Account-Wise Ledger: A New Design of Decentralized System

Yi-Chen Liu*
University of California, Davis
Davis, California

Jingchao Fang
University of California, Davis
Davis, California

Jia-Wei Liang
University of California, Davis
Davis, California

ABSTRACT

Traditional blockchain is designed as a single line that contains every transaction between different nodes, and it needs to be fully copied to every participant to ensure the validity of the whole system. The main issue of this mechanism is that it will cause the "booming data problem", which means the whole blockchain may be too large to be stored by an individual.

Instead of taking everyone's transaction data together to form the blockchain, this paper brings up a new method called *Account-Wise Ledger*, which categorizes every transaction by account.

Account-Wise Ledger allows every individual to first, acquire the transaction history more efficiently since every ledger is an isolated unit that is stored by "some" members of the network (node) rather than "every" node. Second, the average data usage of each individual can be controlled under a specific amount because every ledger can be treated as an individual atom and grouped by different sets of nodes, storing some part of the data and not the entire data. In addition, the Account-Wise Ledger solves the issue of individuals purposely reclaiming private keys from others and it also provides a new method of preventing spams not by using current "proof of work method" that increases energy wastes day by day. All in all, with the power of Account-Wise Ledger, the whole data structure can remain divided, distributed and decentralized but fix the current concerns of blockchain.

KEYWORDS

blockchain, cryptocurrency, Bitcoin, distributed system, decentralized system, Byzantine fault tolerance

ACM Reference Format:
Yi-Chen Liu, Jingchao Fang, and Jia-Wei Liang. 2019. Account-Wise Ledger: A New Design of Decentralized System. In *Proceedings of UC Davis class (ECS 265)*. University of California, Davis, CA, USA, 12 pages.

1 BACKGROUND INTRODUCTION

Bitcoin has been established for almost a decade, and in recent years its value has always been a hot topic discussed by people. The main idea of Bitcoin is to use a data structure, which is also known as a blockchain, as a secure way to support the exchange of virtual currency and record the transaction history.

The rise of the willingness to do transaction in a decentralized systems has grown significant recently. A decentralized system can also be called as a trust-less and distributed consensus system.[13]It means that if one wants to send and/or receive money from someone, one does not need to trust in a third-party service. In comparison, when doing a traditional methods of payment, we need for example Visa or MasterCard to do the verification. Those third-party services keep their own private register which stores transaction history and balances of each account. However with bitcoin and a few other digital currencies, everyone has a copy of the ledger (the blockchain), so no one has to trust in third parties because anyone now can directly verify the written information.

To sum up, a centralized systems has a core authority that dedicates the truth to the other participants in the network. Only privileged users or institutions can access the history of transactions or confirm new transactions. While a decentralized systems have no core authority to dedicate the truth to other participants in the network. Every participant in the network can access the history of transactions to confirm new transactions.

Blockchains can be categorized into three major types: public blockchain, private blockchain consortium blockchain. Public blockchains are permissionless, meaning that anyone can join the blockchain and perform reading and writing without any permission. Examples of public blockchains include cryptocurrencies such as Bitcoin, Ethereum and Litecoin. By using public blockchain, cryptocurrencies ensure that it is completely open, everyone can join the network and everyone is able to perform transactions. Private blockchains, in contrast, are permissioned blockchains. The network place restrictions on participating and performing read and write. In a private blockchain, only nodes that have been given rights to access to the network can participate. To that end, private blockchain provides less anonymity because it is difficult for the network to give or deny permissions without knowing basic information about the nodes. Private chains are usually being used in enterprise or organization scenarios, where we want to make sure that the network is only accessible to a certain group of people. Thus, they are more centralized in nature. Consortium blockchains are somewhere in between, they are more centralized than public blockchains and more decentralized than private blockchains.

The design of the blockchain is a distributed ledger, which contains every transaction record in a single but gigantic data chain. This protocol requires every member who participates in this network to store a full copy of data in the whole chain. As time goes by, the whole chain would grow to an extremely huge level that none of any single participant can fully save. According to the latest static record in 2019, Bitcoin experienced high-level size growth, reaching nearly 242.39 gigabytes, since its creation[11]. This leads to a first problem: how to reduce the storing burden per participant? Jiaping Wang and Hao Wang[14] had provided a solution that divides a single blockchain into multiple sub-chain. To be more specific, their works were focusing on how to divide the whole blockchain system into multiple sub-networks, called "Zone". This

*All authors contributed equally to this research.

ECS 265, December 2019, Davis, CA, USA

solution might sound robust at first; however, in the long run, each Zone would eventually grow to a certain level that the same issue may occur again.

Cryptocurrencies, using blockchain as their data structure base, depends on miners to create blocks by performing proof-of-work to complete the transaction. Proof-of-work, by its original design, should be used to prevent spams. Its principle, according to Nakamoto's design[12], was a tool to solve the Byzantine Generals' Problem. Byzantine Generals' Problem can be seen as an adversary trying to disrupt the ledger's operation/network latency. It is said that a distributed ledger will inevitably have forks, which means that some nodes would consider the block A is the latest block, while some other nodes might consider others. However, since the reward comes along with solving puzzles and the longest chain would be recognized as genuine only, tons of participants compete with their peers to seek the candidacy position of the longest chain node. Even worse, since the benefits can only be acquired when a miner win the competition, lots of participants in the network are starting to seek any cooperate or collusion opportunity that try to form themselves as a "relative majority" to gain higher possibilities that vanquish others. Those co-operations are on climbing trend and would eventually lead to a serious 51% attack problem[1, 6].

Furthermore, the proof-of-work has another role that it acts as the only approach to issue money into the network, which causes many related problems. First and foremost, a fixed principle of money issuing (e.g. fixed reward per block creating) would lead to great inflation or deflation of money value. Second, by the greedy mind of humankind, every participant would do his or her best to solve proof-of-work puzzles in order to gain the reward. This "greedy action", which is the main motivation of the mining competition, may lead to great energy wasted on our planet with no actual contribution to our society. According to Digiconomist's statistics number, in October 2019, the latest power consumption of Bitcoin mining is 73.12 TWh per year globally and the trend is still climbing significantly[4].

In order to solve the issues mentioned previously from proof-of-work, Proof of Stake (PoS) was first introduced in a paper by Sunny King and Scott Nadal in 2012 and intended to solve the problem of Bitcoin mining's high energy consumption.[3] Rather than relying on the energy-dependent work of miners to add blocks, Sunny and Scott suggested an alternative method called "staking" where a deterministic algorithm would choose nodes based on the number of coins an individual had. In other words, stakers would have more chances of being selected to add a block to the chain and reap the reward if they "staked" more coins in their wallet. They hoped this would avoid the ever increasing energy costs and hash-rate difficulty of mining.

All in all, in proof-of-work, when adding each block to the chain, miners must compete to solve a difficult puzzle using their computer's processing power, and in order to add a malicious block, we need to have a computer more powerful than 51% of the network. The first miner to solve the puzzle is given a reward for their work. As for proof-of-stake, there is no competition as the block created is chosen by an algorithm based on the user's stake. In order to add a malicious block, you need to own 51%of all the cryptocurrency on the network. There is no reward for making a block, so the block creator takes a transaction fee.

For distributed system, Lamport described safety property as "bad things" do not happen during execution, it means that if a program starts with correct input, it does not stop until the correct output is produced. He also described liveness property as "good things" do eventually happen, which means a program with correct input must eventually terminate [9].

Double Spending Problem is the potential problem in which a single digital token can be used in more than one payment. Although the Bitcoin payment verification scheme is designed to prevent double-spending, the system requires a relatively long time to verify a transaction. When Bitcoin is used in the form of fast payment, studies shown that the measures recommended by Bitcoin developers for the use of Bitcoin are not always effective in detecting double-spending. Thus, the Bitcoin system can be fatally flawed in fast payment scenarios[8].

Byzantine Generals' Problem is a computer system problem that can be explained by a battlefield scenario. In a war, all generals might be separated by distance and can only communicate with each other via messages. Generals may defeat their enemies if and only if they agree on the same battle plan simultaneously with no central coordinators. The difficulties of this problem, which is also a major challenge exist in modern distributed systems, is that each node could be a traitor or the message they sent might lost. In order to make sure the network is reliable, we need to find a way to ensure every node will reach an agreement without being deceived or disconnected[10].

51% attack is a problem occurs when a party of miners controls more than half of the network. The party would be able to decide if each transaction gets approved or not given that the majority of the mining power is controlled by it. The party would also be able to reverse completed transactions, which could further lead to reusing digital tokens and double-spending problem. In practice, attacks can be performed even with less hash power rely on the ability to generate the longest chain in the long run[1]. Since the high volume of the competition of the mining task, a single-member who tries to utilize its own ability to defeat other rivals then obtain the reward is nearly impossible. A phenomenon that isolated participant tends to cluster together in order to gain much higher possibilities of block-creation. Thus, a seemly contest regarding block creation is the strongest stimulation for peers to gather. 51% attack will happen accordingly. Worth to be mentioned, 51% attack problem is different from the classical Byzantine Generals' Problem since 51% attack problem request not only the number of malicious nodes exceed 50% of the whole network but also ask those malicious nodes to collude together. That is to say, an individual who wants to control the whole network, the one should provide benefits to convince over one-half of participants of the network to obey its command. Thus, in order to tackle the problem, the new method should leave no reasons for replicas gathering together, namely, to eliminate potential motivations for everyone to collude.

2 MOTIVATION AND STEPS TOWARD SOLVING QUESTIONS

To summarize, based on the original design of Bitcoin, we will ask following two questions:

- How to reduce the restoring burden per participant effectively?
- How to come up with a better method of preventing spams, collusion, and competitions other than the original blockchain approach?

A new decentralized system with lower storage burden and meaningless completions is desired for every participant in this ecosystem. To fulfill the demands is the motivation of our work.

Account-Wise Ledger is the main idea in this study. Different from the original approach of the blockchain which mixes everyone's transaction data and information together into a single chain and requires all participants to store a full copy of the data on that chain in order to join the network, we categorize every transaction by account. Each account is a single ledger book that cannot be sliced into multiple pieces but should contain every transaction respected to the account only, which can be considered as an atom-like unit. With the power of Account-Wise Ledger, the complete blockchain can be divided and distributed among different sub-networks. Every member of the community has to store some parts of the whole chain of data only (i.e. every member store the transaction data of the sub-network that he is in). The average data size and space usage of each individual can be controlled within a specific amount.

Three-End Commitment is introduced to increase the security and to fairly distribute rewards. The motivation for mining races and potential fraud clusters are coming from the reward that gains from the block creation. Thus, we cancel the reward from each proof-of-work dedication in order to ease the greedy competition among block creators, we call them as *operators*. Instead, we fixed the total transaction fee, paid by the sender, for operators as the only reward. If there exists more than one operator trying to solve the same task, they will need to share the fixed reward amount together. This policy forces the competition to become meaningless since no matter how hard or how fast an operator solves a puzzle, the operator needs to share its reward with other peers who tackle the same task. Three-End Commitment means when operators finish their work, they need to send their results to both the sender, receiver, and every participant in the sub-network in order to ensure the transaction validity. Three-End Commitment chooses representative operators randomly from each sub-networks, then those operators will form a high council to make a judgment of a transaction, meaning the fault tolerance level of the protocol could be considered as a combination puzzle of a possibility plus majority problem. Furthermore, since we ease the motivation of the competition, the system leaves no reason for participants to collude. The 51% attack would be solved.

Contributions in this research might include the following:

- Account-Wise Ledger, a novel data structure that performs an atomic data unit pointed to each account, causes the division of a blockchain to become possible.
- Three-End Commitment, cooperates with canceling extra mining reward, ease the intention of collusion and increase the overall security.

By implementing these two ideas into the current blockchain database, we can solve the two issues highlighted above. In our Account-Wise Ledger model, there will be no mixed transaction data since we categorize each transaction into accounts. Every member of the network just have to store parts of the data from the whole blockchain instead of complete data from the blockchain. In addition, it enhances security and fairly distributes rewards to users, which reduce the unhealthy competitive environment.

3 SYSTEM DESIGN

Since the system design should be able to be divided into several sub-networks in order to reduce restoring burden per participant, the network needs a new data structure to act like an atom and a new communicating protocol to handle both in-network transactions and cross-network transactions. Account-Wise Ledger is introduced to work as an atom, a new data structure that can be distributed into different sub-network evenly, for each participant. The Three-End Commitment protocol is designed to handle the transactions both in-network and cross-network communicating scenarios.

3.1 Account-Wise Ledger

A single Account-Wise Ledger is designed as a container that records basic information of the account and the blockchain of transactions that is related to the specific account. To be more specific, a basic ledger would at least contains the following information: account number, address, transaction chain.

The account number is the hash of the public key. An account owner should hold both public key and private key to reclaim the account. The address is an indicator of which sub-network that the account is located in, denoting as a number k which $0 \leq k \leq K$ where K is the total number of sub-networks. The value of the address should be dynamic since a single ledger can be relocated to other sub-network for some purposes. The transaction chain is a simple and short blockchain such that every node inside of the chain must be related to the account, no matter if the account is the sender or the receiver of the transaction. The benefits of this design are: 1. One can check the balance of this account much easier and faster in order to verify whether if the account is honest or not, rather than has to scan the whole transactions among all participants in the network. 2. The account owner can decide to delete his/ her account from the network entirely in order to ensure privacy. 3. Restoring burden per participant can be reduced. 4. Transaction efficiency can be improved because senders do not need to worry about whether their tasks are truly recognized as the transaction blocks locate in the longest blockchain or not. Details as below.

Because the Account-Wise Ledger groups all the data related to a single account together, each ledger has atomic attribute that it cannot be sliced into multiple pieces. The ledgers, However, can be distributed into different groups or sub-networks, which means the whole network can be sliced or separated into pieces rather than stay as a sole complete one. Furthermore, by the power of slicing, the environment may allow every node not to have a full copy of entire data, meaning that only the data within the same sub-network as the node need to be restored by it. By doing so, the system allows every individual node to save more spaces compared to the traditional blockchain approach.

As figure 2 shows, Ideally, the whole network can be separated into multiple sub-networks. For example, we can set a network

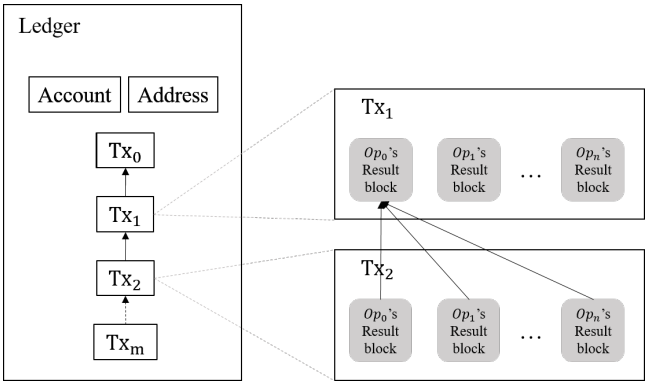

Figure 1: The structure of a single ledger.

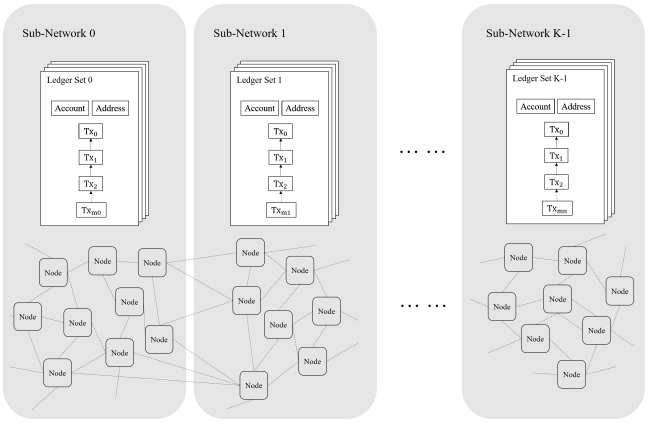

Figure 2: The separated restoring network model.

includes R replicas and there exist A accounts (the number of total ledgers is equal to the number of existing accounts, which is A in this case). Also, set K as the number of sub-networks. The number of replicas of each sub-set should be $\frac{R}{K}$ if the system distributes nodes evenly. The number of distributed ledgers that a single sub-set member has to restore should be $\frac{A}{K}$ which shows that Account-Wise Ledger structure may cost less disk space for a single node compared to the traditional approach. Furthermore, the possibility P of completely data loss from the whole environment is $P = \prod_{n=0}^{\frac{R}{K}} Q_n$, where Q is denoted as the possibility of a single crash; $0 \leq Q \leq 1$. If the number of nodes/replicas of each sub-network is large enough, the possibility P would be extremely small, which means that the larger the network is, the more secure the environment would be.

As mentioned, each Account-Wise Ledger contains a blockchain that stores every transaction history related to the account. The blockchain design, according to the original definition[12], has not to be a single path chain since the time latency may happen from time to time and forks may happen due to time latency. Nakamoto defined a rule that the system recognizes the longest chain as the only valid path when forks appear in the blockchain. In our design, however, the system recognizes every fork as valid. We call this mechanism as *endorsement*. As figure 1 shows, a single transaction is formed by the transaction blocks created by every operator in the network. Each new transaction block should be linked to the previously recognized transaction block to form the chain. Our design, however, has multiple forks for every transaction, which means the system must assign one block as the *chain-block* for the following participants to attach. Since each operator is also a member of the sub-network, meaning those operators have unique account numbers. The system would always send the latest block with the lowest operator's account number as the chain-block. Details will be explained in the next section.

3.2 Three-End Commitment

To fit the new Account-Wise Ledger structure, the transaction algorithm must be modified. First, we canceled the mining rule, which means there will be no block-creating bonus at all. The only incentive for block creating is the transaction fee. The reason for setting such a rule is trying to eliminate meaningless competition among participants and to moderate the cue of participant clustering. Therefore, the encryption difficulty, whose primary objective is preventing the DDOS attack and/or info spam of the block hashing would be limited to a certain level. Because the job of creating blocks is making the transaction actually happened, like the role of the bridge, we prefer to call those block creators as *operators*.

To initiate the transaction, the sender needs to create a transaction instruction called the *action* with his or her digital signature, asymmetric encryption of hash of action and the sender's private key. The action should contain the info of 1. The sender's account and the receiver's account. 2. The sender's address and the receiver's address. 3. The data that is meant to be transferred. For any transaction, the sender should create a pair of transaction tasks for a single transaction: departure-task and destination-task. Once the creation of the transaction task is done, the sender needs to broadcast the departure-task to the entire network and each sub-network will randomly choose one replica as the representative to handle the task. Also, once the departure-task has to be handled, a corresponding destination-task would be sent from the sender to the entire network then each sub-network will randomly choose one representative accordingly to handle the destination task respectively. The reason for two tasks, the departure-task, and the destination-task, are required is that the Account-Wise Ledger separates each individual into isolated unit, which contains its own specific transaction blockchain. The system needs two groups of operators to create two sets of transaction blocks to append to both two ledgers accordingly.

A new transaction protocol should consider 1. How to eliminate or moderate the 51% attack problem. 2. How to ensure security when the transaction protocol is changed. Rather than using the longest chain as the sole recognized path, we introduced the endorsement policy which treats each fork as the reinforcement to ensure a specific task. The protocol has a well-designed mechanism to determine operators regarding each transaction. Once the operators have been chosen, those operators who handle the same task should share the transaction fee, provided by the sender, together. This policy would encourage another phenomenon that because everyone in the network could be an operator, sharing a

fixed amount of transaction fee all together, becoming a malicious operator is meaningless since a faulty transaction block not only cannot influence the final result but also gain no reward due to being kicked out of the rewarding list. Also, The risk of 51% attack can be eliminated or moderated and the working efficiency would be increased because there is no "longest-chain competition" anymore. We will elaborate on this part in Section 4 Fraud and Risk for more details.

Without the longest-chain policy, we need a new way to recognize which blocks are valid. In our algorithm, all the blocks toward the same task are recognizable because of the endorsement policy. The question lays down to how to determine a list of operators and how to decide whose works can be recognized as valid. This problem is crucial because 1. The system should assign the transaction fee as the reward to those operators. 2. The group of operators who accept the reward should be settled eventually because we cannot allow any malicious operator who does nothing but always tackles old tasks in order to share others' rewards to gain its own fortune. Three-End Commitment asks three parties to get involved in a single transaction: 1. Senders and receivers. 2. Operators. 3. Everyone in the sub-network. In order to easily understand the protocol and the potential problems that it prevents, we use a theme to explain how it works. Let's define a school, which represents the whole network, is about to host an exam. A professor, represent as the sender, designed a test (transaction task) and ask some students (operators) to answer it. The rule is that each class needs to elect one student as a representative. Every representative needs to finish the test then we recognize the majority of the answers as the sole result. Thus, once each representative finishes the exam (by solving PoW), those representatives need to duplicate the answer sheet and share it with everyone in this "representative high council". When everyone has a group of answer sheets, they need to verify the answer and pick the majority of answers as the correct result. The names of authors of those answer sheets would be considered as the valid "rewarding list" and those answer sheets will be preserved. To formalize the above steps of communication operations, we list down the steps as below (Three-End Commitment Partial):

(1) The sender announces the transaction task to everyone in the network and the receiver.
(2) The receiver announces the acknowledgment to everyone. The message includes the task information that is sent from the sender and receiver's signature to prove that the task is accepted and verified by the receiver.
(3) Each sub-network randomly elect one node as the representative operator
(4) Every representative operator from each sub-network will create the block by solving Proof-of-Work, then broadcast to everyone in the whole network.
(5) Everyone starts to verify each block that it received, calculating the majority of the correct answer. The majority of the operators who create the correct answer would share the reward.

The above steps show just a partial phase of the whole Three-End Commitment since the system needs to create at least two transaction blocks for both sides, the sender's ledger and the receiver's ledger. After above steps are completed, the departure-task can be considered as solved. The next step for the sender is to announce the destination-task, containing the same information as the departure-task plus the latest chain-block in the sender's ledger which should be the result of the departure-task, to the receiver's sub-network. As to the receiver's sub-network, each participant in that sub-network would start to process the same procedures as the above steps. Thus, the complete Three-End Commitment is shown as below:

(1) The sender announces the departure-task to everyone in the network and the receiver.
(2) The sender's side transaction-block-creating procedure start:
    (a) The receiver announces the acknowledgment to everyone. The message includes the task information that is sent from the sender and receiver's signature to prove that the task is accepted and verified by the receiver.
    (b) Each sub-network randomly elect one node as the representative operator
    (c) Every representative operator from each sub-network will create the block by solving Proof-of-Work, then broadcast to everyone in the whole network.
    (d) Everyone starts to verify each block that it received, calculating the majority of the correct answer. The majority of the operators who create the correct answer would share the reward.
(3) The sender announces the destination-task to everyone in the network and the receiver.
(4) The receiver's side transaction-block-creating procedure start:
    (a) The receiver announces the acknowledgment to everyone. The message includes the task information that is sent from the sender and receiver's signature to prove that the task is accepted and verified by the receiver.
    (b) Each sub-network randomly elect one node as the representative operator
    (c) Every representative operator from each sub-network will create the block by solving Proof-of-Work, then broadcast to everyone in the whole network.
    (d) Everyone starts to verify each block that it received, calculating the majority of the correct answer. The majority of the operators who create the correct answer would share the reward.

The steps mentioned that the representative operators should be randomly chosen from each sub-network. This simple idea might be difficult since we have no centralized leader to do the decision. The protocol should provide a electing method that fulfill following attributes: 1. The election procedure should be perfectly even to make sure that everyone in the sub-network would have exactly same possibilities to be chosen. 2. The election procedure should be easily performed. That is to say, the time complexity of the procedure should be $O(1)$. 3. The election procedure should allow everyone in the same network to achieve consensus without ambiguous area easily and solely. To order to meet the criteria, a formula is introduced that everyone in the network should use the same formula to calculate the representative of the sub-network. We define ι_i as the representative operator's ID of the i^{th} sub-network, R_i as the number of total nodes of the i^{th} sub-network, and the Op_{i0} as the

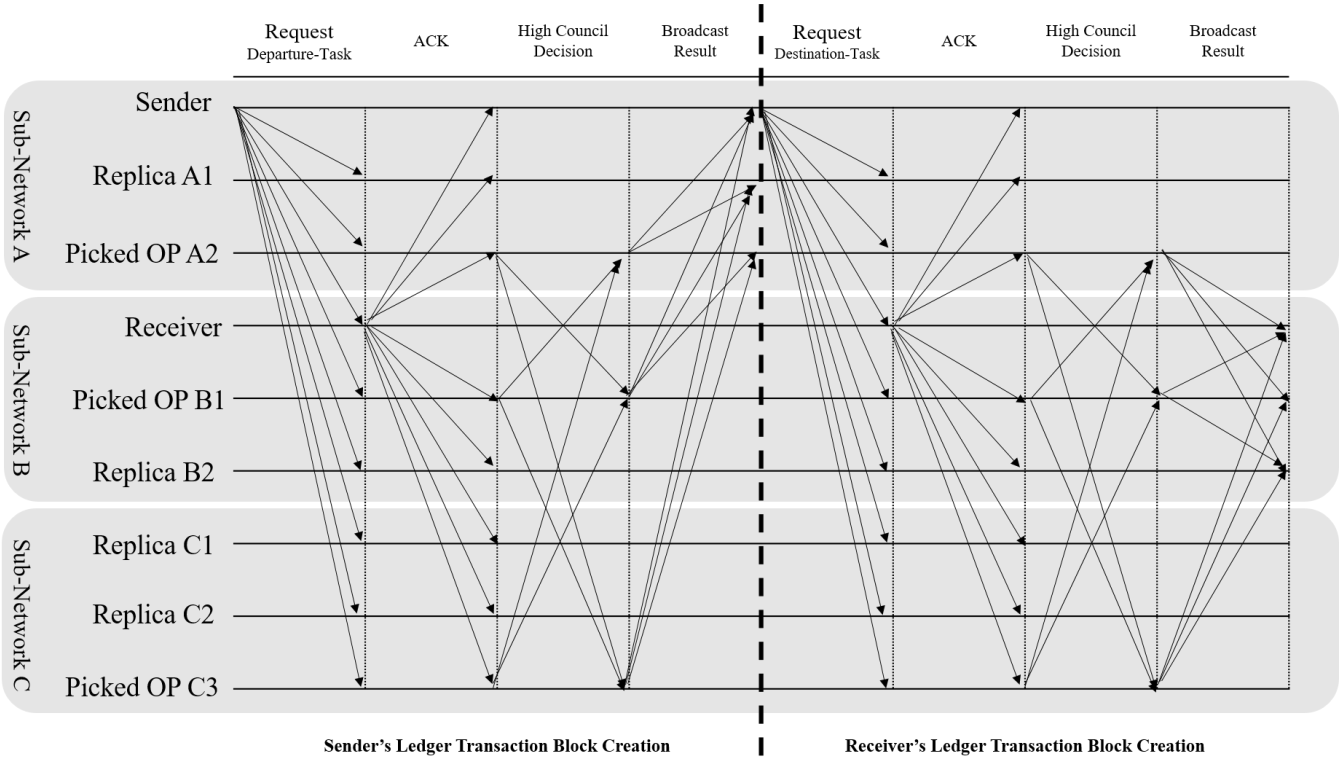

Figure 3: The Communication Diagram of Three-End Commitment Protocol.

first ID number of the i^{th} sub-network.

$$\iota_i = Hash(Task, ACK)\%R_i + Op_{i0} \tag{1}$$

The formula uses a hash function, feeding in the departure task or the destination task and the acknowledgement sent from the receiver as arguments, to calculate the ID number of the representative operator. In order to evenly pick the ID number, the hash function should have uniform distribution attribute. Also, the hash function should have $O(1)$ time complexity when it has fixed size inputs. By using the formula above, everyone in the network can easily reach consensus since the answer of the formula should come out the same answer regarding the same task and the same acknowledgement status that everyone received.

As figure 3 shows the diagram, Three-End Commitment recognizes the final result by the majority of the answers provided by operators. The idea of using majority as the sole result might seem similar to the PBFT protocol[2]. Three-End Commitment, however, requests each sub-network to randomly elect the representative operator first. In this case, the security level is dynamic and flexible respect to different attributes of the network in the Three-End Commitment. Also, Three-End Commitment asks the departure-task and the destination task to be handled in order since the receiver's transaction blocks need to contain the linkage info of the sender's transaction blocks in order to ensure the validity of data receiving and sending. To prevent any collusion happened in this communication, each role in the network can only perform one job that can only partially influence the communication but cannot impact the final decision. To be more specific, for example, the sender can propose a request only, the receiver can only sign the acknowledgment of the transaction task, and operators can only solve the puzzle and hand over the answer-sheet to others to wait for the final decision from everyone in the network. In the original blockchain design, the system uses the Proof-of-Work to increase the cost of being a faulty node for participants in the network in order to prevent some malicious attacks. In Three-End Commitment, the protocol not only uses the PoW to solve the problem but also utilized the random selections and the PBFT's benefits. The protocol can be considered as much maturer and safer.

The basic idea of the protocol is randomly select a group of operators to create the transaction blocks then the network would recognize the majority of the answers as the final result. This mechanism seems it can work pretty well without asking operators to solve any PoW. Indeed, the protocol can perform communication without using PoW to prevent fraud. PoW, however, can ensure the validity of the whole transaction data chain in each ledger. To be more specific, in our system, each ledger has its own blockchain to contain every single transaction regarding the ledger owner. If every transaction block of the blockchain can be formed without solving puzzles, spamming messages or related attacks like DDOS might occur. The whole network would become much vulnerable if the principle of PoW is eliminated completely. On the other hand, however, if we preserve the PoW principle, the transaction handling efficiency might be influenced magnificently. Thus, in order to tackle this trading-off situation, the requirement of PoW might

be controlled under a certain level, which would not take too much time to solve for each operator.

3.3 Account-Wise Leger v.s Database Sharding

Sharding has become a more and more popular topic in the field of the distributed databases since any database in this world that sees significant growth will eventually need to be scaled in order to accommodate the increases in traffic. The protocol we proposed - Account-Wise Ledger is a similar transformation of the idea of database sharding, our Account-Wise Ledger continues the benefits of sharding and mitigates its disadvantages.

Database sharding means to break up the data into two or more smaller chunks and the data held within all the shards collectively represent the entire dataset. There are two obvious categorizations of database sharding: horizontal partitioning and vertical partitioning[5]. Horizontal partitioning is a practice of separating one table's rows into multiple different tables. Each partition will have the same schema and columns, but entirely different rows, which means the data held in each is unique and independent of the data held in other partitions. Vertical sharding , vice versa, holds distinct rows and columns.

Traditional database shards exemplify a shared-nothing architecture. This means that the shards are autonomous, and they don't share any of the same data or computing resources. This results in many major disadvantages, for example, if data is sharded incorrectly, there's a significant risk that the sharding process can lead to lost data or corrupted tables. From our points of view, the most critical points of scaling down data sizes should be done in a way that ensures both the security and integrity of data. Therefore we introduce our Account-Wise Ledger.

Account-Wise Ledger shares the benefits of database sharding as:

(1) It spreads out the load and allows for more traffic and after processing
(2) It speeds up quest response times. For example, When you submit a query on a database in recent blockchain system (data not separated in accounts), it may have to search every data you're querying before it can find the result set you're looking for. By sharing one table into multiple tables, queries have to go over fewer data and the result sets are returned much more quickly.
(3) It is more reliable by mitigating the impact of outages. With an account-sharded database, though, an outage is likely to affect only a single shard. Even though this might make some parts of the application or website unavailable to some users, the overall impact would still be less than the case of the entire database crashed.

In addition, Account-Wise Ledger solves the major drawbacks of database sharding mentioned in the earlier paragraph (sharding incorrectly) because, in our Account-Wise system, all the related users in the account will do the verification. Secondly, Account-Wise Ledger won't encounter the issue of having sharded a database that eventually becomes unbalanced because each of our account is same-sized.

4 FRAUD AND RISK

Account-Wise Ledger structure and Three-End Commitment protocol have been introduced to solve data storing problems mainly. Related security issues, however, should be also considered wisely. Since the algorithm sliced the participants of a single transaction into three parties, the faulty scenarios need to be discussed in multiple angles to ensure that the algorithm is universal and comprehensive. Thus, we introduced the following cases: 1. The sender is a faulty participant. 2. The receiver is a faulty participant. 3. Both the sender and the receiver are faulty participants and they collude together. 4. The chosen operators are faulty participants. 5. Someone in the network is not one of the chosen operators but it sends out the transaction block to delude others in the network. Five cases mentioned above would be discussed accordingly.

4.1 Faulty Cases: Sender

We believe that the sender wants to be a faulty participant is rare since the sender can only undermine the validity of the transaction without gaining any benefit. However, there still exist two motivation for the sender to become a faulty participant: 1. The sender wants to destroy the whole system. 2. The sender wants to implement the Double Spending glitch to earn its own fortune.

If the sender is a malicious participant which wants to destroy the whole network, it can only target on the safety property of the distributed system rather than the liveness property since Three-End Commitment protocol is an asynchronous system which allows every node in the network perform the task individually, meaning no one can ask everyone in the network to stop their works. Thus, the safety issue of discussing how a malicious sender would harm the network is the main field needs to be solved here. Three-End Commitment asks the receiver as the person who accepts the transaction or not. In other words, if the sender is a malicious node, its transaction request would be rejected by the receiver. The transaction request would be dropped by the whole network since no one receives the acknowledgment from the receiver.

Another way for the sender to undermine the network is sending multiple different tasks to the network. More importantly, in order to delude the receiver, the sender would announce a correct request to the receiver to deceive the receiver in order to gain the ACK. This approach would not work since the ACK is not only contain the info that the receiver accept the task or not but also contain the full info of the task that the receiver gained. If a node receives a conflict pair, the task versus the ACK, the node would not use the task to calculate the representative operator in the sub-network. However, if a node in the sub-network receive a match pair and it knows that, base on the task it received, it is the representative operator, it will head to the high council and solve the puzzle no matter what. In this case, the whole network would still accept the final result from the high council in order to ensure the safety property. This rule is called as the Always-Accept Principle.

The design of the Three-End Commitment is an asynchronous system, meaning one can use the advantage of inevitable network latency to accomplish some disgraceful works. The Double Spending issue is one major pitfall that the protocol must solve. In a classical case, a sender broadcast its first task to the network; The sender does not wait for the completion of the first task handling

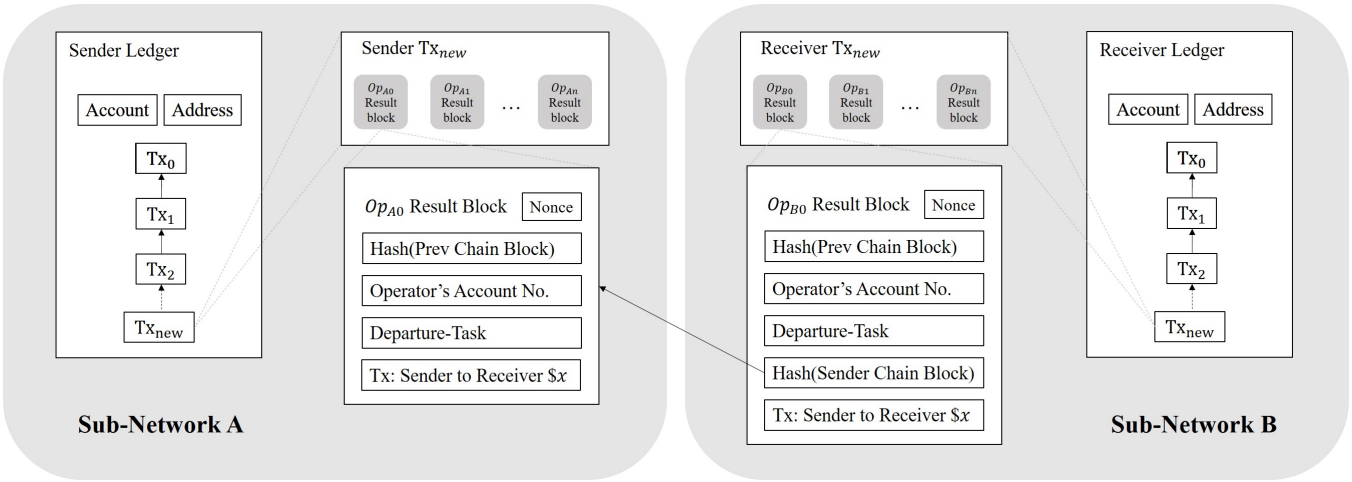

Figure 4: Data structure of Account-Wise Ledger system after performing Three-End Commitment.

but sends out a second task immediately. Normally, when everyone receives the first task, including the task operators, the task will be handled without problems. The second task, however, will be halt by the second high council since the first task has not been finished. If the second high council noticed that the second task is invalid due to the Double Spending issue, the second high council will make a decision that drops the task.

Another much complex scenario is that what if the sender broadcast multiple different tasks for the first request then ask for the second task simultaneously which might lead to not only the Double Spending problem occur but also it may harm the safety property of the network. In this case, the first high council can solve the problem by the Always-Accept Principle; The second high council would still be halt until the first high council comes out of the decision.

4.2 Faulty Cases: Receiver

The motivation for a receiver to act like a faulty participant is also sparse since being a faulty member gain no benefit for the receiver. however, we still cannot ignore this case since the reason for a receiver of being a faulty participant might be the intention of destroying the network. Similarly, even if a receiver intends to be a faulty member, it can only undermine the safety property of the network rather than harm the liveness of the system.

Base on the concept above, the way that a receiver may damage the safety property of the network is sending different statuses of acknowledgments to everyone in the network. In this case, some nodes may receive conflict messages that they have been informed the REJECT message but they receive a valid transaction block, sending from the high council, on the other hand. The Always-Accept Principle would still take the place to ask everyone who faces this contradict situation to accept the decision made by the high council to ensure the safety property.

Another situation is that what if the rejection message has been sent to some representative operator that the operator would not attend to the high council. In this case, the rest member of the high council would ask those absent operators about the status they received. Those absent operators need to provide the task and the acknowledgment that the accepted as solid proof to appeal for their innocence. Once the high council reviews the messages that those absent operators received are reasonable for them to miss the attendance, the high council would drop the task.

4.3 Faulty Cases: Sender and Receiver

The above two section has discussed either the sender or the receiver is a faulty participant and how the Three-End Commitment protocol deal with those cases. However, another much complex scenario that both the sender and the receiver are faulty participants needs to be considered and handled. Furthermore, two sub-cases should be managed in this scenario: 1. Both the sender and the receiver are faulty participants but they do not co-work together. 2. Both the sender and the receiver are faulty participants and they collude together.

When the sender and the receiver are not co-work with each other, both of them may broadcast different messages to different members in the network. A single node may face the following pairs of messages: 1. A valid transaction with a correct acknowledgment. 2. A valid transaction with a wrong acknowledgment. 3. An invalid transaction with a correct acknowledgment. 4. An invalid transaction with a wrong acknowledgment. For case 1 and case 3, a single node would perform perfectly since the acknowledgment information sent from the receiver would help the node make the right decision. As to case 2 and case 4, a single node, if it is not a supposedly chosen representative operator of its sub-network, it can do nothing but wait for the final result makes by the high council, according to the Always-Accept principle. On the other hand, if a node is a supposedly chosen representative operator, it would be miss guided by the wrong acknowledgment sent from the malicious receiver that it may absent or mistakenly attend the high council. In this case, the high council would still ask those supposedly attend but absent operators for their status and also verify the messages that the wrongly attend operators acquired to judge the fault was made by whom. If the majority of the high council believe that the fault was coming from the receiver, the high

council will drop the task. Wroth to be mentioned, even though the sender might be the faulty participant all the time, the high council would still proceed with the task that the sender request by voting the majority of the task consensus since the members of the high council has not been influenced by the sender at all.

What if the sender and the receiver collude together? To be more specific, whatever the task send from the sender, the receiver will broadcast the acknowledgment to endorse the task. In this case, multiple representative operators from a single sub-network will be elected. The high council, formed by all of the representative operators from each sub-network, will notice the problem then drop the task.

4.4 Faulty Cases: Picked Operators

We consider the case when both sender and receiver are honest, but the picked operator is malicious. As described in System Design section, we divide the whole blockchain into sub-chains and we divide all participants into sub-networks in order to reduce the restoring burden per participant, so that each participants only have to restore a copy of information on the sub-chain of the his or her sub-network. In our design, we canceled the traditional competitive mechanism, i.e., the need of doing proof of work to earn reward. Instead, we randomly select one participant from each sub-network, and we rely on the result of the voting from majority of the selected participants.

Suppose we have K sub-network in the whole network. Each of the sub-network may or may not have some malicious attackers that might be selected as the representative to join in the final vote. Let the probability of malicious attacker being selected from the first sub-network to be P_0, the probability of malicious attacker being selected from the second sub-network to be P_1 and so on. That is to say, the probability of malicious attacker being selected from the i^{th} sub-network to be P_i, and $P_i = \frac{f_i}{r_i}$ where f denoted as the number of faulty nodes of each sub-network and r denoted as total numbers of replicas of each sub-network. Thus, the total failure probability of the network is defined as:

$$P(K) = \begin{cases} P_0 & \text{if } K = 1 \\ \max(\prod_{i=0}^{\lceil \frac{K-1}{2} \rceil} P_i) & \text{if } K > 1 \end{cases}, 0 \le P \le 1 \tag{2}$$

If the distribution of faulty nodes among the whole network are even, the probability of malicious replicas should be the same among all the sub-networks. The probability of each sub-network regarding this situation is defined as:

$$P_t = \frac{\sum_{i=0}^{K-1} f_i}{\sum_{i=0}^{K-1} r_i}, 0 \le P_t \le 1 \tag{3}$$

Equation 3 represent the ideal probability of each sub-network. However, in practical approach, no one can ensure that every sub-network would have the same failure possibility since the number of faulty nodes would not evenly distributed among all sub-networks. Thus, a coefficient is needed to be introduced that represent the distributed status of the whole network. The distribution coefficient is defined as τ:

$$\tau = \frac{\sum_{i=0}^{K-1} \mid P_i - P_t \mid}{2KP_t(1 - P_t)}, 0 \le \tau \le 1 \tag{4}$$

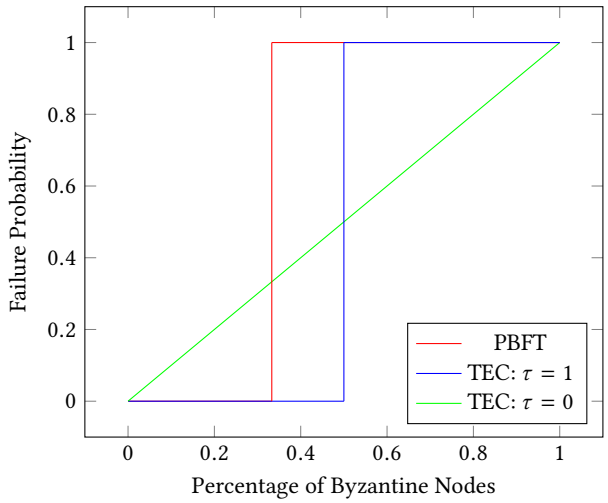

Figure 5: The changes of the failure probability of a transaction corresponding to the number of τ.

According to Equation 4, When the distribution of faulty participants is extremely unevenly, $\tau = 1$. On the other hand, When the distribution of faulty participants is extremely unevenly, $\tau = 0$. In order to discuss the meaning of the value of τ, we consider two extreme cases, and all other cases should fall in between.

4.4.1 When the malicious attackers are distributed extremely unevenly into different sub-networks. The definition of extremely unevenly distribute malicious attackers to each sub-network is that all of the faulty nodes are all squeezing in several sub-networks, leading the ratios of faulty nodes and total nodes in those specific sub-networks are all 100% for each. Furthermore, let's consider the case when over $\frac{1}{2}$ of all participants in the whole network are malicious, and they are all concentrated into several sub-networks, by definition. In this case, the representative operators chosen from those degenerated sub-networks are 100% faulty, and in the rest of the sub-networks, the probabilities of malicious attackers are being picked are 0%. Since our final vote relies on the majority, it is obvious that the attackers will 100% win over the honest participants. On the other hand, when less than $\frac{1}{2}$ of all participants in the whole network are malicious, the result is the opposite. In over at least 50% of all the sub-networks, the probabilities of malicious attackers are being picked are 0%, and in the rest of the sub-networks, the probabilities of malicious attackers are being picked are 100%. According to our design, the honest participants will 100% win over attackers. So the graph of the probability shows as Figure 5:

Compare to the PBFT, in the extremely unevenly distributed cases, the Three-End Commitment protocol can ensure the tolerance level is 50%, which is better than PBFT that its tolerance level can only survive under $\frac{1}{3}$ total nodes are faulty.

4.4.2 When the malicious attackers are distributed extremely evenly into all sub-networks. If the distribution of faulty nodes is even, every sub-network share the same probability of choosing a malicious operator as the representative, namely $P_0 = P_1 = \cdots = P_{K-1} = P_t$.

According to Equation 2, we can simplify the formula by replacing the probability of choosing faulty representatives of each sub-network with the same value P_t. Thus, in this extremely evenly distributed case, we can rewrite Equation 2 as the following:

$$P(K) = \begin{cases} P_0 & \text{if } K = 1 \\ P_t^{\lceil \frac{K-1}{2} \rceil} & \text{if } K > 1 \end{cases}, 0 \leq P \leq 1 \tag{5}$$

For each transaction, the original blockchain system may produce a completely wrong transaction block if a faulty node has mighty power that surpass all the rest participants in the network. Our system, on the other hand, can still partly keep the system safe. In other words, compare to the original blockchain design, Three-End Commitment would not be influenced by the calculation ability of nodes. Since one issue exist in the 51% Attack problem is that more and more participants eager to pursue much powerful calculating ability in order to gain the transaction prize, the original blockchain system would be dominate by one or few nodes which can always control the result of each transaction. In the worst case, if those nodes with dominated power are malicious node, none of any transaction handled by them would be considered as correct. Three-End Commitment, on the other hand, ignore the calculating power of individuals but cares the number of sub-networks and the possibilities of choosing faulty nodes from each sub-network.

Moreover, based on the property of power function, we claim that the total probability of the whole network failed P decrease as the number of sub-networks K increases, which means that the probability of the final result being dishonest decreases as the number of sub-networks increase. For example, When K=1, there is a single network in the system, and the graph is linear, shows as the red line in Figure 6. If the number of sub-networks increases, namely the value of K increase, the failure probability graph would tend to approach the blue line in Figure 6. To be more specific, the more sub-networks exist, the securer that the whole system is. To enhance the security of the system, we need to divide the whole network into more sub-networks, and we claim that when K is large enough, the security level of the system is better than Bitcoin. However, the efficiency of the whole system decrease while K increases since more cross-network communication is needed when there are more sub-networks. Thus, there is a trade-off between system security and system efficiency.

If a malicious participant wants to modify a historical transaction from a certain ledger, it needs to revise the whole blockchain inside the ledger since the chain was designed to use a hash function as the pointer to link every block together. Furthermore, even if the faulty node finishes the modification task, it needs to convince over 50% of members in the sub-network to use its copy. Thus, the question would lay down how a malicious participant convinces others to collude together. In the original blockchain model, everyone can gain a reward if they successfully solve the PoW puzzle and be the sole winner of the whole chain. Everyone would like to collude together in order to gather their calculating ability to increase the chance for them to win the prize. In our model, since we leave no reason for people to collude together, even some of those participants might want to become a faulty node in order to gain its own benefit, it actually raises the barrier for people to collude together. The 51% attack requires not only over one-half participant to become faulty members but also ask those malicious nodes to obey to one single leader and co-work together. Three-End Commitment eliminate the motivation for people to cooperate, meaning the 51% attack problem would not happen. Furthermore, by controlling the coefficient of τ and K, the system can survive each transaction over one-third or even one-half of people are spiteful, working better than PBFT and the original blockchain.

To sum up, we demonstrated that two major parameters would influence the total security of the whole system: K-the number of sub-networks and τ-the distribution coefficient. Ideally, for security purpose, the system designer can choose to control the value of τ nearly 1 in order to make sure the failure would never occur when the number of total faulty nodes is under 50% of total nodes in the whole network. This result works far better than traditional PBFT, which can only survive when there are only $\frac{1}{3}$ faulty nodes to total nodes. On the other hand, if the system designer wants the whole network can have chances to survive when the ratio between faulty nodes and total nodes over 50%, the value of τ should be controlled nearly 0 and the number of sub-networks should be set as large as possible.

4.5 Faulty Cases: Fake Operators

People in the whole network can pretend themselves as the representative operator in order to squeeze themselves into the high board or to deceive the high council that forces the high council to drop the task every time. This approach might not succeed since a person who claims itself as a member of the high council must provide evidence as proof. The evidence includes the departure or the destination task, and the acknowledgment sent from the receiver it acquired. One can easily verify the identity of a person who is truly a high council member or not by using the task and the acknowledgment info as the inputs to calculate Equation 1 to inspect whether the outcome of the formula matches the person's ID or not. This process is extremely fast since Equation 1 used a hash function only which can only take $O(1)$ time. For those malicious members, however, in order to counterfeit the identity, they need to create a fake task block and a fake acknowledgment to pass the verification. This calculation requires a pretty long time that it is impossible to be solved before the high council tackle a transaction and then dismisses it. Even if a faulty member has such unparalleled power, it can only become a single member of the high council who has high chance to become the minority of the judgment, which cannot influence the final result, only. Therefore, the cost of being a fake operator is way too high and the benefit may gain from this process is extremely limited. The problem may not influence the validity of transactions.

4.6 Transaction Latency

Speaking of transaction latency, it is mostly caused by network latency. Network latency can be defined by the time it takes for a request to travel from the sender to the receiver and for the receiver to process that request. In other words, network latency can be described as the round trip time from one point to another point. Undoubtedly we would want this number as close to 0 as possible. However, there can be a few things at play preventing network

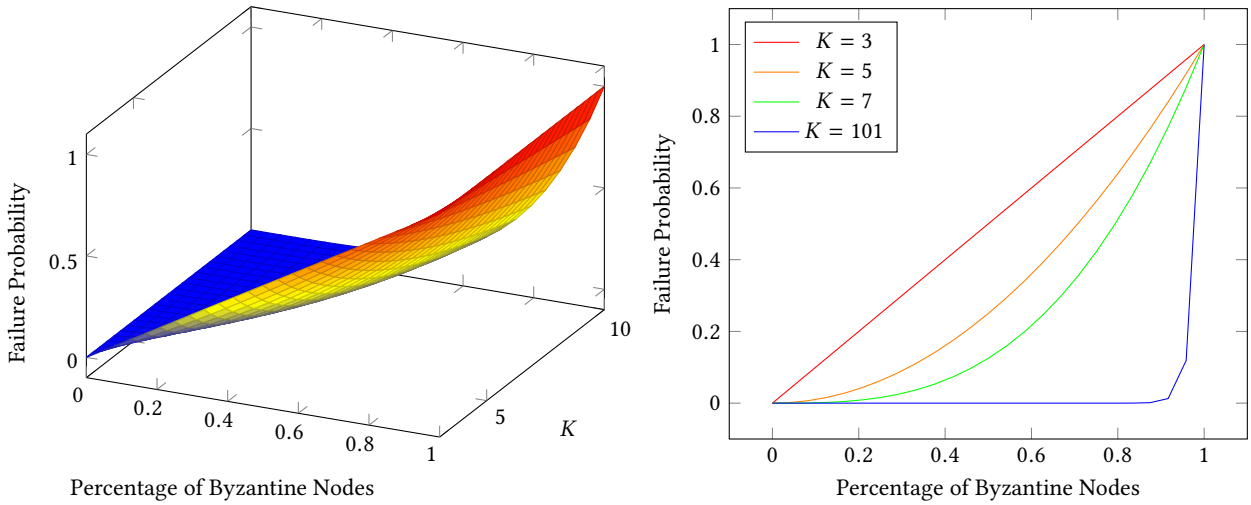

Figure 6: The changes of the failure probability of a transaction corresponding to the number of K. Assume $\tau = 0$

latency to reach 0, which in the end causes the transaction latency to appear.

There are several factors that can affect transaction sequence and latency:[7]

(1) Transactional processing order: Ordinary database processing maintains transactional consistency by applying transactions in the exact order in which they occur. For example, a withdrawal from your bank account will succeed only if the bank first completes your transfer to that account, and make sure you have enough money.
(2) Latency: Low latency is considered a high priority when replication processing. In change data capture, we lower latency and maintain relative transactional consistency by deploying more subscriptions. For example, we can redistribute the replication mappings in an existing subscription into multiple subscriptions.
(3) Parallel Processing: Information management professionals sometimes refer to concurrent apply processing as parallelism, because the target engine applies changes to the target database concurrently.
(4) Subscription Design: We can design our subscriptions to address the needs for low latency and precise order. For example, precise ordering is unlikely if replication stops for a subscription and a transaction cannot finish processing until you restart the subscription. Since subscriptions maintain independent restart positions, the target engine will apply later transactions before the original transaction can finish.

In fact, Three-End Commitment is not a latency-free protocol. That is to say, Three-End Commitment has no mechanism to prevent or reduce the occurrence of network latency. However, Three-End Commitment can survive when network latency happened since the protocol is an asynchronous design. Three-End Commitment allows multiple tasks are handled at the same time. In this case, it is inevitable for a single node has been chosen as representative operators for many different tasks. If any latency phenomenon occurs, members in the high council would ask that operator directly. The delayed operator could either choose to send the result or remain silent. The high council would treat any operator who remains silent like the one who losses the connection then abort it from the high council.

5 CONCLUSION

Our designed structure: Account-Wise Ledger helps individuals lessen the burden of entering the network and helps the whole system to work more efficient. With the design of categorizing every transaction by account (called sub-network in this research), from an individual's standpoint, it reduces the storing burden from every user and allows every individual to acquire the transaction history with less loading. In addition, if we look from the whole system standpoint, the Three-End Commitment mechanism, which we are using in this study, means to choose representative operators from sub-networks, and hold a high council to make decisions. It changes the traditional difficult PoW into a low burden or no barrier PoW since we believe that the motivation for mining races and potential fraud clusters are coming from the reward that gains from the block creation which leads to high energy waste and increasing 51% Attack risk.

In this research, we mainly focus on pointing out and solving the main issues from the current blockchain systems, such as the Double Spending, energy-wasting, and 51% attack. When designing our protocol, we tried not to suffer the same issue as the current blockchain system. Different from database sharding, which also separates the system into small accounts, our Account-Wise Ledger solves the most common database sharing issues as well. First, we will not shard incorrectly, since all the related users in the account is an uncut atom design. Secondly, we will never encounter an unbalanced database because the whole system can monitor the storing burden of each sub-network and slice an unbalanced sub-network into multiple sub-networks in order to control the data size of each sub-network evenly.

In order to show the security level of our designed protocol, in this research, we discussed different situations of encountering fraud or risk and how to solve them. Our protocol shows that we could ease the intention of collusion and increase overall security. In one of the scenarios discussed, we found out that our protocol compared to PBFT, in the extremely unevenly distributed cases, the Three-End Commitment protocol can ensure the tolerance level to 50%, which is better than PBFT that its tolerance level can only survive under $\frac{1}{3}$ total nodes are faulty. Furthermore, we discuss the transactional latency issue that happens all the time when things related to the Internet.

In sum, with the power of the Account-Wise Ledger structure and the Three-End Commitment protocol, a decentralized environment can be much securer and faster. Numbers of novel ideas and methodologies have been introduced to solve contemporary blockchain issues that countless studies nowadays try to manage. Through this research, we pointed out critical keys that deserve further discussions: eliminate reasons for participants to gather, using the Account-Wise data structure to shard a decentralized system and properly use the random technique to penetrate the security ceiling of the majority-consensus method or Nakamoto's consensus approach.

6 LIMITATIONS AND FUTURE WORKS

In our design, we believed that the Account-Wise Ledger and the Three-End Commitment protocol can solve problems described before. However, some works still need to be follow up, implemented and verified in the future. The robust proof is required in order to strengthen the persuasive credibility of this study. The topics are discussed as follows.

6.1 Security: TEC v.s. Nakamoto's Blockchain

The security issue that we have covered in this research has two main view angles: the correctness of each transaction handling and how easy that a record in the network could be modified by any malicious participant. In the traditional blockchain design, Nakamoto's approach, no matter which security issue that one concentrates, a blockchain system cannot survive when the number of faulty nodes over 50% of all nodes in the system. The Account-Wise Ledger structure and the Three-End Commitment protocol, on the other hand, are far complex and difficult to be evaluated compared to Nakamoto's model. Though we concluded that two arguments, τ and K, would significantly influence the security level of the network in our design and In Section 4, this study provides some edge cases that can perform much securer than the traditional blockchain or PBFT, we did not provide a mathematical formula that how to control these two parameter can guarantee the system can completely surpass Nakamoto's design for now. To find out the formula is one of the future works indeed.

6.2 Security: Co-works between τ and K

We defined a new concept, distribution coefficient τ, to describe the degree of dispersion of faulty participants in sub-networks. We described the extreme cases when τ=0 and τ=1 respectively. Also, we demonstrated how the value of K changes would influence the security level when τ is fixed at a certain level. However, we did not provide a combined formula to monitor the changes in the security level of the Three-End Commitment respecting the change of τ and K together for now. We believed that when $0 < \tau < 1$, the security level of the system might survive all time till 100% of participants are malicious. In this case, when the number of faulty nodes is under $\frac{1}{3}$ of all nodes, the security level of the system may in between Nakamoto's approach and PBFT if the number of K is large enough. Also, after $\frac{1}{3}$ or even $\frac{1}{2}$ of all nodes are malicious, the Three-End Commitment protocol is the only method that can survive. In the future, we will provide a solid mathematical proof to verify the concept described above is valid.

6.3 Efficiency

Three-End Commitment applied a random technique in order to penetrate the security ceiling of the majority-consensus method. Furthermore, in order to increase the security level of a scenario that a low number of nodes are faulty, Three-End Commitment asks each sub-network to randomly elect one representative operator to gather together and form the high council. This part can work extremely fast since the protocol did not ask everyone to communicate with each other. However, the communication inside the high council requires time in fact. The consensus method that the high council would use is a majority-based approach. Any better way to boost the handling efficiency inside the high council is the future work for the research to concentrate.

REFERENCES

[1] Martijn Bastiaan. 2015. Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin. In *Availab le at http://referaat. cs. utwente. nl/conference/22/paper/7473/preventingthe-51-attack-a-stochasticanalysis-oftwo-phase-proof-of-work-in-bitcoin. pdf.*

[2] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[3] Cointelegraph. 2017. *The History and Evolution of Proof of Stake.* Retrieved February 7, 2019 from https://cointelegraph.com/news/the-history-and-evolution-of-proof-of-stake#:~:targetText=Proof%20of%20Stake%20(PoS)%20was,to%20maintain%20the%20Bitcoin%20network.

[4] Digiconomist.net. 2017. *Bitcoin Energy Consumption Index.* Retrieved October 9, 2019 from https://digiconomist.net/bitcoin-energy-consumption

[5] Mark Drake. 2019. *Understanding Database Sharding.* Retrieved February 7, 2019 from https://www.digitalocean.com/community/tutorials/understanding-database-sharding

[6] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102.

[7] IBM. 2018. *Understanding database transaction sequence and latencyt.* Retrieved February 7, 2019 from https://www.ibm.com/support/knowledgecenter/en/SSTRGZ_11.4.0/com.ibm.cdcdoc.classiccdcforzos.doc/concepts/iiycdcdbtxns.html

[8] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 906–917.

[9] Leslie Lamport. 1977. Proving the correctness of multiprocess programs. *IEEE transactions on software engineering* 2 (1977), 125–143.

[10] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

[11] Shanhong Liu. 2010. *Size of the Bitcoin blockchain from 2010 to 2019, by quarter (in megabytes).* Retrieved October 9, 2019 from https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/

[12] Satoshi Nakamoto et al. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[13] Ameer Rosic. 2016. *Proof of Work vs Proof of Stake.* Retrieved February 7, 2019 from https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/

[14] Jiaping Wang and Hao Wang. 2019. Monoxide: Scale out Blockchains with Asynchronous Consensus Zones. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19).* 95–112.