”
- GCOIN Layer setting:
 - Alliances amount = 3
 - “GCOIN Token” distribution = [“Alliance 1”:500, “Alliance 2”:240, “Alliance 3”:100]
 - Alliances 1 and 2 are “Oracle Alliance” in Model 4
 - LICENSE_APPROVE_PERCENTAGE = 2/3

4.2 Experiment result

Running 3 times with same flow per model and listing the corresponding transaction’s fee after averaging:

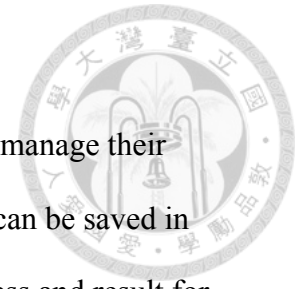
Steps	Model 1	Model 2	Model 3	Model 4
Applying License	0.00014085 BTC	0.0001439 BTC	0.00063535 BTC	0.00048536 BTC
Alliance 1 Voting	0.00013658 BTC	0.00013658 BTC	0.00090243 BTC	0.00022743 BTC
Alliance 2 Voting	0.00013658 BTC	0.00013719 BTC	0.00090243 BTC	0.00013719 BTC
Sending fund and Recording	x	0.0002445 BTC	0.0002439 BTC	x
Alliance 1 Signing	x	x	x	0.00081218 BTC



Alliance 2 Signing	x	x	x	0.00095121 BTC
Sending fund and Recording	x	x	x	0.00027 BTC
Total	0.00041401 BTC	0.00066217 BTC	0.00275301 BTC	0.00288337 BTC

Table 2. Steps for applying and voting for a new “License” and the corresponding bitcoin transaction’s fee for 4 models

5 Conclusion and Future works



Our design and implementation can let a group of people co-manage their fund(bitcoin) according to pre-defined rules and their fund(bitcoin) can be saved in different way. The rules are specified in GCOIN Layer and the process and result for fund(bitcoin) managing are all enforced and recorded by Bitcoin blockchain.

Comparing to “DAO”, which can actually do more things that including the whole life-cycle of an organization like crowdfunding and “DAO Reward Token” for presenting the right for gaining future revenue, the main difference is about how to deal with “Majority Robbing the Minority Attack”. For GCOIN Layer, it is a feature that the Majority can always lead the voting result. However, “DAO” treats it as a bug and providing a solution that the minority can decide to split from original “DAO” and protecting their own fund(ether).^{xii}


For future works and some remaining issues:

- (1) Testing on Bitcoin “mainnet”
- (2) When multiple applicants referring same UTXO for same Multi-Signature address, if one “License” is approved and spend the UTXO successfully, the others will be invalid. Therefore, the applying process should be suspended or at least creating a new transaction for sending fund. One solution for mitigating this issue can be saving fund(bitcoin) with multiple UTXOs and recording the UTXOs which are now being used for applying.
- (3) If the the voting transactions including the Alliance’s signature are in same block, No matter how, there should be some of the Alliances sign and vote again. The votes from same Alliance will not increase “Voting Counter”, but updating the signed transaction HEX.

- (4) The amount of “GCOIN Token” now is fixed if GCOIN Layer is launched.

However, it can also be defined as a “License” that can be granted in the future. We are deciding to deal with it by letting Alliances can make proposal for granting new “GCOIN Token” or the distribution of them and other Alliances will vote for approving the proposal or not.

- (5) The last one is to do more researches on how Bitcoin blockchain fork will affect GCOIN layer. When a fork is happened, it may cause inconsistent of the latest states of Bitcoin blockchain between different nodes. Without enough experiments, we can wait for enough confirmations to mitigating this issue(at least 6 as same suggestion for a simple Bitcoin payment^{xiii}).



Data Type	Field name	Description
std::string	Address	Alliance's address
std::string	name	Alliance's name
std::string	url	Website URL
std::string	data	Description of Alliance
uint256	txid	Txid of Applying transaction
uint32	approve_threshold	Approve Threshold
uint32	approve_count	Approve Counter
uint32	status	Approved/pending/rejected
uint32	money_application	Number of bitcoins for application

fig 7. The structure of "License"

References



- ⁱ Chun-Wei Chang, Governance structure for virtual asset issuance and management on bitcoin-like Blockchain [2016]. P.3
- ⁱⁱ Vitalik Buterin, Ethereum White paper: A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM [2014]. p.1
- ⁱⁱⁱ Andreas M. Antonopoulos, Mastering Bitcoin - Unlocking Digital Cryptocurrencies [2014]. p.1
- ^{iv} Andreas M. Antonopoulos, Mastering Bitcoin - Unlocking Digital Cryptocurrencies [2014]. p.111
- ^v Andreas M. Antonopoulos, Mastering Bitcoin - Unlocking Digital Cryptocurrencies [2014]. p. 112
- ^{vi} Andreas M. Antonopoulos, Mastering Bitcoin - Unlocking Digital Cryptocurrencies [2014]. p.113
- ^{vii} Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System [2008]. p.5
- ^{viii} Andreas M. Antonopoulos, Mastering Bitcoin - Unlocking Digital Cryptocurrencies [2014]. p.128
- ^{ix} CHRISTOPH JENTZSCH, DECENTRALIZED AUTONOMOUS ORGANIZATION TO AUTOMATE GOVERNANCE FINAL DRAFT - UNDER REVIEW [2016]. p.1
- ^x CHRISTOPH JENTZSCH, DECENTRALIZED AUTONOMOUS ORGANIZATION TO AUTOMATE GOVERNANCE FINAL DRAFT - UNDER REVIEW [2016]. p.1
- ^{xi} Blockchain.info. Transaction
552026dade1c9385e4693a4e82f07080d8d1950fc822346f95a0dc1e0a833465 [2014]. URL:
<https://blockchain.info/tx/552026dade1c9385e4693a4e82f07080d8d1950fc822346f95a0dc1e0a833465>
- ^{xii} Vitalik Buterin, DAOs, DACs, DAs and More: An Incomplete Terminology Guide [2014].
URL: <https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/>
- ^{xiii} Bitcoin.it. Confirmation. Retrieved: 14 June 2016.
URL: <https://en.bitcoin.it/wiki/Confirmation>