MONASH University

Building Secure and Efficient Blockchain Systems

by

Bin Yu

A thesis
presented to Monash University
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
information technology

Monash University, Melbourne, Australia, 2019

Publications during enrolment

[1] Yu, B., Wright, J., Nepal, S., Zhu, L., Liu, J. and Ranjan, R., 2018. IoTChain: Establishing trust in the internet of things ecosystem using blockchain. IEEE Cloud Computing, 5(4), pp.12-23.

[2] Yu, B., Liu, J.K., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P. and Au, M.H., 2018, September. Platform-independent secure blockchain-based voting system. In International Conference on Information Security (pp. 369-386). Springer, Cham.

[3] Yu, B., Kermanshahi, S.K., Sakzad, A. and Nepal, S., 2019, October. Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks. In International Conference on Provable Security (pp. 303-318). Springer, Cham.

[4] Yu, B., Liu, J., Nepal, S., Yu, J. and Rimba, P., 2019. Proof-of-QoS: QoS Based Blockchain Consensus Protocol. Computers & Security, p.101580.

Thesis including published works declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes 4 original papers published in peer reviewed journals/conferences and 1 unpublished publication. The core theme of the thesis is secure protocol design for blockchain systems. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the Faculty of Information and Technology, Monash University under the supervision of Joseph Liu, Ron Steinfeld and Surya Nepal (data61 CSIRO).

(The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research.)

My contribution to the work involved the following:

Thesis Chapter	Publication Title	Status	Nature and % of student contribution	Co-author name(s) Nature and % of Co-author's contribution*	Co-author(s), Monash student Y/N*
2	P2PBFS: P2P Based Blockchain File System	Submitted	80%. Concept and collecting data and writing first draft	1. Joseph Liu, Supervision 5% 2. Surya Nepal, Supervision 5% 3. Amin Sakzad, Supervision 5% 4. Jiangshan Yu, Supervision 5%	No

3	Proof-of-QoS: QoS Based Blockchain Consensus Protocol	Published	80%. Concept and collecting data and writing first draft	1. Joseph Liu, Supervision 5% 2. Surya Nepal, Supervision 5% 3. Jiangshan Yu, Supervision 5% 4. Paul Rimba, offer comments 5%	No
4	Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks	Published	70%. Concept and collecting data and writing first draft	1.Shabnam Kasra Kermanshahi, contribute in proof and algorithm design 20% 2.Surya Nepal, Supervision 5% 3.Amin Sakzad, Supervision 5%	Yes
5.1	IoTChain: Establishing Trust in the IoT-based Applications Ecosystem Using Blockchain	published	75%. Concept and collecting data and writing first draft	Jarod Wright 10%, Surya Nepal,Supervision 5% Liming Zhu,Supervision 3% Joseph Liu,Supervision 4% Rajiv Ranjan 3%,	No
5.2	Platform-independent Secure Blockchain-Based Voting System	Published	75%. Concept and collecting data and writing first draft	1. Joseph Liu, Supervision 5% 2. Surya Nepal, Supervision 5% 3. Amin Sakzad, Supervision 5% 4. Ron Steinfeld, Supervision 5% 5.Paul Rimba, offer comments 2.5% 6. Man Ho Au, offer SLRS algorithm 2.5%	No

I have not renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student signature: **Date:**

The undersigned hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor signature: **Date:**

Publications

The following are the submitted/published papers during the P.h.D study.

[1] Yu, Bin, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. "Platform-independent secure blockchain-based voting system." In International Conference on Information Security, pp. 369-386. Springer, Cham, 2018. [published]

[2] Yu, Bin, Jarod Wright, Surya Nepal, Liming Zhu, Joseph Liu, and Rajiv Ranjan. "IoTChain: Establishing trust in the internet of things ecosystem using blockchain." IEEE Cloud Computing 5, no. 4 (2018): 12-23. [published]

[3] Yu, Bin, Shabnam Kasra Kermanshahi, Amin Sakzad, and Surya Nepal. "Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks." In International Conference on Provable Security, pp. 303-318. Springer, Cham, 2019. [published]

[4] Yu, Bin, Joseph Liu, Surya Nepal, Jiangshan Yu, and Paul Rimba. "Proof-of-QoS: QoS Based Blockchain Consensus Protocol." Computers & Security (2019): 101580. [published]

[5] P2PBFS: P2P based blockchain file system [submitted]

Abstract

With the fast development of computer science, the public is becoming more sensitive to the security of their private data. The public wants their data to be processed quickly but also securely. Data encryption, zero-knowledge proof and secret sharing schemes have been proposed to provide a secure environment for participants to exchange their data without leaking it to unrelated parties. However, for the majority of the secure distribution systems, we need a trusted third-party to organise the communication between different participants. For instance, a secured voting system needs all the voters to be able to trust the results published by the tallying centre or voting administrator. The security of the system can be improved if the trusted third party is removed from the distributed systems.

Bitcoin was proposed by Satoshi Nakamoto in 2008. It takes advantage of a peer-to-peer network in order to construct a system described as "a system for electronic transactions without relying on trust". In Bitcoin, a set of transactions are stored in an immutable structure called blocks and through the peer-to-peer network the blocks are duplicated and stored for all the participants. Consensus protocols ensure that the majority of the participants have a consistent view of the order of the blocks, which are generated by participants known as miners. Since the transactions in the block are verified by all the participants there exists no requirement for a trusted third-party to be involved in the system. The trust-free feature has aroused the interests of researchers and the Bitcoin design scheme has been generalised and implemented in a system known as Blockchain. The concept of having computer protocol that is intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract, without the need for third parties, has been achieved with the help of blockchain. Blockchain technology not only brings this revolutionary business model to industry but also drives forward cryptographic research such as multi-party computation, succinct zero-knowledge proof and threshold signatures.

Though blockchain technology brings a great convenience to the industry, the public is still concerned with the security and efficiency of blockchain systems. My research focuses on the following aspects: 1) The issue of increasing storage demand for blockchain systems. The miners in blockchain systems need to store all the blocks created, starting with the genesis block, and with the fast development of blockchain technology, the storage requirement to participate in the system has increased greatly. This discourages participants who have restricted storage, such as IoT and mobile devices, to contribute to the system. It is therefore crucial to quickly resolve this storage consumption issue by proposing new blockchain storage schemes; 2) The issue of the efficiency of the blockchain consensus protocols. The proof-of-work transaction throughout is often blamed, while the proof-of-stake based consensus protocols can be easily dominated by the stakeholders, who occupies a

vast amount of resources. Improving the blockchain consensus protocol throughput while providing a fair environment for all the participants is a popular research topic; 3) The leakage of the payment path in the payment channel networks. Though the payment channel networks increase the transaction throughput, it may leak the payment path from the sender to the receiver; 4) The security of the blockchain-based applications. Since all the data in the blockchain is publicly accessible it is compulsory to have cryptographic schemes to ensure that blockchain-based applications achieve public verifiability, whilst preserving the privacy and security.

In the last three years' of study, I contributed to the security and privacy protection in blockchain research in the following four aspects: 1) To address the issue of growth of blockchain storage consumption, I proposed a peer-to-peer blockchain-based system that decouples the need for miners to store all the blocks locally. With the help of our blockchain-based system, storage restricted devices can also be involved in the blockchain mining process; 2) To address the blockchain consensus protocol low-throughput issue, I proposed a hybrid consensus protocol that takes advantage of the Byzantine Fault Tolerance based consensus protocol to achieve a high transaction throughput while preserving the openness of the scheme, which allows the public to join the system without obtaining any permission from a third party; 3) To address the payment path leakage in the payment channel networks, I propose a Chameleon Hash based payment scheme that avoids anyone recovering the payment path from the sender to the receiver; 4) To address the security of blockchain-based applications, I propose two papers which demonstrate that blockchain can be applied to build a secure and privacy-preserving electronic voting system, and also used to trace and manage the ownership of IoT devices and their data, without any trusted parties.

Acknowledgements

I would like to thank my wife Joyce who supports me for the 3 years study. I would also thank my supervisors Joseph Liu, Surya Nepal and Ron Steinfeld for their time and supports. I would also like to thank all the little people who made this thesis possible.

Table of Contents

List of Tables

List of Figures

Abbreviations

BFT Byzantine Fault Tolerance 10, 11, 39, 40, 42–44, 60, 68, 92, 95

CHTLC Chameleon Hash Time-Lock Contract xi, xv, 12, 69, 71, 78–80, 82, 83, 85, 86

HTLC Hash Time-Locked Contracts 12, 72–74

IoT Internet of Things xi, 20, 88–91, 95, 96, 98–102

MGB Management Block x, 11, 41, 46–48, 50–55, 58, 59, 62, 63

MHTLC Multi-hop Hash Time-Lock Contract 12, 69–71, 85, 86

MPC Multi-party Computation 87

NoT Network-of-Things 89

NTP Network Time Protocol 52

P2P Peer-to-Peer 8, 9, 15–20

P2PBFS Peer-to-peer Blockchain File System viii, ix, xv, 9, 15–24, 26–28, 30, 31, 33, 35, 37, 38

PBFT Practical Byzantine Fault Tolerance 4, 7, 11, 21, 40–44, 46, 53, 54, 58–60, 64–67

PCN Payment Channel Network xi, 69, 70, 74

PCNs Payment Channel Networks xi, 69, 70, 73, 75, 76, 78, 80, 86

PoQ Proof-of-QoS x, xv, 10, 11, 39–45, 47, 48, 52, 53, 56, 59, 62–68

PoS Proof-of-Stake 7, 9, 39, 40, 92, 94

PoW Proof-of-Work 4, 5, 7, 9, 21, 39, 40, 42–44, 92–94

PoX proof-of-X 4, 21

QoS Quality-of-Service xv, 39–41, 45–56, 58

VFS Linux virtual file system 22

VRF Verifiable Random Function 52

Chapter 1

Introduction

1.1 Research Background

1.1.1 Blockchain

Trust plays a critical role in information exchanges. It helps different entities deal with each other more effectively and is often a key element in any collaborative system. Traditionally, trust is established among different entities by centralised institutions, such as banks or government agencies. With the help of these centralised institutions, different entities can cooperate with a certain degree of confidence. Blockchain, known as an electronic ledger, tries to replace such centralised institutions by distributing the trust in a decentralised network. In a blockchain system, the ledger is immutable and not held on a single server, instead it is held on all servers in the network. This open feature of blockchain allows any participant to modify the ledger under a set of rules dictated by a consensus protocol. The consensus protocol requires the majority of the blockchain participants to agree on the modification of the ledger to ensure the trustworthiness of the blockchain. Once the ledger is updated, all participants update their ledger locally and simultaneously. If any of the participants violate the consensus protocol to propose a new data entry on the chain, the network treats that update as an invalid operation. In practical terms, transactions are bundled together and submitted to the blockchain as a block. Cryptographic techniques are applied to link all blocks in a deterministic order. The cryptographic algorithm also guarantees that the blocks are immutable, which means that once a block is appended to the chain it cannot be tampered with. Compared with other distribution systems, blockchain has the following two outstanding features: 1) It distributes the trust of the

system from one, or a few, central node(s) to a set of nodes and 2) The blockchain service can be offered by any of the available participants, thus makes the Denial-of-service attack difficult to be deployed.

1.1.2 Smart Contract

The terminology "smart contracts" was coined by computer scientist Nick Szabo in 1994, to bring together contract law, and related business practices, and the design of electronic-commerce protocols between strangers on the Internet [7]. If we take the chain as an append-only database for a complete blockchain system, a Turing-complete language is needed to guide how the blocks are constructed and stored. Smart contracts can be regarded as the Turing-complete language that guides the block generation [1]. Smart contracts enables interactions between end-users and a blockchain by allowing the end-users to create and/or query data on the blockchain. With the help of the smart contract, a business model in which it is difficult to establish trust can be migrated onto blockchain.

We take Hyperledger fabric [8] as an example to show the general scheme of the smart contract. At least three interfaces should be implemented, which are; **init()**, **invoke()**, and **query()**. **init()** is the interface that is invoked when the smart contract is loaded. **init()** initialises the smart contract system parameters before the end-user interacts with the smart contract. **query()** is how the interface handles the query request from the blockchain end-users. **invoke()** is the interface that is called when the end-user wants to put the data on the blockchain. Unlike **query()**, **invoke()** is executed on all validation nodes to ensure the consistency of data on the blockchain.

We demonstrate how the smart contract is deployed on blockchain in Fig. 1.1. First, the smart contract administrator needs to compile the smart contract application into a binary code, so that it can be executed on the Hyperledger fabric. The administrator then deploys the smart contract on the blockchain through the smart contract front-end. The administrator receives notifications when the status of the smart contract is updated. For instance, when the smart contract is deployed successfully, the application receives a message saying that the smart contract is now running on the blockchain. Finally, end-users can access the service through the interface provided by the smart contract front-end server.

In the high level, we can divide the transaction confirmation on the blockchain into 6 steps.

[1] If there is no factor that forces the smart contract to terminate such as gas limitation in Ethereum.

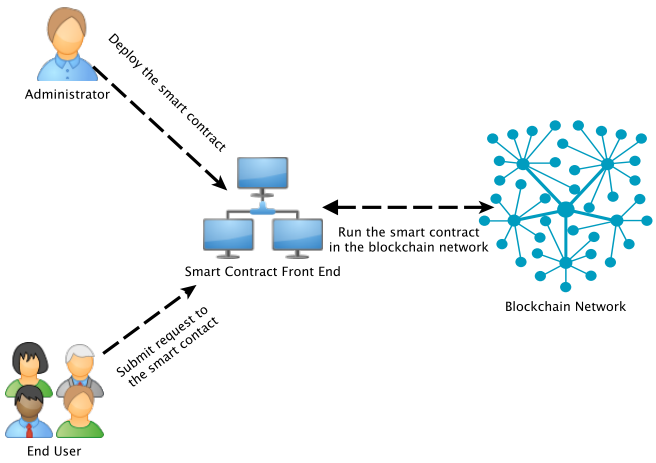


Figure 1.1: Smart contract on blockchain.

1. The administrator compiles and deploys the smart contract on the blockchain through the smart contract front end server.
2. The end-users invoke the smart contract through the smart contract front-end server.
3. The validity of the transaction is verified by the miners and the valid transaction is involved into a block together with other valid transactions.
4. The miner who generates the block broadcasts the block to the whole network.
5. Based on different consensus protocols, for a given time slot, only one of the blocks is regarded as authoritative by the system.
6. The rest of the miners verify the correctness of the block and append new blocks to the chain.

Finally, the execution result of the smart contract is accepted and stored on the blockchain.

1.1.3 Consensus Protocol and Block Storage

Blockchain is an append-only ledger [9]. It can be regarded as a distributed database that stores the transactions in a unique tree structure known as a Merkle tree [10]. If a transaction passes the validity check, it is included in a candidate block together with

other transactions. Nodes in the blockchain system collaborate to determine the sequence of the blocks on the chain (consensus protocol). Consensus protocol guarantees that all the participants in the system have an identical view of the sequence of the blocks. However, in a practical scenario, due to the network latency of transaction propagation and malicious or faulty nodes, nodes may end up with different views of the block sequences (branches in the blockchain system). The consequence of the inconsistent view is the possibility of double-spending. For each block generation interval, called an epoch or a round, the consensus protocol nominates one node as a leader to propose the block while the rest agrees on the proposed block by appending it to their chain. Three main consensus protocols are applied in current blockchain systems. The first one is Proof-of-Work (PoW) consensus protocol, which selects the block nomination node by asking the blockchain participants to resolve a "maths puzzle". The PoW based blockchain achieves maximum system scalability by allowing all the participants to contribute to the block generation. However, poor transaction throughput and huge energy consumption in block generation are often blamed.

For the PoW based protocol, node nomination is based on solving a cryptographic puzzle. For the Practical Byzantine Fault Tolerance (PBFT) based protocol, node nomination is based on negotiation among participants. Finally, the proof-of-X (PoX) protocol nominates a leader based on the resource (denoted as X) that each node has (the resources can be the deposit that nodes made or the storage that the nodes can provide). Hybrid protocols take advantage of different types of consensus protocols in their block generation phase.

For the majority of the blockchain system, the generated blocks are stored and maintained in a given order for all the miners. Once each miner receives a new block from the network, they verify the correctness of the block and append the new block at the end of their local chain. Each given block should be identical among all the miners, otherwise, the system fails to achieve consistency. Additionally, since blockchain is an append-only system, the storage consumption on each miner increases with the growth of the total amount of transactions. Since the origin of Bitcoin, the system data size has been increased from about 10 GB in 2011 to about 150GB in early 2019. Since all the miners have had all identical blocks since the genesis, it has resulted in a vast storage waste. "Lightweight client" has been proposed by Bitcoin [2] to allow some nodes to participant in the block generation without storing all the blocks. When the "lightweight client" needs a specific block, it sends a query to the full nodes it trusts, and so fully trusts the blocks returned by the full node. This scheme has already been deployed in the Bitcoin network and benefits the nodes that have restricted storage.

[2]https://bitcoin.org/en/full-node

1.1.4 Payment Channel Network

For the PoW based blockchain, the transaction throughput is often blamed, as it can only process 3 to 7 transactions per second. To address this issue, an off-chain transaction scheme has been implemented in the Bitcoin network, called the Lightning Network [11]. The high-level idea for the Lightning Network is to have participants involved in a payment session to make a deposit in a joint account, which needs both of their signatures to spend the money. They put this joint account on the blockchain to let the blockchain network confirm its validity. Once their joint account is accepted by the blockchain and a payment channel is established between them they can have transactions off-the-chain and these two parties can arrange how to allocate the deposit in the joint account. If either of them does not comply with the protocol, he/she faces the chance of losing all his/her deposit. When they want to cancel the channel, either of them needs to commit the joint account's latest status on the blockchain. With the help of the payment channel, the two parties can have as many transactions as they require, and the transaction throughput is beyond the restriction of the blockchain.

Though the payment channel addressed the issue of low transaction throughput, it needs the two parties who are involved in a payment to have the payment channel established first, which discourages users who do not have frequent payments. To address this issue, we may find some existing channels in the network and asks these channels to forward the transactions for us by paying them a reasonable fee. This payment model benefits the payer/payee as they can enjoy the high transaction throughput without establishing the channel in advance, additionally, the participants who forward the payment can obtain the transaction fees that are offered by the payer.

1.2 Research Questions and Scope

Blockchain, which is regarded as a prominent solution for providing a self-organised data sharing and computation platform also brings research questions, such as: 1) It is necessary to consider how to build secured trust-free applications on the blockchain platform; 2) It is critical to propose a consensus protocol that blockchain networks can reach the consensus state within a reasonable time latency; 3) It is necessary to increase the blockchain transaction throughput by providing a fast and secure payment channel and 4) It is necessary to deal with the increasing demand of blockchain storage.

To better focus on the security issues and the efficient of the blockchain platform, I define my research questions as:

1. **How to deal with the increase of the storage demand in the blockchain?**

   Since the full nodes (regarded as the miners) are responsible for verifying the correctness of proposed transactions and in replying to transaction queries from the end-users, they need to store all the blocks since the start of the system. As the total number of the blocks is constantly increasing, the storage needs have attracted researchers' attention. Roughly speaking, two approaches exist to address this issue. One is transaction relocation, which means the system inspects the transactions regularly to generate the summary of the previous block status, known as checkpoints. As a result, the system does not need to store all the blocks since the genesis, it only needs to store the blocks since the checkpoint, which was an approach introduced by Amelchenko [12]. However, Amelchenko's approach requires a routine service to inspect every transaction, which results in a high workload, and this scheme is impossible for the light-weight clients, such as Internet of Things (IoT) devices. Another approach is to allow some nodes to only store the block head and when they need to know the transactions within a block, they request the block from the full nodes. In this way, the nodes have full control of whether to store the duplicated blocks or not. In Ethereum, these nodes that do not hold the whole blocks are called light nodes. The light node scheme was first implemented in Ethereum version 1.8 and currently works within the Ethereum main network seamlessly. As the light node is a compromise between functionality and storage, light nodes are not responsible for querying the transactions in a specific block.

   In order to allow some storage restricted devices, like the IoT devices, to participant in the blockchain system, blockchain systems usually provide a scheme to allow the storage-restricted nodes to communicate with some trusted peers in order to obtain the blocks. The issue with this scheme is the unstable connection between the storage-restricted devices and the trusted peers. As a result, the performance of the storage-restricted devices is unpredictable. Since there is no unified block access method/policy among all the participants, the performance of the whole system cannot be easily optimised. All the nodes who offer the block query service are required to have all the blocks created since the start of this blockchain system. The consequence of the aforementioned requirement is the data redundancy, which could be avoided.

2. **How to achieve a high transaction throughput while preserving the predefined security requirement?**

   For any blockchain system, there is no node acting as a management node to coordinate the communication between different participants. Keeping data consistent

among a large number of nodes is a serious problem. Typically, three protocols exist to address this issue: 1.) PoW protocol, which is widely adopted by crypto-currency blockchain systems like Bitcoin [13] and Zcash [14]. 2). PBFT protocol, which is applied by hyperledger [15]. The PoW protocol shows a remarkable performance in nodes scalability, which means the PoW protocol can support large scale networks. However, the network latency is poor, which means it takes a long time before the whole network reaches a consistent state. PBFT protocol has great performance on network latency but suffers from restrictions of network scale, which means it can only support a limited number of nodes and 3.) Instead Proof-of-Stake (PoS) consensus protocol use is proposed in order to address the issues of the PoW and PBFT. PoS allows some nodes that hold the majority of the resources in the network to deal with the blockchain generation. It raises the concern that these stakeholders may collude with each other to dominate the whole system.

3. **How to build a secure payment channel network to achieve high transaction throughput?**

   Due to the nature of blockchain, the transaction throughput is much slower than the centralised trading systems. For the blockchain system, the majority of time is spent on transaction confirmation among all the participants. However, if the two parties that are involved in the trading agree on the transactions, it is not necessary to put their transactions on the blockchain. The idea of having off-chain transactions is implemented in the Bitcoin network known as the lighting network, in which the two parties can have a payment connection, known as the payment channel, directly without interacting in the blockchain. The issue with this scheme is how we can ensure that all the parties in a given payment channel obey the protocol. Some nodes may not have transactions that frequently, so they may not wish to maintain the payment channel while enjoying the benefits provided. One of the solutions is to ask the nodes that have the channel established to forward the traffic, by paying them a transaction fee. It is required that the system preserves the payment path for the nodes that forward the transaction and therefore ensures that if any intermediate nodes do not cooperate in forwarding the transaction, none of the participates involved in the payment paths suffers from any financial loss.

4. **How to build secure applications on blockchain?**

   Although blockchain provides distributed trust and an anonymous environment for applications which are trust sensitive, there is still an issue about how can we leverage these features while preserving the participants' privacy. Since all the data on the

chain is shared among the blockchain miners if the sensitive data is not handled properly it could result in a data leak. Another issue is how to design a proper protocol to address the existing issues that need to be deployed in a trust-free environment, which should preserve the security assumptions while also achieving reasonable system performance. Thanks to the blockchain, the ownership and management of data and devices can also be migrated from the centralised system to the trust-free environment. However, there is the issue of how the blockchain system can manage the data access on devices when they are transferred from one user to another. Additionally, the blockchain system should also stop the previous owner from tracking the ownership of the device once it is transferred to another owner.

1.3 Contribution to Knowledge

My research mainly focuses on the blockchain storage optimisation, blockchain consensus protocols, blockchain payment network and the security and privacy preserving applications. The contributions are summarised as:

1. **Distributed Blockchain File System**

   P2PBFS: P2P Based Blockchain File System in chapter 2.

   In this paper, we propose a Peer-to-Peer (P2P) based file system to deal with the growth of the demand for the block storage, and the vast amount of duplicated block data. With the proposed system, the miners do not need to have the full copy of the whole blockchain, and they can mainly focus on the block generation while free from maintaining the storage of the blocks. With the unified file operation interface, it decouples the block storage from the block generation, so the devices which have limited storage can join the system to do the verification without storing any blocks. With the help of the distributed file system, the performance of the whole distributed system is easier to be managed by asking all the nodes who contribute in the block storage to comply with the distributed file system protocol. The contribution of this paper is summarised as follows:

   (a) **A new economic model for blockchain participants:** In the existing blockchain systems, the primary motivation to participate is to claim mining

rewards. However, the nodes which fail to generate the blocks end up with nothing even though they contribute to providing computing and storage resources. Different from any existing blockchain economic model, in our framework, the participants, who provide the storage resources without participating in mining (e.g., PoW, PoS), can also get rewards by offering the block query service to the blockchain system.

(b) **An innovative distributed file system dedicated for block storage:** The features of P2PBFS are summarised as:

- **New blockchain system architecture to resolve block duplication issue:** To the best of our knowledge, all existing blockchain systems require the full nodes to have a copy of all the blocks since genesis. P2PBFS decouples the function of block storage from block generation by migrating the function of block storage to some dedicated nodes, known as storage nodes. In P2PBFS, miners only need to store the block abstract (detail is discussed in Section 2.6.1). Compared with the Bitcoin system, miners in P2PBFS saves 99.96% storage in blockchain maintenance.
  The innovative blockchain architecture is shown in Fig. 2.1(b). Since the miners are free from storing blocks, the issue of a large number of duplicate blocks can be avoided. Additionally, since the role of storing the blockchain data is separated from the miner, resource-constrained mobile and IoT devices may take this advantage and be involved in the mining process, where intensive computing power is not needed (e.g. PoW), to increase the degree of decentralisation.
- **Transparent block access/store with the optimized query:** We achieve block access transparency by providing a P2PBFS driver. The driver offers the standard file operations (**open**, **write**, **read** and **close**). The block storage transparency is achieved by employing an operating system mutex-lock scheme. Additionally, Since P2PBFS does not record the sequence of the block, it is free from handing the "forks" in the blockchain systems. By caching storage node information in block abstract, miners can locate a block instantly without searching in the storage network, which is dramatically faster than the general P2P storage solutions.
- **Scalability and availability:** We allow the storage node to join and leave P2PBFS freely. When the new nodes join the network, routing service running on the nodes will migrate the blocks to the new nodes according to the XOR distance between the node ID and the block hash. Block migration is transparent to the clients (The implementation of scalability is

discussed in section 2.6.3.7). To achieve availability, we duplicate the blocks on to multiple nodes, and this duplication is independent of the miners and much less than duplications per miner. Based on our experiment results (in the system consists of $20,000$ nodes and $200,000$ blocks), P2PBFS can still achieve 100% block query successful rate even though 35% of nodes are off-line.

- **Secure, practical, and comprehensive performance evaluations:** One of our system design targets is to have a good tolerance under different kinds of attacks (e.g., grinding attack [16] and Sybil attack [1]). We discuss the possibility of launching different types of attacks toward our system in Section 2.7 to show the robustness of our system. The details of our system performance evaluation are provided in Section 2.8.
- **Profit-driven decentralized file system:** Compared with the solution of storing the blocks in the existing cloud file system (e.g., Dropbox and Google Drive), our scheme does not depend on any third party to guarantee service quality. Additionally, each participant who provides storage can claim fees from the data they provided which is irrelevant to the mining fees.

2. **A High Throughput Blockchain Consensus Protocol**

   Proof-of-QoS: QoS Based Blockchain Consensus Protocol [2] in chapter 3.

   In this paper, we address the trade-off between the network latency and deployment scalability [17] by proposing a new hybrid consensus protocol called PoQ. In PoQ, we divide the whole blockchain network into different regions, in a block proposal round, each region nominate a node based on the nodes Quality of service (the deposit the node made, the error rate, the activity rate and the reference factor who refer this node to join in the network). The nominated nodes from different regions run Byzantine Fault Tolerance (BFT) based protocol to propose the blocks that accepted by all the nominated nodes. Once the new blocks are generated, they are broadcast to the whole network, and the rest of nodes append this new block to the end of their own chain.

   The contributions of this paper are summarised as the following:

   - **Openness:** PoQ achieves openness by segregating the whole network into autonomous regions. New nodes are free to join one of the regions they prefer. When a node joins a region, peer nodes check whether this node's deposit has

already existed in the MGB to avoid a node joining multi-regions with the same identity (node address).

- **Fairness and energy-saving:** From the perspective of the nodes, PoQ provides a fair environment by selecting a representative of a region based on its QoS. Activity rate factor is introduced to avoid some nodes dominating the block generation by making a vast amount of deposit. A random function is applied to allow nodes with a similar QoS level to have an equal chance to be the representative. The online QoS update scheme encourages the nodes to comply with the protocol rules; otherwise, they will not only lose their QoS score but also lose their deposit as a punishment. It is an energy-saving protocol as nodes do not waste a tremendous amount of energy for competing for the block proposal.
- **Resilience and Robust:** Our protocol offers the liveness and safety properties, provided that at most $\lfloor \frac{n-1}{3} \rfloor$ out of n regions are simultaneously faulty. In PoQ, compromising the node nomination is more difficult and expensive — the nodes are nominated based on their QoS. If a node is deemed malicious, its nomination frequency drops dramatically as shown in Fig.3.3(c). It is impractical for a malicious node to misbehave in the system while not being caught by the peer nodes for a long time. Since the node nomination is not based on the asset or computing power, and no one can increase its nomination frequency by occupying a large number of resources in the network. Instead, PoQ encourages the node to do the mining by providing high QoS.
- **Performance:** High throughput is achieved by allowing the regions' representative nodes to construct the PBFT network to propose and accept blocks among all regions. Through the experiments, we show that the system creates a fair environment for all the participants, and it achieves a very high transaction throughput of up to 9.7K transactions per second (TPS). Considering Paypal and VISA, which handles hundreds of TPS and 2K TPS respectively [18], the proposed system is suitable for practical use. We also implement PoQ on BFT-SMaRt [19] to demonstrate the feasibility of building PoQ on the existing BFT protocols.

3. **Privacy Preserving Payment Channel Network**

   Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks in chapter 4.

The idea of payment channel is proposed to address the low transaction throughput in blockchain systems. With the help of the payment channel, two parties can have as many off-chain transactions as they wish while free from the transaction throughput limitations on the blockchain. By paying a reasonable amount of fee, end users can employ the existing payment channels to forward the payment thus avoid having the payment channel established firstly. Hash Time-Locked Contracts (HTLC) is one of the protocols applied by Bitcoin lighting network to ensure that all the parties involved in a payment to obey the protocols in a payment. However, for the HTLC protocol, one of the privacy issues is the leakage of the payment path to the parties involved. To address this issue, we propose a new scheme called CHTLC to make the trace of the payment path impossible. The contributions of our scheme are summarised as follows:

- We propose a new payment protocol called CHTLC, which hides the payment path from the view of payment participants and the observer who analyses the blocks that are committed on the chain. We also prove the security of CHTLC and show that CHTLC achieves the same level of security as Multi-hop Hash Time-Lock Contract (MHTLC).
- We conduct computer simulations to show that our proposed CHTLC protocol is efficient both in time and space. Our experimental results indicate that in comparison with MHTLC protocol [20], our protocol is much more efficient in payment forwarding. That is, MHTLC spends 309 ms per user to generate the zero-knowledge proof required for the payment, whereas such procedure is avoided in our protocol. For each intermediate node, MHTLC needs to transmit $1,650$KB data between each node, while it is reduced to only 0.96KB data in our CHTLC protocol.

4. **Blockchain-based applications**

Though blockchain has existed for almost ten years, the application scenarios and how it can contribute to the existing business model are not well studied. Through this study, I proposed two papers that address the security and the privacy of the blockchain-based applications.

IoTChain: Establishing Trust in the Internet of Things Ecosystem Using Blockchain [3] in chapter 5.

For the traditional data/devices management system, it needs a trusted third party to be involved in managing the transferring of the data/devices from one use to

another. This scheme works perfectly if all the participants involved in the system trust the third party will not collude with any use or release the sensitive data to the public. In a practical scenario, this assumption is difficult to achieve.

To address the trust issue described above, we provide the IoTchain which has the following features.

- We elaborate how blockchain can be applied in the device/data ownership management scenario in which trust is difficult to achieve.
- We propose a reference platform called IoTchain to demonstrate the feasibility of employing the blockchain to manage the IoT device/data ownership without trusting any third party.
- We introduce some future research directions and challenges in blockchain-based ownership management systems.

Blockchain-based Privacy Preserving voting System [4] in chapter 5.

Cryptographic techniques are employed to ensure the security of voting systems in order to increase its wide adoption. However, in such electronic voting systems, the public bulletin board that is hosted by the third party for publishing and auditing the voting results should be trusted by all participants. Recently a number of blockchain-based solutions have been proposed to address this issue. However, these systems are impractical to use due to the limitations on the voter and candidate numbers supported, and their security framework, which highly depends on the underlying blockchain protocol and suffers from potential attacks (e.g., force-abstention attacks). To deal with two aforementioned issues, we propose a practical platform-independent secure and verifiable voting system that can be deployed on any blockchain that supports the execution of a smart contract. Verifiability is inherently provided by the underlying blockchain platform, whereas cryptographic techniques like Paillier encryption, proof-of-knowledge, and linkable ring signature are employed to provide a framework for system security and user-privacy that are independent from the security and privacy features of the blockchain platform. We analyse the correctness and coercion-resistance of our proposed voting system. We employ Hyperledger Fabric to deploy our voting system and analyse the performance of our deployed scheme numerically.

The contributions of our electronic voting system are summarised as:

- Our voting system does not depend on a centralised trusted party for ballots tallying and result publishing. Compared with traditional voting systems, which highly depend on a centralised trusted party to tally the ballots and publish the result, our voting system takes advantage of a blockchain protocol to eliminate the need for a centralised trusted party.
- Our voting system is platform-independent and provides comprehensive security assurances. Existing blockchain-based voting systems highly depend on the underlying cryptocurrency protocols. Receipt-freeness [21] and correctness of the voting result are hard to achieve (we analyse the blockchain-based voting system explicitly in Section 5.2.3). The security of our voting system is achieved by cryptographic techniques provided by our voting protocol itself, thus, our voting system can be deployed on any blockchain that supports smart contract. To achieve the goal of providing a comprehensive security, we employ the Paillier system to enable ballots to be counted without leaking candidature information in the ballots. Proof-of-knowledge is employed to convince the voting system that the ballot cast by a voter is valid without revealing the content of the ballot. Linkable ring signature is employed to ensure that the ballot is from one of the valid voters, while no one can trace the owner of the ballot.
- Our voting system is scalable and applicable. In order to support voting scalability, we propose two optimised short linkable ring signature key accumulation algorithms given in algorithm 8 and algorithm 9 to achieve a reasonable latency in large scale voting. We evaluate our system performance with 1 million voters to show the feasibility of running a large scale voting with the comprehensive security requirements.

Chapter 2

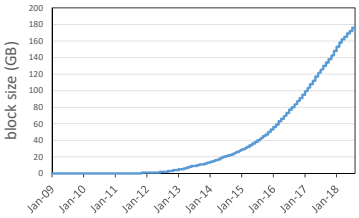
P2PBFS: P2P Based Blockchain File System

2.1 Abstract

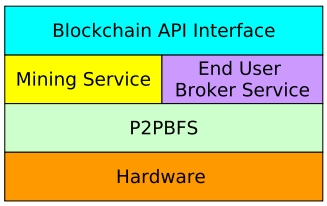
The majority of the blockchain systems require all the participating nodes in block generation to duplicate all the blocks in the whole network. This results in a high level of redundant blocks and a considerable amount of storage cost/waste. Additionally, the current design discourages pervasive, resource-constrained mobile, and IoT devices to participate in the block generation as they usually have limited storage capacity. To address the aforementioned issue, we propose an innovative P2P based file system, called P2PBFS, to store blocks for different blockchain platforms. In P2PBFS, miners are free from storing the blocks, which results in avoiding the high duplication of blocks. Additionally, the storage restricted devices can also contribute to the system by providing the storage. Through the system implementation and evaluation, we demonstrate the feasibility of deploying P2PBFS in a real distributed environment to support P2PBFS based blockchain systems.

2.2 Introduction

With the tremendous success of the Bitcoin system [13] and its use in many real-life applications, a large number of blockchain-based cryptocurrency systems and business applications have been developed and deployed. Founded on the principle of distributed trust, these early blockchain systems have encountered challenges regarding privacy, security,



(a) The growth of blockchain size.



(b) The structure of blockchain system.

Figure 2.1: The growth of blockchain size and recommend structure of P2PBFS based blockchain system.

fairness, and performance, more specifically transaction throughput. Blockchain systems like Monero [22] and Zerocash [14] employ some cryptographic techniques to enhance the transaction privacy protection. Others like snow-white [23] and Ouroboros [16] address the transaction throughput by replacing Proof-of-Work (PoW) consensus protocol with Proof-of-X (X denoting a type of resource, e.g., stake, storage, activity) ones. The performance of PBFT consensus protocol is analyzed by Sukhwani *et al.* [24] and the avaliability of the blockchain-based systems is studied by Weber *et al.* [25]. Researchers like Ekparinya *et al.* [26] focus on the man-in-the-middle attack while others focus on enhancing the security of the blockchain systems by improving the resilience of the blockchain network layer[27, 28]. It is important to note that the underlying P2P network has remained the same since the start of the Bitcoin project[27] and the same problems are carried over to follow-on blockchain systems such as Ethereum [29]. The majority of the blockchain research is focused on enhancing the system performance, availability, and data privacy, while a few studies have been carried out on blockchain storage optimization, one of the key challenges in existing blockchain systems. Furthermore, with the growth of the blockchain size such as Ethereum (shown in Fig. 2.1(a)), it has become urgent to deal with the growth of the block storage requirement. For the current design of blockchain systems, they require all the miners to have a large volume of storage to store all the blocks which discourages a large number of nodes with resource-constrained devices to participate in the network as miners. If this continues, it will have a detrimental effects on the founding principles of the blockchain system as a few rich nodes are only able to perform the mining jobs. To address this challenge, Ethereum has proposed the "lightweight clients" [30] which only store the head of the blocks, and fetch the block from peer members when a block is needed[1].

[1]The scheme is described at project wiki https://github.com/ethereum/wiki/wiki/Light-client-protocol

However, these "lightweight clients" fail to have the features of the "full nodes"[2] and their mining process depends on the "full nodes".

To address the aforementioned problems of ever growing block storage requirement and limitations in the existing proposal, we propose a P2P based file system, called P2PBFS, to distribute the blocks into a P2P network. As a result, all blockchain participants can retrieve the blocks from the P2P network through the standard operating system file interface transparently.

The contributions of our work are summarised as follows:

1. *A new economic model for blockchain participants:* In the existing blockchain systems, the primary motivation to participate is to claim mining rewards. However, the nodes which fail to generate the blocks end up with nothing even though they contribute to providing computing and storage resources. Different from any existing blockchain economic model, in our framework, the participants, who provide the storage resources without participating in mining (e.g., Proof-of-Work, Proof-of-Stake, Proof-of-Storage), can also get rewards by offering the block query service to the blockchain system.

2. *An innovative distributed file system dedicated for block storage:* The features of P2PBFS are summarised as:

   - **New blockchain system architecture to resolve block duplication issue:** To the best of our knowledge, all existing blockchain systems require the "full nodes" to have a copy of all the blocks since genesis. P2PBFS decouples the function of block storage from block generation by migrating the function of block storage to some dedicated nodes, known as storage nodes. In P2PBFS, miners only need to store the block abstract. Compared with Bitcoin system, miners in P2PBFS saves 99.96% storage in blockchain maintenance.

     Since the miners are free from storing blocks, the issue of a large number of duplicate blocks can be avoided. Additionally, since role of storing the blockchain data is separated from the miner, resource-constrained mobile and IoT devices may take this advantage and be involved in the mining process, where heavy computing power is not needed (e.g. PoS), to increase the degree of decentralisation.

[2] "full nodes" are the nodes who store all the blocks in the blockchain.

- **Transparent block access/store with optimized query:** We achieve block access transparency by providing a P2PBFS driver. The driver offers the standard file operations (**open**, **write**, **read** and **close**). The block storage transparency is achieved by employing an operating system mutex-lock scheme. Additionally, Since P2PBFS does not record the sequence of the block, it is free from handing the "forks" in the blockchain systems. By caching storage node information in block abstract, miners can locate a block instantly without searching in the storage network which is dramatically faster than the general P2P storage solutions.
- **Scalability and availability:** We allow the storage node to join and leave P2PBFS freely. When the new nodes join the network, routing service running on the nodes will migrate the blocks to the new nodes according to the XOR distance between the node ID and the block hash. Block migration is transparent to the clients (The implementation of scalability is discussed in section 2.6.3.7). To achieve availability, we duplicate the blocks on to multiple nodes and this duplication is independent of the miners and much less than duplications per miner. Based on our experiment results (in the system consists of $20,000$ nodes and $200,000$ blocks), P2PBFS can still achieve 100% block query successful rate even though 35% of nodes are off-line.
- **Secure, practical, and comprehensive performance evaluations:** One of our system design targets is to have a good tolerance under different kinds of attacks (e.g., grinding attack [16] and Sybil attack [1]). We discuss the possibility of launching different types of attacks toward our system in Section 2.7 to show the robustness of our system. The details of our system performance evaluation are provided in Section 2.8.
- **Profit-driven decentralized file system:** Compared with the solution of storing the blocks in the existing cloud file system (e.g., Dropbox and Google Drive), our scheme does not depend on any third party to guarantee the service quality. Additionally, each participant who provides storage can claim fees from the data they provided which is irrelevant to the mining fees.

The feasibility of P2PBFS is based on the following four facts. (1) With the development of the high bandwidth network infrastructure, the impact of the network delay for retrieving a few MB data has become a minor factor to affect the service quality. (2) The development of storage technology makes it possible to provide a large storage solution with a relatively low cost (e.g., the Network-attached Storage). (3) The unspent transaction output (UTXO) pool scheme [31] allows the miners to ver-

ify the validity of the transactions without communicating with P2PBFS frequently. (4) The simplified payment verification (SPV) scheme [13] allows end-users (e.g., the cryptocurrency wallets) to confirm the transactions without inspecting all the related blocks[3].

2.3 Related work

2.3.1 Blockchain-based File Systems

The idea of proof-of-storage has been applied in mining for some blockchain systems such as FileCoin [32] and Chia [33]. Some of these systems take advantage of blockchain to store general data while our P2PBFS is dedicated to store blockchain blocks for different blockchain platforms. Our work is irrelevant to the research that taking advantage of blockchain to store data distributively. Additionally, in FileCoin and Chia, proof-of-storage is applied in mining while P2PBFS is not involved in any mining process and storage node gets rewards by offering the blocks.

2.3.2 LookUp Scheme in P2P Network

One of the key issue in all the P2P based distributed network is resource lookup. One simple and effective approach is to distribute the resources into a distributed hash table (DHT). Data items are stored in this DHT and found by specifying a unique key for that data. The P2P algorithms like CAN [34], Chord [35], Kademlia [36], Pastry [37], Tapestry [38], and Viceroy [39] are all structured DHT based ones.

In P2PBFS, we applied XOR "distance" to locate a resource in the network, and similar to Kademlia, thus, we avoid the potential attacks which exists in the Chord network that given all nodes address on the ring, the attacker can predict the routing path.

2.3.3 P2P File System

BitTorrent [40] is one of the most popular P2P file systems [41]. It relies on other (global) components, such as websites, for finding files and resources. However, one of the disadvantages is that it depends on a third party channel to publish the resources, and more

[3]Bitcoin paper and the BIP-0037 https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki

seriously, there lacks of incentives for the nodes to provide resources while they need to offer a reasonable bandwidth to benefit other peers who download the resource from them.

IPFS [42] is a Distributed Hash Table based hypermedia protocol which claims to make the web faster, safer and more open. It takes advantage of Merkle tree, GIT version control system, IPLD [43] to locate and update the unique resources efficiently. However, IPFS is too heavy to be applied in storing the blockchain data as the schemes of data updating is not needed. Since P2PBFS is dedicated for storing the blocks, it is optimized with blockchain application scenario and is more light-weighted and efficient. In P2PBFS, clients cache the storage nodes information to avoid searching the blocks in the whole P2P network. Additionally, IPFS is not optimized for the scenario that **read** is more common than **write** in the design.

Compared with general P2P storage system, P2PBFS is optimized for the blockchain storage scenario in the following aspects: 1. Since it is possible for different client writes the same block at the same time, we avoid the atom write operation. 2. Since no block updates function is needed in P2PBFS, we use dictionary structure to store the block with fixed position in the dictionary to achieve constant block query time. 3. For the miners, we cache the frequently visited storage nodes to avoid the search for these nodes in the network.

2.4 Architecture and New Economic Model

We decouple the block storage from the mining. As shown in Fig. 2.1(b), we divide the blockchain system into four layers. The bottom layer is the hardware layer which consists of the storage nodes. The function of the hardware layer is to offer the physical machines to store the blocks. We allow any entities that can provide enough storage space to join the network (it could be X86 servers or IoT devices). We also allow the nodes to join and leave the file system anytime they wish. The second layer is our blockchain file system, P2PBFS, which accepts the API calls from the miners or the broker service. P2PBFS layer collaborates with the operating system to hide the details of the storage service from the view of the mining service and the end-user broker service. It also maintains the data structure to store/query the blocks in the infrastructure layer more efficiently. The third layer is a service layer consisting of blockchain mining service and end-user broker service. Miners participate in the mining service to generate new blocks, and the end-user broker service provides the query and verification services for the end-users such as the blockchain wallet running in smartphones. The top layer is the blockchain API interface, which interacts with blockchain applications.

We also define three different types of nodes in the blockchain system. The first type is the end-user node, denoted as $\mathcal{N}_u$, which submits the transactions to the system; The second type is the miners, denoted as $\mathcal{N}_m$, which proposes new blocks in the system, they are the clients of the P2PBFS which store/query blocks from P2PBFS. The last type is the storage nodes, denoted as $\mathcal{N}_s$, which store the blocks in their local storage. $\mathcal{N}_s$ are paid when they provide blocks to $\mathcal{N}_m$ or $\mathbf{N}_u$. They are not involved in block mining and claim no fees from any mining process.

P2PBFS brings a new economic pattern for blockchain systems. For the P2PBFS based blockchain systems, $\mathcal{N}_s$ can charge $\mathcal{N}_m$ a certain amount of fee for the blocks they offer.

2.5 P2PBFS Design Overview

2.5.1 Assumptions and Scope

The following principals and assumptions guide the design of P2PBFS:

- We focus on how to provide block storage service effectively. The research issues related to mining and block transaction fee optimization are beyond our research scope.
- It is the fact that the majority of blockchain systems have the block size of a few MB. Reads is more common than Writes. Since blockchain is an append-only system, our file system does not support update operation.
- The system is designed for the scenario where participants are heterogeneous and can join and leave the system at any time.
- P2PBFS is dedicated to providing a service to support the functionalities of blockchain systems. It does not fully support standard file system APIs such as POSIX [44].
- There should be at least two storage nodes available to support the functionalities of P2PBFS.
- We assume that there are a few proportions of miners that query the same storage node at the same time. For the PoW based blockchain, the possibility that the miners generate identical blocks at the same time is negligible. For the consensus protocol which nominates a few miners to be responsible for block generation for each round (e.g., PoX or PBFT), the total number of block query requests is limited.

- We focus on how to provide efficient block querying and storing services. The internal storage architecture in each storage node and its reliability is beyond our research scope.

2.5.2 Interfaces

In this section, we describe how P2PBFS works in sync with the operating system. To make the discussion concise, we take the Linux operating system as an example to show how it works.

On the P2PBFS client side, the P2PBFS driver supports the standard file system API (e.g. open, write, close) calls. From the miners' point of view shown in Fig. 2.2(a), it calls the Linux virtual file system (VFS) with the standard POSIX APIs. The request is then sent to the file system in user space (FUSE) module. FUSE then sends the system calls to the P2PBFS driver. P2PBFS invokes low-level functions like LookupBlock, LookupNode to reply the calling from file system API (e.g., open, write). In response, the P2PBFS storage node locates the specific resource and sends it back to the miner as a result of the API call. On the storage node, the requests from the client are handled by performing store/query operation for the specific block from their own storage through the VFS.

In P2PBFS, we implement six APIs to support the interactions with the blockchain storage system.

- open(blockID): This allocates a file handler denoted as fd associated to the given block, and returns the file handle to the client. The input of open is the block hash value blockID.
- write(fd, block): This stores a block into the blockchain file system. The input of write is the file handler fd returned by open and the block that needs to be written. The returned value of write is the status flag indicating whether this block is stored in the system successfully and the list of storage nodes, where this block is stored.
- read(fd): The input of read is the file handler fd and it returns the block from the P2PBFS. It invokes VerifyBlock to avoid the storage node from returning the tampered blocks.
- ioctl(fd, opt, data): This interface allows the end-user to call the special functions that are provided by our file system (e.g., VerifyBlock or RandomWalk). The parameter opt indicates what kind of function the end-user implicate to call and the parameter data is employed to transfer information between end-user and P2PBFS.

- close(fd): This unlinks the file handler fd from P2PBFS.

To support the aforementioned APIs, we implement the following functions in P2PBFS: LookupNode, LookupBlock, VerifyBlock, StoreBlock and RandomWalk. The details of these functions are explained in section 2.6.2.

- LookupNode: The input of the function is the storage node ID and the output is the target node or a list of k_1 storage nodes that may know this target node, where k_1 is a file system global parameter which indicates the number of nodes that may know the target node.
- LookupBlock: The input of this function is the ID of the block, and the output is a set of $k2$ nodes which holds this block or may know the node that holds this block. k_2 is a file system global parameter which indicates the number of nodes may know the target block.
- VerifyBlock: The input of this function is the hash value of the blocks retrieved from the storage node and the block abstract chain. The output is the flag telling whether this block is tampered.
- StoreBlock: The input of the function is the block itself and the output is k_3 storage nodes in which this block may be put on. The file system parameter k_3 indicates how many duplicates we should have for a given block. A larger k_3 provides better block accessibility while increasing the demand for storage.
- RandomWalk: The input of this function is a storage node ID. The output is a set of blocks.

We next discuss how P2PBFS interacts with the blockchain systems. To make the discussion concise, we assume that the blockchain system is a PoW based blockchain.

1. Based on the unspent transaction data, miners verify the validity of the transactions and try to propose a new block containing a set of valid transactions.
2. The miner who generates a new block appends the block abstract to its local block abstract chain.

3. Instead of storing the block locally, it invokes API open provided by blockchain file system driver to obtain the file handler of this new block, then calls API write to store the block into the blockchain file system. The miner who proposes the new block calls the API ioctl to ensure the block can be retrieved from k_3 storage node. Finally, it broadcast the new block to the miners' network.

4. For the rest of the miners, they verify the validity of the block once it is received and store the block abstract in their local block abstract chain.

5. When the miners want to query a block, it calls read to retrieve the block from P2PBFS. read invokes the lower-function verifyBlock to ensure the block has not been tampered by the storage nodes.

For a given block, we cache the storage nodes which may hold the block in block abstract; thus, when API open tries to link a block with a file handler, P2PBFS can locate the storage node by searching in the block abstract rather than calling LookupBlock function to search in the whole network. The details of the cache scheme are discussed in Section 2.6.1.2.

For the storage nodes, they need to maintain node buckets to maintain the information of peers' storage nodes. Additionally, they need to maintain a dictionary to help them to locate the block within a constant time. The details of the data structure on the storage node are discussed in section 2.6.1.

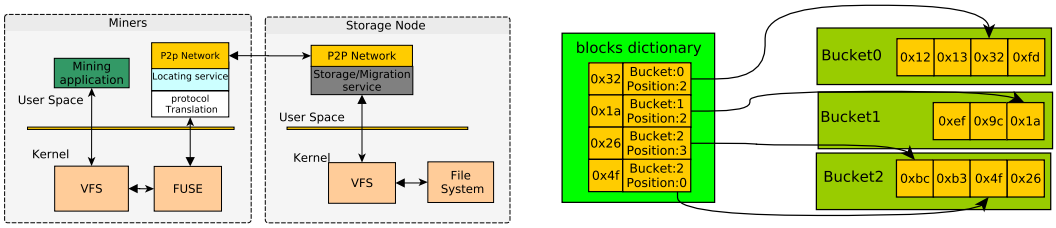
2.6 P2PBFS Internals

In this section, we first explain the data structure of the blockchain storage node and the miners. Then we explain the implementation of the interface in detail. We assume that the block hash value and the node ID are both 512 bits long.

2.6.1 Data Structures and Routing Table

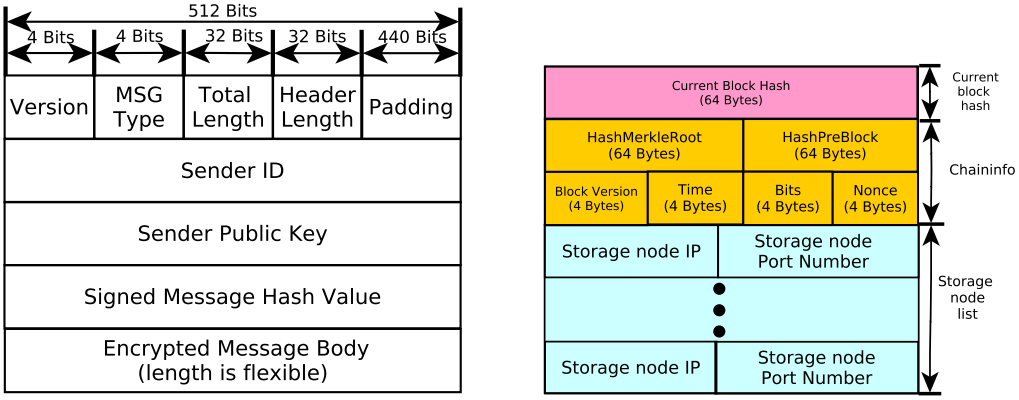
2.6.1.1 Storage Nodes data Structure

For a storage node, it maintains a block dictionary which maps the block hash value to the address where the block is stored. As it is shown in Fig. 2.2(b), to locate a block with the hash value of 0x32, the storage node looks up the key value of 0x32 in the dictionary



(a) Integration with Linux Operating System. (b) Data structure on Storage node.

Figure 2.2: Integration with Linux and Data structure on storage node.



(a) Message structure. (b) Block abstract structure.

Figure 2.3: Message structure and Block abstract structure.

and finally locates the specific block in the array. When a new block appears, it is mapped into one of the buckets and an entry is created in the dictionary.

2.6.1.2 Miners' Data Structure

The miners, $\mathcal{N}_m$, need to store the abstracts of the blocks to verify the integrity of the blocks that received from the storage nodes. We define the block abstract, which includes at least **Current block hash**, **Chaininfo**, and **Storage node list**. The reference block abstract structure is shown in Fig. 3.2.

- **Current block hash:** This segment is to help the miners to verify the validity of the blocks that retrieved from the P2PBFS.
- **Chaininfo:** It is the critical part of the block abstract, similar to Bitcoin block, it contains hashPrevBlock, hashMerkleRoot, and TimeStamp. For the PoW based blockchain, it should also contain the segments to store the difficult Bits and the Nonce. The functions of these variables are similar to that in Bitcoin blockchain systems. All the data in **Chaininfo** is read-only.
- **Storage node list:** This is to cache a number of nodes that may potentially hold this block. Thus, when the miner needs to lookup for a block, it firstly searches from the peers in the peer list to avoid calling LookupBlock. The storage list is updated every time when this block is visited since the block may migrates to other storage nodes due to the variation of the network.

2.6.1.3 Node Routing Table

The routing table contains the object $< ip, portnumber, nodeID >$ for each peer, known as the address. With the ip address and the $portnumber$ of the peer, a storage node can easily connect to a given service running on the peers. To store the address in a hierarchical way, for a given node j, we create n buckets where n is the bit length of the nodes' address. For different peer nodes, node j allocates them into different buckets according to the longest common prefixes (LCP) the peer nodes share with node j. For the bucket-0, it stores the nodes that have the address which the most significant bit is different from node j. For the nodes with 5 common prefix in addresses, they should be stored in the fifth-bucket, bucket-5. For the 512-bit address space, half of the total nodes belong to the bucket-0 and $2^{\frac{512}{k+1}}$ nodes belongs to bucket-k.

One can observe that bucket-511 has only 2 nodes while bucket-0 could have 2^{512} nodes. To address this un-balanced node distribution in buckets, we set the space in each bucket as a dynamic value to allow some buckets have more space to contain more peer nodes. Let L be the bit length of the $nodeID$; For the bucket from bucket-500 to bucket-511, we set the size of bucket-x as 2^{511-x}. For the rest of the buckets, we set them to have fix sizes, which can contain 1024 peer storage nodes.

2.6.2 Low-Level Operations

In this section, we describe the algorithms of low-level operations in P2PBFS including LookupNode, LookupBlock, StoreBlock, and RandomWalk.

- LookupNode: When a node a (a miner or a storage node) locates another storage node denoted as node b, it invokes the function LookupNode() with node b's ID as input. LookupNode() first calculates the LCP with node b. We denote the Exclusive OR (XOR) value between node a and node b as z. It then looks up the nodes in bucket-z. In the best scenario, node a can locate the node b in the first attempt. Otherwise, it asks γ closest nodes to node b in bucket-z to help it find the target peer simultaneously. If these peers know the address of the target peer, they will return it; otherwise, they will return k_1 nodes that are closest to node b to the best of their knowledge. For each lookup operation, the node a obtains the nodes that are closer to the node b and appends these nodes to the search queue. Then, node a randomly picks up γ nodes from the search queue and repeats the aforementioned process. The lookup process terminates when the node b's address is found or there is no new nodes, which potentially know node b appended to the search queue [4]. In the worst case, node a has to search $\lceil \log_2 n \rceil$ times to locate the target peer where n is the number of storage nodes in the network. In practice, the time consumption of search operations is much less than the worst case since we store more than one peer in a bucket and we involve multiple nodes in search operation simultaneously.

- LookupBlock: To find a block denoted as B in the network, miner c firstly contacts the storage nodes that stored in the **Storage node list** in this block's abstract to retrieve the block. If none of these storage nodes can offer the requested block, the block search operation is performed which can be divided into two steps. In step I, miner c communicates with one of the storage nodes denoted as d to ask for the block B. If node d does not have the block, it calculates the LCP between its ID

[4] We assume that the number of the storage node is larger than k_1.

and the block B's hash value denoted as z. Node d then returns k_2 storage nodes that in its bucket-z that may know block B to miner c (if the number of nodes in this bucket is smaller than k_2, all the nodes in the bucket are returned). Miner c puts these returned nodes in the search queue G and repeats step I with γ nodes in queue G. Step I terminates when a(some) storage node(s) in G claim they have the block or no new storage node that potentially has block B appended to the queue G. In step II, miner c pays one of the storage nodes who has the block and retrieves the block from it.[5] c checks the hash value of the block against the block's abstract it holds to ensure the validity of the block. Node c updates the **Storage node list** with the nodes in G that can provide the valid block.

- StoreBlock: The operation of storing a block can also be divided into two stages. In step I, the miner m maps the block W hash value to the node ID space and then applies the same algorithm described in LookupNode to return the nodes that are currently top-k_3 closest to the block W. In step II, the node m sends the block to all the nodes in the set S, and the storage nodes in the set S will store the block locally and create a new entry in their block dictionary.

- VerifyBlock: To verify the integrity of a retrieved block, the miners calculate the hash value of the returned block and compare it against the block hash value stored on the block abstract. VerifyBlock avoids the possibility that the miner modify the block.

- RandomWalk: It is designed to help the miners to receive a batch of the blocks in a more efficient way. It starts from a storage node S and then ask node S to return the list of peer storage nodes in S's bucket denoted as F, and the miner randomly picks up some storage nodes from F and gets all the blocks from these selected nodes. Then, it replaces node S with a random node in F and repeats the aforementioned process. This function can be run recursively.

2.6.3 System Operations and Maintenance

In this section, we first introduce the message structure of P2PBFS. We then use the lower-level operations described in section 2.6.2 to implement the functions such as query/store the block from the $\mathcal{N}_s$'s point of view.

[5] The payment is a transaction, if the storage node provides an invalid block, this transaction is discarded by the miners. By checking the signature, other miners mark this storage node as dishonest. It may loose the chance to get profit from offering the blocks since miners do not trust and accept its block any more.

2.6.3.1 Protocol Message Structure

The file system message is shown in Fig. 2.3(a) which consists of two parts: message head and message body which is the encrypted message content. **Message head:** The structure of the message head consists of the following segments:

- **Version**: This represents the version number of the message.
- **Msg Type:** The message type enables the receiver to call the suitable function to handle it.
- **Total Length and Header Length:** Since the length of the content is flexible, we need to have the total length and header length to locate the message.
- **Sender ID:** This is the ID of the sender. It is also known as the $nodeID$. The ID is the hash value of the public key of the sender.
- **Sender Public Key:** This is the public key of the sender. It is applied to verify the correctness of the $nodeID$ to prevent the adversary from pretending to be another node (impersonate attacks [45]).
- **Signed Message Hash Value:** This segment is included to verify the integrity of the message body. It also guarantees that the message sender holds the public key, which is claimed in **Sender Public Key** segment.
- **Encrypted Message Body:** The message body could be either a list of the peers information or the block data.

Message type: In the message type segment, we support the following operations:

- Ping – This message is sent by the nodes to check whether the destination node is still active.
- Pong – This message is the response to the Ping message.
- FindNode – This message indicates the request of finding a specific node.
- FindNodeReply – This message indicates a list of the nodes that are closer to a given nodes.
- FindBlock – This indicates the request of finding a specific block.
- FindBlockReply – This could be either the a list of nodes that may hold the block or the target block itself.

2.6.3.2 Bootstrapping the File System Storage Node

In order to bootstrap the storage nodes, we should have at least two nodes whose addresses are publicly known. These two nodes are served as the bootstrap nodes in the network, which help the new nodes to update their buckets. Other nodes can also have the role of bootstrap nodes to help other storage nodes to join the network by publishing their address to the public.

The file system global parameters k_1, k_2, and k_3 should be agreed among all the file system participants. These parameters can be assigned with different values according to message version. Parameters like γ is a local parameter which allows different storage node to be customised. Larger γ may increase the systems working load as we invoke multiple threads to search for a block/node simultaneously, while it could decrease the block/node lookup waiting time.

2.6.3.3 Client Nodes Join the Blockchain Network

All the nodes that retrieve/store blocks from storage nodes are regarded as the clients. For the clients, they follow the similar process of joining the existing blockchain network (the detail of how a node joins the blockchain network is beyond our research scope). The P2PBFS driver should be loaded into the operating system.

2.6.3.4 Storage Nodes Join the Network

When a storage node joins the file system, it needs to connect to a set of the bootstrapping nodes. When the new node connects with the bootstrapping nodes, it sends a request LoopkupNode to the bootstrapping node with the NodeId of itself. In this way, the new node has the knowledge of the peer nodes that are closer to it.

2.6.3.5 Storage Nodes Leave the Network

When a node quits the network, it does not need to notify any peers. Since the blocks that the node holds are also stored on other peers. The system can still provide the service normally.

2.6.3.6 Address Bucket Update

To update their address bucket, they can "ping" each node in the buckets and evict the off-line node. They can also invoke LookupNode() with the ID of themselves to update the knowledge of the nodes that around them.

2.6.3.7 Block migration (Scalability)

To achieve scalability, P2PBFS provide a on-the-fly block migration and address buckets updating scheme to fit in with the variation of the storage network. When a node j checks the availability of its neighbours and finds a new neighbour denoted as node k, node j checks whether the blocks that share the same LCP with node k closer to node k or not. If the block is closer to the node k, node j sends the blocks to the node k and deletes it from node j's local storage. When the node k receives the block, node k checks whether its node ID is closer to the received block compared with node j or not, and decides whether to store or ignore the block.

2.6.3.8 Blockchain Construction

For a practical blockchain system, it should allow the new miners to construct the blockchain by visiting all the blocks since genesis. In P2PBFS based blockchain system, we suggest applying a hybrid strategy which combines storage node RandomWalk and LookupBlock. For the miner j, it firstly queries any block that is broadcast by other miners. It then, invokes LookupBlock to retrieve this block from the storage node S, furthermore, miner j retrieves all the blocks on the node S and invokes RandomWalk to get a batch of blocks. It constructs the chain according to the preblockhash stored in Chaininfo segment. When few new blocks can be reterieved by RandomWalk, miner j then applies LookupBlock to query for the missing blocks.

2.7 Security Discussion

In this section, we present a security discussion considering the possibility of launching different attacks toward our file system.

2.7.1 Impersonate Attack

For this attack, the adversary may create a node with the ID that is identical with one of the existing nodes in the network; thus, the communication to the victim node could be redirected to the adversary.

This attack is impossible in our system. Since the $nodeID$ is generated from the node's public key, the verifier can check whether the $nodeID$ matches with the node's public key. It is impossible for the adversary to provide a fake public key as the sender needs to sign on the message body with the private key that matches with the public key it claims. In conclusion, since the adversary cannot generate the private key based on the public key it carefully crafted, it is impossible for the adversary to impersonate other participates.

2.7.2 Grinding Attack

For the grinding attack, the adversary tries to generate a special set of $nodeID$'s that are close to each other thus he could ensure a proportion of blocks are stored on the nodes controlled by the adversary. However for the system with the $nodeID$ of 512 bits length, the possibility of finding 100 nodes that are closest to a given node is $\frac{100}{2^{512}}$ which is negligible. The huge $nodeID$ space also makes the brute force attack impossible.

2.7.3 Eclipse Attack

In this attack, the adversary eclipses a number of nodes from the rest of the network so that the compromised nodes cannot respond to any requests from the file system. Since our system allows one block to have multiple copies that are stored on different nodes, the cost of launching an effective attack is increased dramatically. According to our experiments in Fig. 2.4(b), in a $20,000$ nodes' system, even if 60% of the total nodes are eclipsed, the system can still respond to 99.40% of the requests.

The adversary may also firstly look up all the nodes that hold the block L and then eclipse the nodes which store the block L from the rest of the network. However since different nodes have different security levels, and they may work in different physical networks. The cost of eclipse the block L can be dramatically high, thus, make the attack towards a set of blocks impractical.

2.7.4 Nodes Collusion Attack

For the node collusion attack, the adversary tries to collude a set of nodes to disable the accessibility of a certain amount of nodes. However, this type of attack is also not feasible due to the following reasons: (1) The benefit of being honest overwhelms being dishonest. Since the majority of the storage nodes are profit-driven, they can claim fees continuously from offering the storage service. Once the service quality of the file system degrades, fewer users will use the blockchain file system and the income of offering the storage resource is also decreased. (2) The attack is of low efficiency. With $k_3 = 10$, the adversary needs to collude with at least 10 nodes to invalid a block. According to our experiment result in the 20,000 nodes system with 200,000 blocks, we found that with $\epsilon = 30$, the average blocks redundancy is 85% and each node stores 100 blocks; this mean for a group of nodes, 15 nodes are unique which account for 0.0075% of the total nodes. Roughly speaking, for our test system, even if 100 groups have colluded which includes $1,000$ nodes, it only affects 0.75% of the total blocks in the system.

2.7.5 Sybil Attack [1]

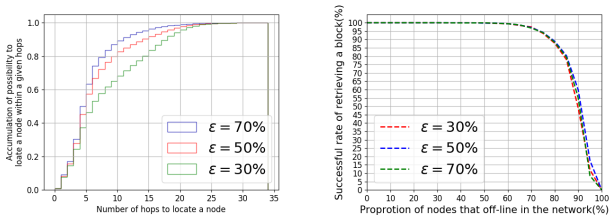
The adversary may take advantage of the scalability of the system to employ cloud resources and generate a huge amount of fake storage nodewithin a second and deploy in the blockchain file system. After waiting for a certain amount of time to allow the blocks to be migrated on to them, they reject all the block requests from the client.

According to our experiment result, for a system which has $20,000$ nodes and $200,000$ blocks, even if 60% of the node are compromised, the system can still respond to 99.40% of the requests. Additionally, we ask all the storage nodes to make a deposit in the blockchain before they are accepted by the system. In this case, compared with other participants, if the adversary dominates the system by making the majority of the deposit, he has no motivation to ruin the system and lost all his deposit.

2.8 System Evaluation

2.8.1 Experiments Setup

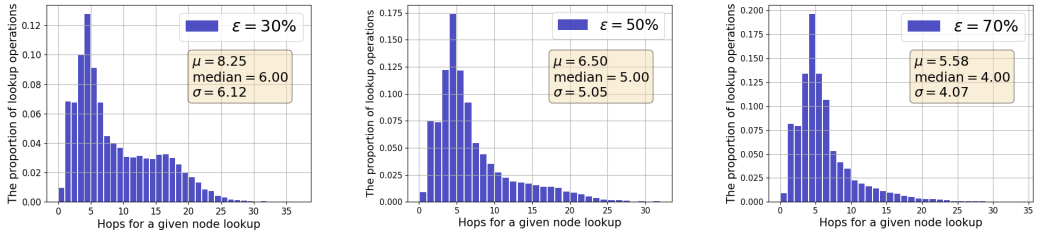
In this section, we evaluate the performance of P2PBFS in a 200,000 blocks blockchain with 20,000 nodes. The nodes we mentioned in this section are storage nodes by default.



(a) The accumulation of the possibility to locate a node within a given hops.

(b) Successful rate of retrieving blocks.

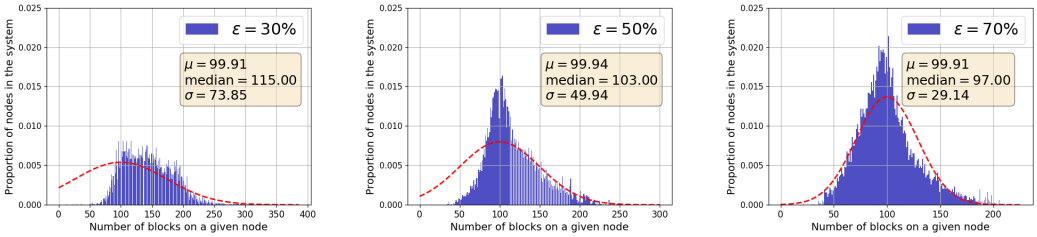
Figure 2.4: CDF of locating a node and successful rate of retrieving blocks.



(a) Hops for node lookup.

(b) Hops for node lookup.

(c) Hops for node lookup.

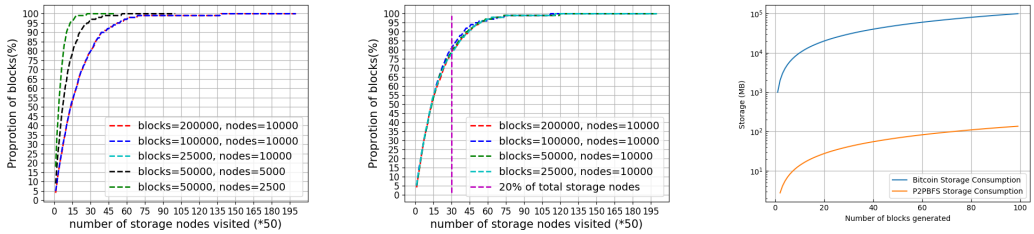


(d) block distribution.

(e) block distribution.

(f) block distribution.

Figure 2.5: Nodes lookup and blocks distribution with different ϵ.



(a) RandomWalk effiency ($\epsilon = 50\%$). (b) RandomWalk effiency ($\epsilon = 70\%$). (c) Storage consumption on P2PBFS and Bitcoin.

Figure 2.6: RandomWalk coverage with different ϵ and system storage consumption.

We set $k_1 = 20$, $k_2 = 20$, $k_3 = 10$, and $\gamma = 3$ for all nodes. In P2PBFS, the ideal number of blocks for each node is denoted by B_{idea} and is calculated as:

$$B_{idea} = \frac{Num_{block} * k_3}{Num_{node}},$$

where Num_{block} and Num_{node} are the number of blocks and nodes in the system, respectively.

In reality, it is impossible for any node to communicate with the rest of the nodes in the network. For a node, we define the proportion of peer nodes it once communicated with as the node's communication coverage. We denote such quantity by ϵ. In our evaluation, we set $\epsilon = 70\%$, $\epsilon = 50\%$, and $\epsilon = 30\%$, respectively.

We evaluate: (I) the number of hops that are needed to locate a storage node in the system, (We avoid LookupBlock evaluation since it is similar to LookupNode) (II) the distribution of blocks in the file system, (III) the success rate of locating blocks with the variation of the number of online nodes, and (IV) the efficiency of RandomWalk to retrieve blocks from P2PBFS.

According to our evaluation tasks, we run the P2PBFS simulator on 8 core i7 CPU with 32GB memory. The simulator runs 20,000 nodes structure and updates the nodes information according to the exchange of the peer information with other nodes. Multiple threads are adopted in the simulator to simulate the parallel communication in the P2P network.

2.8.2 Evaluation of Finding a Node

The experiment results of the hops that are needed for locating a node with different ϵ are shown in Fig. 2.5(a), Fig. 2.5(b) and Fig. 2.5(c), respectively. We define β as the ratio between the number of lookup operations that are needed for a given hop and the total number of lookup operations. It can be observed that the lookup operations which takes 5 hops have the highest β, which are 13.0%, 17.5%, and 20.0% for different chosen ϵ values. The amount of β drops dramatically with the increase of hops for a given LookupNode operation. It can also be observed that with the increase of ϵ, the average of number of hops drops, which indiates the more nodes communicate with each other, the less hops to locate a block(in Fig. 2.5(a), the average of hops is 8.25 while it is 5.58 in Fig. 2.5(c).).

In Fig.2.4(a), we demonstrate the accumulation of possibility to locate a node within a given number of hops. It can be observed that for all different ϵ, half of the node lookup operations can be finished within 7 hops (For $\epsilon = 70\%$, it is 74%, for $\epsilon = 50\%$, it is 67% and for $\epsilon = 30\%$, it is 53%). When $\epsilon = 50\%$, 80% of the node lookup operations can be finished within 10 hops, whereas the value is 65% for $\epsilon = 30\%$.

It can be concluded that fewer hops can be achieved by allowing nodes to exchange their peer nodes' address more frequently. Our system achieves a high node locating efficiency as 74% of LookupNode operation can be completed within 7 hops.

2.8.3 Blocks Distribution

In this experiment, we discuss how the blocks are distributed among different nodes in the file system. The experiment results are shown in Fig. 2.5(d), Fig. 2.5(e), and Fig. 2.5(f), respectively. For the whole network, about 6.8%, 14%, and 32% of nodes store none of the blocks with the $\epsilon = 70\%$, $\epsilon = 50\%$, and $\epsilon = 30\%$, respectively. It demonstrates that with the larger ϵ, the more storage nodes can be involved in the blocks storage. The lower standard deviation σ indicates the blocks are distributed more closer to average number of blocks on a node. The higher blocks are distributed evenly, the higher efficiency for the LookupBlock and RandomWalk. It can be concluded that with more storage nodes exchange the peer address in the network, the blocks are distributed more evenly in the network.

2.8.4 File System Resilience Evaluation

In this experiment, we evaluate how the success rate of retrieving blocks varies with the growth of the proportion of online nodes. We conduct three evaluations with $\epsilon = 30$,

$\epsilon = 50$, and $\epsilon = 70$ respectively. We conduct retrieving a block operation $20,000$ times in a system which has $200,000$ blocks and consists of $20,000$ nodes. It is shown in Fig. 2.4(b), with the growth of proportion of off-line nodes, the success rate of retrieving blocks drops slightly before 70% of the nodes are offline. It is shown that even 35% of the nodes are off-line, the system can still achieve 100% success rate, which means all the blocks can be located in the system. The system can still achieve 97.12% successful rate even if 70% of the nodes are off-line. It also demonstrates that system can achieve same level of resilience with different ϵ's. The high system resilience is achieved by duplicating every blocks on $k_3 = 10$ nodes.

2.8.5 RandomWalk Evaluation

In this section, we evaluate the efficiency of RandomWalk to see how many block can be retrieved by visiting a certain number of nodes. We evaluate the system with $\epsilon = 50\%$ and $\epsilon = 70\%$ respectively. For each scenario, we generate different number of blocks and evaluate the increase of the proportion of blocks we retrieved while considering $10,000$ storage nodes. It can be concluded from Fig. 2.6(a) and Fig. 2.6(b) that for different ϵ, the growth trend of the blocks that are retrieved in the system is similar with the increase in visited nodes. It can also be concluded that the system has a similar result with different number of blocks in the system. For the first 750 nodes, the number of blocks we retrieved in RandomWalk increases dramatically (shown in magenta dash line in Fig. 2.6(a)). Thereafter, the proportion of retrieved blocks increases slightly, which indicates the drop of the RandomWalk efficiency. Fig. 2.6(a) also indicates that the number of blocks in the system makes little impact on the RandomWalk efficiency. Fig. 2.6(b) demonstrates that RandomWalk can retrieve similar proportion of blocks with a similar proportion of storage nodes that visited regardless of the total number of nodes and blocks in P2PBFS. For instance, as shown in Fig. 2.6(b), to achieve 80% of the blocks in the system, miners need to visit $1,500$ storage nodes (15% of total nodes), 800 storage nodes (16% of total nodes) and 400 storage nodes (16% of total nodes) for the P2PBFS consists of $10,000$ nodes, $5,000$ nodes, and $2,500$ nodes, respectively.

2.8.6 Storage Consumption Comparison Between Bitcoin and P2PBFS

In this section, we analysis the storage consumption between P2PBFS and Bitcoin. To make the discussion concise, we assume each block consumes $B_{size} = 1MB$ and analysis

the growth of the storage consumption from the view of miners and the whole blockchain system respectively.

Compared with Bitcoin, miners in P2PBFS based blockchain system does not need to store the whole block but the block abstract. We assume each block abstract contains 10 storage nodes. The current block hash consumes 64 bytes, Chaininfo section consumes 148 bytes and the rest storage node list section consumes 180 bytes. In total, the block abstract denoted as B_a consumes 392 bytes. Compared with 1MB Bitcoin block, for the miners, P2PBFS saves about 99.96% of the total storage.

In P2PBFS, the storage cost for the system to store a block consist of two parts, one is the block abstract on the miners, and the other is the storage that consumes on the storage node to store the block and block directory. We assume that the system has P_m miners and P_s storage nodes. We ignore the storage of the blockchain directory 2.2(b), since 12MB dictionary can store 1 million blocks. Bitcoin system consumes $B_{size} * P_m$ MB while P2PBFS consumes $(B_{size} * P_s) * k_3 + B_a * P_m$ bytes. For a system that has $P_m = 1000$, $P_s = 1000$ and $k_3 = 10$, the storage growth with the increasing of the blocks is given in Fig 2.6(c). It shows roughly that for the whole blockchain system, bitcoin spends about 10GB to store 60 block while P2PBFS only spends 100MB.

Experiments results demonstrate that our P2PBFS can achieve high efficiency in LookupNode and the RandomWalk. We also demonstrate that P2PBFS achieves a high resilience by duplicate blocks on k_3 different nodes. The efficiency of the system can be improved by allowing storage nodes exchange the peer nodes' address frequently. We also demonstrate that P2PBFS achieve the design goal of saving more storage compared with other blockchain systems.

2.9 Conclusion

In this paper, we also propose an innovative distributed blockchain file system (P2PBFS) to address the increasing demand for the block storage which also benefit the participate who offer their storage. Through the system evaluation, we demonstrated that our file system can store/retrieve a block with reasonable time and have a good system resilience, when some participants are faulty.

Chapter 3

Proof-of-QoS: QoS Based Blockchain Consensus Protocol [2]

3.1 Abstract

The consensus protocol is the foundation of all blockchain systems. Existing consensus protocols like PoW consume a vast amount of energy. However, they are severely limited to transaction throughput. Consensus protocols like PoS have been proposed to address this challenge. However, these protocols have compromised the fairness by discouraging the "poorer" participants and allowing "richest" stakeholders to have full control over the generation of blocks. Towards meeting these conflicting requirements on throughput and fairness, we propose a blockchain consensus protocol based on the QoS. In our PoQ protocol, the entire network is divided into small regions. Each region nominates a node based on its QoS. A deterministic BFT consensus is then run among all nominated nodes. PoQ aims to achieve a very high transaction throughput as a permissionless protocol and provides a fairer environment for participants. Our experimental results show that PoQ can achieve 9.7K transactions per second (TPS) for a network of 12 regions.

3.2 Introduction

Blockchain transits the trust from one or a few centralized nodes to the decentralized system consisting of many nodes. The trust-free feature brings great advantages to the security and privacy of sensitive applications such as anonymous payment systems like

Bitcoin [13] and Ethereum [46]. The fundamental of all blockchain systems is the consensus protocol which synchronizes the data among all the participants. The goal of consensus protocol is to provide a consistent view of the data from the user's perspective. One of the core considerations for all consensus protocols is the "double-spending" problem, where a blockchain participant can consume the same resource more than once [47]. In the last 40 years, the problem of consensus among distributed nodes has been studied by the distributed system community extensively; research like [48, 49] developed the robust and practical protocols that can tolerate faulty and malicious nodes in a closed system. However, achieving such an agreement in a permissionless system, where anyone can join and leave at any time of the protocol can be difficult. Traditional consensus protocols, which depend on a fraction of honest nodes, do not work as the adversary can launch a Sybil attack by creating an arbitrary number of "pseudonyms" nodes [50].

Bitcoin addresses the "double-spending" issue by using the PoW-based consensus protocol [13] to elect the leader for block generation in a probabilistic order. However, the transaction throughput of PoW is poor [51] and a vast amount of energy is wasted [52]. In addition, the winner-takes-all scheme discourages the nodes, who do not have the advanced hardware to win in the block proposal process, known as mining. Although BFT [48] based protocols have a high transaction throughput, the number of participants for reaching an agreement is restricted, and it is designed for the closed systems. Hyperledger Fabric [8] takes advantage of the traditional BFT protocol to pre-define a set of blockchain participants to achieve consensus with a high transaction throughput. PoS [23, 53, 16] is another popular protocol in which the probability of creating a block depends on the ratio of asset the nodes hold in the system. In effect, this yields a self-referential discipline — the maintenance of the blockchain relies on the stakeholders who assign work and reward to themselves [16]. This can be unfair to the nodes with a small number of assets as they have to trust the "richest" nodes in the network.

We propose a new blockchain consensus protocol in this paper, called PoQ. It is a hybrid protocol which selects nodes using Proof-of-QoS (we take the deposit ratio, error rate, activity rate and reference factor as the QoS factors of a given node) for running a BFT-style consensus. In PoQ, the order of blocks is deterministic; which avoids the probability of creating forks (so the possibility of double spending the same resources) in the chain. We segregate the whole blockchain network into autonomous regions. For each block proposal round, based on the QoS score, each region nominates a node to propose the transactions to be put on the chain. These selected nodes form a committee and will reach a consensus on a block using classic deterministic Byzantine agreement protocols, such as PBFT protocol [48]. Finally, all the nodes clone the new block and append it to their chain. As a result, the system reaches a consensus on the order of the transactions.

Honest nodes are motivated to improve their QoS to gain the probability to be elected as the committee member in this region in order to claim a reward from the proposed blocks.

The unique contribution of our consensus protocol is we not only take "stake" (*e.g.,* deposit) as the criterion for participating in block generation but also take nodes' overall real-time performance (*e.g.,* activity rate, error rate) into consideration. Our real-time QoS evaluation scheme encourages the nodes competing for providing better QoS while despairing the nodes which want to dominate the block generation only by making a vast amount of deposit. Other features of PoQ are summarized as follows:

- **Openness:** PoQ achieves openness by segregating the whole network into autonomous regions. New nodes are free to join one of the regions they prefer. When a node joins a region, peer nodes check whether this node's deposit has already existed in the MGB to avoid a node joining multi-regions with the same identity (node address).

- **Fairness and energy-saving:** From the perspective of the nodes, PoQ provides a fair environment by selecting a representative of a region based on its QoS. Activity rate factor is introduced to avoid some nodes dominating the block generation by making a huge amount of deposit. A random function is applied to allow nodes with a similar QoS level to have an equal chance to be the representative. The online QoS update scheme encourages the nodes to comply with the protocol rules; otherwise, they will not only lose their QoS score but also lose their deposit as a punishment. It is an energy-saving protocol as nodes do not waste a tremendous amount of energy for competing for the block proposal.

- **Resilience and Robust:** Our protocol offers the liveness and safety properties, provided that at most $\lfloor \frac{n-1}{3} \rfloor$ out of n regions are simultaneously faulty. In PoQ, compromising the node nomination is more difficult and expensive — the nodes are nominated based on their QoS. If a node is deemed malicious, its nomination frequency drops dramatically as shown in Fig.3.3(c). It is impractical for a malicious node to misbehave in the system while not being caught by the peer nodes for a long time. Since the node nomination is not based on the asset or computing power, and no one can increase its nomination frequency by occupying a large number of resources in the network. Instead, PoQ encourages the node to do the mining by providing high QoS.

- **Performance:** High throughput is achieved by allowing the regions' representative nodes to construct the PBFT network to propose, and accept blocks among all regions. Through the experiments, we show that the system creates a fair environment for all the participants and it achieves a very high transaction throughput of up

to 9.7K transactions per second (TPS). Considering Paypal and VISA, which handles hundreds of TPS and 2K TPS respectively [18], the proposed system is suitable for practical use. We also implement PoQ on BFT-SMaRt [19] to demonstrate the feasibility of building PoQ on the existing BFT protocols.

The remainder of the paper is organized as follows. We compare PoQ with existing consensus protocols in Section 3.2.1. The PoQ protocol and algorithms are described in Section 3.3. We discuss the threat model and analyze the security of the protocol in Section 3.6. The protocol implementation and evaluation are described in Section 3.7. We conclude our work in Section 3.8 and discuss future research directions.

3.2.1 Related Work

The blockchain is an append-only ledger [9]. It can be regarded as a distributed database which stores the transactions in a special tree structure known as Merkle tree [10]. If a transaction passes the validity check, it is included in a candidate block together with other transactions. Nodes in the blockchain system collaborate on the sequence of the blocks that can be on the chain (consensus protocol). All the participants in the system should have an identical view of the sequence of the blocks. However, in a practical scenario, due to the network latency of transaction propagation and malicious or faulty nodes, nodes may end up with different views of the block sequences (branches in the blockchain system). The consequence of the inconsistent view is the possibility of double-spending. For each block generation interval called an epoch or a round, the consensus protocol nominates one node as a leader to propose the block while the rest agrees on the proposed block by appending it to their chain. For the PoW based protocol, the node nomination is based on solving a crypto puzzle. For the PBFT based protocol, the node nomination is based on the negotiation among participants. While the proof-of-X (PoX) protocol nominates a leader based on the resources each node has (the resources can be the deposit that nodes made or the storage that the nodes provide). Hybrid protocols take advantages of different types of consensus systems by combining them in novel ways.

- **Proof-of-Work (PoW):** As initially proposed and used in Bitcoin, PoW-based consensus (a.k.a. Nakamoto consensus) is the first permissionless consensus protocol where anyone can join and leave at any time.

  To propose a block, nodes, also known as miners, should solve a crypto puzzle by finding a specific nonce which matches with the generated block. It is possible to have multiple valid blocks proposed concurrently, which forms a fork of the chain. In this

case, the miners work on the first block (of the same height) that they received and regard the longest branch as authoritative. To confirm the authoritative of a block in the chain, the applications must wait for several blocks (6 blocks are recommended in Bitcoin) to be appended to it. For time-sensitive applications, this latency may not be acceptable. Another issue is that a winner-takes-all scheme encourages the miners to upgrade their hardware to increase their chances to be rewarded and thus creating a destructive competition for all the nodes. Moreover, wasted energy is another big concern for PoW-based systems.

- **Byzantine agreement:** BFT consensus is designed to replicate a service in a small group of pre-defined nodes. Inspired by this idea, Hyperledger Fabric[1] applies a variant of PBFT [48] to make all the participants have the same order of the transactions. A membership server is needed to control the accessibility of the blockchain to avoid a Sybil attack. Although BFT is a good solution for permissioned blockchain systems, it is not suitable for permissionless ones.

- **Proof-of-stake:** Proof-of-Stake (PoS) protocols [54] are proposed to eliminate the issues of energy wasting. In particular, PoS protocols randomly choose a subset of system participants to reach consensus, rather than requiring heavy computational work. However, a pure PoS system has been criticised as the safety mainly depends on the correctness of rich participants, which may not be desired.

- **Delegated-proof-of-stake:** Delegated-proof-of-stake (DPoS) is employed in EOS[2] to provide a democratic and vote-based consensus. In DPoS, a small number of nodes, called *producer* are elected to run the consensus. It is possible for an adversary to vote for a dedicated producer by launching a ballot-buying attack.

- **Hybrid consensus protocol:** Roughly speaking, hybrid consensus protocols combine PoW and BFT protocols, where PoW is used to select voters to run BFT schemes. Hence, Hybrid consensus protocols achieve a high transaction throughput in a permissionless environment. In particular, RepuCoin [55] is the most relevant protocol to PoQ. With RepuCoin, each miner has a reputation, and top reputed miners will be dynamically selected to run consensus. The reputation of each miner is computed based on both the total amount of valid work a miner has contributed to the system, and the regularity of that work in the system. However, it consumes a huge amount of energy.

[1] https://www.hyperledger.org/
[2] https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

Blockchain Systems	Openness	Throughput	Safety	Penalty	Type of consensus	Resource/stake
Bitcoin [13] (PoW)	Flexible for nodes to join and leave	Poor	51% of the computing power to attack the system	No penalty applied when misbehaving	Probabilistic	computing power
Hyperledger fabric [8] (PBFT)	Can only support a few numbers of nodes	Excellent	More than 1/3 of the nodes need to be compromised	No penalty applied when misbehaving	Deterministic	N/A
EoS [56] (DPoS)	Flexible for nodes to join and leave	Excellent	More than 1/3 of the committee need to be compromised	Lose their deposit when misbehaving	Deterministic	Deposit
PoQ (Hybrid)	Flexible for nodes to join and leave	Excellent	Node nomination process needs to be compromised to ensure more than 1/3 of committee members are malicious	Lose the chance to propose blocks and deposit as well	Deterministic	Overall performance

Table 3.1: Comparison with some popular blockchain protocols (platforms)

PoQ follows the direction of a hybrid system by combining proof-of-stake protocol with BFT protocol thus eliminates the problem of wasted energy. In addition, it avoids the destructive competition by providing fairness — allowing all the nodes with similar QoS to have a similar probability to claim the reward. Our protocol adopts BFT based technology to achieve a reasonable transaction throughput. Furthermore, to achieve the goal of openness, PoQ segregates the whole system into sub-regions and nodes can join the regions freely with a certain amount of deposit.

PoQ takes the deposit one node holds alone with other factors (*i.e.,* error rate, activity rate, and reference factor) to evaluate the QoS of a node. As the node's activity rate is updated after each block proposal round, the fairness of the nomination is achieved by allowing some low activity rate nodes to have a chance to participate in the block generation.

In conclusion, compared with PoW based consensus protocol, PoQ has a higher transaction throughput and is energy-saving. Compared with BFT based protocols, our protocol has better network scalability and openness. To compromise PoQ, more than 1/3 of the committee members from different regions should be compromised. Compared with hybrid consensus protocol, we mitigate the case whereby different legal nodes are nominated at the same time, which is known as forks in nodes nomination. Table 3.1 shows that PoQ not only achieves a high transaction throughput but also avoids the issue that some nodes dominate the block generation.

3.3 Protocol Overview

In this section, we firstly give an overview of the proposed protocol, then we discuss the terminologies and the protocol assumptions, at last, we present the design of the protocol.

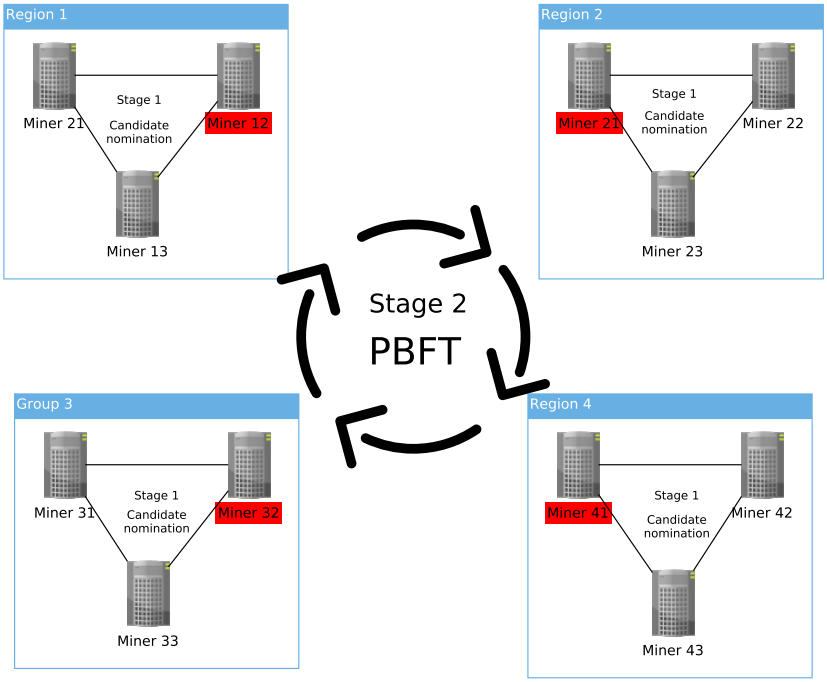


Figure 3.1: QoS protocol overview

3.3.1 Protocol Overview

In PoQ, new miners should make a deposit in the blockchain before they join the region. They also need to upload their public key to the blockchain for further communications. By observing the updates on the blockchain, the new miners are accepted by the corresponding regions. As it is shown in Fig. 3.1, we assume that there are 4 regions in the system. Each region has 3 miners. In PoQ, the block generation can be divided into three stages. In the first stage, miners in different regions run the PoQ candidate nomination scheme to elect the representative of the region according to the QoS of the miners. In the second stage, the nominated miners (miner 12 from region 1, miner 21 from region 2, miner 32 from

region 3 and miner 41 from region 4) share the transactions they have with each other and apply PBFT to decide which transactions should they involved in the proposed block. In the third stage, the nominated miners broadcast the new block within the region, and the rest of the nodes from the region verify the identity of the miner and the correctness of the block to decide whether to append the new block onto their own chain. In this way, the whole system is updated with the new block.

3.3.2 Terminologies in the Protocol

- **QoS:** QoS is employed to evaluate the quality of service that one node can provide. For our system, we employ deposit ratio, error rate, activity rate, reference factor as the QoS factors.
- **Region:** A region is a collection of nodes. The region is predefined before bootstrapping the protocol. When a new node joins the protocol, it chooses the region not based on the physical distance to that region but based on the QoS it could be evaluated. High QoS means a higher chance to be nominated.
- **Committee members:** A committee member is a node that is elected based on their QoS to represents its region. Its role is to communicate with other committee members to work out the transactions that needed to be put on the chain for this round.
- **Committee members' Queue:** For each nodes nomination round, a certain number of nodes is elected and then ordered in a queue. The queue is served as a cache since a nomination process can prepare the committee members for more than one block proposal rounds.
- **MGB block:** The role of the management block (MGB) is to store the address of deposit transaction made by a node and the signature from the node who recommends this new node.
- **Seed:** A seed is the hash value generated by a one-way hash function (*e.g.,* sha2, sha3) with the input of the content in the latest MGB and the time stamp when this MGB is created.
- **Nodes:** We define the nodes as the protocol participants in the way of contributing to generating blocks. Based on the QoS score, nodes in different regions have the chance to be nominated as the committee member to propose blocks; otherwise, they replicate the new blocks in the system to keep their local chain updated.

3.3.3 Protocol Assumptions

PoQ can run securely in an application scenario with the following assumptions.

- Honest users execute defects-free software and will not disclose their secret key to the public. How to protect the individual from attacking is beyond our research scope.
- We assume that the network within the region is strongly synchronized while allowing some nodes to be off-line for a while before it can recover liveness. We allow the attacker to block some nodes' communication, while it is impossible for the attacker to block all the communication within the region know as eclipse attack. The probability and countermeasures of eclipse attack are beyond our research scope and are discussed by Heilman [27].
- We consider the scenario that the malicious committee member always remains under 1/3 of the total.
- Secured bootstrapping can be achieved which means the initial participants are trustworthy during the system bootstrapping stage.

3.4 Protocol Design

In this section, we discuss the design of the protocol. We mainly focus on the design of the QoS evaluation scheme, the MGB block and the function of the seed. We also discuss how the whole system gets synchronized and protocol time-out strategy for handling exceptions.

3.4.1 Quality of Service (QoS)

We define the QoS evaluation factors as follows:

- **The deposit ratio:** We denote the deposit that a node i made as m_i. We define $M = \sum m_i$ as the sum of the deposit in a region. We define the deposit factor for the node i as $\eta_i = \frac{m_i}{M} \in [0, 1]$. For any nodes, their deposit must larger than a specific value agreed among all nodes in this region to participant in block generation.

- **Error rate:** Let s_e be the number of times that node e failed to generate a block and S be the total number of attempts that made by this node to generate blocks. We define the error rate factor for node e as $\beta_e = \frac{s_e}{S} \in [0, 1]$ and we define the error rate of the nodes which has generate no blocks as 0. To enhance the efficiency, we set a sliding window with the width of d_e for the calculation of the Error rate; which means we only calculate the latest d_e blocks that generated by this node. Since the errors one node made are historical data recorded on peer nodes, it is impossible for the attacker to revise. The width of the window d_e reflects how much historical data we applied to evaluate current status.

- **Activity rate:** Let B be the total number of block generation attempts since node i joined the network and b_i be the times of node i has been nominated. We define the activity rate as for node i as $\gamma_i = \frac{b_i}{B} \in [0, 1]$, and we define $\gamma_i = 0$ when there is no block generated. Same as the calculation of error rate, we set the window width of d_a for the activity rate evaluation. Different from other QoS systems, we take activity rate as a negative factor which indicates that the nodes nomination probability decreases with the increase of activity rate.

- **Reference factor:** Let ϕ be the reference factor. This factor reflects the quality of the node that refers to this node. If the recommended nodes' error rate is lower than the threshold, this factor is increased by a given value in MGB. This incentivizes the existing nodes to recommend new nodes and helps to bootstrap the system. In order to have a higher score for ϕ, nodes are motivated to recommend high QoS nodes to increase their reference factor score.

For our PoQ protocol, we denote QoS vector for node i as $\vec{v_i} = [\eta_i, \beta_i, \gamma_i, \phi_i]$ and we define the weight vector as $\vec{w} = [\alpha_1, -\alpha_2, -\alpha_3, \alpha_4]$. Different regions may have different $\vec{w}$. We define QoS of node i as $\xi_i = \vec{v_i} \times \vec{w_i}$.

It can be seen that all the QoS factors we introduced here are calculated on the nodes' historical data, and the evaluation result is subjective; which means that all the peer nodes should have the identical value for a given node. Since these QoS evaluation data are stored on the chain as a history, it is immutable; thus the adversary cannot introduce a bias on the QoS score by simply revising nodes' historical performance. For instance, to calculate a node's error rate, we need to divide the number of times that a node failed to propose the block by the total number of times it attempts to propose the blocks. Similarly, to compute the deposit ration, we need to find the total deposit that this node has made divided by the net worth of the region where the node belongs to.

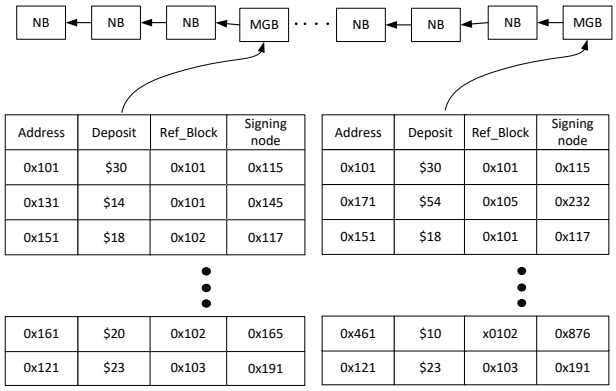


Figure 3.2: Block structure for nodes enrollment/resignation

The factor deposit ratio affects the nomination frequency of a given node. We set a maximum deposit for the whole system to avoid some nodes dominate the node nomination by making a tremendous amount of money. With the same Error rate and reference factor, the new node with a higher deposit has a higher chance to be nominated at first a few rounds. However, from the long-term perspective, its nomination frequency is roughly equal to the nodes that have the same error rate as the activity rate has a negative impact on the QoS evaluation. For an investor, considering their nomination chance is not directly enhanced by making a higher deposit, they may be unwilling to make the deposit much higher than the average in that region. This feature contributes to our design goal which encourages nodes to provide better service rather than competing by making a higher deposit.

We introduce the factor activity rate to balance the node nomination probability. As a result, no nodes can dominate the block generation just by proving low error rate and high deposit. For instance, there are two nodes denoted as node p and node q with the same error rate and reference factor. Node p makes a higher deposit, without introducing the activity rate factor, compared with node q, nodes p always has a higher chance to be nominated through their error rate is the same. When the activity rate factor is introduced in the system, the nomination probability of q increases with the growth of the idle time (the negative of the activity rate) in the system. As a result, it is unlikely to see that compared with node p, node q has no advantage in committee member election competition.

3.4.2 Region

The purposes of the region can be summarised in the following aspects: 1) It simplifies the block broadcasting process. Compared with broadcasting new blocks in the whole network, in our protocol, the peer nodes only need to query the new block from the committee members within the region. The network has less convergence time. 2) It is more difficult to launch the eclipse [27] attack which targets at partitioning the network to generate an inconsistent view among different regions. Even if the network partition happened in less than 1/3 regions, the rest of the regions could still nominate nodes, and these nominated nodes can collaborate to generate new blocks. 3) It reduces the nodes' workload as the node within the region only needs to maintain the QoS of the nodes within the region. 4) It optimizes the node distribution. Since the nodes are motivated to have a higher QoS score, they prefer to stay with the nodes that have less network latency. As a result, the network latency among all the regions as a whole is optimized.

To maximize the node's profit, before a node joins a region, it should investigate the existing number of nodes in the region, the network latency with the nodes in that region and the deposit other nodes made. The best strategy is to join the network that has fewer nodes and also maximize the effect of the deposit which means increasing the ratio between the reward it gets and the deposit it makes. With more nodes join a specific region, the competition in that region becomes fierce, and finally, the number of nodes among all regions reaches a balance dynamically. Since all nodes make the decision independently and they want to maximize their profits while having no idea about peer nodes strategy, the protocol will reach the Nash Equilibrium [57]. In other word, all the protocol participants have nothing to gain by changing by moving from one region to another.

Each of the regions elects a node to act on behalf of the region to propose the transactions to be put in the block. In order to increase the system throughput, each region nominates and caches T committee members which can be used for the next $T-1$ block proposal rounds without performing the nomination process again[3].

3.4.3 *MGB* Block

We propose the MGB block to store the deposit one node made and its recommendation information. As it is shown in fig.3.2, MGB block is created regularly in the region. For

[3]The attacker does not have enough time to attack the cache as it is flushed within 1 second. For instance, if the throughput is 9.7K TPS (about 24 blocks per second) and the cache contains 100 nodes, the cache will be flushed roughly every 4.1 seconds.

any blockchain generation round, if the committee members fail to generate the block with a given time. This round's block generation is marked as a failure, and the deposit of the committee members is deducted in the MGB block proposed by the next round committee members. The nodes which have less than a specific amount of deposit are not qualified for participating in block generation.

The block structure for node enrollment and resignation is shown in Fig. 3.2. In the MGB, "ref_block" indicates the block that this node made the deposit and "signing node" indicates which node creates this record. The block tagged with "NB" means a normal data block. In our protocol, to ensure that the seed can be updated frequently to avoid the grinding attack (if a seed exists for a long time, the adversary may find the suitable address to win the committee member election), we ask every MGB block is followed by a certain number of "NB" blocks. If none of the node information in the MGB is updated, we update the time stamp of the MGB to ensure the seed is updated.

We make these information public accessible to avid a node joining two different regions with the same identity. When the peer nodes verify the amount of the deposit one node made, it can get the data in the blockchain without trusting any third party. The deposit in the blockchain is regarded as a term deposit (we borrow the deposit idea that implemented in Ethereum[4], and the amount of the deposit can be located by investigating the deposit transaction address in the blockchain.). The node can withdraw or spend this deposit after a certain time.

3.4.4 Seed

A seed is the hash value generated by a one-way hash function (*e.g.,* sha2, sha3) with the input of the content in the latest MGB and the time stamp when this MGB is created. Since the time when the new block is proposed is unpredictable, a timestamp is introduced in the seed generation to make the seed hard to be predicted. For the node nomination round, the seed is combined with the node's address [5] to generate the "lucky number" for this node. Among the nodes which offer top-n QoS (n is the protocol parameter which defines the size of the committee member candidates set), the node which has the minimum "lucky number" is elected as the committee member. To avoid the adversary preparing the address to gain the chance to be elected by launching the grinding attack, a seed is employed to randomize the output of the hash function mentioned in Algorithm 4 and thus

[4]https://medium.com/coinfi/how-to-create-your-ethereum-address-323d176e5aab

[5]In this paper, the node's address is the public key of this node. The concept of node address is the same as the client address in Bitcoin system.

enabling the nodes with the same level of QoS to have a fair chance to be elected. The value of the $seed_0$ which bootstraps the committee member election is not generated from the MGB block but chosen by the initial participants using distributed random number generation [58] to avoid any participants prepare the parameters in the MGB to bias the hash value of MGB.

3.4.5 Node

In our protocol, we take the nodes as the entities that contribute to generating blocks. When a block is generated by a node, the node can claim transaction fees from the block. The nodes have two roles. The first one is to calculate the QoS, and "elects" the committee member. The "election" process here is different from the traditional election which asks the nodes to cast the votes. In our protocol, as the QoS evaluation is subjective, the potential committee members are known, and the peer nodes only need to check the qualification of the committee member by calling the Verifiable Random Function (VRF). The detail of the node nomination is discussed in section 3.5.5. Another role is to provide the block query service to the end user and wait for a chance to propose the blocks.

3.4.6 Network Synchronization and Time-out Scheme

For all the distributed systems, the synchronization of all participants is of great importance. In our protocol, we suggest applying the Network Time Protocol (NTP) server in the network to achieve weak synchronization for the off-line node to catch up. A typical delay of NTP protocol is 5 ms to 100 ms [59]. This is acceptable for our PoQ protocol as we do not rely on all nodes to update the blocks and evaluate the QoS simultaneously (PoQ only checks the blocks on a given node when it is elected to be a committee member or involved in verifying the blocks on a committee member's chain).

We also apply the time-out scheme to ensure the functionality of the protocol when an exception happens (*e.g.,* the committee member quits the protocol during the block proposal round). For each block proposal round, a timer is set within a region for all nodes. If no new block is appended to this round's committee member after a given time, the nodes will request the latest block from the committee members from other regions. We also set an upper timer for the committee member who proposes new blocks. Within a given time, if the committee members cannot agree on the blocks they propose, they quit the block proposal round.

In PoQ, we allow the cross-region communication to have weak synchronization, while we suggest the communication within the region to have strong synchronization to achieve higher performance.

3.5 Protocol Interactions

In this section, we discuss how the nodes join/leave the protocol and then, we describe how the nodes collaborate to propose new blocks.

In PoQ, on a high level, we divide the block proposal into three stages. In Stage I, each region nominates a node as a committee member who proposes the transactions to be included in the chain on behalf of its region. In Stage II, the nominated nodes from different regions form a block proposal committee to determine the transactions that should be put on the chain and to be shared among all the regions. In Stage III, all nodes request the new updates from the committee member of their region to synchronize their local chain. We allow Stages I and II to be run in parallel. For each node nomination round, all the nodes' QoS should be re-evaluated (Stage I) and this is relatively time-consuming and may take longer than Stage II. To avoid the case where Stage II waits for the committee members to be nominated, more than one node is nominated in each region for each node nomination round, and the remaining committee members are cached in a queue. The block proposal (Stage II) can only be run serially to keep the generated block in a strict order.

In the following paragraphs, we first discuss the bootstrapping of the system. Then, we discuss how the new nodes join the system and how their QoS is evaluated. At last, we discuss how the system nominates the committee members and how the committee members collaborate to propose new blocks.

3.5.1 Bootstrapping the System

Before the system starts, the initial participants need to work out the number of regions. To kick-off the system, at least four regions need to participate in the protocol (with the tolerance of 1 faulty replica), and each region should nominate a committee member; otherwise, the PBFT protocol cannot reach an agreement on the block generation. Similar to the Hyperledger Fabric, we need to create a genesis MGB block before the protocol starts. All initial participants should agree on the contents (e.g., the deposit each participant made, reference factor of each participant) in the genesis MGB. We ask all the initial participants disclose their address before applying the distributed random number

generation function [58] to avoid any participant preparing the address to advantage themselves while performing the random sorting for the committee member election. We call the generation of the genesis MGB as the secured bootstrap as all the initial participants should be trusted and they have to cooperate to propose the first MGB.

3.5.2 Bootstrapping the New Nodes

Nodes that join the protocol should first choose one of the regions provided by the system. Since the honest nodes are profit-driven, they prefer to join the region where they have a higher chance to be nominated; thus they can claim rewards from the block proposal. To find a suitable region, the following factors should take into consideration. 1) It should have lower network latency in that region. If the network latency is high, the data on the nodes may be out-of-date. If a node which elected as a committee member proposes the out-of-date data, its proposal is rejected by the peer committee members. 2) The average deposit in this region. The node should find the region that they can benefit from their deposit.

To join in a specific region, the enrollment process is shown in Algorithm 1 and is accomplished by finishing the following two compulsory steps. a) It makes a deposit to the blockchain (line 1 Algorithm 1. If the node which has the same address already made the deposit in the system, this process is rejected) b) It broadcasts its registration information (e.g., address, the deposit it has made and the QoS information) to the whole region (line 2-6 of Algorithm 1). The node which is nominated as a committee member for this region is responsible for proposing this new node's information (*e.g.,* node default QoS, node join time stamp) to the MGB block. During the block generation, committee members will apply the PBFT protocol to add the new nodes into the system (the validity of the new nodes' registration information is verified by all committee members). These two steps avoid nodes joining different regions simultaneously and also increase the cost for the malicious nodes to join different regions and attack those regions. For the new node, there are two approaches to have customized QoS. One is to have the QoS derived from the node that recommends it – a referential method (line 6 of Algorithm 1). The other is to increase the total amount of deposit and has the default QoS value (line 4 of Algorithm 1). When the deposit term is expired, we allow the nodes to update their deposit by updating their deposit in an MGB block. For security concerns, we also allow the nodes to update their addresses (public keys) by updating their address ina MGB block and use the previous key to sign the block to make it valid.

Now we explain how can a new node join the system with different QoS values.

- Deriving the QoS by reference: For the nodes that want to join the network with a derived QoS, they should ask one of the active nodes from the same region to register them on the blockchain. As it is shown in Algorithm 2, the blockchain can verify the validity of this record by checking the signature of the referee (line 1 of Algorithm 2) and also ensures the new nodes' score of each QoS factor is no larger than the referee (line 9 of Algorithm 2). The committee also verifies the qualification of the referee by checking its online time and reference factor (line 3-5 of Algorithm 2). The referee is motivated to assign reasonable scores for the new nodes; otherwise, its reputation will be affected. For a node that wants to recommend other nodes, the following two requirements should be satisfied. Firstly, this node should be online for a given time. Secondly, this node should meet the minimum QoS recommendation requirements. These two requirements avoid malicious nodes from recommending their partners into the region within a short time and avoid the decrease in QoS of the given region caused by a large amount of poor QoS nodes.

- Deriving the QoS by deposit: The new nodes can also increase its QoS score and the possibility to be selected by making a higher deposit. As a result, compared to other new nodes, the protocol has a higher chance to select it in the first few rounds. However, because of the negative factor of the activity rate, with the time progresses, the QoS of this node becomes stable and has an equal chance to other nodes that offer the same QoS level. Fig. 3.3(b) shows the nomination frequency comparison between the nodes that have a higher deposit and a regular deposit. This feature discourages the nodes who want to make profits by investigating a large amount of money.

3.5.3 Node Resignation

The node has two ways to quit the protocol. The first one is to quit the protocol without withdrawing the deposit (line 2 of Algorithm 3). The protocol will realize the exit of the node when the matured deposit is not claimed by its owner.

Another way is to wait for the maturity of the deposit. When its deposit term is at maturity, the protocol refunds it automatically. Then, the nodes need to update the addresses in the MGB (line 4 of Algorithm 3) before claiming the refund with the receipt showing that he has quit the protocol by removing its address from the MGB block (line 6 of Algorithm 3).

Algorithm 1 **Nodes enrollment**

```
1: ret, receipt ← MakeDeopsit(m)
2: if ret == false then
3:     return
4: end if
5: if ref == null then
6:     v⃗ ← AssignDefaultQoS()
7:     TimeNow ← GetCurrentTime()
8:     RecordItem ← CreateRecord(v⃗, TimeNow, receipt)
9:     sig ← CreateSignature(RecordItem, sk)
10:    ret ← PutOnAddressChain(record, sig)
11: else if
    then
12:    ret ← PutRefChain(ref, receipt)
13: end if
14: return ret, receipt
```

3.5.4 QoS Evaluation

In PoQ, we take the node λ as an example to show how the QoS of all the other nodes is evaluated. As the data on λ is synchronized with the blockchain, it retrieves the QoS information locally. To improve the efficiency, λ creates a local metadata pool to cache the QoS information for all the nodes including itself. Thus, it does not need to search on chain frequently (*nodepool* in Algorithm 4 is the local cache for the QoS data). To evaluate the QoS of node ψ , node λ retrieves the deposit made by ψ divided by the total amount of deposit in the given region. For the calculation of the error rate, λ counts the times a node ψ is nominated, denoted as S and the times ψ failed to propose the block denoted as s, then calculate $\beta = \frac{s}{S}$. The calculation of the other two factors are quite similar, and we skip them to make the discussion concise. Since the QoS data is stored on the chain and read-only, the QoS calculation result among all the honest nodes should be the same.

3.5.5 Node Nomination

The nomination process is explained in Algorithm 4. Firstly, each node retrieves the QoS records of all the nodes in the same region from their local chain and calculates the QoS score for each node. To accelerate the evaluation speed, each node may maintain the statistic data for each node in a node pool, so that, it gets the QoS score from that pool as shown in line 2 of Algorithm 4. After that, it maps the QoS with the node address and sorts the nodes according to the QoS score (line 3 of Algorithm 4). Let $\mathcal{K}$ be the number of candidates for the committee member. For the nodes which have the top-$\mathcal{K}$ score, they provide a hash value according to their addresses appended with the seed (line

Algorithm 2 QoS Reference Check

```
1: signature, v⃗ ← RetrieveSig(record)
2: ret ← CheckSig(pk, signature)
3: if ret == false then
4:     return false
5: end if
6: if timenow − timestamp < C then
7:     return false
8: end if
9: if Reffactor_ref < C then
10:     return false
11: end if
12: if Deposit_new > Deposit_ref then
13:     return false
14: end if
15: for i, j ← v⃗_new, v⃗_ref do
16:     if i > j then
17:         return false
18:     end if
19: end for
20: return true
```

Algorithm 3 Nodes resignation

```
1: if ForceQuit == true then
2:     ret ← UpdateAddress(sk, record)
3: else if
   then
4:     ret ← UpdateAddress(sk, record)
5:     if ret == false then return false
6:     end if
7:     refundret ← AskForRefund(sk, receipt)
8:     if refundret == false then
9:         return retfundret
10:    end if
       return ret
11: end if
```

5-6 of Algorithm 4). At last, each node sorts the hash value, and the node which has the smallest hash value is regarded as the committee member for this round. It generates the proof of the committee member π and publishes this hash value together with π (line 7-8 of Algorithm 4). The nodes from the same region who fail to be in the committee member verify the identity of the node who claims to be in the committee (line 9 of Algorithm 4) to decide whether clone the new block from this node.

Algorithm 4 NODES NOMINATION

for $address, \vec{v} \leftarrow nodespool$ **do**
  $\vec{R} \leftarrow append(\vec{v} \times \vec{w})$
end for
$\vec{R} \leftarrow sort(\vec{R})$
$\vec{A} \leftarrow TopKmap(\vec{R}, addresspool)$
for $each \leftarrow \vec{A}$ **do**
  $\vec{QA} \leftarrow hash(each||seed)$
end for
$sort(\vec{QA})$
if myself in $\vec{QA}[1]$ **then**
  $hash, \pi \leftarrow VRF_{sk}(seed, address)$
  $broadcast(hash, \pi)$
else if
then
  $verify_{pk}(hash, \pi, address, seed)$
end if

3.5.6 Block Proposal

The committee members from different regions propose a list of y transactions based on the transaction fee they offer. The PBFT network is constructed based on the nodes nominated by different regions. Before employing PBFT to process the transactions, all peer committee members conduct the committee member identity check with each other: The peers need to check whether the proof matches with the address of the committee member. If there is more than one node claim to be the committee member from the same region, the peer nomination nodes investigate the correctness of tuple $< hash, \pi >$ and also the QoS of these two nodes. The node has a smaller hash value, and its QoS is within $\mathcal{K}$ is regarded as authoritative. Fake committee member's deposit is reduced in the MGB block as a punishment.

We treat the PBFT algorithm as a black box. With the given transactions from different regions, the PBFT algorithm generates the ordered transactions in this block that are agreed upon by all the committee members. All the committee members sign on this block to make it valid, and finally, this block is appended to the committee members' chain. If the PBFT algorithm fails to reach an agreement on the transactions, all the nodes will be notified this failure by the PBFT algorithm and nodes in each region can evaluate the error rate of peer nodes within the region. The committee member for the next round reduces the deposit of previous committee members who failed to propose the blocks in the present MGB block.

3.5.7 Agreement Among All Nodes

The nodes in the same region monitor the committee member's chain. If it is updated, they verify the validity of this new block and put it on their chain. If they fail to observe the new block on the committee member, they get updates from the committee member in other regions by searching the committee member transaction in the MGB block in that region. The error rate of committee members who fail to append the blocks on its chain is increased and recorded by peer nodes.

3.6 Threat Models and Countermeasures

3.6.1 Threat Model

At the high level, to launch an effective attack against PoQ, an adversary need to compromise at least 1/3 of the regions, due to the well-known quorum theory [60]. The adversary can achieve that by compromising the nomination process in more than 1/3 of the regions or by compromising the nominated committee members directly. For the latter approach, the adversary can only attack one block proposal round as the committee members are replaced by new members in the next round. To make the attack more efficient, the adversary needs to compromise the protocol to elect the malicious nodes to be the committee members constantly.

In this section, we discuss the thread models of our consensus protocol, we first present the attack on the network infrastructure, and we then present the attack towards the committee member nomination and block proposal.

3.6.1.1 Attacking the Network

The adversary attacks the network to delay message propagation. Two types of attacks are possible: the first one is to isolate the committee members. If the communication between the committee members is blocked due to the attack on the network infrastructure or disabling the committee members to respond to any requests by a Denial-of-Service attack, the PBFT protocol is suspended if the attack affects more than 1/3 of the total committee members. The second type of attack is to attack nodes within the regions. An adversary may attack the network infrastructure to delay the communication between nodes. Thus, nodes in the same region cannot update their local chain immediately when the new block

is generated by the committee members. As a result, the nodes in a region may evaluate the QoS with out-of-date information, and when it is nominated to be the committee member, the out-of-date local chain will be rejected by other committee members.

3.6.1.2 Attacking the Node Nomination

For this attack, the adversary needs to revise the malicious nodes' QoS information in the blockchain. As the QoS information is append-only, the adversary can only create a fake QoS when the block generation is under the adversary's control. It causes the chicken or the egg causality dilemma. Another passive approach is to behave honestly for a period to gain a high QoS, then attack the system when malicious nodes are nominated for PBFT in more than 1/3 regions.

3.6.1.3 Attacking the Block Proposal

An adversary can attack the block proposal process in the following three ways: 1) It tries to attend the BFT block proposal stage even if it is not the elected committee member (*i.e.,* a fake committee member). Since the block generation is based on PBFT algorithm, the adversary can control the block generation by pretending to be the committee members in more than 1/3 regions. 2) Compromise a committee node to refuse to append the new block onto its chain or to try to replace it with a tampered block. The nodes that are in the same region with this compromised committee node may append a fake block to their local chain. This will create an inconsistency of the nodes in this region, and thus the nodes from this region will be rejected by nodes in other regions for further block generation. 3) the malicious nodes refuse to update the latest generated block to invalid the transactions in the latest generated block.

3.6.1.4 Network Partition Attack

An adversary may attack the network by partitioning the whole network into small segments. As a result, different nodes may have a different view of the whole network. Because of the inconsistency view of the blockchain, the committee members fail to reach the agreement on the proposed transactions that proposed, and the QoS of these committee members will be affected. To launch an effective network partition attack, the adversary needs to delay the new transactions' propagation

3.6.2 Attack Analysis and Countermeasures

In this section, we analyze the possibility of deploying the attacks that we mentioned in section 3.6.1. We also discuss the countermeasures to these attacks.

3.6.2.1 Analysis of the Attack on the Network Infrastructure

The attack that targets at the network infrastructure is hard to deploy. When the network congestion happens on some network infrastructures, other network infrastructures are smart enough to redirect the data flow on the congested devices to other network infrastructures. During the attack, even the committee members fail to generate the new blocks. The timeout scheme allows the blockchain participants to reform the committee to avoid the deadlock situation.

To avoid the adversary blocking the communication of a given node by filling the nodes' p2p address bucket with invalid addresses, protocol participants can cache some peers that they trust in their p2p address bucket. Thus they can always get updated information from these trusted nodes.

3.6.2.2 Analysis of the Attack on the Node Nomination

Based on the assumption that the majority of the nodes are motivated to comply with the protocol to claim the reward, it is difficult for the adversary to collude with a certain number of nodes to vote for the compromised nodes as the committee members. Once the nodes are found to have disobeyed the protocol, their deposits are deducted. For our protocol, the probability of being nominated depends on the objective QoS score, so the adversaries should maintain a high QoS before the attack. For simplicity, we assume that before the attack, the malicious node maintains a high QoS and has the probability of κ to be nominated across all the compromised regions. Then the probability of attacking the protocol successfully is shown in equation 1.

$$1 - \sum_{i=0}^{\lfloor \frac{\mathcal{N}-1}{3} \rfloor} \binom{i}{\mathcal{N}} (\kappa^i * (1-\kappa)^{\mathcal{N}-i}) \tag{3.1}$$

For a 12 regions system, if $\kappa = 30\%$ of the nodes are malicious in more than three regions, then, the adversary has 50% probability of controlling the block generation. However, in reality, maintaining κ as a high value for a long time is difficult, because 1) the

malicious node should behave normally; otherwise its QoS drops and the result in the drop of κ. 2) the malicious nodes need to wait for all its partners to be the committee member; otherwise, it will be caught cheating by other committee members. There is no time guarantee when more than 1/3 committee members are its partner. If anyone has 30% of his nodes to be nominated in more than 1/3 regions, he takes 50% of the total block proposal reward. Since he can get half of the block proposal reward, he has no motivation to break the protocol.

3.6.2.3 Analysis of the Attack on the Block Proposal

In PoQ, the committee member identity check, which is mentioned in Section 3.5, avoids the probability that the malicious node is pretending to be the committee member. Because the peer committee members can investigate the QoS of this node; if one is caught cheating, it loses its deposit. If the malicious committee member refuses to update the block, the nodes in the same region with the malicious node can query the latest block from the nodes in other regions. We ask the committee member to sign on the generated block to avoid any committee member tampering the block. If a node is trying to avoid some transactions by refusing to update the block, the inconsistency of the blocks on its chain with that on other nodes will be identified during the committee member identity check.

3.6.2.4 Analysis of the Attack on the MGB Block and QoS Value

Attacking the MGB block is impossible in our protocol. First, MGB as a block, once, it is put on the blockchain, it is immutable. Second, for the malicious committee member who wants to alter the node's information during the MGB proposal stage is also difficult since all the data on the blockchain is verified by other peer committee members.

It is possible for the adversary to delay an honest node's network to increase its error rate thus affect this node's QoS. However, this attack will not affect the PoQ unless the adversary can deploy the attack on a large scale. Additionally, since the nodes can have some peers as the trusted nodes, the adversary cannot block the honest nodes' communication by filling up the nodes' address books with invalid addresses. The related analysis is discussed in attacking the network section. How to help the administrator to protect a node from such an attack is beyond our research scope. Since all the QoS factors are objective statistic data, the malicious node has no way to bias the QoS evaluation on honest nodes.

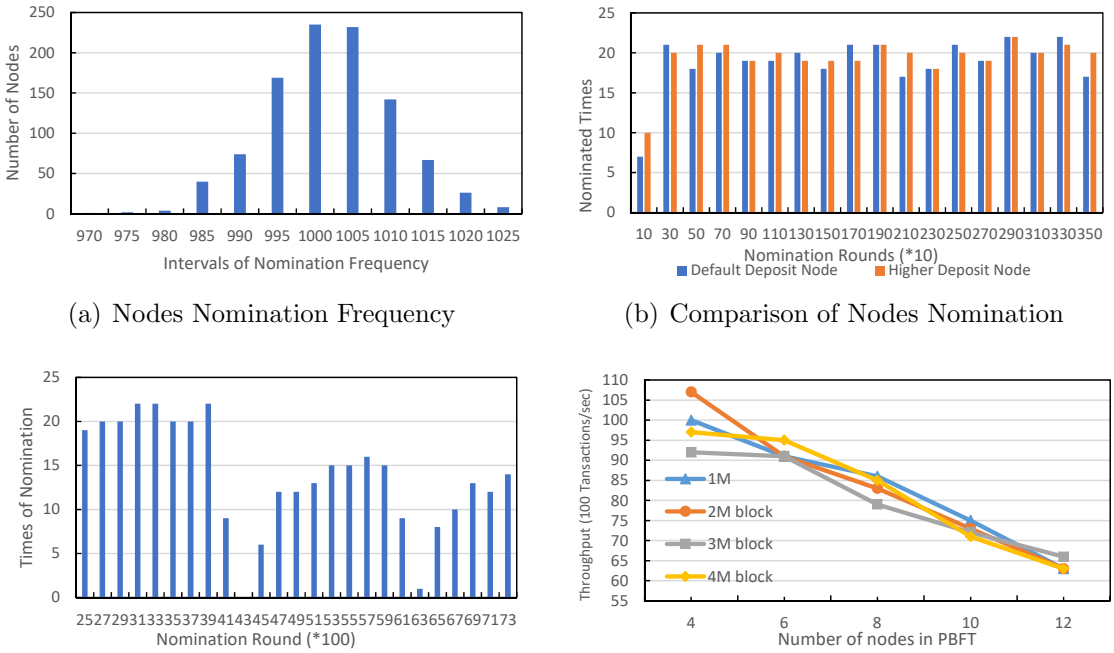
3.6.2.5 Sybil Attack and Grinding Attack

To launch an effective Sybil attack, the adversary should have enough financial support to create enough malicious nodes in different regions. In the section of analysis of the attack on the node nomination, we analyze the proportion of malicious node to lunch a successful attack with 50% probability. The grinding attack by which the adversary tries different address with the seed from MGB to ensure he has the smallest "lucky number" is also impractical in our protocol. Since the one-way hash function generates all the hash values, it takes a long time for the adversary to search for a suitable address to make the "lucky number" smallest in a large search space. Additionally, as the seed are updated regularly because of the routing updating of the MGB block. Concerning the seed expiration frequency, it is impossible for any adversary to find a suitable address before the seed gets expired.

In conclusion, attacking the network infrastructure could be possible to delay the nodes' communication; however, the attack is usually difficult to deploy in an open distributed environment when the nodes' address book cannot be updated easily by arbitrary nodes. Compromising the committee members can stop the system from generating the blocks correctly; however, the number of compromised committee members should pass 1/3 of the total numbers. The online QoS update scheme makes it difficult to attack the protocol by simply employing a certain number of malicious nodes. Our reference scheme only allows the qualified nodes to introduce new nodes in the region to guarantee the QoS of the whole region. It is impossible to compromise the nodes in the network to nominated the malicious node as the peer committee members will check the validity of each other. "Sybil" attack is difficult to launch in PoQ as 1) variation of QoS factor weights avoid the same attack pattern applied into different regions. 2) Sharing of the node address prevent the same node from joining different regions simultaneously. 3) The deposit scheme creates a barrier for introducing a large number of malicious nodes.

3.7 Protocol Architecture and Evaluation

In this section, we first evaluate the performance of PoQ in different simulation scenarios. Then, we discuss one of the implementations to demonstrate the feasibility of the practicality of the proposed implementation architecture.



(a) Nodes Nomination Frequency

(b) Comparison of Nodes Nomination

(c) Nodes Nomination Frequency Variation

(d) Transaction Throughput in PBFT

Figure 3.3: PoQ performance evaluation

3.7.1 Protocol Evaluation

We evaluate the PoQ performance by two approaches. In the first approach, we employ two 16 core E5-2660 CPU with 128GB memory to evaluate the performance of the node nomination scheme. We employ 40 docker instances to simulate the nomination process. For each docker instance, we run 25 threads, and each thread stores the QoS information of a simulated node. We observe the QoS and nominated frequency of the "simulated nodes" within a given time. For the second approach, we deploy the PoQ core algorithm in a cluster which contains 7 virtual machines representing 7 regions to evaluate the throughput. Inside each virtual machine, we run 15 docker instances which represent PoQ participates/nodes. The PoQ architecture which is deployed in the cluster is shown in Fig. 3.4(a).

3.7.1.1 Nomination Frequency Among All the Nodes

Firstly, we create 1,000 nodes and set all the nodes to have the default QoS values. We run the experiments for 10,000 times to give all the nodes enough time to be evaluated.

It is shown in Fig. 3.3(a) that only one node is nominated 970 times (least) and one node is nominated 1,025 times (most). The nomination difference between "luckiest" node and "unluckiest" is 55 for 10,000 rounds nomination and the majority of the nodes are nominated between 990 times and 1,015 times. The fairness is achieved as the nomination difference for the majority of the nodes is within 35 nominations for 10,000 rounds.

3.7.1.2 Nomination Frequency Between Different Nodes

For this experiment, we choose two nodes, one of them makes the default deposit, and the other makes the deposit five times higher than the default deposit. We trace how these two nodes are nominated in the region. The experiment result is shown in Fig. 3.3(b). It is observed that for the first 300 rounds, the nomination frequency of the node which makes a higher deposit is almost doubled compared to the node that makes the default deposit. After the 3000th round, the nomination difference between these two nodes is within 2 for a given interval. The experiment result shows that the nodes that made a higher deposit are evaluated more frequently for the first few rounds. However, after the protocol reached the balance of its QoS evaluation, the higher deposit has no advantages in further nomination.

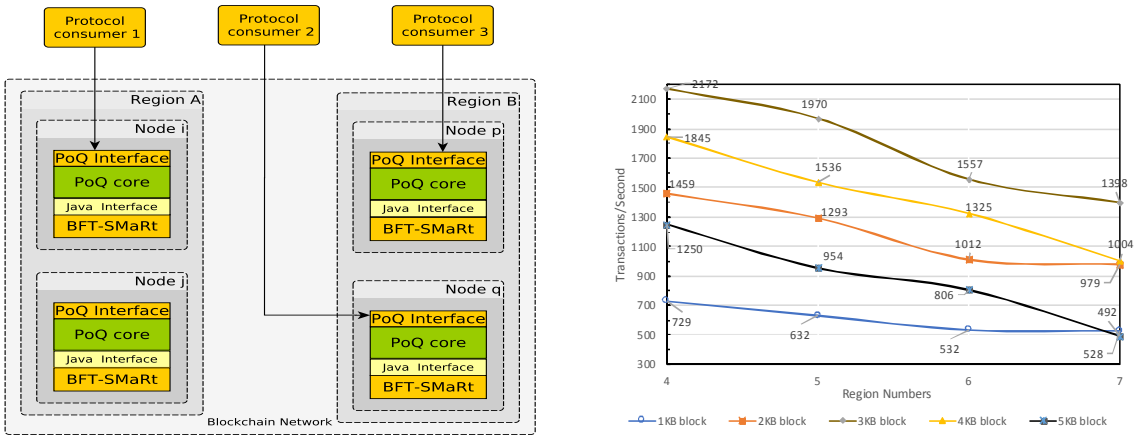
3.7.1.3 Impact of QoS Variation on Node Nomination

To show how the nomination frequency varies with the fluctuation of the QoS, we adjust some nodes' QoS by increasing their error rate. The nomination experiment result is shown in Fig. 3.3(c) and it can be observed that it is not nominated for the interval between the round 4,100 and 4,500 after the increase of the error rate and recovers to normal between the round 4,700 and 5,900. We increase its error rate again at the round 6,100, and the consequence of that is it is nominated less than five times during the interval 6,300 and 6,500, and then, the nomination frequency increases gradually after the round 6,500. The experiment result shows that if the node does not comply with the protocol, the reduced QoS evaluation has an impact on the probability to be elected. It takes time for the node to recover from bad behavior.

3.7.1.4 PoQ Emulation Transaction Throughput

For this experiment, we set the size of a transaction as 256 Byte. We evaluate the performance of the PBFT network in our block proposal process, and the result is shown in

Fig. 3.3(d). Firstly, we increase the PBFT network from consisting of 4 nodes to 12 nodes. Then, we test the impact of the block size increased from 1MB to 4MG for a given PBFT network. For all the throughput tests, we set one node as a malicious node. It can be observed that with the growth of the number of nodes in the PBFT network, the transaction throughput drops dramatically. It shows that for the network that contains six committee member, the throughput of the system has the blocks size of 4MB is 9.5K TPS which is higher than 1MB block network which has the throughput of 9.1K TPS. It demonstrates that though larger blocks need more time to be processed by the PBFT network, it can contain more transactions. As a result, the transaction throughput for the larger block can be higher than a small block. The experiment shows for a network that contains eight nodes, setting the block size as 4MB can achieve the best transaction throughput while 3MB block size is the most suitable for a network that contains 12 nodes.



(a) PoQ deployment architecture

(b) Transaction Throughput in PBFT

Figure 3.4: PoQ architecture and throughput

3.7.2 Demonstration System

3.7.2.1 Protocol Architecture

The architecture of the PoQ we implemented is shown in Fig. 3.4(a). To make the architecture diagram concise, we only draw two regions and two nodes in each region in the diagram (for a practical setting, at least four regions are needed). For each region, it has two nodes. PoQ is running independently on each node. Protocol consumers send the

transactions that need to be put on the chain to any nodes in the blockchain. The PoQ protocol sends the transactions to all the nodes and finally puts them on the chain. To implement PoQ, we take advantage of the open source project called BFT-SMaRt to run the PBFT among committee members. We customized the BFT-SMaRt and integrated it into our PoQ. We implement a Java interface to allow PoQ core to interact with BFT-SMaRt. Protocol consumers interact with PoQ through the PoQ interface layer. However, we have not implemented the Committee members' Queue in our current design. As a result, after the current committee member's block proposal, a certain amount of time is needed for the new committee members to confirm each other before the block generation. The preparation time for different regions is discussed in section 3.7.2.2. It is our future work to implement the committee member cache scheme to allow block generation and committee member preparation to run in parallel. Currently, the purpose of this work is to demonstrate the feasibility of PoQ protocol.

3.7.2.2 Initial Observation

We deploy the referenced PoQ system in Docker containers to demonstrate the feasibility of the architecture. We set the size of a transaction as 256 Byte. We define the throughput of the system as the transactions that have been successfully processed for a given time. Since it does not have a committee member queue scheme, this single-thread design has an impact on transaction throughput. The experiment results are shown in Fig. 3.4(b). Firstly, we increase the PBFT network from consisting of 4 regions to 7 regions. Then, we test the impact of the block size increased from 1KB to 5KB for a given PBFT network. It can be observed that with the growth of the number of nodes in the PBFT network, the transaction throughput drops dramatically. The system throughput increases gradually with the growth of the block size and reaches the maximum value of 2.2 TPS. While the throughput of the system which has the block size of 5MB drops to about 0.5 TPS in the seven regions system, the transaction throughput increases with the increase of block size as for each block proposal, the larger blocks contain more transactions. However, with the growth of the block size, the committee member needs more time to process the blocks. According to our experiments, it takes 2 seconds to finalize committee members confirmation in the region of 4. However, it takes 6 seconds for updating the committee in the region of 7.

The throughput difference between our initial observation and the throughput simulation can be explained as follows: 1) The committee members cache scheme is not implemented in our referenced PoQ. Thus a certain amount of waiting time is consumed for forming the committee before block generation. 2) The implementation of PoQ is not

optimized. Instead of implementing the native BFT library, we take advantage of java native call interfaces to invoke a third-party BFT library to run the BFT. The overhead of the cross-module invocation is not negligible. Additionally, the third-party BFT library itself needs to be tuned to achieve the best performance. 3) We do not deploy our referenced PoQ on some high-performance infrastructures. The overhead caused by the docker containers is not negligible. It is our future work to minimize the throughout gap between the simulation and our implementation.

3.8 Conclusion

PoQ is a new permissionless blockchain consensus protocol which achieves a high transaction throughput and fairness among all nodes. PoQ's design is based on a hybrid protocol which selects nodes using QoS for running a BFT-style consensus. Experimental results show QoS based node nomination protocol creates a fairness environment for all the nodes to compete for block nomination. The 9.7K TPS throughput demonstrates that PoQ can work with the high-frequency trading scenarios.

In the future, we will optimize our PoQ architecture and deploy PoQ in a real environment. We will also evaluate the impact of Cross-region eclipse attack in PoQ and propose practical countermeasures. We notice some blockchains [61, 62, 18] apply trees and directed acyclic graph to replace the chain structure. We would investigate whether PoQ can benefit from such a data structure to increase transaction throughput. We also notice that a new cryptocurrency model called Payment Channel networks [63] has been proposed to address the issue of poor transaction throughput in blockchain systems. However, there is no dedicated underlying consensus protocol to support the payment channel networks. It is our research interest to integrate PoQ with payment channel networks to provide a more secure and efficient payment platform.

Chapter 4

Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks

4.1 Abstract

PCN have been proposed to address the low transaction throughput of the permissionless blockchain protocols. Though the PCNs allow users to have the unlimited number of transactions in the channel without interacting with blockchain, it leaks the entire payment paths to the public. To address the payment path leakage issue, we propose a Chameleon-hash based payment protocol, called CHTLC. Using Chameleon-hash function in a multi-layer fashion guarantees that no user can recover the payment path if at least one intermediate payment node is honest. For the same payment path, compared with MHTLC protocol of Malavolta et al. [20], CHTLC is 5 times faster in the payment data initialisation, and the communication bandwidth is reduced significantly from 17,000KB to just 7.7KB.

4.2 Introduction

Bitcoin [13] and Ethereum [46] are two largest cryptocurrencies in the world. Instead of storing the transactions on a centralised ledger, these transactions are stored on different participants in an immutable chain structure database. Consensus protocols are applied to ensure the consistent view of the ledgers on different participants. However, due to the

scalability nature, it is difficult to have all the nodes to achieve consistency in a relatively short time which results in the low transaction throughput. On average, Bitcoin can only handle 7 transactions per second [64], while Visa can handle 2000 transactions per second [64].

To address the low transaction throughput issue, Bitcoin proposes a new payment scheme called "lightning network" [11] which supports off-chain transactions to avoid the transaction confirmation time. In a nutshell, a pair of users open a payment channel by locking their bitcoins in the smart contract as deposits. After that, off-chain payment transactions can be placed by agreeing on the new distribution of the deposits. The payment channel is closed by publishing the allocation of the final deposit on the blockchain by any party. Though the transaction throughput is increased by avoiding putting all the transactions on the blockchain, it limits the payment to be settled between two direct users. It is a great convenience if the existing payment channels can forward the payment for the users who has no direct payment channels. To address this issue, payment channel networks (PCNs) are proposed and instantiated by some popular protocols [65, 66, 67, 68]. However, there exists a serious payment privacy leakage in the PCNs [69, 70, 71]. If some/all the intermediate nodes who are involved in the payment path put the transaction on the chain as the closing transaction, anyone can recover the partial/full payment path. Additionally, intermediate nodes can collude with each other to identify part/all of the nodes that are involved in a given payment path.

For all the PCN, all the payment protocols should ensure that the payment is secured which means none of the participants loses their money if the payment channel is terminated unexpectedly. Additionally, to make the PCNs more attractive to privacy-sensitive users, the system should prevent others from knowing who is paying whom. PriPay [72] leverages the trusted hardware to encrypt the PCNs data at the server and uses oblivious algorithms to hide the access patterns. Nevertheless, PrivPay suffers from the low scalability and single point failure. TumbleBit [73] employs a trusted intermediary to achieve the privacy of the payment path, however, all the participants should trust the intermediary. SlientWhisper [71] employs the long-term keys and temporary keys schemes to ensure that the payment between each intermediate nodes are signed by different keys, thus, no one can link the transactions in a payment. However, the participants need to run a complicated key management process. MHTLC [20] avoids the complicated key management and can work in a trust-free environment. Our protocol aims to prevent the blockchain observer from knowing the payment value between the sender and the receiver. In our framework, unless all the intermediate nodes collude with each other, none can recover the payment path between the sender and the receiver. Compared with MHTLC, for each intermediate node, we reduce the communication size from 1650KB to 0.32KB; additionally, we avoid

the time-consuming zero-knowledge proof generation process, which consumes 309ms in MHTLC. Hence the contributions of our work are as follows:

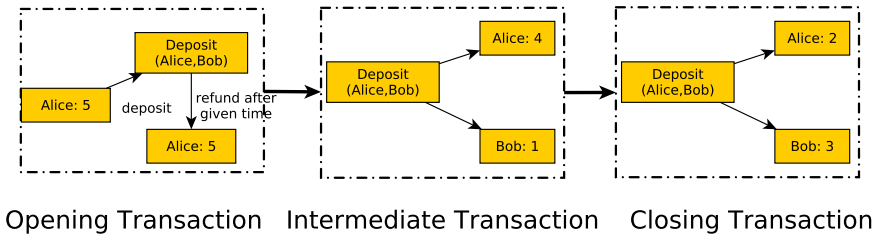
- We propose a new payment protocol called CHTLC, which hides the payment path from the view of payment participants and the observer who analyses the blocks that are committed on the chain. We also prove the security of CHTLC and show that CHTLC achieves the same level of security as MHTLC.
- We conduct computer simulations to show that our proposed CHTLC protocol is efficient both in time and space. Our experimental results indicate that in comparison with MHTLC protocol [20], our protocol is much more efficient in payment forwarding. That is, MHTLC spends 309 ms per user to generate the zero-knowledge proof required for the payment, whereas such procedure is avoided in our protocol. For each intermediate node, MHTLC needs to transmit $1,650$KB data between each node, while it is reduced to only 0.96KB data in our CHTLC protocol.

4.3 Background

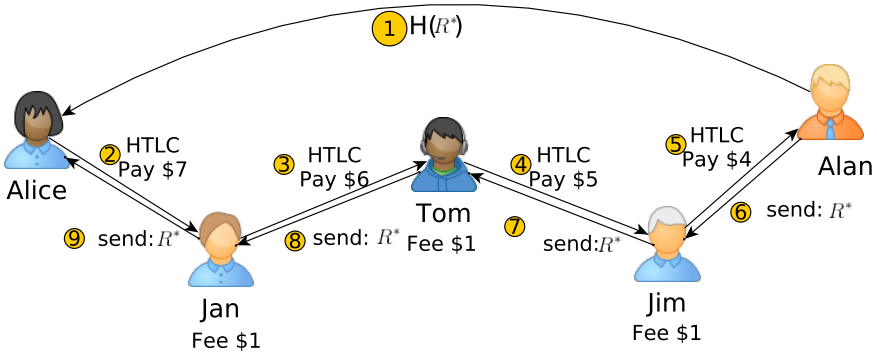
We first present the preliminaries required for understanding this paper. For the sake of readability, the notations which are used frequently are presented in Table 4.1.

4.4 Payment Channel

A payment channel [74, 75, 76] establishes a private peer-to-peer medium, ruled by a set of pre-set instructions, e.g., smart contract. The payment channel allows the involved participants to consent to the state updates unanimously by exchanging authenticated state transitions off-chain [77]. Fig.4.1(a) demonstrates how the payment channel is established between two entities. Before the payment is placed, they need to agree on a transaction known as the opening transaction to be put on the blockchain. In the opening transaction, two parties make the deposit to the "joint-account" of which the signatures from both parties are required to spend the money. After any party making the deposit in the opening transaction, both parties can have the transactions off the chain. Any party can close the channel by putting the latest transaction known as the closing transaction on the blockchain which shows how the money in the "joint account" is distributed back to them. To avoid any party publish the historical transaction unilaterally as the closing transaction



(a) Off-chain payment.



(b) HTLC payment channel networks.

Figure 4.1: Payment Channel networks.

of his favor to invalid the rest of the transactions [1], asymmetric revocable commitment schemes are employed [78]. Payment channels can be further extended with a special type of smart contract (e.g.,HTLC) [74, 79], Global Preimage [70]) that allows the participants to commit funds to a redeemable secret with an expiration time [78]. HTLC has already been integrated with Bitcoin Lightning network [75] and DMCs [74]. HTLC enables the cross-channel synchronization by allowing the sender to lock x coins in the senders' and receivers' joint account that are only redeemable if the contracts conditions are fulfilled [77]. We assume the sender Alice wants to pay Bob, x dollars, with the expectation y, the hash function Hash and the locked time t, the **HTLC** looks like the following [75]:

[1]For instance, the sender pays the receiver 10 times through the channel, however, the sender may put the first transaction on the chain to invalid the rest of the transactions.

Notation	Definition
B	Blockchain
t	Expiration time
f	fee
v	Channel capacity
$c_{\langle u,u' \rangle}$	A unique channel identifier between users u and u'
$\mathcal{F}$	Ideal functionality
$\mathcal{L}$	List of the off-chain payments
$\mathcal{C}$	List of the closed channels
$\lvert\mathsf{B}\rvert = t$	Time corresponds to the number of entries of the blockchain
h	Entry identifier in $\mathcal{L}$
u	A user
$H_i(x, r)$	Chameleon hash using public key of u_i with input x and randomness r

Table 4.1: Notations

HTLC(Alice, Bob, x, y, t) **If** B can provide the condition R^* such that Hash(R^*) $= y$ before t seconds, Alice pays Bob, x dollars.
Else if t seconds elapsed, Alice will be fully refund.

4.5 Payment Channel Networks

Though the off-chain payment path increases the transaction throughput, it has disadvantages like different parties need to setup and maintain a "joint account". It discourages some parties who may not be willing to make a deposit to the one they do not have the transactions frequently. To address this issue, HTLC based PCNs are proposed to avoid setting up the payment channels while preserving the high transaction throughput. In PCNs, the sender involves the nodes in the network to help them relay the payment by offering transaction fees as the award for forwarding the payment. Fig.4.1(b) demonstrates how Alice as a sender pays \$4 to Alan as a receiver through the payment network (we assume the transaction fees for all the intermediate nodes are set to \$1). Firstly, Alan sends hash value of a random secret R^* to Alice. In the second step, Alice creates a payment with Jan asking her to forward the payment to Tom. In the payment, "Alice is committing 7 of her channel balance to be paid to Jan if Jan releases the secret R^* in 10 seconds, or the money is refunded back to Alice if 10 seconds elapsed". On receiving this payment,

Jan knows that Tom can show her the secret R^* and helps her to get the money that Alice committed to her. She deducts \$1 from the payment amount as the transaction fee and creates a similar payment between herself and Tom. Finally, in step 5, Alan receives this \$4 payment commitment, as he is the one who has the secret R^*, thus revealing R^* to Jim to redeem the money from Jim's commitment. Since Jim can only redeem the payment from Tom by revealing the R^*, Jim sends the R^* to Tom and claims the money. In this way, the payment between Alice and Alan is settled without having a payment channel established directly.

In HTLC, every participant is assigned with a maximum time frame that they can pull the money from the sender to avoid any part suspends the channel by refusing forwarding the payment. Since the intermediate node needs to pay the descendant before they pull money from the predecessor, he needs to confirm that he is eligible to pull the money from the predecessor, otherwise, he should refuse to make a commitment with the descendant. Though HTLC is compatible with Bitcoin, it leads to serious privacy leakages. First, for any colluded nodes, by exchanging the $\mathsf{Hash}(R^*)$ they received and sent, they can tell whether they are involved in the same payment. Additionally, if they are the nodes that linked directly with the sender and receiver, they can release the identities of the sender and receiver. Second, if the HTLC commitments are broadcast on the blockchain, observers who are not involved in the payment can recover the payment path by identifying the transactions on the blockchain with the same $\mathsf{Hash}(R^*)$.

4.5.1 Syntax of PCN

We define the payment channel as a directed graph $\mathbb{G} := (\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ is the set of Bitcoin accounts and $\mathbb{E}$ is the set of currently open payment channels. A PCN consists of following algorithms [20].

- $\mathsf{OpenChannel}(u_i, u_j, \beta, t, f) \to \{0, 1\}$: This algorithm admits two Bitcoin addresses $u_i, u_j \in \mathbb{V}$, an initial channel capacity β, a timeout t, and a fee value f, if the operation is authorized by u_i, and u_i owns at least β bitcoins. Then, it creates a new payment channel $\left(c_{\langle u_i, u_j \rangle}, \beta, f, t\right) \in \mathbb{E}$, where $c_{\langle u_i, u_j \rangle}$ is a fresh channel identifier. Then this channel identifier is uploaded to B and returns 1. Otherwise, it returns 0.
- $\mathsf{CloseChannel}(c_{\langle u_i, u_j \rangle}, v) \to \{0, 1\}$: This algorithm gets a channel identifier $c_{\langle u_i, u_j \rangle}$ and a balance v as inputs. If the operation is authorized by both users, $\mathsf{CloseChannel}$ removes the corresponding channel from $\mathbb{G}$, includes the balance v in B, and finally returns 1. Otherwise, it returns 0.

- $\mathsf{Pay}\left(\left(c_{\langle u_1,u_2\rangle},\ldots,c_{\langle u_{n-1},u_n\rangle}\right),v\right) \rightarrow \{1,0\}$: This algorithm inputs a list of channel identifiers $\left(c_{\langle u_1,u_2\rangle},\ldots,c_{\langle u_{n-1},u_n\rangle}\right)$ which form a path from the sender u_1 to the receiver u_n and a payment value v. If each payment channel $c_{\langle u_i,u_{i+1}\rangle}$ in the path has at least a current balance $\gamma_i \geq v_i'$, with $v_i' = v - \sum_{j=1}^{i-1} \mathsf{fee}u_j)$, the Pay operation decreases the current balance for each payment channel $c_{\langle u_i,u_{i+1}\rangle}$ by v_i' and returns 1. Otherwise, none of the balances at the payment channels is modified and the Pay operation returns 0.

4.5.2 PCN Security and Privacy Goals

The security and privacy goals of our PCNs system are summarised as follows:

- Balance security: It guarantees that any honest user involved in a payment does not lose money even when the other involving participants are corrupted.
- Serializability: We require that the executions of PCN are serializable. That is, for every concurrent execution of Pay operation, there exists an equivalent sequential execution.
- (Off-path) Value Privacy: This ensures that for a Pay operation involving only honest users, corrupted users outside the payment path learn no information about the payment value.
- (On-path) Relationship Anonymity: Given two simultaneous successful Pay operations of the form $\left\{\mathsf{Pay}_i\left(\left(c_{\langle s_i,u_1\rangle},\ldots,c_{\langle u_n,r_i\rangle}\right),v\right)\right\}_{i\in[0,1]}$ with at least one honest intermediate user $u_{j\in[1,n]}$ corrupted intermediate users cannot determine the pair (s_i,r_i) for a given Pay_i with probability better than $1/2$.

4.5.3 Ideal World Functionality

To satisfy the security and privacy of our construction, we apply the ideal functionality as defined in [20]. This model captures Balance security, Serializability, Value privacy, and Relationship anonymity (see [20] for detailed discussions). The ideal world functionality $\mathcal{F}$ for PCNs consists of three main algorithms: OpenChannel, CloseChannel, and Pay. This is a trusted functionality, which interacts with the users and maintains the blockchain B using two lists $\mathcal{L}$ and $\mathcal{C}$. The adversary $\mathscr{A}$ is a probabilistic polynomial-time machine which is capable of adding users to the system and corrupt them at any time to gain the internal state of the users and all of incoming/outgoing communications.

- OpenChannel: This algorithm inputs $\big(\text{Open}, c_{\langle u,u'\rangle}, v, u', t, f\big)$ from a user u. The ideal functionality $\mathcal{F}$ checks $c_{\langle u,u'\rangle}$ for valid identifiers and not being duplicated, then sends $\big(c_{\langle u,u'\rangle}, v, t, f\big)$ to u'. If u' authorizes the operation, $\mathcal{F}$ appends $\big(c_{\langle u,u'\rangle}, v, t, f\big)$ to B and $\big(c_{\langle u,u'\rangle}, v, t, h\big)$ to $\mathcal{L}$, for some random h. $\mathcal{F}$ returns h to u and u'.
- CloseChannel: This algorithm inputs $\big(\text{Close}, c_{\langle u,u'\rangle}, h\big)$ from u or u'. In this framework, $\mathcal{F}$ checks B for $\big(c_{\langle u,u'\rangle}, v, t, f\big)$ and $\mathcal{L}$ for $\big(c_{\langle u,u'\rangle}, v, t, h\big)$ where $h \neq \perp$. If ($c_{\langle u,u'\rangle} \in \mathcal{C}$ *or* $t > |\mathsf{B}|$ *or* $t' > |\mathsf{B}|$) the functionality aborts. Otherwise, the ideal functionality $\mathcal{F}$ adds $\big(c_{\langle u,u'\rangle}, u', v', t'\big)$ to B and adds $c_{\langle u,u'\rangle}$ to $\mathcal{C}$. Then, $\mathcal{F}$ notifies both users involved with a message $\big(c_{\langle u,u'\rangle}, \perp, h\big)$.
- Pay: Given $\big(\text{Pay}, v, \big(c_{\langle u_1,u_2\rangle}, \ldots, c_{\langle u_{n-1},u_n\rangle}\big), (t_0, \ldots, t_n)\big)$ from u_1, the ideal functionality $\mathcal{F}$ performs the interactive payment protocol as presented in Algorithm 5.

As defined in Algorithm 5, $\mathcal{F}$ first ensures that the channel has enough capacity. Then, each user decides to accept or reject a payment. At the end, $\mathcal{F}$ updates the $\mathcal{L}$ and notifies the involving users.

4.6 Routing in PCNs

Discovering the payment path from the sender to the receiver is another research issue in the PCNs. For an effective routing protocol, it should work out the payment path from the sender to the receiver with a short time delay. It is also important that the routing protocol can be applied in the dynamic PCNs, in which nodes may join/leave the network frequently. Since it is impossible for the sender to store all the payment paths in the network, landmark routing technique [80] is proposed to maintain a set of paths between the sender and the receiver. The key idea is to provide a path from the sender to the receiver through an intermediate node called landmark node. However, the landmark nodes may not contain all the possible paths which may result in a payment path with low success probability [72, 69]. Flare [81] asks all the participants to maintain some of the path information of the neighbors. This design discourages the client which has limited computation source (e.g., smart phone payers), additionally, it cannot guarantee that the provided payment path has the relatively low transaction fee. SpeedyMurmurs [82] is another routing algorithm for PCNs which provides formal privacy guarantees in fully distributed settings. However, because of the overhead in privacy guarantees, it is not that effective in a dynamic PCNs.

Algorithm 5 Payment protocol in Ideal world

Input: $\left(\mathsf{Pay}, v, \left(c_{\langle u_1,u_2\rangle}, \ldots, c_{\langle u_{n-1},u_n\rangle}\right), (t_0, \ldots, t_n)\right)$
Output: updated $\mathcal{L}$

for $i = 2, \ldots, n$ **do**
  Sample h_i at random
  if $\left(c_{\langle u_i-1}, u_i'\right\rangle, v_i, t_i', f_i) \in \mathsf{B}$ **then**
    Send $\left(h_i, h_{i+1}, c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i+1}\rangle}, v - \sum_{j=i}^{n} f_j, t_{i-1}, t_i\right)$ to $u_{i\neq n}$ via private channel
    Send $\left(h_{n+1}, c_{\langle u_{n-1},u_n\rangle}, v, t_n\right)$ to the receiver
    for $\left(c_{\langle u_{i-1},u_i}\right\rangle, v_i', \cdot, \cdot) \in \mathcal{L}$ **do**
      if $v_i' \geq \left(v - \sum_{j=i}^{n} f_j\right)$ & $t_{i-1} \geq t_i$ **then**
        Add $d_i = \left(c_{\langle u_{i-1},u_i\rangle}, \left(v_i' - \left(v - \sum_{j=i}^{n} f_j\right)\right), t_i, \perp\right)$ to $\mathcal{L}$
      else
        Delete all d_i added in this phase to $\mathcal{L}$ and abort.
      end if
    end for
  else
    Abort
  end if
end for
for $i = n, \ldots, 1$ **do**
  Query u_i with (h_i, h_{i+1}) via private channel
  if $\exists\ u_j$ return $\perp$ s.t. all u_i returned $\top$ $(i > j)$ **then**
    $j = 0$
  end if
end for
for $i = j + 1, \ldots, n$ **do**
  Update $d_i \in \mathcal{L}$ to $(-, -, -, h_i)$
  Send (success, h_i, h_{i+1}) to u_i
end for
for $i = 1 \ldots, j$ where $j \neq 0$ **do**
  Remove d_i from $\mathcal{L}$
  Send $(\perp, h_i, h_{i+1})$
end for
return updated $\mathcal{L}$

4.7 Chameleon-hash Functions

Chameleon-hash functions [83] also known as trapdoor-hash functions are the hash functions which have a trapdoor allowing one to find arbitrary collisions in the domain of the functions. However, as long as the trapdoor is not known, Chameleon-hash functions are collision resistant. A chameleon-hash function **CH** consists of the following algorithms:

- CHSetup: This algorithm first chooses two large prime numbers p and q such that $p = kq + 1$ for an integer k. Then, selects g of order q in $\mathbb{Z}_p^*$. Finally, it outputs $\xi \in \mathbb{Z}_q^*$ as the private key sk and $y = g^{\xi} \mod p$ as the public key pk.

- CHash: On an input value x, this algorithm chooses a random value $r \in \mathbb{Z}_q^*$ and outputs $H_{\text{pk}}(x, r) = g^x y^r \mod p$.
- Trapdoor collision: Given $x, x', r \in \mathbb{Z}_q^*$ as input, this algorithm outputs r' such that $H_{\text{pk}}(x, r) = H_{\text{pk}}(x', r')$. This is done by solving for r' in $x + \xi r = x' + \xi r' \mod q$.

Definition 1 (Indistinguishability)**.** *For all pairs of message x and x', the probability distribution of the random value $H_{\text{pk}}(x, r)$ and $H_{\text{pk}}(x', r)$ are computationally indistinguishable.*

Definition 2 (Collision-Resistance)**.** *Without the knowledge of trapdoor key sk, there exists no efficient algorithm that, on input x, x', and a random string r, outputs a string r' that satisfy $H_{\text{pk}}(x, r) = H_{\text{pk}}(x', r')$, with non-negligible probability.*

4.8 CHTLC Construction Overview

We consider the following assumptions and research scope regards to our CHTLC.

- The underlying blockchain system which PCNs interacts with is secure and free from attacks. The security issues related with blockchain itself is beyond the scope of this paper.
- We focus on design of the CHTLC protocol, the efficiency of the routing protocols in PCNs is beyond our research scope. We applied the routing protocol proposed in Flare [81] in our CHTLC.
- All the intermediate nodes in the PCNs are reasonable nodes. That is, they are motivated by collecting transaction fees to forward the transactions unless they are corrupted. Reasonable nodes will not disclose their secret key for encrypted communication between other nodes or any message they received through private channels to the public.
- Some intermediate nodes might collude, while it is impossible for all the nodes that include in a payment path to collude with each other.
- The communication between each pair of nodes in the network is encrypted.
- The network is bounded by a weak synchronous communication [84]. This indicates that the participants in the network can achieve the same status within a suitable time t. This assumption can be achieved by applying a loosely synchronised clock among the users in PCN [85].

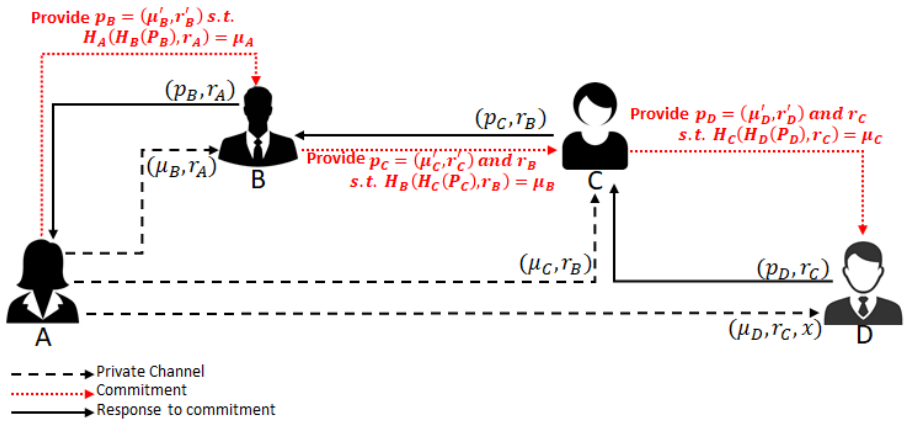


Figure 4.2: CHTLC diagram.

- The security of the individual node is beyond our research scope. The system cannot prevent the compromised nodes from paying other nodes through PCNs.

We now present a brief overview of CHTLC through an example. To make the discussion concise, we take as an example the payment between u_A and u_D with no direct payment channel to demonstrate the functionally of our protocol. To make the illustration simple we avoid the details of the messages (e.g., the payment value, the time for lock the deposit) that exchanged between two nodes and assume there exists a payment path of 4 nodes shown in Fig. 4.2(a). We define u_A as the $\mathring{Sdr}$ and u_D as the $\mathring{Rcv}$, the intermediate nodes are u_B, u_C and u_D. Firstly, u_A receives the random value x from u_D and calculates $\mu_D = H_D(x, r_D)$, $\mu_C = H_C(\mu_D, r_C)$, $\mu_B = H_B(\mu_C, r_B)$, and $\mu_A = H_A(\mu_B, r_A)$ with the public key of each node retrieved from B. Second, u_A sends (μ_B, r_A), (μ_C, r_B) to u_B and u_C respectively through the private channels. Now, the payment can be carried out as u_A makes a commitment to u_B saying if u_B can provide a value (p_B, r_A) such that $H_A(H_B(p_B), r_A)$ collides with μ_A given seconds[2], u_A pays u_B. User u_B firstly checks that the μ_B on the blockchain satisfies the condition that $\mu_A == H_A(\mu_B, r_A)$, otherwise aborts the payment. Since u_B does not know the input μ'_B, r_B such that $\mu_B = H_B(\mu'_B, r_B)$, u_B makes a commitment with u_C saying that if u_C can provide (p_C, r_B) such that $\mu_B == H_B(H_C(p_C), r_B)$, node u_B will pay u_C the promised money. Finally, u_D with its secret key and secret value x, generates the collision against $H_d(x, r_D)$ with (x', r') and sends $p = (x', r', r_C)$ to u_C. Similarly, u_C generates p_C that satisfy $H_B(H_C(p'), r_B)$ collides with μ_B and forwards (p_C, r_B) to u_B. Finally, all the nodes are paid with the promised amount of money.

[2]The money is locked within this time slot, if u_B fails to satisfy u_A, the money is refunded to u_A

4.8.1 CHTLC Construction

In this section, we discuss the details of the following operations:

$\mathsf{OpenChannel}(u_i, u_j, \beta, t, f)$: This operation establishes a direct channel between u_i and u_j. f indicates the transaction fee charged by u_i if u_i helps other users to forward the transaction to u_j. β indicates the total amount of money that u_i can transfer to u_j. To open a channel, u_i needs to create an opening transaction in which the input is β from u_i's wallet and the output is the joint-wallet in which the money need both u_i's and u_j's permission to be spent. To avoid the scenario that u_j's does not cooperate and β is locked in the contract, the money in the join-wallet will refund to u_i if has not been spent within a given time. After u_i puts the opening transaction and path information $c_{ij} = (e_{ij}, f_{ij}, \beta, t)$ on B, this payment channel is accepted by PCNs.

$\mathsf{CloseChannel}(c_{\langle u_i, u_j \rangle}, v)$: When node u_i wants to terminate the channel with node u_j, it needs to create a closing transaction. The input of the closing transaction is the joint wallet and the output of the transaction is the u_i's private wallet and u_j's private wallet. Let v as the balance in the joint wallet, v_i, v_j as the money that paid to node u_i and u_j respectively. The close commitment is invalid if $v \neq v_i + v_j$. u_i and u_j sign on the closing transaction and any node upload the close commitment on the B to finalize the close channel operation.

$\mathsf{Pay}\left(\left(c_{\langle u_1, u_2 \rangle}, \ldots, c_{\langle u_{n-1}, u_n \rangle}\right), v\right)$: The function pay pays v dollars from the u_1 to u_n (as the sender and receiver, respectively). The sender initiates the payment protocol by running the setup algorithm shown in Algorithm 6.

We assume that there exists a payment path denoted as $\mathbb{P} = \{u_1, u_2, \ldots, u_n\}$ where u_1 is the payer and u_n is the receiver. u_n samples a random value denoted as x and sends it to u_1 in a private channel. Sender u_1 retrieves the public key of all the nodes from B to generate the commitment value μ for each intermediate nodes.

Algorithm 6 *Setup*

function $Setup(\mathbb{P}, x, n)$
  for $u_i \in \mathbb{P}$ **do**
    $\mu_i \leftarrow GenMsg(x, u_i, n)$
  end for
  Return μ_i
end function
function $GenMsg(x, u_i, n)$
  for $i = n, ..., 1$ **do**
    Choose a random value $r_i \in Z_q^*$
    if $i = n$ **then**
      $\mu_i = H_i(x, r_i)$
    else
      $\mu_i = H_i\left(GenMsg(x, \mu_{i+1}, n)\right), r_i)$
    end if
    $i \leftarrow i - 1$
  end for
  Return μ_i
end function

We denote the cost of sending v dollars from u_1 to u_n as v_1 which is $v_1 = v + \sum_{i=2}^{n-1} fee(u_i)$. If u_1 does not have enough money to pay all the transaction fees, it aborts the process.

The detailed algorithm is shown in algorithm 7 (pay sender). u_1 finds out the length of the payment path, works out the path and locked time for each intermediate node. It forwards (μ_i, r_{i-1}) and path $c_{\langle u_i, u_{i+1}\rangle}$ to the intermediate nodes (line 9 in **Sender node** section algorithm 7). Then, it creates a HTLC commitment with u_2 saying that if u_2 can provide a pair (p_2, r_1) within t seconds such that $H_1(H_2(p_2), r_1) = \mu_1$, user u_1 pays v_1 dollars to u_2 (line 10 in **Sender node** section algorithm 7). Finally, u_1 sends message m_n to the receiver u_n to enable u_n claim the money from u_{n-1}.

For the intermediate node u_i, when it receives the payment commitment from the node u_{i-1}, it verifies that 1. it has enough money to fulfill the payment. 2. The correctness of the contract lock time t_{i+1}. 3. whether node u_{i-1} provides the valid commitment (line 2 in **Intermediate node** algorithm 7). Then it makes the HTLC commitment with the successor node u_{i+1}. Finally, u_i waits for u_{i+1} to send back $m* = (p_{i+1}, r_i)$ to claim the money from u_i (line 7 in **Intermediate node** algorithm 7). With the help of p_{i+1}, node u_i generate (p_i) (line 9,10 in **Intermediate node** algorithm 7) and sends (p_i, r_{i-1}) to node u_{i-1} and claims the money (line 11 in **Intermediate node** algorithm 7).

For the receiver u_n, once it receives the commitment μ_{n-1} from the previous node, it verifies the validity of the commitment and whether it can meet the condition within the time tn (line 1 in **Receiver** algorithm 7). Then it applies (x, r) with its secret key to generate p_n and send (p_n, r_{n-1}) to u_{i-1} to claim the money.

Since every intermediate node redeems the money by generating the collision with the parameter it received from the successor nodes, it has to wait for the successor node to redeem the money firstly. Finally, all the nodes are paid with the promised money.

Algorithm 7 Payment Protocol

Sender node:

1: $v_1 = v + \sum_{i=2}^{n-1} fee(u_i)$
2: **if** $v_1 \leq \text{cap}(c_{\langle u_1,u_2\rangle})$ **then**
3: $\text{cap}(c_{\langle u_1,u_2\rangle}) := \text{cap}\left(c_{\langle u_1,u_2\rangle}\right) - v_1$
4: $t_0 := t_{\text{now}} + \Delta \cdot n$
5: **for** $1 < i < n$ **do**
6: $v_i := v_1 - \sum_{j=1}^{i-1} \text{fee}\,(u_j)$
7: $t_i := t_{i-1} - \Delta$
8: $\mu_i \leftarrow Setup(\mathbb{P}, x, n)$
9: Send $m_i = (c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i+1}\rangle}, v_{i+1}, t_i, t_{i+1}, \mu_i, r_{i-1})$ to u_i
10: **end for**
11: **HTLC**$(u_1, u_2, v_1, \mu_1, t_1)$
12: Send $m_n = (c_{\langle u_{n-1},u_n\rangle}, v_n, t_n, \mu_n, r_{n-1}, r_n)$ to u_n
13: **else**
14: Abort
15: **end if**

Receiver node (m_n)**:**

1: **if** $H_n(x, r_n) = \mu_n$ and $t_n > t_{now} + \Delta$ **then**
2: Select x' and compute r' s.t. $H_n(x', r')$ collides with μ_n
3: $p_n \leftarrow (x', r')$
4: Send (p_n, r_{n-1}) to u_{n-1}
5: **else**
6: Abort
7: **end if**

Intermediate node (m_i)**:**

1: Read μ_{i-1} from B
2: **if** $v_{i+1} \leq \text{cap}(c_{\langle u_i,u_{i+1}\rangle})$ and $t_{i+1} = t_i - \Delta$ and $H_{i-1}(\mu_i, r_{i-1}) = \mu_{i-1}$ **then**
3: $\text{cap}(c_{\langle u_i,u_{i+1}\rangle}) := \text{cap}\left(c_{\langle u_i,u_{i+1}\rangle}\right) - v_{i+1}$
4: **HTLC**$(u_i, u_{i+1}, v_{i+1}, \mu_i, t_{i+1})$
5: **else**
6: Abort
7: **end if**
8: **if** receive $m^* = (p_{i+1} = (\mu'_{i+1}, r'_{i+1}), r_i)$ from u_{i+1} **then**
9: **if** $H_i(H_{i+1}(p_{i+1}), r_i) = \mu_i$ **then**
10: Select μ'_i and compute r'_i s.t. $H_i(\mu'_i, r'_i)$ collides with μ_i
11: $p_i \leftarrow (\mu'_i, r'_i)$
12: Send (p_i, r_{i-1}) to u_{-1}
13: **else**
14: Abort
15: **end if**
16: **else**
17: Abort
18: **end if**

4.8.2 Security Discussion

The proposed construction provides the same level of the security as [20] without requiring zero knowledge proofs. The security of CHTLC follows the security model introduced by [20] according to the universal composable (UC) security paradigm [86]. Let $\text{EXEC}_{\pi,\mathscr{A},\mathcal{E}}$ be the ensemble of the outputs of the environment $\mathcal{E}$ when interacting with the adversary $\mathscr{A}$ and parties running the protocol π. The UC-Security is defined as follows.

Definition 3. *A protocol π UC-realizes an ideal functionality $\mathcal{F}$ if for any adversary $\mathscr{A}$ there exists a simulator $\mathcal{S}$ such that for any environment $\mathcal{E}$, the ensembles* $\text{EXEC}_{\pi,\mathscr{A},\mathcal{E}}$ *and* $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ *are computationally indistinguishable.*

Theorem 1. *Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a Chameleon hash function modelled as a random oracle, then CHTLC UC-realizes the ideal functionality $\mathcal{F}$ (as defined in Section 4.5.3).*

Proof. We define the simulator $\mathcal{S}$ which simulates the real world execution protocol while interacting with the ideal functionality $\mathcal{F}$ as defined in Section 4.5.3. It also handles users corrupted by the adversary $\mathscr{A}$ and impersonates them until the environment $\mathcal{E}$ makes a corruption query on one of the users. Upon such query, $\mathcal{S}$ hands over to $\mathscr{A}$ the internal state of the target user and routes all of the subsequent communications to $\mathscr{A}$, who can reply arbitrarily. $\mathcal{E}$ does not expect any interaction with $\mathcal{S}$ regarding the operations exclusively among corrupted users. Moreover, $\mathscr{A}$ does not learn anything about communication between honest users that happened through secure channels. $\mathcal{S}$ simulates the random oracle H via lazy-sampling. The operations to be simulated for a PCN are described in the following.

$\mathsf{OpenChannel}\left(c_{\langle u_1,u_2\rangle}, \beta, t, f\right)$: Let u_1 be the user that initiates the request, the are two possible cases as follows:

- *Corrupted* u_1: $\mathscr{A}$ sends a request $\left(c_{\langle u_1,u_2\rangle}, \beta, t, f\right)$ on behalf of u_1 to $\mathcal{S}$ who in turn initiates a two-user agreement protocol with $\mathscr{A}$ to convey upon a local fresh channel identifier $c_{\langle u_1,u_2\rangle}$. If the protocol successfully terminates, $\mathcal{S}$ sends $\left(open, c_{\langle u_1,u_2\rangle}, \beta, t, f\right)$ to $\mathcal{F}$, which eventually returns $\left(c_{\langle u_1,u_2\rangle}, h\right)$.
- *Corrupted* u_2: $\mathcal{S}$ receives a message $\left(c_{\langle u_1,u_2\rangle}, v, t, f\right)$ from $\mathcal{F}$ engages $\mathscr{A}$ in a two-user agreement protocol on behalf of u_1 for the opening of the channel. If the execution is successful, $\mathcal{S}$ sends an accepting message to $\mathcal{F}$ which returns $\left(c_{\langle u_1,u_2\rangle}, h\right)$, otherwise it outputs $\perp$.

If the opening was successful the simulator initializes an empty list $\mathcal{L}_{c_{\langle u_1,u_2\rangle}}$ and appends the value $(h, v, \perp, \perp)$.

$\mathsf{CloseChannel}\left(c_{\langle u_1,u_2\rangle}, v\right)$: similar to Open Channel, there are two cases that might happen as follows (assuming u_1 is an initiator):

- *Corrupted* u_1: $\mathscr{A}$ sends a closing request on behalf of u_1 to $\mathcal{S}$ who fetches $\mathcal{L}_{c_{\langle u_1,u_2\rangle}}$ for some value (h, v, x, y). If such a value does not exist then it aborts. Otherwise it sends $\left(\text{close}, c_{\langle u_1,u_2\rangle}, h\right)$ to $\mathcal{F}$.

- *Corrupted* u_2: $\mathcal{S}$ receives $\left(c_{\langle u_1,u_2\rangle}, h, \perp\right)$ from $\mathcal{F}$ and simply notifies $\mathscr{A}$ of the closing of the channel $c_{\langle u_1,u_2\rangle}$.

$\mathsf{Pay}\left(\left(c_{\langle u_1,u_2\rangle}, \ldots, c_{\langle u_{n-1},u_n\rangle}\right), v\right)$: the users acting differently according to their role in the protocol, thus we consider the cases separately.

Sender: In order to initiate a payment, $\mathscr{A}$ must provide each honest user u_i with $m_i = \left(c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i+1}\rangle}, v_{i+1}, t_i, t_{i+1}, \mu_i, r_{i-1}\right)$ and notifies the receiver with $\left(c_{\langle u_n,u_{n+1}\rangle}, v_n, t_n, \mu_n, r_{n-1}, r_n\right)$.

If $t_i \geq t_{i+1}$ then $\mathcal{S}$ sends $\left(\mathsf{Pay}, v_i, (c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i+1}\rangle}), t_{i-1}, t_i\right)$ to $\mathcal{F}$ and sends $\left(\mathsf{Pay}, v, c_{\langle u_{n-1},u_n\rangle}, t_n\right)$ to the receiver, otherwise it aborts. For each intermediate user u_i the simulator confirms the payment only when receives from the user u_{i+1} a pair (p_{i+1}, r_i) such that $H_i(H_{i+1}(p_{i+1}), r_i)$ collides with μ_i. If the receiver is honest then $\mathcal{S}$ confirms the payment if the amount v corresponds to what agreed with the sender and if $H_n(p_n, r_{n-1}) = \mu_n$. If the payment is confirmed the entry $(h_i, v^* - v_i, \mu_i)$ is added to $\mathcal{L}_{c_{\langle u_{i-1},u_i\rangle}}$, where $(h_i^*, v^*, \cdot, \cdot)$ is the entry of $\mathcal{L}_{c_{\langle u_{i-1},u_i\rangle}}$ with the lowest v^*, and the same happens for the receiver.

Receiver: $\mathcal{S}$ receives some $\left(h, c_{\langle u_{n-1},u_n\rangle}, v, t_n\right)$ from $\mathcal{F}$, then it samples two random $x, r \in \{0,1\}^\lambda$ and returns $p_n = (x, r)$ to $\mathscr{A}$ the tuple $(p_n, H_n(p_n, r_{n-1}), v)$. If $\mathscr{A}$ returns $p^* = p_n$, then $\mathcal{S}$ returns $\top$ to $\mathcal{F}$, otherwise it sends $\perp$.

Intermediate user: $\mathcal{S}$ is notified that a corrupted user is involved in a payment with a message of the form $\left(h_i, h_{i+1}, c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i+1}\rangle}, v, t_{i-1}, t_i\right)$ by $\mathcal{F}$.

$\mathcal{S}$ samples three random values $r, x', r' \in \{0,1\}^\lambda$, sets $p' = (x', r')$ then sends the tuple $\left(c_{\langle u_{i-1},u_i\rangle}, c_{\langle u_i,u_{i-1}\rangle}, \mu_i = H_i(H_{i+1}(p'), r), \mu_{i+1} = H_{i+1}(p'), p', v, t_{i-1}, t_i\right)$ to $\mathscr{A}$. If $\mathscr{A}$ outputs r^* such that $H_i(H_{i+1}(p'), r^*)$ collides with μ_i, then $\mathcal{S}$ aborts. At some point of the execution the simulator is queried again on (h_i, h_{i+1}), then it sends r to $\mathscr{A}$ on behalf of u_{i+1}. If $\mathscr{A}$ outputs $\mu_i' = H_i(H_{i+1}(p'), r)$ which collides with μ_i, the simulator sends $\top$ to $\mathcal{F}$ and appends $(h_i, v^* - v, \mu_i', H_{i+1}(p'))$ to $\mathcal{L}_{c_{\langle u_{i-1},u_i\rangle}}$, where $(h_i^*, v^*, \cdot, \cdot)$ is the entry of $\mathcal{L}_{c_{\langle u_{i-1},u_i\rangle}}$ with the lowest v^*, otherwise it sends $\perp$.

The $\mathsf{OpenChannel}$ and $\mathsf{CloseChannel}$ algorithms are exactly the same as [20] and the indistinguishability argument is trivial. Thus, we exclude further discussion about them. For the payment, the sender provides the values to the user via private channel which mimics exactly the real-world protocol. Each user u_i confirms the transaction to $\mathcal{F}$ only once it receives the values p_{i+1} and r_i such that $H_i(H_{i+1}(p_{i+1}), r_i)$ collides with μ_i. The payment chain does not stop at a honest node (excluding the sender), thus the simulation does not aborts. The simulation aborts if adversary aim to interrupt the payment by outputting r^* such that $H_i(H_{i+1}(p_{i+1}), r^*)$ collides with μ_i without getting r from the simulator. According to *Indistinguishability* and *Collision-Resistance* properties of Chameleon hash function

Performance / Scheme	Key pair generation	Generate message	Verify message	Redeem payment
Multi-Hop HTLC	NA	309 ms	130 ms	Not Provided
CHTLC	8 ms	55 ms	20 ms	8 ms

Table 4.2: Performance comparison

as defined in Section 4.7, the probability that $\mathscr{A}$ be able to output p^* in such a way is negligible, hence $Pr[abort] \leq negl(\lambda)$.

4.9 Experimental Results

To evaluate the CHTLC protocol, we implement it in Golang language. We deploy our experiment on the server equipped with i7-4770k CPU and 32GB memory. We set the chameleon-hash key size as 2048 bits and the output of the hash function is 2048 bits. According to our observation, 90% of the payment can be finished within 10 nodes, thus we set our evaluation in the payment path consists of 10 nodes. We compare the size of data transmitted and the protocol time consumption in CHTLC with MHTLC proposed in [20].

4.9.1 Data size

In CHTLC, the sender needs to forward the secret value to each of the intermediate nodes which accounts for 2048 bits (256 Bytes). For the path that consists of 10 nodes, the total number of data sent by the sender is roughly 2.56 KB. However, according to the experiment demonstrated in [20], the sender needs to forward about 17MB data. In MHTLC protocol, the communication between the intermediate nodes is required to ensure the correctness of the received data from the sender while it is not necessary for our scenario. In conclusion, the data needed to be transmitted in much smaller than the MHTLC approach.

4.9.2 Time consumption

We evaluate the time consumption in CHTLC and make the comparison with MHTLC protocol in Table 4.2. For CHTLC, we need each node to generate the chameleon hash key pairs which takes about 8ms. In MHTLC, since it is based on a general hash function,

this step is not needed(As it is shown as not available in Table 4.2). It takes 55 ms (for the first intermediate node, it take 10 ms while the last node it takes 100 ms) on average to generate the message which sent to each intermediate node. However, in MHTLC, it takes 309 ms. To verify the correctness of the message(proof in MHTLC), CHTLC needs 20 ms while MHTLC consumes 130ms. To redeem the commitment from the previous node, our protocol needs 8ms to generate the collision of the committed hash value, while in [20], this evaluation is not addressed by the authors (As it is shown as not discussed in Table 4.2).

In conclusion, the data transferred in our protocol is much small than that transferred in MHTLC. our protocol also has a great advantage in the time consumption of generating/verifying the message (proof). It takes 8 ms to redeem the money from the commitment which is totally accepted by most of the scenarios.

4.10 Conclusion

In this paper, we propose a new payment protocol called CHTLC to address the payment path privacy issue in PCNs. With the help of chameleon hash function, no one can recover the payment path by analysing the payment commitment made by the payment participants. It is demonstrated by the evaluation that compared with MHTLC, our protocol consumes less bandwidth while much faster in transaction processing.

Chapter 5

Secured Blockchain-Based Applications

User privacy leakage has aroused the public's attention. For all the privacy preserving systems, at least one party should be trusted to manage the cryptographic schemes or the processing of the data. In the traditionally privacy preserving systems, a trusted third party needs to be presented to bootstrap the system and ensure all the participants obey the system scheme. To remove the trust on the single party, secure Multi-party Computation (MPC) scheme was firstly introduced by Andrew Yao in 1986 [87]. Though MPC transfers the trust on one single device to multi-parties, the protocol itself is difficult to extend to different application scenarios and suffers from high overhead. With the increase of the parties numbers, the communication overhead becomes unaccepted for most of the applications. Additionally, the MPC protocols suffer from the force-absent attack which means once one or some party that involved in the MPC refuse to collaborate, the MPC is suspend.

Blockchain which builds the trust not only on one single party but on all the participants who are involved in the system can be applied in the scenario that trust is difficult to be established. Additionally, thanks to the help of smart contract, compared with MPC, the protocol which needs multi-party to be involved can easily to be generalized in different application scenarios.

In this chapter, through three blockchain-based applications (IoTchian, blockchain-based Evoting system and privacy preserved payment channel network), we demonstrate how the blockchain can be applied to build the secured Evoting systems in which trust is difficult to be established.

5.1 IoTChain: Establishing Trust in the Internet of Things Ecosystem Using Blockchain [3]

5.1.1 Application Background

IoT has already reshaped and transformed our lives in many ways, ranging from how we communicate with people or manage our health to how we drive our cars and manage our homes. With the rapid development of the IoT ecosystem in a wide range of applications, IoT devices and data are going to be traded as commodities in the marketplace in near future, similar to cloud services or physical objects. Developing such a trading platform has previously been identified as one of the key grand challenges in the integration of IoT and data science. Deployment of such a platform raises concerns about the security and privacy of data and devices since their ownership is hard to trace and manage without a central trusted authority. A central trusted authority is not a viable solution for a fully decentralized and distributed IoT ecosystem with a large number of distributed device vendors and consumers. Blockchain, as a decentralized system, removes the requirement for a trusted third-party by allowing participants to verify data correctness and ensure its immutability. IoT devices can use blockchain to register themselves and organize, store, and share streams of data effectively and reliably. We demonstrate the applicability of blockchain to IoT devices and data management with an aim of providing end-to-end trust for trading. We also give a brief introduction to the topics and challenges for future research toward developing a trustworthy trading platform for IoT ecosystems.

The number of IoT devices has already exceeded the world population. With rapid advancement in hardware technologies, these smart devices have been applied in almost every aspect of our daily lives. A large amount of data is generated every second and data science research is actively defining algorithms to process such data to make and enact better decisions for us in our daily activities. For example, wearable smart devices such as smartwatches sense our heartbeat and blood pressure continuously to monitor our health condition; a smart fridge enables us to control the fridge remotely and plan a healthier diet; a smart air conditioner can track our living preferences and adjust the temperature automatically; an autonomous vehicle frees our hands and minds while making our journey safe.

One of the key grand challenges is how we ensure that users trust the IoT ecosystem to make the right decisions and act on them. This involves trusting devices, data and analytics, as previously identified in this column [88]. The focus of IoTchain is to analyze different research and technical issues related to managing trust using blockchain in a fully

decentralized IoT ecosystem.

Though these smart devices bring great convenience to us in our daily life, news such as US cell carriers (including AT&T, T-Mobile. and Sprint) selling access to customers' real-time phone location data to a little known company called Securus[89] raises public concerns about the risk of personal data leakage and abuse. Such news prompts a debate on whether these IoT devices are our friends or enemies. Trust is not a one-way street in the IoT ecosystem. Data analysts have concerns about the integrity of the data that data owners provide. At the same time, data owners are concerned about whether data analysts only use their data for its declared purposes. Additionally, data owners care about how to protect their own data (sometime captured by manufacturers) when IoT device ownership changes during its life. For example, what happens to the data of a car owner when an autonomous car is sold or the ownership of a car is changed?

Users find difficulties in enjoying the services provided by these smart devices if they don't meet the high security expectations from them. Some key challenges for building a trustworthy trading platform for IoT devices and data are outlined below:

Lack of trust among participating entities Trust is hard to achieve among different entities involved in IoT data processing due to the lack of a governance framework. As defined by NIST in its Network-of-Things (NoT) [90] report, which aims to define IoT formally, five key primitives are involved in real IoT applications: sensors (IoT devices for generating data), aggregators (edge, fog or mist infrastructure for aggregating data), communication channels (wired and wireless communication provided by communication service providers), eUtility (SaaS, PaaS, IaaS provided by clouds), and decision triggers (data analysis pipelines, decision making and enacting processes). Each of these primitives is likely to be supported by different service providers. How can they trust to each other? For example, in many cases, IoT device owners would not know who are the data processing entities or cloud service providers. Unless they have a mechanism to trust them to handle the data properly, they cannot use the services they provide to support the five primitives. More seriously, there is no standard agreement among different entities to define the data usage policy and it is hard to supervise the usage of personal data. The reliability and security of all entities providing five primitives is important to establish the trust.

Lack of data supervision and management In many applications, data collected by IoT devices is mostly maintained and processed by either the device manufacturer or a trusted third party. For example, consider an IoT application of monitoring chronic patients at home [91] where a patient is monitored for his activities (like exercises) and health (blood pressure, heart beats) using IoT devices. A service provider may share patient data with health data analytics, general practitioners, and related service providers,

including cloud data service providers. Patients have limited knowledge about how their data is processed and used. Additionally, when the data generated by IoT devices is transferred from one party to another, there is often no data integrity verification. The tampered data could result in misleading decisions and the source of the fraud is difficult to identify.

Lack of devices' life cycle management Every product undergoes a series of phases in its life cycle: design, sourcing of components, manufacturing, distribution, retail, repair, resale, and so on. For IoT devices, the management of devices and related data are critical because the data generated by devices at different phases should be isolated and well protected. To date, the visibility remains highly siloed and opaque across entities. For instance, patient health data generated by an artificial cardiac pacemaker is fragmented to the manufacturer, doctor, and insurance company. During the device's life cycle, a patient could change from one doctor to another; in such a scenario, the data accessibility should also be transferred. Currently, the data is fragmented and held by many entities and there is no regulation on auditing the ownership of IoT devices. Similarly, there is no guarantee that the data held by each of them is consistent with others. If the patient is transferred from one doctor to another, there is no mechanism on how the data accessibility is managed.

A lot of effort has been put into resolving the issue of trust among different entities in the IoT application ecosystem. Unfortunately, there is not a single reference scheme that satisfies all stakeholders. The key issue for the traditional solutions is that they all depend on a trusted third party, which has to be trusted by all stakeholders. Blockchain, as a new data-sharing model, addresses this issue by removing the need for a trusted third-party. It allows all stakeholders to participate in maintaining an immutable ledger in which the data is consistent among all stakeholders. Since the data on the ledger is immutable, we avoid the possibility that any participant tampers with the data by allowing all participants to verify the correctness of the data. In this article, we argue that with the help of the blockchain technology, the management of the IoT device life cycle and the corresponding data privacy can be enhanced in the following ways:

- instead of trusting a third party, IoT devices can exchange data through the blockchain;
- IoT devices and the data generated by IoT devices can be traced to avoid the manipula- tion of the data by malicious parties;
- different stakeholders can trust the validity and integrity of the data on the chain;
- the communication among different entities can be simplified as they only need to inter- act with the blockchain to retrieve/upload data;

- the deployment and operation cost of IoT can be reduced through a blockchain since there is no intermediary;
- computation-intensive operations like end-user authentication and access control can be processed on the blockchain instead of IoT devices;
- it is more convenient for the blockchain to maintain device and data ownership; and
- the distributed ledger eliminates a single point of failure within the ecosystem IoT devices and end users can interact with any blockchain nodes to access the data;

In the following sections, we present a blockchain-based application called trustchain to demostrate the feasibility of a trustworthy trading platform with the above stated capabilities.

5.1.2 Blockchain

Trust plays a critical role in information exchanges. It helps different entities deal with each other more effectively and is often a key element in any collaborative system. Traditionally, centralized trusted institutions such as banks or government agencies manage the trust problem. With the help of these centralized institutions, different entities can cooperate with each other with a certain degree of confidence. Blockchain, known as an electronic ledger, tries to replace such centralized institutions by distributing the trust in a decentralized network. In a blockchain system, the ledger is immutable and not held on a single server but among all servers in the network. The openness feature of blockchain allows any participant to modify the ledger under a set of rules dictated by a "consensus protocol." The "consensus protocol" requires the majority of the blockchain participants to agree on the modification of the ledger to ensure the trustworthiness of the blockchain. Once a new consensus is achieved, all participants update their own ledger simultaneously. If any of the participants violates the consensus protocol to propose a new data entry, the network treats that entry as an invalid one.

Practically, transactions are bundled together and submitted to the blockchain as a block. Cryptographic techniques are applied to link all blocks in a deterministic order. The cryptographic algorithm also guarantees that the blocks are immutable, which means that once a block is appended to the chain, it cannot be tampered with.

We take Hyperledger Fabric as an example blockchain platform to illustrate how a smart contract service works in the blockchain. Figure 5.1 shows how a smart contract is deployed

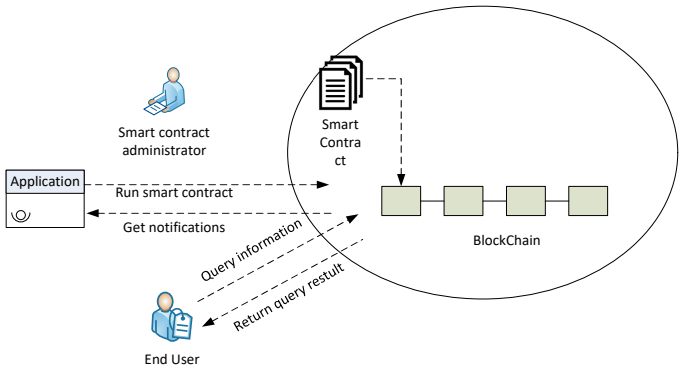


Figure 5.1: Smart contract on blockchain.

on the blockchain. First, the smart contract administrator needs to compile the smart contract application into a binary code so that it can be executed on the Hyperledger fabric. The administrator then deploys the smart contract on the blockchain. The application receives status notifications when there is any change. For instance, when the smart contract is deployed successfully, the application receives a message saying that the smart contract is now running on the blockchain. Finally, end users can access the service through the interface provided by the blockchain.

There are typically three different types of consensus protocols. The first is the PoW [92], adopted by cryptocurrencies like Bitcoin [75] and Zcash [14]. For the PoW protocol, all participants are competing with each other to win the block proposal by solving a specific math puzzle. The first participant to find the solution authorized to propose the transactions in the block and the rest of the participants copy this block to their own chain. A PoW based blockchain can provide an open environment, which allows anyone to join/leave the blockchain freely. However, in this protocol, a high amount of energy is consumed by each participant to solve the math puzzle. As a result, they have a poor throughput. The second type of consensus protocol is based on BFT [48], which is adopted by blockchain systems like Hyperledger Fabric [8]. The size of the network of a BFT-based blockchain is relatively small. As a result, the majority of BFT-based blockchain systems are permissioned blockchains, in which only authorized users can participant in block generation and verification. The last one is the PoS [93], which employs a certain number of nodes to generate the blocks on behalf of the whole network. Typical examples of PoS-based blockchain systems are Ouroboros [53], Neo [94], and Redd-coin [95]. However, in such systems, rich nodes have more chances to generate blocks. To address such problems, Bitcoin-NG[96] advocates applying a hybrid consensus protocol, which combines the advantages of the PoW and PoS. GHOST [18], SPECTRE [62], and MESHcash [61] are

recent proposals for increasing the throughput by replacing the underlying chain structure with a tree or a directed acyclic graph (DAG) structure. These protocols still rely on the Nakamoto consensus using PoW. By carefully designing the selection rules between branches of the trees/DAGs, they are able to substantially increase the throughput [47].

Since the blockchain removes a centralized trusted party and allows a participant to verify the correctness of the data on the blockchain, it is widely applied in two domains—cryptocurrency and the smart contract [97]. The great success of Bitcoin [13], Litecoin [98], Zcash [14], Ripple [67], and EOS [56] demonstrates the potential market value of the blockchain technology. The smart contract is another application based on blockchain technology. In essence, the smart contract is a service that links different entities together to construct a system to achieve dedicated functions. The approximate turning-complete feature provided by the smart contract allows the majority of existing programs to migrate to the blockchain.

Three key features that the blockchain technology brings to the industry and acdamia are:

Openness: Trustworthiness is always a key issue for systems that involve multiple parties. All communications between parties are based on a certain kind of trust assumptions. Blockchain technology is so revolutionary that it allows exchanging value directly between parties without them trusting each other. The openness feature allows anyone interested in the system to join and verify the correctness of the data. Because the data on the chain is immutable, it resolves concerns that the data owner might tamper with or modify the data in the future.

Robustness: Denial of service and a single point of failure are common issues for existing centralized systems. If the centralized servers are under attack, the quality of service might be affected and system security could be compromised. However, since every participant holds a copy of the data, and the network size could be large, it is impossible for an adversary to attack the blockchain system by compromising the majority of the distributed blockchain servers.

Cooperation: Blockchain enables a new cooperation pattern among multiple parties in which untrusted parties can exchange data more confidently by hosting the servers locally to construct a blockchain network. For example, take an electronic voting system, when the voting is conducted in a traditional way, all voters and stakeholders should agree on a trusted third party to organize the voting. Blockchain removes this trusted party by allowing all stakeholders to participate in the voting administration. That is, all stakeholders can verify the correctness of the voting result by looking up the data on the blockchain node held by themselves. There are two paradigms for blockchain resource management:

	permissioned Blockchain	Permissionless Blockchain
Operational costs	Depends on the redundancy requirements	High (Bitcoin estimate $657,000,000 per year in 2017 at $1000/BTC)
Interoperability	Poor	Excellent
Transaction Throughput	Good	Poor
Data Privacy	Good	Poor
Scalability	Poor	Good
System robust and resilience	Fair	Good

Table 5.1: Comparison between permissioned and permissionless blockchain

permissioned blockchain and permissionless blockchain. For the permissioned blockchain, a membership service exists that asks all parties who want to contribute in the blockchain maintenance to register with the blockchain system. Hence, only authorized users can access the blockchain. In contrast, for the permissionless blockchain, everyone can access the data on the blockchain and participate in the blockchain management without registering. We make a comparison between permissioned and permissionless systems in Table 5.1. We briefly describe them below.

Permissionless blockchain: The advantages of a permissionless blockchain are: 1) it has an open network to enable anyone to join/quit the protocol freely; 2) the network typically has an incentivizing mechanism to encourage more participants to join the network; and 3) it is suitable for cryptocurrency and applications that do not have strict privacy requirements. However, it consumes a lot of power to maintain the distributed ledger at a large scale for PoW based systems and the trust of the blockchain is hard to achieve for PoS based systems. Furthermore, very limited transaction privacy is preserved since any nodes in the permissionless blockchain can have a copy of all transactions.

Permissioned blockchain: The advantages of a permissioned blockchain are: 1) all blockchain participants are registered and verified by the protocol administrator and as a result, it is easy to identify nodes that do not comply with the protocol; 2) since the public has no access to the blockchain, privacy is preserved; and 3) since the blockchain administrator can control the network size by controlling the number of nodes involved in the blockchain, the permissioned blockchain usually has a high transaction throughput. However, the permissioned blockchain has a number of disadvantages: 1) the public may have low confidence in the correctness of the blockchain because they have no access to the verification of the data on the chain; 2) some stakeholders may collude with each other to make some transactions invalid; 3) participants need to follow a series of strict policies to join/quit the protocol (i.e., a membership server should assign/withdraw its access policy)

and 4) some protocols (e.g., practical BFT) are based on assumptions that two or three of the total nodes are always online.

5.1.3 Trustchain — a platform for IoT device and dat trcking and trading

In this section, we provide a concrete example of how the blockchain is applied to enhance the trust in a practical scenario to track and trade IoT devices and the corresponding data.

The popularity of wearable devices is increasing and people are regularly upgrading their IoT devices, while few of them understand how to dispose of their out-of-date devices. Trade-in or resale to another customer usually means the potential or accidental transfer of all personal data on the device to the new buyer, resulting in personal data leakage. On the other hand, when a device is put up for sale, the manufacturer needs to trace the ownership of the device to provide a warranty to the correct customer and recall the device if any defects are found. Current faulty airbag recalls on vehicles is a good example—many consumers are not aware that their vehicles have been recalled.

Applying the blockchain technology in the above scenario, we demonstrate how our IoT device and related data tracking and trading system resolves the trustworthiness issue in the IoT ecosystem. Four entities are involved in our system: 1) manufactures, who sell the products to retailers; 2) retailers, who buy the products from manufactures and sell them to customers; 3) customers, who consume the service provided by products; and 4) data analysis companies that buy the personal data for analysis.

For the IoT device and data tracking and trading system, we would like to have the following functions: 1) a manufacturer can trace the status and ownership of the device during its lifecycle; 2) a customer can transfer the ownership of his/her devices to another customer; 3) a customer can share/sell the data generated by his/her IoT devices; and 4) a smart contract that only allows the data owner to sell his/her own data; once the IoT device is sold, he/she cannot access the data generated by that device any longer.

The logical architecture of our tracking and trading system is shown in Figure 2. It demonstrates how cell phones, as IoT devices are transferred from a manufacturer to a retailer, a retailer to a customer, and finally, a customer to another customer. With the help of a smart contract, the manufacturer can transfer the ownership of the cell phone to the retailer. When Alice as a customer purchases this item, a record that corresponds to this purchase is appended to the smart contract, which demonstrates that the cell phone is owned by the customer, Alice. If Alice agrees to sell her cell phone as a used device to

Bob, the smart contract can inform the manufacturer that the device is owned by Bob. The manufacturer then provides any remaining warranty service to Bob. Since the device is transferred to Bob, Bob can sell the personal data generated by his cell phone to a data analysis company. At the same time, he has no rights to handle the data generated by the same device previously under the ownership of Alice.

Fig. 5.2(a) shows the interactions between different entities that are carried out through a smart contract. Since the smart contract is regarded as an independent trusted party, no entity can cheat others or modify the existing data related to this device.

The above case studies can be implemented in both permissioned and permissionless blockchains. Let us first consider the permissioned blockchain. We employ Hyperledger Fabric as a private blockchain that only allows relevant stakeholders to store IoT devices and the data ownership information. Users who own IoT devices can verify the manufacturer of those devices and sell the data generated by them. Since the public does not own the IoT device, they cannot trace any information related to the device. In practice, to preserve the data privacy, we only allow the manufacturer, retailer, and government consumer affairs office to host the blockchain validation nodes.

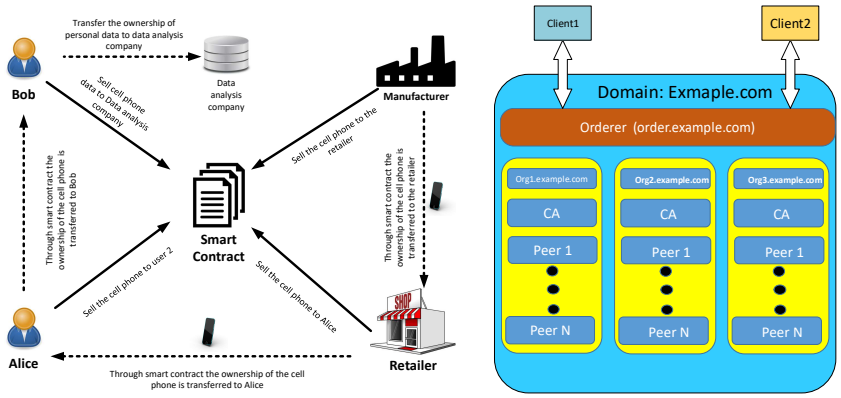
dddd The logical structure of Hyperledger Fabric is shown in Figure 3. It consists of the following components.

Client: The client represents the entity that acts on behalf of an end-user. It must connect to a peer to communicate with the blockchain. Clients create and thereby invoke transactions.

Peer: The peer receives the ordered state updates in the form of blocks from the ordering service and maintains the state and the ledger. The peer nodes are held by different stakeholders to ensure that the data on the blockchain are verified by all stakeholders to avoid any party tampering with or creating an incorrect block on the chain.

Ordering service: This service provides a communication channel to clients and peers, and offers a broadcast service for messages containing transactions. The channel outputs the same messages to all connected peers in the same logical order.

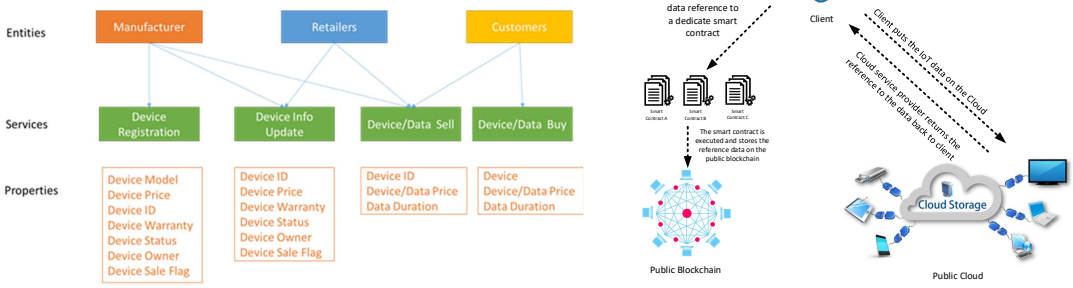
Certificate Authority (CA) server: The server is responsible for creating user/server certificates and verifying servers' validity in the network. Peer nodes in the blockchain network also ask the CA server to verify the identity of peer nodes.



(a) Device tracking and data trading system.

(b) Hyperledger Fabric structure.

Figure 5.2: Smart contract on blockchain and device tracking and data sharing trading system.



(a) Smart contract logic.

(b) Our reference implementation.

Figure 5.3: Hyperledger Fabric structure and our reference implementation.

5.1.4 Smart Contract Logic

In this section, we explain the portal and the interfaces of different entities in our system to illustrate the logic of our smart contract in Fig. 5.3(a).

The smart contract runs atop the Ethereum blockchain as a form of distributed computation. When the contract is interacted with, either by a retailer or a manufacturer selling/buying a device on the blockchain or by an end user seeking to buy/modify a device, they call one of the functions outlined below. Both inputs and operations of these functions are then computed across the whole blockchain. An individual who invokes the function pays for all nodes in the network to perform that function through the use of "gas." Gas is the execution fee that scales according to the amount of computational power needed to perform the invoked function by all nodes doing the computation. This execution fee is the incentive for nodes in the system to actually perform the necessary computations to ensure the contract is obliged by the blockchain. As the computation is distributed and performed by all nodes in the Ethereum blockchain, the need for a trusted authority to validate the operation is superseded by the use of the consensus protocol. By needing all nodes performing the computation to agree to the output, the ability of bad actors and malicious nodes to tamper with the operation of the contract is entirely negated. However, due to the nature of the contract being executed across the entire network, the contract will not be executed unless the blockchain is sure that the execution will be successful. This prevents wasted operations on the network. The consensus-based nature of this security model means that an attack to the system requires more than 50% of the network to inject any malicious transactions into the blockchain, making it a highly secure decentralized autonomous marketplace (when the continuous expansion and the current size of the Ethereum network are considered).

Fig 5.3(b) shows an example of how cloud storage can be used. We next describe current implementation of the services in our platform.

Manufacturers are the entity that produce new IoT devices. They have three functions: Registration, Update, and Sale. The registration service is responsible for registering a newly made device onto the blockchain. Actually, it creates device information containing the metadata related to the device including device model, device ID, and device warranty information. Manufacturers also need to update the device state, which indicates the device can be traded on the market. This update service allows manufacturers to update the device information including the device owner, device price, device warranty, and device status. For instance, when a manufacturer finds a specific product has a safety defect and wants to recall the product, it can set the device status to untradeable. Customers need to return them to the manufacturer; otherwise, they lose the rights to trade them in the

platform. The sale service allows the manufacturer to put a specific device on the market for sale. A retailer can buy the device from manufacturers. This service can only operate when the manufacturer sets the device for sale indicator as true.

Retailers have the following three functions: Buy, Update and Sell. The buy service enables retailers to buy a specific device from manufacturers with an expected price. Once the agreement is made between a retailer and a manufacturer, the manufacturer transfers the product's ownership to the retailer. The update service allows the retailer to update the device information including the device owner and the device price. This service allows the retailer to set the price for a given device and allows the retailer to transfer the ownership of the device to another retailer or customer when the ownership transferring agreement is achieved. The sell service allows the retailer to put the specific device on the market, enabling customers to buy the device from the retailer. The retailer needs to set a reasonable price for the device before putting it on the market.

Customers also have three functions: Buy, Update and Sell. The buy service enables a customer to buy a specific device or data from a retailer with the expected price. Once the agreement is achieved between the customer and retailer, the retailer transfers the device's or data's ownership to the customer. The update service allows the customer to update the device/data information including the device/data owner, device/data price. The device/data owner can also set the price for a given device/data and allow the device/data owner to transfer the ownership of the device to another customer when the ownership transferring agreement is achieved. The sell service allows the customer to put a specific device/data on the market; thus, customers can buy the device/data from another customer.

5.1.4.1 Future Research Challenges in Blockchain-Based Ownership Managemt Systems

Through the case study, we demonstrated how a blockchain can be applied to develop a trusted device and data trading platform for the IoT ecosystem where different entities can cooperate with each other. However, blockchain is not a panacea to resolve the trust issue in IoT environments. There are some challenges that still need to be studied. We outline some research challenges and potential future works below:

Data privacy: Currently there is a dilemma/trade-off between public verifiability and privacy. In a trustworthy data trading platform, the trust is established by providing verifiability. The challenge is how to protect privacy of data, device and individuals without losing the verifiability property. One simple solution is to encrypt all data on

the blockchain. This might help to address the privacy problem, but the data cannot be verified by other validation nodes. One potential avenue to further this research is use of Zero-Knowledge-Proof (ZKP). However, the computation overhead is often cited as a key problem in using ZKP.

Delegating trust: Another challenge of any effective trust model is trust delegation. In practice, it means how one can practically delegate trust to someone else. For example, Bob brings a device home and he claims/registers it as his device, perhaps with a straightforward method. Bob is the sole person who can control it and is privy to the data it collects. In certain circumstances, Bob may want to give others access to his device. There needs to be a scheme to ensure that operation can be done reliably and Bob has a full understanding of the implications. The challenge is how to support a trust delegation function without violating underlying security and privacy.

Dynamic access control: The access control mechanism is widely used to control access to the data. These methods have been directly applied to IoT environments. However, the IoT system is very dynamic and it operates in a context. For example, an emergency doctor might be able to access the IoT health data or the IoT health data becomes accessible from the IoT devices worn by patients when they are in the emergency room. In essence, it is difficult to predefine all potential access control rules. The device itself should have a mechanism to generate access control rules dynamically based on the contextual information. For example, a drunken driver would not be able to start the car. Though there has been some progress in this area, further research is needed to build a reliable dynamic access control mechanism for IoT applications.

IoT device identification: For the data and device trading platform to function properly, IoT devices that are trusted need to be identifiable. This requires an easy to use identity management system to be made available for all IoT devices at all times. However, the identity management systems (such as username/password pairs, and X.509) are invented for general purpose identification and are thus inadequate and rarely address many known issues that exist in the IoT environment. There is no defined and accepted standard for device identification management in the IoT environment. One of the key features of such a device identification mechanism should be automatic discovery of devices. The challenge is as soon as the device is powered and in operation, it would be discoverable (without violating the underlying security and privacy).

Human-centric trust model: Human-centric trust models are another research topic, which means a trust model aimed at giving effective administration of security and privacy not to computing professionals, but to average users. This is a cross-cutting topic to all of the challenges stated above. For example, a human-centric trust model can

be designed for people to sensibly delegate the controls of data and device to others with full understanding of security and privacy implications. The goal of a human-centric trust model is to let the service itself evaluate the security risks and apply the security policies according to the potential attack; thus, an average person can enjoy the same device security levels as security professionals.

Developing a holistic benchmarking kernel: Understanding performance bottlenecks of blockchain-based large scale IoT application systems remains a challenge, hence it is useful to identify benchmark kernels that are relevant for testing particular aspects (e.g., overlay networking, consensus protocol, querying) of the blockchain. Existing benchmarking literature in the context of IoT systems is limited as they are largely focused on studying the scalability of data processing programming models. For instance, the benchmark (kernels) that are available in context IoT systems focus on following data processing programming model aspects (not applicable to benchmarking performance of blockchain):

- Edge layer. TPCx-IoT (for data aggregation, real-time analytics & persistent storage), Google ROADEF & Linear Road benchmarks (for stream processing).
- Cloud layer. TeraGen, TeraSort, TeraValidate, and BigDataBench (for batch-oriented processing).

Hence, creating benchmark kernels that can test different aspects of the blockchain, and more importantly, identify performance bottlenecks and dependencies need more attention from the research community.

Scalable Search and Communication Developing a scalable protocol for searching data blocks and smart contracts within a large scale blockchain network remains a challenge. Existing search and consensus communication protocol adopted by the state of the art blockchain networks are based on a complete broadcast routing algorithm. Drawbacks for broadcast-based routing include high network communication overhead and non-scalability. Hence, the future investigation will need to develop scalable methods for search and consensus communication protocol. To this end, one possible research direction can be to interconnect peer nodes in the blockchain network based on Distributed Hash Tables (DHTs) overlay. In contrast, to broadcast based peer-to-peer systems, DHT overlays (e.g., Chord, Pastry, and Tapestry) have been proven to be more scalable as the size of the network grows.

Solutions to these research challenges will form a building blocks and contribute to components required for building a trustworthy data and device trading platform. We believe the blockchain technology is key to finding appropriate solutions to these challenges as it provides a basic foundation for trust in an untrusted, distributed IoT environment.

5.1.5 Conclusion

Many our lives are influenced by IoT-driven data science applications, ranging from precision health/medicine to self-driving cars. While enjoying the convenience and benefits brought by IoT devices, we must also face the challenge of trusting such IoT ecosystems. Traditionally, a trusted party performs a supervisory role in data collection and analysis. Blockchain, which is designed to remove the trusted third-party in a decentralized system, is an ideal solution to resolve the trust issue in IoT ecosystems. With the help of blockchain, different parties can trust and verify the data. Additionally, the ownership of IoT devices and their related data can also be traced. Though the blockchain can resolve the trust issue, it is not the panacea to every IoT challenge. A number of research challenges need to be addressed including data security and privacy on the blockchain, trust delegation, device identification, discovery, and authentication.

5.2 Blockchain-based Privacy Preserving voting System [4]

5.2.1 Abstract

Cryptographic techniques are employed to ensure the security of voting systems in order to increase its wide adoption. However, in such electronic voting systems, the public bulletin board that is hosted by the third party for publishing and auditing the voting results should be trusted by all participants. Recently a number of blockchain-based solutions have been proposed to address this issue. However, these systems are impractical to use due to the limitations on the voter and candidate numbers supported, and their security framework, which highly depends on the underlying blockchain protocol and suffers from potential attacks (e.g., force-abstention attacks). To deal with two aforementioned issues, we propose a practical platform-independent secure and verifiable voting system that can be deployed on any blockchain that supports an execution of a smart contract. Verifiability is inherently provided by the underlying blockchain platform, whereas cryptographic techniques like Paillier encryption, proof-of-knowledge, and linkable ring signature are employed to provide a framework for system security and user-privacy that are independent from the security and privacy features of the blockchain platform. We analyse the correctness and coercion-resistance of our proposed voting system. We employ Hyperledger Fabric to deploy our voting system and analyse the performance of our deployed scheme numerically.

5.2.2 Introduction

Voting plays a significant role in a democratic society. Almost every local authority allots a significant amount of budget on providing a more robust and trustworthy voting system. Cryptographic techniques like homomorphic encryption and Mix-net [99] are usually applied in contemporary electronic voting systems to achieve the voting result verifiability while preserving voters' secrecy. However, incidents like a security flaw that has erased 197 votes from the computer database in the 2008 United States elections [100] and the compromise of 66,000 electronic votes in the 2015 New South Wales (NSW) state election in Australia [101] raise the public concerns on the security of electronic voting systems. For voting systems based on bulletin (e.g., [102, 103]), one of the major concerns is whether the voting result that is published on the bulletin can be trusted. Blockchain with the growing popularity and remarkable success in cryptocurrency provides a new paradigm to achieve the public verifiability in such electronic voting systems.

In a blockchain-based system, there is no trusted centralised coordinator; instead, each node that is involved in the blockchain system holds the data block locally. Based on the assumption that the decentralised consensus protocol is secure and a sufficiently large proportion of blockchain network nodes are honest, the blockchain can be thought of as a conceptual third party that can be trusted for correctness and availability [104]. The data on the blockchain is append-only and any operation that alters the data in any block violates the blockchain consensus rule and are rejected by the blockchain network.

Recently a number of blockchain-based electronic voting systems have been developed by exploiting its inherent features. These systems can be classified into three broad categories. (1) Cryptocurrency based voting systems (e.g., [105, 106, 107]). The ballots to a candidate are based on the payment he/she receives from the voters; the problem with such systems are malicious voters may refuse to "pay" the candidates to retain the money. Furthermore, a centralised trusted party, who coordinates the payment between the candidates and voters must exist. (2) Smart contract based voting system [108], which only supports two candidates and the voting is restricted to limited number of participants. Furthermore, it requires all voters to cast their ballots before reaching an agreement on the voting result. (3) Using blockchain as a ballot box to maintain the integrity of the ballots [109, 110].

In summary, the security of these blockchain-based systems highly depends on the specific cryptocurrency protocol they employed. Additionally, these voting systems can only work with a specific blockchain platform, and support a limited number of candidates and voters. Based on our observations and studies, we believe that any blockchain-based voting systems should have the following three features: (1) Platform-independence — this means

the changes in the underlying blockchain protocols should not affect the voting schemes. (2) Security framework — the voting system should be implemented with comprehensive security features (the detail of security features are discussed in Section 5.2.6). The nature of the blockchain allows everyone to obtain the data on it; thus, the comprehensive security features have critical importance to ensure that the ballots are fully secured on the blockchain. (3) Practical — it should be scalable, which means the large amount of ballots casting and tallying can be finished in a reasonable time.

Our Contributions: In this paper, we propose an electronic voting (evoting) system that supports the above identified three features as follows.

1. *Our voting system does not depend on a centralised trusted party for ballots tallying and result publishing.* Compared with traditional voting systems, which highly depend on a centralised trusted party to tally the ballots and publish the result, our voting system takes the advantage of a blockchain protocol to eliminate the need for a centralised trusted party.

2. *Our voting system is platform-independent and provides comprehensive security assurances.* Existing blockchain-based voting systems highly depend on the underlying cryptocurrency protocols. Receipt-freeness [21] and correctness of the voting result are hard to achieve (we analyse the blockchain-based voting system explicitly in Section 2). The security of our voting system is achieved by cryptographic techniques provided by our voting protocol itself, thus, our voting system can be deployed on any blockchain that supports smart contract. To achieve the goal of providing a comprehensive security, we employ the Paillier system to enable ballots to be counted without leaking candidature information in the ballots. Proof-of-knowledge is employed to convince the voting system that the ballot cast by a voter is valid without revealing the content of the ballot. Linkable ring signature is employed to ensure that the ballot is from one of the valid voters, while no one can trace the owner of the ballot.

3. *Our voting system is scalable and applicable.* In order to support voting scalability, we propose two optimised short linkable ring signature key accumulation algorithms given in algorithm 8 and algorithm 9 to achieve a reasonable latency in large scale voting. We evaluate our system performance with 1 million voters to show the feasibility of running a large scale voting with the comprehensive security requirements.

The rest of the paper is organised as follows: we discuss the cryptographic techniques applied in voting systems and analyse some typical voting systems in Section 5.2.3. Cryptographic primitives and our voting protocol are presented in Section 5.2.4 and 5.2.5,

respectively. We analyse the correctness and security of our voting system in Section 5.2.6. In Section 5.2.7, we deploy our voting system and analyse its performance.

5.2.3 Related work

Secure evoting is considered as one of the most difficult problems in security literature as it involves many security requirements. To satisfy these security requirements, cryptographic techniques are mostly applied in constructing a secure evoting system. In this section, we discuss the existing voting systems based on traditional public bulletin and blockchain technology.

5.2.3.1 Public bulletin based voting systems:

In the following, we outline the key cryptographic techniques used in public bulletin based voting systems and how such techniques address the corresponding security requirements.

- **Homomorphic encryption:** Homomorphism feature allows one to operate on ciphertexts without decrypting them [111]. For a voting system, this property allows the encrypted ballots to be counted by any third party without leaking any information in the ballot [112, 113, 114]. Typical cryptosystems applied in a voting system are Paillier encryption [115, 116] and ElGamal encryption [117, 107].
- **Mix-net:** Mix-net was proposed in 1981 by Chaum [99]. The main idea of mix-net is to perform a re-encryption over a set of ciphertexts and shuffle the order of those ciphertexts. Mix node only knows the node that it immediately received the message from and the immediate destination to send the shuffled messages to. The voting systems proposed in [102, 118, 119] apply mix-net to shuffle the ballots from different voters, thus the authority cannot relate a ballot to a voter. For the mix-net based voting systems, they need enough amount of mix nodes and ballots to be mixed.
- **Zero-knowledge proof:** Zero-knowledge proof is often employed in a voting system [103, 120, 121] to let the prover to prove that the statement is indeed what it claimed without revealing any additional knowledge of the statement itself. In a voting system, the voter should convince the authority that his ballot is valid by proving that the ballot includes only one legitimate candidate without revealing the candidate information.

- **Blind signature and linkable ring signature:** Voting systems like [122, 123, 124] employ blind signature [122] to convince the tallying centre that the ballot is from a valid voter while not revealing the owner of the ballot. Simultaneously, the authority who signs the ballot learns nothing about the voter's selections. In blind signature, both voters and tallying centre must trust the signer. If the signer is compromised, the signature scheme may stop working. Unlike blind signature, linkable ring signature [125] is proposed to avoid the untrusted signer. Instead, it needs a certain number of voters to participate in the signing process. By comparing the linkability tag, the authority can easily tell whether this voter has already voted. When the voter signs on the ballot, he/she needs to include other voters' public keys to make his/her signature indistinguishable from other voters' signatures.

5.2.3.2 Blockchain-based voting systems

The blockchain-based voting systems can be discussed under three broad categories as follows.

- **Voting systems using cryptocurrency:** In [106], authors propose a voting system based on Bitcoin. In their voting system, the ballot does not need to be encrypted/decrypted as they employ the protocol for lottery. Random numbers are used to hide the ballot that are distributed via zero-knowledge proof. Making deposit before voting may keep the voters to comply with the voting protocol while the malicious voters can still forfeit the voting by refusing to vote. However, only supporting "yes/no" voting may restrict the adoption of this voting system.

  In [105], authors proposed a voting system on the Zcash payment protocol [14] without altering the inner working of Zcash protocol. The voter's anonymity is ensured by the Zcash address schemes. The correctness of the voting is guaranteed by the trusted third-party and the candidates. In this system, the authority, who manages the Zcash and voter status database should be trusted. If the authority is compromised, double-voting or tracing the source of the ballot is possible.

- **Voting systems using smart contract:** In [108], the authors claim that their open voting network is the first implementation of a decentralised and self-tallying Internet voting protocol with maximum voter privacy using Blockchain. They employ smart contract as a public bulletin to achieve self-tallying.[1] However, their voting system

[1]Self-tallying means that after the casting phase, voters can count the ballots themselves.

can only work with two candidates voting (yes/no voting) and the limitation of 50 voters makes it impractical for a real large scale voting system.

- **Voting systems using blockchain as a ballot box:** Tivi and Followmyvote [109, 110] are commercial voting systems which employ the blockchain as a ballot box [2]. They claim to achieve verifiability and accessibility anytime anywhere, while the voters' privacy protection in these systems is hard to evaluate.

To summarize, most traditional voting systems need a centralised trusted party to coordinate the whole voting process. In these systems, the centralised trusted party plays a critical role in storing the ballots, counting the ballots, and publishing the voting result. If it is compromised, the adversary can control the ballot counting and the whole voting result, and there is no efficient approach for participants to detect any compromises. Hence, there is a need of an independent public verifiability feature in such systems. Although the existing blockchain-based voting systems take advantage of blockchain public verifibility, the system security and user privacy of these systems depend on the features provided by the underlying blockchain platform, which are limited and thus make such systems vulnerable to a number of known attacks. Our proposed approach not only takes the benefits of a decentralised trust offered by the blockchain technology to remove the need of a centralised trusted party to do the ballots tallying, voting result decoding and publishing, but also considers key security primitives proposed in the literature including traditional evoting systems to build a practical platform independent secure evoting protocol that can be deployed to any blockchain platforms that support smart contract.

5.2.4 Cryptographic Primitives

In this section, we describe the cryptography primitives borrowed from traditional voting systems and apply in our evoting system. Note that the syntaxes, correctness conditions, and security models of a linkable ring signature and a public key encryption are given in Appendix A and Appendix B, respectively.

5.2.4.1 Message encode and decode

Before the voting starts, we must encode the candidate ID to make it suitable for vote tallying. The encode/decode functions are defined as follows:

[2]The authors call the storage of the ballot as the ballot box.

- **Candidate encoding:** We define $\zeta := \mathsf{Encode}(m) \in \mathbb{Z}$ as the candidate encoding function. For ρ candidates, each labeled with an ID from $\mathcal{P} = \{1, 2, \ldots, \rho\}$, $\beta = 2^{\rho+1}$ be the basis of encoding operation. We encode the m^{th} candidate as $\zeta = \beta^m$ where $m \in \mathcal{P}$. We choose 2 as the basis of β as the division operation can be replaced by the CPU register right shift instruction to achieve a better performance.

- **Candidate decoding:** Let $k = k_t\beta^t + \cdots + k_n\beta^n + k_{n-1}\beta^{n-1} + \cdots + k_1\beta + k_0$ be the representation of k base β, $k \in \mathbb{Z}$, then we define the right shift k with n positions as $\mathsf{rsh}(k, n) = k_t\beta^{t-n} + k_{t-1}\beta^{t-n-1} + \cdots + k_{n+1}\beta + k_n$. Let $\mathsf{sum} = s_\rho\beta^{\rho-1} + s_{\rho-1}\beta^{\rho-2} + \cdots + s_2\beta + s_1$ be the addition of all the ballots where s_j is the total number of ballots that the candidate j^{th} acquires. We define $s_j := \mathsf{Decode}(\mathsf{sum}, j)$ for $1 \leq j \leq \rho$ and is computed as $s_j = \mathsf{rsh}(\mathsf{sum}, \beta^{j-1}) \bmod \beta$.

5.2.4.2 Paillier Encryption System [5]

Paillier Encryption system is employed to enable our voting system to tally the encrypted ballots. In our voting system, we implement the following functions in Paillier system and the detail of these functions are described in Appendix B.1.

- **Key Generation:** $(\mathsf{sk}_{\mathsf{Paillier}}, \mathsf{pk}_{\mathsf{Paillier}}) := \mathsf{Gen}_{\mathsf{Paillier}}(K_{\mathsf{len}})$ is the function to generate the secret key $\mathsf{sk}_{\mathsf{Paillier}}$ and the corresponding public key $\mathsf{pk}_{\mathsf{Paillier}}$ with the given key length K_{len}.The voting administrator invokes this function to generate the key pair and uploads the public key $\mathsf{pk}_{\mathsf{Paillier}}$ to the blockchain.

- **Encryption:** $C_{\mathsf{Ballot}} \leftarrow \mathsf{Enc}_{\mathsf{Paillier}}(\zeta_{\mathsf{Ballot}}, \mathsf{pk}_{\mathsf{Paillier}})$ where $\zeta_{\mathsf{Ballot}} \in \mathbb{Z}_n$ is the plaintext ballot to be encrypted. Voters download the $\mathsf{pk}_{\mathsf{Paillier}}$ from the blockchain and call this function to encrypt their ballots. This function generates the encrypted ballot C_{Ballot}.

- **Decryption:** $\zeta_{\mathsf{Res}} := \mathsf{Dec}_{\mathsf{Paillier}}(C_{\mathsf{Res}}, \mathsf{sk}_{\mathsf{Paillier}})$ where $C_{\mathsf{Res}} \in \mathbb{Z}_n^*$ is the encrypted voting result; the voting administrator invokes this function to decrypt the voting result.

- **Message Membership Proof of Knowledge [126]:** $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(C_{\mathsf{Ballot}}, \Upsilon)$ where C_{Ballot} is the encrypted ballot, Υ is the set of the encoded candidates. When a voter publishes his/her ballot, he/she invokes this function to generate the proof $\{v_j, e_j, u_j\}_{j\in\mathcal{P}}$ that demonstrates his/her ballot encrypts only one of the encoded candidates in Υ.

- **Decryption Correctness Proof of Knowledge:** $(\zeta_{\mathsf{Res}}, r) := \mathsf{PoK}(C_{\mathsf{Res}}, \mathsf{sk}_{\mathsf{Paillier}})$, where ζ_{Res} is the plaintext and r is the random factor that generate the encrypted voting result C_{Res}. After publishing the voting result, the voting administrator invokes this function to generate a unique value pair $(\zeta_{\mathsf{Res}}, r)$ that constructs the C_{Res} to prove that he/she decrypts the voting result C_{Res} correctly.

5.2.4.3 Linkable Ring Signatures

Linkable ring signature (LRS) can be applied in our system to protect the privacy of the voters. In practice, we apply the short linkable ring signature (SLRS) [6] which extends all the SLRS features to make the signature size constant with the growth of voter numbers, it has the following features: (1) every ballot that is accepted by the system is from one of the valid users, (2) the voter can check whether his ballot is counted by the blockchain, (3) the size of the signature is constant to support scalability and (4) double-voting is prevented. In our voting system, we implement the function tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$. The details of these functions are explained in Appendix A.2.

- **Setup:** $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$ is a function that takes λ as the security parameter and generates system-wide public parameters param such as the group of quadratic residues modulo a safe prime product N (explained in Appendix A.2) denoted as $\mathsf{QR}(N)$, the length of the key, and a random generator $\tilde{g} \in \mathsf{QR}(N)$.

- **Key Generation:** $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is a function to generate a key pair for each voter i.

- **Signature:** $\sigma \leftarrow \mathsf{Sign}(\mathcal{Y}, \mathsf{sk}, \mathsf{msg})$ is a function to generate the signature σ using all voters' public keys $\mathcal{Y} = \{y_1, y_2, \ldots, y_b\}$, the message msg to be signed, and the voter's secret key sk. Voters should invoke this function to sign on his/her encrypted ballot.

- **Verification:** $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(\sigma, \mathcal{Y}, \mathsf{msg})$; our voting system invokes this function to test the validity of every ballot. Based on the input of the encrypted ballot itself, the voter's signature and all the voters' public keys, the blockchain accepts or rejects this ballot to be put on the chain.

- **Linkability:** $\mathsf{Link}(\sigma_1, \sigma_2) \rightarrow \mathsf{linked/unlinked}$. When a voter casts his/her vote, our voting system invokes this function to check whether this voter has already catted his vote. If this function returns linked, our voting system rejects this ballot; otherwise, the ballot is recorded on the chain.

5.2.4.4 Blockchain

Blockchain as a new scheme targets at removing the centralised trusted party or regulatory actors to achieve public verifiability. We employ these time-based blocks to store both ballots and the voting results.

There are two typical approaches to achieve consensus; one is based on proof-of-work (PoW) [13] and the other is based on Byzantine Fault Tolerance (BFT) network. To achieve better network scalability, we can deploy our voting system on the PoW based blockchain; the BFT based blockchain can be deployed to achieve better transaction processing performance. Because of the page limitation, we give a brief introduction of a Practical BFT (PBFT) protocol which is applied in our voting system.

PBFT protocol: PBFT protocol can tolerate any number of faults over the lifetime of the system provided fewer than 1/3 of the replicas become faulty [48]. Compared with the PoW protocol, PBFT based blockchain can achieve better performance (less network latency) while it has the restriction of node scalability [17]. There is a leader node accompanied by some validation nodes in PBFT network, When a transaction is submitted to the leader node, the PBFT protocol does the following:

- The leader orders the transaction candidates that should be included in a block and broadcasts the list of ordered transactions to all the validation nodes.
- When each of the validation node receives the list of transactions, it executes the transactions on that list one by one.
- The validation nodes calculate a hash value for this newly created block (hash value includes hashes for executed transactions and final state of this distributed system).
- Validation node broadcasts its hash value to other nodes in the network and starts listening to responses from them.
- If any node finds that 2/3 of all nodes broadcast the same hash value from the execution of ordered transactions list, it regards this block as a valid block and commits this new block to its local ledger.

5.2.4.5 Smart Contract

For the blockchain system, the "smart contract" is widely used as a general purpose computation that takes place on a blockchain. Smart contract enables interactions between end

users and a blockchain by allowing end users to create/query data on the blockchain. We adopt Hyperledger Fabric [8] as a smart contract running environment in our voting system. To use the consistent terminologies, we call hyperledger chaincode as smart contract hereafter. We discuss the smart contract roles and deployment scheme as follows.

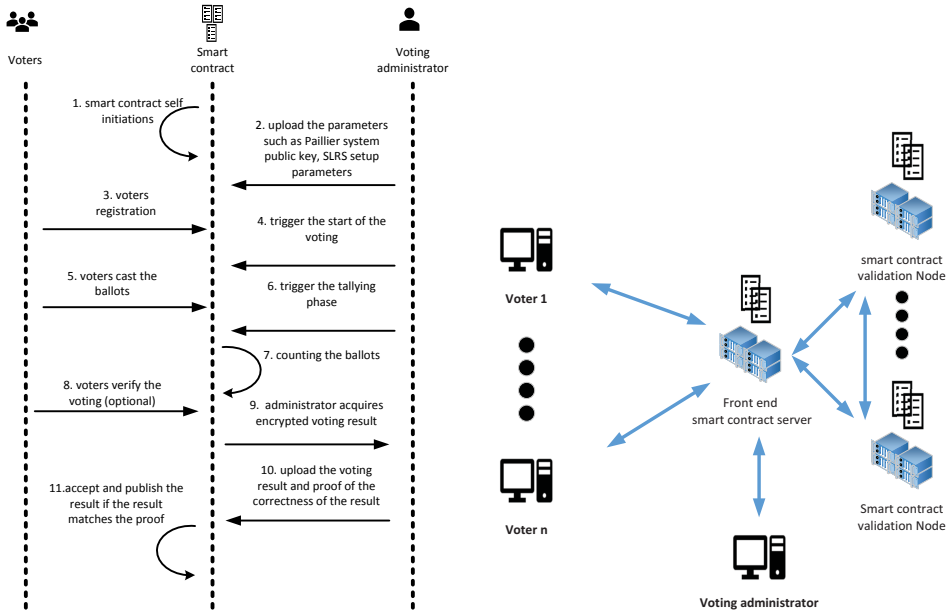
Smart Contract Deployment Scheme: For a practical smart contract, at least three interfaces should be implemented that are **init()**, **invoke()**, and **query()**. **init()** is the interface that is invoked when the smart contract is loaded. **init()** initialises the smart contract system parameters before end-user interacts with the smart contract. **query()** is the interface handling the query request from the blockchain end users. **invoke()** is the interface that is called when the end user wants to put the data on the blockchain. Unlike **query()**, **invoke()** is executed on all validation nodes to ensure the consistency of data on the blockchain.

A smart contract needs to be compiled before being deployed on the blockchain. The Hyperledger administrator is responsible for running the smart contract on every validation node and nominates one node as the front-end server to handle end users requests. The detail of smart contract deployment is discussed in Section 5.2.5.

5.2.5 The Voting Protocol

In this section, we first provide an overview of the whole voting protocol and then discuss each step in details. The whole voting process can be divided into 11 steps as shown in Fig. 5.4(a). Except the smart contract administrator who setups the voting smart contract, three entities are involved: voters, smart contract, and voting administrator. We take Bob as a valid voter in this section to show how the voting protocol works. First, the smart contract is initialised to prepare a voting. Then, the voting administrator uploads the voting parameters. After all voters register themselves in this voting and upload their SLRS public key to the blockchain (the SLRS secret keys are kept by voter themselves), the administrator triggers the start of the voting. Bob as a voter casts his ballot before the administrator triggers the tallying phase. It is optional for Bob to verify the tallying result before the administrator acquires the encrypted tallying result. The administrator needs to upload the voting result and the proof to the blockchain to show the correctness of the result to the voters and all the stakeholders. The smart contract verifies whether the result matches the proof uploaded by the administrator and finally publishes the decrypted voting result on the blockchain.

The voting system consists of one front-end smart contract and several validation nodes as shown in Fig. 5.4(b). The role of a validation node is to replicate the execution of the



(a) The voting protocol diagram. (b) The blockchain evoting system.

Figure 5.4: The general voting protocol and how entities are connected.

smart contract codes to ensure its correct execution. The role of the front-end is similar to the validation node except exporting RESTful API interfaces for communication with voters and administrator. For a practical voting, the validation nodes could be held by different entities/stakeholders, thus all ballots on the blockchain have been verified by different entities/stakeholders. As all the entities/stakeholders have the agreement on the data stored on the blockchain, the blockchain built on the servers owned by different entities can be regarded as trustworthy. It is impractical for the attackers to compromise most of the entities/stakeholders[3].

5.2.5.1 Entities in the voting process

Four entities should be involved in our voting system shown in Fig. 5.4(a), and details are explained as follows:

- **smart contract administrator:** he/she has the ability to access the smart contract platform to deploy/terminate smart contract. In Hyperledger fabric, this ac-

[3]The number of entities to be compromised depends on the underlying consensus protocol.

count is authorised by the membership service and a permission is granted to deploy/terminate the Smart contract. In our voting system, we need at least one smart contract administrator to deploy the voting smart contract.

- **voting administrator:** The role of voting administrator is to organize the vote by setting up the voting parameters and triggering the the tallying and result publishing phase. Although there are underlying mechanisms in hyperledger to authenticate users, we use SLRS to prevent administrator from linking the ballots with the users.
- **smart contract:** The role of smart contract include 1.) store the encrypted ballots. 2.) verify the validity of the ballots. 3.) count the encrypted ballot. 4.) verify the correctness of the voting result. 5.) publish the voting result and provide the platform for the voters to verify the voting process.
- **voters:** Voters are the people who have the rights to cast their vote. They need to register into the voting system before they cast their vote.

5.2.5.2 Smart contract initiation

The smart contract is initialised by generating an RSA keypair $(\mathsf{pk}_s, \mathsf{sk}_s)$. This keypair is employed to sign/verify every transaction between the smart contract and the end users to avoid man-in-the-middle attacks. [4] Additionally, ensuring all the validation nodes run the identical smart contract is of critical importance; otherwise, validation nodes may tamper the smart contract with back doors and colludes with the adversary to get the ballots' information. We require the smart contract to be deployed as follows: Firstly, the smart contact accompanied with a digital fingerprint is verified by all the voting entities/stakeholders to ensure that there is no backdoor. This eliminates the possibility that an entity/stakeholder colludes with a node to run tampered smart contract. Then, this fingerprint is tagged as verified evoting smart contract fingerprint in blockchain platform. MD5, SHA1, and/or SHA2 hashes are mostly employed to generate the fingerprint for a given file [127]. Secondly, when a new node wants to join the evoting blockchain, the fingerprint of this smart contract is checked, if it is identical to the verified fingerprint, the smart contract is loaded, otherwise, is rejected by the blockchain platform.

[4] As we will discuss later, we only accept verified smart contract to run on validation node, thus, avoiding any malicious smart contract exporting the sk_s to anyone.

5.2.5.3 Voting system set up

During the system set up, the voting administrator uploads the following three parameters to the blockchain:

- The public key of the Paillier system $\mathsf{pk}_{\mathsf{Paillier}}$.
- A set of encryptions of zero denoted as T, generated by the administrator's Paillier public key $\mathsf{pk}_{\mathsf{Paillier}}$. For voting with 1 million voters, we suggest the set should contain enough elements to make the randomised pool large enough and the detail of T is discussed in receipt-freeness analysis [5].
- The SLRS scheme parameters, param.

5.2.5.4 Voter registration

Bob must register this voting system with his identity. The registration information could be: (1) email address with a desired password, or (2) the identity number with a desired password, or (3) an invitation URL sent by the administrator with a desired password. After Bob passes the identity check conducted by the smart contract, he can login with the desired password to download the SLRS param and the administrator's Paillier public key $\mathsf{pk}_{\mathsf{Paillier}}$, then generates the SLRS key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ by calling $\mathsf{KeyGen}(\mathsf{param})$; Bob then uploads the public key pk_i to the smart contract (Bob's secret key is kept off-chain by himself). If the smart contract accepts his SLRS public key, the smart contract puts his public key pk_i on the blockchain to complete his registration phase.

5.2.5.5 Vote casting phase

During this phase, Bob casts his vote as follows: (1) Bob chooses one of the candidates $m \in \mathcal{P}$ and encodes it as $\zeta := \mathsf{Encode}(m)$. (2) Bob invokes the Paillier encryption function $C \leftarrow \mathsf{Enc}_{\mathsf{Paillier}}(\zeta, \mathsf{pk}_{\mathsf{Paillier}})$. (3) Bob needs to prove that C is an encryption of an element in $\{\zeta_1, \ldots, \zeta_\rho\}$ (set of all encoded candidates) by calling $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(C, \Upsilon)$; hence he sends $\pi = \{C, \{v_j, e_j, u_j\}_{j\in\mathcal{P}}\}$ to the smart contract.[6] (4) Upon receiving $\{v_j, e_j, u_j\}_{j\in\mathcal{P}}$,

[5] To avoid requiring the administrator to upload the encryption of zero pool, coin flipping protocol can be applied to generate the consistent encryption of zero on all the validation nodes and this is our future work.

[6] Bob can choose none of the actual candidates by casting his ballot to a dummy candidate. When the smart contract publishes the voting result, it ignores all the ballots that the dummy candidate gets.

the smart contract verifies the validation of the encrypted ballot C. We denote ϕ as a mapping of this transaction's session id to T domain. If C is valid, the smart contract takes an encryption of zero at ϕ^{th} position. Let ϵ be the addition of $T[\phi]$ and C. The smart contract signs on ϵ denoted as s and sends (ϵ, s) back to Bob. (5) If Bob accepts s, he invokes $(v, \tilde{y}, \sigma') := \mathsf{Sign}(\epsilon, (\mathsf{pk}, \mathsf{sk}), \mathcal{Y})$ to generate the $\mathsf{Sign}_{\text{bob}}$ on ϵ and sends $(\epsilon, \mathsf{Sign}_{\text{bob}})$ to the smart contract. (6) If the smart contract detects that $\mathsf{Sign}_{\text{bob}}$ has already been recorded on the blockchain or cached in the memory, it rejects Bob's vote; otherwise, $(\epsilon, \mathsf{Sign}_{\text{bob}})$ is put on the blockchain.

5.2.5.6 Ballots tallying and result publishing

Due to the Paillier system's homomorphic feature, counting the vote is as simple as fetching the encrypted ballots from the blockchain and adding them together. The result publishing mechanism is described in the following steps: (1) Let E_{sum} be the sum of all the encrypted votes and Sign_s be the signature signed by the smart contract on E_{sum}. The smart contract sends $(E_{\mathsf{sum}}, \mathsf{Sign}_s)$ to the administrator. (2) The administrator invokes $\mathsf{sum} := \mathsf{Dec}_{\mathsf{Paillier}}(E_{\mathsf{sum}}, \mathsf{sk}_{\mathsf{Paillier}})$ to compute plaintext sum, which encodes the ballots of each candidate. The administrator also invokes $(\mathsf{sum}, r) := \mathsf{PoK}(E_{\mathsf{sum}}, \mathsf{sk}_{\mathsf{Paillier}})$ to calculate the random r that constructs this E_{sum}, and sends (sum, r) to the smart contract. (3) The smart contract verifies the correctness of (sum, r) by checking if $E_{\mathsf{sum}} \stackrel{?}{=} g^{\mathsf{sum}} r^n$ (g is one of the elements of $\mathsf{pk}_{\mathsf{Paillier}}$). (4) If the smart contract accepts the sum, it iteratively invokes $m := \mathsf{Decode}(\mathsf{sum}, i)$ to compute the ballots for each candidate i. Let Π be the dictionary holding the voting result of all candidates. The smart contract finally puts Π on the blockchain.

5.2.5.7 Ballot verifying

After tallying ballots and before the voting administrator decrypts the tallying result, the public can verify ballots on the blockchain to make sure the validity of the voting process. We define two roles for people who can verify the voting. The first one is those who have the right to access the data on the blockchain while they do not have the right to vote. The second one is those who have both rights to vote and access the data on the blockchain. The public verifiability to those who have first role is as follows 1) Checking the number of ballots that were counted and the number of people registered for this voting. 2) Checking the correctness of the tallying result by downloading and adding all the encrypted ballots to match with the tallying result published on the blockchain. Compared to those who have the first role, people assigned to the second role can also verify that his/her ballot is

recorded on the blockchain by checking whether there exists one ballot that is signed with his/her signature; This ensure his/her vote is recorded and counted.

5.2.5.8 Validation nodes and the trustworthiness of blockchain

Under the assumption that the voting administrator will not disclose the Paillier secret key and the encryption of zero (this is discussed in Security and Coercion-Resistance Analysis section later), we discuss the role of the blockchain validation nodes and how they enhance the trustworthiness of the blockchain. The responsibilities of the validation nodes include (1) verify the validity of the ballots (check whether the ballots are from the same voters and/or check whether the given ballot encrypts only one candidate), (2) check the validity of the signature on the given ballot, (3) ballots tallying, and (4) verify the correctness of the voting result. If any interaction between the blockchain and the voting participants does not pass the verification conducted by the validation nodes, this interaction is rejected by the voting system.

In practice, we suggest enhancing the trustworthiness of the blockchain by allowing different political parties/stakeholders host their own validation nodes. The data on the blockchain is verified by different entities/stakeholders, and it is unlikely that these entities/stakeholders collude with each other to publish a false voting result.

5.2.5.9 Comparison with other non-blockchain-based voting protocols

Compared with other non-blockchain-based evoting protocols, our voting system can be differentiated using the following three security features. Firstly, there is no need for a centralised trusted party to tally the ballots and publish the result. Our trustworthiness is built on the assumption that it is impossible that most of the voting entities/stakeholders who own the blockchain validation servers collude with each other. Smart contract fingerprint guarantees that the smart contract which is deployed on the validation node is verified by all the voting entities/stakeholders and no one can replace that with a tampered one. Second, the correctness of the ballots processing (in vote casting phase) and tallying is achieved by asking all the validation nodes to verify the validity of the process; the blockchain network rejects the ballot if the validation nodes disagree on the verification of the ballot process. Third, we allow the voters to verify that his/her ballot is recorded and the correctness of the tallying. Additionally, voting entities/stakeholders can also verify the validity of the vote tally and the voting result.

5.2.6 Correctness and Security Analysis

5.2.6.1 Correctness analysis

The correctness of our voting system is guaranteed by the public verifiability provided by the smart contract and the proof of knowledge provided by the cryptographic schemes. More than that, the smart contract ensures the consistency of a transaction execution. Any inconsistencies generate an error which results in the rejection of the transaction. This means the voting participants can be assured that every transaction on the blockchain is verified and accepted by all participating nodes. This prevents compromised nodes from putting an invalid data on the blockchain unless the adversary can take control of a proportion of the nodes in the whole blockchain network [7].

5.2.6.2 Security features of our voting system

- **Privacy:** The ballots on the public ledger are encrypted and only the voting administrator who initiates the voting can decrypt the ballots. This ensures that the tallying center can count the ballots without knowing the content of the ballots.
- **Anonymity:** The voters, candidates, or smart contract cannot tell the public key of the signer with a probability larger than $1/b$, where b is the number of the voters. This can be guaranteed by the anonymity property of the linkable ring signature (LRS) scheme. Details are explained in Appendix A.2.
- **Double-voting-avoided:** In our voting system, we take advantage of linkability provided by the short ring signature scheme. This means it is infeasible for a voter to generate two signatures such that they are determined to be unlinked. Our system can detect whether the signatures are from the same voter. Hence, a voter can only sign on one ballot and cast his/her ballot only once. This can be guaranteed by the linkable property of the LRS scheme. Details are explained in Appendix A.2.
- **Slanderability-avoided:** A voter cannot generate a signature which is determined to be linked with another signature not generated by him/her. In other words, an adversary cannot fake other voters' signature. This can be guaranteed by the non-slanderability property of the LRS scheme. Details are explained in Appendix A.2.

[7] The number of nodes to be compromised depends on the underlying consensus protocol.

- **Receipt-freeness:** Even if an adversary obtains a voter's secret key, the adversary cannot prove that this voter voted for a specific candidate. This is guaranteed by the addition of encryption of zero which provides additional randomness to the ciphertext which is unknown to the voter. Thus, even if the voter's secret key is disclosed, no one can prove his casted ballot. For our voting system, the security level of the receipt-freeness is affected by the size of the encryption of zero pool, as the voters can collude with each other to reconstruct the encryption of zero pool. One solution is increase the size of zero pool thus more voters is required to reconstruct the pool. Another solution is applying coin flipping protocol on all validation node to work out a consistent randomness encryption of zero for each valid ballot. We have taken the first approach in this paper with 4096 encryptions of zeros.

- **Public verifiability:** Anyone who has the relevant rights to access the blockchain can verify that all the ballots are counted correctly; moreover, voters can also verify whether their ballots have been recorded.

- **Correctness:** Proof-of-knowledge ensures the correctness of the voting process. Voting participants need to prove the correctness of the interactions with the blockchain. Even if some blockchain nodes are compromised, others can still verify whether the proof is correct.

- **Vote-and-Go:** Compared with the voting system proposed in [21], our voting system does not need the voter to trigger the tallying phase. Moreover, in our system, voters can also cast their vote and quit before the voting ends, unlike [108] which needs all voters to finish voting before tallying the ballots.

5.2.6.3 Security and Coercion-Resistance Analysis

To address the security and coercion-resistance, we make the following assumptions:

- The trustworthiness of the blockchain platform is achieved by allowing different stakeholders/entities to host the blockchain validation nodes.

- Voters cast their ballots in a secure terminal, which means it is assumed that no one stand behind a voter or uses digital devices to record the voting process. We do not take the physical voting environment security into our consideration.

- The possibility of an attacker to create a blockchain and apply a social engineering technique to launch a phishing attack is beyond our research scope.

- The administrator should not reveal the Paillier system sercret key and the encryption of zeros to anyone.
- Voters should cast their ballots by themselves. No one else can cast the ballot with a voter's identification except the voter himself.

We demonstrate the robustness of our system under two typical attacks below.

Man-in-the-Middle Attacks: Our voting system has strong resistance to this attack. First, as the voters and the smart contract both sign their messages and the voting data is encrypted, it is impossible for an adversary to forge the signature or alter the data on any parties involved in the transactions. Second, the public keys used for signature verification are all published on the blockchain, preventing the adversary from cheating any parties by replacing the original public key with the adversary's public key. The encryption of the ballot also eliminates the possibility of the ballot leakage.

Denial-of-Service (DoS) Attacks: DoS attack is feasible to launch since the network service is provided in a relatively centralised way. In addition, the servers have relatively limited processing ability for a large number of requests. Distributing the service on different nodes is one of the solutions to DoS attack as it is almost impossible for the adversary to compromise all the servers.

Coercion-Resistance Analysis: Coercion-Resistance means it is infeasible for the adversary to determine whether a coerced voter complies with the demand. Our voting system security features discussed in Section 5.2.6.2 make it impossible to launch the Ballots-buyer attack and Double voting attack. Additionally, our voting system is free from randomization attack.
For the Ballots-buyer attack, an attacker coerces a voter by requiring that he submits a randomly composed ballot. Under such circumstances, both the attacker and the voter have no idea about which candidate this voter casts the ballot for. The purpose of this attack is to nullify the ballots. For our system, it is impossible to launch this attack as the voter should prove that the ciphertext is one out of ρ encrypted candidates by calling $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(\mathsf{Enc}_{\mathsf{Paillier}}(\zeta, \mathsf{pk}), \Upsilon)$. Since Υ is held by the smart contract, any ballot that is not the encryption of any element in Υ is rejected and the voter is notified that this transaction is failed.

Step	Time
generate T	$13,560$ms
bottom half key accumulation	< 34s
top half key accumulation	< 23ms
download parameters	4ms
upload ballots	≈ 776ms
tally	$3,850$ms
decode and publish	$< 2,000$ms

Table 5.2: Time consumed on each step.

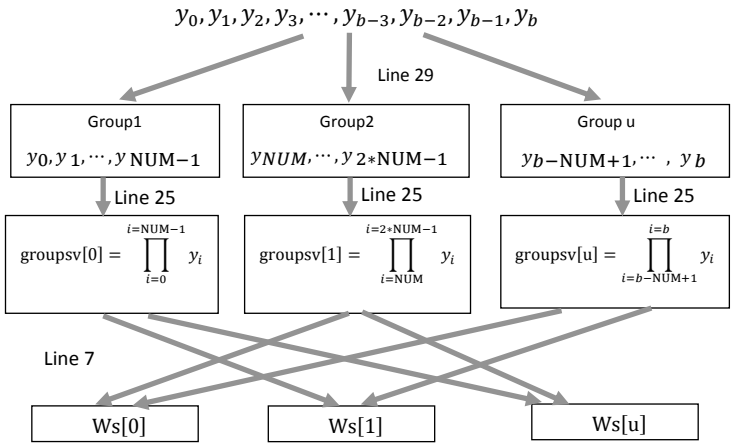


Figure 5.5: The diagram of Algorithm 1.

5.2.7 System deployment and Experiments

5.2.7.1 System deployment

Our voting system can be deployed in any blockchain platforms with smart contract capability and achieve the same level of security. There might be some other reasons to choose a particular platform such as voting latency and flexibility requirements. Different consensus protocols have significant impact on the blockchain network latency and node scalability [17]. If the ballots' confirmation latency is not a major issue for the voting system, the PoW-based blockchain system could be a good option to achieve maximum node scalability. Otherwise, a BFT-based blockchain platform is a better solution. In our scenario, we employ the BFT-based blockchain platform Hyperledger Fabric and deploy our voting system in a practical scenario.

5.2.7.2 Experiments and performance evaluation

We deploy our system in docker containers running on a desktop with 4 cores i5-6500 CPU and 8 GB DDR3 memory. We conduct 1 million voters voting process on the blockchain that consists of 4 validation nodes and 1 PBFT leader node. Each of the validation nodes runs in one dedicated container; thus, we run five docker containers to build our testing blockchain system. We set a voter's public key as 1024 and 2048 bits respectively and the Paillier key pairs as 1024 bits. The deployment pattern is shown in Fig. 5.4(b). We

summarize the time spent on our employed cryptographic processes for 1 million voters' voting in Table 5.2.

Algorithm 8 Bottom half: Server side WS generation.

Input: $\mathcal{Y}$: voter's SLRS public key set with $|\mathcal{Y}| = b$.
Input: Num: number of keys in one group.
Input: ψ, N, and $\phi(N)$: SLRS parameters.
Input: thisgroup :: temporary set containing SLRS keys belonging to a group.
Output: WS, gkeys.

```
1: function GenWs(Num, 𝒴, b)
2:     groupsv ← GenGroupsv(𝒴, b, Num)
3:     for j = 0 : 1 : len(groupsv) do
4:         val ← ψ
5:         for i = 0 : 1 : len(groupsv) do
6:             if j ≠ i then
7:                 val ← val^groupsv[i] mod N
8:             end if
9:         end for
10:        WS[j] ← val
11:    end for
12:    return WS
13: end function
```

Voting parameters setting up time (administrator side): To initialise the voting, the administrator is responsible for uploading the voting parameters as discussed in Section 5.2.5. Let t_{cal} be the time taken to generate T, and t_{upload} be the time spent on uploading T to the blockchain. With 1024 bits key length, the pool size is 1MB. According to our test, t_{upload} is < 1 second and T_{cal} is about 14s. In conclusion, under 100MB bandwidth network, on the smart contract side, the majority of the time is spent on bottom half key accumulation, and on the administrator side, the most time-consuming phase is generating and uploading T (the pool of encryption of zeros).

SLRS parameter setting up time: Compared with LRS, SLRS enables the size of the signature constant no matter how many signers are involved in this signature. This feature is critical important for a large scale voting (i.e. the number of voters $> 100,000$ 1024-bits keys) as the signature should be constant to be suitable for storing on the blockchain. Compared with LRS, SLRS needs an extra step that accumulates all the signers' public keys. Let y_i be the public key of i^{th} voter and ψ be the SLRS public parameter for all

```
14: function GenGroupsv(𝒴, b, Num)
15:     if b <= Num then
16:         localv ← 1
17:         for i = 0 : 1 : b do
18:             localv = localv · 𝒴[i]
19:         end for
20:         groupsv[0] = localv
21:         return groupsv
22:     else
23:         thisgroup ← 0
24:         for i = 0 : 1 : b − 1 do
25:             localv = (localv · 𝒴[i]) mod (ϕ(N))
26:             thisgroup = append(kgroup, 𝒴[i])
27:             if (i mod Num) == Num − 1 then
28:                 groupsv = append(groupsv, localv) and gkeys = append(gkeys, thisgroup)
29:                 thisgroup ← 0 and localv = 1
30:             end if
31:         end for
32:         groupsv = append(groupsv, localv)
33:     end if
34:     return groupsv, gkeys
35: end function
36:
```

Algorithm 9 Top half: Voter computes the key accumulation.

Input: WS: array contains key accumulation from bottom half.
Input: Y': public keys array that this voter's key belongs to.
Input: N and $\phi(N)$: SLRS parameters.
Input: j: the index of this voter.
Input: idx: the index of this voter's public key in WS.
Output: ret: the result of key accumulation for this voter.

```
    function GenKeyAcc(WS, Y, j, idx, Num)
 2:     x ← 1
        for i = 0 : 1 : Num − 1 do
 4:         if then j ≠ i
                x = (x · Y'[i]) mod φ(N)
 6:         end if
        end for
 8:     ret ← WS[idx]^x mod N
        return ret
10: end function
```

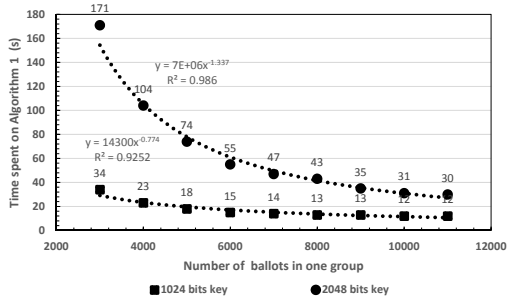
voters. We define the key accumulation operation for all the voters' SLRS public keys for the i^{th} voter as $w_i = \psi^{y_1 y_2 \cdots y_{i-1} y_{i+1} y_{i+2} \cdots y_b}$. In order to make the time spent on key accumulation acceptable, we divide the key accumulation into two halves. The bottom half is run on smart contract and the top half is run by each voter.

Bottom half time consumption (smart contract side): For the bottom half (shown in Appendix Algorithm 8), on the smart contract, we divide the voter SLRS public keys into m groups and pre-calculate the accumulation of all the public keys except the keys in the given group i and denote this key accumulation as ws_i. A diagram that shows how Algorithm 8 works is also given in Fig. 5.5. We only discuss a case in which the number of the voters is larger than 500; otherwise, the voters can generate the key accumulation themselves within a reasonable computation time. We denote G as the group that contains the voters' SLRS public key pk and f the public key accumulation function. We invoke an array operation function *append* to add an element into the array. We distribute the voters' SLRS public keys into u groups and each group has Num of keys (except the last group). We denote the array WS to store all ws, and gkeys to store the voters' SLRS public keys groups. The most time-consuming part is the multiplication of public keys for each group. In our implementation, we calculate the WS using four threads to save time.
We evaluate the performance for 1 million voters' voting and the result is shown in

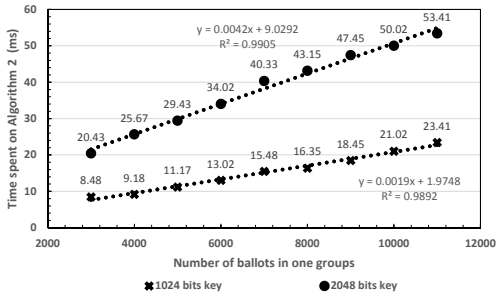
Fig. 5.6(a). We find that the time spent on calculating WS decreases with the growth of the voter numbers in one group. For example, for 1024 bits length and 2048 bit length key accumulation, it decreases from 34s and 171s for the group that contains 3000 voters to 13s and 35s for the group that contains $8,000$ voters, respectively. This is due to 1) the time spent on running the exponential computation loop at line 7 in Algorithm 8 that dominates the time spent for the whole algorithm2.) For the 1 million voters' public key accumulation, we decrease the number of groups by increasing the number of the voters in a group to decrease the time spent on the loop at line 7 in Algorithm 8.

Top half time consumption (Voter side key accumulation): As shown in Appendix Algorithm 9, the voter downloads the array WS and the key group that his key belongs to in gkeys from the blockchain. The time spent on downloading these parameters is acceptable because of two reasons 1) the key size and the element size in WS are constant. 2) The number of groups is relatively small. For instance, if we have 1 million voters and we group 5000 voters in one group, and set the public key size as 1024 bits and N as 1024 bits, then the size of all the public keys in this group, denoted as $\mathcal{Y}'$, is approximately 624KB. The size of an element in WS is restricted by SLRS parameter N; thus with the parameters above, WS is 256KB. Therefore, the total size of $\mathcal{Y}'$ and WS is about 880KB. The voter only needs to do one exponential operation, regardless of the voting scale. From Fig. 5.6(b), it can be seen that with the key size of 1024 bits, it increase from 8.48ms for the group that contains 3000 voters to 16.35ms for the group that contains 8000 voters. The increase of time spent for the key accumulation on the voter's side can be explained as the increase of time spent at line 4 in Algorithm 9, as it dominates the total time spent for the voter side key accumulation.
For 1 million voter's voting, we could set the number of voters in one group smaller to reduce the time spent on the voter side for key accumulation. For the key length of 1024 bits, we recommend to set each group contains about 7000 voters so that it takes 14s and 15.48ms on the smart contract side and the voter side key accumulation, respectively.

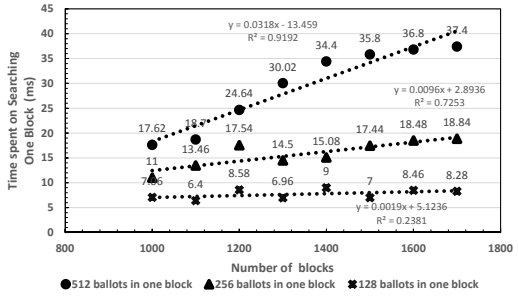
The time spent on casting votes can be divided into three parts. The first part is the parameter preparation, that is, downloading the voting parameters from the server, denoted as t_1. The second part is the time spent for calculating the ring signature, denoted as t_2. The last part is the interactions between voters and the smart contract for uploading the ballots and performing the proof of correctness, denoted as t_3. t_1 can be evaluated roughly as the time spent on downloading WS and $\mathcal{Y}'$. For downloading the parameters of 1 million voting, it takes about 4ms under 100MB network (see Table 5.2). t_2 is approximately 15ms and the average value of t_3 in our test system is 776.60ms. In conclusion, it takes about 1s for a voter to cast his vote in a setting of 1 million voters' voting.



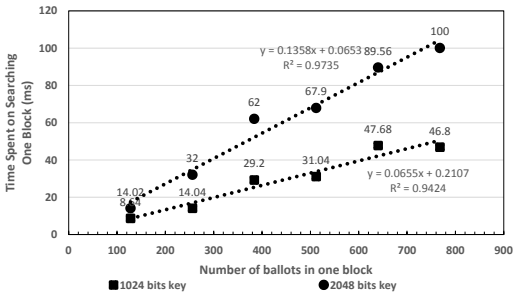
(a) Time spent in Algorithm 1.



(b) Time spent in Algorithm 2.



(c) Time versus growth of ballots in a block.



(d) Time versus growth of voter numbers.

Figure 5.6: SLRS public key accumulation and searching a block on a given blockchain in 1 million voters' voting

Ballot tallying and result publishing time: As the Paillier system restricts the length of each encrypted ballot within 2 times of the Paillier keypair size, the addition of encrypted ballot can achieve constant and reasonable performance. We test the time spent on an addition operation of encrypted ballots in a docker container. With 2048 bits length cipher, it takes 3.85s to add 1 million encrypted ballots. The time spent on publishing the result is < 2s as we optimise the decoding algorithm by using shifting operation.
Ballot verification time: Our voting system provides the public verifiability. Compared with the time spent on checking the signature and tallying the result, the most time-consuming part is searching the block that contains a given ballot. The length of the blockchain and the size of each block have great impact on the search performance. We evaluate the time spent on these two factors as follows. For the first case, we set the evoting SLRS key size to 1024 bits length, and let each block contains 128 ballots, 256 blocks, and 512 blocks, respectively. It is observed in Fig. 5.6(c) that the time spent on these three blockchains increases linearly with the increase on the number of ballots in one block. For the second case, we set the total number of ballots to 1 million and the SLRS key size as 1024 bits long; it is observed in Fig. 5.6(d) that the time spent on searching one block grows linearly from 8.64ms in the blockchain that each block contains 128 ballots to 46.8ms in the blockchain that each block contains 768 ballots.

Based on the above experiments, it is clear that both the increases of the block size and chain length increase the time spent on searching one given block in the blockchain. For 1 million voters' voting system, the blockchain that consists of smaller blocks has better search performance. However, the drawback of the smaller block size is that it increases the number of searching operations (e.g., if we put all ballots in one block, users only searches once to get them; whereas if we allocate all of them in 10 blocks, users have to search 10 times to get them). Based on the experiment results shown in Fig. 5.6(c) and Fig. 5.6(d), in practice, we recommend to set each block contains 640 ballots to achieve both a reasonable search time latency and the average number of search operations.

5.2.8 Conclusion

To solve the problems that the current blockchain voting system cannot provide the comprehensive security features, and most of them are platform dependent, we have proposed a blockchain-based voting system that the voters' privacy and voting correctness are guaranteed by homomorphic encryption, linkable ring signature, and PoKs between the voter and blockchain. We analyse the correctness and the security of our voting system. The experimental results show that our voting system achieves a reasonable performance even in a large scale voting.

Chapter 6

Conclusion

Trust plays a critical role in information exchanges. It helps different entities to deal with each other more effectively and is often a key element in any collaborative system. Blockchain distributes the trust from one party to all the participants in the system. Despite the advantages that blockchain brings, it has disadvantages like huge storage wast and the low transactions throughput. Through my Ph.D. study, I propose a blockchain file system to address the duplication of the blocks, a hybrid consensus protocol to address the low transaction throughput. I also address the security of the blockchain-based applications by applying the cryptographic schemes in the blockchain systems. In conclusion, blockchain is an open system which allows all the participants to verify the correctness of the data on the chain.

It is my future research direction to employ the cryptographic scheme to make the blockchain-based system more secure and efficient. I would like to do the research which combines the multi-party computation with the blockchain to free the protocol participants from having a trusted party to initialise the communication. I would also like to do the research on how to make the cryptographic schemes more applicable in industry with a reasonable cost. It has no doubt that blockchain not only brings new research directions to the applied cryptography but also brings the new message exchange model to the industry.

References

[1] Z. Trifa and M. Khemakhem, "Sybil nodes as a mitigation strategy against sybil attack," *Procedia Computer Science*, vol. 32, pp. 1135–1140, 2014.

[2] B. Yu, J. Liu, S. Nepal, J. Yu, and P. Rimba, "Proof-of-qos: Qos based blockchain consensus protocol," *Computers & Security*, p. 101580, 2019.

[3] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, and R. Ranjan, "Iotchain: Establishing trust in the internet of things ecosystem using blockchain," *IEEE Cloud Computing*, vol. 5, no. 4, pp. 12–23, 2018.

[4] B. Yu, J. K. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, and M. H. Au, "Platform-independent secure blockchain-based voting system," in *International Conference on Information Security*. Springer, 2018, pp. 369–386.

[5] T. Volkhausen, "Paillier cryptosystem: A mathematical introduction," in *Seminar Public-Key Kryptographie (WS 05/06) bei Prof. Dr. J. Blömer*, 2006.

[6] M. H. Au, S. S. Chow, W. Susilo, and P. P. Tsang, "Short linkable ring signatures revisited," in *European Public Key Infrastructure Workshop*. Springer, 2006, pp. 101–115.

[7] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the technology behind Bitcoin is changing money, business, and the world*. New York: Penguin, 2016.

[8] "hyperledger whitepaper," http://blockchainlab.com/pdf/HyperledgerWhitepaper.pdf, accessed on June 24, 2017.

[9] J. B. A. M. J. Clark, A. N. J. A. K. Edward, and W. Felten, "Research perspectives and challenges for bitcoin and cryptocurrencies," *url: https://eprint. iacr. org/2015/261. pdf*, 2015.

[10] "Merkle tree," https://en.wikipedia.org/wiki/merkle_tree, accessed on June 24 2018.

[11] "Bitcoin lightning network," https://en.wikipedia.org/wiki/Lightning_network, accessed on February 14 2018.

[12] M. Amelchenko and S. Dolev, "Blockchain abbreviation: Implemented by message passing and shared memory," in *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on*. IEEE, 2017, pp. 1–7.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[14] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," Tech. rep. 2016-1.10. Zerocoin Electric Coin Company, Tech. Rep., 2016.

[15] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, 2016, p. 4.

[16] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[17] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125.

[18] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[19] "Bft smart," https://github.com/bft-smart/library, accessed on June 24, 2018.

[20] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 455–471.

[21] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *International Workshop on Security Protocols*. Springer, 1997, pp. 25–35.

[22] A. Miller, M. Möser, K. Lee, and A. Narayanan, "An empirical analysis of linkability in the monero blockchain," *arXiv preprint*, vol. 1704, 2017.

[23] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake." *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, 2016.

[24] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017, pp. 253–255.

[25] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A. B. Tran, and P. Rimba, "On availability for blockchain-based systems," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017, pp. 64–73.

[26] P. Ekparinya, V. Gramoli, and G. Jourjon, "Impact of man-in-the-middle attacks on ethereum," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2018, pp. 11–20.

[27] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network." in *USENIX Security Symposium*, 2015, pp. 129–144.

[28] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 375–392.

[29] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network." *IACR Cryptology ePrint Archive*, vol. 2018, p. 236, 2018.

[30] "lightweight node wiki," https://en.bitcoin.it/wiki/Lightweight_node, accessed on July 24 2018.

[31] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartı, "Analysis of the bitcoin utxo set," in *Proceedings of the 5th Workshop on Bitcoin and Blockchain Research Research (in Assocation with Financial Crypto 18), Lecture Notes in Computer Science*, 2018.

[32] "Filecoin website," https://filecoin.io, accessed on June 24 2017.

[33] "Chia website," https://www.chia.net/, accessed on June 24 2017.

[34] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*. ACM, 2001, vol. 31, no. 4.

[35] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[36] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[37] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2001, pp. 329–350.

[38] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed object location in a dynamic network," *Theory of Computing Systems*, vol. 37, no. 3, pp. 405–440, 2004.

[39] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly," in *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. ACM, 2002, pp. 183–192.

[40] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[41] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?[p2p traffic measurement]," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 3. IEEE, 2004, pp. 1532–1538.

[42] "Ipfs official document," https://docs.ipfs.io/, accessed on June 24 2018.

[43] "Ipld website," https://ipld.io/, accessed on February 14 2018.

[44] B. Gallmeister, *POSIX. 4 Programmers Guide: Programming for the real world.* " O'Reilly Media, Inc.", 1995.

[45] G. Lowe, "An attack on the needham- schroeder public- key authentication protocol," *Information processing letters*, vol. 56, no. 3, 1995.

[46] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[47] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[48] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[49] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.

[50] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[51] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 106–125.

[52] "Total cryptocurrency market cap hits new all-time high," https://www.coindesk.com/150-billion-total-cryptocurrency-market-cap-hits-new-time-high/, accessed on June 24, 2017.

[53] B. M. David, P. Gazi, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol." *IACR Cryptology ePrint Archive*, vol. 2017, p. 573, 2017.

[54] QuantumMechanic, "Bitcoin forum - proof of stake instead of proof of work," 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0

[55] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," Cryptology ePrint Archive, Report 2018/239, 2018, https://eprint.iacr.org/2018/239.

[56] "Eos white paper," https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md, accessed on June 24 2017.

[57] X. Vives, "Nash equilibrium with strategic complementarities," *Journal of Mathematical Economics*, vol. 19, no. 3, pp. 305–321, 1990.

[58] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Annual International Cryptology Conference*. Springer, 2001, pp. 524–541.

[59] "A survey of the ntp network," http://alumni.media.mit.edu/~nelson/research/ntp-survey99, accessed on June 24 2018.

[60] D. Malkhi and M. K. Reiter, "Byzantine quorum systems," in *Distributed Computing*, 1997, pp. 203–213.

[61] I. Bentov, P. Hubácek, T. Moran, and A. Nadler, "Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies." *IACR Cryptology ePrint Archive*, vol. 2017, p. 300, 2017.

[62] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[63] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions," Cryptology ePrint Archive, Report 2019/360, 2019, https://eprint.iacr.org/2019/360.

[64] "Bitcoin transaction throughput," https://en.wikipedia.org/wiki/Bitcoin_scalability_problem, accessed on June 24, 2017.

[65] A. Ghosh, M. Mahdian, D. M. Reeves, D. M. Pennock, and R. Fugger, "Mechanism design on trust networks," in *International Workshop on Web and Internet Economics*. Springer, 2007, pp. 257–268.

[66] "Stellar protocol," https://www.stellar.org/, accessed on February 14 2018.

[67] "Ripple network," https://ripple.com/, accessed on February 14 2018.

[68] R. Fugger, "Money as ious in social trust networks & a proposal for a decentralized currency network protocol," *Hypertext document. Available electronically at http://ripple. sourceforge. net*, vol. 106, 2004.

[69] B. Viswanath, M. Mondal, K. P. Gummadi, A. Mislove, and A. Post, "Canal: Scaling social network-based sybil tolerance schemes," in *Proceedings of the 7th ACM european conference on Computer Systems*. ACM, 2012, pp. 309–322.

[70] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *arXiv preprint arXiv:1702.05812*, 2017.

[71] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing security and privacy in credit networks," in *24th Annual Network and Distributed System Security Symposium, NDSS*, 2017.

[72] P. Moreno-Sanchez, A. Kate, M. Maffei, and K. Pecina, "Privacy preserving payments in credit networks," in *Network and Distributed Security Symposium*, 2015.

[73] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and Distributed System Security Symposium*, 2017.

[74] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Symposium on Self-Stabilizing Systems*. Springer, 2015, pp. 3–18.

[75] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[76] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Australasian Conference on Information Security and Privacy*. Springer, 2016, pp. 57–76.

[77] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions," Cryptology ePrint Archive, Report 2019/360, 2019, https://eprint.iacr.org/2019/360.

[78] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[79] "Hashed timelock contracts," https://en.bitcoin.it/wiki/HashedTimelockContracts, accessed on February 14 2018.

[80] P. F. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," in *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4. ACM, 1988, pp. 35–42.

[81] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," *White Paper*, 2016.

[82] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling payments fast and private: Efficient decentralized routing for path-based transactions," *arXiv preprint arXiv:1709.05748*, 2017.

[83] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures." *IACR Cryptology ePrint Archive*, vol. 1998, p. 10, 1998.

[84] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004, vol. 19.

[85] F. Cristian, H. Aghili, and R. Strong, "Approximate clock synchronization despite omission and performance failures and processor joins," in *Proceedings of the 16th International Symposium on Fault-Tolerant Computing*, 1986, pp. 218–223.

[86] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 2001 IEEE International Conference on Cluster Computing*. IEEE, 2001, pp. 136–145.

[87] A. C.-C. Yao, "Protocols for secure computations," in *FOCS*, vol. 82, 1982, pp. 160–164.

[88] R. Ranjan, O. Rana, S. Nepal, M. Yousif, P. James, Z. Wen, S. Barr, P. Watson, P. P. Jayaraman, D. Georgakopoulos *et al.*, "The next grand challenges: Integrating the internet of things and data science," *IEEE Cloud Computing*, vol. 5, no. 3, pp. 12–26, 2018.

[89] "us cell carriers sell location data," https://www.zdnet.com/article/us-cell-carriers-selling-access-to-real-time-location-data/, accessed on July 24 2018.

[90] J. Voas, "Networks of 'things'," *NIST Special Publication*, vol. 800, no. 183, pp. 800–183, 2016.

[91] B. Celler, M. Varnfield, S. Nepal, R. Sparks, J. Li, and R. Jayasena, "Impact of at-home telemonitoring on health services expenditure and hospital admissions in patients with chronic conditions: before and after control intervention analysis," *JMIR medical informatics*, vol. 5, no. 3, p. e29, 2017.

[92] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 3–16.

[93] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[94] "Neo whitepaper," http://docs.neo.org/en-us/, accessed on June 24 2017.

[95] "Redcoin whitepaper," https://reddcoin.com/, accessed on June 24 2017.

[96] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 45–59.

[97] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, p. 37, 2014.

[98] "Litecoin network," https://litecoin.com/, accessed on June 24 2017.

[99] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[100] Theguardian, "Why machines are bad at counting votes," 2009. [Online]. Available: https://www.theguardian.com/technology/2009/apr/30/e-voting-electronic-polling-systems

[101] "Nsw election result could be challenged over ivote security flaw," 2015. [Online]. Available: https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw

[102] B. Adida, "Helios: Web-based open-audit voting." in *USENIX security symposium*, vol. 17, 2008, pp. 335–348.

[103] S. S. Chow, J. K. Liu, and D. S. Wong, "Robust receipt-free election system with ballot secrecy and verifiability." in *NDSS*, vol. 8, 2008, pp. 81–94.

[104] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 839–858.

[105] P. Tarasov and H. Tewari, "Internet voting using zcash," Cryptology ePrint Archive, Report 2017/585, 2017, http://eprint.iacr.org/2017/585.

[106] Z. Zhao and T.-H. H. Chan, "How to vote privately using bitcoin," in *International Conference on Information and Communications Security*. Springer, 2015, pp. 82–96.

[107] B. Lee and K. Kim, "Receipt-free electronic voting scheme with a tamper-resistant randomizer," in *ICISC*, vol. 2587. Springer, 2002, pp. 389–406.

[108] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy." *IACR Cryptology ePrint Archive*, vol. 2017, p. 110, 2017.

[109] "Tivi voting," https://tivi.io/, accessed on June 24, 2017.

[110] "follow my vote," https://followmyvote.com/, accessed on June 24, 2017.

[111] C. Gentry, *A fully homomorphic encryption scheme*. Stanford University, 2009.

[112] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *Transactions on Emerging Telecommunications Technologies*, vol. 8, no. 5, pp. 481–490, 1997.

[113] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology—EUROCRYPT 2000*. Springer, 2000, pp. 539–556.

[114] J. Katz, S. Myers, and R. Ostrovsky, "Cryptographic counters and applications to electronic voting," *Advances in Cryptology Eurocrypt 2001*, pp. 78–92, 2001.

[115] Z. Xia, S. A. Schneider, J. Heather, and J. Traoré, "Analysis, improvement, and simplification of prêt à voter with paillier encryption." in *EVT'08 Proceedings of the Conference on Electronic Voting Technology*, 2008.

[116] P. Y. Ryan, "Prêt à voter with paillier encryption," *Mathematical and Computer Modelling*, vol. 48, no. 9, pp. 1646–1662, 2008.

[117] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *Public Key Cryptography*, vol. 2274. Springer, 2002, pp. 141–158.

[118] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. ACM, 2005, pp. 61–70.

[119] S. J. Laskowski, M. Autry, J. Cugini, W. Killam, and J. Yen, "Improving the usability and accessibility of voting systems and products," *NIST Special Publication*, pp. 256,500, 2004.

[120] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM, 2001, pp. 116–125.

[121] S. Weber, "A coercion-resistant cryptographic voting protocol-evaluation and prototype implementation," *Darmstadt University of Technology, http://www. cdc. informatik. tudarmstadt. de/reports/reports/StefanWeber. diplom. pdf*, 2006.

[122] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *International Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1992, pp. 244–251.

[123] R. Joaquim, A. Zúquete, and P. Ferreira, "Revs–a robust electronic voting system," *IADIS International Journal of WWW/Internet*, vol. 1, no. 2, pp. 47–63, 2003.

[124] C.-T. Li, M.-S. Hwang, and Y.-C. Lai, "A verifiable electronic voting scheme over the internet," in *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*. IEEE, 2009, pp. 449–454.

[125] J. K. Liu and D. S. Wong, "Linkable ring signatures: Security models and new schemes," in *International Conference on Computational Science and Its Applications*. Springer, 2005, pp. 614–623.

[126] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. ACM, 2001, pp. 274–283.

[127] C. Perrin, "Use md5 hashes to verify software downloads," 2007, https://www.techrepublic.com/blog/it-security/use-md5-hashes-to-verify-software-downloads/.

[128] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *International Conference on Security in Communication Networks*. Springer, 2002, pp. 268–289.

[129] "Ethereum fundation," https://www.ethereum.org/, accessed on February 14 2018.

[130] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A survey of peer-to-peer storage techniques for distributed file systems," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2. IEEE, 2005, pp. 205–213.

[131] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West, "Scale and performance in a distributed file system," *ACM Transactions on Computer Systems (TOCS)*, vol. 6, no. 1, pp. 51–81, 1988.

[132] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 202–215.

[133] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 188–201.

[134] G. J. Popek, R. G. Guy, T. W. Page, and J. S. Heidemann, "Replication in ficus distributed file systems," in *Management of Replicated Data, 1990. Proceedings., Workshop on the*. IEEE, 1990, pp. 5–10.

[135] T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Y. Wang, "Serverless network file systems," in *ACM SIGOPS Operating Systems Review*, vol. 29, no. 5. ACM, 1995, pp. 109–126.

[136] J. J. Kistler and M. Satyanarayanan, "Disconnected operation in the coda file system," *ACM Transactions on Computer Systems (TOCS)*, vol. 10, no. 1, pp. 3–25, 1992.

[137] J. K. Ousterhout, A. R. Cherenson, F. Douglis, M. N. Nelson, and B. B. Welch, "The sprite network operating system," *Computer*, vol. 21, no. 2, pp. 23–36, 1988.

[138] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 1–14, 2002.

[139] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the sun network filesystem," in *Proceedings of the Summer USENIX conference*, 1985, pp. 119–130.

[140] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.

[141] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus," *CoRR, abs/1612.02916*, 2016.

[142] "Ripple network," https://www.ripple.com, accessed on June 24 2017.

APPENDICES

.1 Linkable Ring Signature

.1.1 Syntax of Linkable Ring Signature

- $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$ is a probabilistic polynomial time (PPT) algorithm which, on input a security parameter λ, outputs the set of security parameters param which includes λ. We denote by $\mathcal{EID}$, $\mathcal{M}$ and Σ the domains of event-id, messages and signatures, respectively.
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is a PPT algorithm which, on input a security parameter $\lambda \in \mathbb{N}$, outputs a private/public key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$. We denote by $\mathcal{SK}$ and $\mathcal{PK}$ the domains of possible private keys and public keys, respectively.
- $\sigma \leftarrow \mathsf{Sign}(e, n, \mathcal{Y}, \mathsf{pk}, M)$ which, on input event-id e, group size n, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a private key whose corresponding public key is contained in $\mathcal{Y}$, and a message M, produces a signature σ.
- $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(e, n, \mathcal{Y}, M, \sigma)$ which, on input event-id e, group size n, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a message-signature pair (M,σ) returns accept or reject. If accept, the message-signature pair is *valid*.
- $\mathsf{linked/unlinked} \leftarrow \mathsf{Link}\ (e, n_1, n_2, \mathcal{Y}_1, \mathcal{Y}_2, M_1, M_2,, \sigma_1, \sigma_2)$ which, on input event-id e, group size n_1, n_2, two sets $\mathcal{Y}_1, \mathcal{Y}_2$ of n_1, n_2 public keys respectively, two valid signature and message pairs $(M_1, \sigma_1, M_2, \sigma_2)$, outputs linked or $\mathsf{unlinked}$.

Correctness. LRS schemes must satisfy:

- (Verification Correctness.) Signatures signed according to specification are accepted during verification.

- (Linking Correctness.) If two signatures are signed for the same event according to specification, then they are linked if and only if the two signatures share a common signer.

.1.2 Notions of Security of Linkable Ring Signature

Security of LRS schemes has four aspects: unforgeability, anonymity, linkability and non-slanderability. Before giving their definition, we consider the following oracles which together model the ability of the adversaries in breaking the security of the schemes.

- $pk_i \leftarrow \mathcal{JO}(\perp)$. The *Joining Oracle*, on request, adds a new user to the system. It returns the public key $\mathsf{pk} \in \mathcal{PK}$ of the new user.
- $\mathsf{sk}_i \leftarrow \mathcal{CO}(\mathsf{pk}_i)$. The *Corruption Oracle*, on input a public key $\mathsf{pk}_i \in \mathcal{PK}$ that is a query output of $\mathcal{JO}$, returns the corresponding private key $\mathsf{sk}_i \in \mathcal{SK}$.
- $\sigma' \leftarrow \mathcal{SO}(e, n, \mathcal{Y}, pk_\pi, M)$. The *Signing Oracle*, on input an event-id e, a group size n, a set $\mathcal{Y}$ of n public keys, the public key of the signer $\mathsf{pk}_\pi \in \mathcal{Y}$, and a message M, returns a valid signature σ'.

If the scheme is proven in random oracle model, a random oracle is simulated.

1. Unforgeability. Unforgeability for LRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$, $\mathcal{SO}$ and the random oracle:

   (a) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

   (b) $\mathcal{A}$ may query the oracles according to any adaptive strategy.

   (c) $\mathcal{A}$ gives $\mathcal{S}$ an event-id $e \in \mathcal{EID}$, a group size $n \in \mathbb{N}$, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a message $M \in \mathcal{M}$ and a signature $\sigma \in \Sigma$.

   $\mathcal{A}$ wins the game if:

   (1) $\mathsf{Verify}(e, n, \mathcal{Y}, M, \sigma) = \mathsf{accept}$;

   (2) All of the public keys in $\mathcal{Y}$ are query outputs of $\mathcal{JO}$;

   (3) No public keys in $\mathcal{Y}$ have been input to $\mathcal{CO}$; and

(4) σ is not a query output of $\mathcal{SO}$.

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda) = \Pr[\mathcal{A} \text{ wins the game }]$$

Definition 4 (unforgeability)**.** *A LRS scheme is unforgeable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda)$ is negligible.*

2. Anonymity. It should not be possible for an adversary $\mathcal{A}$ to tell the public key of the signer with a probability larger than $1/n$, where n is the cardinality of the ring, even assuming that the adversary has unlimited computing resources.

   Specifically, anonymity for LRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the unbounded Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracle $\mathcal{JO}$.

   (a) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

   (b) $\mathcal{A}$ may query $\mathcal{JO}$ according to any adaptive strategy.

   (c) $\mathcal{A}$ gives $\mathcal{S}$ an event-id $e \in \mathcal{EID}$, a group size $n \in \mathbb{N}$, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$ such that all of the public keys in $\mathcal{Y}$ are query outputs of $\mathcal{JO}$, a message $M \in \mathcal{M}$. Parse the set $\mathcal{Y}$ as $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$. $\mathcal{S}$ randomly picks $\pi_R \in \{1, \ldots, n\}$ and computes $\sigma_\pi = \mathsf{Sign}(e, n, \mathcal{Y}, sk_\pi, M)$, where sk_π is a corresponding private key of pk_π. σ_π is given to $\mathcal{A}$.

   (d) $\mathcal{A}$ outputs a guess $\pi' \in \{1, \ldots, n\}$.

   We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda) \quad = \quad \left| \Pr[\pi' = \pi] - \frac{1}{n} \right|$$

   Definition 5 (Anonymity)**.** *A LRS scheme is anonymous if for any adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Anon}(\lambda)$ is zero.*

3. Linkability.

   Linkability for LRS schemes is mandatory, that is, it should be infeasible for a signer to generate two signatures such that they are determined to be unlinked using LRS.Link. The following definition/game essentially captures a scenario that an adversary tries to generate two LRS signatures, using strictly fewer than 2 user private keys, so that these two signatures are determined to be unlinked using LRS.Link. If the LRS scheme

is unforgeable (as defined above), then these signatures can only be generated if at least 2 user private keys are known. If less than 2 user private keys are known, then there must be one common signer to both of the signatures. Therefore, this model can effectively capture the definition of linkability.

Linkability for LRS scheme is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$, $\mathcal{SO}$ and the random oracle:

(a) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

(b) $\mathcal{A}$ may query the oracles according to any adaptive strategy.

(c) $\mathcal{A}$ gives $\mathcal{S}$ an event-id $e \in \mathcal{EID}$, group sizes $n_1, n_2 \in \mathbb{N}$ (w.l.o.g. we assume $n_1 \leq n_2$), sets $\mathcal{Y}_1$ and $\mathcal{Y}_2$ of public keys in pk of sizes n_1 and n_2 resp., messages $M_1, M_2 \in \mathcal{M}$ and signatures $\sigma_1, \sigma_2 \in \Sigma$.

$\mathcal{A}$ wins the game if

(1) All public keys in $\mathcal{Y}_1 \cup \mathcal{Y}_2$ are query outputs of $\mathcal{JO}$;

(2) $\mathsf{Verify}(e, n_i, \mathcal{Y}_i, M_i, \sigma_i) = \mathsf{accept}$ for $i = 1, 2$ such that σ_i are not outputs of $\mathcal{SO}$;

(3) $\mathcal{CO}$ has been queried less than 2 times (that is, $\mathcal{A}$ can only have at most 1 user private key); and

(4) $\mathsf{Link}(\sigma_1, \sigma_2) = \mathsf{unlinked}$.

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Link}}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}].$$

Definition 6 (Linkability)**.** *A LRS scheme is linkable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Link}$ is negligible.*

4. Non-Slanderability.

Non-slanderability ensures that no signer can generate a signature which is determined to be linked by LRS.Link with another signature which is not generated by the signer. In other words, it prevents adversaries from framing honest users.

Non-Slanderability for LRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$, $\mathcal{SO}$ and the random oracle:

(a) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

(b) $\mathcal{A}$ may query the oracles according to any adaptive strategy.

(c) $\mathcal{A}$ gives $\mathcal{S}$ an event e, group size n, a message M, a set of n public keys $\mathcal{Y}$, the public key of an insider $\mathsf{pk}_\pi \in \mathcal{Y}$ such that pk_π has not been queried to $\mathcal{CO}$ or has not been included as the insider public key of any query to $\mathcal{SO}$. $\mathcal{S}$ uses the private key sk_π corresponding to pk_π to run $\mathsf{Sign}(e, n, \mathcal{Y}, sk_\pi, M)$ and to produce a signatures σ' given to $\mathcal{A}$.

(d) $\mathcal{A}$ queries oracles with arbitrary interleaving. Except pk_π cannot be queries to $\mathcal{CO}$, or included as the insider public key of any query to $\mathcal{SO}$. In particular, $\mathcal{A}$ is allowed to query any public key which is not pk_π to $\mathcal{CO}$.

(e) $\mathcal{A}$ delivers group size n^*, a set of n^* public keys $\mathcal{Y}^*$, a message M^* and a signature $\sigma^* \neq \sigma'$.

$\mathcal{A}$ wins the game if

(1) $\mathsf{Verify}(e, n^*, \mathcal{Y}^*, M^*, \sigma^*) = \mathsf{accept}$;

(2) σ^* is not an output of $\mathcal{SO}$;

(3) All of the public keys in $\mathcal{Y}^*, \mathcal{Y}$ are query outputs of $\mathcal{JO}$;

(4) pk_π has not been queried to $\mathcal{CO}$; and

(5) $\mathsf{Link}(\sigma^*, \sigma') = \mathsf{linked}$.

We denote by

$$\mathbf{Adv}^{\mathsf{NS}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}].$$

Definition 7 (Non-Slanderability)**.** *A LRS scheme is* non-slanderable *if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{NS}_{\mathcal{A}}$ is negligible.*

.1.3 Short Linkable Ring Signatures [6]

SLRS schemes are described by the tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$. For our SLRS, we define N as a safe prime product if $N = pq = (2p'+1)(2q'+1)$ for some primes p, q, p', and q' such that p' and q' are of the same length. We further denote by $\mathsf{QR}(N)$ the group of quadratic residues modulo a safe prime product N.

- **Setup:** $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$ is a function that takes λ as the security parameter and generates system-wide public parameters param such as the group $\mathsf{QR}(N)$, the length of the key, and a random generator $\tilde{g} \in \mathsf{QR}(N)$.

- **Key Generation:** $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is a function to generate key pair for each voter i. This function generates $(\mathsf{sk}_i, \mathsf{pk}_i) = ((p_i, q_i), y_i \in \mathcal{Y})$ for voter i where $y_i = 2p_iq_i + 1$.

- **Signature:** $\sigma \leftarrow \mathsf{Sign}(\mathcal{Y}, \mathsf{sk}, \mathsf{msg})$ is a function to generate the signature σ using all voters' public keys $\mathcal{Y} = \{y_1, y_2, \ldots, y_b\}$, the message to be signed, $\mathsf{msg} \in \{0,1\}^*$, and the voter's secret key sk. The output of this function is the accumulation of public keys v, the linkable tag $\tilde{y}$ and the signature σ'. For a voter i, the generation of SLRS is described below:

  1. Let $\psi \in \mathsf{QR}(N)$ be the public SLRS parameter that is generated once for all voters. We compute the witness w_i for voter i according to Algorithm 8 and Algorithm 9 in Section 5.2.7.
  2. We select parameters ℓ and u according to methods in [128]. Let $y \in S(2^\ell, 2^u)$ denote $|y - 2^\ell| < 2^\mu$. We compute the signature of message $\mathsf{msg} \in \{0,1\}^*$ for

$$\mathsf{SPK}\left\{ \begin{pmatrix} w & y \\ p & q \end{pmatrix} : \begin{array}{l} (w^y = v \bmod N) \wedge (y = 2pq + 1) \wedge \\ (y \in S(2^\ell, 2^u)) \wedge (q \in S(2^{\ell/2}, 2^u)) \wedge \\ (\tilde{y} = \tilde{g}^{p+q} \bmod N) \end{array} \right\}(\mathsf{msg}) \tag{1}$$

  3. Let σ' be the result of signatures based on proofs of knowledge (SPK). Note that the voter's linkability tag $\tilde{y}$ is uniquely determined by the voter's secret key. The result of the Sign function is $\sigma = (v, \tilde{y}, \sigma')$.

- **Verification:** $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(\sigma, \mathcal{Y}, \mathsf{msg})$ is a function to verify that the signature of $\mathsf{msg} \in \{0,1\}^*$ is from one of the valid voters and this voter has only voted once. With the given key set $\mathcal{Y}$, message msg, and signature σ, the verifier checks whether

$$v \stackrel{?}{=} \psi^{\prod_{y \in \mathcal{Y}} y} \bmod N$$

  and the validity of σ' respect to the SPK represented in Equation (1). If these two conditions hold, the Verify function returns accept otherwise reject.

- **Linkability:** $\mathsf{Link}(\sigma_1, \sigma_2) \rightarrow \mathsf{linked/unlinked}$ is a function to test the linkability of the given signatures σ_1 and σ_2. It extracts their respective linkability tags $\tilde{y_1}$ and $\tilde{y_2}$ from σ_1 and σ_2, respectively and returns linked if they are signed by the same person or $\mathsf{unlinked}$ otherwise.

Theorem 2. *Under the assumptions that Decisional Diffie-Hellman (DDH) problem over $\mathsf{QR}(N)$, the Link Decisional RSA (LD-RSA) problem, and the strong RSA (SRSA) are hard, the SLRS construction of A.3 is not only unforgeable in the random oracle model but also linkably-anonymous and non-slanderable w.r.t. the definition 1 to definition 4.*

Proof. The proof is given in [6]. □

.2 Public-Key Encryption

Syntax: A public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms and a finite message space $\mathcal{M}$ (which we assume to be efficiently recognizable). The key generation algorithm Gen outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, where pk also defines a randomness space $\mathcal{R} = \mathcal{R}(\mathsf{pk})$. The encryption algorithm Enc, on input pk and a message $m \in \mathcal{M}$, outputs an encryption $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ of m under the public key pk. If necessary, we make the used randomness of encryption explicit by writing $c := \mathsf{Enc}(\mathsf{pk}, m; r)$, where $r \xleftarrow{\$} \mathcal{R}$ and $\mathcal{R}$ is the randomness space. The decryption algorithm Dec, on input sk and a ciphertext c, outputs either a message $m = \mathsf{Dec}(\mathsf{sk}, c) \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$ to indicate that c is not a valid ciphertext.

Correctness: We call a public-key encryption scheme is correct if

$$\mathsf{E}[\max_{m \in \mathcal{M}} \Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq m | c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)]] \leq \sigma,$$

where the expectation is taken over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$.

Security: Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}$. We define the indistinguishable against Chosen-Plaintext Attacks (IND-CPA) game is shown as below, and the IND-CPA advantage function of an adversary $A = (A_1, A_2)$ against PKE (such that A_2 has binary output) as

$$\mathsf{Adv}^{\mathsf{IND\text{-}CAP}}_{\mathsf{PKE}}(A) := \left| \Pr[\mathsf{IND\text{-}CAP}^A \Rightarrow 1] - \frac{1}{2} \right|.$$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2. $b \xleftarrow{\$} \{0, 1\}$
3. $(m_0^*, m_0^*, st) \leftarrow A_1(\mathsf{pk})$

4. $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*)$
5. $b^{'} \leftarrow A_2(\mathsf{pk}, c^*, st)$
6. **return** $[[b^{'} = b]]$

.2.1 Paillier Encryption System [5]

Let $G = \mathbb{Z}_{n^2}^*$ and g be a random element from G, the Paillier encryption system is a randomised encryption scheme that encrypts the message msg by raising basis g to the power of msg and randomises it by a random factor. Given public key and the encryptions of msg_1 and msg_2, one can compute the encryption of $msg_1 + msg_2$ without knowing msg_1 and msg_2. We use $\gcd(v, w)$ and $\mathrm{lcm}(v, w)$ to denote the greatest common divisor and least common multiple of two values v and w, respectively. The quotient of a divided by b is denoted by $a \div b$.

- **Key Generation:** $(\mathsf{sk}_{\mathsf{Paillier}}, \mathsf{pk}_{\mathsf{Paillier}}) := \mathsf{Gen}_{\mathsf{Paillier}}(K_{\mathsf{len}})$ is the function to generate the secret key $\mathsf{sk}_{\mathsf{Paillier}}$ and the corresponding public key $\mathsf{pk}_{\mathsf{Paillier}}$ with the given key length K_{len}. We choose two large prime numbers p and q randomly and independently of each other and make sure $\gcd(p, q-1) = \gcd(p-1, q) = 1$. Let $\lambda = \mathrm{lcm}(p-1, q-1)$ and $L(b) = \frac{b-1}{n}$, where $b \in \mathbb{Z}_{n^2}^*$ and $n = p \cdot q$. Select random integer g where $g \in \mathbb{Z}_{n^2}^*$ and compute $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$. The public key is $\mathsf{pk}_{\mathsf{Paillier}} = (n, g)$ and the secret key is $\mathsf{sk}_{\mathsf{Paillier}} = (\lambda, \mu, p, q)$. We store p and q in our secret key as we will use these parameters to prove the correctness of the decryption in Paillier cryptosystem.
- **Encryption:** $C \leftarrow \mathsf{Enc}_{\mathsf{Paillier}}(\zeta, \mathsf{pk}_{\mathsf{Paillier}})$ Let $\zeta \in \mathbb{Z}_n$ be the plaintext to be encrypted. Select random $r \in \mathbb{Z}_n^*$ and compute $C = g^\zeta r^n \bmod n^2$.
- **Decryption:** $\zeta := \mathsf{Dec}_{\mathsf{Paillier}}(C, \mathsf{sk}_{\mathsf{Paillier}})$ Let $C \in \mathbb{Z}_n^*$ be the ciphertext, compute the plaintext as $\zeta = (L(C^\lambda \bmod n^2) \cdot \mu) \bmod n$.
- **Message Membership Proof of Knowledge [126]:** $\{v_j, e_j, u_j\}_{j \in \mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(C, \Upsilon)$. In [126], the authors propose an efficient method to prove that a given encrypted message is 1 out of n messages in a set. The non-interactive version of this proof of knowledge is described as follows. Let n be the RSA modulus from Paillier system, $\Upsilon = \{\zeta_1, \zeta_2, \ldots, \zeta_\rho\}$ the set of ρ encoded candidates, $\mathcal{P}$ be the set of n messages, and C the encryption of one encoded candidate. In this proof, the prover P convinces the verifier V that C encrypts the i^{th} message in Υ:

1. P picks κ randomly from $\mathbb{Z}_n^*$, $\rho-1$ values $\{e_j\}_{j\neq i}$ in $\mathbb{Z}_n$, and $\rho-1$ values $\{v_j\}_{j\neq i}$ in $\mathbb{Z}_n^*$. Then, (s)he computes $u_i = \kappa^n \bmod n^2$ and

$$\{u_j = v_j^n (g^{\zeta_j}/C)^{e_j} \bmod n^2\}_{j\neq i}$$

The prover sets $e \in \{0,1\}^L$ as the hash value of $\sum_{k=1,k\neq i}^{n} u_k$ (in our system, we set L as 80). The prover further lets $e_i = e - \sum_{k\neq i} e_k \bmod n$ and calculates

$$v_i = \rho \cdot r^{e_i} \cdot g^{\left(e-\sum_{j\neq i} e_j\right) \div n} \bmod n.$$

Finally, the prover sends $\{v_j, e_j, u_j\}_{j\in\mathcal{P}}$ to the verifier.

2. The verifier sets $e \in \{0,1\}^L$ as the hash value of $\sum_{k=1,k\neq i}^{n} u_k$ and checks whether $e \stackrel{?}{=} \sum_{k=1}^{p} e_k$ and that $v_j^n \stackrel{?}{=} u_j (C/g^{\zeta_j})^{e_j} \bmod n^2$ for each $j \in \mathcal{P}$.

- **Decryption Correctness Proof of Knowledge:** We define $(\delta, r) := \mathsf{PoK}(C, \mathsf{sk}_{\mathsf{Paillier}})$, which is the function to compute the plaintext δ and the random factor r to the ciphertext C with the given Paillier system secret key $\mathsf{sk}_{\mathsf{Paillier}} = (n, g, p, q)$. As Paillier system is bijective [5] meaning $\mathsf{Enc}_{\mathsf{Paillier}} : \mathbb{Z}_n^* \times \mathbb{Z}_n \to \mathbb{Z}_{n^2}$ is both one-to-one and onto. The prover sends δ and r to the verifier to prove that (δ, r) is the only pair to construct C.

  The main idea to compute $r \in \mathbb{Z}_n^*$ is described as follows: we denote $g(r) = r^n \bmod n^2$. The $g(r)$ can be calculated by $c \cdot g^{-m}$. Then, based on the Little Fermat theorem, $r^p = r \bmod p$, thus, $g(r) = r^n \equiv r^q \bmod p$. Since $\gcd(q, p-1) = 1$, there exists i_1 such that $q \cdot i_1 = 1 + k(p-1)$ and we get

$$g(r)^{i_1} = (r^q)^{i_1} = r^{1+k(p-1)} = r \cdot r^{k(p-1)} = r \bmod p$$

  Similarly, we denote i_2 as the modular inverse of p modulo $q-1$ and we get $g(r)^{i_2} = r \bmod q$. We finally get $r \bmod p$ and $r \bmod q$ respectively and apply the Chinese Remainder Theorem to obtain $r \bmod n$.