A P2P Networking Simulation Framework For Blockchain Studies

Bozhi Wang

A thesis in fulfillment of the requirements for the degree of

Doctor of Philosophy

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

June 2021

Thesis submission for the degree of Doctor of Philosophy

Thesis Title and Abstract | Declarations | Inclusion of Publications Statement | Corrected Thesis and Responses

Thesis Title

A P2P networking simulation framework for blockchain studies

Thesis Abstract

Recently, blockchain becomes a disruptive technology of building distributed applications (DApps). Many researchers and institutions have devoted their resources to the development of more effective blockchain technologies and innovative applications. However, with the limitation of computing power and financial resources, it is hard for researchers to deploy and test their blockchain innovations in a large-scape physical network.

Hence, in this dissertation, we proposed a peer-to-peer (P2P) networking simulation framework, which allows to deploy and test (simulate) a large-scale blockchain system with thousands of nodes in one single computer. We systematically reviewed existing research and techniques of blockchain simulator and evaluated their advantages and disadvantages.

To achieve generality and flexibility, our simulation framework lays the foundation for simulating blockchain network with different scales and protocols. We verified our simulation framework by deploying the most famous three blockchain systems (Bitcoin, Ethereum and IOTA) in our simulation framework.

We demonstrated the effectiveness of our simulation framework with the following three case studies: (a) Improve the performance of blockchain by changing key parameters or deploying new directed acyclic graph (DAG) structure protocol; (b) Test and analyze the attack response of Tangle-based blockchain (IOTA) (c) Establish and deploy a new smart grid bidding system for demand side in our simulation framework.

This dissertation also points out a series of open issues for future research.

Thesis submission for the degree of Doctor of Philosophy

Thesis Title and Abstract | Declarations | Inclusion of Publications Statement | Corrected Thesis and Responses

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

☑ I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Thesis Title and Abstract | Declarations | **Inclusion of Publications Statement** | Corrected Thesis and Responses

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed **greater than 50%** of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

☑ The candidate has declared that **their thesis has publications - either published or submitted for publication - incorporated into it in lieu of a Chapter/s. Details of these publications are provided below.**.

Publication Details #1

Full Title:	A simulation approach for studying behavior and quality of blockchain networks
Authors:	Bozhi Wang, Shiping Chen, Lina Yao, Bin Liu, Xiwei Xu, Liming Zhu
Journal or Book Name:	International Conference on Blockchain
Volume/Page Numbers:	18-31
Date Accepted/Published:	2018
Status:	published
The Candidate's Contribution to the Work:	Model establishment. Code writing. Analysis. Paper writing.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	Part of Chapter 3. Part of Chapter 4. Chapter 5.1.

Publication Details #2

Full Title:	Security analysis on tangle-based blockchain through simulation
Authors:	Bozhi Wang, Qin Wang, Shiping Chen, Yang Xiang
Journal or Book Name:	Australasian Conference on Information Security and Privacy
Volume/Page Numbers:	653-663
Date Accepted/Published:	2020
Status:	published
The Candidate's Contribution to the Work:	Model establishment. Code writing. Part of Analysis. Part of paper writing.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	Chapter 6

Publication Details #3

Full Title:	ChainSim: A P2P Blockchain Simulation Framework
Authors:	Bozhi Wang, Shiping Chen, Lina Yao, Qin Wang
Journal or Book Name:	CCF China Blockchain Conference
Volume/Page Numbers:	1-16
Date Accepted/Published:	2020
Status:	published
The Candidate's Contribution to the Work:	Model establishment. Code writing. Analysis. Paper writing.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	Chapter 3. Chapter 4.

Candidate's Declaration

I confirm that where I have used a publication in lieu of a chapter, the listed publication(s) above meet(s) the requirements to be included in the thesis. I also declare that I have complied with the Thesis Examination Procedure.

Abstract

Recently, blockchain becomes a disruptive technology of building distributed applications (DApps). Many researchers and institutions have devoted their resources to the development of more effective blockchain technologies and innovative applications. However, with the limitation of computing power and financial resources, it is hard for researchers to deploy and test their blockchain innovations in a large-scape physical network.

Hence, in this dissertation, we proposed a peer-to-peer (P2P) networking simulation framework, which allows to deploy and test (simulate) a large-scale blockchain system with thousands of nodes in one single computer. We systematically reviewed existing research and techniques of blockchain simulator and evaluated their advantages and disadvantages.

To achieve generality and flexibility, our simulation framework lays the foundation for simulating blockchain network with different scales and protocols. We verified our simulation framework by deploying the most famous three blockchain systems (Bitcoin, Ethereum and IOTA) in our simulation framework.

We demonstrated the effectiveness of our simulation framework with the following three case studies: (a) Improve the performance of blockchain by changing key parameters or deploying new directed acyclic graph (DAG) structure protocol; (b) Test and analyze the attack response of Tangle-based blockchain (IOTA) (c) Establish and deploy a new smart grid bidding system for demand side in our simulation framework.

This dissertation also points out a series of open issues for future research.

Acknowledgement

There are numerous people offered help during my candidature in UNSW Sydney and CSIRO Data 61, and this dissertation would not have been possible without their contributions.

First of all, I am beholden to my primary academic supervisor, Dr. Shiping Chen for his support and guidance throughout my PhD study. It is August 2017 that we first met, and since then he gave me great freedom in terms of research, provided me all kind of resources, while spent generous amount of time in supporting and guiding me.

In addition, I am grateful for my secondary supervisor Dr. Lina Yao for her advice and supporting, no matter in work or life. Her supporting include also sharing computing resources like a group shared GPU-accelerated computing server Dr. Strange which tremendously speed up my experiments.

The teammates in Lina's group: Xiang Zhang, Chaoran Huang, Lei Bai and Zhe Liu gave me many help in my PhD study though we worked in different fields, especially Xiang Zhang, who encouraged me in some dilemma.

I would also thank my cooperators and other co-authors: Sinkuang Lo, Qin Wang and Fangfei Zhang for the work produced and joy shared together. For COVID-19, some work has been delayed, but I'm very happy that we've come through these difficulties together.

My friends are also very supportive, especially: Ziyi Tao, Xinyi Yu and Yichun Hu.

Finally, while the most, this dissertation is dedicated to my parents, Wei Wang and Xiaotao Song for their love, support and encouragement. They have given me freedom to explorer since childhood, trained me to think individually, and unconditionally supported me. They are also the source of my power, driving me pursuing my goal.

Abbreviations

AMI Advanced Metering Infrastructure

B2B Business to Business

B2C Business to Consumer

BESS Battery Energy Storage System

BFT Byzantine Fault Tolerance

BTC Bitcoin

C2B Consumer to Business

C2C Consumer to Consumer

CA Certificate Authority

CA Contract Account

CEMS Community Energy Management System

DAG Directed Acyclic Graph

DApps Decentralized applications

DPoS Delegated Proof of Stake

DS Double Spending Attack

ECC Elliptic Curve Encryption Algorithm

EOA Externally Owned Account

EVM Ethereum Virtual Machine

GMT Greenwich Mean Time

HAN Home Area Network

HB Hybird Attack

HEMS Home Energy Management System

IoT Internet of Things

IoTA Internet of Things Application

LTC Litecoin

MCMC Markov Chain Monte Carlo Algorithm

NAN Neighborhood Area Network

P2P Peer to Peer

PBFT Practical Byzantine Fault Tolerance

PKI Public Key Infrastructure

PLC Power Line Carrier

PoE Proof of Existence

PoS Proof of Stake

PoW Proof of Work

PRC Remote Procedure Call

PS Parasite Attack

PV Photovoltaic

QoB Quality of Blockchain

QoS Quality of Service

SDN Software Defined Networking

SDT Secure Distributed Trading mechanism

SegWit Segregated Witness

SHA Secure Hash Algorithm

SPV Simplified Payment Verification

TCP/IP Transmission Control Protocol/Internet Protocol

TPS Transaction per second

TTP Trusted Third Party

Tx Transaction

UI User Interface

USDT Understanding Tether

UTXO Unspent Transaction Outputs

WAN Wide Area Network

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

For the past few years, people could do almost everything on the Internet. However, most of the online transactions from person to person relied on large intermediaries, such as email services, banks and telecommunications operators, which brought increasing problems to centralized systems. For example, our privacy was under infringement; sending money overseas took days and cost a lot. Things started to change in 2008, when Satoshi Noakmoto published the paper “Bitcoin: A Peer-to-Peer Electronic Cash System”[1]. The key technology of Bitcoin, called Blockchain, shows the strength in the future Internet of enabling decentralization to become a new paradigm for large-scale distributed systems. Blockchain is a peer-to-peer distributed ledger database, which consists of connected data blocks. The connection pointer between blocks is the Header Hash, which is processed by cryptographic hash function that protects the transactions in every block and the connections between blocks[2]. Thus, none of the historical transaction can be changed without invalidating a chain of Header Hash. While TCP/IP[3] is the communicating protocol between computers, Blockchain is the trust mechanism and cooperation protocol. The unique features offered by blockchain, such as decentralization, immutability and reliability, enable people to conduct transactions without relying on a trusted third party. From cryptocurrency[4] to cross-border payment, distributed storage[5], and supply chain management[6], more and more fields are going to take blockchain as the fundamental technology[7].

1.1 Motivations

Before Blockchain being widely adopted in other areas, a number of issues need to be solved. First, the current mining (PoW[47]- Proof of Work) mechanism of Blockchain are wasting a large amount of computing power on computation that is necessary for PoW but meaningless and costly, which is quite power inefficient: It was reported that the Bitcoin network consumes about 73 TWh per year, i.e., on average using 640 KWh per transaction. Second, every node in the network has a complete ledger. As time passing by, the ledger will become larger and larger[8]. And when a miner verifies a transaction, it requires to track all the historical transactions recorded on the blockchain, making the performance issue worse. At the same time, its scalability is poor: Each block can store 1 MB data, which means the system throughput is limited by the size of its block. Then, though Blockchain is an anonymous system, all the transactions are publicly available, which may cause privacy issue.[9] Also, in the current Bitcoin network, a common security strategy is to wait six blocks to confirm the transactions in the last block, which lasts around one hour. During this time, the forks will be cut. Lastly, the system could withstand double spending attack only if the number of honest nodes is more than 51 percent, which significantly limits the adoption and applications of the current Blockchain technologies.

To meet the needs of the development of various applications, several decentralized applications (DApps) platforms have been published[10], such as Ethereum[48] and Fabric[96]. In Ethereum, anyone can “upload” DApps to Ethereum and these DApps will always run as programmed. This enables people to develop varieties of decentralized financial applications without any single organization or person controlling them. The running program employs a native cryptocurrency called Ether (ETH) which is mined in the way similar to Bitcoin is. The consensus mechanism used in Ethereum is proof of work (PoW), and will become proof of stake (PoS)[11] in the coming future. Fabric is the other famous DApps platform that provides a modular and extendable architecture. While Ethereum uses anonymous authentication, Fabric supports digital certificate identity authentication form. Also, unlike ETH, there is no general token in Fabric since it mostly focuses on consortium blockchain that used within a group of semi-trust organizations. Fabric is based on several consensus with Practical Byzantine Fault Tolerance (PBFT)[46] being the main one.

In addition to the DApps platform, a lot of new ideas burst out to address the issues in Bitcoin blockchain. One is to use DAG (Direct Acyclic Graph)[12] aiming at improving the performance and scalability of Blockchain. Tangle structure[24], proposed by IOTA, is one of the leading DAG-based projects. Tangle has properties of high throughput: transactions can be in parallel attached to the network from different directions and verified by previous transactions without serious congestion; high performance: newly arrived transactions are confirmed by the previous two transactions via a tiny Proof of Work (PoW) mechanism, in which the computer consumption can be ignored when compared to traditional PoW; low cost: no transaction fees are charged to fit for situations such as IoT and edge computing. However, Tangle structure confronts potential threats on the fork of subgraphs due to the multi-directional expansive network. Specifically, Tangle works on delayed confirmation and partial consistency in multiple directions, instead of an instant confirmation in BFT-style consensus. Uncertainty and reversibility caused by the gap between delayed confirmation and instant confirmation makes the network vulnerable to attack. Existing chains are also threatened by miners who own insurmountable computing power to send massive transactions. Newly issued transactions are unpredictably attached to different subgraphs without global reconciliation. As a result, no leading subgraph is formed to maintain stability. The forks frequently happen and the system confronts the risk of parasite chain attack and double-spending attack. In addition, so far there is too limit research conducted to confirm the idea that DAG-based chain outperforms Block-based chain.

With regard to the current blockchain research and development, there remains gaps between development and deployment of new consensus protocols and applications. As the mining part of blockchain needs a lot of power, the cost of blockchain usage stays high, which also hinders the research and development of blockchain. Researchers requires enough fund and computing power to set up a private chain or to deploy on the real chain to test their new ideas. However, all blockchains are distributed, cannot be controlled by a single person, and are also hard to change. It is difficult for researchers to change key parameters of main chain and investigate the effects of changes. Also, it is almost impossible to reproduce certain special issues in a real blockchain, such as vulnerability

attacks like transaction attack, network attack and so on. Therefore, we need an approach to test new ideas in different scopes and sizes of blockchain networks with few costs and impacts on the existing real blockchain networks. Simulation is a good way out. It doesn't cost much, and could find out any tiny change in the system.

Current blockchain simulation still have many limitations. Most simulators are packed, users can only change some settings that defined by the coder. The consensus part is usually invisible as well. Some simulators mainly focus on network performance. The users cannot observe the changes in blockchain during the simulation time.

Therefore, we want to establish a blockchain simulation framework to solve the problems described above.

1.2 Contributions

In this dissertation, we establish a P2P networking simulation framework, called ChainSim. With this simulation framework, we realize the following characteristic:

Simplicity Blockchain is a mix of many different technologies, such as hash calculate, database, BFT, Consensus, Merkle tree, etc. It is hard to understand how everything works. Our simulation framework will be modular and users can only focus on specific parts.

Cost efficitive and efficient Our simulator can afford more than 30000 nodes working together in one single computer. That may lower the cost of blockchain research. The real time cost for the same simulation time is in an acceptable range. When the number of nodes is in thousand orders of magnitude, the real time takes less than one tenth of the simulation time, which shows great efficiency.

Extensibility Due to the feature of Blockchain, the parameters are hard to change. Using our simulator could help find out the best settings of the system before deployment. In addition, we can deploy blockchains based on different consensus protocols in the

simulator. Meanwhile, it can test the performance of the system when facing different events.

Through the framework we have developed, we try to solve the problems existing in the previous simulator. We deconstruct the whole blockchain and then modularize it, which turns out that Chainsim is applicable to a variety of blockchains, including Bitcoin, Ethereum and IOTA. After verifying the correctness and effectiveness of the simulation framework, we have more exploration on blockchain based on it. We try to improve the performance of blockchain by both basic settings and new consensus algorithm. We also replicated different kinds of attacks in IOTA network and tested its network response. Finally, we propose a smart grid model, which shows that Chainsim can be widely used at the application level.

1.3 Dissertation Organization

The rest of the dissertation is organized as follows. In Chapter 2, we introduce the main technologies in blockchain, including consensus protocols, several main blockchain systems and blockchain classification. Chapter 3 presents a P2P blockchain simulation framework which can run in one single personal computer. Then we deploy three different blockchain with this framework in Chapter 4. In Chapter 5, we try to improve the performance of blockchain by changing key parameters and researching new ways of consensus. In Chapter 6 we build a set of attack tests to test DAG-based blockchain. Chapter 7 presents the applications of our framework by simulating a power grid application based on blockchain. We conclude this dissertation in Chapter 8.

Chapter 2

Background

In a distributed system, multiple hosts form a network cluster through asynchronous communication. In such an asynchronous system, state replication is required between hosts to ensure that each host reach a consensus. However, error information may spread in the system due to the host communication failure, performance degradation, and network congestion. Therefore, in order for all hosts to reach a safe and reliable state consensus, it is necessary to define fault-tolerant protocols in the unreliable asynchronous network. The most original fault-tolerant problem is called Byzantine Generals Problem.

Byzantine Generals Problem[13] was raised by Leslie Lamport in the 1980s. Because of the great distance between the stations, Byzantine Generals had to rely on messengers to deliver messages. When war happens, generals must make a unified plan of action. But there are traitors among the generals, who want to destroy the consistency of loyal generals' actions by influencing the designation and dissemination of the action plan. Therefore, the generals must master a predetermined method to reach agreement, so that traitors cannot interrupt the plan made by loyal generals and all loyal generals can reach an agreement. The essence of Byzantine Generals Problem is to find a way for generals to build consensus on battle plans in an untrusted environment with traitors.

The problem of Byzantine Generals can be described as follows: a general sends an order to the other N-1 generals, so that 1) all loyal generals who receive the order would obey the order; 2) if the general who transmits the order is loyal, then other loyal generals would abide by the order. Lampiort's[14] research on Byzantine Generals shows that when the number of traitors m is less than 1 / 3 of the total number of generals n, through synchronous communication, the generals can reach an agreement. If the communication

is tamper proof and verifiable, a solution can be found whatever the number of traitors is (at least two loyal generals). In asynchronous communication, however, Fischer-Lynch-Paterson theorem proves that as long as one traitor exists, there is no solution to Byzantine general problem.

In the distributed system[15], especially in the blockchain network, the environment is similar to that of Byzantine Generals. There are normal servers (loyal generals), faulty servers, and destroyer servers (traitor generals). The core of consensus algorithm is to form consensus on network state among normal nodes. Byzantine Fault refers to the fault that any observer shows different states from different angles. In a distributed system with Byzantine Faults, the Consensus Problem is to find an algorithm and protocol to satisfy the following three attributes:

- Agreement: All non-fault processes must agree with the same value.
- Validity: If all non-fault processes have the same initial value, the value they agree with must be the same initial value.
- Termination: Each non-fault process must determine a value.

According to Fischer-Lynch-Paterson theory[16], in the distributed system of asynchronous communication, as long as any process has Byzantine Fault, it is impossible to find a consensus algorithm that can meet the above requirements at the same time. In practice, due to a wide range of application scenarios and goals, various consensus algorithms are designed. For private chain and consortium blockchain, there are strong requirements for Agreement and Validity, so consensus algorithm with strong consistency is generally used. The public chain usually uses the consensus algorithm with Eventual Consistency as the limitation of Agreement and Validity is not that strict. These algorithms have their own advantages and limitations though.

Next, we will introduce several common consensus protocols in blockchain in detail.

2.1 Consensus Protocols

2.1.1 Practical Byzantine Fault Tolerance

The original Byzantine fault-tolerant system lacks practicability because it needs to demonstrate its theoretical feasibility. In addition, it needs additional clock synchronization mechanism, and the complexity of the algorithm increases exponentially with the increase of nodes. Practical Byzantine Fault Tolerance (PBFT)[17] system reduces the complexity of Byzantine Agreement from exponential level to polynomial level, which makes it possible to use Byzantine Agreement in distributed system.

PBFT is a Byzantine system of state machine, which requires all nodes to take the same action and maintain a state together. In PBFT, the part supporting the daily operation of the system is completed by the consistency protocol. Consistency protocol requires that the requests from clients have to be executed in a certain order on each service node. This protocol divides server nodes into two types: master node and slave node, in which there is only one master node. In the protocol, each server node works under the same configuration information. The master node is responsible for sorting the clients' requests, and the slave node executes the requests in the order provided by the master node. A consistency protocol may include the following stages: request, pre-prepare, prepare, commit, reply, etc.

PBFT systems usually assume that the number of faulty nodes is m, and the number of nodes in the whole server is 3M + 1. Each client's request needs to go through five stages, through twice interaction between two nodes to reach a consistency, then execute the client's request. Because the client can't get any information about the running status of the server from the server, whether the master node in PBFT has an error can only be detected by the server. If the server fails to complete the client's request for a period of time, the master node replacement protocol will be triggered.

The figure shows a simplified PBFT protocol communication mode, C is client, N_0~N_3 means service node, especially, N_0 is a master node, and N_3 is a faulty node. The basic process of the whole protocol is as follows:

1) The client sends a request M to activate the service operation of the master node

2) When the master node receives the request M, it starts the three stages of protocol and broadcasts the request to each slave node

2.1) Pre-prepare: The master node assigns a sequence number N to the request, constructs PRE-PREPARE messages with sequence number assignment message N and client request message M, broadcasts PRE-PREPARE message to each slave node.

2.2) Prepare: Receive PRE-PREPARE messages from nodes and broadcast PREPARE messages to other service nodes.

2.3) Commit: After each node verifies N and M in the received message, it broadcasts the COMMIT message, executes the received M and responds to the client.

3) The client waits for responses from different nodes. If M+1 response are the same, the response is the result of the operation.

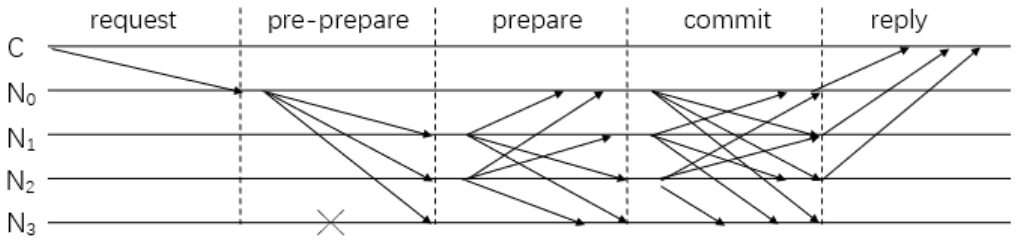


Figure 1. PBFT protocol

PBFT is used in many cases. In blockchain, it is generally applicable to private chain and consortium blockchain that require strong consistency.

2.1.2 Proof of Work

Proof of Work (PoW) is a confirmation that the working node has done a certain amount

of work. The main characteristic of the PoW system is the asymmetry of computation. The working node needs to do some difficult work to get a result. The verifier can easily check whether the working node has done the corresponding work through the result.

In Bitcoin, a block consists of a block head and a transaction list contained in the block. The size of the block head is 80 bytes. This block head is the input string used for bitcoin PoW. The PoW requirement is: link an integer string called nonce after this string, do SHA256 hash operation on the connected string. If the hash result starts with n zeros, the verification passes. In order to achieve the goal of Proof of work, we need to continuously increase the nonce value and perform sha256 hash operation on the new string. Generally speaking, the larger the n value is, the more hash operations need to be completed. Because of the pseudo-random property of the hash value, it needs 2^16 attempts to find hash values with four leading 0. The mathematical expectation of this number of calculations is the amount of PoW required.

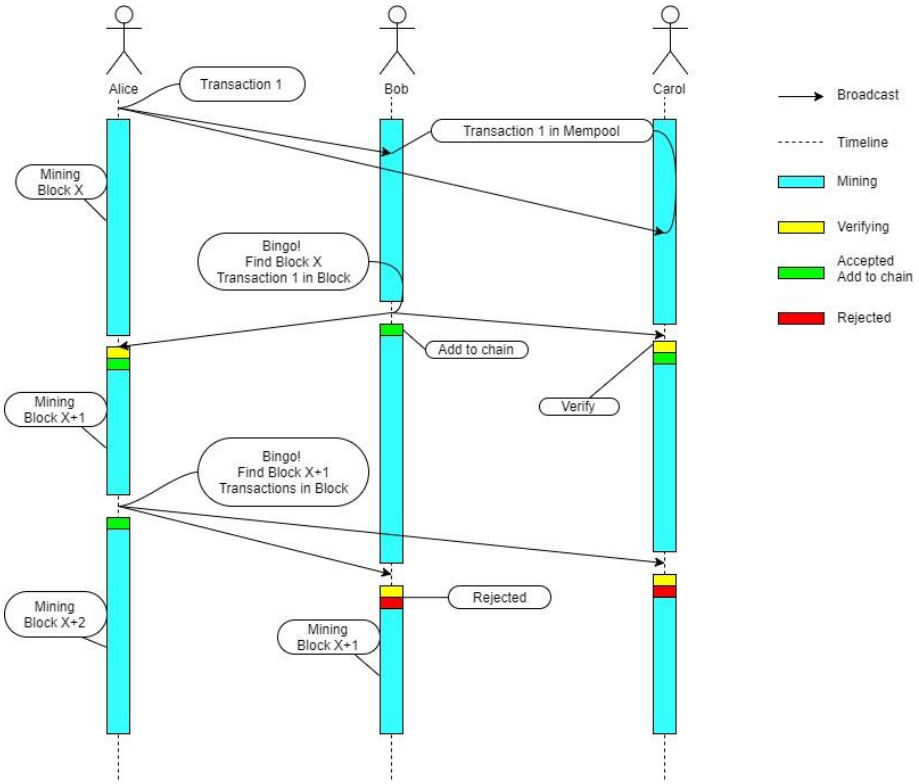


Figure 2. Proof of Work procedure

We take consensus accounting in bitcoin network as an example to illustrate the process of consensus accounting based on PoW:

a) The client generates new transactions, broadcasts them to the whole network, and requires the transaction to be recorded.

b) Once each node receives the request, it will put the received transaction information into a block.

c) Each node tries to find a Proof of Work with enough difficulty in its own block.

d) When a node finds a Proof of Work, it broadcasts to the whole network.

e) When all transactions contained in the block are valid and have not existed in historical blocks, other nodes agree with the validity of this block.

f) The other nodes accept the block and extend the chain by creating new blocks with defining the current block as the previous block.

Through the above accounting process, the transaction information required by the client is written into the blockchain of each accounting node, forming a distributed high probability consistent ledger.

The PoW of bitcoin is a Probabilistic Byzantine Agreements, has low efficiency of consensus. When dishonest computing power has a certain scale, it cannot guarantee that most blocks are provided by honest nodes. But in the bitcoin network, it improves the security of the network by skillfully using the miner reward mechanism.

2.1.3 Proof of Stake

As PoW consumes a lot of resources, people have proposed some alternatives to PoW, and Proof of Stake (PoS) is one of them. In PoS based cryptocurrency, the creator of the next block is selected by random selection and various combinations of wealth or age. Peercoin's Proof of Stake system combines randomization with the concept of "coin age", a number derived from the product of the number of coins multiplied by the number of days the coins have been held. Coins that have been unspent for at least 30 days begin competing for the next block. Older and larger sets of coins have a greater probability of signing the

next block. However, once a stake of coins has been used to sign a block, it must start over with zero "coin age" and thus wait at least 30 more days before signing another block. Also, the probability of finding the next block reaches a maximum after 90 days in order to prevent very old or very large collections of stakes from dominating the blockchain. This process secures the network and gradually produces new coins over time without consuming significant computational power.

2.1.4 Tangle

In order to solve the problem of parallel verification of blockchain, consensus protocols based on DAG (Directed acyclic graph), such as Tangle, have been produced. Different from the traditional chain structure, tangle uses directed acyclic graph, which in essence allows or even encourages blockchain forking.

In Tangle, the node that needs to initiate the transaction walks along the DAG randomly through the algorithm, selects the path and checks, then links the new transaction to the tail, which indirectly checks the old transaction. Although Tangle stores the data in DAG mode, the path selected each time is still a chain, and the transaction verification is still processed as a single chain. When the transaction frequency is very low, the DAG actually degenerates into a chain. In the selection of walking path, giving more weight to the old nodes who get more confirmation is essentially to encourage new transactions to choose the longest chain for verification. The path of each transaction is not fixed, and multiple transactions can be carried out simultaneously. When the transaction is very frequent, almost unlimited TPS can be obtained through forking. The more nodes involved, the higher the TPS. Iota based on tangle has become the first choice of IOT blockchain due to its high throughput and no handling charge.

2.2 Blockchain Networks

2.2.1 Bitcoin

The concept of bitcoin was first proposed by Nakamoto on November 1, 2008. On January 3, 2009, bitcoin Genesis block was born. Different from traditional currency, bitcoin does not rely on specific currency institutions. It is generated by a large number of calculations based on specific algorithms. Bitcoin uses the distributed database composed of many nodes in the whole P2P network to confirm and record all transactions, and uses cryptography design to ensure the security of all aspects of currency circulation. This system enables anyone on earth to exchange money and transfer value through the Internet without any third-party organization. Bitcoin has not invented any new technologies and algorithms, and the technologies involved, including proof of work, time stamp, public key system and so on, are already mature. Bitcoin solves the ownership problem of digital assets without trusted third party through the combination of these technologies. Broadly speaking, the collection of these technologies and ideas is what we call blockchain.

Essentially, Blockchain is a peer-to-peer distributed ledger database, which consists of connected data blocks. The connection pointer between blocks is the Header Hash processed by cryptographic hash function that protects the transactions in every block, as well as the connected blocks. Thus, any of the historical transaction cannot be changed without invalidating a chain of Header Hash.

Data block

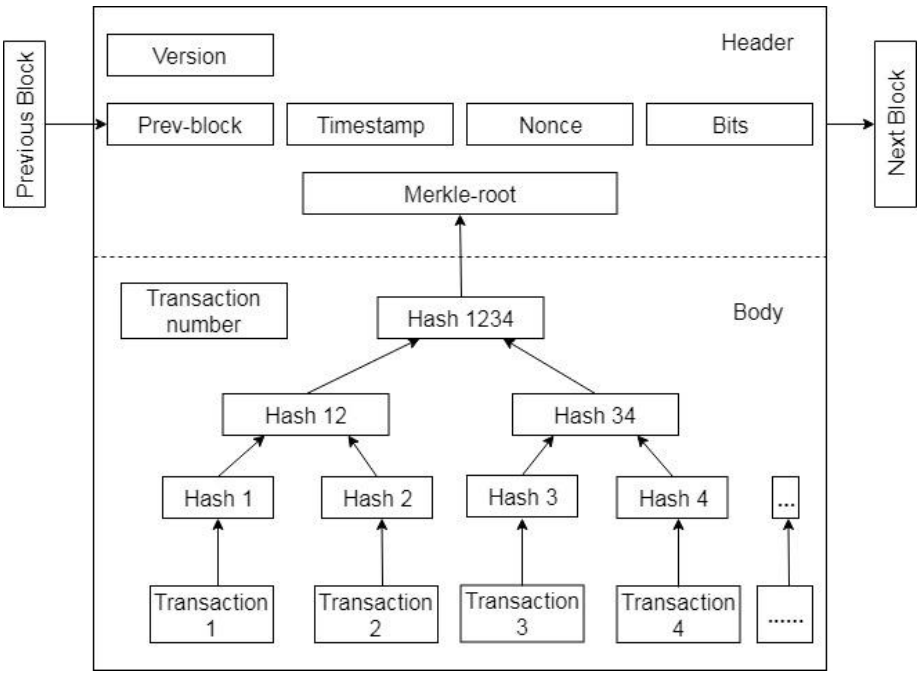


Figure 3. Block structure

In Bitcoin system, one block is created about every ten minutes. Transactions happening in the Bitcoin system are saved in the blocks. A block contains a Header and a Body, as shows in Figure 3. The Header contains metadata of the block, such as Version, Prev-block pointing to the previous block, Timestamp, Nonce, Bits, Merkle-root etc. The body mainly includes the details of the transactions in the structure of Merkle Tree. These transactions form a publicly global ledger in Blockchain system, where the transactions recorded in the ledger could be queried by anyone who can access Internet. Every transaction is signed by a digital signature of the sender to ensure they are unforged and not duplicated. The Merkle Tree with all the transactions has a unique Merkle-root calculated by the Hash procedure, which is recorded in the Header.

Merkle tree

Merkle tree is a kind of data structure, which has all the characteristics of tree structure. Merkle binary tree is used in bitcoin blockchain system. Its main function is to quickly summarize and verify the integrity of the block data. It will group the data in the block for hash operation and recursively generate new hash nodes. Finally, only one Merkle root is

left in the header. Each hash node always contains the hash values of two adjacent data node. The key advantage of using Merkle tree in bitcoin system is that it greatly improves the operation efficiency and scalability of the blockchain. Merkle tree supports Simplified Payment Verification (SPV), that is, it can verify transaction data without running a complete blockchain network node. So that the block header only needs to contain the root hash value without encapsulating all the underlying data, which makes the blockchain run efficiently on smart phones and Internet of things devices.

Time stamp

Time stamp refers to the total number of seconds from 0:00:00 Greenwich mean time (GMT) on January 1, 1970 to now. It is usually a character string that uniquely identifies the time of a certain moment. In the bitcoin system, the node with accounting right needs to stamp the time stamp in the Header when linking blocks, which is used to record the writing time of the current block. The time stamp of each subsequent block will enhance the PoW of the previous time stamp, forming a time increasing chain. Time stamp makes the data in the blockchain easier to trace, and can also be used as an important parameter of Proof of Existence. It can confirm that certain data must exist at a certain time. This ensures that the blockchain database cannot be tampered with and forged. It also makes blockchain technology can be applied to notarization, intellectual property registration and other time sensitive areas.

Mining

The Block is created through mining process, which is an exhaustive random number algorithm. During this process, the miners package the hash value of the previous block and all the transactions have happened in the latest ten minutes, and find a value for Nonce to calculate a hash value with 256bits. The mining process is aimed to find the value of Nonce that makes the hash value meets some requirements, such as having a certain number of zeros in the first bits. The miner, who successfully find such a value, gets the

right to write the new block to the blockchain by broadcasting the new block to the Bitcoin other nodes to verify.

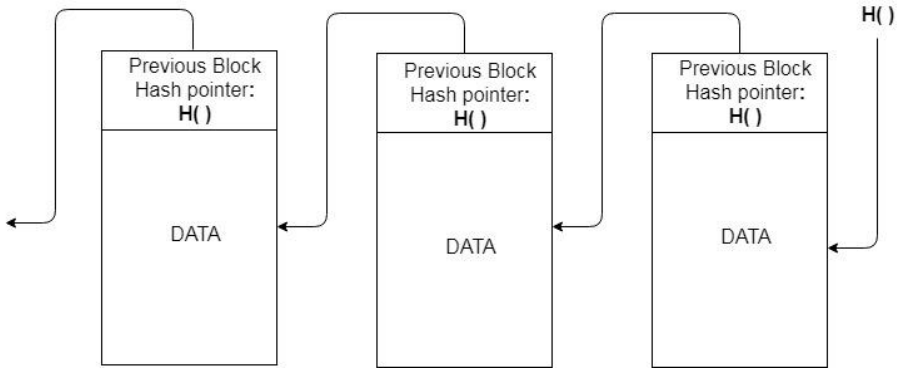


Figure 4. Hash pointer chain

The system has a competition mechanism for miners to compete for the right of writing, which calls Proof of work. During the process, the more computing power a miner spends, the larger possibility the miner can get the right. If the new block is successfully added into the blockchain, the miner will get some reward. Also, it is possible for two different miners to find new blocks almost at the same time. The two blocks might be verified and accepted by a subset of the Bitcoin network, which form a fork. In such a case, the other miners need to choose the fork which with larger number of blocks (implying heavier work).

Through the mining process, the transactions from different users are written in the blockchain, which is hosted on every single node within the network. So we can get a distributed and high reliable and consistent global ledger.

UTXO transaction mode

UTXO means Unspent Transaction Outputs, it is the basic unit of bitcoin trading. Except for the genesis block, there are several inputs (Tx_in) and several outputs (Tx_out) for transactions (Tx) in all blocks. There is no input for the transaction to reward miners who mined new block. In bitcoin system, the input of a transaction must be the unused output of another transaction, and the input also needs the private key corresponding to the last output address to sign. At present, the UTXO in the whole blockchain network will be stored in each node. Only the transactions that meet the conditions of UTXO and digital signature

are legal. Therefore, new transactions in the bitcoin system do not need to trace the whole transaction history to confirm whether the current transaction is legal.

Hash function

Hash function has an important application in bitcoin system, that is, the original data is encoded into a specific length string composed of numbers and letters and then recorded in the blockchain. The data processed by hash function is unidirectional, and it is almost impossible to calculate the original input value through the processed output value. Even if the input value of hash function is only one byte different, the result of output value will be completely different. In bitcoin system, SHA256 hash function is usually used to identify and store data uniformly.

Difficulty

In the bitcoin blockchain system, there is a 256-bit target value which is used to adjust the difficulty of mining. The mining process is to constantly modify the nonce value in the header then hash the block information until the hash calculation result is smaller than the target. The mining difficulty increases when the target decreases. The target value is adjusted every two weeks to keep the mining time at about ten minutes.

Encryption algorithm

In addition to hash algorithm, there is also an asymmetric encryption algorithm for transaction encryption in bitcoin, namely elliptic curve encryption algorithm (ECC). Asymmetric encryption algorithm is the existence of a bunch of mathematically related keys, using one key to encrypt data information, only using another key can decrypt the information. These keys are called public key and private key respectively. We obtain the bitcoin address through the public key, and the private key represents the control right of these bitcoins.

Characteristic

Bitcoin network has the following characteristics:

Decentralization The storage, transmission and verification of blockchain data are based on distributed system structure, and the whole network does not rely on a centralized hardware or management organization. All participating nodes in the public chain network can have the same rights and obligations.

Reliable database The database of bitcoin system adopts distributed storage, and any participating node can have a complete copy of the database. Unless more than half of the computing power in the system can be controlled, the modification of the database on the node is invalid. The more nodes involved in the system, the higher the security of the database. Due to the time stamp of data storage, the time dimension is added to the data, which has high traceability.

Collective maintenance The data blocks in the system are maintained by all the nodes with accounting ability in the whole system. The damage or loss of any node will not affect the work of the whole system.

Safe and trustworthy Blockchain technology uses asymmetric cryptography to sign transactions, so that transactions cannot be forged. At the same time, hash algorithm ensures that the transaction data cannot be easily tampered with. Finally, with the help of the PoW of each node of the distributed system and other consensus algorithms to form a strong computing power to resist the attack of the destroyer, it ensures that the blocks in the blockchain and the transaction data in the block cannot be tampered and forged, which has high security.

Pseudo anonymous Bitcoin system uses the address linked with the user's public key as the user identification. It doesn't need the traditional Public Key Infrastructure (PKI) based Certificate Authority (CA) to issue digital certificate to verify the identity. Bitcoin system uses the address linked with the user's public key as the user identification. It doesn't need the

traditional PKI based CA digital certificate to verify the identity. By running consensus algorithm in the whole network, the trust between nodes is established. Users only need to disclose their addresses, not their real identities. At the same time, users can change their addresses constantly. Therefore, the transaction in bitcoin network is not related to the user's real identity, only to the user's address, and has the pseudo anonymity of transaction.

2.2.2 Ethereum

Since the emergence of bitcoin in 2008, the existence of digital currency has been gradually accepted by people. People also began to think and develop the commercial application based on bitcoin. However, with the expansion of applications, people find that the design of bitcoin is only suitable for virtual currency scenarios. Due to the existence of non-Turing completeness, the lack of saved state account concept and the resource waste and efficiency problems caused by PoW mining mechanism, it is not applicable in many blockchain scenarios. Ethereum came into being in this situation.

Ethereum is a platform and also a programming language, including digital currency Ether and EtherScript used to build and publish distributed applications. Ether is not only a decentralized currency to ensure that the currency supply is not controlled by one party, but also Ethereum provides a complete programming language environment. It is a multi-layer, cryptography based open-source technology protocol. In Ethereum, smart contract can be written to realize the development of decentralized application. The smart contract deployed on Ethereum runs on Ethereum Virtual Machine (EVM) and interacts with the underlying blockchain through EVM and Remote Procedure Call (PRC) interface.

EVM

Ethereum virtual machine is the running environment of smart contract in Ethereum. Anyone can upload programs and let them execute automatically, while ensuring that the state of the program is always publicly visible. These programs run on the blockchain in

strict accordance with the EVM definition. Anyone can create logic for ownership, transaction format and state transition function.

Account

Ethereum has two types of accounts, one is Externally Owned Account (EOA), the other is Contract Account (CA). EOA is a general user account, which is controlled by the private key. The address of the EOA is determined by the public key. CA is a special programmable account. Contracts are stored on the Ethereum blockchain. It is a collection of code and data. Contracts are code controlled and activated by the messages sending by EOA. The address of CA is calculated by the address of the contract creator and the transaction volume sent by this address. The status of EOA is balance, while the status of CA can be balance, code execution and contract storage. The status of Ethereum network is the status of all accounts, which is changed by the transactions of each block, and a consensus is formed in the whole network. The interaction between users and Ethereum blockchain needs to be realized through the transaction between accounts.

Transaction

The transaction of Ethereum can be created by EOA or CA, and the transaction can contain data. If the recipient of the transaction is a CA, the CA can respond. A transaction usually contains the signature of the sender, the address of receiver, the balance of Ether, data, STARTGAS and GASPRICE。

Gas

Each transaction on Ethereum will be charged a certain amount of gas. The purpose of setting gas is to limit the amount of work required for transaction execution. Startgas is the largest gas consumed in this transaction. Gasprice is the gas price to be paid to miners in each calculation step, which is set by the transaction creator. If there are gas remaining at

the end of execution, these gases will be returned to the sending account. If the consumed gas exceeds the startgas, the out of gas exception will be triggered. All current state changes will be rolled back.

PoW

Ethereum's PoW is similar to bitcoin, called Ethash. It dynamically adjusts the difficulty to find a new block in the whole network every 15 seconds. Ethash adds memory difficulty to PoW, that is, every 30000 blocks need a new DAG data to calculate the target hash. This is equivalent to a window of about 125 hours, called epoch. A new miner needs to generate this DAG before it starts mining.

In addition, Ethereum encourages miners to mine uncle blocks. Uncle block refers to the block that meets the difficulty condition but is not confirmed, also known as Stale. For example, miners A and B mined blocks that met the standards almost at the same time, but due to the network delay, block a was confirmed by consensus, and block B became a stale. Since Ethereum generates blocks much faster than bitcoin, stale is more likely to occur. In bitcoin, there is no reward for mining stales. In Ethereum, miners who generate uncle blocks and miners who include uncle blocks in their chain can be rewarded. Uncle blocks are the historical blocks which are up to 6 blocks away from the current connected blocks, and each block can link up to two uncle blocks. This makes it more difficult for attackers to catch up with a main chain with uncle blocks, which effectively enhances the security.

DApps

DApps is similar to traditional web application, which is composed of user interface (UI) as front-end and smart contracts as back-end business logic. The difference is that its logic can only run on the blockchain instead of the server host, and its client code runs on browsers and/or mobile Apps Mist.

DApps has the following characteristic:

- DApps data is encrypted and stored on the blockchain.
- DApps decentralized operation through network nodes. It can run on users' personal devices and does not rely on a central server to deliver messages or a central database to record data.
- DApps participant information is safely stored and can be used for transactions and sales without intermediaries
- DApps must be open source and autonomous. It can be freely packaged and generated by users, and the signature marks the ownership. Its publication is not subject to any institutional restrictions.

2.2.3 Other Cryptocurrencies

In recent years, in addition to bitcoin and Ethereum, a large number of cryptocurrencies based on blockchain have emerged. We choose a few of them for a brief introduction.

Litecoin

Litecoin (LTC)[19] is a cryptocurrency that was designed to provide fast, secure and low-cost payments by leveraging the unique properties of blockchain technology. The cryptocurrency was created based on the Bitcoin (BTC) protocol, but it differs in terms of the hashing algorithm used, hard cap, block transaction times and a few other factors. Litecoin has a block time of just 2.5 minutes and extremely low transaction fees, making it suitable for micro-transactions and point-of-sale payments.

IOTA

Iota (Internet of things application) is a new type of distributed network. It is to help solve some scalability problems in bitcoin and other cryptocurrencies. At present, blockchain based systems like bitcoin and Ethereum have limited transaction efficiency. It uses a unique

method to verify transactions, which is called Tangle. Instead of using chain structure, Tangle uses DAG (Direct Acyclic Graph) structure. It has the advantages of zero transaction cost, fast transaction confirmation, unlimited scalability, which makes it an ideal choice for massive data exchange needed by the Internet of things. It is widely regarded as a major innovation after bitcoin and Ethereum.

Tether

Tether (USDT)[18] is a stable-value cryptocurrency that mirrors the price of the U.S. dollar. The token's peg to the USD is achieved via maintaining a sum of dollars in reserves that is equal to the number of USDT in circulation. Tether is usually used to hedge against fluctuations in the cryptocurrency market. Each Tether coin is linked to one dollar, so keeping money in Tether can protect it from market volatility. For this reason, most bitcoin transactions are completed in Tether, it is the bridge of cryptocurrency transactions.

2.3 Blockchain Classification

According to the classification of users participating in the blockchain, the blockchain can be divided into three types: Public Blockchain, Consortium Blockchain and Private Blockchain[20]. Bitcoin and Ethereum described above both belong to the public Blockchain. Next, we will introduce these three kinds of blockchains.

2.3.1 Public Blockchain

The Public Blockchain is open to the public, users can participate anonymously without registration, and can access the network and blockchain without authorization. It is also known as Permissionless Blockchain. Nodes can freely choose whether to participate in the network. The blocks on the Public Blockchain can be viewed by anyone, and anyone can also send transactions or participate in consensus. Public Blockchain ensures that

transactions cannot be tampered with by cryptography. At the same time, it also uses cryptography verification and economic incentives to build consensus in an unfamiliar network environment, thus forming a decentralized credit mechanism. The consensus mechanism in the Public Blockchain is generally PoW or PoS. The influence of users on consensus formation directly depends on the resources they have in the network. The Public Blockchain is generally suitable for virtual currency, public oriented e-commerce, Internet Finance and other B2C, C2C or C2B application scenarios.

2.3.2 Consortium Blockchain

Consortium Blockchain[21] refers to that the nodes participating in the blockchain are selected in advance, and there are usually good network connections and other cooperative relationships between the nodes. It is a kind of blockchain that needs to register license, also known as Permissioned Blockchain. Read and write permissions and bookkeeping permissions on the blockchain are formulated according to the consortium rules. It can determine the degree of openness to the public according to the application scenarios. Because there are few nodes participating in consensus, the Consortium Blockchain generally does not use the mining mechanism of PoW, but uses consensus algorithms such as PoS, PBFT or RAFT. The transaction confirmation time and the number of transactions per second in Consortium Blockchain are quite different from those in Public Blockchain, and the requirements of security and performance are also higher than those in Public Blockchain. Consortium Blockchain is suitable for B2B scenarios such as inter agency transaction, settlement or clearing. The R3 Corda supported by more than 40 banks and the Hyperledger project supported by Linux foundation both belong to the Consortium chain architecture.

2.3.3 Private Blockchain

There is only a limited range of nodes in the Private Blockchain[22], such as the users of a specific organization. The access and usage of data have strict authority management.

Similar to Consortium Blockchain, it is also a Permissioned Blockchain. The value of Private Blockchain is to provide a secure, traceable, tamper proof, automatic computing platform, and also to prevent attacks on data. The application scenarios of Private Blockchain are generally internal applications, such as database management, audit and so on. It also supports information registered by the government and supervised by the public. At present, many private blockchains work in the way of attaching to existing blockchains such as bitcoin, and record system snapshot data to bitcoin system regularly. The typical application is Eris Industries.

2.4 Summary

In this chapter, we have a detailed understanding of different consensus protocols and some existing blockchains. Based on these information, we will establish our peer-to-peer blockchain network model in the next chapter.

Chapter 3

Simulator Model

The blockchain is essentially a distributed ledger database over a peer-to-peer network. Instead of storing all ledgers in a centralized server or cluster, the distributed structure of the blockchain allows all nodes in the network to store all data in the chain synchronously. At the same time, the generation of distributed data blocks are subject to a specific consensus algorithm and protocol executed on each peer node. These blocks form a cryptographic account book which is under time sequence and hard to be changed. Through the blockchain technology, any network users who do not know each other can rely on the data on the blockchain to make transactions without any centralized trustworthy institutions.

In order to simulate the entire blockchain network on a single personal computer, we plan to establish a simulation structure which meets the need of different blockchain. At present, the application of most blockchain technologies is similar to bitcoin, and most of them are based on the extension of bitcoin structure. Therefore, the simulator model would be constructed based on the structure of bitcoin.

3.1 Blockchain Structure

We deconstruct the structure of bitcoin and then divide the basic structure of the blockchain into three layers: network layer, data layer and application layer, as shown in Figure 5.

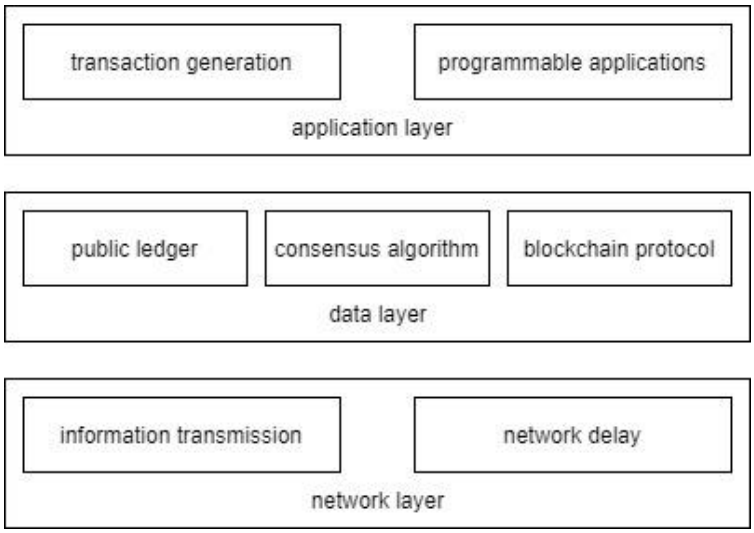


Figure 5. Basic blockchain structure

The network layer mainly implements the information transmission and network delay in P2P network. Blockchain is a distributed system based on TCP / IP communication protocol and peer-to-peer network. Unlike the traditional distributed system, it does not rely on centralized server nodes to forward messages, but each node participates in the forwarding of messages. Therefore, in the blockchain, P2P network has higher security than the traditional network - any node attacked will not affect the whole network. All nodes store the state information of the whole system.

The data layer, in other words, miner layer, mainly contains the public ledger, consensus algorithm and corresponding blockchain protocol. Blockchain is a distributed database system that block can only be appended and cannot be changed. It is a kind of distributed ledger. Users can query the ledger anytime and anywhere in the public chain. In the blockchain network, nodes use consensus algorithm to maintain the consistency of the ledger database in the network. At the same time, the blockchain uses cryptographic signature and hash algorithm to ensure that the database cannot be tampered with and forged, and the data can be traced back.

The application layer, which is equivalent to the user layer, mainly focuses on the transaction generation and business logic such as smart contracts. We use blockchain instead of traditional transaction registration and settlement system. Through the point-to-point distributed timestamp server of blockchain, the electronic transaction proofs are generated

and recorded according to the time order, so that the problem of double payment can be solved and the settlement cost can be reduced. Meanwhile, it can reduce the maintenance cost of the ledger. In addition, the blockchain platform is able to provide a programming environment for users to write smart contracts. Through smart contracts, trading rules can be transformed into contracts that automatically executed on the blockchain platform. The implementation of this contract does not rely on a trusted third party, nor is subject to human intervention. Theoretically, as long as it is deployed, the contract will be executed automatically, and the results can be publicly checked on the blockchain, ensuring the fairness and transparency of the contract. The smart contract of blockchain lays the foundation for programmable currency and programmable finance.

Thus, the blockchain structure has the following characteristics.

Decentralization The storage, transmission and verification of blockchain data are based on distributed system structure. The whole network does not rely on a centralized hardware or management organization, and all participating nodes have the same rights and obligations.

Reliable database The database of the blockchain system adopts distributed storage, and any participating node can have a complete copy of the database. Unless more than half of the computing power in the system is controlled, the modification of the database on the node would be invalid. The more nodes involved in the system, the higher the security of the database is. Blockchain data is also stored with a time stamp, which adds a time dimension to the data and provides high traceability.

Open source programming Blockchain system is usually open source, and the code is highly transparent. The blockchain platform also offers a flexible script code system to support users to create advanced smart contracts, currencies and decentralized applications.

Collective maintenance The data blocks in the system are maintained by all the nodes with accounting function in the whole system. The damage or loss of any node will not affect the operation of the whole system.

Safe and trustworthy Blockchain technology uses asymmetric cryptography to sign transactions, so that transactions cannot be forged. Also, hash algorithm is adopted to ensure that the transaction data cannot be easily tampered with. Finally, the security of the block is guaranteed by the consensus algorithm such as PoW in the whole distributed system.

3.2 Simulation Model

In order to simulate the whole process of blockchain operation, we define message as the main unit to connect different layers. The message is created in application layer. The data layer processes the content of the message. And the message is transmitted between different nodes through the network layer. The order of message propagation in blockchain is shown in Figure 6.

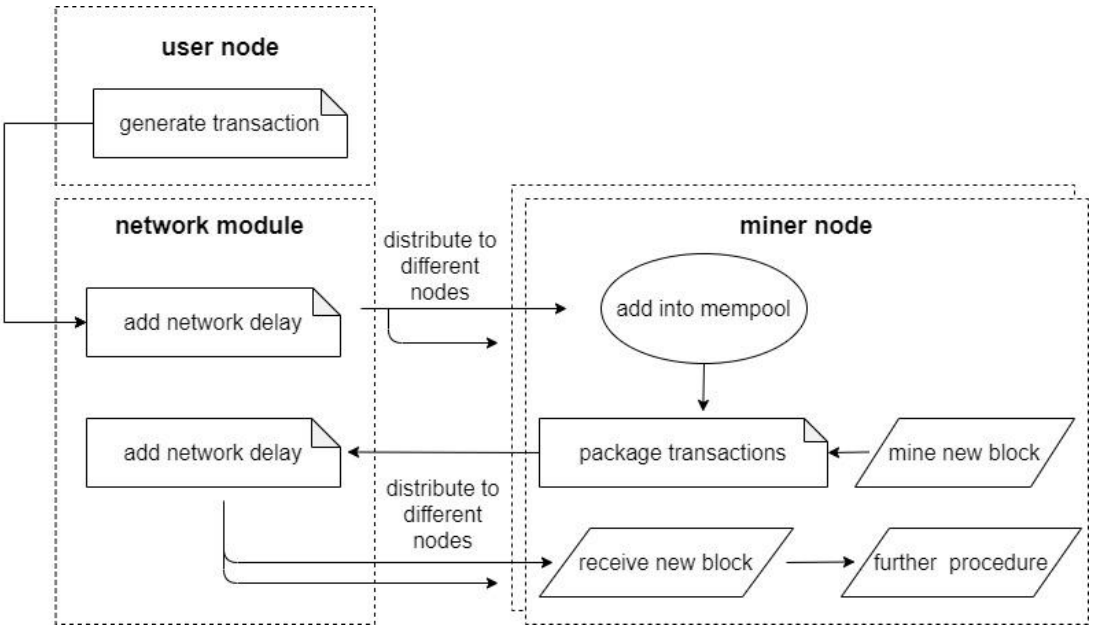


Figure 6. Transaction propagation flow in normal blockchain network

a) The user node generates a new transaction. It packages the transaction and other necessary information into a message. Then the user node sends the message to the network module.

b) The network module adds delay to the message and distributes it.

c) After receiving a message with a transaction included, the miner node puts it into the mempool.

d) The miner node competes the right of recording the new block. If the node wins, it packages the transactions as a block and sends the block to the network module.

e) The network module adds delay to the message and distributes it. (For the network module, step 2 and step 5 are totally the same)

f) After receiving a message with a block included, the miner node processes it according to the consensus algorithm or blockchain protocol.

In order to save computing power and speed up the simulation, we removed the encryption and decryption processes (including hash operations) of transactions and blocks in the simulation demos mentioned in this article, and replaced all this kind of operation with a random time-passing in a certain range. Nonetheless, this delay can be reverted to cryptographic operations as needed later on.

Next, we will introduce the three layers of blockchain simulator according to the consensus process of message.

3.2.1 Application Layer

The application layer is the first step of the entire message consensus process. The user node is the module with the largest amount and the simplest function in our simulation system. Its main functions include generating transactions, sending transactions, and receiving information on transaction progress. Each user node maintains a local ledger, which contains historically completed transactions and transactions that have not been confirmed in the blockchain.

In our simulation, the user node will continuously send transactions to the network module at a random time interval, which can be configured as a simulation parameter. After the

user node sends a transaction, the node's procedure will be in sleep mode until the historical transaction confirmation message is received or the next transaction is ready to be created. Under our simulation framework, the system supports more than 30,000 nodes working simultaneously with an ordinary laptop (hardware/software) configuration, which will be specified later in the paper.

Message is the most basic unit of information in the entire simulation system. A simple message usually includes: timestamp, user ID, transaction ID, sending mode, message size, specific transaction information. The sending mode is usually broadcast to all miner nodes. The specific transaction information is usually related to the blockchain protocol. For example, in the Bitcoin demo, the specific transaction information includes: timestamp, sender ID, receiver ID, token number, and other text information. In the Ethereum demo, we further expand the text information so that it can send smart contracts or call smart contracts, which will be introduced thoroughly in Chapter 4.

In a blockchain system with currency, we usually hang the tokens when a transaction is sent but unconfirmed. These tokens cannot be used again before receiving the confirmation message, but the node can still send a new transaction with the remaining tokens. In contrast, we can set some user nodes not to hang these tokens, in order to test the impact of the double spending attack.

There are two kinds of messages that the user node may receive. One is confirmation message. This kind of message may include the result of your smart contract call and whether your historical transaction is accepted or declined. The user node may remove the hanging tokens or reset these tokens' status. The other one is successful transfer message. This message means someone paid to you and this transaction has already been accepted. We add these tokens into node's account. Each node will update its own ledger.

3.2.2 Network Layer

In order to more efficiently achieve different transmission methods and network delays, we

transfer all the messages in the system that need to be transmitted through the network, no matter the message is an independent transaction, a block or a request, through a network module. The transmission method may be unicast, broadcast, multicast, or gossip.

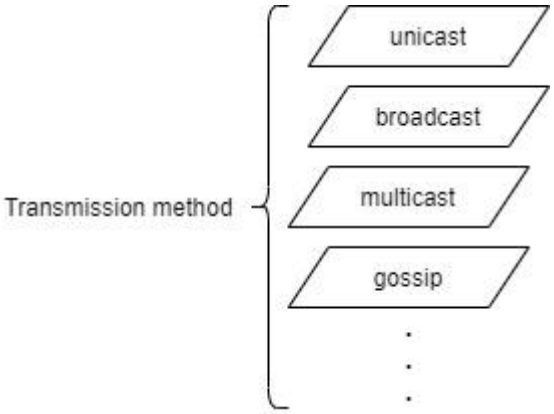


Figure 7. Transmission method

When the network module receives a message, it will check the transmission method first. The transmission method will determine the transmission speed, also the number of message targets if the target nodes are not included in the message or transmission method is just broadcast. It will split a single message containing N target nodes into N single messages, and each message will be sent to a single node. Then the network module determines the network delay of each message by the transmission method and the distance between sending node and target node. This delay value will be added to the message and sent to the information receiving module of the target node.

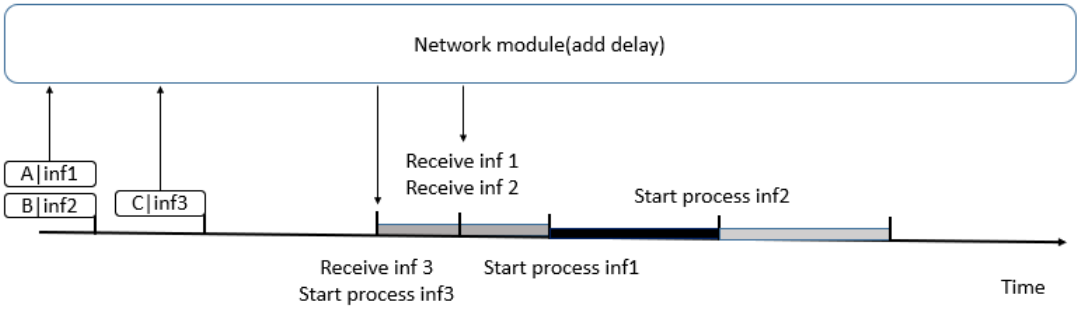


Figure 8. Network delay

As shown in Figure 8, when the node's information receiving module receives a new message, it will determine the specific time that the node receives the message header according to the timestamp and delay time in the message, and insert it into the waiting

queue in chronological order. When the receiving module is under the status of ‘idle’, it starts to receive the first message in the waiting queue which current time has already exceeded the header receiving time. Then the information receiving module calculates the transmission delay according to the current node's transmission speed and message size. The receiving module will remain the status ‘busy’ during the delay time. Once the complete message finishes receiving, it will be appended to the storage space of the node and start to receive next message. If there is no new message received currently, the module remains the status ‘idle’. This whole procedure works similar to the way that the sender sends a header with almost no size, the receiver receives the header after a delay caused by distance, no matter how busy it is at that time, and then the receiver starts to get the complete message one by one in order.

3.2.3 Data Layer

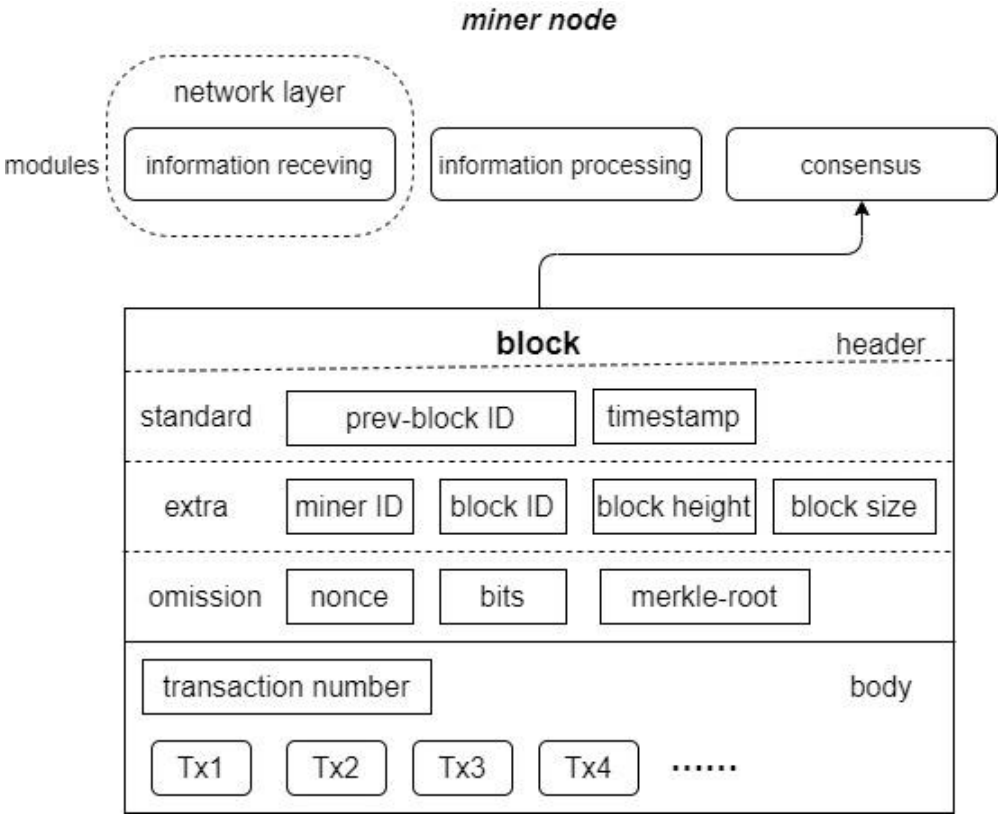


Figure 9. Detail of miner node and block in bitcoin demo

The work of the data layer is mainly completed by the miner nodes. The miner node is the

part with the largest workload and the most complicated work in the entire simulation framework. It mainly includes an information receiving module (which actually belongs to network layer), an information processing module and a consensus module.

As mentioned in last section, the information receiving module arranges the external information in time sequence, checks whether this message is received completely and whether it enters the storage space of this node at the current moment.

The information processing module will further process the message. These messages include various information.

Transaction Check whether the transaction is new. If it is, put the transaction in mempool. The transaction may include token transfer, a smart contract or a contract call.

Block Check the block height. If the block height is lower than the current mining block height, just record this block. If equals, verify this new block. A valid block will be written to the node's chain, switch the state of the block before 6 blocks to be 'confirmed', and send a stop command to the consensus module to stop current mining. If higher, send a request to other miner nodes to get the blocks lost.

Request Check the blocks on the chain of this node, then make a response.

Response If the blocks in the response are valid and have filled the lost blocks of the node, write these blocks to the node's chain, switch the state of historical blocks, and send a stop command to the consensus module to stop current mining.

The work of the consensus module is mainly based on different blockchain protocols. For example, in Bitcoin case, the consensus module will follow the most recent block to mine the new block. We use a random mining time to represent the hash operation, making the consensus module remain the status 'busy' during this mining time. If at the end of the mining time, the stop command sending by the information processing module has not been received, it is deemed that the miner node has successfully mined a new block. The node will package a series of transactions in mempool into a block, record this block in the node's chain, and send it to the network module with all the remaining miners being the

target. In one block, it usually contains the following information: current block ID, current block height, previous block ID, timestamp, miner ID, block size, the number of transactions, and complete transaction information. In order to save computing power, we did not encrypt and complicatedly verify the content of the block, so there is no Nonce, Bits and Merkle-root in the simulation block header, and the simulator will give the block a unique block ID as its current block ID. Each miner node maintains a complete public ledger. In different miner nodes, there may be slight differences in this public ledger. But from the systematic view, there still exists consensus between these ledgers. The main design idea is to use a calculated range of time to replace the hash calculate(mining) part. This range may be affected by the difficulty of blockchain and the mining hashrate of current node. We can also replace this range with a different distribution of mining time in reality.

3.3 Functions Summary

According to our ChainSim simulation framework described above, ChainSim can provide the following functionality and capability.

High number of nodes support in one personal computer During historical experiment, simulator can easily afford more than 30k nodes. The cost of high number of nodes is just more real time for the same simulation time.

Flexible transmission mode ChainSim can define its transmission mode as unicast, broadcast, multicast, anycast or any special transmission method we want to set. We can also change the transmission speed for different nodes, which can help us to investigate the influence of network status.

Variable basic parameters We can change the basic parameters of the blockchain, such as block size, mining difficulty, miner computing ability, etc.

Diverse protocols Based on the overall p2p module, we can achieve diverse consensus methods and blockchain protocols by defining node behavior. Under the same consensus

protocol, we can also define different nodes to have different functions. For example, we can set different nodes as fair miner or selfish miner, which can be used to reproduce various blockchain attacks.

Time-sequence and event-driven In ChainSim we can observe the status of any node at any simulation time, also we can observe the response of the whole network after some special command with no need to stop simulation at that time.

3.4 Related Work

Rajitha et al.[26] used architectural performance modelling and the same incident management exemplar for this approach provided by Weber et al.[28] to measure the latency arising from the Blockchain-related factors, such as the configuration of the number of confirmation blocks and inter-block times. Their management system shows that predictions of median system level response time with a relative error mostly under 10%. Their approach could be used in the design of blockchain-based systems. They modelled the resource and performance characteristics of a local node as a whole, which means they ignore the consensus algorithm (such as mining). In our research, the consensus is the core of the blockchain, which may highly influence the performance and the security. On the other hand, their idea of development based on modular modeling is very similar to ours, which also supports the re-use of constructed models and components.

Gobel et al.[29] developed two Blockchain simulators, based on the DESMO-J simulation framework[30]. They studied the effect of communication delay in Bitcoin Blockchain under a 'selfish-mine' strategy. First, they use a simplified Markov model that tracks the contrasting states which includes a small amount of dishonest(selfish) miners to establish that the use of block-hiding strategies, such as selfish-mine, which may cause the increase of orphan blocks. Then they use a spatial Poisson process model to study values of Eyal and Sirer's parameter γ , to find out the proportion an honest miner mine a block which previous

block is mined by an honest miner. Finally, they use discrete-event simulation to study the behaviour of a network of Bitcoin miners, which includes selfish-mine action under a network with communication delay between miners. Their study found out that if dishonest miners are exist, the performance of both honest and dishonest miners will become worse, the system also can monitor the production of orphan blocks to find out the behavior of selfish-mining. We haven't mention selfish miner yet, we could consider it in the further research.

Grevais et al.[47] provided a complete Bitcoin simulator written by NS2. They introduce a novel quantitative framework to analyze the security and performance implications of various consensus and network parameters of PoW blockchains. They find some method to fight against or limit double-spending and selfish mining under their framework, by changing the basic settings such as network propagation, different block sizes, block generation intervals, information propagation mechanism, and the impact of eclipse attacks. Under their framework, they can find a balance between performance and security in Blockchain Network. Compared with our simulator, their simulator is kind of complete but facing the problem on the number of nodes. All the simple node's location and its network delay should be defined individual. And it simulates the procedure of mining as well, which cost a lot of performance. When the node number rises, the simulator takes a bad respond.

Goswami[40] discuss the factors that make Block-chain largely non-scalable. They provide the simulator written by java. This research delves into the scalability issue of blockchains and provides a comparative analysis of several blockchain parameters with real time data. It delves into the factors that make block chains largely non-scalable. This is done by the simulation of blockchain. It then addresses the various mechanisms that can be employed to resolve this limitation through measuring the differences between the simulator and real time scenarios. Their simulator which simulated the PoW work without node communication, just finish the work in one client. It's effective but getting troubles in combination with real network.

Maher et al.[31] organized their simulator in three layers: incentive layer, connector layer and

system layer. The connector layer is the core part of the simulator, which includes the consensus algorithm, mostly PoW in this paper. They allow simulating a large number of nodes to study the behavior of the nodes and the incentive mechanisms. But they have already defined and locked the working mode of the miners which means miners are all assumed to be honest. They must append as many transactions as they can in one single block and the transactions that offer the highest fees must be included first. They also add an extra new vote mechanism to resolve fork instead of upgrading the copy of the chain by the miner himself.

Lyubomir et al.[32] design their simulator to explore important characteristics and metrics of the network, reason about interactions between nodes, and compare different scenarios in an intuitive way. It shows the simulation time usage comparison as the transaction number changes. It can also simulate large amounts of nodes in one single personal computer. In their simulator, they mostly focus on the transaction sending and network response to replicate the parallel and concurrent nature of the network. But as a blockchain simulator, instead of a network simulator, it weakens the consensus part, which is more important in blockchain.

Yusuke et al.[33] produce a simulator which can simulate the block transmission with good accuracy. They conduct two experiments which clarify the influence of neighbor node selection algorithms and relay networks on the block propagation time. In their simulator, they ignore the transactions generator, which always equals to users, just starting the simulation from block level. To speed up the simulation time, they simulate all the messages as 0 byte except the block message. As they try to change the transmission protocol to decrease block propagation time, they set an unchangeable consensus PoW as well.

3.5 Summary

In this chapter, we deconstruct the classical blockchain structure and analyze the blockchain

workflow with message as the basic unit. Based on the information above, we build our blockchain simulation architecture Chainsim. Then we compare our simulation model with some other blockchain simulation tools, which highlight the generality and efficiency of Chainsim. In next chapter, we will deploy three different blockchains in the simulation architecture to prove the correctness and effectiveness of Chainsim.

Chapter 4

Blockchain Deployment

To investigate the correctness and effectiveness of ChainSim, we implement three different blockchain protocols: Bitcoin, Ethereum and IOTA, then deploy and test using our ChainSim.

4.1 Bitcoin

Bitcoin is known to be the most classic blockchain which most consensus in different private blockchains is based on. We will next introduce in detail how we implement bitcoin under our simulation system.

4.1.1 User Node

User nodes mainly produce new transactions, update the historical transaction status, and are hardly affected by blockchain protocols. When initializing nodes, we give each node a balance to generate subsequent transactions. Each user node contains two ledgers. One includes the confirmed transactions associated with current user node, no matter the current node is the sender or the receiver of the transaction. With this ledger, we can trace the whole process of balance change. The other ledger contains all the transactions send by current node. These transactions include an extra status which indicates whether this transaction is confirmed.

There are two key parameters we can change in this case study: user number and the transaction interval. With these two parameters, we can realize different load of the system,

which is always calculated as transaction per second (TPS). During our research, in a normal system, since all the user nodes have little difference, if these two strategy results in a similar TPS, the simulation result will be the same between high user nodes number, long interval and low user nodes number, short interval. This result will be detailed shown in part 5. With our ChainSim, we can test the network status and the blockchain response when the system is light load or overload.

In each user node, two modules are working at the same time. Message generation module is used to produce new transactions. A transaction contains the following information: timestamp, sender ID, receiver ID, token number and other information. In our simulation we choose a receiver ID randomly and then randomize token number in the range of available token. After the transaction is created, record this new transaction in the second ledger, define the status as unconfirmed, hang these tokens from the available token balance. These hanging tokens cannot be used in future transactions until its status changes. Then we package the transaction as a message, which includes timestamp, user ID, transaction ID, sending mode, message size, and transaction information. Instead of using the original encryption, we make the module remain the status 'busy' for a short period of time, so the timestamp will be different from the timestamp in transaction. We also give the transaction a unique ID to distinguish different transactions in the miner node. The sending mode is usually broadcast to all miner nodes. After packaging, we send the message to network module, and keep the user node the status 'idle' for an interval. The range of interval is defined by simulation setting.

The transaction processing module is used to update historical transaction status. The user node may receive the following messages:

Transaction confirmed Record the transaction in the first ledger. Set the transaction status in second ledger as confirmed. Remove the tokens from hanging tokens.

Transaction rejected Set the transaction status in second ledger as rejected. Remove the tokens from hanging tokens and add these tokens to available token balance.

Transaction received Record the transaction in the first ledger. Add the tokens in this transaction to available token balance.

Message generation module and transaction processing module work independently. They jointly maintain the ledgers of this node.

4.1.2 Network Module

In the bitcoin demo, the main transmission mode is Gossip. The Gossip process is initiated by a seed node. A seed node sends a message to its neighbor nodes in the network. The neighbor nodes that receive the message will repeat the process until all nodes in the network have received the message.

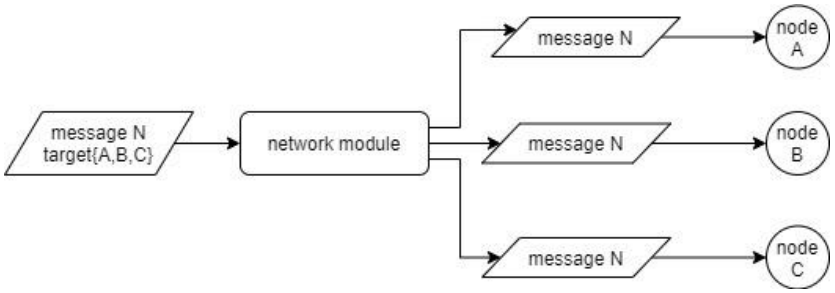


Figure 10. Network module distributes message

In ChainSim, the first seed node's work is finished by the consensus module, and the message expansion of Gossip is included in the miner's information process module. In both parts they will send a message with a list of target nodes to the network module. Then network module will distribute it to its goal.

In network module, there is an address book for every single node, including both user nodes and miner nodes. We can set a speed for information transmission in the network cable. Then this speed and the distance between two nodes will be used to calculate first step delay. The second step delay is calculated in the miner's information receiving module, which uses the transmission speed and the size of the message. The transmission speed can be set by each node individually. With these settings we can set some nodes as high-speed network with remote location, slow speed network with close location, etc.

4.1.3 Miner Node

The information processing module are divided into three pieces according to the received information:

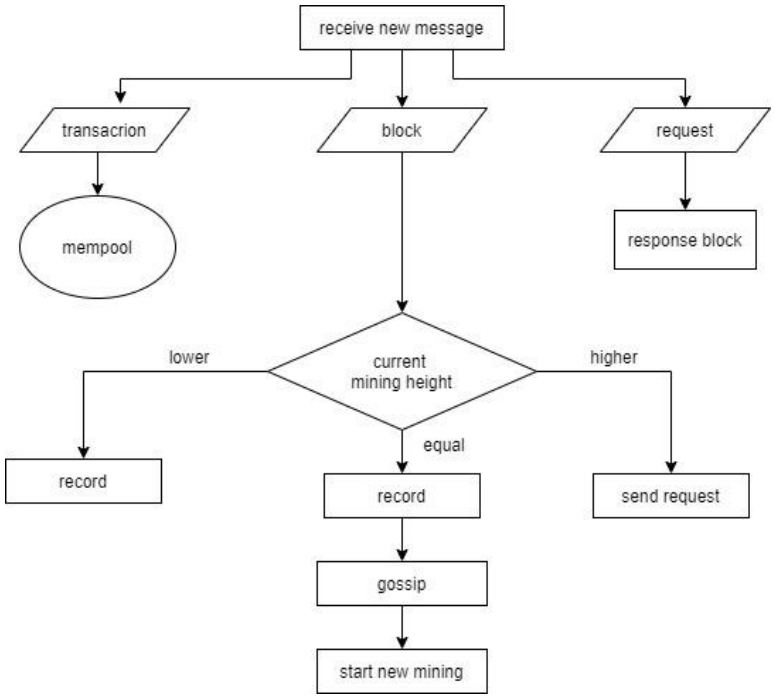


Figure 11. Information processing module

Transaction Transactions will be verified first. If the transaction is new and valid, it will be put into mempool and wait to be packaged in a new block.

Block The module will first check whether this block is an unreceived block. If it is, compare the height of this block with the block that consensus module is currently mining. Then spread this block to several other nodes by sending it in Gossip mode.

- *Lower:* Just record this block and do nothing.
- *Equal:* Verify this block. If valid, record this block in the blockchain. Then remove all the transactions in the new block out of the mempool. After that, find the sixth block forward the new block along the chain and then confirm it. Finally, give the consensus module a command to stop current mining and start a new mining after this new block.
- *Higher:* Which means either this block is invalid or current node has not received past

few new blocks. The node will send a request to several nearby nodes to get the blocks between current mining height and the new block height.

Request Most requests are block synchronization requests from other nodes. Once received, the node will check its blockchain and send the block information back directly. In some cases there will be requests from user nodes to check their transaction status, we didn't include it to reduce the network flow.

The consensus module is under the status 'busy' at most of the time though, it actually hibernates in our case. When the processing module gives the command of stop mining, the mining block height is added first. Next, a mining time is generated randomly from a range. We obtain this range by calculating the distribution of real blockchain. We can change the calculating difficulty by changing this distribution. Then the module keeps hibernate. If the mining time is up and it has not received the command to mine the next block, we define that this node has successfully mined a new block. A block may include following information: previous block ID, timestamp, number of transactions, complete transaction information. To package a block, we choose some transactions from the mempool. We can set different strategies to choose transactions, such as choose half transactions with no fees and half transactions with highest fees. In this section, we just choose the transactions which have waited for the longest time. How the strategy may influence the bitcoin network will be discussed in future research. Also, if there are enough transactions in mempool, the miner node is more likely to fill the entire block. We set a size limit for the block. The total size of transactions and block head information will not exceed this limit. During block generation, we record the ID of the previous block which is linked to the new block, and then record current time as timestamp. The miner will write the new block on its own chain. Before we send the block to other miners, we record some extra information in the message. We give the new block a unique ID to distinguish different blocks in the miner node and write the block ID, the block height and the miner ID in the message. For message transit, we also record the total size of the whole block and set the transmission mode as gossip. The message will be sent to the network module. Then it will

start a new round of mining.

4.1.4 Bitcoin Demo Verification

To verify the correctness of our ChainSim, we compared the simulation results with that of real bitcoin network. All the simulations run on the single personal computer as shown.

Computer parameter

OS: Windows 10 64 bit

CPU: Inter® Core™ i7-7700 CPU @ 3.60GHz

Memory: 16G

Programming language: Python 3.6.0

IDE: JetBrains PyCharm 2019.2.5

Then we decide some parameters for ChainSim bitcoin case before simulation. Some of the parameters have distribution rely on real network, we only show its range in Table 1**.**

Table 1. Simulation settings

Parameters	Default	Parameters	Default
Simulation time	24h	Miner number	2000
Block size	1MB	Transaction size	[0.4,2]KB
User number	1000	Mining time	[0,30]min
Transaction interval	[100,1000]s	Node transmission speed	[5,100]Mbps

We counted all the block information in ten days from the network, then analyzed the distribution of mining time, as shown in Figure 12. The mining time in simulation is generated randomly according to this distribution.

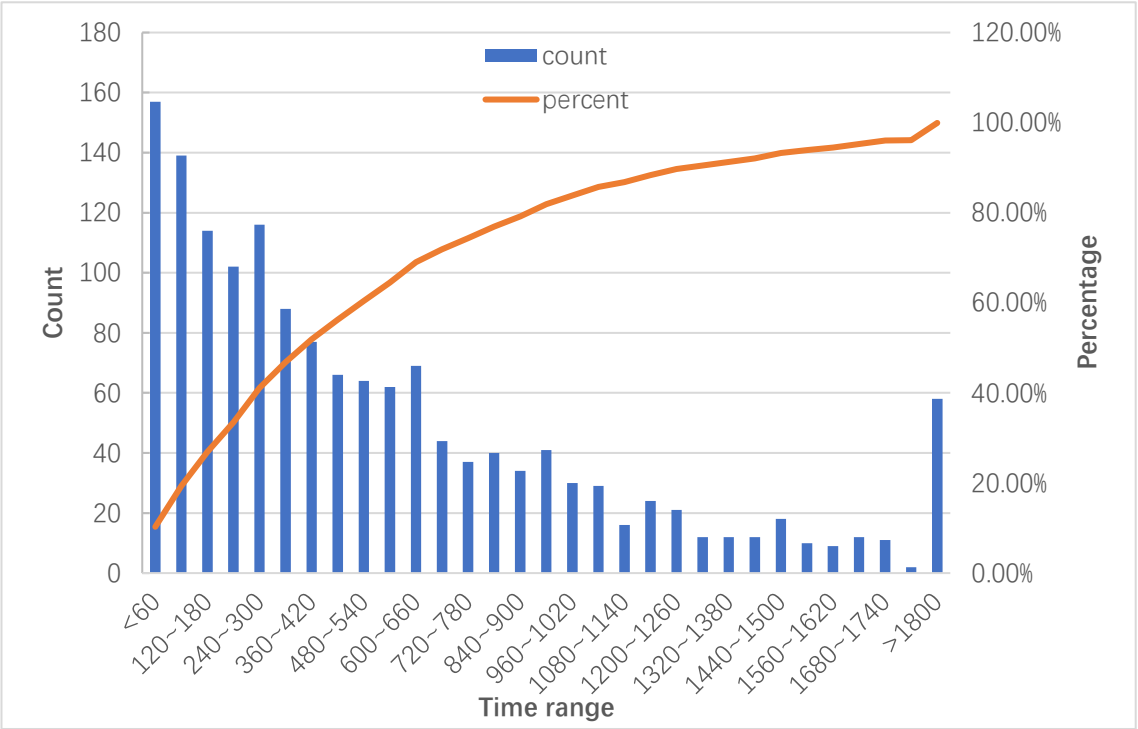


Figure 12. Mining time distribution (10 days in April 2020)

After simulation, we get the results shown as follows[1].

Table 2. Bitcoin network comparison result

Parameters	***Simulation***	***Real network***
Avg block time	548s	524s
Avg block size	0.96MB	1.13MB
Transaction per second	3.43	3.26
Transaction per block	1640	1836
Mempool size	Depends on time	
Median confirm time	520s	370s(with fee)

After comparison, we can find that most of the simulation results are in an acceptable error range, as the real network data is also changing every day.

[1]The real blockchain data gets from the following web in April 2020:

https://www.blockchain.com/charts

https://bitinfocharts.com/

But there are still some differences between real bitcoin network and simulation, which are mostly affected by miner strategy. We limited the block size to 1 MB. In current bitcoin network, SegWit (Segregated Witness) can be chosen by miners to expand block size up to 4MB, which turns out the average block size to be 1.13MB or even higher. For mempool size, it is changing all the time in one day. It may be influenced by a sudden congestion of transactions. So it turns out to be incomparable in long time stable simulation. For median confirm time, it is totally caused by strategy and can be further researched. In current simulation, we set all the transactions with no fee, so the miner chooses transactions fairly depend on time. If the user node set a fee in one transaction, how much fee should it contain to make the miner more likely package its transaction first, how many percent of transactions with no fee will one miner package in its block, all these questions can be discussed under our ChainSim.

To quantify the efficiency of the simulation, we define a middle size bitcoin network with 2000 miners and 20000 users. To simulate the blockchain network of this scale for 24 hours may take 3 hours of real time, which not only saves computing power, but also saves time.

In summary, ChainSim shows great ability in simulating a bitcoin network with basic block data and network performance. It is also a great tool for users and miners to find out a better strategy efficiently.

4.2 Ethereum

Since Ethereum still uses the PoW consensus protocol, and has not modified to PoS consensus yet, Ethereum and Bitcoin has little difference in terms of the operating structure and simulation ideas of the consensus protocol. Based on the Bitcoin simulation case, we made the following two important corrections according to Ethereum unique features.

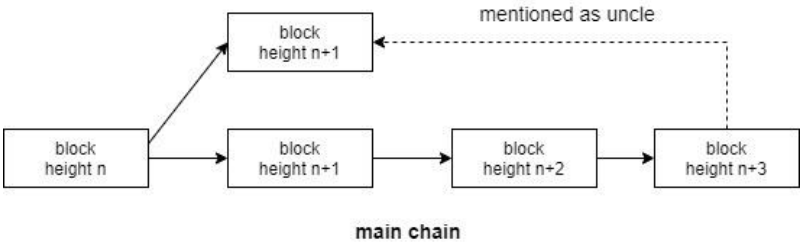


Figure 13. Uncle block

Uncle block If a direct sub-block within 7 layers of the main chain is received, it will no longer be discarded directly, but will be placed in the candidate uncle block. If a new block is mined, miner can add up to two uncle blocks in the current block. When the height difference exceeds 6 layers, or the uncle field of other blocks already contains this uncle block, the block will be removed from the candidate uncle block.

Smart contract As mentioned above, each individual transaction contains some transaction information. We add an identifier to these transactions to distinguish whether this transaction is for currency exchange, building a smart contract or calling a smart contract. In order to use a single program to run the entire blockchain, smart contracts need to be written in Python syntax instead of Serpent syntax, so that operations such as calling the network module to send tokens to an account can be performed. The entire contract is stored in the transaction information field in a string format. When a miner node receives a request to create a new smart contract account, it will create a separate storage space for the smart contract, decode the string contract and keep the program running, or wait to receive the transaction to call the smart contract. When all the gas in the contract is used, an error status will be reserved in the storage space of the contract, and a new transaction that calls the contract will be fed back with an error.

```
"""from transmissiontype import unicast\nx=x+600\n\
unicast(self.now,self.id,target,transaction,size)"""
```

Figure 14. Smart contract example

Since Ethereum provides tools for building a private chain, in order to verify the effectiveness of this simulation model, we built a private chain for comparison. The genesis block of private chain is saved in a json document as shown in Figure 15. Then we generate 4 miner nodes that keep mining empty blocks to test the chain performance.

```
{
  "config": {
    "chainID": 128,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "alloc": {},
  "coinbase": "0x0000000000000000000000000000000000000000",
  "difficulty": "0x400",
  "extraData": "0x0",
  "gasLimit": "0x2fefd8",
  "nonce": "0x000000000000dfa6",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00"
}
```

Figure 15. Private chain genesis block

We compared the simulation results with the real chain and those with the private chain under the same scale and data volume. The comparison results are as follows:

Table 3. Ethereum comparison result

Parameters	***Simulation***	***Real network***
Avg block time	14s	13.4s
Transaction per second	9.42	9
Uncle rate	6.82%	6.16%
Avg block time(private)	14.2s	13.8s
Uncle rate(private)	1.31%	1.03%

As shown in Table 3, all the results are similar between simulation and real network. The significant difference is that as we only deploy 4 miners in the private chain comparison, uncle rate of the network reduces in a high level, but is still almost the same between private chain and simulation.

Given the impact of transaction incentives on miners choosing transactions to include in blocks, it is not uncommon for Ethereum to have empty blocks in the main chain due to

the permission of not full (or even empty) blocks. This simulator can also be used as a tool for researching the optimal harvesting strategy of Ethereum miners.

4.3 IOTA

Unlike the previous two chain-structured blockchains, IOTA is a DAG (directed acyclic graph)-structured blockchain. We hope to verify the scalability of our framework through the IOTA demo. We also test some user behavior attacks against IOTA using this demo, which will be shown detailly in part 6.

In IOTA, its consensus method is named tangle. Instead of separating the process of making transactions by local users and achieving consensus by online miners, tangle integrates these processes into one step. In the IOTA network, the identities of users and miners overlap, whenever you wish to send a transaction or need to complete a certain amount of proof of work. So we deploy IOTA in our framework with all the nodes working as miner nodes. Then we divide the work content of these nodes in IOTA into the following two parts.

Send a transaction

When the user node attempts to send a transaction, it randomly selects two transactions as the parent transaction from the previous parent transaction pool (the storage space of the DAG structure) based on a special selection strategy called the weighted random walk. Then it executes certain proof of work, package important information into a bundle, and send it to the network module to broadcast it to all nodes.

Receive a transaction

The user node receives a new bundle and verifies the validity of the transaction in the bundle. If it passes the verification, the transaction is added to the transaction pool. The

weight of the parent transaction is superimposed according to the weight and attenuation ratio of this transaction. Then iteratively accumulates the parent transaction weight of the parent transaction. When the weight of a transaction reaches a certain threshold, it is deemed that the transaction has been confirmed.

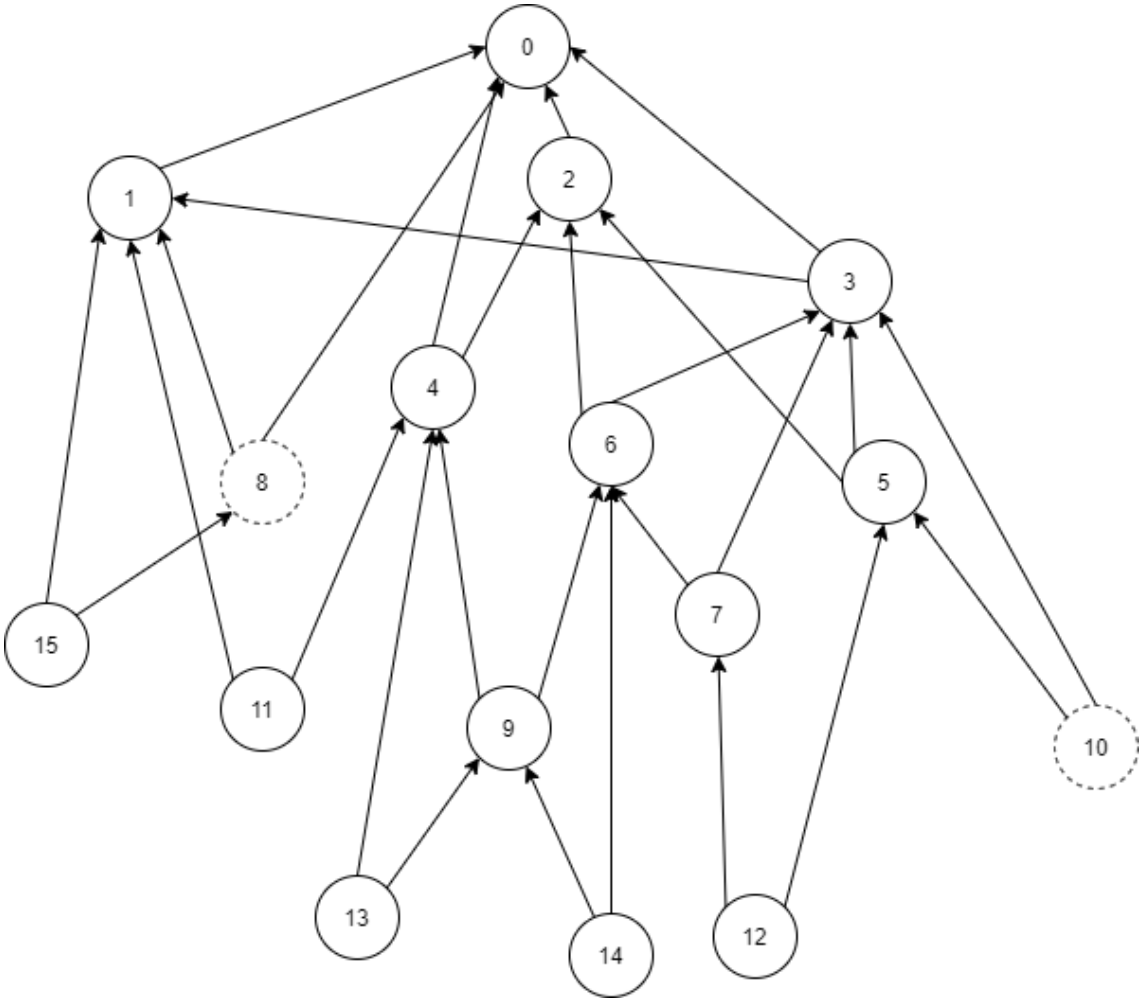


Figure 16. IOTA simulation result (solid-confirmed, dotted-unconfirmed)

As IOTA doesn't provide some detailed settings nor working status analysis, we don't compare simulation results and real network results. We only shown a simple IOTA simulation result in Figure 16. We choose to reconstruct IOTA, including defining the key parameters of its selection strategy, removing the Coordinator and testing its attack response. These will be discussed in part 6.

4.4 Limitations

In Chainsim, we can deploy various protocols, but it still has its own limitations.

First, we need to understand the workflow of the protocol then deconstruct it. After deconstruction, we can write code according to node classification and function classification. Users who do not know enough about blockchain protocols can only deploy blockchains by modifying key parameters depending on existing simulation models, which will be shown in Chapter 5. Secondly, there is a lot of redundancy in the network module of Chainsim. The message will be sent to the network receiving module of all nodes, and then the network receiving module determines whether the node processes this message according to the information records in it. When the number of messages in the system is fixed, the bigger node number, the slower the system runs, which may cost a longer real time for simulation. Then in this dissertation, we use a fixed time to replace the encryption and decryption process, and use a label to identify whether the information is valid, so as to replace the complex verification process. These processes can be deployed in Chainsim when they are needed. Finally, the hash operation in PoW is replaced by a random time according to a certain distribution. We can but do not recommend deploying hash operations in Chainsim. This may cause the real time it spends close to the number of nodes multiplied by the simulation time, which lose the significance of single machine simulation.

In future research, we will provide more complete modular platform including encryption and authentication for lightweight users, try to reduce system redundancy and improve simulation speed.

4.5 Summary

In this chapter, we deploy three types of blockchain protocols demo under our framework. The simulation results show that our simulator could simulate the blockchain accurately and effectively in one single computer. Also, it can change different parameters and strategies to realize different kinds of attack in blockchain. ChainSim can be used by either researcher

to test and evaluate their new consensus protocols, or blockchain foundations to predicate and plan their resources for their blockchains before deployment.

In next chapter, we will try to improve the performance of blockchain by changing parameters and testing new protocols.

Chapter 5

Use Case 1. Blockchain Performance Improvement

Blockchain is a peer-to-peer distributed ledger database, which consists of connected data blocks. The connection pointer between blocks is the Header Hash processed by cryptographic hash function, which protects the transactions in every block, as well as the block connection. Thus, any of the historical transaction cannot be changed without invalidating a chain of Header Hash.

As TCP/IP is the communicating protocol between computers, Blockchain is a trust mechanism and a cooperation protocol. With its unique capabilities, it enables people to establish trust and make transactions without relying on a trusted third party.

In Bitcoin, the system uses Proof of Work[27] (PoW) to ensure that all the transactions have been approved. The miner must do a lot of hash-calculating (which has a variable difficulty to ensure each block to be mined in about 10 minutes) to compete the right of setting its block on the chain. And the mechanism suggests that users should wait at least six blocks before their transactions have been confirmed, which last almost an hour. During this time, the forks will be cut. With this consensus, the system could withstand double spending attack when the number of honest nodes is more than 51 percent.

With the increasing need on Blockchain, several issues in the protocol prohibit Bitcoin from many applications:

Poor scalability Each block can store 1 MB data, which means the system throughput is limited by the size of its block.

High latency The average confirmation time is more than 100 minutes, which seems too long for some situations.[33]

Power inefficient It was reported that the Bitcoin network consumes about 73 TWh per year, i.e. using on average 640 KWh per transaction.[35]

In this part, we will try two different ways to resolve these limitations. In part 5.1, we will change the key parameters of bitcoin and observe its influence on the whole network. In part 5.2, we will introduce a new structure of DAG blockchain.

5.1 Parameter Change

In this section, we collect and define some metrics for QoS of Blockchain. We hope to verify through this section that our simulator has effective response to different basic parameters settings.

5.1.1 Metric Definition

First, we define some metrics and use these metrics to measure the quality of the Blockchain system. The role of these metrics in Bitcoin blockchain network is shown in Figure 17.

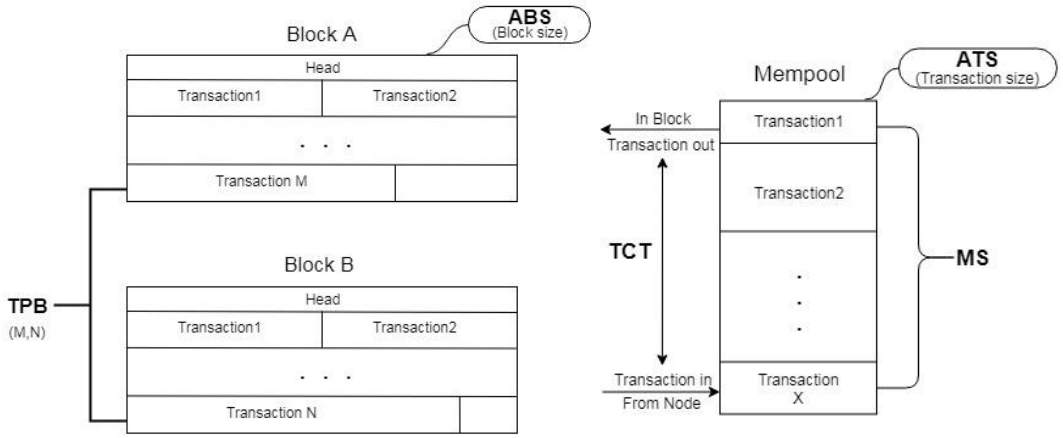


Figure 17. QoS Metrics for a Blockchain Network

TBN (Total Block Number) is the number of blocks that have been mined in a specific time

period. It directly related to simulation time. BCT (Block Commit time) is the average time needed to commit a block to the main chain since being created, which is approximately equals to 6 times of mining time. Block Size includes the header and some transactions, and is limited by 1000000B (defined by Bitcoin source code). ABS (Average Block Size) is the average block size in MB. It is an indicator that shows whether the block is filled. TPB (Transactions per Block) is the average number of transactions per block contains. One transaction needs to use the storage 4 times the simple message. After the update 'Segregated Witness'[25] on 8/24/2017, the extra 3 times does not calculate in Block size, which expand TPB. Transaction size range and transactions per seconds are the settings we want to change to measure the performance, which turns to be ATS (Average Transaction Size) and TPD (Transactions per day) after statistics. TCT (Transaction Confirmation Time) shows how long one transaction is accepted in average, MS (Mempool Size) shows the waiting list of transactions.

In the system sight, we want the block to be filled (ABS as large as possible, TPB as many as possible), so we can transport more information at the same time. In client's view, TCT is the most important QoB, i.e. the less TCT is, the more effective the Blockchain network.

5.1.2 Simulation and Results

There are several parameters we can change during simulation:

Table 4. Bitcoin network default settings

Parameters	Description	Default
SIM_TIME	The amount of time that simulation runs.	24H
NUN_OF_NODES	The number of nodes in simulation.	512
MINGING_TIME	The rank of time that a block could be mined.	[8, 10] min
BLOCKSIZE	The limited size of a single block.	1000000B
TRANSAC_SIZE	The rank of size that a single transaction could be.	[100, 2000] B

The time in the simulation is calculated in 0.1s, and the transaction is in 1 Byte.

The parameters above configure how the simulator runs. In the real Bitcoin network, most of these parameters are fixed or limited. But we can change these parameters in our framework to find out how they affect the Bitcoin network.

With this simulator, we can get some important information about the Blockchain in each node. Due to Blockchain's working method, every node gets a similar but not the same chain. Once we consider the network delay, the chain on different nodes may have a few differences which may show in the commitTime in block information. One Blockchain in node 0 created with default settings shows below. The first diagram in Figure 18 shows the first few blocks in the network including Master Block, the second diagram shows the latest blocks including some Blocks haven't been committed yet.


```
{"owner": -1, "blockID": 0, "state": 1, "blockSize": 0, "transactionNO": 0,
 "creatTime": 0, "power": 0, "commitTime": 33166, "note": "Master block"}
{"owner": 84, "blockID": 1, "state": 1, "blockSize": 999483, "transactionNO": 960,
 "creatTime": 5901, "power": 5901, "commitTime": 38996, "note": "null"}
{"owner": 26, "blockID": 2, "state": 1, "blockSize": 998162, "transactionNO": 955,
 "creatTime": 11451, "power": 11451, "commitTime": 44008, "note": "null"}
{"owner": 161, "blockID": 3, "state": 1, "blockSize": 999801, "transactionNO": 978,
 "creatTime": 16791, "power": 16791, "commitTime": 49133, "note": "null"}
{"owner": 253, "blockID": 4, "state": 1, "blockSize": 998736, "transactionNO": 959,
 "creatTime": 21938, "power": 21938, "commitTime": 54455, "note": "null"}
{"owner": 379, "blockID": 5, "state": 1, "blockSize": 999108, "transactionNO": 941,
```

```
{"owner": 121, "blockID": 153, "state": 1, "blockSize": 999862, "transactionNO": 969,
 "creatTime": 824690, "power": 32049, "commitTime": 856996, "note": "null"}
{"owner": 304, "blockID": 154, "state": 1, "blockSize": 998800, "transactionNO": 958,
 "creatTime": 830524, "power": 32765, "commitTime": 862147, "note": "null"}
{"owner": 92, "blockID": 155, "state": 0, "blockSize": 999817, "transactionNO": 947,
 "creatTime": 835648, "power": 32631, "commitTime": 0, "note": "null"}
{"owner": 166, "blockID": 156, "state": 0, "blockSize": 998939, "transactionNO": 957,
 "creatTime": 840779, "power": 31774, "commitTime": 0, "note": "null"}
{"owner": 180, "blockID": 157, "state": 0, "blockSize": 998882, "transactionNO": 928,
 "creatTime": 845959, "power": 31650, "commitTime": 0, "note": "null"}
{"owner": 75, "blockID": 158, "state": 0, "blockSize": 999461, "transactionNO": 960,
 "creatTime": 851238, "power": 31586, "commitTime": 0, "note": "null"}
{"owner": 195, "blockID": 159, "state": 0, "blockSize": 999956, "transactionNO": 945,
 "creatTime": 856979, "power": 32289, "commitTime": 0, "note": "null"}
{"owner": 28, "blockID": 160, "state": 0, "blockSize": 999908, "transactionNO": 935,
 "creatTime": 862131, "power": 31607, "commitTime": 0, "note": "null"}
```


Figure 18. Blockchain information in node 0

We do some simulations with different settings. The result shows below.

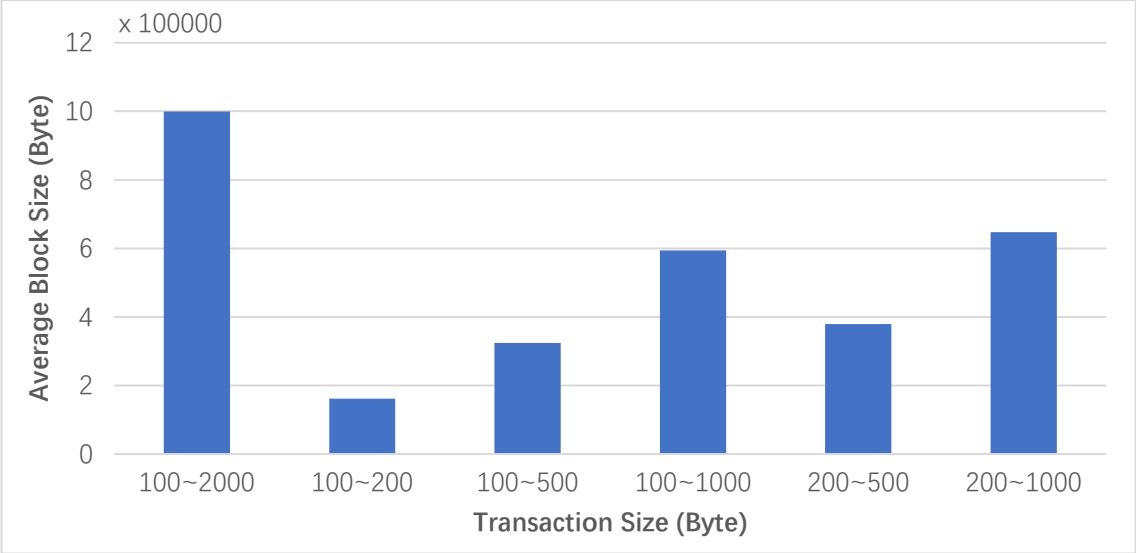


Figure 19. ABS

In Figure 19, when the number of transactions per day and the block size remain unchanged, only if the average transaction size is up to about 1000B, will transactions fill the block to the full. When the transaction size is small, more transactions could be recorded in one block. Once there are not enough transactions, the block won't be filled any more. In real system, mining a block costs a lot, so we want the block to contain as much information as possible. But when the transaction size is too big, due to the limited size of transactions that each block can contain, more and more transactions will be piled up in the Mempool. The confirmation time of the transaction will also be lengthened.

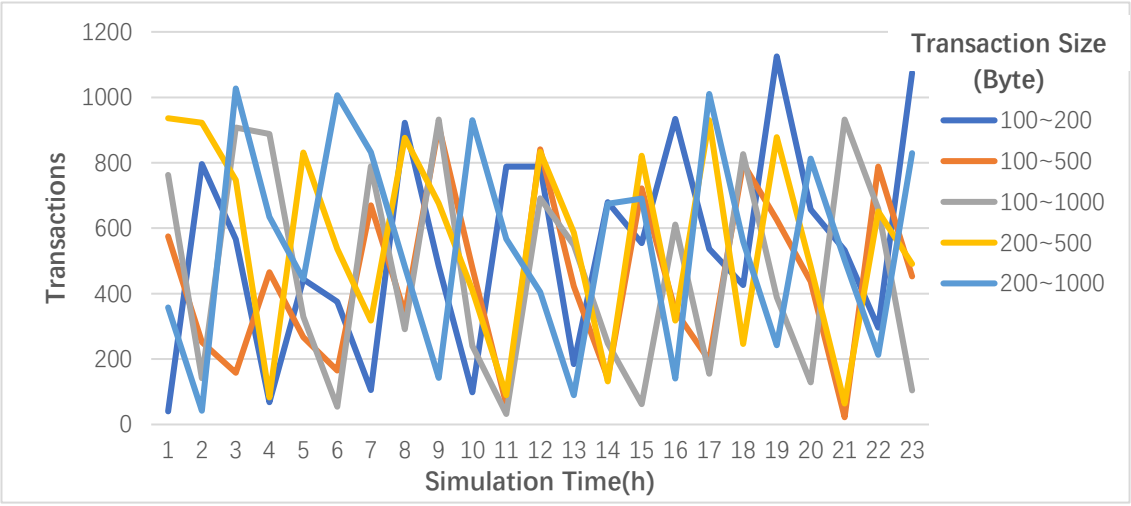


Figure 20. Mempool Size I

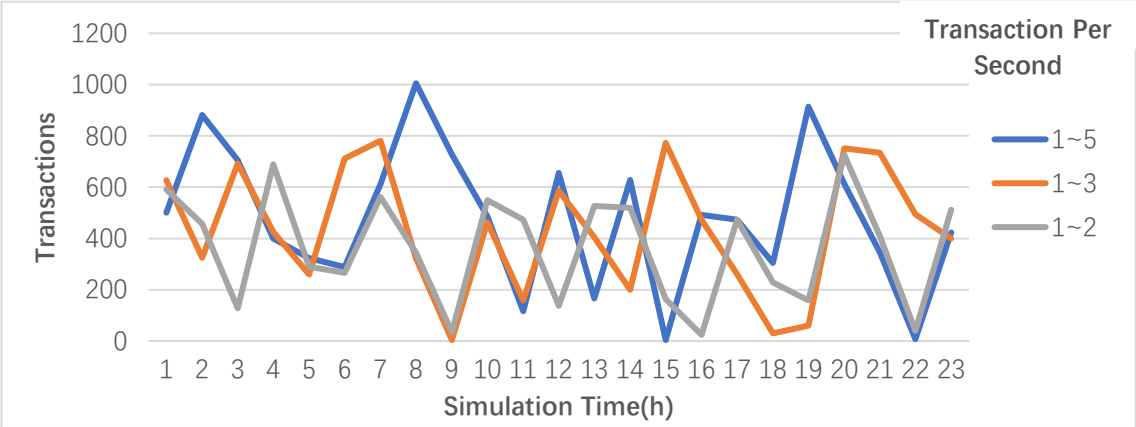


Figure 21. Mempool Size II

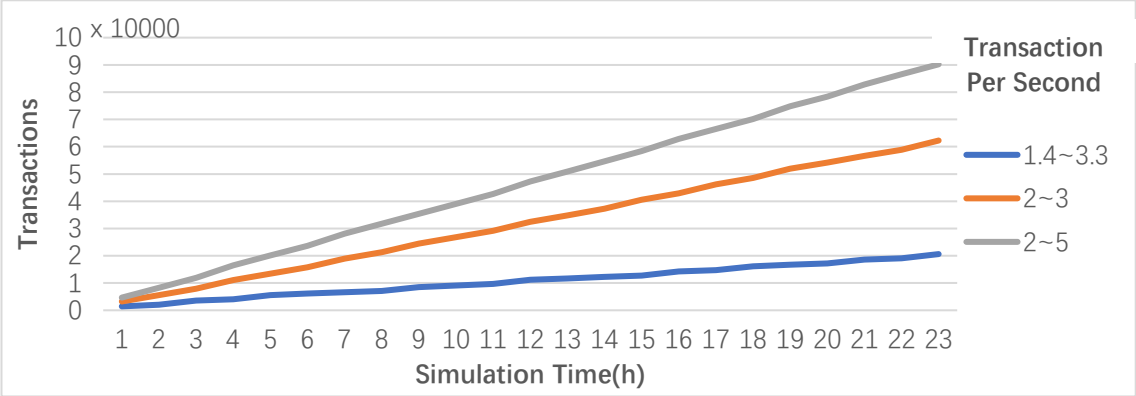


Figure 22. Mempool Size III

Figure 20 shows the details in one day that when the processing capacity of the block is stable, mempool size keeps in a low level. The transactions do not need to wait for a long time to be set in a block. Similar in Figure 21, when the transaction per second increases, the mempool size may keep in a low level before the block is filled. Once filled, as shown in Figure 22, the mempool size will go straight high, which means the system will become more and more redundant. Just like in Figure 23, the transaction commit time rises with time. In the real network, the transaction number depends on various events, which have peak and off-peak time, this may cause unexpected peek in commit time and mempool size.

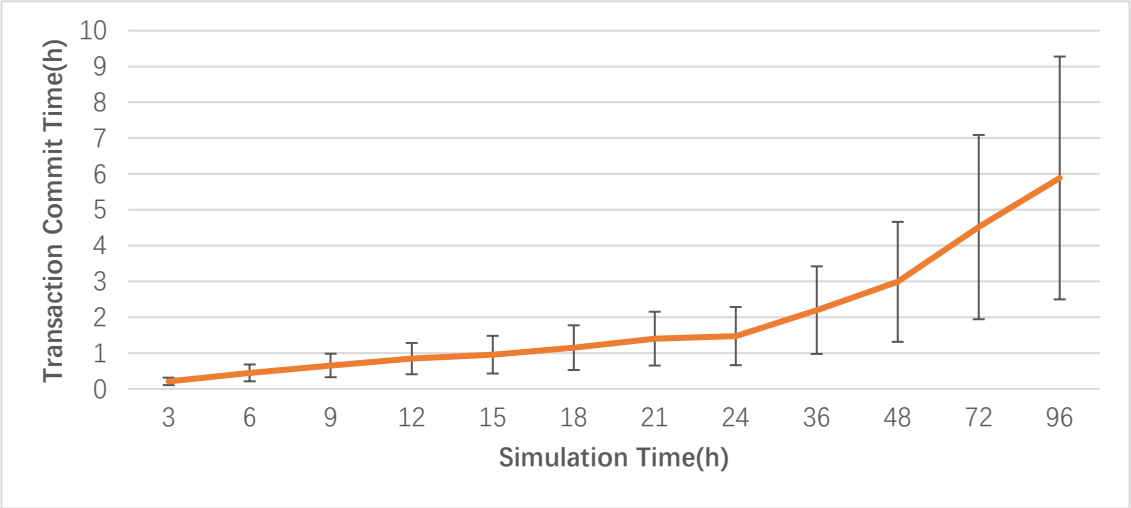


Figure 23. Transaction Commit Time

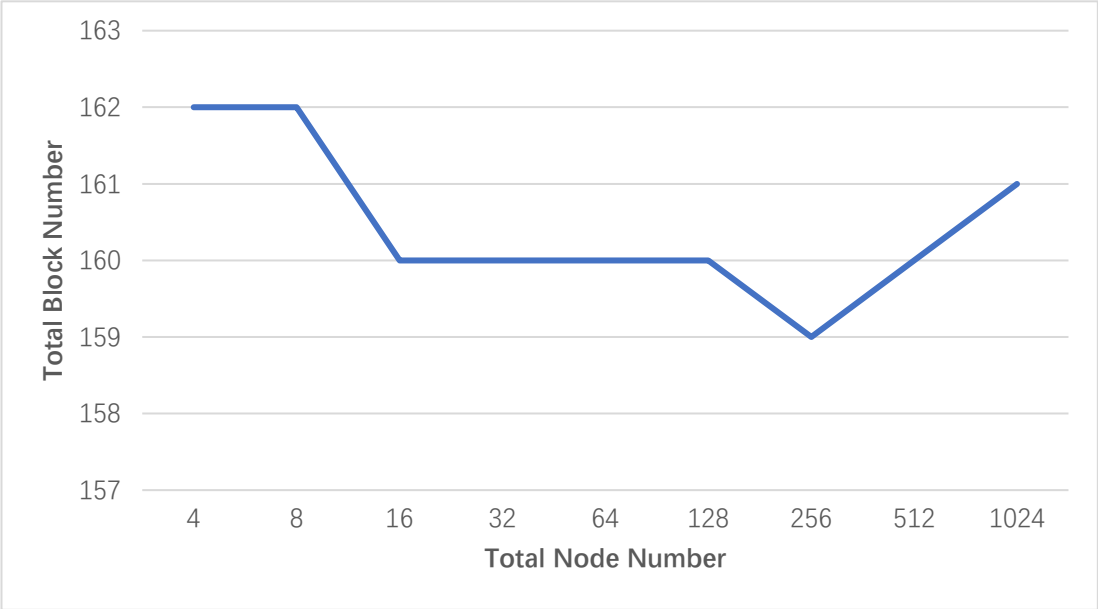


Figure 24. Total block number

In Figure 24, the number of miner nodes has little impact on the Blockchain performance. In our simulation, the node number could be up to 20000. Actually, the increase of node number will provide the system with more computing power, which may cause a shorter mining time. Once the mining time is out of range, the system will increase the difficulty of calculation, which finally keeps the mining time in the given range. As we jump the mining part and set a range for the mining time, the increase of miner number won't influence our simulation in general. But it may bring higher possibility for nodes to create an orphaned block as shown in Figure 25.

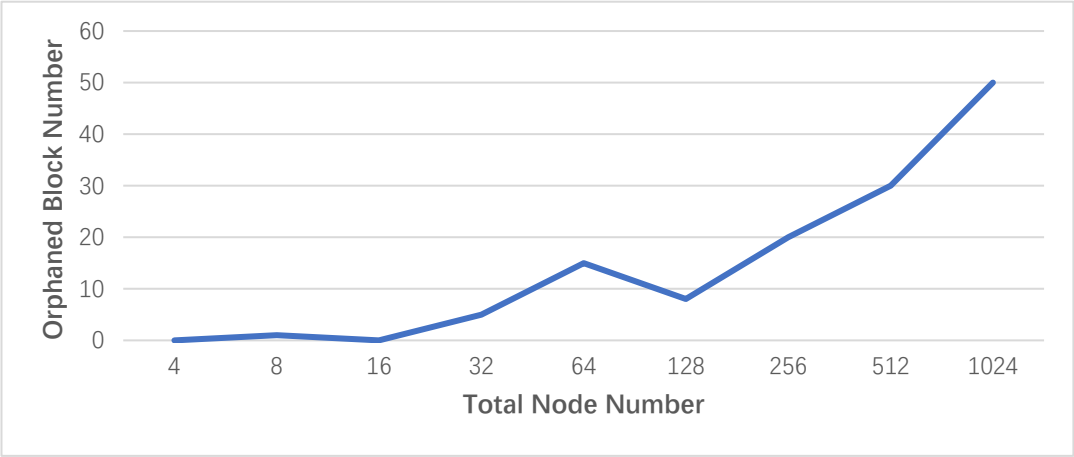


Figure 25. Orphaned block number (96 hours)

5.1.3 Summary

As the results shown in 5.1.2, we can find out that our metrics are effective in QoS of Blockchain, and we can get a clear view on performance of the Blockchain. Our simulation shows that different parameters change the behavior of the whole network. In the future usage, we can figure out more groups of appropriate parameters that show good performance in both clients' and miners' view before the deployment of new blockchains.

5.2 New DAG Structure

DAG (Direct Acyclic Graph) is one of the ideas which aims at improving the performance and scalability of Blockchain. However, new DAG-based chains have different chain structure and consensus protocols, which may bring different performance and scalability. In addition, so far little implementation has been published to confirm the assumption that DAG-based chain outperforms Block-based chain.

In this part, we will introduce a new DAG-based blockchain model. Then we will deploy this model in our simulation framework to test its performance and scalability.

5.2.1 Related Work

In this section, we review and discuss current well-known research to improve blockchain performance with DAG as follows:

IOTA IOTA[24] (Internet of Things Application) is a new kind of blockchain. IOTA's ledger data structure uses DAG, instead of a chain, where the atomic data element is an individual transaction, instead of a block, with aim to support massive transaction data from IOT. It uses unique method to verify transactions, called *Tangle*. Tangle is the distributed account book of IOTA based on DAG structure, rather than a continuous chain structure, adding blocks regularly. Through DAG and Tangle, IOTA is expected to support high transaction throughput (through parallel validation) and avoid miners and transaction fee. In Tangle, each node represents a transaction. The initiator of the transaction has to approve 2 past transactions, and point his own transaction to these two transactions. And also, a little PoW has to be done in case of network attacking. With this design, the duty of verify transactions has been moved from the traditional miners to everyone who wants to use this system. With the continuous development of Tangle, more participants will initiate transactions, the whole system will become securer, the confirmation time will be shortened, and the transactions will be completed faster and faster. The main advantage of IOTA is its high TPS, no miners and transaction fee and extensible. But it still has some problems. IOTA is easy to be attacked at low TPS. It must have enough transactions before it can work properly. In addition, if you want to make your transaction more reliable in other user's view, you should verify the transaction created by trusted node, called Coordinator, which may make the whole system kind of centralized.

Nano/RaiBlocks Nano[65] uses a block-lattice (DAG-based) structure. Each account has its own blockchain (account-chain) equivalent to the account's transaction/balance history. Each account-chain can only be updated by the account's owner; this allows each account-chain to be updated immediately and asynchronously to the rest of the block-lattice, resulting in quick transactions. Each transaction is split into "paid" and "received" payments, which are recorded by the payer and the payee respectively. A set of corresponding receipt and payment records constitute a complete transaction entry. Nano uses DPoS(Delegated

Proof of Stake) to keep the system reliable. Verifier nodes track the state of the entire network by storing block chains for each address. When a published transaction conflicts with the internal state of the verifier node, conflicts or "bifurcations" occur. To resolve the conflict, the node initiates the voting process by notifying other nodes of the conflict. The voting process takes place in a predefined period of time, and each node votes on what it considers to be the correct network state. In order to speed up the processing of small transactions, Nano uses UDP protocol to conduct transactions. Nano's biggest technical advantage is zero transaction fee and instantaneous transfer, low system energy consumption, low network broadband and storage requirements. But at the same time, transactions without handling fees may not stimulate the initiative and enthusiasm of nodes. The maintenance of all-node accounts in block chains requires a great deal of effort from miners. Bitcoin uses the PoW consensus mechanism to reward miners with Bitcoin and handling fees. Nano does not have any incentive mechanism, which makes the calculation power limited and make bad response to a large-scale system attack. Also may lead to the centralization of computing power.

Byteball Byteball[52] does not have a concept of block either. Each new transaction references one or more transactions earlier (parental transactions) by including and signing their hashes. As more transactions are added, the number of confirmations you receive will also increase like snowballing. The link in the transaction forms a DAG. Byteball added a main chain, which creates partial ordering between transactions. When a double spending occurs, an earlier transaction is considered valid. The cost of saving a transaction in a Byteball database is equal to the size of the data stored. Some of the transaction fee is payed to the parent transaction owner, the others are getting by witnesses. The first kind of incentive mechanism makes the users are more likely to choose a recent transaction to be its parent, which will make the DAG structure convergent. The advantage of Byteball is its unlimited message size and its safety. But which confused researchers is its “witness” seems to be a signal of centralization, and also limits the expansion capability and its confirm speed.

5.2.2 Model and Structure

In this section, we present a simplified DAG-based blockchain model to study DAG's improvement on Blockchain performance.

Overall architecture/assumptions

A Directed Acyclic Graph (DAG) is a finite directed graph with no directed cycles. DAG is in a topological order, and has a sequence of the vertices such that every edge is directed from earlier to later in the sequence.

In the traditional DAG-based blockchain network, transactions do not have logical relationship between each other. Most of them need a monitor/main chain or other ways to generate their parent node randomly and keep the chain convergence, which limits the concurrency of the network.

In our research, we found that the transactions are basically connected by the source of these tokens. We can make DAG as a storage structure in different verifiers. As in one transaction the tokens have certain accessible source, the miners can easily get consensus with each other. A simple DAG structure is shown in Figure 27.

After comparison, we choose to use pBFT(Practical Byzantine Fault Tolerance) consensus and make some changes to it. In order to get consensus, we want to use two kinds of character, which are named as *Verifier* and *Collector*. Each verifier node is responsible for: (a) verifying and voting each transaction; and (b) constructing and storing the DAG leger with the verified transactions. One of the collectors counts these votes and takes corresponding action based on the voting results. The overall architecture and consensus process are shown as below.

Workflow

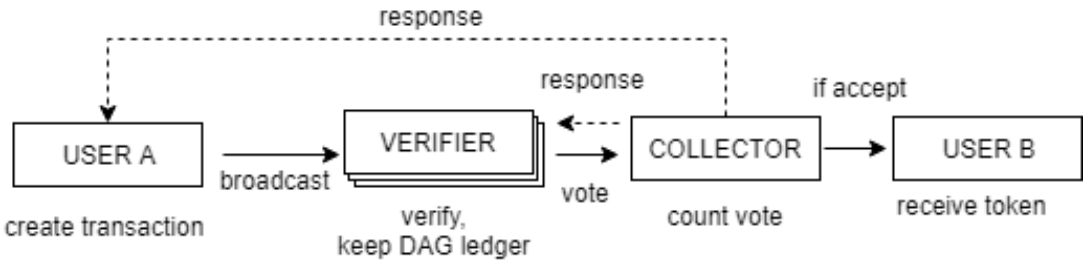


Figure 26. System workflow

- A user (Alice) creates a transaction, hangs on the tokens she wants to exchange, claims the source of these tokens and packages other information.
- A verifier receives the transaction, verifies its validity, records it into the ledger and sends the result to the collector.
- A collector is selected from a group of collectors, e.g., using the Aglorand Sortation algorithm, which collects the results from each verifier. Once reach the threshold, make responses to all the verifiers, sending users and receiving users.
- All verifiers update the status after collecting the validation results from the collector.
- The user (Alice) either take the hanging tokens back to Alice or successful send it out. The receiving user (Bob) write the new tokens into its own account.

Block/Transaction

In our structure, one block contains one transaction, they are equivalent to some degree. The transaction is created/sent by the users, and the block is created by the verifier. This structure may help us to package the transactions in the future research.

Ledger

Each Verifier has a ledger (Figure 27). It contains every transaction in the whole network. Due to the network delay, the transactions may arrive the verifier in different order, but it

will produce a same DAG structure, which realize the concurrency of Blockchain.

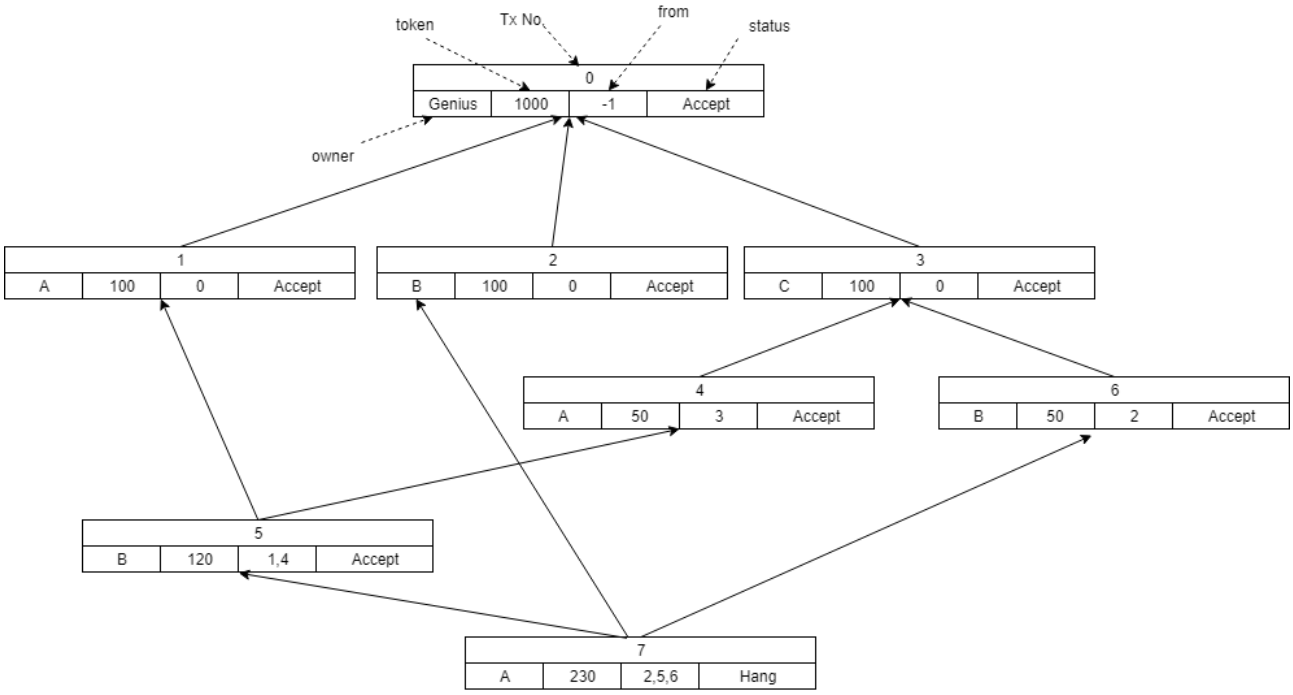


Figure 27. DAG model structure

Account/User

A user is the basic unit of the system. They create transactions and then send them to the Verifier. As shown in Tab I, every user maintains their own account/wallet.

Table 5. User B Account

Tx No.	From	Token	Balance	Hanging
2	Genius	100	0	100
5	A	120	0	120
6	C	50	40	10

Verifier

Verifier is the main part of this structure. All the verifiers are individual and could not communicate with each other. It verifies all the transactions and vote its validity. Also, it

should maintain its own ledger (Figure 27), which may influence its future verify work.

Collector

The collector is the part we use to reach consensus. In our structure, we use a pBFT like consensus. The collector realizes the function of "primary node (leader)". In order to realize decentralization, collector is not a single unit. In our structure, verifier and collector might be the same server/computer. When a transaction is created, it will do some calculating using its time stamp and some other details to randomly decide a verifier to be the collector of this transaction. The detailed cryptology method and its proof will be shown in the future research. In this case, we just random the collector so the verifier has the same opportunity to become a collector.

The collector does not receive the detail of the transactions. It just counts the verify results from the verifier. A threshold is set in the system. In this paper, the threshold for 'accept' is 50% and 30% for 'reject'. Once reach the threshold, it will make responses to all the verifiers and the sending/receiving user.

On the other hand, because of our structure, the transactions create by different users are independent, the verifier and the collector can deal with different transactions at the same time. Therefore, we create multithreading for both verifier and collector to increase the performance of this model further.

There are three status in both account and ledger. 'Hang' is the basic status. Once a transaction is created, the tokens it wants to use will be hanging on. These hanging tokens could not be used in case of double spending. In verifier it means the verifier has already received this transaction and made its own vote, but it hasn't received the response from the collector yet. In this status, any new transactions want to set this transaction as its parent will be vote as 'Reject'. 'Accept' and 'Reject' are the status voting by the verifier and responding by the collector. If 'Accept' the transaction will be finished, the hanging token will be taken away and the receiver will get a message to collect the tokens. If 'Reject' the

hanging token could be take off, but this transaction will still be recorded in the ledger.

Simple DAG ledger has been shown in Figure 27. In Tx1~Tx3, Genius give User A, B, C 100 token each. In Tx4 and Tx6, User C pay for A and B 50 tokens each, these tokens are from Tx3. In Tx5, User A wants to pay 120 tokens for User B, but A cannot get enough token from a single historical Tx, so the source of Tx5 become Tx1 and Tx4. In Tx7, the issue is similar. It shows another status of the transaction, 'Hang', which means the transaction hasn't finish yet.

5.2.3 Consensus and Algorithm

In this section, we deployed the DAG structure we have mentioned in 5.2.2 in our simulation framework. Our simulator is working with three processes.

User Process

- Create a new transaction based on own account. Hang on the tokens have been used. Send the transaction to the Verifiers.
- Receive the message from Collector. Deal with the hanging tokens. Update the account.

Algorithm 1 User Process

```
// Create Tx
Tx.token ← token
while token>0 do
    if account[i].balance>0 then
        if account[i].balance>token then
            account[i].hanging ← token
            account[i].balance ← account[i].balance – token
              token ← 0
        else
            account[i].hanging ← account[i].balance
            token ← token – account[i].balance
            account[i].balance ← 0
        end if
        Tx.parent append i
```

```
15:     end if
16:     i ← i + 1
17: end while
18: Broadcast Tx
19: // Receive message from Collector
20: if Tx is “Accepted” then
21:     j ← 0
22:     while j < len(Tx.parent) do
23:         account[Tx.parent[j]].hanging ← 0
24:         j ← j+1
25:     end while
26: else
27:     j ← 0
28:         while j < len(Tx.parent) do
29:             account[Tx.parent[j]].balance ←
account[Tx.parent[j]].balance + account[Tx.parent[j]].hanging
30:             j ← j+1
31:         end while
32: end if
33: // Receive token
34: account append Tx
```

Verifier Process

- Receive the transaction, add it to waiting list.
- Deal with the waiting list, until reach the max thread or the waiting list is empty. Each message takes 1ms to complete.
- Verify the transaction from users. Write it into the ledger. Send the result to the Collector.
- Check the message from Collector. Change the status in the ledger.

Algorithm 2 Verifier Process

```
1:  // Receive message
2:  waitlist append message
3:  while waitlist is not empty and thread < Maxthread do
4:      message ← pop waitlist
5:      // Message from user
```

```
6:        Tx ← message
7:        Ledger append Tx
8:        Verify Tx.parent
9:        if Tx is legal then
10:         send Tx “Accept” to Collector
11:     else
12:         send Tx “Reject” to Collector
13:     end if
14:     // Message from Collector
15:     Update Ledger[message.id] and Ledger[message.id].parent by message
16: end while
```

Collector Process

- Receive the message, add it to waiting list.
- Deal with the waiting list, until reach the max thread or the waiting list is empty. Each message takes 1ms to complete.
- Count the votes.
- Once reach the threshold, response to the User and Verifiers. Stop this vote.

Algorithm 3 Collector Process

```
1:  // Receive message
2:  waitlist append message
3:  // Count Vote
4:  while waitlist is not empty and thread < Maxthread do
5:      vote ← pop waitlist
6:      id ← vote.id
7:      if voting[id] is not existing then
8:          create voting[id]
9:      end if
10:     if voting[id] is not closed then
11:         if vote is “Accept” then
12:             voting[id].accept ← voting[id].accept + 1
13:         else
14:             voting[id].reject ← voting[id].reject + 1
15:         end if
16:         if voting[id].accept or voting[id].reject reach threshold then
17:             Broadcast voting[id] Result
```

```
18:             Close voting[id]
19:         end if
20:     end if
21: end while
```

There are several parameters we can change during simulation:

Table 6. New DAG structure simulation parameter

Parameters	Description	Default
SIM_TIME	The amount of time that simulation runs	30s
NETWORK_DELAY	The amount of time spent on network for each transaction	[20,50] ms
USER_NUM	The number of Users	1000
MINER_NUM	The number of Verifier/Collector	5
THREAD_NUM	Number of concurrent threads of each verifier/collector	[1,2,4,8]
WORK_DELAY	The time for verifier/collector to deal with one message.	1 ms

With this simulator, we can get some important information about the ledger in each node. Due to our structure's working method, every node gets a similar but not the same ledger (causing by different receiving time). When it is shown as DAG, all the legers are the same except the status in each single transaction.

5.2.4 Analyze and Results

We conduct simulations using different settings, and the result shows below.

Figure 28 shows part of the DAG ledger in verifier. The arrow means Pay By, the circle means one transaction and the number inside is its id. It shows that the ledger in verifier works well as a DAG structure.

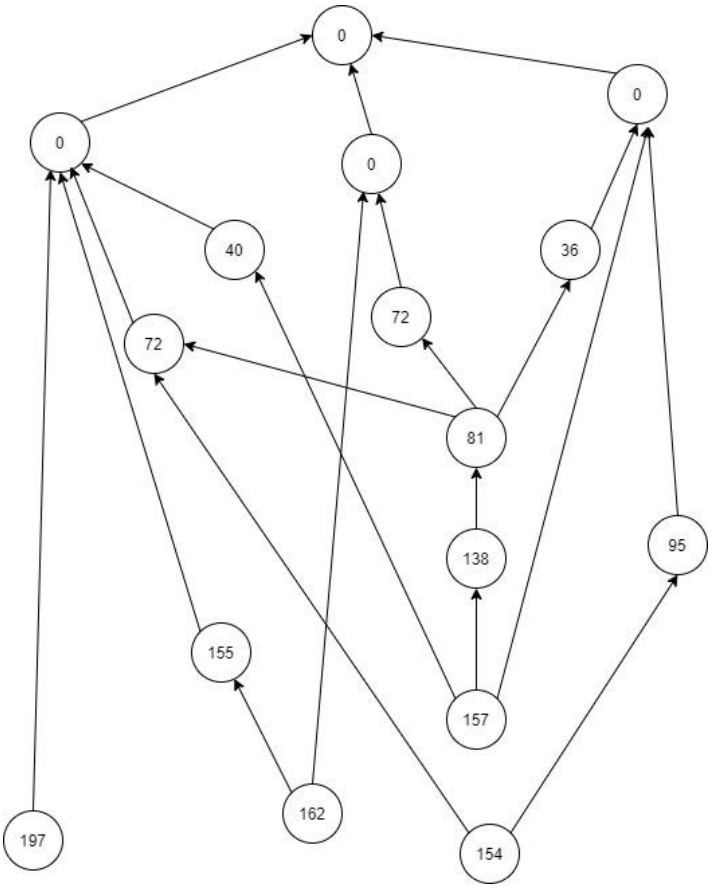


Figure 28. Simulation result (print by Neo4j)

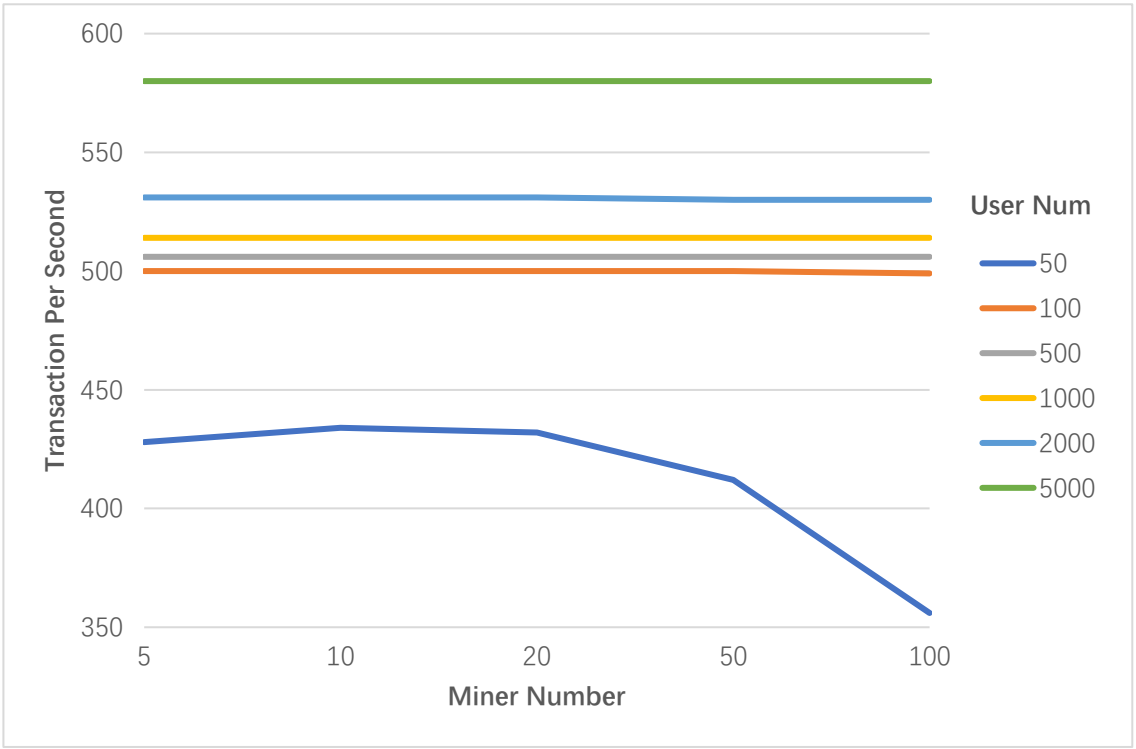


Figure 29. TPS (thread=1)

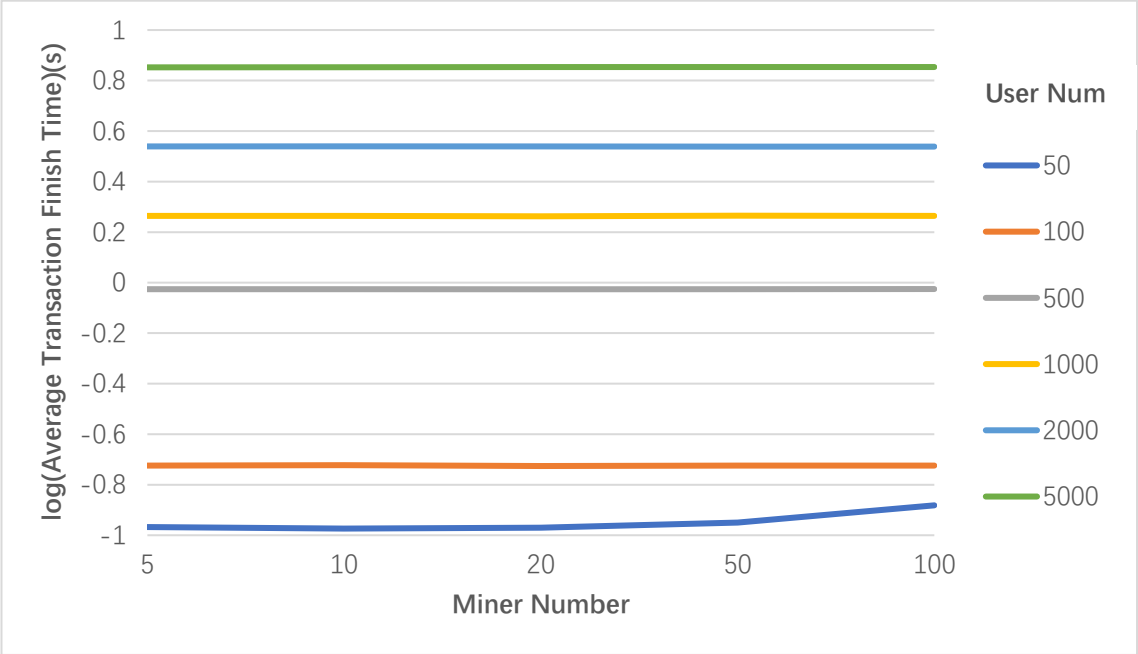


Figure 30. Average transaction finish time (thread=1)

In Figure 29 and Figure 30, we can find that with the increasing miner number, the performance of the network keeps stable. Only when the network scale is very small (user number is lower than the miner number), the performance will be influenced, which means our structure is very balanced and not affected by the number of miners. The more miner increases the security of our network, which shows great decentralization and reliability.

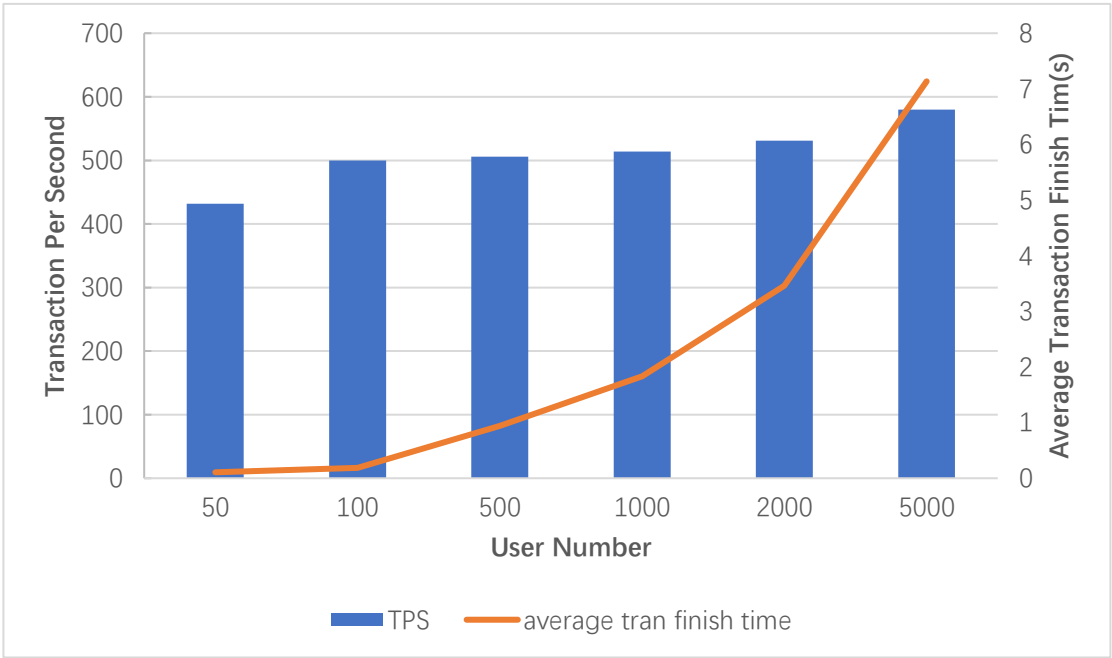


Figure 31. Network performance (Miner number=20)

In Figure 31, as the user number increases, the TPS remains the same and the waiting time of users become much longer (from less than 0.2s to more than 7s).

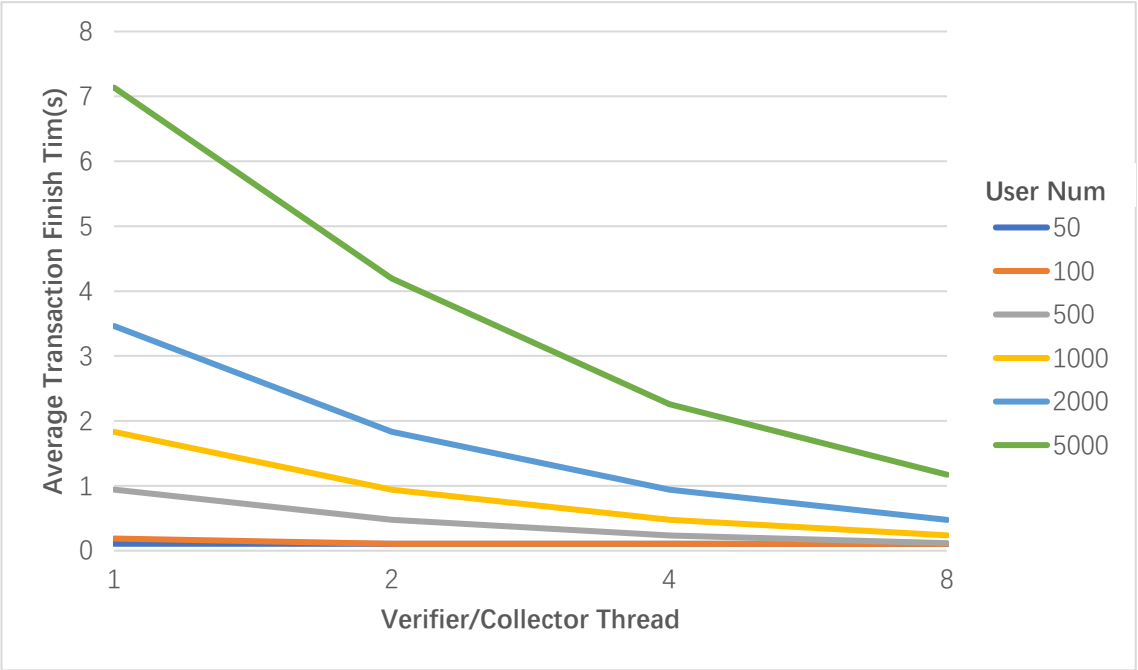


Figure 32. Average transaction finish time (Miner number=20)

In Figure 32, we increase the thread of miners, which improves the handling capacity of one single miner. When the transaction is saturated, the more threads will highly improve the performance of the network, which lowers the waiting time more than 3 times.

In Figure 33, when the scale of network increases, the TPS is limited by the thread of the miner. As the verifier thread and the collector thread are both 8(16 which is the average thread of current PC), the TPS of the network could reach 4000, which indicates a great performance.

In Figure 34, we can find that the waiting time is influenced more by the verifier thread than by the collector thread, which means we can pay more attention to verifier thread in the future deployment.

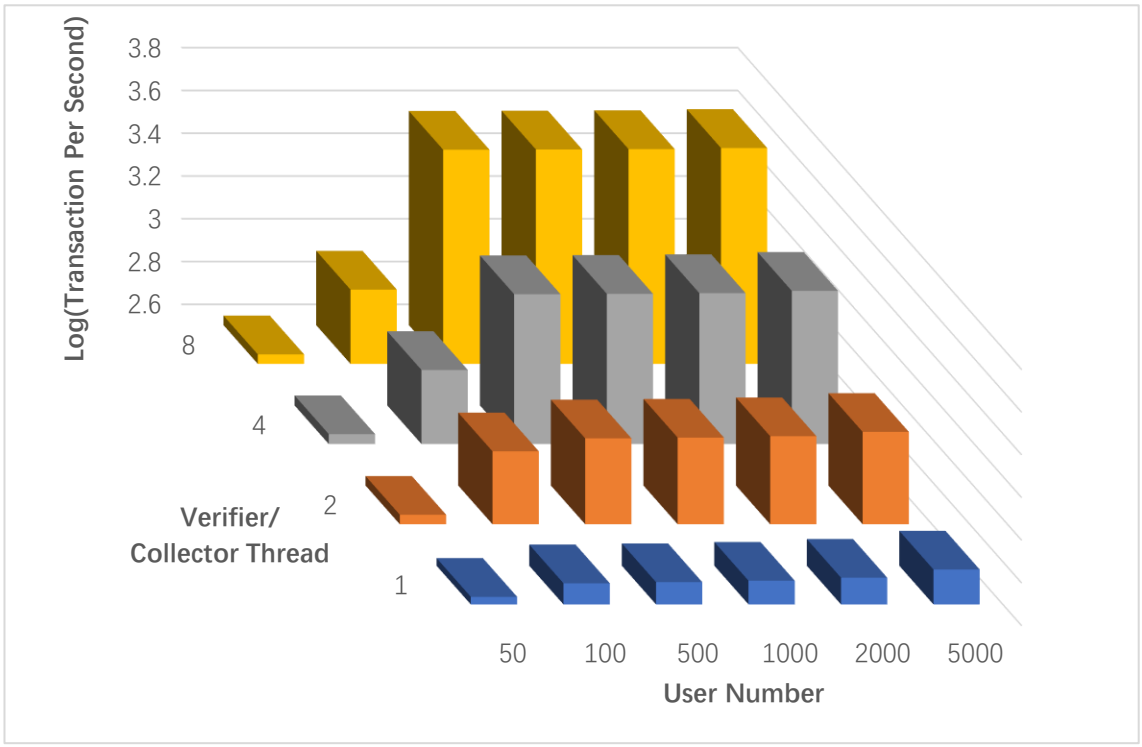


Figure 33. TPS (Miner number=20)

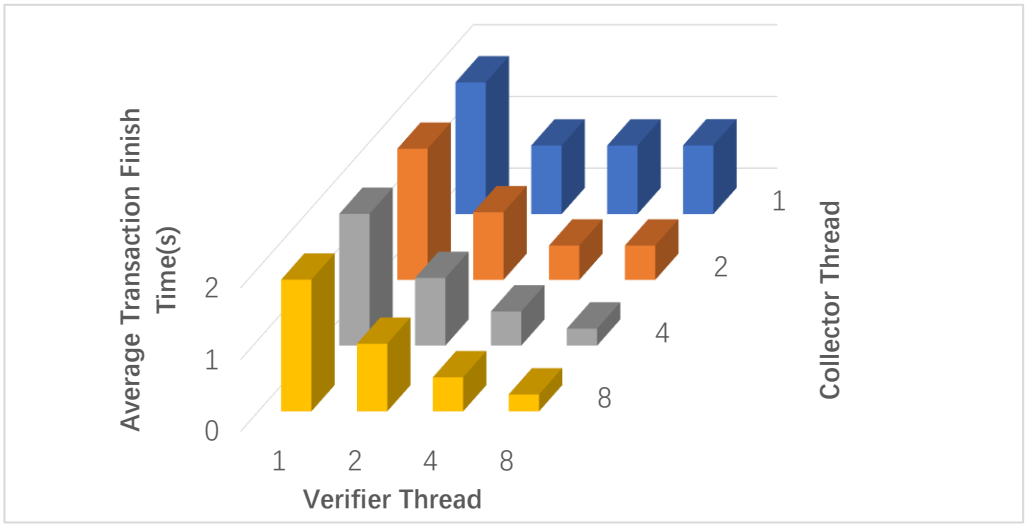


Figure 34. Average transaction finish time (User number=1000, Miner number=20)

Based on these results, we make the following observations:

- Our DAG-based structure is feasible, effective and realizable.
- Miner number doesn't influence the performance of the network. Our structure is stable.
- Increasing the scale of the network does not affect the TPS, however it will make the

time for one transaction finished much longer than expect, but still under 10 seconds.

- The performance of the network is limited by the handling capacity of the miner. As we need 1 ms to deal with one transaction, the TPS is nearly 500 per thread.

5.2.5 Summary

In this part, we propose a new model and a simple consensus protocol on DAG-based blockchain. We also use our simulation framework to test different configurations of our model. The results confirm that our model is able to reach a considerable good performance and it is reliable and effective in different scales of network. As in our DAG structure one user cannot spend one token twice (double spending) or spend the token that it doesn't own, the transactions in the waiting list are not relevant to each other, which means the verifiers can conduct the validation work in parallel without waiting for each other. The limitation of the performance is changed from the blockchain structure to the handling capacity of miners, and the performance is still not influenced by the scale of network. It promises a better extendibility, portability and performance than IOTA and Nano in different cases. The next step of our work is to reduce transit traffic by packaging the transactions as a block and may use ballot encryption or Algorand[39] to elect the collector.

Chapter 6

Use Case 2. IOTA Security Analysis

In current time, more and more specific situations, such as IoT, micropayments, and edge computing, requires high properties on scalability, performance, and cost-efficiency. Therefore, Directed Acyclic Graph (DAG) is designed to fundamentally improve the bottleneck of a traditional blockchain system. Unlike singly linear-chain topology in traditional blockchains, DAG-based blockchain removes the limitation of blocks, by expanding the network through a directed acyclic graph. Newly generated transactions, without being packaged into blocks, directly establish the network in some directions by confirming the parent transactions, in order to get a higher probability to be confirmed by the next transactions. Through several iterative rounds, the main graph is formed with a low probability to be reversed.

Tangle structure is proposed by IOTA[24], one of the pioneers of DAG-based projects, to overcome the original bottlenecks including poor throughput on concurrency, low efficiency on performance, and high-cost on transaction fees. Tangle is formed via the continuously issued transactions, and employs the block-less data structure for transactions rather than traditionally block-based blockchain. It regards the transaction as the smallest elements, and all kinds of atomic operations are competed within the series of transactions, including token transferring, witness validation, path extending and so on. It possesses the properties on:

High throughput Transactions can be attached to tangle from different directions without serious congestion.

High Performance Newly arrived transactions are confirmed by the previous two

transactions via a tiny Proof of Work (PoW) mechanism, where the computer consumption can be ignored with comparison to traditional PoW.

Low cost Transaction fees in tangle-based blockchain is zero to suit for the high frequent situations including IoT, micro payment and edge computing.

Here, we clarify the following DAGs inspired by the architecture and properties of tangle as tangle-based blockchains.

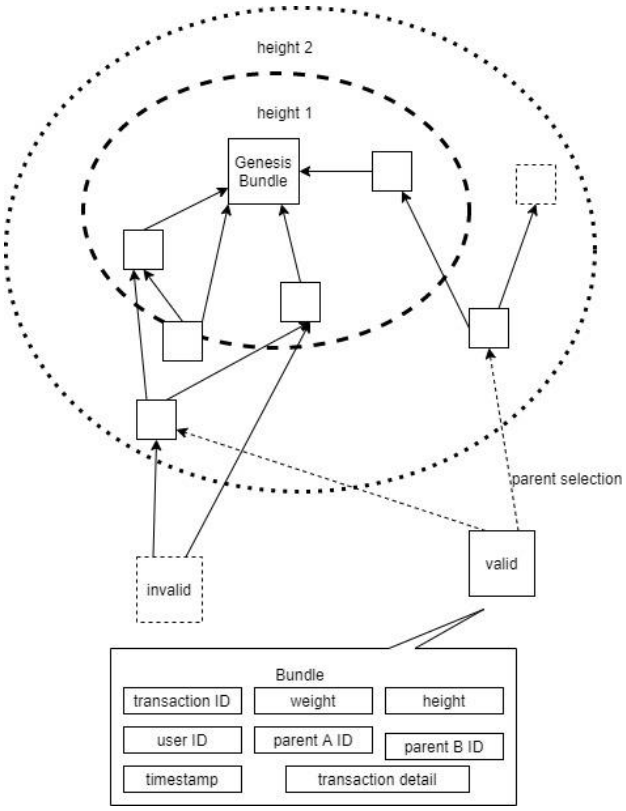


Figure 35. Tangle structure

However, tangle-based blockchains confront the potential threats on the forks of subgraphs in different directions. Following the specified tip selection mechanism, tangle forms a multi-directional expansive network to improve the scalability[45]. More specifically, tangle achieves the delayed eventuality and partial consistency based on the previously confirmed transactions on each direction, instead of an instant eventuality like BFT-style consensus[46]. The gap between delayed eventuality and instant eventuality leaves the possibility of uncertainty and reversibility for attackers[47], the same as other probabilistic protocols such as PoW[1][48]. And the existing chains are always threatened by miners who

own insurmountable computing power or equally the users who can send multiple and continuous transactions. Therefore, newly issued parallel transactions are unpredictably attached to different subgraphs without control, and there is no leading subgraph to maintain a stable structure. The forks, as a result, are always along with the leading subgraph, under the risk of parasite chains attack and double-spending attack[49].

In order to solve the contradiction, an additional centralized Coordinator is embedded in tangle (e.g. IOTA project) to maintain the ordered sequences. All transactions selected by it are immediately considered to be confirmed with 100% confidence. Milestones are periodically sent to nodes by the Coordinator, to reach the consensus across multiple subgraphs for stability and record snapshots by removing useless branches. Nodes accordingly rebuild the Merkle tree which contains the Coordinator's address to verify the milestone. This makes tangle inherently a centralized system. IOTA official claims to cancel this centralized coordinator in the future. However, problems still exist without the Coordinator, no matter what scale it can reach. Tangle will spread into different directions in form of subgraphs under its inherent mechanism. We aim to provide comprehensive analyses on such difficulties. Our contributions are summarized as follows.

Deconstruction of tangle We abstract the features of tangle-based blockchain from the basic actions to the meshed graph network, and deconstruct current tangle projects like IOTA into components including generation of transactions, bundle as a unit, and selection algorithms. Our deconstruction lays the foundation of the following constructions and evaluations.

Construction of attack strategy We define three actions as the basic benchmarks to construct our attack strategies layer by layer. The bottom layer (Layer0) describes the role of each basic action. The middle layer (Layer1) presents the possible behaviors made up by actions. And the top layer (Layer2) provides the attack strategies made up by multiple behaviors.

Evaluations of attacks We evaluate the attacks through multiple metrics to discuss the potential influences, including: 1) Different proportions of behaviours at the same attack

strategy; 2) Different attack strategies at the same parameter configuration; 3) Different parameter configurations at the same strategy.

The rest of this part is organized as follows: Detailed related works are reviewed in 6.1. The constructions of our attack strategies are expanded in 6.2. Based on that, the implementation is shown in 6.3, followed by evaluations in 6.4. Finally, 6.5 concludes our contributions and future work.

6.1 Related Work

DAG[50], as a primitive in mathematics and computer science, is a finite directed graph with no directed cycles. Deeply rooted in the graph theory, DAG-based structure can be applied to various areas including data processing networks, genealogy and version history, citation graphs, data compression, etc. Currently, researchers and developers are trying to bring the DAG into a blockchain, to address the bottleneck on scalability and performance. GHOST[51] as the backbone selection protocol instead of sidechain protocol in the earlier time provides a prototype of the current DAG structure combined with blockchain. Driven from the classic PoW mechanisms which choose the longest chain, GHOST protocol selects the chain who holds the maximum sub-trees[52][53][54][55]. GHOST greatly improves the throughput comparing to the PoW while holding the same block size. Inspired by GHOST, several blockchains replace the subtree structure into graph structure and redesign the whole consensus mechanism and network topology. Rather than focusing on the consensus at the block level, DAG prioritizes consensus at the transaction level in a separate mechanism. The transaction-based structure is inherently suitable for the micro-services, and tangle-based blockchains also inherit the advantages. Although DAG is a competitive player, there are still many technical problems. IOTA provides many strategies for the designation and protection, which are detailedly and explicitly described in[1][24][56][57][58].

Several improvements by researchers are proposed to strengthen the potential weakness

of tangle. Cullen[49] proposes a matrix model to analyzes the efficacy of IOTAs core MCMC algorithm, and present the improvement to resist parasite chain attacks. The matrix model clarifies the formulation for H, to provide the explicit definition in the MCMC algorithm. Ferraro[59] proposes a modified tip selection algorithm to make all honest transactions eventually be confirmed. The hybrid selection algorithm achieves the balance between biased preference for honest tips and high probability for older transactions. Gewu Bu[59] modifies the tip selection mechanism, called G-IOTA, by choosing three verifying transactions at one time instead of two. Through the proposed algorithm, G-IOTA can tolerate several attacks.

Besides tangle, there are also many other DAGs. Byteball[52] proposes the consensus based on a total order within DAG. The uni-direction is achieved by selecting the main chain, to gravitate towards units issued by commonly recognized reputable witnesses. Spectre[60][61][62][63] aims to establish the DAG structure based on concurrent and parallel block creation. It utilizes a recursive voting procedure where every block submits a vote for every pair of blocks. Accepted transactions are confirmed according to the votes. Hedera[65] develops hashgraph consensus algorithm as the underlying model. Inspired by BFT consensus, hashgraph sets the 2/3 as the threshold, where successful confirmation of newly generated transactions requires less more than twothird witnesses from their ancestors. Nano[65] designs a lowlatency cryptocurrency built on block-lattice data structure, where each account has its blockchain. Users, in charge of their accounts, update their blockchains asynchronously and keep track of account balances rather than transaction amounts. All of the above-mentioned DAG-based blockchains offer a high scalability and low/no transaction fees.

6.2 Reconstruction of Tangle

Tangle represents a permission-less network, providing the environment for data shared by all participants. In this section, we deconstruct the architecture of tangle-based

blockchains into three related components, which separately answers the question on how to generate a transaction, why we need to bundle the transactions, and how to select the parent transactions for verification. Based on that, we abstract the key features as the foundation to construct the attack strategies.

Tangle bases on DAG where the vertex represents transaction and the edge represents verification relationship. Instead of the separation process between making transactions by local users and achieving consensus by online miners, tangle integrates these processes into one step. Whenever the transaction is generated and attached to tangle, the consensus is simultaneously launched. Newly generated transactions are continuously attached to the network as the participants' increase, which inevitably forms subgraphs in different directions. In order to prevent the network split into the isolated subcliques, tip selection algorithm is essential to lead the main graph in one direction, maintaining the stability. Here, we provide the skeleton to show how tangle forms and works, with the following procedures.

Generate a transaction

To generate a transaction, some key fields are required to be clarified including *index*, *address*, *bundle*, *trunkTransaction* and *branchTransaction*. The index is linearly increased along with the growth of transactions. The address, as the unique identity of each transaction, is generated via cryptographic sponge function, with the input including subseed, index and security level. Subseed is an 81-tryte derived from seed and index under Keccak-384: *hash (seed + index)*. It should be noted that each address can be only spent once. If the address is used for the second time, it becomes poorly secure with high risk, due to the exposure of private subsided. Therefore, once the user withdraws the tokens from one address, it immediately creates a new address and the index counted 1. To achieve a compete round of withdraw/deposit cycle, multiple pairs of private keys and addresses are necessary. As a consequence, it leads to the concept of bundle.

The transactions are the smallest components serving for further construction in tangle.

Each token transferring and transaction verification are based on the newly generated transaction. All the trends of the network are caused by the behaviours of the transactions. Therefore, we focus on the transactions and analyse the probable behaviours it may happen. The transaction can be used as an honest approver or a malicious approver for the previous transactions. We define three basic actions to build up the behaviors of a transaction in 6.3.

Packaged transactions as Bundle

Bundle, as the basic unit for money transferring, is a top-level construction that links the related transactions into one clique. The bundle itself cannot be broadcast while a collection of individual transactions is broadcast. All transactions can be regarded as part of the bundle, and the metadata is recorded on every single transaction (in *trytes* format) instead of the virtual bundle. In other words, a bundle can be reconstructed from the transaction collection at any time through the fields of *bundle hash, index, trunkTransaction* and *branchTransaction*.

The bundle is the smallest unit serving for money transferring. There are no real entities in the network, and all the steps are proceeding through transactions. Collective transactions inherently share the same seeds from one node, so that we can regard the bundle as a separate vertex in the network. Packaging a collective transaction into one bundle guarantees the security level in an open environment. We abstract the key features of bundle, such as PoW, parent selection, etc. and apply them into transactions for simplicity in a closed testing environment, as shown in 6.3.

Tip selection algorithm

Tip means the newly generated transactions that have not been validated. Tip selection represents the selection strategies of newly generated transactions. The strategy provides the principle of selecting two tips, and it decides the direction of the graph as shown in Figure 35. As mentioned above, transactions in tangle are organized in bundles, and the tail transaction is selected by the approver through the trunkTransaction field for final consistency. The trunkTransaction field connects different transactions in the same bundle.

Each transaction with a higher index can trace back to a lower index. The branchTransaction field inside bundle is slightly different, all of the transactions are filled with same string, except for the field of initial and tail transactions. The initial and tail transactions are generated by tip selection mechanism, as the connection between different bundles.

There are three kinds of Tip selection mechanisms provided in: *Uniform Random*, *Unweighted Random Walk* and *Weighted Random Walk*. Note that the Markov Chain Monte Carlo Algorithm (MCMC) is based on a weighted random walk. For the higher probability of being selected, several matrices are proposed for better evaluation. *height (h)* represents the length of the longest path, equal to the path from genesis to the current transaction. *depth (d)* means the longest reverseoriented path, equal to the path from current transaction to a certain tip. The *cumulative weight (cw)* is counted as the number of transaction v being verified both directly and indirectly. *cw* reflects the probability of a tip being selected by the random walk algorithm. α is the configurable parameter to control the effectiveness of *cw* in MCMC. When α converge towards 0, tip selection becomes uniformly random, while towards 1, tip selection becomes deterministic.

Tip selection describes how a newly generated transaction selects its parents. No matter which detailed mechanism the node chooses, the selection processes are all based on the variants of the random algorithm. The algorithm guarantees a tip can select in the large range without strict rules. We divide the selection algorithm in a general way shown in 6.3.

Mapping to simulation

Based on the deconstruction of tangle, we conclude three insights as the guidelines for our attack simulations in tangle network. Then, we capture the key features on how to reconstruct our attack strategies.

Structure The simulation is based on UTXO model proposed by Bitcoin. The UTXO model is naturally fit for the DAGs since there is no account in DAG, and UTXO is responsible for the beginning and end of each transaction to guarantee the correctness of the balance.

Basic unit the simulation employs the transaction directly as the smallest unit instead of

bundle. Since in a closed environment, the key-exposure problem can be ignored. The transaction-based exchanging provides us a flexible reconstruction.

Topology The simulation follows the original pattern of tangle: every newly generated transaction verifies another two-parent tip by selection mechanism. The selection mechanism refers to various factors such as cumulative weight, level difference and operating time as discussed in the definition 1.

Tip selection mechanism We capture the three most influential factors in tangle including cumulative weight, level difference, and operation time. Newly generated transactions select parent tips according to their joint influence by equations 1.

Attack strategy construction We build the three typical attacks faced by tangle, containing PS, DS, and HB. The attacks are based on the smallest actions layer by layer, which can be referred in the next section.

6.3 Attack Strategies

In this section, we provide three main attack strategies: Parasite Attack (PS), Double Spending Attack (DS) and Hybrid Attack (HB). Each strategy represents a family of concrete attacks that inherit the same foundations. To make it clear, we start from the smallest units to progressively establish a practical attack. We construct the strategies by three layers: layer0 for unit actions, layer1 for atomic behaviors and layer2 for combined attack strategies. Note that, all of the actions and behaviors defined below refer to the malicious node since the honest node can only conduct the definitely honest transactions and behaviors. Therefore, only the malicious/dishonest nodes have multiple combinations. Here we provide the details.

6.3.1 Layer0: Unit Actions

We define the smallest unit actions in the bottom layer, denoted as layer0, the describe the actions a node can select. It can be regarded as a binary selection at each unit action, and the combination of unit actions make up an atomic behavior in layer1. More specifically, we list three-unit actions as the metrics, including:

Action A The unit action A represents whether a newly generated transaction is 1) valid [A_1] or 2) invalid [A_2].

Action B The unit action B represents whether a newly generated transaction is attached to the parent tips 1) by the random selection mechanism [B_1] or 2) by selecting the transactions which are issued by same nodes/entities with itself [B_2]. Here we call B_2 as selfish selection.

Action C The unit action C represents whether a newly generated transaction is selected from 1) the pool with valid transactions [C_1] or 2) the pool with invalid transactions [C_2]. We denote the first one as Valid Pool and the second as Invalid Pool

From the definitions of unit actions, we can see that each action can be represented as a binary selection. We use 1 to represent the independent selections of the first line X_1 where $X \in \{A, B, C\}$. And we use 0 to represent the independent selections of the second line X_2 where $X \in \{A, B, C\}$. Here, we summarize the possible selections in Table 7.

Table 7. User action

Action A	Action B	Action C
Valid Tx [A1]	Random Selection[B1]	Valid Pool [C1]
Invalid Tx[A2]	Selfish Selection[B2]	Invalid Pool[C2]

Seen from Table 7, we can see that each unit action sets are made up by two possible choices to describe the instant state. We denote this process as a binary selection for simplicity and clarity. It should be noted that selfish selection means a newly generated transaction and the parent transaction is issued by the node with the same identity (honest/malicious).

6.3.2 Layer1: Atomic Behaviors

Layer1 is a collective set of various behaviors made up by the different combinations of unit actions. The behavior is used to completely present how to generate a transaction at the initial stage. The behaviors are atomic for the construction of attack strategies and are closely related to the category of attack types. Here, we summarize the feasible atomic behaviors in Table II. Note that, every single behavior covers the unit actions A, B, and C, and we employ the binary selection 0 and 1 to distinguish the combination of unit actions.

Table 8. Atomic behaviors

Binary Selection	Action Combination	Feasibility	Index
111	(A1,B1,C1)	Y	a
110	(A1,B1,C2)	Y	b
101	(A1,B2,C1)	Y	c
100	(A1,B2,C2)	N	-
011	(A2,B1,C1)	Y	d
010	(A2,B1,C2)	Y	e
001	(A2,B2,C1)	Y	f
000	(A2,B2,C1)	N	-

From Table 8, we can see that there are 8 possible atomic behaviors. Take *101* - (A_1, B_2, C_1) as an example, it means the behaviour that: A malicious node generates a valid transaction, being selfishly attached to the parent tips from the invalid pool. These behaviors can be achievable in the simulation without logic error, denoted as the Y in Feasibility column. On the contrary, the behaviors *100* is infeasible, since there is no invalid transaction in the network if the malicious nodes only send valid transactions. And the behaviors *000* is infeasible since for a malicious node, selfishly attaching process from the invalid pool is equal to the random selection from the invalid pool, so that *000* is equal to *010*. Therefore, only 6 of them are feasible, and we mark them with the index from a to f.

6.3.3 Layer2: Combined Attack Strategies

Based on the behaviors in layer1, we construct the attack strategies in layer2. We categorize three types of attacks, including Parasite Attack (PS), Double Spending Attacks (DS) and Hybrid Attacks (HB). The parasite attack means an attacker secretly creates a subtangle with the high weight for the profit. It may reverse the main tangle when the newly generated

transactions are widely distributed. Double spending attack means an attacker continuously spends the same transaction more than one time. The attack splits tangle into two branches so that one can spend a coin multiple times in these different branches.

We define PS and DS are pure attacks without mutual overlaps, and HB are the attacks including overlapping behaviors, such as f (Detailed explanation in the later paragraph). Before the construction of attack strategies, we provide the decision principle, denoted as Φ, on how to categorize the attacking types. After looking through the whole logic, we conclude four stages of decision process shown in the Equation 1.

$$\left.\begin{array}{c}\text{Confusion behavior} - \text{stage1} \\ \text{Send the invalid Tx} - \text{stage2} \\ \text{Verify the invalid Tx} - \text{stage3} \\ \text{Overlapping behavior} - \text{stage4}\end{array}\right] \Rightarrow \Phi \tag{1}$$

For the stage1, the confusion behavior means the malicious node pretends to act as an honest node, such as the behavior a: A malicious node sends a valid transaction, being randomly attached to the parent tips from the valid pool. We cannot obtain any useful knowledge to distinguish whether the transaction is issued by an honest or malicious node. stage2 represents the malicious node sends valid transactions or not, the related behaviors in layer2 are (d, e, f). stage3 refers to the parent tip selection, and the related behaviors are (b, e). stage4 provides the overlapping behaviors including (c, f). Therefore, the general decision principle Φ is shown in Equation 2. The symbol"-" means do not exist.

$$\Phi: (a,-)|(d,e,f,-)|(b,e,-)|(c,f,-) \tag{2}$$

From the Φ, we provide the concrete decision principle for the PS, DS, and HB, which are separately denoted as Φ[PS], Φ[DS] and Φ[HB] as in the Equation 3. The key principle of PS is to selfishly select the parent transactions, while the key of DS is to send/verify the invalid transactions. But there are some overlapping behaviors in the strategy and some logic errors. Therefore, we provide each attacking type by the specified decision principles. The detailed principle is listed in the second column of Table 9.

$$\{\, \Phi \mid \Phi[PS], \Phi[DS], \Phi[HB] \,\} \tag{3}$$

Table 9. Attack Strategies

Attacking Types	Decision Principle	Confusion Behavior	Feasible Behavior	Attack Strategies
	(a,-)|(d,e,f,-)|(b,e,-)|(c,f,-)			
Parasite Attack(PS)	(a,-)| - | - |(c)	a	c,f	c,ac(2)
Double Spending(DS)	(a,-)|(d,e)|(b,e)|-	a	b,d,e,f	e,ae,bd,de,abd,ade,bde,abde(7)
Hybrid Attack(HB)	(a,-)|(d,e,f)\(b,e)|(c,f)	a	f	ce,bf,ef,cef,bcf,bef,bce,def,cde,bdf,bcd,aef,acef,abf,abcf,ace,abef,abce,adef,acde,abdf,abcd(22)

Based on the decision principles for each attacking type, we back to the reason why the behavior f is an overlapping behavior. The behaviors f means: A malicious node sends an invalid transaction, being selfishly attached to the parent tips from the valid pool. On the one hand, f selfishly selects its parent tips, satisfying the condition of PS (Φ[PS]). On the other side, f sends an invalid transaction to the network, satisfying the condition of DS (Φ[DS]). Therefore, f is an overlapping behavior applied in the hybrid attacks (Φ[HB]).

6.4 Experiment Design

6.4.1 Parameters and Notations

In this subsection, we define the notations used in our implementations and tests. Actually, there two types of parameters in the system, the first are binary parameters, such as (A_1, A_2) as discussed in 6.3, used for the construction of attacks. Since the transactions are atomic in the simulation, we ignore the duplicated introductions. The second is the continuous parameters such as operating time, number of total transactions and so on. They are used for adjusting the configurations during the simulations to obtain the results. The related continuous parameters are denoted below.

The parameters include total transactions *T*, the honest transaction *H*, the invalid

Transaction *F*, the interval between two newly generated transaction *D*, the time of PoW *I*, the height of block *h*, simulation operating time $\mathcal{T}$, level difference $\mathcal{L}$ and cumulative weight $\mathcal{W}$. Besides, there are derived parameters: strategy space $\mathbb{S}$, transaction generation speed $T/(D+I)$, the ratio of invalid transaction and the total transaction $Ratio(\mathcal{F}) = F/T$ and the ratio between differentbehaviors in one strategy $Ratio(\mathcal{B}) = x:y:z$, where xyz is depended on the initial settings when launching the attacks.

6.4.2 Key Principles

The growth of a DAG is based on the continuously increasing transactions. The old transactions will get weighted, also denoted as *cumulative weight (cw)*, whenever the newly generated transactions are attached. There are two sides need to be considered: the configuration of the unit weight and the methods to select the parent tips. For the first side, the unit weight of every single transaction randomly varies from 1 to 4, to provide a better simulation for the real scenario. For the second side, a parent selection mechanism is required to take the key metrics into consideration which contains the cumulative weight, operation time, level difference (Level represents the transactions with the same height). Here, we present the definition of these three metrics.

Level difference Denoted as $\mathcal{L}$, level represents the transactions that identified the same height. Level Difference is the distance of height between current tips and the selected parent transactions.

Cumulative weight Denoted as $\mathcal{W}$, the weight is calculated as an accumulated value each time the newly generated transactions attached. The unit weight of each transaction varies from 1 to 4. And the cumulative weight is the sum of the weights from its attached transactions.

Operation time Denoted as $\mathcal{T}$, the operation time represents the executed time of a transaction since it generated. A transaction will be discarded when times out.

Definition 1 (Mechanism for Tip Selection).

$$p = 3 * |15 - \mathcal{W}| + \frac{100}{5^{\mathcal{L}}} - 1.5^{\frac{\mathcal{T}}{60}} \ (p \geq 0)$$

$$where, \omega[1] = \omega_c \ (\mathcal{L} = 1)$$

$$\omega[i] = 0.8\omega[i-1] \ (\mathcal{L} = 2 - 6)$$

$$= 0.9\omega[i-1] \ (\mathcal{L} = 7 - 16)$$

$$= 0.01\omega[1] \ (\mathcal{L} = 17 - 29)$$

$$\mathcal{W} = \mathcal{W} + \omega[\mathcal{L}]$$

where $\mathcal{L}$ is height difference, $\mathcal{W}$ is cumulative weight, ω represents individual round weight, and ω_c is current weight. p represents tip selection probability.

From the definition 1, we can see that the selection probability p is mainly influenced by three factors: $\mathcal{L}$, $\mathcal{W}$ and $\mathcal{T}$. $\mathcal{L}$ varies inversely with the p, which means a tip tends to be impossibly selected as the level difference increases. $\mathcal{W}$ is an iterative algorithm within three intervals according to $\mathcal{L}$. The equation at different intervals has a different decay rate. The tips need to be smoothly decayed with a small level difference (which means near to the latest transaction) while be sharply decayed at the high difference. $\mathcal{T}$ is used to prevent the tip from being suspended for too long.

Definition 2 (Decision for Invalid Transaction). *A newly generated transaction will be discarded, as $Tx = \perp$, when triggering the conditions:*

$$\{Tx = \perp \ | \mathcal{L} > 30 \ || \ \mathcal{W} < 30 \cap \mathcal{T} > 1000s\}$$

where $\mathcal{L}$ is the height difference, $\mathcal{W}$ is the cumulative weight. $\mathcal{T}$ represents the operating time.

From the definition 2, we can see that a newly generated transaction will be decided as invalid when exceeding the thresholds either on the specified level difference (30) or on the operation time (1000 s). Our simulations and evaluations of the attack strategies only consider valid transactions.

6.4.3 Implementation Logic

Our implementation is based on the basic logic of tangle-based blockchains and the reconstruction of the attack strategies. We provide detailed workflows including receiving the transactions from peer nodes, generating/sending new transactions, and launching the attack strategies.

Launch the attack strategy

- Configure all the initial parameters including total transactions, $Ratio(\mathcal{F})$, $Ratio(\mathcal{B})$, and operating time $\mathcal{T}$.
- Select the attack types based on strategies from the behaviors and the actions (see in 6.3).
- Set different $Ratio(\mathcal{F})$, $Ratio(\mathcal{B})$ for the test goals (see in 6.4.4).
- Launch the atttack transactions by Send New Transactions and Receive New Transactions.
- Collect the results for evaluation and analysis. (See in 6.4).

Receive the transactions

- Listen to the peer nodes to receive the transactions, with up to n transactions per second.
- Put the valid transactions into the valid pool and the invalid transaction into the invalid pool for the verification.
- Calculate the maximum height of the current DAG.
- Count the cumulative weight $\mathcal{W}$ for the parent transactions through weight iteration in Definition 1.
- Remove the timeout/expired transactions from the transaction pool according to the Definition 2 and change the current status of the transaction.

Send new transactions

- Generate transactions with fixed parameters including height, weight, timestamp.
- Select two parent tips to verify according to tip selection mechanism in Definition 1.
- Launch the PoW verification for attach tips.
- Broadcast the transaction for times.

6.4.4 Implementation Goals

Goal I The first goal aims to test the influence of hybrid attacks (HB) across different strategies. The initial configurations include the attack strategies of each type. More specifically, we randomly select three typical strategies: ace, abe, and ade. There are three testing sets (Set1, Set2, Set3) in this section. The main variables include different Ratio(F) and the attack strategies $\mathbb{S}$ with corresponding $Ratio(\mathcal{F})$. Table 10 presents the detailed variables of this testing.

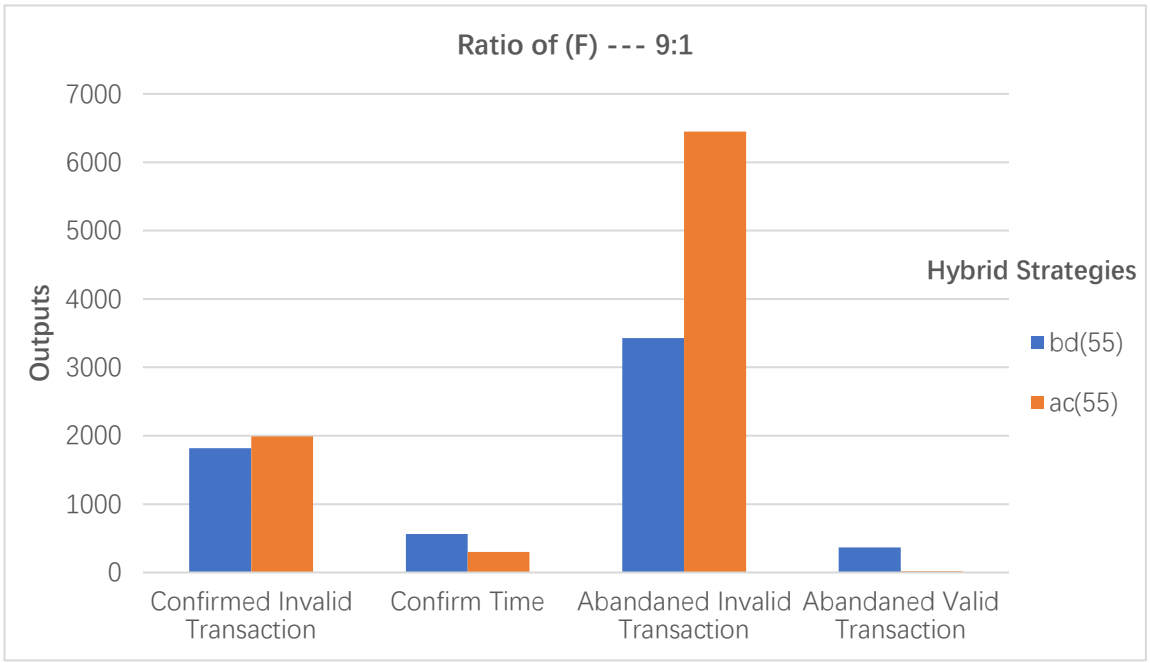
Goal II The second goal is going to test the influence of the different $Ratio(\mathcal{F})$ of the same strategy. The initial configurations include the attack strategies of each type. We randomly select three typical strategies: ace, abe, and ade. There are three testing sets (Set4, Set5, Set6) in this section. The main variables include total transactions T and different $Ratio(\mathcal{F})$. Table 10 presents the detailed variables of the testing.

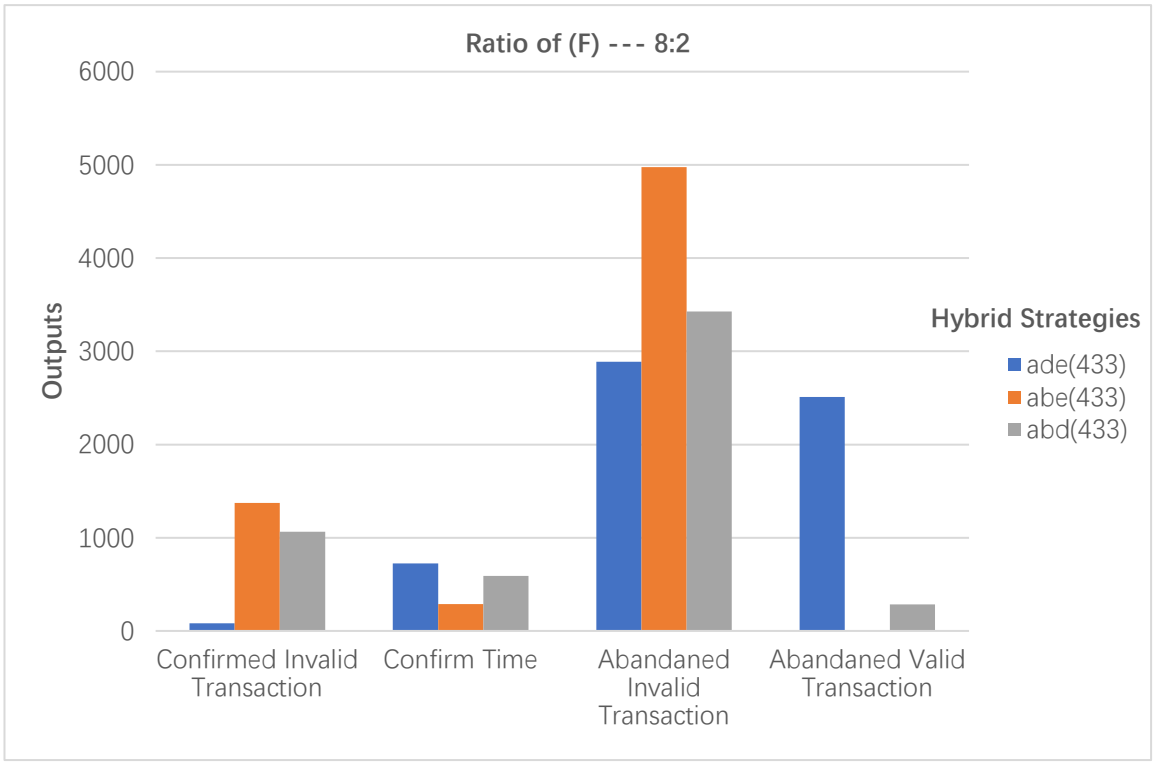
Goal III The third goal is aimed to test the influence of the different $Ratio(\mathcal{B})$ of the same strategy. The initial configurations include the attack strategies of each type and the ratio of behaviours behind the strategy. More specifically, we randomly select three typical strategies: ace, abe, and ade. The ratio of behaviors is 6 : 2 : 2. The total interval time $D + I$ is made by PoW time of each transaction I and the interval time between two transactions D. There are three testing sets (Set7, Set8, Set9) in this section. The main variables include total transactions T and the different $Ratio(\mathcal{F})$. Table 10 presents the detailed variables of the testing.

Table 10. Configurations on goals

Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$	Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$
	Testing Set 1	HB			Testing Set 2	HB	
100	20%	bd	5:5	100	20%	ade	4:3:3
100	20%	be	5:5	100	20%	abe	4:3:3
100	20%	-	-	100	20%	abd	4:4:3
Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$	Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$
	Testing Set 3	HB			Testing Set 4	PS/DS	
100	20%	e	-	100	10%	*	8:1:1
100	20%	abcd	4:2:2:2	100	10%	*	6:2:2
100	20%	abdf	4:2:2:2	100	10%	*	4:3:3
-	-	-	-	100	10%	*	6:3:1
-	-	-	-	100	10%	*	6:1:3
Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$	Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$
	Testing Set 5	PS/DS			Testing Set 6	PS/DS	
100	20%	*	8:1:1	100	30%	*	8:1:1
100	20%	*	6:2:2	100	30%	*	6:2:2
100	20%	*	4:3:3	100	30%	*	4:3:3
100	20%	*	6:3:1	100	30%	*	6:3:1
100	20%	*	6:1:3	100	30%	*	6:1:3
Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$\mathcal{F}$	Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$\mathcal{F}$
	Testing Set 7				Testing Set 8		
20	10%	*	2	20	20%	*	4
50	10%	*	5	50	20%	*	10
100	10%	*	10	100	20%	*	20
200	10%	*	20	200	20%	*	40
Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$\mathcal{F}$	Total Node	$Ratio(\mathcal{F})$		$Ratio(\mathcal{B})$
	Testing Set 9				Testing Set 11		
20	30%	*	6	20	10%, 20%, 30%		8:2
50	30%	*	15	50	10%, 20%, 30%		8:2
100	30%	*	30	100	10%, 20%, 30%		8:2
200	30%	*	60	200	10%, 20%, 30%		8:2
Total Node	$Ratio(\mathcal{F})$	$\mathbb{S}$	$Ratio(\mathcal{B})$	Total Node	$Ratio(\mathcal{F})$		$Ratio(\mathcal{B})$
	Testing Set 10				Testing Set 12		
100	10%	ac	9:1	100	10%, 20%, 30%		9:1
100	10%	ac	8:2	100	10%, 20%, 30%		8:2
100	10%	ac	7:3	100	10%, 20%, 30%		7:3
100	10%	ac	6:4	100	10%, 20%, 30%		6:4
100	10%	ac	5:5	100	10%, 20%, 30%		5:5
100	10%	ac	4:6	100	10%, 20%, 30%		4:6

6.5 Testing Results Analysis





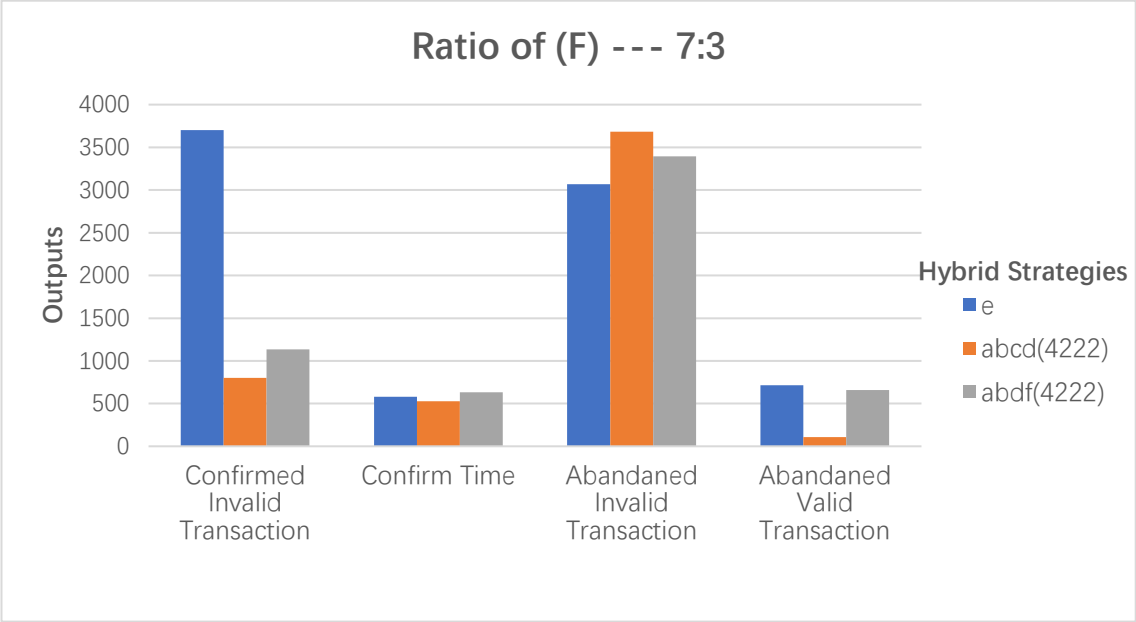
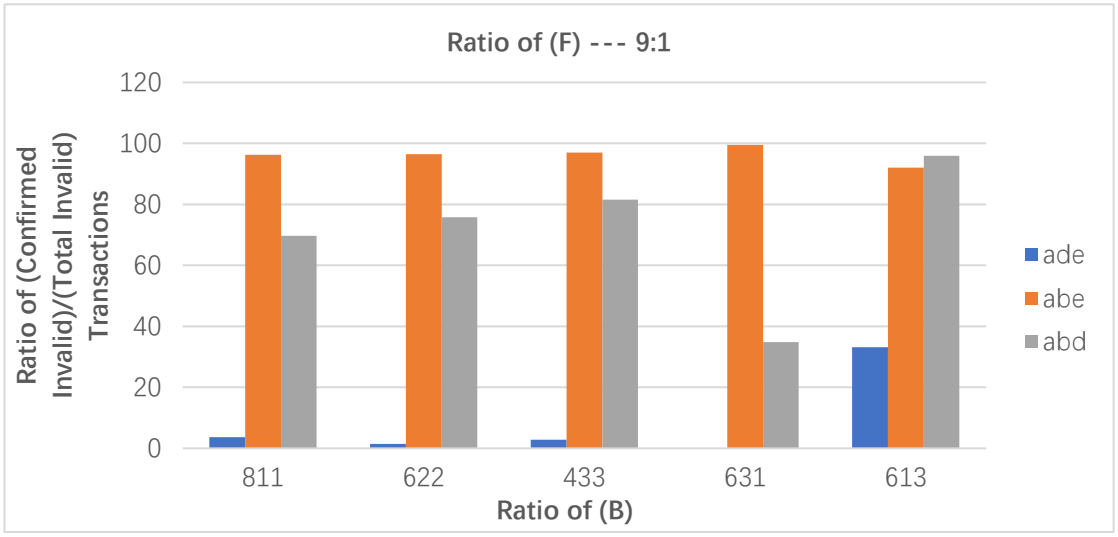


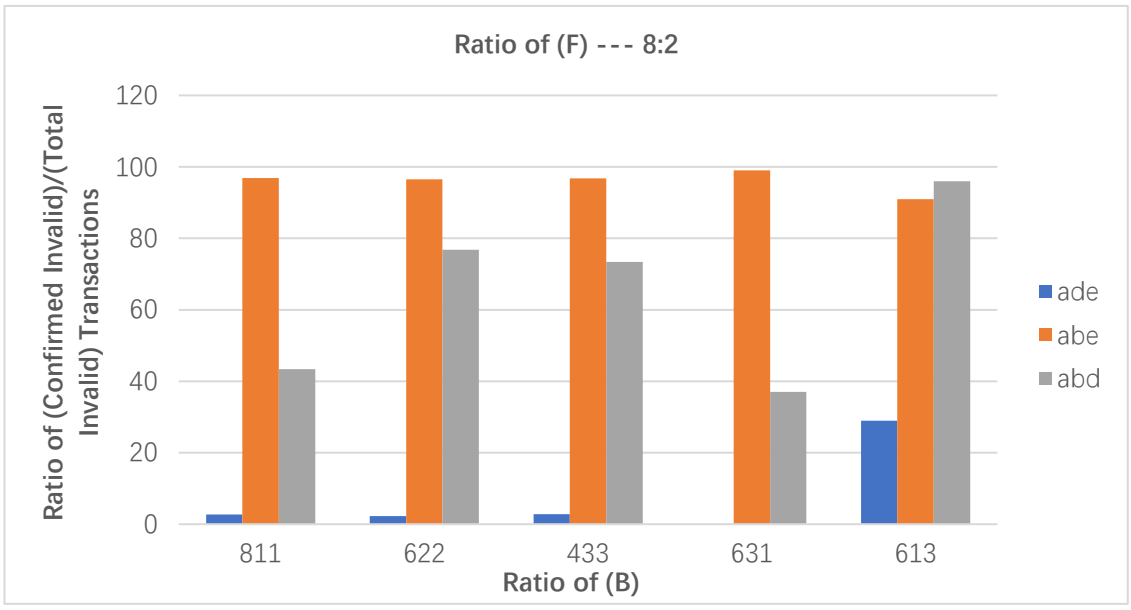
Figure 36. Testing results I

Result I

In the simulation I, we set eight attack strategies $\{\mathbb{S}|\ bd, be, ac, abe, ade, abd, e, abcd, abdf\}$, the corresponding $Ratio(\mathcal{B})$ of each strategy, and three $Ratio(\mathcal{F})$: $\{\mathcal{F}\ |\ 10\%\}$ as the input parameters. The outputs contain confirmed invalid transactions, confirm time, abandoned invalid transactions and abandoned valid transactions. Detailed data and other outputs can be referenced from the raw results.

From the Result I in Figure 36, we can find the trend caused by different factors. (1) For the same strategies, no matter how it is made up, such as ade,e and $abcd$, the confirmed invalid transactions are increasing with the number of malicious nodes in a positive correlation. The confirm time varies in a range of 200-800s. The abandoned transactions significantly increase with the number of malicious nodes. (2) For the different hybrid strategies, $Ratio(\mathcal{F})$ has different influences on them. Several strategies are sensitive to the changes like abe, be. (3) The abandoned invalid transactions and valid transactions increase at the same time along with the modifications of $Ratio(\mathcal{F})$ where malicious nodes have a significant influence.





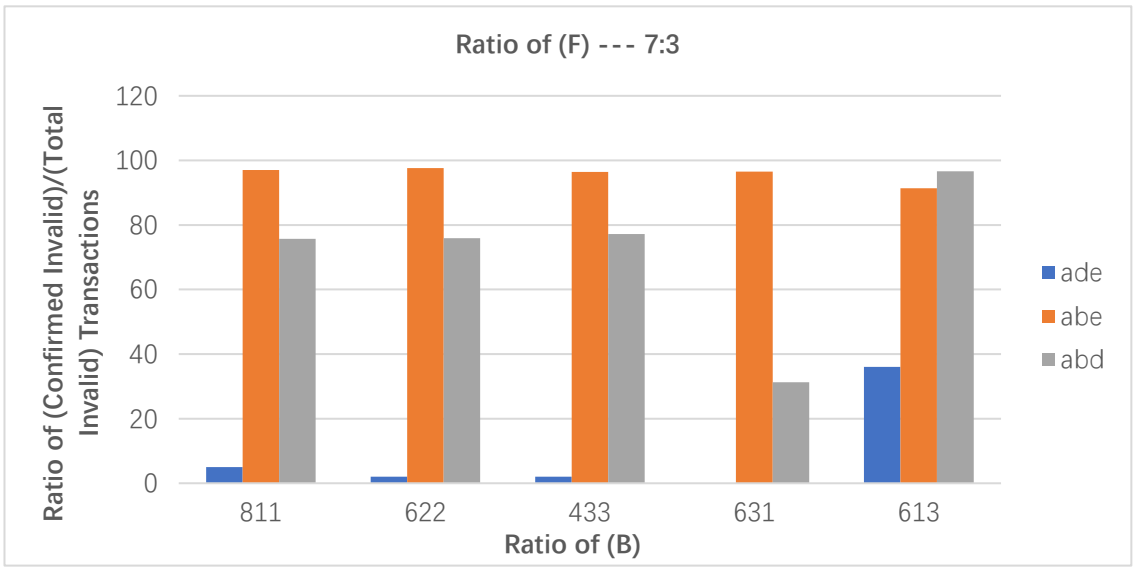


Figure 37. Testing results II

Result II

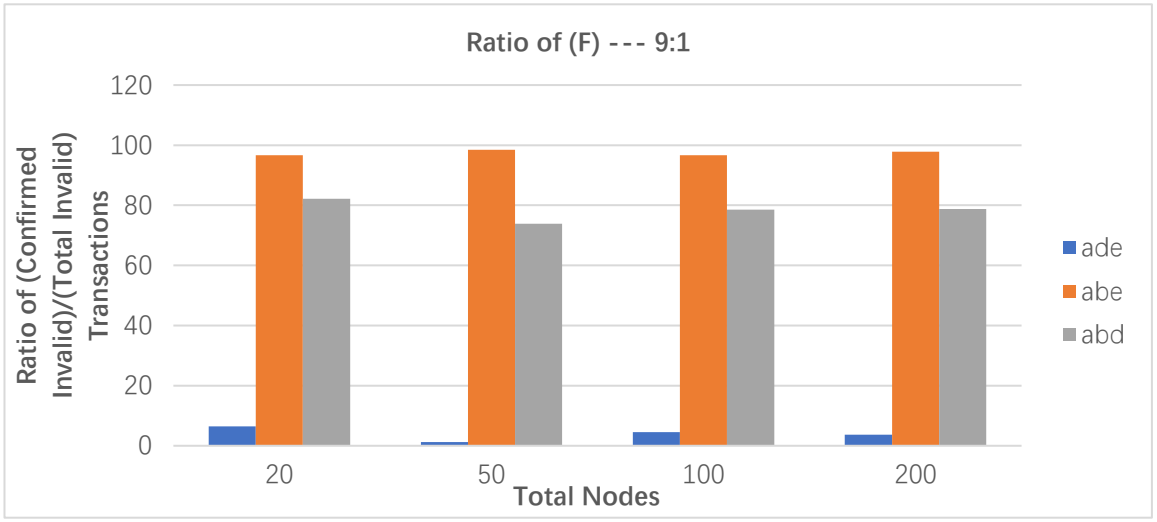
In the simulation II, we set three attack strategies $\{\mathbb{S} \mid abe, ade, abd\}$, three $Ratio(\mathcal{F})$: $\{\mathcal{F} \mid 10\%, 20\%, 30\%\}$, and five $Ratio(\mathcal{B})$: $\{\mathcal{B} \mid 8{:}\,1{:}\,1, 6{:}\,2{:}\,2, 4{:}\,3{:}\,3, 6{:}\,3{:}\,1, 6{:}\,1{:}\,3\}$ as the input parameters. The outputs are the ratio between invalid transactions and total transactions.

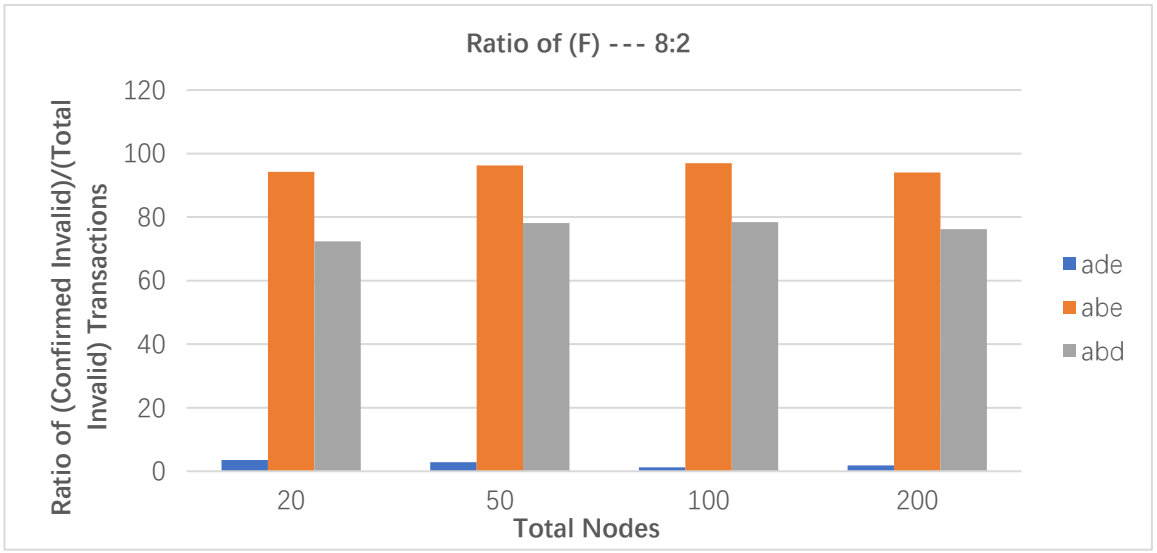
From the Result II in Figure 37 we can find that (1) For the $Ratio(\mathcal{F})$ on the same strategies, such as ade (the blue column in the histogram), invalid transactions will significantly increase along with the malicious nodes. The trend is determinate for such situations. (2) For the different strategies, we can find that the $Ratio(\mathcal{F})$ has different influences on them. ade varies monotonously with the ratio, while the other two have a peak value at a certain ratio. (3) The attack is sensitive to some behaviors such as b. Invalid transactions in strategies containing b (abe, abd) are significantly that without b.

Result III

In the simulation III, we set three attack strategies $\{\mathbb{S} \mid abe, ade, abd\}$ as the basic testing strategy with initial $Ratio(\mathcal{B})$: $\{\mathcal{B} \mid 6:2:2\}$. There are three $Ratio(\mathcal{F})$: $\{\mathcal{F} \mid 10\%, 20\%, 30\%\}$ and four sets of total nodes $\{Tx \mid 20, 50, 100, 200\}$ as the input parameters. The outputs are the ratio between confirmed invalid transactions and total transactions. The ratio provides a direct and visualized relationship.

From the Result III in Figure 38, we can find (1) For the same strategy, such as ade (the blue column in the histogram), the trend of ratio is relatively stable under different $Ratio(\mathcal{F})$. (2) The ratio maintains stable when the $Ratio(\mathcal{B})$ increases. This also means the ratio of invalid with total transactions varies slightly with the malicious nodes. The number of malicious nodes has little influence on the ratio. (3) For the different strategies, some strategies like ade are sensitive to variations than others like abe, abd. The results show a significant difference in these strategies.





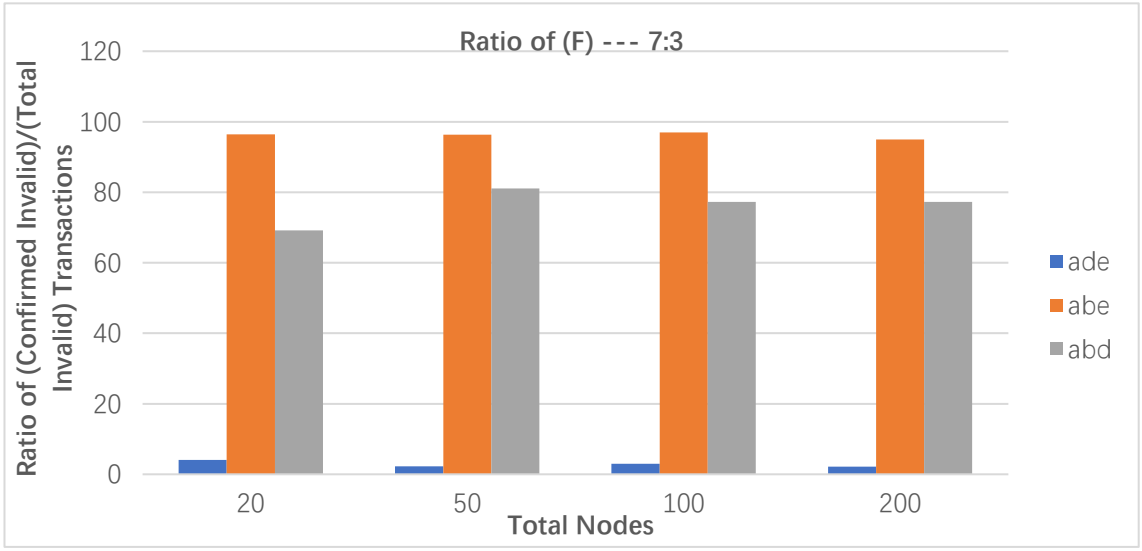


Figure 38. Testing results III

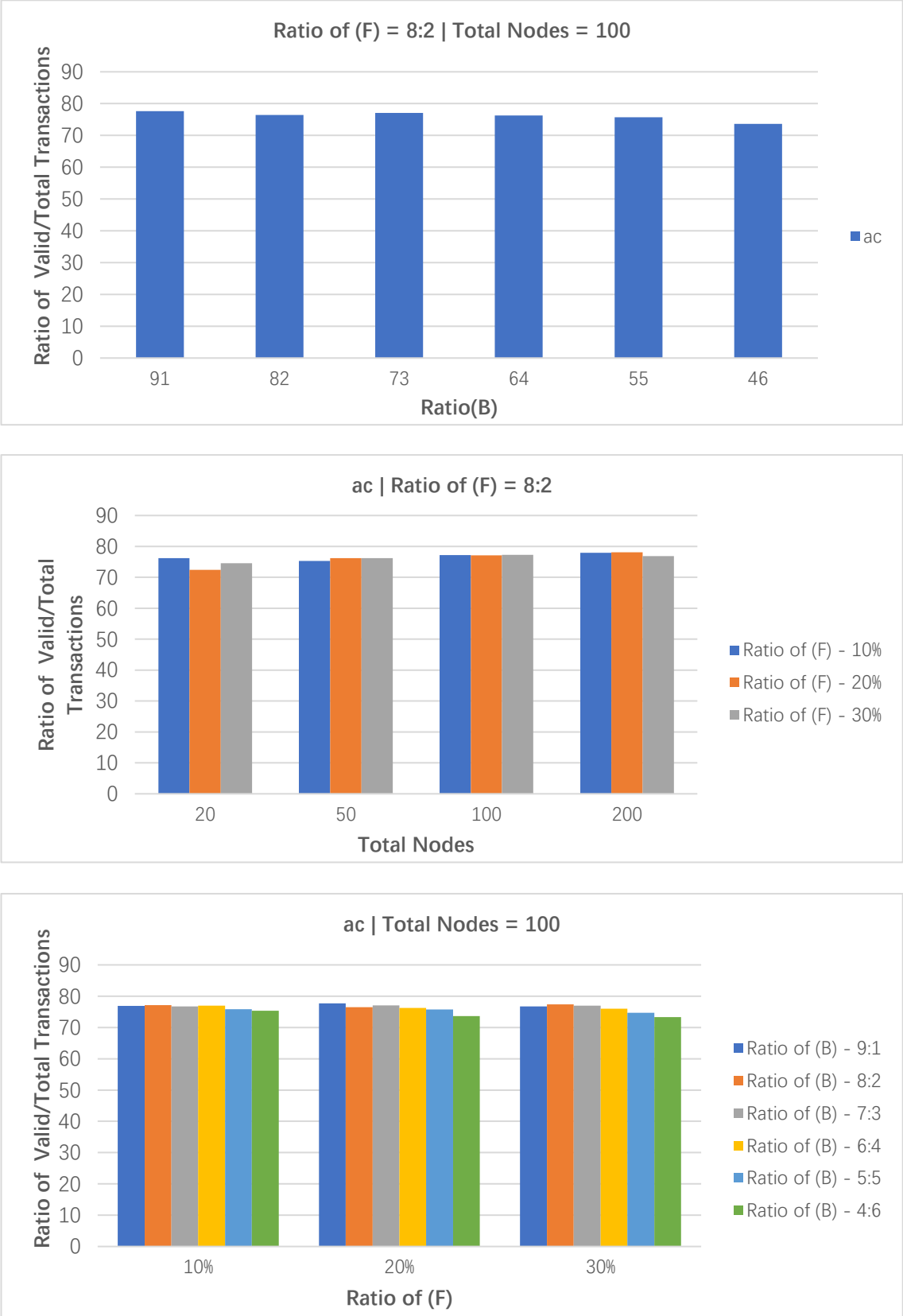


Figure 39. Testing result IV

Result IV

In the simulation IV, we set only one attack strategies $\{\mathbb{S} \mid ac\}$ as the basic testing strategy. The first test initializes 100 total nodes, $Ratio(\mathcal{F})$: $\{\mathcal{F} \mid 10\%\}$ and provides six $Ratio(\mathcal{B})\{\mathcal{B} \mid 9{:}\,1, 8{:}\,2, 7{:}\,3, 6{:}\,4, 5{:}\,5, 4{:}\,6\}$. The second test sets three $Ratio(\mathcal{F})$: $\{\mathcal{F} \mid 10\%, 20\%, 30\%\}$, $Ratio(\mathcal{B})\{\mathcal{B} \mid 8{:}\,2\}$ and four sets of total nodes $\{Tx \mid 20, 50, 100, 200\}$. The third setsinitializes 100 total nodes, $Ratio(\mathcal{F})$: $\{\mathcal{F} \mid 10\%, 20\%, 30\%\}$ and $Ratio(\mathcal{B})\{\mathcal{B} \mid 9{:}\,1, 8{:}\,2, 7{:}\,3, 6{:}\,4, 5{:}\,5, 4{:}\,6\}$ as the input parameters. The outputs are the ratio between valid transactions and total transactions. The ratio provides a direct and visualized relationship. The test set focus on the selfish strategy ac.

From the Result IV in Figure 39, we can find (1) the valid transactions maintains relatively stable under different $Ratio(\mathcal{B})$. (2) The ratio is stable whenever the $Ratio(\mathcal{F})$ increases or the total nodes increase. This means the ratio of valid with total transactions varies slightly with the malicious nodes and the number of malicious nodes has little influence. (3) The changes of $Ratio(\mathcal{B})$ and $Ratio(\mathcal{F})$ have slight influence on the ratio results. All above-mentioned results show that the selfish results only related to the selfish behavior instead of its combination or strategy. The DAG will maintain stable under the selfish behaviors.

Summarized results

The above-mentioned tests on tangle-based attacks provide us some enlightening points. (1) The DAG can maintain stable in case of selfish behaviors no matter how it made up or how many selfish nodes exist. (2) The attacks (DS, PS, HB) are sensitive to the $Ratio(\mathcal{B})$, which means how to make up a strategy is influential to the attack effect. (3) The successful attacks (ratio of Confirmed Invalid Transaction / Total Invalid Transaction) maintain stable under different $Ratio(\mathcal{F})$, which means the increasing malicious nodes will significantly increase the absolute number of transactions instead of probability. (4) Tangle-based structure is sensitive to the binary actions in Layer0, the actions are deterministic for the

final success.

6.6 Summary

Tangle, as one of the earliest DAG-based blockchain structure, offers references of chain building for future researchers. Various DAG-based blockchains without block structure are influenced by the concept of tangle. We abstract the principles of tangle-based blockchain from the basic actions to the meshed graph network, including removing the structure of blocks, verifying multiple previous transactions, and configuring tip selection algorithms. Then, according to the above features, we define three actions as the basic benchmarks to construct our attack strategies layer by layer. We provide three types of attack strategies, containing parasite attack, double spending attack, and the hybrid attack. Each attack strategy is made up by multiple behaviors, where the behaviors are made up by different actions. We further evaluate these attacks through metrics and provide 14 sets of testing results. The evaluations cover the influence of both the binary selection of actions and the changeable parameters of configuration. We present a comprehensive discussion on the attacks towards tangle-based blockchain based on our constructions and evaluations. The results show the trends under different strategies and configurations. Our reconstruction and revaluations on attacks provide a paradigm for both attack and defense of tangle-based blockchains. In the future, we will provide a general model for other blockchain/DAG structures.

Chapter 7

Use Case 3. Smart Grid Simulation

The incorporating of distributed energy systems and advanced network, e.g., advanced metering infrastructure, solar panels, and residential battery energy storage system make it possible for individuals generate, store and transfer energy among demand side[67][68]. Modern buildings have large flexibility to implement self-supply in a daily life with the prevalence of distributed renewable energy resources (e.g., rooftop solar panel and wind turbine). The neighboring energy trading enables energy transfer and transaction in neighborhood to achieve greater economic benefits[69]. In order to support individual energy trading market, the traditional approach is to develop a centralized trusted third party (TTP) to control collection and exchange of entire transmission data[70]. However, great amounts of work have indicated the weakness behind. For instance, the unified and centralized management of trading networks are similar to other centralized computing solutions, which leads to a series of privacy and security challenges[71]. The electricity consumption behaviors of users and trading price ranges are easily eavesdropped by attacking the control center[72]. Also, it is difficult to adapt large-scale utilization of renewable energy in the unified management[73]. In order to solve these issues, several decentralized energy management solutions attract more and more attention.

However, most of the issues have not been addressed yet. There still remains a gap between theory and application. The sheer volume of data and limited computing resources make it difficult to meet demand and work properly. Also, although the concept of decentralized system has been developed, the bid trade mechanism among local producers and consumers is still being ignored. In the neighborhood trade event, individuals would bid simultaneously, so the detailed regulation need to be provided in bid operation.

Based on above discussions, in this part we propose an efficient and secure distributed trading mechanism (SDT) that enables residential individuals to trade energy in a distributed manner. SDT is adopted in the smart contract to ensure confidentiality, authenticity, incontestability and integrity of shared information. We deploy the SDT in our simulation framework and analyze different scales of residential users from generation sides, to research whether those homes and businesses will become smarter and more self-sufficient, as we install solar, battery and proposed smart contract based trading system.

7.1 Related Work

In Li's study[74], proposed a new type of network innovation architecture of Software Defined Networking (SDN) based on blockchain computing. By separating all control commands and operations of energy internet, the system becomes more flexible. Blockchain helps Peer-to-Peer (P2P) interactions in the electricity market and reduces the threshold of local retailer participation[75]. Yorozu[76] study the conversion problems between virtual currency and real currency when traded in a blockchain. A credit-based payment scheme can be established to enable users manage and transfer energy coins.

7.2 Implementation Technology

7.2.1 Communication Protocols

In practice, there are three types of communication protocols can be applied to implement the developed scheduling algorithm: Wide Area Network (WAN), Neighbor- hood Area Network (NAN), and Home Area Network (HAN).

A Wide Area Network (WAN) is the network that cover large geographical areas and is used to connect the CEMS and the grid utility. For example, the smart contract receives the

forecasting peak load information from the utility using WAN. NAN provides support for communication between Smart Contract and home energy management system (HEMS). Several wireless standards, such as IEEE 802.11, IEEE 802.15 and IEEE 802.16, can be used in NAN. HAN is used in homes and buildings to facilitate communication between HEMS and enable controllable home energy resources. Several protocols, including WiFi, HomePlug Green PHY, Zigbee, Bluetooth and IEEE802.11n, can be used to support the HAN.

7.2.2 Metering Infrastructures

The proposed model could be implemented via an advanced metering infrastructure (AMI) system installed with smart meters for data processing, energy management and communication between the Smart Contract and HEMS. The Integration of smart meters enhances facilitation of decentralized generation and provide the capability of bidirectional communication. When trading occurs, smart meters of multiple HEMSs would predict individual energy consumptions according to numerous historical data profiles collected from smart meter, and then transfer the redundant/ required load to HEMS and calculate the expected price to Smart Contract. After receiving different prices from a number of residents, the Smart contact can be coordinately employed to control and monitor the paired trading of customers. We assume that smart meters analyze the real-time user behaviors, solar penetration and demand response once an hour. In the view of implementation of smart meters, Power Line Carrier (PLC) technology, Broadband Power Line, Bluetooth based energy meter, WIFI, RS0232/485, and WiMAX are highly efficient for automation of data in smart meter applications.

7.2.3 Smart Contract

Smart contract is a kind of protocol which aims to spread, verify or execute the contract independently. When Bitcoin was born, people found that blockchain could provide a trusted execution environment for smart contract. Ethereum first realized the integration of blockchain and smart contract. Smart contracts allow trusted transactions, which are

traceable and irreversible, without a third party. The purpose of smart contract is to provide a security method easy to program and realize, and also to reduce other transaction costs related to the contract. It packages logic, rules, processing steps and protocols between the two parties, and is called only when certain conditions are met.

7.3 Structure and Deployment in Simulator

The SDT trading prototype considered in this paper comprises multiple autonomous residential buildings which actively respond demand side management signals and utilize user-side energy. There is a regional smart contract which performs entire energy management in trading and transforming. The control architecture of the proposed system is shown in Figure 40. In the residential end, some smart buildings are equipped with a rooftop PV solar source and a battery energy storage system (BESS). The BESS is used to accommodate the solar power and provide power for home using. It may also store the extra solar panel, or get cheap power from the grid for future usage. Smart meter is installed in every home, taking the role of a communication agent between the smart contract and the home.

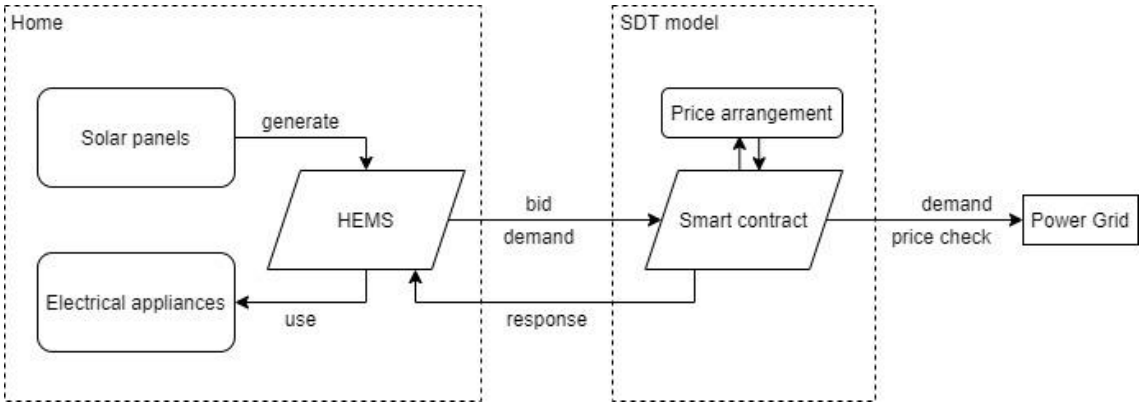


Figure 40. System architecture

A home energy management system (HEMS) is deployed in the smart home, managing home energy resources, including the solar penetration data, some electrical appliances and BESS. It communicates with the smart meter, home energy resources, and the smart

contract, and also interacts with the user. When the trading happens, the HEMS collects the information about forecasted personal power usage, the real-time pricing data from grid, and real-time personal solar generation data.

When residential trading happens, the HEMS obtain the biding information from the regional smart contract. Simultaneously, it forecasts load consumption profile in the next stage duration and submits the forecasting excrescent/missing load profiles to SDT model for selling/buying. Based on this, the smart contract sets up an optimal energy management model to determine the proper selling/buying strategy, optimizing the amount of individual value and shifting system peak demand.

In simulation level, our model works in the flow shown in Figure 41.

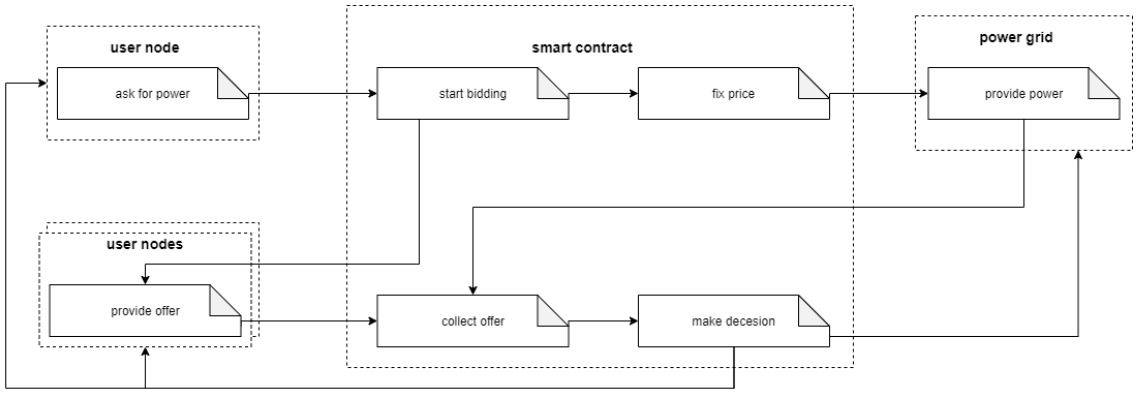


Figure 41. Energy trading workflow

Following the system schematic presented in 7.2, in this section[77] we firstly illustrate the generic SDT trading models; then, we formulate the proposed HEMS model with different kinds of operational dependencies due to living habits, which are considered as an additional constraint of the SDT based real-time trading model.

Generic classification of resident users

Consider residential units with N controllable users and denote the set of users as Ω , i.e. $|\Omega| = N$. The controllable users are further categorized into following sets based on their operational characteristics:

Ω^1: Set of users operating with power in the range of $\left[P_a^{min}, P_a^{max}\right], a \in \Omega^1$, but without the energy generation ability. Besides that, a disutility function is applied to measure the dissatisfaction of the user on deviating from a nominal operating point. Typical users in this set consume but not generate electricity.

Ω^2: Set of users generating power in the range of $\left[P_a^{min}, P_a^{max}\right], a \in \Omega^2$ and having a prescribed energy consumption that must be completed in a specific time range. Typical users in this class equip battery energy storage system (BESS) and generate electricity that can supply itself and other residents.

Residential Photovoltaic Solar Power Model

Power output from PV solar panel is related to solar radiation, panel's surface area, and energy conversion efficiency of the panel, expressed as:

$$P_t^{pv} = A \cdot \sigma \cdot r_t \tag{4}$$

Where P_t^{pv} denotes solar power output at time t (kW); $A$is the surface area of the PV solar panel (m^2) and σdefines the energy conversion efficiency of the PV solar panel (%); $r_t$is the solar radiation at time t (kW/m2).

SDT based Energy Trading Model

Denote the energy consumption schedule for each controllable user as:

$$\boldsymbol{P}_k = [P_{k,1}, P_{k,2}, \ldots, P_{k,T}] \; \forall k \in \Omega \tag{5}$$

The power consumption schedule of Ncontrollable appliances can then be represented as a matrix with $N \times T$ dimensions, where the entry $P_{a,t}$ represents the power consumption of appliance a at time interval t. The net-power consumption of the home can be correspondingly represented as:

$$\tilde{\boldsymbol{P}}^{k} = [\tilde{P}_1^{k}, \ldots, \tilde{P}_t^{k}, \ldots, \tilde{P}_T^{k}] \tag{6}$$

$$\tilde{P}_t^{k} = P_t^{mr} + \sum_{i \in \Omega} P_{k,t} - P_t^{pv} \; t = 1{:}\,T \tag{7}$$

where P_t^{mr} *is the must-run home load at time t (W).*

Objective:

The objective of proposed SDT trading model in this study is to realize the energy self-arrangement of the demand side. All bid strategies will be decided and recorded in the smart contract and the selling/buying prices will be arranged from high to low.

$$p^{S} = \lambda \sum_{t=1}^{T} \left(\tilde{P}_t^{k}\right) + \mu \cdot \sqrt{\sum_{t=1}^{T} (p^{S1} + p^{S2} + p^{S3} + \ldots p^{Sn-1})} \tag{8}$$

Where p^{S} *denotes the selling price determined by the smart contract.*

There are two components in model (8). The first is the redundant energy generated by the users, Ω^2 ,at time t; the second is the average biding price. λ and μ are different weighting coefficients.

$$p^{B} = \beta \sum_{t=1}^{T} \left(P_t^{k}\right) - \nu \cdot \sqrt{\sum_{t=1}^{T} (p^{B1} + p^{B2} + p^{B3} + \ldots p^{Bn-1})} \tag{9}$$

Where p^{B} *denotes the buying price determined by the smart contract.*

The first component is the user requirement energy at time t; the second is the average buying price. β and ν are different weighting coefficients.

Mandatory constraints:

The SDT energy trading model is subjected to following constraints:

1) Operational constraints of users in Ω^2, i.e. an BESS in this study. The operation of the BESS must satisfy following constraints:

$$E_{t+1}^{ess} = \begin{cases} E_t^{ess} + \Delta t \eta^c P_t^{ess} - E_t^{ess} \eta^l \Delta t \; P_t^{ess} > 0 \\ E_t^{ess} - |P_t^{ess}| \eta^d \Delta t - E_t^{ess} \eta^l \Delta t \; P_t^{ess} \leq 0 \end{cases}, \; t = 1{:}T-1 \tag{10}$$

$$SOC_t^{ess} = \frac{E_t^{ess}}{E^{ess,rate}} \; t = 1{:}T \tag{11}$$

$$|P_t^{ess}| \leq P^{ess,rate} \; t = 1{:}T \tag{12}$$

$$SOC^{min} \leq SOC_t^{ess} < SOC^{max} \; t = 1{:}T \tag{13}$$

$$SOC_t^{ess} \geq SOC^{dsr} \; t = T \tag{14}$$

Eqs. (10) and (11) model the variation of energy stored in the BESS; constraint (12) specifies the BESS's maximum charging/discharging power; constraint (13) ensures that the BESS's SOC is maintained within an allowable range; constraint (14) ensures that the SOC level of the BESS is larger or equal to a pre-specified threshold at the end of the day.

2) Bidding ranking constraints of users in Ω^2in this study. Smart contract arranges the selling/buying price as follows:

$$p_t^{S1} > p_t^{S2} > p_t^{S3} > \ldots > p_t^{Sn}, t = 1,2 \ldots T \tag{15}$$

$$p_t^{B1} < p_t^{B2} < p_t^{B3} < \ldots < p_t^{Bn}, t = 1,2 \ldots T \tag{16}$$

The smart contract will arrange selling price from low to high and buying price from high to low, and pair each of them following this sequence. For example, smart contract will pair $p_t^{S1}$and p_t^{B1} first and pair others as so on.

User node sends a request to smart contract to ask for power. When receiving the request, the smart contract starts a bidding and asks for offer. Other user nodes who have capacity to provide power will reply an offer to the contract. The smart contract will also ask the power grid for its power price. Then the smart contract will collect these offers and make decision.

7.4 Results and Discussion

There are several parameters we can change in simulation:

Table 11. Key parameters in power grid simulation

Parameters	Description	Default
SIM_TIME	The amount of time that simulation runs	24h
Ω	The number of users	60
Ω^2/Ω	The percentage of users who have the ability to generate power	20%
P^{pv}	Power output from PV solar panel	5000W
P_STORAGE	The power storage ability of each user.	5000Wh
ST	The settlement interval between two transactions	[45,75]

Before simulation, we get some data of power usage from government publicity. Figure 42 shows the average hourly power usage of 15 different users in 11/2012, NSW. We set these 15 users' data as 15 types of user power usage input.

After investigation, the most popular home use power generator is 5kW. So we set a default power output as 5kW. Figure 43 shows the average hourly power generation of 3 different users with a 5kW power generator. in 11/2012, NSW. We set these 3 user data as 3 types of user power generation input. To fit the generation capacity of the users and meet the need of settlement, we give each user a 5000Wh power storage ability to store the power they have generated. The users are disabled to store the power they get from the power grid, but can be activated in the future research.

Since the number of users will be set to more than 15, each user will randomly select one of these 15 types as its input. It is similar to the input selection of user power generation. To differentiate the same type users, we set a random settlement interval (ST) instead of an hourly interval. The settled power is calculated as:

$$P_{k,T}^{s} = P_{k,T} * \frac{ST}{60}, P_{k,T}^{s,pv} = P_{k,T}^{pv} * \frac{ST}{60} \quad (17)$$

This settlement interval will also help us to reduce the network congestion.

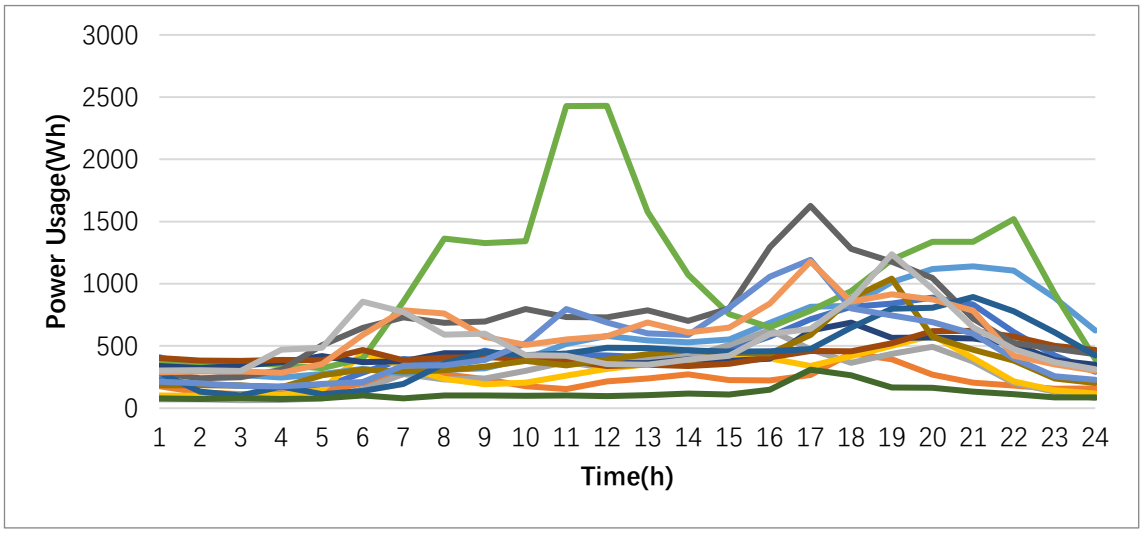


Figure 42. Power usage types

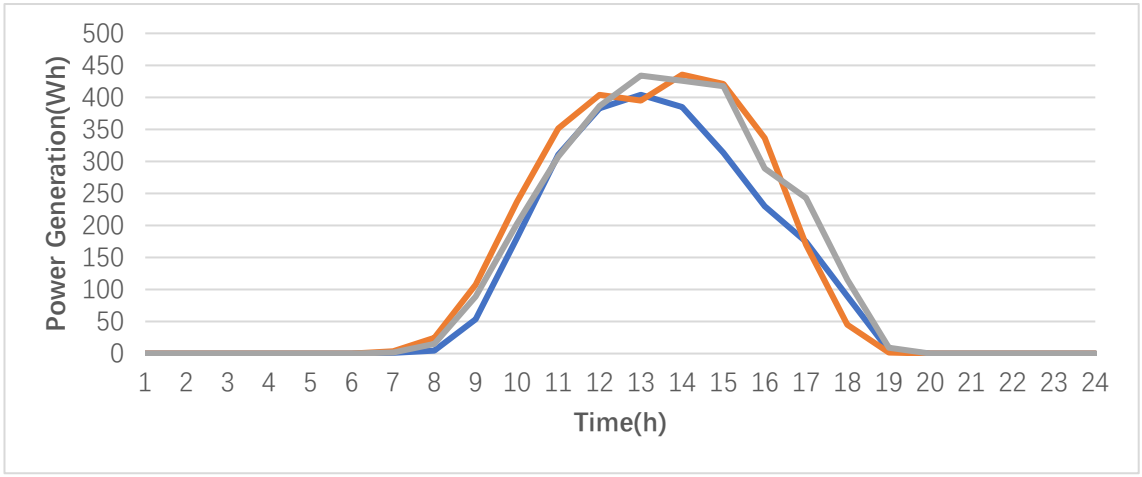


Figure 43. Power generation types

The price of power in November from Energy AU is shown as below.

Table 12. Australia power price hourly

Time(h)	0~7	7~14	14~20	20~22	22~24
Price(AUcent/kwh)	17.7	27.71	54.11	27.71	17.7

The solar feed-in tariff is about 10 cents/kWh

With the definition shown above, we get the following results:

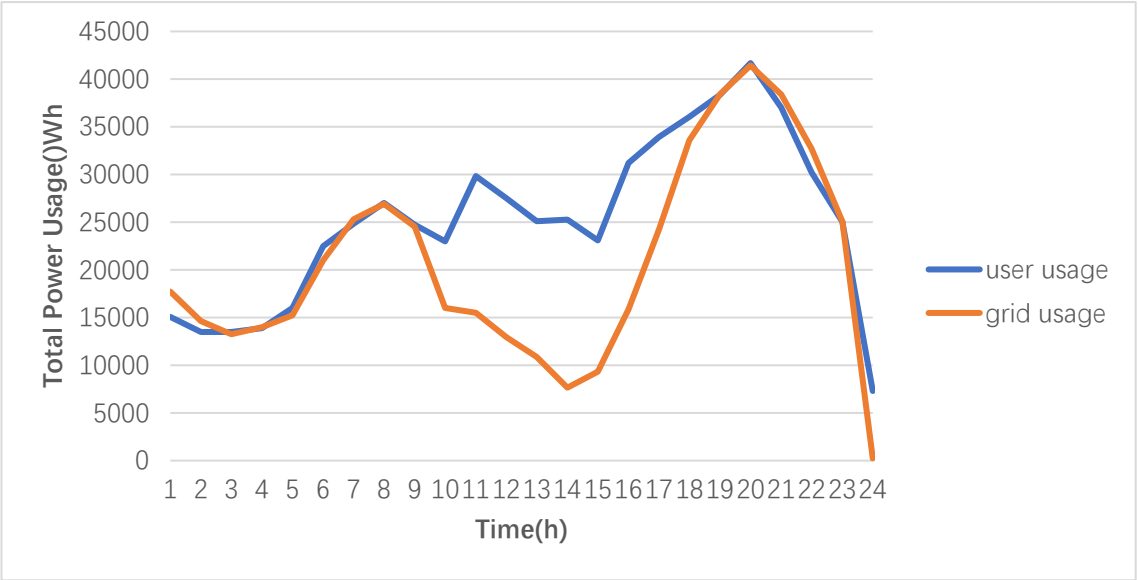


Figure 44. Total power usage with default settings

Figure 44 shows the power usage of the system under default settings. The user usage line means the quantity of power that users need from the system, which has already subtracted the power generated and used by themselves. The grid usage line means the quantity of power the grid has provided to the users. The gap between the two lines is the power transacted through the bidding system.

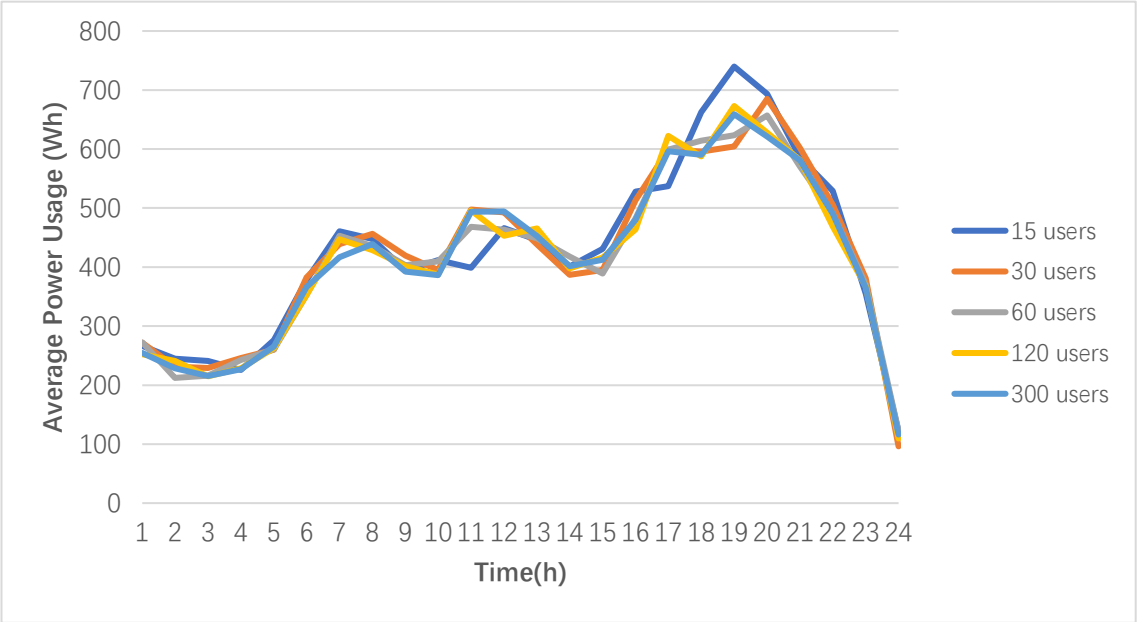


Figure 45. User power usage with different user number

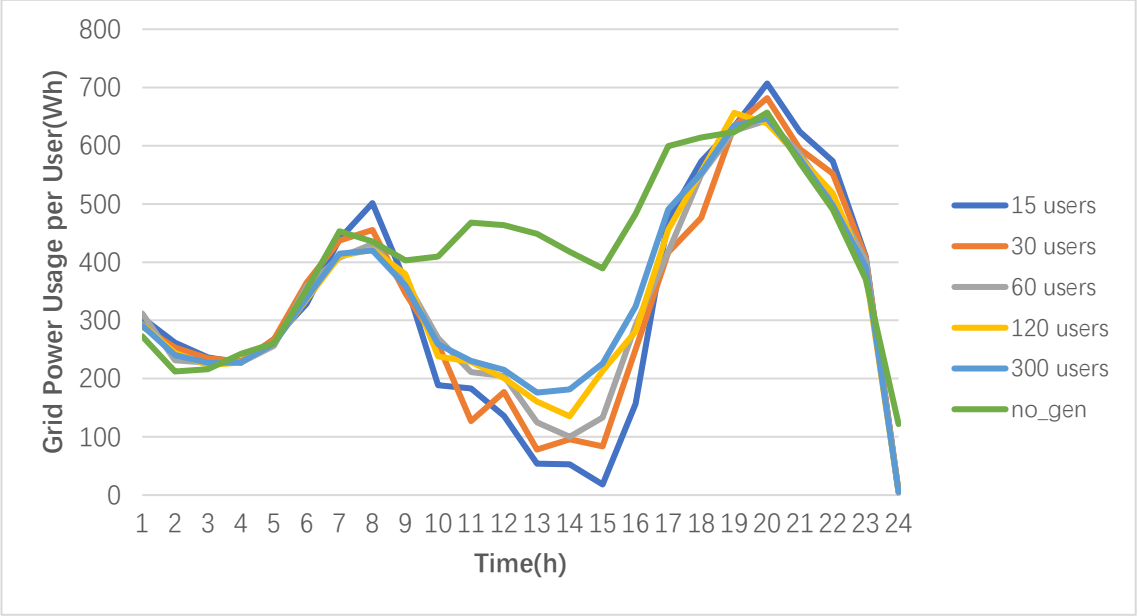


Figure 46. Grid power usage with different user number

As shown in Figure 45, when the number of user increases, the average power that each user needs from the system keeps stable. On the other hand, in Figure 46, the average power that grid provides to each user decreases at the middle of the day. It means that the more users, the transactions are more likely to be completed between users, and our system will be more activated. Our system illustrates great scalability.

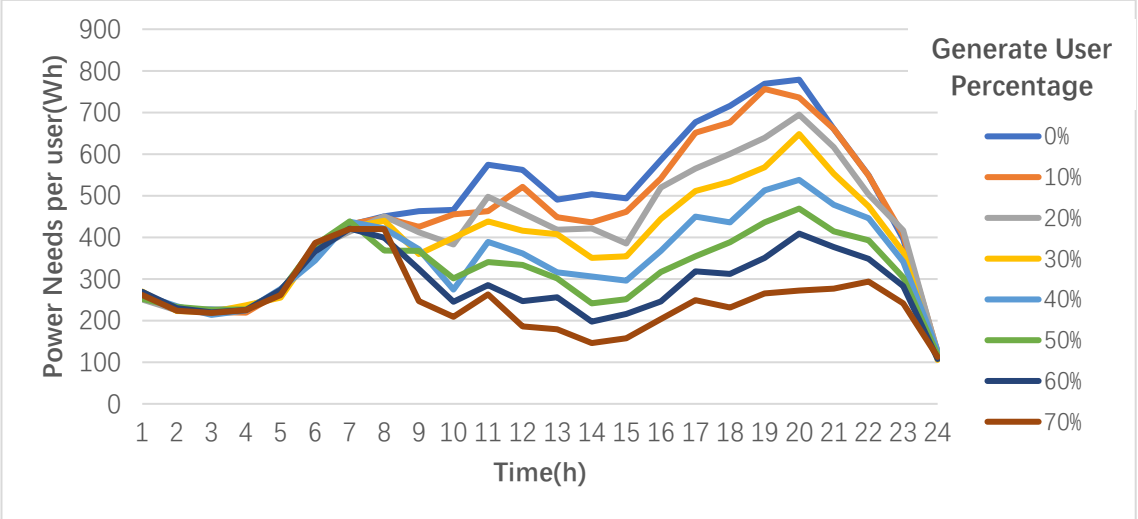


Figure 47. User power needs with different generate user percentage

Figure 47 shows that when the percentage of users who have power generation capacity increases, the power each user need from the system will decrease since 9am.

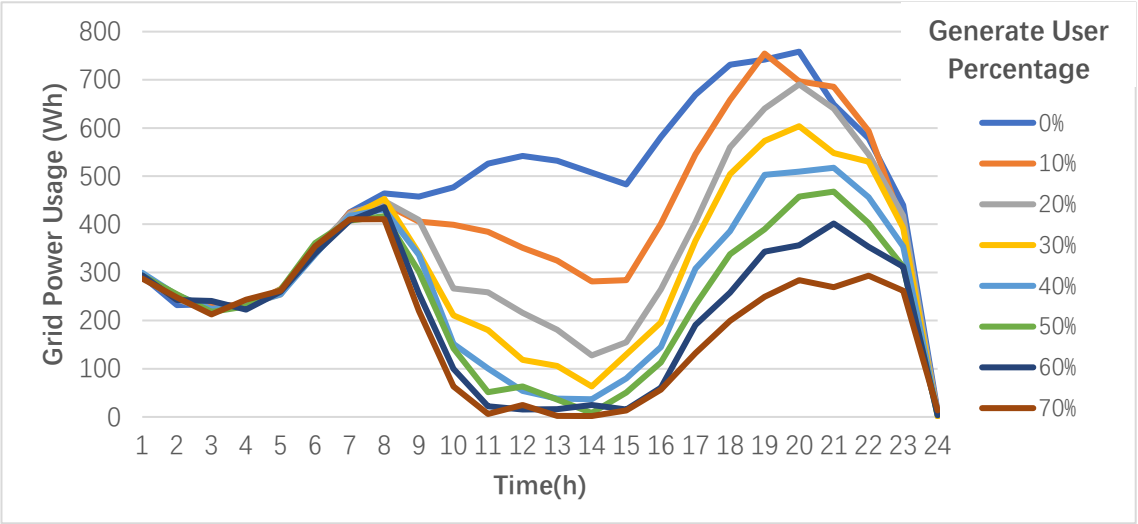


Figure 48. Grid power usage with different generate user percentage

As shown in Figure 48, as the percentage rises, the power grid provides decreases in a high level, which means that we can utilize this to help the grid to conduct peak load shifting at noon. But we can also find out that when the percentage is more than 50%, the generation capacity exceeds electricity consumption, which means that some power is wasted. This may be solved by giving the users a higher power storage capacity.

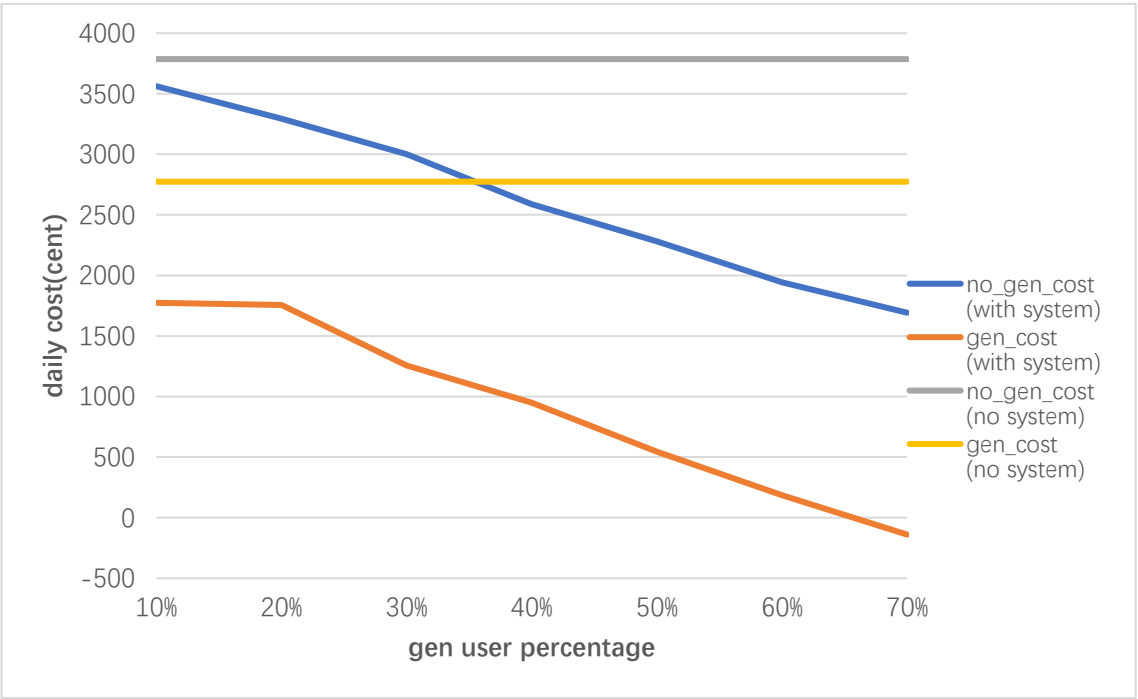


Figure 49. Average cost with different settings

In Figure 49, we can find out that, as the percentage rises, the daily power cost has been decreasing, no matter whether the user have power generation capacity or not. Ω2 can

even earn money when the percentage is up to 70%. When percentage is over 30%, Ω1 in our system can spent less money than Ω2 in traditional power system. It shows that our system can help users to reduce their spent in power usage.

7.5 Summary

In this section, we illustrate an efficient and secure distributed trading mechanism. We introduce the model and then deploy it in the simulation framework to test its feasibility. The result shows that our SDT is stable and extendable. It also can help to shift the load and the peak of power usage and reduce users' cost. As traditional blockchain calculation needs a lot of power, in the future, we want to find out a way to do the verification work using power grid bidding system instead of smart contract.

Chapter 8

Conclusion

In this dissertation, we mainly build a blockchain simulation model based on P2P network. Firstly, we summarize, compare and discuss the existing blockchain technologies. Secondly, we deploy three main blockchains in the simulator and verified them. Then, we improve the performance of the blockchain in two different ways. Finally, we implement two application scenarios of blockchain in the simulator.

8.1 Summary

Blockchain simulation model deployment

In the past few years, blockchains have been one of the most attractive emerging technologies. Many researchers and institutions have devoted their resources to the development of more effective blockchain technologies and innovative applications. However, with the limitation of computing power and financial resources, it is hard for researchers to deploy and test their blockchain innovations in a large-scape physical network. In this paper, we design a peer-to-peer blockchain simulation framework to address this challenge. Our framework provides a foundation and skeleton to simulate as large as thousand-nodes P2P blockchain network with a single computer. This dissertation presents the basic structure and simulation mechanism of us proposed structure, and showcase its capabilities and usefulness. With our simulation framework, researchers can test their new consensus protocols, reproduce a subtle security attack and evaluate its risks

with a large number of nodes under heavily transaction loads.

Blockchain performance improvement

Parameters change In this dissertation, we focus on selection and definition of key performance metrics to quantify Quality of Blockchain (QoB). We aim to demonstrate the proposed idea by using an existing simulation tool to duplicate a simplified Blockchain Proof of Work (PoW) protocol with different parameters and observations. Our case study shows that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes.

New DAG blockchain structure This dissertation explores an approach to improving Blockchain performance using Directed Acyclic Graph (DAG). We propose a new DAG-based Blockchain model. Then we use the simulation system to evaluate the performance of the DAG-based Blockchain. Our simulation results show that DAG is able to improve the Blockchain performance to some degree, though there are still physical bottlenecks (network bandwidth and CPU processing time) that needs to overcome by adding more resources and improving parallel processing capability.

IOTA attack

Tangle-based blockchains become one of the most promising structures in current DAGs. It improves the scalability by directly verifying the previous transactions in parallel instead of the blocks. However, this performance gain may bring potential security risks if not being designed well. In this dissertation, we construct three types of attacks and corresponding evaluations, including parasite attack (PS), double spending attack (DS) and hybrid attack (HB), to test the security of tangle-based blockchains. To achieve that, we deconstruct the tangle-based projects (e.g. IOTA) into fundamental components to explore the underlying principles. And then, we define three basic actions as the bottom benchmarks to build up the attack strategies layer by layer. Based on that, we provide comprehensive analysis to

evaluate the different attacks in multiple dimensions. To the best of our knowledge, this is the first study to provide a comprehensive security analysis towards tangle-based blockchains.

Smart grid simulation

Intelligent smart metering facilities and two-way communication infrastructures enable residential buildings to actively generate and trade energy in demand side. However, traditional centralized power management mode causes information exchange suffering from unreliability and lack of privacy. In this dissertation, we propose a privacy-preserving, efficient and robust distributed energy exchange scheme supported by the smart contract. An efficient bidding model is first proposed for demand side to achieving peer-to peer trading of electricity in the real-time market. The entire exchanges of data are implemented on the smart contract to guarantee information safety and traceability with decentralized scheme. The proposed solution achieves reliable and stable trading objects under the premise of information security. Finally, we conduct a systematic and comprehensive applicability analysis of the proposed mechanism, and further confirm that the system can be practically used in home energy management system.

8.2 Future Work

We have mentioned some future directions in previous chapters. Here we discuss possible future work more detailly to extend the current research.

Blockchain simulation model

In this paper, we introduce a blockchain simulator based on P2P network. The basic simulation model is now open source on GitHub. However, there are still some problems in the existing simulators. The network module of the existing simulator is divided into two

parts - message distribution module and message receiving module. The message distribution module is parallel to the miner / user node - in other words, it can be considered as a single node that completes the message distribution. The message receiving module exists inside each node, and is at the same level as the consensus module of the node. In our research, the message receiving module first receives all the messages in the network, and then determines whether the message can enter the network of current node. At low network load, this method has no impact. However, when the number of messages in the network is too large, the simulation time will increase exponentially. In the future research, we can further optimize the existing simulation model, in order to send the message to the corresponding node directly through the message distribution module. With this improvement, we can make the simulator closer to the real network situation.

Blockchian performance improvement

In Chapter 5.1, we modify the block size, block time and other basic parameters to improve the network performance of bitcoin. Since bitcoin is not modifiable in reality, the improvement of blockchain performance is more used to establish Consortium Blockchain or Private Blockchain. We can further modify the basic parameters of the blockchain according to the requirements of node number, network load and safety, so as to get the most efficient specific blockchain.

In Chapter 5.2, we propose a DAG-based blockchain. It requires less computing power for general user nodes, and is more suitable for consortium/private blockchain systems with multiple known centres. In our blockchain structure, we can more clearly trace the flow direction of each transaction with high security. In the future research, we will do more research on the election mechanism of collectors to ensure that the blockchain will not be controlled by a few nodes.

In the future research, there are two kinds of blockchain application scenarios. One is that the system has a lot of computing power to ensure network security. We can build the blockchain by modifying the existing blockchain settings. In another scenario, the system

has a large number of users and requires low transaction cost and real-time performance. We can build the blockchain depends on our DAG consensus.

In Chapter 6, we set up the attack test for iota. This set of tests can also be applied to the new built consortium/private blockchain to ensure that the blockchain has resistance to different network attacks.

Based on our simulation model, in addition to the research on establishing blockchain in the future, we will also help existing miners and users to study better mining strategies. Since we can simulate the behaviour of different miners and users, the following questions can be explored through large amounts of calculations: how transaction costs set by users affect confirmation time of transaction; how transaction costs included in the transaction affect the overall network when miners pack blocks; and how to make a balance between getting more fees and reducing transaction commit time for low fee users.

Smart grid simulation

In Chapter 7, we build a smart contract based on power grid price adjustment system. As we all know, a lot of hash operations are required in the process of blockchain mining. In order to obtain more computing power, miners need to consume a lot of electric energy. In future research, we hope to change the way blockchain reaches consensus. There can be local division in the power grid itself, and the blockchain will reach consensus in the region first, and then send the results of local consensus to the whole network for verification. In this way, we can make use of the surplus power in the power grid for mining, and realize peak load shifting by transferring the consensus area.

Bibliography

[1] Nakamoto, Satoshi, "Bitcoin: A peer-to-peer electronic cash system." *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* 4 (2008).

[2] Lin, Iuon-Chang, and Tzu-Chun Liao. "A survey of blockchain security issues and challenges." *IJ Network Security* 19, no. 5 (2017): 653-659.

[3] Forouzan, Behrouz A. "TCP/IP protocol suite". (2002).

[4] Scott, Brett. "How can cryptocurrency and blockchain technology play a role in building social and solidarity finance?". *No. 2016-1. UNRISD Working Paper* (2016).

[5] Shafagh, Hossein, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. "Towards blockchain-based auditable storage and sharing of IoT data." In *Proceedings of the 2017 on Cloud Computing Security Workshop* (2017):45-50.

[6] Saberi, Sara, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. "Blockchain technology and its relationships to sustainable supply chain management." *International Journal of Production Research* 57, no. 7 (2019): 2117-2135.

[7] Pilkington, Marc. "Blockchain technology: principles and applications." In *Research handbook on digital transformations*. (2016).

[8] Pervez, Huma, Muhammad Muneeb, Muhammad Usama Irfan, and Irfan Ul Haq. "A comparative analysis of DAG-based blockchain architectures." In *2018 12th International Conference on Open Source Systems and Technologies* (2018):27-34.

[9] Conti, Mauro, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. "A survey on security and privacy issues of bitcoin." *IEEE Communications Surveys & Tutorials* 20, no. 4 (2018): 3416-3452.

[10] Antonopoulos, Andreas M., and Gavin Wood. "Mastering ethereum: building smart

contracts and DApps." (2018).

[11] Li, Wenting, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. "Securing proof-of-stake blockchain protocols." In *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (2017):297-315.

[12] Handley, Simon. "On the use of a directed acyclic graph to represent a population of computer programs." In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence* (1994):154-159.

[13] Lamport, Leslie, Robert Shostak, and Marshall Pease. "The Byzantine generals problem." In *Concurrency: the Works of Leslie Lamport* (2019):203-226.

[14] Lamport, Leslie. "The weak Byzantine generals problem." *Journal of the ACM (JACM)* 30, no. 3 (1983): 668-676.

[15] Foster, Ian, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. "Grid services for distributed system integration." *Computer* 35, no. 6 (2002): 37-46.

[16] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." *Journal of the ACM (JACM)* 32, no. 2 (1985): 374-382.

[17] Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery." *ACM Transactions on Computer Systems (TOCS)* 20, no. 4 (2002): 398-461.

[18] "Tether: Fiat currencies on the Bitcoin blockchain". *https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf*

[19] Reed, Jeff. "Litecoin: An introduction to litecoin cryptocurrency and litecoin mining." (2017).

[20] Lai, Roy, and David LEE Kuo Chuen. "Blockchain–From public to private." In *Handbook of Blockchain, Digital Finance, and Inclusion, Volume 2* (2018):145-177.

[21] Li, Zhetao, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang.

"Consortium blockchain for secure energy trading in industrial internet of things." *IEEE transactions on industrial informatics* 14, no. 8 (2017): 3690-3700.

[22] Pongnumkul, Suporn, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. "Performance analysis of private blockchain platforms in varying workloads." In *2017 26th International Conference on Computer Communication and Networks* (2017):1-6.

[23] Popov, Serguei, Olivia Saa, and Paulo Finardi. "Equilibria in the Tangle." Computers & Industrial Engineering 136 (2019): 160-172.

[24] Popov, Serguei. "The tangle." White paper 1 (2018): 3.

[25] Lombrozo, Eric, Johnson Lau, and Pieter Wuille. "Segregated witness (consensus layer)." *Bitcoin Core Develop. Team, Tech. Rep. BIP* 141 (2015).

[26] Yasaweerasinghelage, Rajitha, Mark Staples, and Ingo Weber. "Predicting latency of blockchain-based systems using architectural modelling and simulation." In *2017 IEEE International Conference on Software Architecture* (2017):253-256.

[27] Jakobsson, Markus, and Ari Juels. "Proofs of work and bread pudding protocols." In *Secure information networks* (1999):258-272.

[28] Weber, Ingo, Xiwei Xu, Régis Riveret, Guido Governatori, Alexander Ponomarev, and Jan Mendling. "Untrusted business process monitoring and execution using blockchain." In *International Conference on Business Process Management* (2016):329-347.

[29] Göbel, Johannes, Holger Paul Keeler, Anthony E. Krzesinski, and Peter G. Taylor. "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay." *Performance Evaluation* 104 (2016): 23-41.

[30] Page, Bernd, and Wolfgang Kreutzer. "Simulating discrete event systems with UML and JAVA." (2006): 441-441.

[31] Alharby, Maher, and Aad van Moorsel. "Blocksim: a simulation framework for

blockchain systems." *ACM SIGMETRICS Performance Evaluation Review* 46, no. 3 (2019): 135-138.

[32] Stoykov, Lyubomir, Kaiwen Zhang, and Hans-Arno Jacobsen. "Vibes: fast blockchain simulations for large-scale peer-to-peer networks." In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos* (2017):19-20.

[33] Aoki, Yusuke, Kai Otsuki, Takeshi Kaneko, Ryohei Banno, and Kazuyuki Shudo. "Simblock: A blockchain network simulator." In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops* (2019):325-329.

[34] https://www.blockchain.com/charts/avg-confirmation-time

[35] https://digiconomist.net/bitcoin-energy-consumption

[36] King, Sunny, and Scott Nadal. "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake." *self-published paper, August* 19 (2012): 1.

[37] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." *Software: Practice and experience* 41, no. 1 (2011): 23-50.

[38] Decker, Christian, and Roger Wattenhofer. "Information propagation in the bitcoin network." In *IEEE P2P 2013 Proceedings* (2013):1-10.

[39] Gilad, Yossi, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. "Algorand: Scaling byzantine agreements for cryptocurrencies." In *Proceedings of the 26th Symposium on Operating Systems Principles* (2017):51-68.

[40] Goswami, Sneha. "Scalability analysis of blockchains through blockchain simulation." (2017).

[41] Li, Chenxing, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. "Scaling nakamoto consensus to thousands of transactions per second." *arXiv preprint arXiv:1805.03870* (2018).

[42] Abd-El-Malek, Michael, Gregory R. Ganger, Garth R. Goodson, Michael K. Reiter, and Jay J. Wylie. "Fault-scalable Byzantine fault-tolerant services." *ACM SIGOPS Operating Systems Review* 39, no. 5 (2005): 59-74.

[43] Bentov, Iddo, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y." *ACM SIGMETRICS Performance Evaluation Review* 42, no. 3 (2014): 34-37.

[44] Gervais, Arthur, Hubert Ritzdorf, Ghassan O. Karame, and Srdjan Capkun. "Tampering with the delivery of blocks and transactions in bitcoin." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015):692-705.

[45] Kogias, Eleftherios Kokoris, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. "Enhancing bitcoin security and performance with strong consistency via collective signing." In *25th security symposium* (2016):279-296.

[46] Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery." ACM Transactions on Computer Systems (TOCS) 20, no. 4 (2002): 398-461.

[47] Gervais, Arthur, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. " On the security and performance of proof of work blockchains " In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (2016):3-16.

[48] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." *Ethereum project yellow paper* 151, no. 2014 (2014): 1-32.

[49] Acharya, Anasuya, Manoj Prabhakaran, and Akash Trehan. "CellTree: A New Paradigm for Distributed Data Repositories." *IACR Cryptol. ePrint Arch.* 2019 (2019): 516.

[50] "Dag," *https://en.wikipedia.org/wiki/Directed acyclic graph* (2019).

[51] Sompolinsky, Yonatan, and Aviv Zohar. "Secure high-rate transaction processing in bitcoin." In *International Conference on Financial Cryptography and Data Security*

(2015):507-527.

[52] Churyumov, Anton. "Byteball: A decentralized system for storage and transfer of value." *https://byteball. org/Byteball. pdf* (2016).

[53] Sompolinsky, YL Yonatan, and A. Zohar. "Serialization of proof-of-work events: Confirming transactions via recursive elections." *IACR Cryptology ePrint Archive.*

[54] Nguyen, Quan, Andre Cronje, Michael Kong, Alex Kampa, and George Samman. "Stairdag: Cross-dag validation for scalable bft consensus." *arXiv preprint arXiv:1908.11810* (2019).

[55] Nguyen, Quan, and Andre Cronje. "ONLAY: Online Layering for scalable asynchronous BFT system." *arXiv preprint arXiv:1905.04867* (2019).

[56] Kusmierz, B. "The first glance at the simulation of the Tangle: discrete model." *IOTA Found. WhitePaper* (2017): 1-10.

[57] Kusmierz, Bartosz, Philip Staupe, and Alon Gal. "Extracting tangle properties in continuous time via large-scale simulations." *Working Paper* (2018).

[58] Nagaitsev, S., A. Valishev, V. V. Danilov, and D. N. Shatilov. "Design and Simulation of IOTA-a Novel Concept of Integrable Optics Test Accelerator." *arXiv preprint arXiv:1301.7032* (2013).

[59] Ferraro, Pietro, Christopher King, and Robert Shorten. "IOTA-based directed acyclic graphs without orphans." *arXiv preprint arXiv:1901.07302* (2018).

[60] Sompolinsky, Yonatan, and Aviv Zohar. "Accelerating bitcoin's transaction processing." *Fast money grows on trees, not chains* (2013).

[61] Lewenberg, Yoad, Yonatan Sompolinsky, and Aviv Zohar. "Inclusive block chain protocols." In *International Conference on Financial Cryptography and Data Security* (2015):528-547.

[62] Sompolinsky, Yonatan, Yoad Lewenberg, and Aviv Zohar. "SPECTRE: Serialization of

proof-of-work events: confirming transactions via recursive elections." *Cryptology ePrint Archive, IACR* 1159 (2016).

[63] Sompolinsky, Yonatan, Yoad Lewenberg, and Aviv Zohar. "SPECTRE: A Fast and Scalable Cryptocurrency Protocol." *IACR Cryptol. ePrint Arch.* 2016 (2016): 1159.

[64] Baird, Leemon. "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance." *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep* (2016).

[65] LeMahieu, Colin. "Nano: A feeless distributed cryptocurrency network." *https://nano.org/en/whitepaper* (2018).

[66] E. IO, "Eos. io technical white paper," *https://github. com/EOSIO/Documentation* (2017).

[67] Gai, Keke, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen. "Privacy-preserving energy trading using consortium blockchain in smart grid." *IEEE Transactions on Industrial Informatics* 15, no. 6 (2019): 3548-3558.

[68] Zhang, Yan, Rong Yu, Shengli Xie, Wenqing Yao, Yang Xiao, and Mohsen Guizani. "Home M2M networks: Architectures, standards, and QoS improvement." *IEEE Communications Magazine* 49, no. 4 (2011): 44-52.

[69] Logenthiran, Thillainathan, Dipti Srinivasan, and Tan Zong Shun. "Demand side management in smart grid using heuristic optimization." *IEEE transactions on smart grid* 3, no. 3 (2012): 1244-1252.

[70] Yang, Zijian, and Lin Wang. "Demand Response Management for multiple utility companies and multi-type users in smart grid." In *2016 35th Chinese control conference* (2016):10051-10055.

[71] Aitzhan, Nurzhan Zhumabekuly, and Davor Svetinovic. "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams." *IEEE Transactions on Dependable and Secure Computing* 15, no. 5 (2016): 840-852.

[72] Niyato, Dusit, Lu Xiao, and Ping Wang. "Machine-to-machine communications for home energy management system in smart grid." *IEEE Communications Magazine* 49, no. 4 (2011): 53-59.

[73] Lu, Xin, Lingyun Shi, Zhenyu Chen, Xunfeng Fan, Zhitao Guan, Xiaojiang Du, and Mohsen Guizani. "Blockchain-based distributed energy trading in energy Internet: An SDN approach." *IEEE Access* 7 (2019): 173817-173826.

[74] Li, Yinan, Wentao Yang, Ping He, Chang Chen, and Xiaonan Wang. "Design and management of a distributed hybrid energy system through smart contract and blockchain." *Applied Energy* 248 (2019): 390-405.

[75] Jacobs, I. S. "Fine particles, thin films and exchange anisotropy." *Magnetism* (1963): 271-350.

[76] Yorozu, T., M. Hirano, K. Oka, and Y. Tagawa. "Electron spectroscopy studies on magneto-optical media and plastic substrate interface." *IEEE translation journal on magnetics in Japan* 2, no. 8 (1987): 740-741.

[77] Luo, Fengji, Kong, Weicong, Ranzi, Gianluca, and Dong, Zhao Yang. 'Optimal Home Energy Management System With Demand Charge Tariff and Appliance Operational Dependencies'. IEEE Transactions on Smart Grid 11, no. 1 (2020): 4–14.

[78] Koay, B. S., S. S. Cheah, Y. H. Sng, P. H. J. Chong, P. Shum, Y. C. Tong, X. Y. Wang, Y. X. Zuo, and H. W. Kuek. "Design and implementation of Bluetooth energy meter." In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, vol. 3 (2003):1474-1477.

[79] Son, Young-Sung, Topi Pulkkinen, Kyeong-Deok Moon, and Chaekyu Kim. "Home energy management system based on power line communication." *IEEE Transactions on Consumer Electronics* 56, no. 3 (2010): 1380-1386.

[80] Lee, P. K., and L. L. Lai. "Smart metering in micro-grid applications." In *2009 IEEE*

Power & Energy Society General Meeting (2009):1-5.

[81] Tasdighi, Mohammad, Hassan Ghasemi, and Ashkan Rahimi-Kian. "Residential microgrid scheduling based on smart meters data and temperature dependent thermal load modeling." *IEEE Transactions on Smart Grid* 5, no. 1 (2013): 349-357.

[82] Young, Matt. "The technical writer's handbook: writing with style and clarity". *University Science Books* (2002).

[83] Weingartner, Elias, Hendrik Vom Lehn, and Klaus Wehrle. "A performance comparison of recent network simulators." In *2009 IEEE International Conference on Communications* (2009):1-5.

[84] Pazmiño, Juan Eduardo, and C. K. S. Rodrigues. "Simply dividing a Bitcoin network node may reduce transaction verification time." *The SIJ Transactions on Computer Networks & Communication Engineering (CNCE)* 3, no. 2 (2015): 17-21.

[85] Nicole, R. "Title of paper with only first word capitalized, J." *Name Stand. Abbrev* (1987).

[86] Memon, Raheel Ahmed, Jian Ping Li, and Junaid Ahmed. "Simulation model for blockchain systems using queuing theory." *Electronics* 8, no. 2 (2019): 234.

[87] Kosba, Ahmed, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts." In *2016 IEEE symposium on security and privacy* (2016):839-858.

[88] Göbel, Johannes, and Anthony E. Krzesinski. "Increased block size and Bitcoin blockchain dynamics." In *2017 27th International Telecommunication Networks and Applications Conference* (2017):1-6.

[89] Fairley, Peter. "Blockchain world-Feeding the blockchain beast if bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous." *IEEE Spectrum* 54, no. 10 (2017): 36-59.

[90] Eyal, Ittay, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. "Bitcoin-ng:

A scalable blockchain protocol." In *13th symposium on networked systems design and implementation* (2016):45-59.

[91] Christidis, Konstantinos, and Michael Devetsikiotis. "Blockchains and smart contracts for the internet of things." *Ieee Access* 4 (2016): 2292-2303.

[92] Chen, Chen, Zhuyun Qi, Yirui Liu, and Kai Lei. "Using virtualization for blockchain testing." In *International Conference on Smart Computing and Communication* (2017):289-299.

[93] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." *white paper* 3, no. 37 (2014).

[94] Bu, Gewu, Önder Gürcan, and Maria Potop-Butucaru. "G-IOTA: Fair and confidence aware tangle." In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops* (2019):644-649.

[95] Augot, Daniel, Hervé Chabanne, Thomas Chenevier, William George, and Laurent Lambert. "A user-centric system for verified identities on the bitcoin blockchain." In *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (2017):390-407.

[96] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In *Proceedings of the thirteenth EuroSys conference* (2018):1-15.

[97] Kuo, Tsung-Ting, and Lucila Ohno-Machado. "Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks." *arXiv preprint arXiv:1802.01746* (2018).

[98] Zhang, Aiqing, and Xiaodong Lin. "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain." *Journal of medical systems* 42, no. 8 (2018): 1-18.

[99] Gai, Keke, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen. "Privacy-preserving energy trading using consortium blockchain in smart grid." *IEEE Transactions on Industrial Informatics* 15, no. 6 (2019): 3548-3558.

[100] Crosby, Michael, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. "Blockchain technology: Beyond bitcoin." *Applied Innovation* 2, no. 6-10 (2016): 71.

[101]Gabison, Garry. "Policy considerations for the blockchain technology public and private applications." *SMU Sci. & Tech. L. Rev.* 19 (2016): 327.

[102]Zheng, Zibin, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. "Blockchain challenges and opportunities: A survey." *International Journal of Web and Grid Services* 14, no. 4 (2018): 352-375.

[103]Huh, Seyoung, Sangrae Cho, and Soohyung Kim. "Managing IoT devices using blockchain platform." In *2017 19th international conference on advanced communication technology* (2017):464-467.

[104]Taskinsoy, John. "Blockchain: moving beyond bitcoin into a digitalized world." *Available at SSRN 3471413* (2019).

[105]Koutmos, Dimitrios. "Return and volatility spillovers among cryptocurrencies." *Economics Letters* 173 (2018): 122-127.

[106]Valdeolmillos, Diego, Yeray Mezquita, Alfonso González-Briones, Javier Prieto, and Juan Manuel Corchado. "Blockchain technology: a review of the current challenges of cryptocurrency." In *International Congress on Blockchain and Applications* (2019):153-160.

[107]Veronese, Giuliana Santos, Miguel Correia, Alysson Neves Bessani, Lau Cheuk Lung, and Paulo Verissimo. "Efficient byzantine fault-tolerance." *IEEE Transactions on Computers* 62, no. 1 (2011): 16-30.

[108]Sukhwani, Harish, José M. Martínez, Xiaolin Chang, Kishor S. Trivedi, and Andy Rindos. "Performance modeling of PBFT consensus process for permissioned blockchain

network (hyperledger fabric)." In *2017 IEEE 36th Symposium on Reliable Distributed Systems* (2017):253-255.

[109]Schiper, André. "Early consensus in an asynchronous system with a weak failure detector." *Distributed Computing* 10, no. 3 (1997): 149-157.

[110]Aspnes, James. "Randomized protocols for asynchronous consensus." *Distributed Computing* 16, no. 2-3 (2003): 165-175.

[111]Wu, Jie. “Distributed system design”. (1998).

[112]Marzullo, Keith, and Susan Owicki. "Maintaining the time in a distributed system." In *Proceedings of the second annual ACM symposium on Principles of distributed computing* (1983):295-305.

[113]Lamport, Leslie. "Time, clocks, and the ordering of events in a distributed system." In *Concurrency: the Works of Leslie Lamport* (2019):179-196.

[114]Wilkes, Maurice V., and Roger M. Needham. "The Cambridge model distributed system." *ACM SIGOPS Operating Systems Review* 14, no. 1 (1980): 21-29.

[115]Gasser, Morrie, Andy Goldstein, Charlie Kaufman, and Butler Lampson. "The Digital distributed system security architecture." (1989).

[116]Dolev, Danny. "The Byzantine generals strike again." *Journal of algorithms* 3, no. 1 (1982): 14-30.

[117]Lamport, Leslie, and Michael Fischer. *Byzantine generals and transaction commit protocols*. Vol. 66 (1982).

[118]Rabin, Michael O. "Randomized byzantine generals." In *24th Annual Symposium on Foundations of Computer Science* (1983):403-409.

[119]Shipley, Bill. "A new inferential test for path models based on directed acyclic graphs." *Structural Equation Modeling* 7, no. 2 (2000): 206-218.

[120]Shrier, Ian, and Robert W. Platt. "Reducing bias through directed acyclic graphs." *BMC*

medical research methodology 8, no. 1 (2008): 1-15.

[121]VanderWeele, Tyler J., and James M. Robins. "Four types of effect modification: a classification based on directed acyclic graphs." *Epidemiology* 18, no. 5 (2007): 561-568.

[122]Benčić, Federico Matteo, and Ivana Podnar Žarko. "Distributed ledger technology: Blockchain compared to directed acyclic graph." In *2018 IEEE 38th International Conference on Distributed Computing Systems* (2018):1569-1570.

[123]Bentov, Iddo, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y." *ACM SIGMETRICS Performance Evaluation Review* 42, no. 3 (2014): 34-37.

[124]Gaži, Peter, Aggelos Kiayias, and Dionysis Zindros. "Proof-of-stake sidechains." In *2019 IEEE Symposium on Security and Privacy* (2019):139-156.

[125]Li, Wenting, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. "Securing proof-of-stake blockchain protocols." In *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (2017):297-315.

[126]Saleh, Fahad. "Blockchain without waste: Proof-of-stake." *The Review of financial studies* 34, no. 3 (2021): 1156-1190.

[127]Raval, Siraj. *Decentralized applications: harnessing Bitcoin's blockchain technology*. " O'Reilly Media, Inc.", (2016).

[128]Cai, Wei, Zehua Wang, Jason B. Ernst, Zhen Hong, Chen Feng, and Victor CM Leung. "Decentralized applications: The blockchain-empowered software system." *IEEE Access* 6 (2018): 53019-53033.

[129]Benčić, Federico Matteo, and Ivana Podnar Žarko. "Distributed ledger technology: Blockchain compared to directed acyclic graph." In *2018 IEEE 38th International Conference on Distributed Computing Systems* (2018):1569-1570.

[130]Vukolić, Marko. "The quest for scalable blockchain fabric: Proof-of-work vs. BFT

replication." In *International workshop on open problems in network security* (2015):112-125.

[131]Bentov, Iddo, Ariel Gabizon, and Alex Mizrahi. "Cryptocurrencies without proof of work." In *International conference on financial cryptography and data security* (2016):142-157.

[132]King, Sunny. "Primecoin: Cryptocurrency with prime number proof-of-work." *July 7th* 1, no. 6 (2013).

[133]Sankar, Lakshmi Siva, M. Sindhu, and M. Sethumadhavan. "Survey of consensus protocols on blockchain applications." In *2017 4th International Conference on Advanced Computing and Communication Systems* (2017):1-5.

[134]Kuo, Tsung-Ting, Hyeon-Eui Kim, and Lucila Ohno-Machado. "Blockchain distributed ledger technologies for biomedical and health care applications." *Journal of the American Medical Informatics Association* 24, no. 6 (2017): 1211-1220.

[135]Chen, Guang, Bing Xu, Manli Lu, and Nian-Shing Chen. "Exploring blockchain technology and its potential applications for education." *Smart Learning Environments* 5, no. 1 (2018): 1-10.

[136]Angraal, Suveen, Harlan M. Krumholz, and Wade L. Schulz. "Blockchain technology: applications in health care." *Circulation: Cardiovascular quality and outcomes* 10, no. 9 (2017): e003800.

[137]Cole, Rosanna, Mark Stevenson, and James Aitken. "Blockchain technology: implications for operations and supply chain management." *Supply Chain Management: An International Journal* (2019).

[138]Hackius, Niels, and Moritz Petersen. "Blockchain in logistics and supply chain: trick or treat?." In *Digitalization in Supply Chain Management and Logistics: Smart and Digital Solutions for an Industry 4.0 Environment. Proceedings of the Hamburg International Conference of Logistics , Vol. 23* (2017):3-18.

[139]Kshetri, Nir. "1 Blockchain's roles in meeting key supply chain management objectives." *International Journal of Information Management* 39 (2018): 80-89.

[140]Li, Ruinian, Tianyi Song, Bo Mei, Hong Li, Xiuzhen Cheng, and Limin Sun. "Blockchain for large-scale internet of things data storage and protection." *IEEE Transactions on Services Computing* 12, no. 5 (2018): 762-771.

[141]Dai, Mingjun, Shengli Zhang, Hui Wang, and Shi Jin. "A low storage room requirement framework for distributed ledger in blockchain." *IEEE Access* 6 (2018): 22970-22975.

[142]Chen, Yi, Shuai Ding, Zheng Xu, Handong Zheng, and Shanlin Yang. "Blockchain-based medical records secure storage and medical service framework." *Journal of medical systems* 43, no. 1 (2019): 1-9.

[143]Yang, Zhe, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. "Blockchain-based decentralized trust management in vehicular networks." *IEEE Internet of Things Journal* 6, no. 2 (2018): 1495-1505.

[144]Mengelkamp, Esther, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. "A blockchain-based smart grid: towards sustainable local energy markets." *Computer Science-Research and Development* 33, no. 1 (2018): 207-214.

[145]Dorri, Ali, Salil S. Kanhere, and Raja Jurdak. "Towards an optimized blockchain for IoT." In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation* (2017):173-178.

[146]Longo, Francesco, Letizia Nicoletti, Antonio Padovano, Gianfranco d'Atri, and Marco Forte. "Blockchain-enabled supply chain: An experimental study." *Computers & Industrial Engineering* 136 (2019): 57-69.

[147]Mengelkamp, Esther, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. "A blockchain-based smart grid: towards sustainable local energy markets." *Computer Science-Research and Development* 33, no. 1 (2018): 207-214.

[148]Mylrea, Michael, and Sri Nikhil Gupta Gourisetti. "Blockchain for smart grid resilience:

Exchanging distributed energy at speed, scale and security." In *2017 Resilience Week* (2017):18-23.

[149]Gao, Jianbin, Kwame Omono Asamoah, Emmanuel Boateng Sifah, Abla Smahi, Qi Xia, Hu Xia, Xiaosong Zhang, and Guishan Dong. "GridMonitoring: Secured sovereign blockchain based monitoring on smart grid." *IEEE Access* 6 (2018): 9917-9925.

[150]Li, Xiaoqi, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. "A survey on the security of blockchain systems." *Future Generation Computer Systems* 107 (2020): 841-853.

[151]Park, Jin Ho, and Jong Hyuk Park. "Blockchain security in cloud computing: Use cases, challenges, and solutions." *Symmetry* 9, no. 8 (2017): 164.

[152]Khan, Minhaj Ahmad, and Khaled Salah. "IoT security: Review, blockchain solutions, and open challenges." *Future Generation Computer Systems* 82 (2018): 395-411.

[153]Zhang, Rui, Rui Xue, and Ling Liu. "Security and privacy on blockchain." *ACM Computing Surveys (CSUR)* 52, no. 3 (2019): 1-34.

[154]Halpin, Harry, and Marta Piekarska. "Introduction to Security and Privacy on the Blockchain." In *2017 IEEE European Symposium on Security and Privacy Workshops* (2017):1-3.