Fall 12-14-2018

GRADUBIQUE: AN ACADEMIC TRANSCRIPT DATABASE USING BLOCKCHAIN ARCHITECTURE

Thinh Nguyen
San Jose State University

GRADUBIQUE: AN ACADEMIC TRANSCRIPT DATABASE USING BLOCKCHAIN ARCHITECTURE

Presented to

Dr. Jon Pearce

Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for

CS 298

By

Thinh Nguyen

Dec 2018

Abstract

Blockchain has been widely adopted in the last few years even though it is in its infancy. The first well-known application built on blockchain technology was Bitcoin, which is a decentralized and distributed ledger to record crypto-currency transactions. All of the transactions in Bitcoin are anonymously transferred and validated by participants in the network. Bitcoin protocol and its operations are so reliable that technologists have been inspired to enhance blockchain technologies and deploy it outside of the crypto-currency world. The demand for private and non-crypto-currency solutions have surged among consortiums because of the security and fault tolerant features of blockchain. To introduce blockchain concepts, we survey the three most popular blockchain architectures: Bitcoin, Ethereum, and Hyperledger Fabric. We then build Gradubique, a blockchain network built on top of Hyperledger Fabric. Gradubique allows instructors from any school to post exam and course grades to the Gradubique network. Employers and graduate schools can extract transcripts from Gradubique. Security is guaranteed by the blockchain technology. Standardization and translation of transcripts can be built into the network, and the distributed nature of the network can make it virtually cost-free.

Keywords: **Blockchain, Bitcoin, Crypto Currency, Ethereum, Smart Contract**.

Table of Contents

INTRODUCTION

Ledgers have been used from ancient times to record ownership of valuable things including property, assets, and money [1]. For example, thousands of years ago, the Egyptians recorded the inflow and outflow of grains which were brought in and taken out from the government store houses [2]. Over time, the type of ledgers has evolved into clay tablet, papyrus, vellum, and paper, respectively. Today, information technology not only transformed the traditional paper-based ledger into the digital ledger, but also has proposed multiple algorithms and techniques to secure and automate the book keeping process.

Traditionally, official ledgers are maintained at the central points such as banks and warehouses. These ledgers are the final words on who owns what. Recently, the concept of distributed ledgers has begun to replace this idea. In this case, there are many copies of a ledger, each maintained by a different bookkeeper. Transactions are broadcast through a network, and each bookkeeper updates his ledger. Protocols are in place to deal with mistakes and frauds.

The distributed ledger technology (DLT) obviously expedites the cross-border financial industry. However, in this internet era, internet of trust becomes a debatable issue of digital transactions. Centralized system proponents think that a centralized system is easier to control, cost effective, and (one point of trust is) better than multiple points of trust. In contrast, decentralized system advocates believe a centralized system is the single point of failure which includes fault tolerance, security (central point of attack), and trust (administrators' reputation). In recent years, blockchain has become the key concept to build a decentralized system which distributes the trust from a central point to multiple nodes in a network. Bitcoin is the first public blockchain application that utilizes the decentralized idea

of a currency system which is historically and centrally controlled by the government. In addition, businesses involve more than just financial transactions, Ethereum emerged as a public blockchain solution not only to process financial transaction, but also to record and verify contracts (called smart contracts).

However, these public blockchain solutions include disadvantages that deter consortiums from adopting them. First, the entities that participate in a public blockchain can be anonymous while the business environment needs participants who must be legally identified. Next, scalability and customizable consensus algorithms are not well supported on public blockchains. As a result, many private blockchain solutions are offered to solve these problems such as: HyperLedger by Linux Foundation, Corda, TenderMint, Chain.com, etc. In this paper, we will approach the concept of blockchain in chapter II, then we explain the details of the real world blockchain solutions which includes Bitcoin, Ethereum and HyperLedger Fabric in chapter III. Toward that end, on chapter IV, we develop a sample application on HyperLedger Fabric called Gradubique, a suite of applications for sharing academic transcripts among universities, employers, and students. Finally, chapter V envisages the future works.

BLOCKCHAIN OVERVIEW

A. History of Blockchain

In 1991, Haber and Stonertta [3] proposed techniques to protect the integrity of digital documents using timestamps and hash functions. The concept is to timestamp documents in chronological order and link signed certificates of documents using a hash function in the way that the current certificate request will be included in the next certificate request. In 1997, Adam Back [4], a British cryptographer, invented the hashcash algorithm which is considered the first

notion of proof-of-work. Hashcash is used to prevent spam emails. Hashcash algorithm issues a token embedded into an email which can be verified by the spam filter systems. This one-time used token is the evidence to prove that the received email was created by a computer which spends valuable hashing power to compose the email. Later, Hal Finney [5] presented the reusable proof-of-work (RPOW) theory in the IBM Research program to extend the reusable form of hashcash token.

In 2008, Satoshi Nakamoto [6] published a paper to propose a peer-to-peer digital cash system which brought the first digital currency system to the internet. This peer-to-peer network allows users to directly send coins to each other in the form of a RPOW token which is named Bitcoin. First, transactions are bundled into a block, then this block is chained with previous blocks by the block's hash value. In addition, when a Bitcoin transaction is validated, its embedded locking script and unlocking script are executed in sequence to see if it satisfies the spending condition. Although the functionality of the script embedded in bitcoin transaction is very limited, it ignites the notion of smart contracts in the next generation of blockchain.

In 2013, Vitalik Buletin [7] created Ethereum, a smart contract based blockchain network, which provides tools and an environment to develop decentralized applications on a cryptocurrency-based blockchain network. In 2017, Enterprise Ethereum Alliance (EEA) [8] was founded by a group of companies to standardize Ethereum software that can be used by businesses around the world to track financial contracts. Along with Ethereum, there are many other blockchain solutions proposed to enhance blockchain architecture. With a vast number of blockchain platforms and massive adoption of blockchain solutions taking place, a new generation of internet is emerging, the internet of blockchain.

B. Blockchain Architecture

A blockchain network is a peer-to-peer network in which a peer (also called bookkeeper, miner, or node) is responsible for validating transactions, creating new blocks, and verifying the validity of newly created blocks. A majority of peers in the network has to agree on a certain rule of the network not only to confirm the validity of transactions in a block, but also force a newly created block to be compliant with the predefined agreement. In other words, all the peers have to follow a known consensus algorithm to protect the data integrity without the necessity to check whether a node in the network is honest or malicious. Fig. 1 below is a class diagram of a generic blockchain network that constitutes the following main elements: peer (miner, bookkeeper, or node), consensus, and blockchain. Agent is a base class that provides the application program interface (API) for clients to interact with blockchain network. Client can be developed in the form of a desktop, web, or mobile application which allow users to request IDs to join the network before creating any transaction.

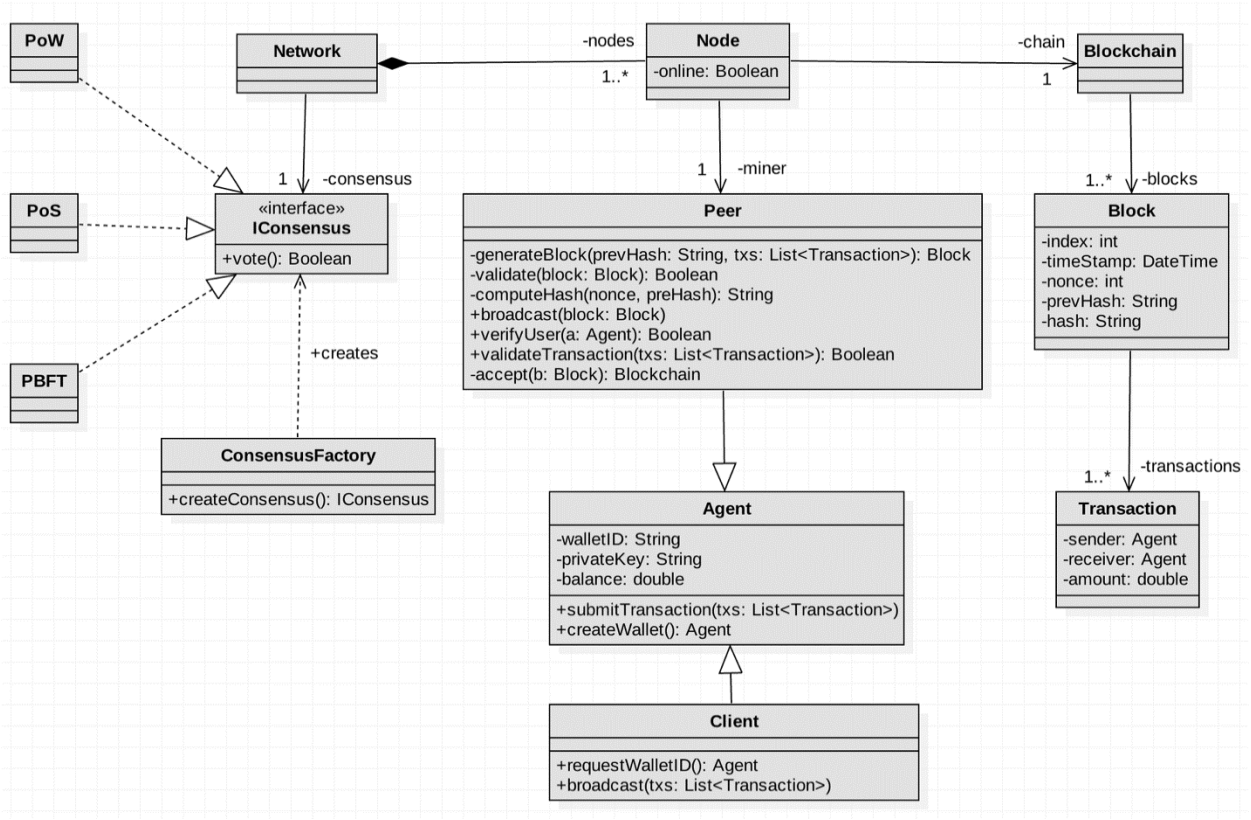


Fig. 1. Blockchain class diagram.

1) Block

A block is a container that stores meta and transaction data. Meta data is stored in a block header. For example, a block in the bitcoin blockchain stores hash value, previous block hash value, nonce, timestamp, size, number of transactions, difficulty of the block and so on. A block also contains a list of transactions, and these transactions can be valid or invalid depending on the architecture of the blockchain solution. For example, all of the transactions included in a block of the bitcoin blockchain must be valid transactions, while a transaction in the Hyperledger Fabric blockchain is not necessarily a valid one. Bitcoin blockchain and HyperLedger Fabric blockchain will be discussed in detail in the next sections.

2) Blockchain

Blockchain is literally a chain of blocks linked together by pointers which are indicated by hash values of the previous blocks. This mechanism guarantees that data in a current block cannot be tampered because if a single bit of data in the current block changed, it would change the hash value of the block. Sequentially, it not only breaks the link between the current block and the next block, but also decouples the next block with the block after that and so on. Blockchain is also known as a distributed ledger in a decentralized peer-to-peer network in which a peer can hold either a full copy or a light copy of a blockchain file. Because blockchain is distributed all over the network and is tamper-proof, blockchain is theoretically resistance against the Sybil attack which is also known as the 51% attack. In other words, perpetrators have to hold the control over the majority of the peers participating in the network to manipulate the transactions accordingly. Fig 2 shows the fundamental idea of how the blocks are linked together, although there are different tactics to concatenate blocks together to create a blockchain.

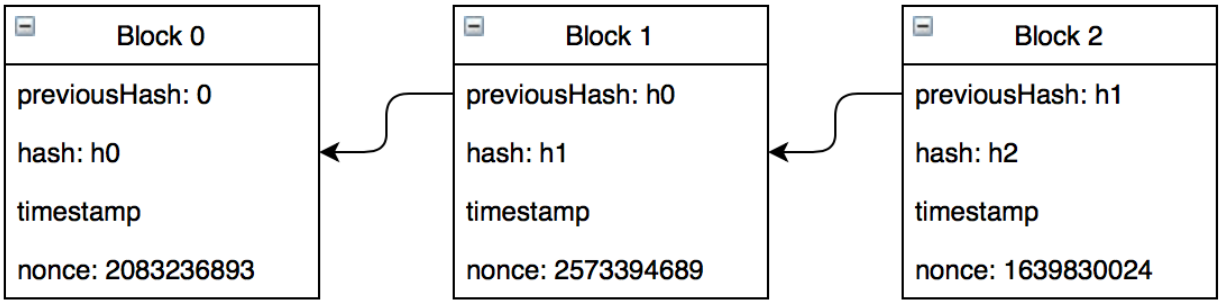


Fig. 2. Blockchain

3) How does blockchain work?

In a peer-to-peer blockchain network, a peer broadcasts transactions to all other peers. Each peer will collect all of the transactions submitted by the clients and validate these

transactions based on some defined criteria. Then, the validated transactions are added to the block currently under construction, which is a snapshot of the blockchain state and will eventually be appended to the blockchain. When it is time to add a new block to the blockchain, each peer must compute a hash value which is dependent on the transactions in the block as well as the hash value of the previous block. A blockchain network has to deal with contradictions as many peers can create multiple valid blocks at the same time. Therefore, all peers in a network have to agree on a defined consensus in the network protocol. For example, a public blockchain such as bitcoin or ethereum encourages peers (miners) race each other to find a hash value of a block. When a new valid block is created (or mined) a peer will receive the incentive in crypto currency and this new created block is broadcast to all other peers to append to the blockchain. Consequently, peers verify a new created block and append into the blockchain. We translate the block creating process into a 2-phases sequence diagram in Fig 3. In phase 1, a client sends a transaction to all of the peers, then peers validate the transaction before adding it to a block. In phase 2, each peer creates a new block, bundle all the valid transactions into the newly created block, race to compute a valid hash value for a block, and finally broadcast the block to all other peers. When a peer receives a new block, it validates the block and add it to the blockchain if that block is valid.

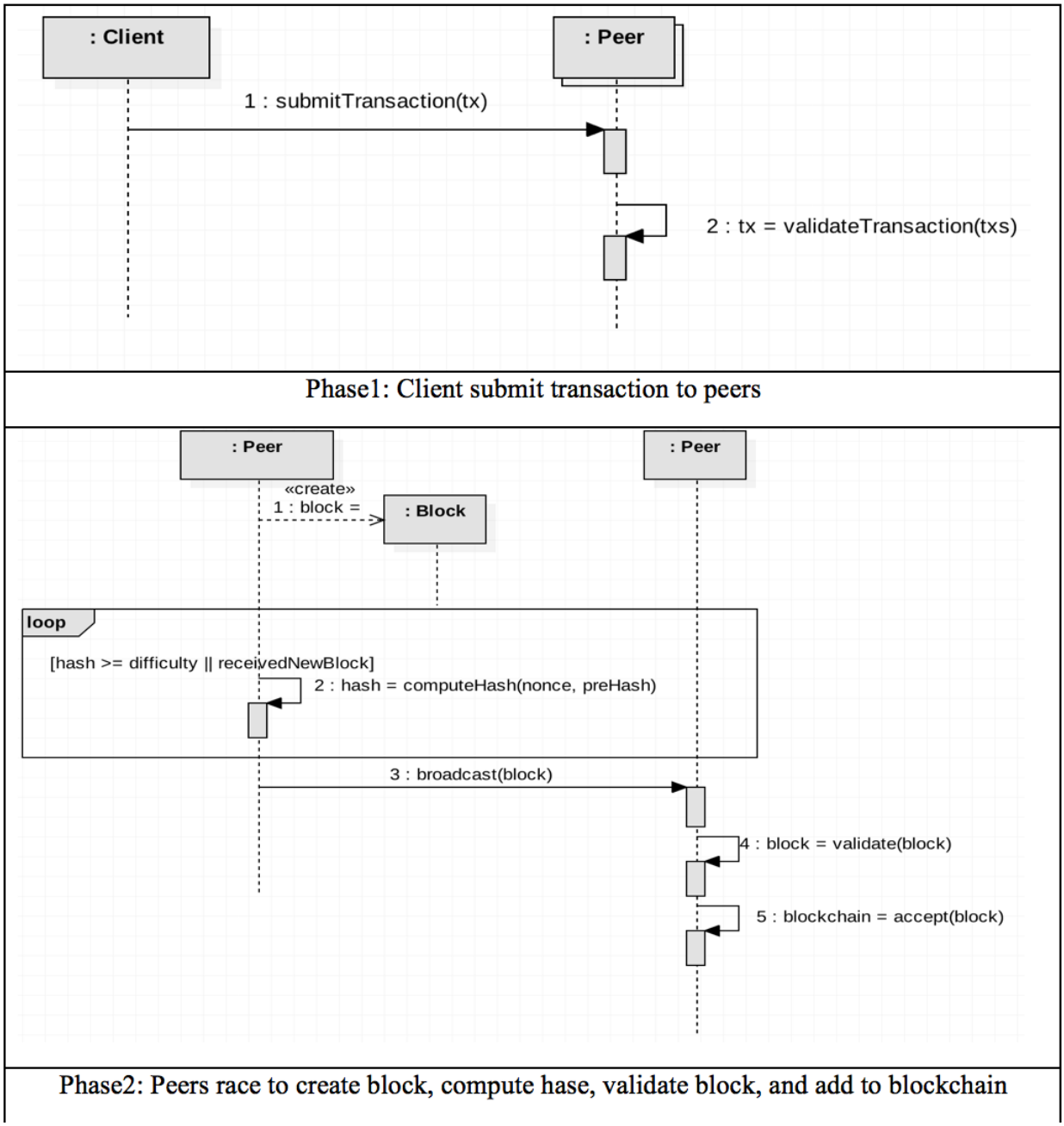


Fig. 3. Blockchain sequence diagram.

Public blockchain or private blockchain utilizes different consensus techniques to solve the conflict in the network. Public and private blockchain will be discussed in the coming sections, but consensus will be discussed first.

4) Consensus

If we look at blockchain file as a database, transactions must be validated before committing to the database, and because blockchain is a write once and read many (WORM) database it is illegal to update the previously written data. Trust is a key element of a public blockchain network because any entity can submit transactions into blockchain and it is uncertain that all of the entities are benign. So, protocol or consensus algorithms in a public blockchain must be designed to prevent dishonest entities from harming the network. Likewise, a private blockchain network also needs mechanisms to protect the blockchain database in case a peer in the network becomes a malicious node due to an attack. There are many consensus algorithms implemented in public and private blockchains. Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerant (PBFT) are the most popular ones [9].

a. Proof of Work

In 1993, Dwork and Naor [10] proposed a solution which combines the pricing function and hashing function to punish computers sending junk mail because it requires a computer using its CPU computational power to calculate hash value of the email. Therefore, Dwork and Naor's proposal can make generating and sending junk mails expensive. In 1997, Adam Back invented Hashcash algorithm [4] to calculate the computational work of creating an email. This computational work is recorded as a stamp attached in the email. In other words, Hashcash calculates and records a considerable value of computational power of email composing, so Hashcash deters spammers from sending enormous amounts of email because generating spam email becomes costly. In turn, receivers can verify the hashcash stamp attached in the email efficiently. Furthermore, a hashcash stamp reduces the number of lost emails because a content

based anti-spam system will put hashcash users into a white-list even though the email contains some suspicious words that can be put into the black-list.

In 1999, Jakobsson and Juels [11] formulated the notion of Proof of Work (PoW) which allows a verifier can check the amount of computational work of a prover. For example, computer A performs a task T in a certain amount of time. This computational work can be recorded then computer B can look at this record and evaluate the value of computational work that computer A did on task T. As a result, hashcash algorithm and PoW concept is creatively utilized in the Bitcoin network. In the Bitcoin network, computers compete with each other to create a new block by solving SHA256 function that returns a hash and a nonce value. This hash value of a new block must reference to the previous block, while nonce value must be less than a certain number which is known as difficulty that increases every 2016 blocks [12]. This race is called a mining process and all the computers participate in the mining process are called miners. The first miner that finds a new block is rewarded by bitcoins whose value is considered as proof-of-work.

b. Proof of Stake (PoS)

While PoW requires computational power to create a new block in crypto-currency blockchain, the idea of PoS algorithm is to determine a block creator by validating the stake of that user. PoS tries to convince users in the network that if a person invests enough of their asset(stake) to join the network, that person will not act as a malicious node in the network to protect their asset and reputation. For example, Alice, Bob, Charlie, David and Elly participate in the mining process and own 30, 15, 20, 35, 40 coins respectively. Assuming PoS in this scenario will randomly pick a user, who owns at least 30 coins. Hence, PoS algorithm can be implemented to randomly pick either Alice, David or Elly because each of them owns at least 30

coins (stake) in the network. Suppose Alice is picked, she will collect all transactions and put them into a new block, then broadcasts this new block to all other peers. There are many different PoS implementations, but considering the stake of users to grant the authority is the main idea. In the real-world cryptocurrency market, Peercoin and NXT [13] are two examples utilizing PoS algorithm.

c. Practical Byzantine Fault Tolerant (PBFT)

In Byzantine General Problem (BGP), the army has many divisions and each division has a general. The generals not only communicate to each other, but also to their lieutenants within their divisions. The BGP requires all the generals or lieutenants have to agree upon one of two plans of action which is either attack or retreat with a full strength. It is possible some of the generals and lieutenants may not be trustworthy because they're spies or traitors. The goal of solving this problem is to prevent the minority of traitors affect the loyal commanding general execute the plan. This classical problem is used to solve the fault tolerance problem in a traditional computer system as well as in a distributed computer network.

In July 1982, Lamport, Shostak and Pease [9] introduced the Byzantine Fault Tolerant (BFT) model which simulates the BGP to deal with fault tolerance in computer systems. In general, the BFT model requires (2f + 1) peers to overcome f number of failed peers in the system. The BFT model was an impractical model until 1999 when Miguel Castro and Barbara Liskov [14] implemented a real BFT model, named PBFT. The communicational protocol between peers in PBFT is separated into three phases: pre-prepare, prepare, and commit. Fig. 4 shows a normal case operation of PBFT protocol in which client C sends a request to node 0 which includes a failed node 3.

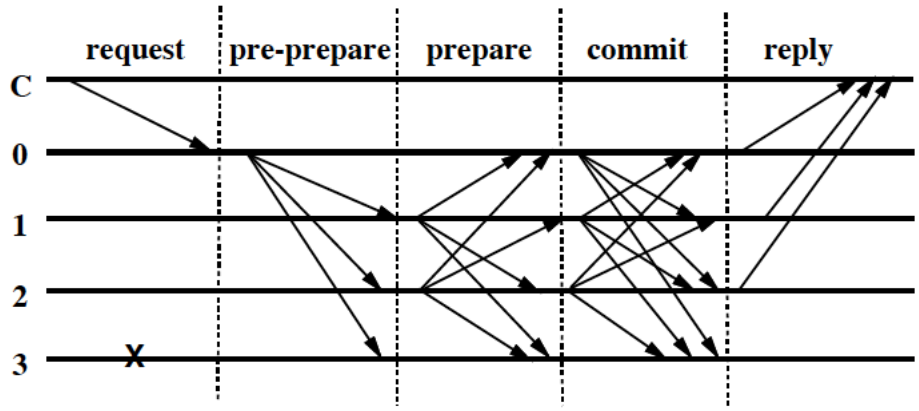


Fig. 4. PBFT operation [14]

The protocol functions as the following steps:

1. A client sends a request m to a primary peer p in a cluster (a group of peers in a network) which contains (3f +1) replicas, with f is the number of faulty replicas.
2. p creates, signs, and multicasts a pre-prepare message in the form of <<PRE-PREPARE, v, n, d>, σ_p, m> to other replicas. The pre-prepare message includes:
    - v: view of the message being sent
    - n: sequence number of the message
    - d: digest or hash value of m
    - σ_p is signature of p

Replicas can verify the pre-prepare message by public key of primary p, and compare hash value of m with digest d

3. If replica *i* accepts pre-prepare message, it multicasts prepare message in the form of <PREPARE, v, n, d, i>σ_i to other replicas and add both <<PRE-PREPARE, v, n, d>σ_p, m> and <PREPARE, v, n, d, i>σ_i to its log. A function

prepare(m, v, n, i) is define and return true if and only if replica *i* has inserted all of the following information into its log:

- the request m
- a pre-prepare for m in view v with sequence number n
- 2f prepares from different replicas that match the pre-prepare (same v, n, and d)

Function prepare (m, v, n, i) guarantees that a majority (f + 1) of non-faulty replicas agree with the pre-prepare and prepare message, so the replica can continue to step 4.

4. Replica *i* multicasts a commit message c = <COMMIT, v, n, D(m), i>σ_i to other replicas and checks the return value of function committed-local (m, v, n, i) which returns true when all of the following conditions are met:
    - Prepare(m, v, n, i) is true
    - Replica *i* has accepted (2f + 1) commit message c.

C. Types of Blockchains

Bitcoin is the first well-known blockchain network which implements the multiple necessary security aspects of distributed ledger technology. These include: double-spending, integrity check (to guarantee an occurred transaction cannot be tampered), and central point of failure. These features completely fulfill the requirements of the digital money problems that computer scientists have tried to solve for decades. Blockchain is not only viewed as a distributed ledger to record digital currency transactions, but also as a distributed database system. Consequently, a blockchain solution can be expanded to other domains such as: supply chain, health care, digital asset management, voting, etc. Over the years, blockchain solutions have been divided into two

groups: permission-less and permissioned. From another perspective, blockchains are categorized as: public, private, or consortium. Or simply put, a public blockchain is a permission-less blockchain, while private and consortium blockchain are permissioned blockchain.

1) Public blockchain

As the name indicates, a public blockchain allows everyone to view and submit transactions into the ledger without permissions. A person can also join the network as a peer to validate transactions, create blocks and maintain a blockchain. A public blockchain is a totally decentralized network because there is no single organization or entity that controls the network. Anyone can download software to participate in a consensus process. This is known as mining. For example, information of blocks and transactions in the Bitcoin network are publicly accessible on *blockexplore.com*.

2) Private blockchain

In a traditional database system, the centralized administration manages the read and write operations. Similarly, a private blockchain consolidates the full-access right to a central point. Even though read permission is shared or granted to a restricted group of users, write permission is controlled by the central administration role. For example, considering the use case of a blockchain solution for the voting system. People who have voting rights can cast their votes and submit the transactions into the network, but only a central node can validate the submitted transactions and create new blocks (write), while everybody can view the voting results on the blockchain database (read).

3) Consortium blockchain

In a consortium blockchain network, a pre-selected set of nodes take the responsibility of the consensus process to validate transactions and add new blocks into the blockchain. The right to access the blockchain data can either be public or restricted to a group of users. A consortium blockchain does not grant the write permission to a single central node, but the consensus process is distributed to multiple chosen nodes in the network. For example, Gradubique, described in chapter IV of this paper, is a consortium blockchain solution for academic transcripts. Universities participating in Gradubique are considered administration nodes in the network. When San Jose State University submits Alice's official transcript into the network, all of the universities in the United States will validate the transcript before adding the transcript into the blockchain. Alice's transcript is now officially published in Gradubique, so if Alice wants to attend a higher education program at Tokyo University in Japan, Tokyo University can evaluate her transcript directly without consulting a third party to evaluate it although USA and Japan may have a slightly different grading system.

EXAMPLES OF BLOCKCHAINS

A. Bitcoin

In 2008, a pseudonym called Satoshi Nakamoto, [6] wrote a white paper to introduce a peer-to-peer electronic cash system which is also known as the first application of blockchain technology called Bitcoin. In a country, the national central bank controls everything in the traditional currency system. For example, in the US, the Federal Reserve Bank controls the money supply, and the Treasury Department is responsible for paper money printing and coin minting. For the digital use cases of monetary transfer, all of the transactions must go through Fedwire [15], an online fund transfer system operated by the Federal Reserve Bank. Conversely,

Bitcoin is fully decentralized and has been proven not only as secure and stable, but also revolutionized the digital currency. Each node holds a copy of a blockchain file or a ledger in the Bitcoin network to keep track of all the transactions, and a ledger is considered valid if it is consistent with the majority's ledger. This massive redundant tactic guarantees that dishonest nodes or attackers cannot untruthfully tamper the blockchain file because they have to change 51% of the blockchain file in bitcoin network at the same time. In addition, bitcoin leverages the distinguished features of cryptographic research of Merkle tree, hash functions, public key cryptography, and digital signature which have evolved in decades to ensure that altering a single bit in a confirmed block is impossible. However, bitcoin miners have to vote for any enhancement feature in bitcoin network before it is implemented because the disagreement in an enhancement feature may cause discrepancy of software being used by miners. Hence, the inconsistency will create a fork (or branch) in blockchain file among the nodes. To get a better understanding of the security features in Bitcoin architecture, we explore a little detail of the following elements: wallet, transaction, block, and mining. Then, we will explore forking scenarios in bitcoin.

1) Wallet

A bitcoin wallet is a pair of public key and private key that allow user to invoke transactions in the Bitcoin network. An analogy to a bitcoin wallet is an email address. A public key or wallet address is comparable to an email address, while a private key is considered an email password. For example, Alice can only send an email to Bob's email address if Alice knows (remembers) her email password. Similarly, Alice can invoke a transaction to send bitcoin from her wallet to Bob's wallet with her private key. If Alice loses her private key, there is no way for Alice to spend bitcoin in her wallet. As the name indicates, a public key is known by

anyone, so people can view all the transactions and check the balance of a particular bitcoin wallet, but nobody can execute a transaction without an associated private key.

2) Transaction

A transaction in the Bitcoin may contain one input and multiple outputs which are used to calculate and validate an acceptable transaction. Input/output presents the quantity of bitcoin unit in a transaction, and satoshi is the smallest unit of bitcoin, 1 BTC = 1,000,000 satoshi [12]. In addition, a transaction also includes an input and output script which can be used as a smart contract embedded in a transaction. This smart contract acts like a pre-defined rule that need to be evaluated when a transaction is validated [16]. For example, a user can use a script to indicate that a transaction is only executed when there are at least 3 signatures to confirm the transaction. Transactions are concatenated together in a way that the output of the previous transaction is the input of the next transaction. Although a transaction can create multiple outputs in case of payment to multiple addresses, each output of a certain transaction can only be the unique input of other transaction. As a result, an input cannot be spent more than one time – this is known as the double spending problem. For example, Fig. 5 shows the connections between transactions in the Bitcoin. Transaction 0 (TX0) has 100k satoshis as the input. Thereafter, we are going to spend all 100k which are separated into 3 parts: 40k on TX1, 50k on TX2, and 10k fee for both transactions which is collected by a miner. Later, TX1 is spent 30k on TX3, 10k for transaction fee in total of 40k. At this point, the maximum balance of unspent value if we make another transaction from TX3 is 20k because 10k will be allocated for the transaction fee. The 20k output from TX3 and 10k from TX6 is defined as unspent transaction output (UTXO) which is used to calculate the balance of a wallet (discuss in the next section).

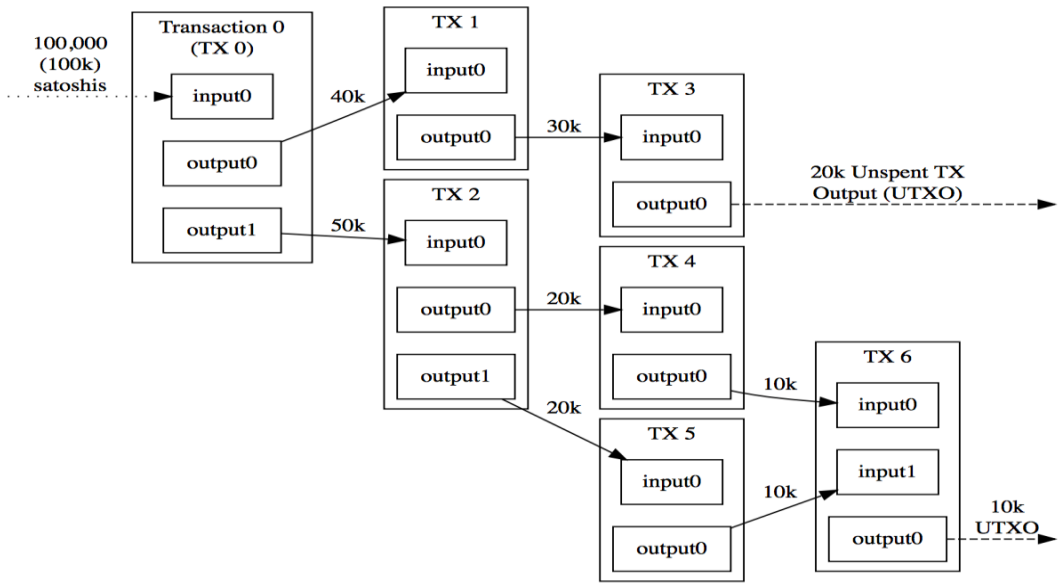


Fig.5.Transaction to transaction payment [17] [18]

Coinbase transaction is the first transaction in a block which is created by bitcoin miner. This transaction has no output link to it because it is an incentive compensation for the miner who first computes the valid hash value of a new block. Bitcoin protocol suspends the coinbase transaction from being spent for at least 100 blocks to prevent a miner from expending the block reward from a block that may later be determined as stale after blockchain fork. [17]

3) Block

Bitcoin transactions are tied to a certain block using Merkle tree data structure [6]. As an illustration, we examine the process to create the Merkle root of block 11 in Fig. 6 below. First, Tx0, Tx1, Tx2, Tx3 are hashed to 4 unique values Hash0, Hash1, Hash2, Hash3 respectively. Second, each adjacent pair of hash value are combined and hashed together to create an upper level of hash value. In this case, Hash0 and Hash1 are combined then create a new hash Hash01; Hash2 and Hash3 is used to generate Hash23. The process is recursively repeated until a unique Merkle root is found. In any case, if a single bit of a transaction has changed, the merkle root value is changed. Subsequently, transactions are tamper proof in a block as long as a block is

confirmed and appended to blockchain. Moreover, blocks in bitcoin are permanently tied together via the hash value of the previous block because a hash value of a block 11 is determined by its Merkle root, its nonce, and block 10 hash value. The process to calculate hash value of a block is called a mining process which is elaborated in the next section.

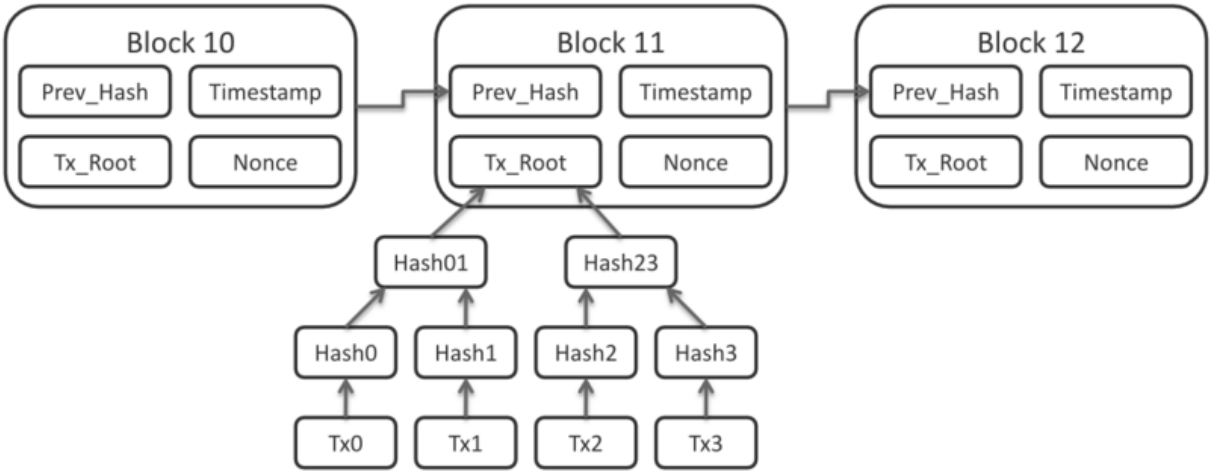


Fig.6. Bitcoin block data

4) Mining

Mining in bitcoin network is a financial incentive process in which nodes race each other to create a new valid block that is added to the blockchain. In return, a node is rewarded with new coins as it generates a confirmed valid block and is paid a transaction fee collecting from transaction validation. A block mining reward is compensated in the coinbase transaction which is mentioned in the transaction section above. Approximately, a new block is created every 10 minutes in the bitcoin network, and the rate of mining compensation is reduced 50% every 210,000 blocks or roughly every 4 years [12]. In 2008, when the bitcoin has first introduced, the block reward was 50 bitcoins; then the number was cut down to 25 bitcoins in 2012. Currently, miners receive 12.5 bitcoins to mine a new block. The whole mining process is illustrated in the following steps:

i. A node compiles a definite number of transactions broadcast in the network, then verifies the signature and validates output in each transaction.

ii. Then a node calculates the Merkle root of all transactions, combines the Merkle root with previous block's hash and tries to find the random number—called nonce. This nonce is valid if it can be combined with the Merkle root and the previous block's hash to create a new hash which is less than a certain number. This number is defined as a difficulty of the network. The found hash value is considered a proof-of-work which is a certificate to prove that a node has worked hard to solve the puzzle. The following equation summarizes the PoW in the Bitcoin:

$$H(N, P\_Hash, Merkle\_Root) < Difficulty$$

Where N is nonce, P_Hash is the previous block's hash, Merkle_Root: is the hash value of transactions following the merkle tree data structure.

iii. A node proposes its new block by broadcasting this block to the network.

iv. When a node receives a first proposed block from the network, it evaluates the block by verifying the transactions within that block and the hash value of the block. If the block is valid, the node will add this block into its blockchain and continue the race to find a new block.

5) Forking

In case of 2 or more miners that create new valid blocks at the same time, other nodes will pick the block that they first receive and proceed to mine the next block. Therefore, the bitcoin network might exist at least 2 difference versions of a blockchain — this is called forking. Ultimately, at a certain time, one blockchain is higher than the others, and the bitcoin

protocol forces the nodes to follow the highest chain, so all the nodes in the network have to synchronize its local blockchain file with the highest chain holding by majority of nodes. Even though rejecting the shorter chain in the bitcoin network means negating the proof-of-work of miners which consumes hash power and electricity to perform, this protocol ensures that the minority of malicious nodes are deterred from attacking the network.

Another case of forking is when the network changes the consensus rule in the network which forms 2 types of forking: soft-fork and hard-fork

i. Soft-fork happens when the bitcoin software is upgraded to a new version which implements a new consensus rule to validate a new block while it is still backward compatible to the old rule. For example, in Fig. 7, non-upgraded nodes will reject a proposed block which follows the new rule, while upgraded nodes accept block created from both old rule and new rule. If majority of the nodes upgrade to newer version, the non-upgrade nodes will recognize that they are no longer be able to validate a block proposed by upgraded nodes. Eventually, the non-upgrade nodes have to upgrade to the newer software, so they can validate proposed blocks from upgraded nodes.

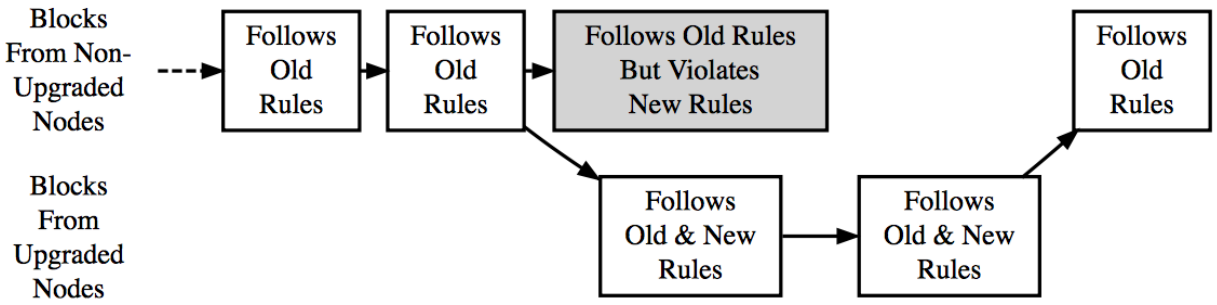


Fig.7.Soft-fork: Upgraded node accept both old and new rule [17]

ii. In hard-fork scenario: upgraded nodes reject the block from non-upgraded nodes which stick to the old rule. Whereas, non-upgraded nodes reject the block

proposed by the upgraded nodes which follow the new rule. This disagreement separates the nodes into 2 different networks with 2 different blockchain protocols. For example, a group of developers forked bitcoin to a new crypto currency – called Bitcash – in August, 2017 [19]. They upgraded the block size to 8 MB compared to 1 MB of Bitcoin and claimed that the Bitcash block has the ability to hold more transactions than bitcoin. As a result, transaction fee is reduced in the Bitcash network. Fig. 8 illustrates the hard-fork in a blockchain network.

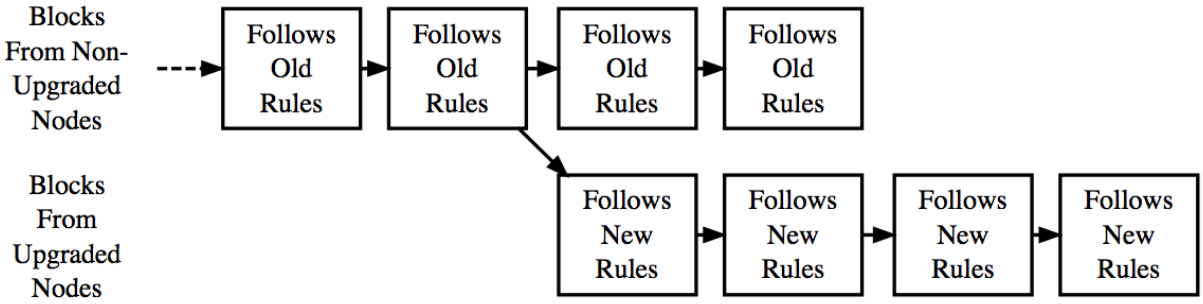


Fig.8.Hard-fork: Non-upgraded nodes reject the new rule [17]

B. Ethereum

Although Bitcoin transactions are embedded scripts to support a smart contract, the capabilities of the script to build a functional smart contract is very limited. In fact, a script in Bitcoin is not Turing-complete, so it does not support loops. Furthermore, UTXO in bitcoin can only be either spent or unspent; thus, it is difficult to code a contract that requires multiple stages. For example, company A and B sign a 10k-satoshis service contract which contains 3 phases. The payment will be split into 3 parts as follows: B pays A 2k as phase 1 is completed, additional 4k when phase 2 is completed, and the rest when phase 3 is completed. With the limitation of script in bitcoin, building a such described contract in a single transaction is

infeasible. In 2013, Vitalik Buterin and his team [20] proposed Ethereum as a solution for a brand-new blockchain platform to develop decentralized applications. The team envisaged Ethereum to be not only a platform to develop smart contracts to automatize the payment process for business, but also a "world machine" or environment to execute smart contracts. Obviously, Ethereum is a platform for many other blockchain solutions run on top of it today. Ethereum blockchain is generally interpreted as a world state machine because a certain ethereum block indicates the current state of the network. As a new block is created, it records a new state by bundling all of the transactions has occurred in a period of time (15 seconds) into this new block. State transition in ethereum network is illustrated in Fig. 9 below.

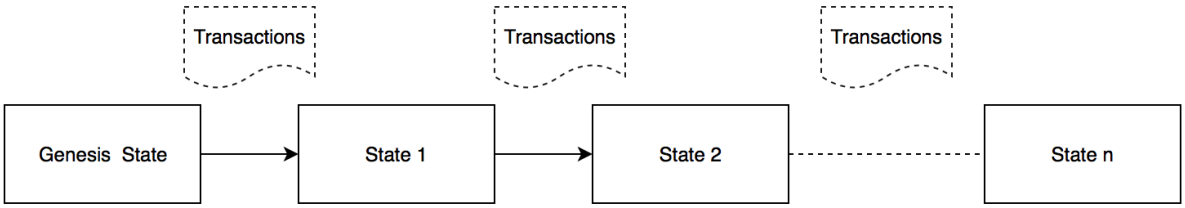


Fig. 9. Ethereum State Transition.

Here we explore the core components of Ethereum to understand how this blockchain records its state within a block.

1) Ethereum Virtual Machine (EVM)

EVM is a sandbox and a run-time environment for smart contracts, which are installed on nodes (miners) of the Ethereum network [21]. Because EVM is isolated from the network, file system or other processes of the host machine [22], it is impossible for hackers to take the control of EVM to manipulate smart contracts. Solidity is a programing language used to compose smart contracts which are compiled to EVM bytecode and uploaded to blockchain using an Ethereum client software.

2) Gas and Payment

In EVM, each operation has an associated cost that a requester has to pay for a miner who hosts EVM to execute the operation. In other words, a miner is paid in Ether to verify transactions, execute chain codes, and generate a new block to update a new state of the Ethereum network. Gas is the term used to define the cost of all the operations that EVM executes. With every transaction, a sender sets a gas limit and gas price to indicate the fee that the sender is willing to pay for the miner to process the transaction. Gas limit and gas price are measured in Wei (the smallest unit in ethereum), and 1 Ether equals 10^{18} Wei [21]. For example, Bob sets a gas limit at 50,000 and gas price at 20 Wei for his transaction to transfer 2 Ethers to Alice. At most, Bob's account will be deducted 2.001 ETH which include 50,000 x 20 = 0.001 ETH transaction fees and 2 ETHs transferred to Alice. However, each operation has a defined gas price in EVM, so if the defined operation fee is lower than the fee set by the user, EVM will return the unused gas to the sender.

4) Smart Contract

In reality, a contract refers to a set of legal rules or conditions that participants agree to exchange for services. For example, Bob offers to transport a package from Alice's house to Trudy, and Alice will pay Bob 1 Ether in return. Apparently, trust is the most imperative element in every contract. For example, in one case, Bob has delivered the package and Alice received the confirmation from Trudy, but Alice may not want to pay Bob. So, Bob has to believe that Alice will pay him after he has provided the service. In another case, Bob may ask Alice to pay him first before doing his job, but Alice does not know Bob well. Bob may get the money without fulfilling his responsibility. Or, Alice and Bob do not trust each other, so they need a trustworthy broker to hold their collaterals. When Bob has done his job, he can receive the

payment from this broker while this intermediary party can collect the money from Alice. In this case, both Bob and Alice have to be loyal to their commitment because neither of them wants to lose their deposit money. However, what if the broker that both Alice and Bob trust is actually a fraudulent one, so they both may lose their money.

In the Ethereum network, a smart contract is an agreement between accounts to materialize a transaction when a pre-defined set of rules in a contract was met. The term "smart contract" implies that the contract can be applied by itself to trigger a transaction without the interference from any actor. In fact, a smart contract is a software program written in Solidity programming language which is then compiled to bytecode and submitted to the ethereum blockchain. This bytecode is executed within isolated EVMs which are installed on distributed nodes of the Ethereum network. When a smart contract is submitted, no one can alter the code in the contract since it is protected by the security features of the Ethereum blockchain.

3) Account

There are 2 types of accounts in the Ethereum network: external and contract account. An external account is similar to the wallet concept in the Bitcoin network which is controlled by a user to submit transactions into the network. A contract account is a place holder of smart contracts which are executed on EVMs. While an external account is the place where humans interact with the Ethereum network, a contract account invokes the chain code (smart contract) on behalf of humans [22]. For example, Alice agrees to sell her green marble to Bob for 10 Ethers. Trudy provides delivery services and he charges Alice 5 Ethers for transporting the marble from Alice to Bob. To ensure all 3 participants trust each other in this deal, we can define a contract such that:

- Alice is required to deposit 5 Ethers (to pay for Trudy after he fulfills the delivery service) to contract account C
- Bob and Trudy deposit 10 Ethers each to the contract account C. This is the collateral to keep Bob and Trudy honest with the contract.

The sales process should be divided into 2 phases: deposit and payment. Fig. 10 illustrates the transactions invoked before and after the contract fulfillment.

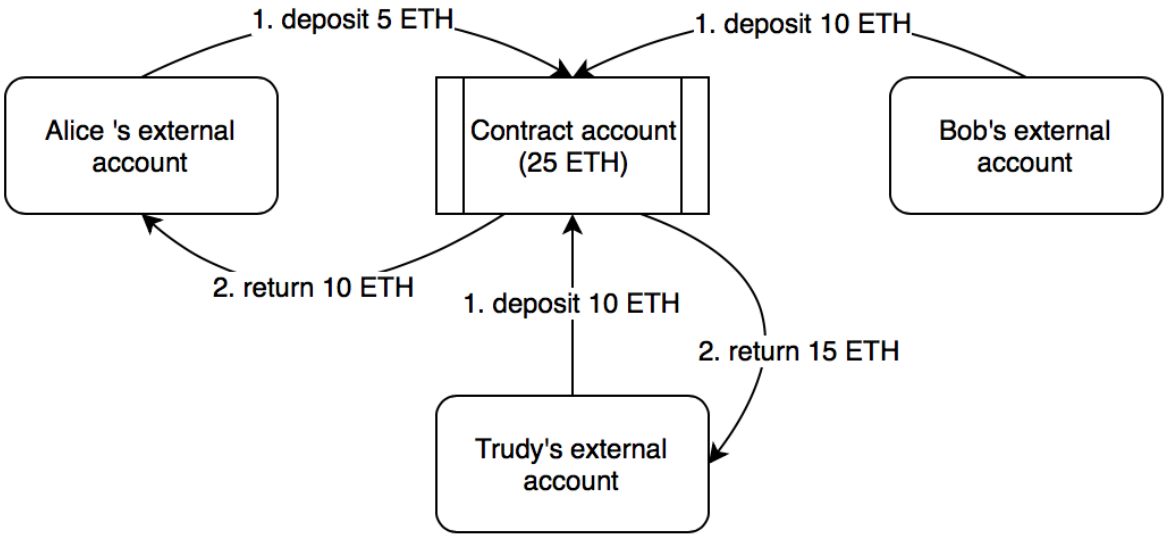


Fig. 10. External and contract account interaction.

i. Alice deposits 5 Ethers from her external account A to contract account C

   Bob deposits 10 Ethers from his external account B to contract account C

   Trudy deposits 10 ethers from his external account T to contract account C

ii. As Bob receives the marble delivered by Trudy, he signs off the contract. Contract account C will deposit 10 Ethers to Alice’s account (payment from Bob for the marble). Trudy will receive 15 Ethers (10 Ethers from his deposit and 5 Ethers payment for the delivery service)

Characteristics of an external account include:

- Balance of ether (ETH)
- Capable of sending transactions
- Control by a private key
- Has no contract code

Contract accounts has the following characteristics:

- Balance of Ether (ETH)
- Hold contract code in the memory
- Can be triggered by the message send from external account or another contract account
- When executed, can perform complex operations
- Have no owner after release to the EVM
- Have their own persistent state and can invoke other contracts

4) Transaction and Message

A transaction in the Ethereum network can either be a normal transaction which transfers crypto currency Ether (ETH) from an external account to another external account, or a transaction contains data to invoke smart contract codes in a contract account. A transaction contains:

- The recipient of the message
- A sender's signature (private key)
- A value field representing the amount of ETH being sent
- An optional data field for a message (if this transaction invokes smart contract)
- A STARTGAS value indicating the maximum GAS value paid for the transaction to be executed

- A GASPRICE value, the fee that the sender is willing to pay for one computational step.

In addition, a contract account can also invoke a smart contract from another contract account via a message. A message is a chunk of data containing instructions to guide a contract account on which function to execute. A message contains:

- The sender of the message
- The recipient of the message
- The value of ETH being sent
- An optional data field (containing input data for contract)
- A STARTGAS value to limit the computational operations

STARTGAS and GASPRICE are essential in ethereum to prevent an infinite loop and denial of service attack (DoS). For example, if a smart contract contains a snippet of code which loops endlessly, EVM will quit the code at a certain time when it consumes all of the STARTGAS value. Or if attackers want to send an enormous amount of transactions to the EVMs in the Ethereum network to overwhelm the computational capacity, attackers have to spend a large amount of ETH. Therefore, STARTGAS is not only a disincentive medium to prevent DoS, but also an incentive to engage peers to join the network because peers are paid for their computational power which is identical to the mining process in the Bitcoin network.

5) Block and Mining

A block in Ethereum is also divided into 2 parts like a Bitcoin block: a block header and block body. While a Bitcoin block header contains a unique Merkle tree root to verify transactions within a block, an Ethereum block header contains 3 different Patricia trees to reference all transactions in a block, determine the current state of the network, and identify the

proof of events occurred in a block. Before going into the detailed information of a block and how mining works in Ethereum, we need to understand Patricia tree data structure.

a) Patricia Tree

Patricia tree is a combination of Radix tree and Merkle tree. Patricia tree is added some extra complexity of data structure to optimize the space and enhance the speed of querying data. A node in Patricia tree is either one of the following:

- NULL (represent an empty string)
- branch (a 17-item node, [v0, v1, …v15, vt])
- leaf (a 2-item node [encodePath, value])
- extension (a 2-item node [encodePath, key])

Fig. 11 illustrates an example of how wallet balances are stored into the Patricia tree data structure. In this example, there are 4 wallets: a711355, a77d337, a7f9365, a77d397 which have the same prefix “a7”. This is called shared nibbles in Ethereum string format. Because all the wallets in this example have “a7” as prefix, the root node (extension node) starts with “a7” as encodePath while this root node’s key refer to a branch node. A position in a branch node refers to an extension node if there are at least 2 wallets that share a nibble at this position. Otherwise, a position in a branch node links to a leaf node. An extension node will continue referring to an extension node or a branch node until the encodePath reaches a leaf node. In this example, **a7**11355 and **a7**f9365 refer to 2 leaf nodes “1” and “f” where they can identify the final state of the key 1355 and 9365 respectively to determine the balance of their wallets. **a77d**337 and **a77d**397 share nibble “d3” after position 7, so position 7 in the branch node refers to an extension node then a branch node until they reach the leaf nodes.

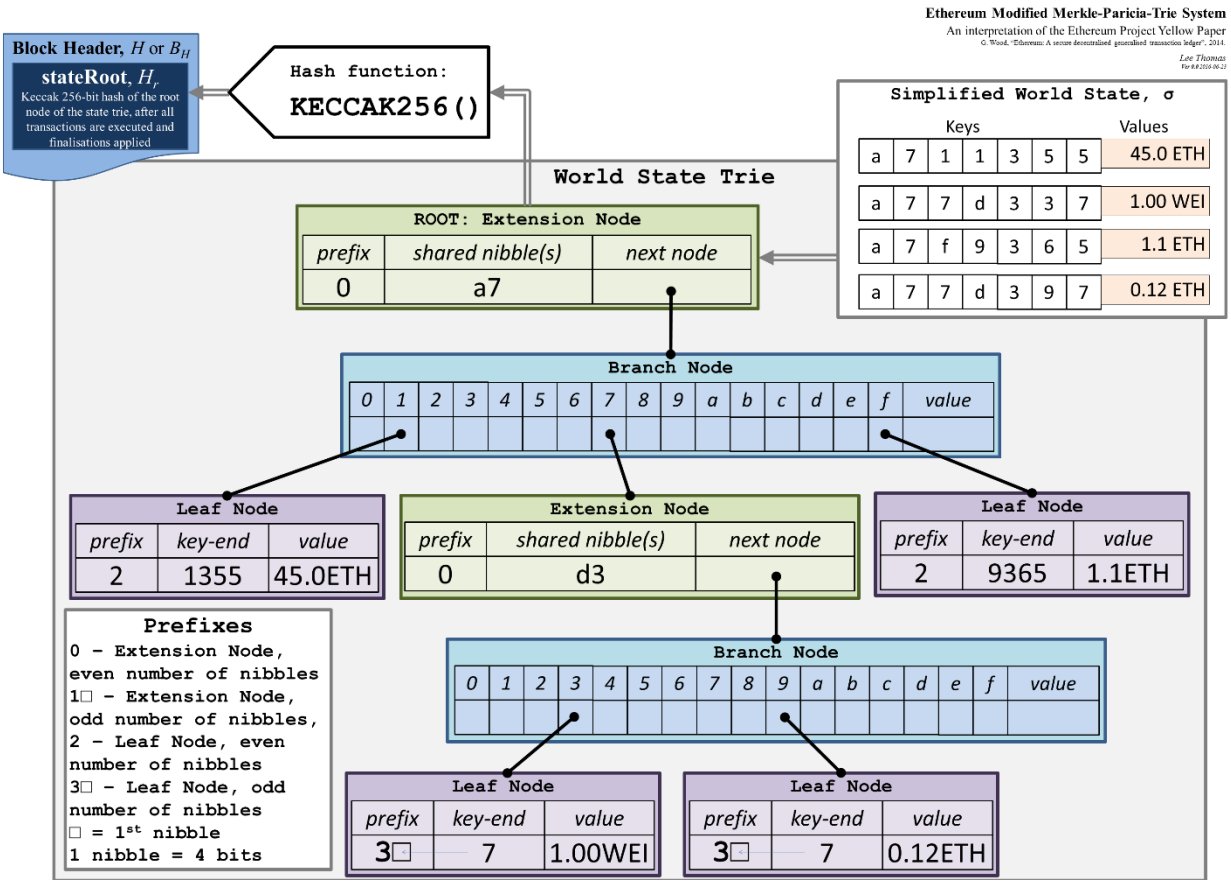


Fig. 11. State Patricia Tree [23]

b) Block

The most important elements in a Ethereum block header are 3 Patricia tree roots for 3 different trees in its block body. They are state root, transaction root, and receipt root. Each root is a hash value of an associated Patricia tree in a block body. In the given example in part a, the state tree is stored in a block body while the state root is written on a block header. So, a client can check a wallet's state (account balance) at a certain block by submitting a wallet ID and the state root of a block. Similarly, a transaction root can be used to validate if a transaction contains in a block, while a receipt root determines if an instance of event occurs in a block. Fig. 12 shows the detailed information stored in an ethereum block header.

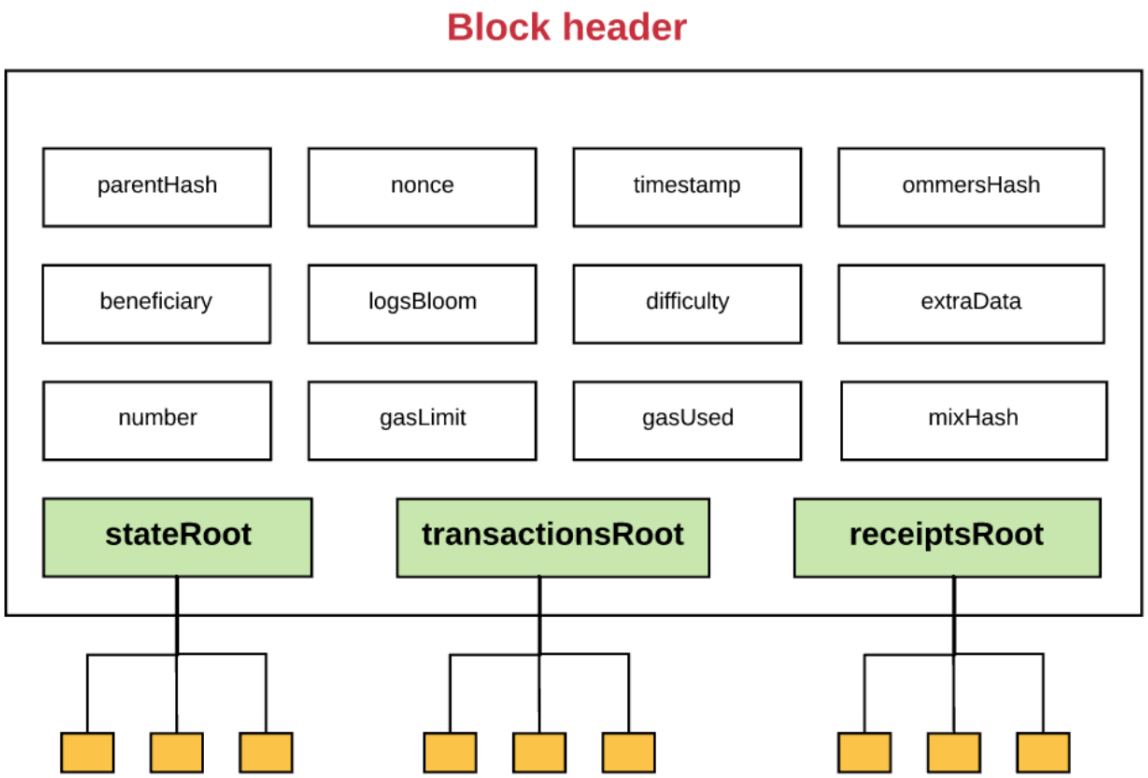


Fig. 12. Ethereum block [24]

In a blockchain network, a lightweight node does not store a full copy of a blockchain file. Instead, a light weight node only stores a chain of block headers to fit in its limited storage space. When this light weight node needs to verify if a transaction is included in a block or check an account balance, it dispatches the request to a full node. How a block is formed and finalized will be explained in the next section.

c) Mining

Currently, Ethereum is using proof-of-work to reach consensus between nodes to create a new block, but the development team plans to deploy proof-of-stake in the coming years. Hashing algorithm for proof-of-work in ethereum is Ethash which is designed to resist against application-specific integrated circuit (ASIC). This specific chip is customized to serve a particular purpose [25]. For example, Google builds tensor processing unit (TPU) to optimize the

training speed in machine learning [26]. In the circumstance of bitcoin, all of the miners today are ASIC built to swiftly solve SHA256 hashing algorithm to find a new block. Ethereum, on the other hand, does not want its mining process to be centrally controlled by chip makers. Ethash algorithm relies on directed acyclic graph (DAG) file which stores the latest epoch a blockchain. An epoch contains 30,000 blocks which are approximately 2GB in the Ethereum network [27]. Fig. 13 is the sequence diagram of a block mining process in ethereum.

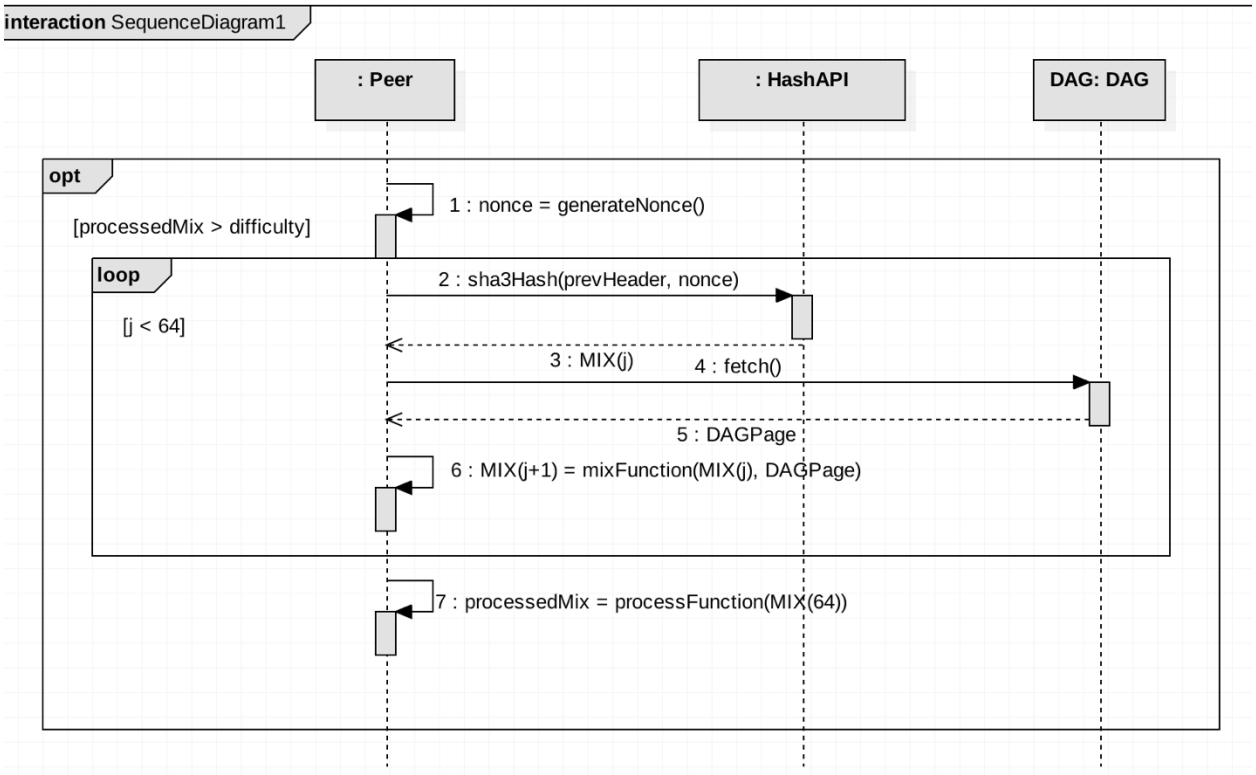


Fig. 13. Mining a block in Ethereum

First, a miner (peer) generates a random nonce, then hash this nonce with the previous block header. Second, the result hash value is put through a mix function to create a 128B MIX(j) value. Then, this MIX(j) value is combined with a random page of DAG file to create MIX(j+1). This process is repeated 64 times until MIX(64) is processed again to truncate to a 32B mix. If this result is less than or equal to the difficulty, the nonce is valid. Finally, a miner broadcasts the

block included with this nonce to the network to prove its completed successful works. Fetching the DAG pages from a DAG file is the key step to resist ASIC mining because this step requires a lot of memory which is expensive for chip makers to build an integrated chip [28].

C. HyperLedger Fabric (HLF)

While public blockchain networks such as Bitcoin and Ethereum allow users to collaborate anonymously and trust each other on transactions based on proof-of-work protocol, Hyperledger Fabric is a framework to build a private blockchain solution for trusted identity members. In other words, Bitcoin and Ethereum are permission-less blockchains which invite everyone participating into the network without a disclosed identity, whereas HLF is a permissioned blockchain that issues certificates for participants to identify invited users.

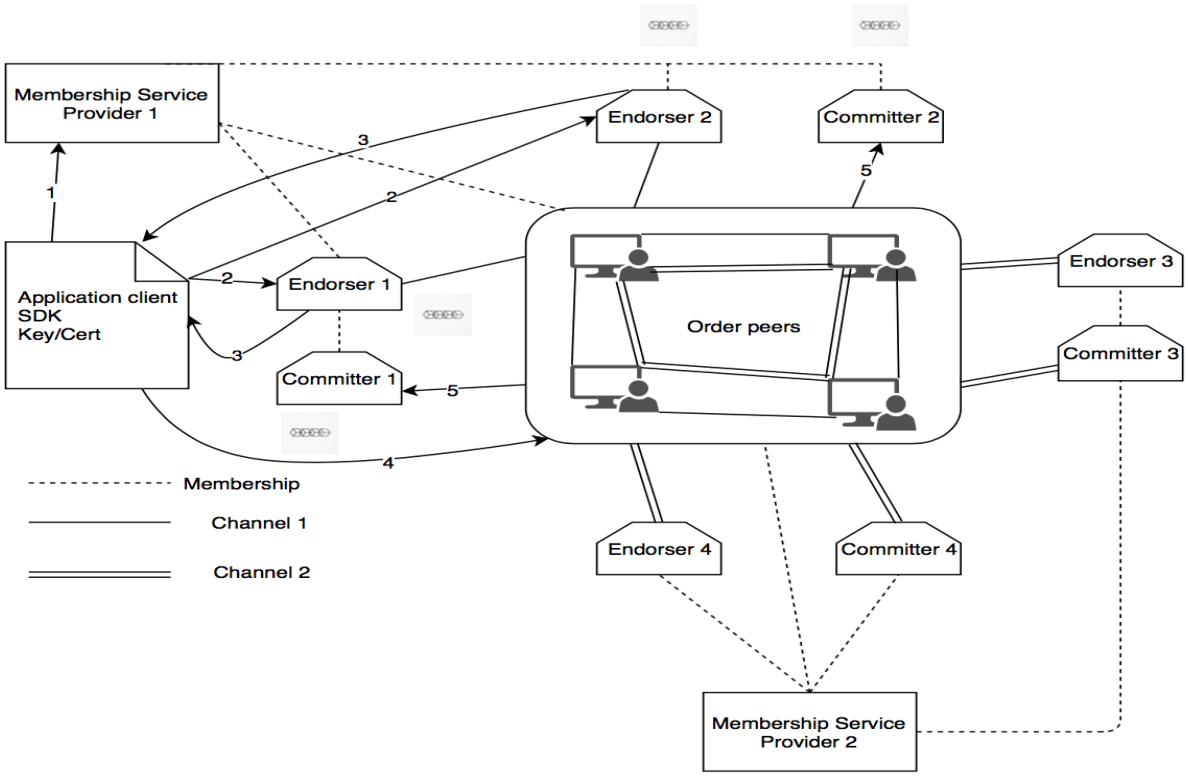
HLF framework is part of Hyperledger project hosted by Linux Foundation whose goal is to advance cross-industry blockchain technologies. HFL possesses some enhanced characteristics to speed up blockchain adoption in the business environments. First, a smart contract in HLF can be coded with the most popular programing languages: Java, Python, Golang, and node.js. So, building blockchain solution on HLF can immediately utilize the skill sets of the majority of developers. Second, the consensus protocol in HLF is designed to plug-in multiple approaches depending on business requirements instead of proof of work [29]. Third, although HLF literally does not require establishing a cryptocurrency to deploy a network, developers still can leverage the blockchain concept to build a cryptocurrency system with HLF framework for their business requirements.

Hyperledger Fabric contains the following core components: Membership Service Provider (MSP), Chaincode (smart contract), Peers (endorsing peers, committing peers, and ordered

nodes), Channels, Shared Ledger, and Gossip Network Protocol. Fig. 14 shows the high-level 5 steps of a transaction flow between components in the HFL framework.

1. A user enrolls as a member via MSP
2. A user submits a proposal transaction to endorsing peers.
3. Endorsing peers execute the chain code, sign (endorse) and return the transaction to the client.
4. A client then submits the endorsed transaction to the ordering service
5. The ordering service collects, verifies, and adds the endorsed transactions to a new block by channel. Then, the ordering service returns the created block to a leader peer associated with a channel. Finally, peers validate transactions in the returned block and append this block to the blockchain.

Fig. 14. High level transaction flow between components in HLF.



1) Membership Service Provider (MSP)

MSP provides the identity management service that not only let users enroll as members of the network, but also constitutes rules and access control list for authorization management and access privilege control. MSP in HLF has a built-in public key infrastructure (PKI) to publish public/private key pair to members in the network [30]. All of the cryptographic mechanisms and certificate authority (CA) services are provided to issue, validate, certify, and verify digital signatures. Each member (user, peer, ordering service, client) needs a verifiable identity (digital certificate) to participate in HLF network. MSP is a component to define who are the reliable certificate authorities in HLF network, and also verifies which certificate is valid or invalid in HLF protocol. Three ways we can create the certificate for the MSP configuration:

- Using OpenSSL, generate X.509 standard certificate
- Hyperledger CA issues certificates
- Cryptogen tool, a mini built-in tool to generate certificates for network simulation and development.

2) Chain code

A Chain code or smart contract is a program, written in Go, Java, Python, or node.js, to handles business logic agreed by members of the network. In other words, a chain code is written by developers to ensure a proposal meets all app-specific constraints before it's added to the block chain. Chain codes can be invoked to update or query the ledger in a proposal transaction. Chain code A also be able to invoke chain code B within the same network if A has appropriate permissions [31].

3) Channels

In the Hyperledger network, an administrator can maintain multiple channels to separate exchanged data between sub-network [32]. By comparison, a channel in HLF is a group in a messenger software in which a member can see the message in the group he/she belongs to. In addition, HLF allows a member to join multiple channels, but the messages cannot be shared between the channels. For example, A participates in both channel C1 and channel C2, while B is a member of channel C1, and C is a member of channel C2. So, A and B can exchange a message with each other, for example message “ab”, but A cannot share the message “ab” to C because C is not a member of channel 1. Each channel is associated with a blockchain or ledger (more detail later) which works as a database to record the transactions created by chain codes.

4) Ledger

Ledger is the term used in HLF to represent a blockchain file for a channel. Ledger is the place to maintain an append-only, sequenced, and tamper-proof of records as well as the logs of successful or unsuccessful transactions [33]. In other words, a ledger is a log book to document all the transitional states of a blockchain in the HLF. HLF also manages a database system (LevelDB or CouchDB) to maintain the current state of the whole network. This can be thought of as the world state database in the HLF because it stores the current state of all channels [33]. Each transaction affects a key-value pair record in the ledger regardless of the action to either create, update, or delete. For example, application A submits a legitimate transaction, and the peers consent to add a key/value pair K1-V1 to the ledger. At the same time, HLF also inserts a transaction K1-V1 into the world state database. When A requests to update key K1 with value V12, the ledger will log a new transaction K1-V12 while the world state database updates K1- with V12. The main purpose of the world state database in HLF is performance optimization for

query requests. For example, if a client wants to check the current balance of a bank account on HLF blockchain, chaincode designed for this application just need to query the current balance of the account in the world state database instead of traversing all over blockchain file and calculating the balance from transaction logs. Even though a world state database is not tamper-proof, it is backed up by blockchain files in a channel. If the world state database is corrupted, it can be restored from the ledger files of a channel.

5) Peers

Peers are elemental components of the Hyperledger network because they are the places to host ledgers and smart contracts. A peer must be authenticated by MSP before it can join the network, and a peer can be configured to be either an endorsing, ordering, or committing peer. A transaction flow in HLF can be separated by three phases [34], and each phase associates with the responsibility of the type of a peer.

a. Endorsing peer(Endorser)

At phase 1, client A sends a proposal to endorsing peers who host the chain codes and ledgers. As mentioned in section 2, a chain code is a smart contract designed to govern the business rule of an app-specific blockchain network. A legitimate transaction in HLF is generated by endorsing peers and this transaction must garner enough endorsement among endorsing peers. An endorsement in HLF means an endorsing peer independently executes the chain code as required from the proposal to generate a response, then signs on this response by its digital signature to endorse the proposal. Because a response is created from a smart contract which agreed and signed by peers in the network, a transaction cannot be tampered or fraud by a malicious client. Endorsing peer then sends the proposal, signed response, and read-write set to client which are needed for the validation step in phase 3. A client collects all the endorsed

responses from endorsing peers, wraps these endorsed responses into a transaction message, and send this message to ordering service. However, a transaction message must satisfy an endorsing policy which indicates how many endorsements a proposal need to create a valid transaction. For example, transaction T1 is considered a legitimate transaction if 2 out of 3 peers P1, P2, P3 endorse T1. So, when client A proposes a transaction, peer P1, P2, P3 execute the installed chain code individually to generate transaction T1. If the proposal is legitimate, these peers will sign the proposal to endorse T1. When application A collects enough 2 out of 3 endorsements, it is ready for phase 2.

b. Ordering peer (orderer)

Ordering service is a term to describe a cluster of ordering peers in the HFL. At the second phase, client A submits a transaction message, a wrap of the endorsed responses in phase 1, to the ordering service which will collect all the transaction message from the different channels. Next, the ordering service sorts transactions in chronological order, and bundles transactions into blocks. Each block is a list of transactions by channel. The ordering service then sends a block to a leader peer of each channel to start phase 3. The leader peer will propagate the block to other peers in its organization using the gossip protocol which is described in the next section.

c. Committing peer

While an endorsing peer hosts a chaincode and a ledger, a committing peer only hosts a ledger. In other words, an endorsing peer is a committing peer, but a committing peer is not necessary to be an endorsing peer. This concept is practical because an organization may only need 1 endorsing peer to generate transaction, while it has multiple committing peers to distribute the ledger to many nodes. When committing peers receive a block from the ordering

service, each peer will independently validate all transactions within that block. Committing peers will check if transactions meet the endorsement policy, and the outcomes of endorsing process is the same for each endorsing peer. If all the transactions are endorsed correctly and the endorsing outcomes are consistent, committing peers will perform a ledger consistent check to ensure the ledger state of all nodes are agreeable. Finally, committing peers update not only valid transactions into the ledger, but also invalid transactions because invalid transactions are useful for audit purposes. For instance, failed transactions may be caused by a malicious endorsing peer or the chain codes in endorsing peers are not consistent. Therefore, network administrators can look at these records to identity the trouble peers.

6) Gossip Network Protocol

Gossip protocol in HLF is designed to optimize network performance by separating the workload between the types of peers. Endorsing peers execute chain code, ordering peers package transactions into a block, and committing peers validate transactions as well as update the ledger.

a. Gossip message

An online peer P1 continuously broadcasts “alive” messages to all other peers to signify its available status. If all other peers do not receive the alive signal from P1, P1 is marked as dead and eliminated from channel membership [35]. Each message is signed by a private key, so peers can check the sender’s identity by its public key. Therefore, only legitimate peers can communicate to each other and the offline peers are removed from the contact list of the online peers. In addition, online peers can pull the block data from other local peers within an organization to reduce network latency.

b. Leader election

Peers in an organization will elect a leader committing node to communicate with the ordering service to receive a new block. Then, this leader node will propagate the received block to all other nodes in its organization. In other words, the ordering service does not broadcast a new block to all the peers in all the channels. Instead, it only contacts and sends a new block to the leader node of each channel to save the bandwidth. A leader node in an organization can be picked either by a static or dynamic method. An administrator can configure the network and decide which is the leader node in an organization—this is called a static vote. Alternatively, an administrator can also set up a dynamic election vote and interval time to reelect a new leader. In this case, a leader node regularly sends its heartbeat (alive message) to all other peers within its organization. If one or more peers do not receive the leader's heartbeat in a certain amount of time, these peers will start a new round of leader election procedure to select a new leader node [35].

Subsequently, the gossip protocol can be summarized in the three following steps:

- Peers continuously determine the available status of other peers by checking the sender's alive message.
- In a channel, peers propagate their ledger data to check if their data are consistent with the majority of the channel. If data in a node is out of date, a node will synchronize its data with the correct data from other nodes.
- Bring newly connected peer up to date by peer-to-peer sharing the update ledger data.

GRADUBIQUE

A. The Problem

Every school around the world has its own system to manage its students' grades or transcripts, and most of the schools do not share their students' transcripts for privacy reasons.

When an international student applies for further education in a foreign country, his/her transcript may need to be evaluated by an international evaluator because the academic grade systems are different between countries. For one reason, if the language in the transcript is a foreign language, it must be translated. Or the grading system in the transcript of an international student may not be consistent with the grading system in the country where this student wants to apply. In my situation, when I applied to a program in the United State, I had to complete following steps to get my paper work done: First, I had to request an official transcript from my school in Vietnam and mail it in a sealed envelope to the U.S. Next, I had to hire a legitimate transcript evaluator to translate and convert my transcript to the U.S grading standard.

Moreover, an employee may need to evaluate his/her academic transcript if he/she moves to another country while the employer requires an applicant's official transcript. Although most of the schools allow their students to print a transcript online, there is no guarantee of integrity check for the printed transcripts.

From a dollar-and-cent point of view, a student may have to spend $275 on average for a transcript evaluation which includes translation, evaluation, and shipping cost. Table 1 is the summary of evaluation cost we collected from the most popular credential evaluation companies: WES [36], ECE [37], and AERC [38].

Table. 1. Transcript evaluation cost summary.

	Translation cost (in USD)	Evaluation cost (in USD)	Shipping cost (in USD)	Total (in USD)
WES	$50	$205	$25	$280
ECE	$50	$195	$25	$270

AERC	$50	$200	$25	$275

The process of evaluating a paper-based transcript is complicated and time consuming. First, an international student has to look at the school website to find out what are the acceptable evaluation companies of the school. The best-case scenario is that the school accepts the original transcript then translate and evaluate the transcript by itself. However, if the school only accepts evaluated transcript from a professional company, it takes almost a month to evaluate the transcript. Then, a student has to contact the credential evaluation company to ask about the required documents. Some evaluation companies only accept translated documents. If this is the case, a student has to contact a certified translation company to translate his/her transcript first. We summarize the waiting time for transcript evaluation process in Table 2. This is the time frame to mail, translate and evaluate a transcript from an Asia country to a U.S school.

Table 2. Waiting time.

Mail from old school	Translation	Evaluation	Mail to new school	Total
7	5-7	5-7	3-5	20-26

According to Homeland Security [39], there was over 1 million international students of all fields in the U.S until March 2018. Homeland Security also reports that 77% of international students in the U.S are from Asia. Therefore, total cost for transcript evaluation at least 200 million (270 * 0.77 * 1 million).

The monetary cost and waiting time are substantially high if a student wants to apply to multiple programs on different continents. For example, Mike applies 3 school in the U.S and

these three schools accept 3 different evaluators. In addition, Mike also wants to apply 2 other schools in France. So, he has to repeat the process with the schools and evaluators in France.

Although some advanced technologies are used on the paper-based documents such as passport, and money to prevent counterfeit, it is unreasonable to utilize those technologies on paper transcripts. There is no guarantee for a paper-based transcript in a sealed envelope is a genuine document. Because counterfeit money can be made, it is not too hard to create an inauthentic paper transcript.

A blockchain solution for academic transcripts will fix these mentioned problems for following reasons. First, unofficial and official transcripts are recorded on a distributed ledger, so participants (school and employer) can access it globally in the internet. Second, transcripts stored on a distributed ledger is tamper-proof; a malefactor cannot alone alter a transcript because it is impossible to change transcript data in multiple nodes at the same time. Finally, it saves time and money while the academic records are totally reliable and accessible in the internet. We build Gradubique system to make the academic transcripts transparent and convenient for participants in the network.

B. Use cases

We define the following actors in our Gradubique system. Mike Nguyen finished his undergrad at Viet Nam National University (VNU). Nguyen took some courses under professor Ngo at VNU, before he studied at San Jose City College (SJCC) as a transition step to pursue his graduate education at San Jose State University (SJSU). Professor Paul is a grad coordinator and teaching the Object-Oriented Programing course at San Jose State University (SJSU). Employer IBM is interested to look at student’s transcript in its hiring process. Here are the use cases in Gradubique systems:

Scenario 1: To record data, Gradubique supports some fundamental functions. Professor Ngo defines a grading scheme for his teaching course at VNU. This scheme is the formula to calculate a course grade when he submits an assignment or test grade into the system. Ngo's grading scheme is the business logic that can be turned into a smart contract code which automatically calculates and updates the total course grade into the blockchain. This grade is an unofficial transcript which can be shared between universities. If Nguyen transfers to another university, this record might be useful for him as a completed pre-requisite class or fulfill graduation requirements at his new school. Fig. 15 is the use case diagram of scenario 1

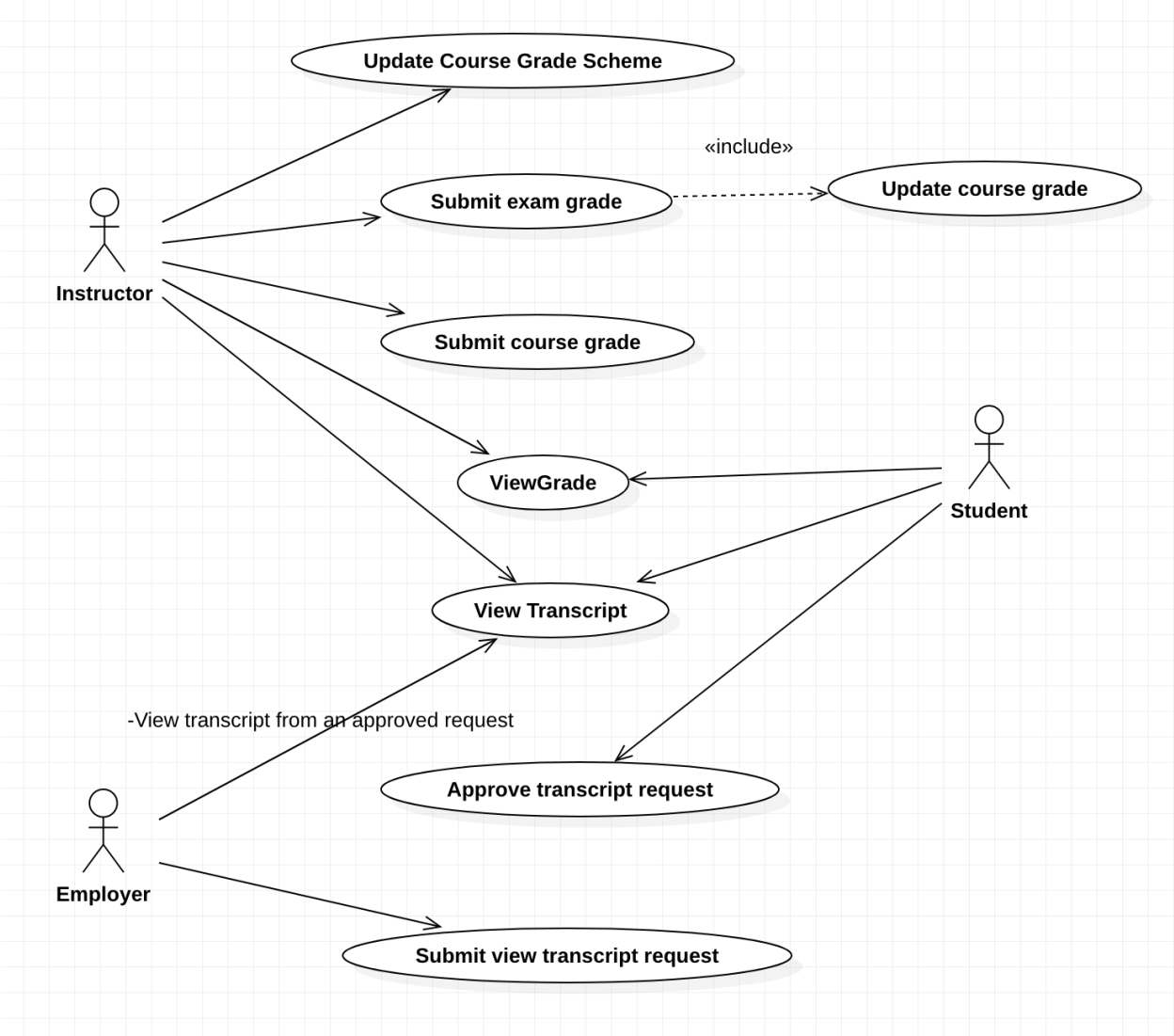


Fig.15. Scenario 1 use case diagram

Scenario 2: Nguyen is now studying at SJCC to fulfill some pre-requisite courses. As an instructor, Smith not only can define a grade scheme for his CS1B class, but also views Nguyen's transcript when Nguyen was at VNU. Nguyen certainly can view his transcript at both school of VNU and SJCC.

Scenario 3: We assume employers only consider an official transcript when they receive a job application. So, when Nguyen graduated from SJSU, Gradubique will generate an official transcript and add it into the distributed ledger which can be shared with employers. For privacy

reasons, employers cannot view a student's transcript unless the student agrees to share his/her transcript to employers.

Scenario 4: Employer IBM wants to look at Nguyen transcripts in its hiring process; however, for privacy reasons, Nguyen doesn't want his transcripts are publicly shown. So, IBM can send a request to Nguyen in Gradubique and view his transcript whenever the request is approved.

Scenario 5: Because coordinators at universities has similar roles to instructors, so computer science graduate coordinator Paul at SJSU can view Nguyen's transcript without asking his permission.

Scenario 6: Nguyen has a friend who works as a network administration at SJSU. Nguyen exploits this connection to steal the administration credential of SJSU server, so he can access the file server where the Gradubique ledger file is stored. However, Nguyen cannot change his grade in the ledger file even he has full-access rights and knows how to modify it. If Nguyen modifies the ledger in SJSU file server, the file is not consistent with other ledgers in the network, so Gradubique will sync the ledger files with other nodes to recover to the correct state.

C. System Design

We choose HLF to build Gradubique for the following reasons. First, HLF gives developer options to pick the programing languages among the very popular ones: Golang, Nodejs, Java, and Python. This is a pluggable modular architecture approach that allows developers to build their chain codes independently in terms of programing languages. For example, developer A uses Golang to write a chain code which functions grade submission, while developer B uses Nodejs to implement a code that lets users to view official transcript. Therefore, this platform utilizes the skillsets of developers. Second, Gradubique requirements fit with permissioned

blockchain because the participants need to be identified, and also the consensus mechanism is very flexible in HLF. At the channel-level policy, an administrator defines how a consensus can be met. In fact, when an administrator instantiates the chain-code into a channel, he will set the rule that indicates who and how many endorsements needed for a transaction. Finally, HLF has an enormous community support. Some of the big companies using and contributing to HLF are IBM, Intel, SAP, J.P. Morgan, Cisco, Airbus, etc. [40]

Gradubique is illustrated in Fig. 16 and the following terms are used to describe components in our system design.

- Gradubique represents the whole network which contains 4 organizations
- SJSU, SJCC, VNU, and IBM are 4 current organizations. Each organization has one or more peers.
- Peer is either an endorsing peer or a committing peer. To simplify the network configuration, we set up each organization contains one peer where chain codes is installed. This peer is also a committing peer to validate transactions within a block.
- Gradubique is designed with only 1 channel (academic channel) event though a network may contain more than one channel.
- A channel associate with a ledger (blockchain file) to record the transactions within a channel.
- Transactions in Gradubique are grade scheme, assignment grade, course grade, and requests to view transcript.

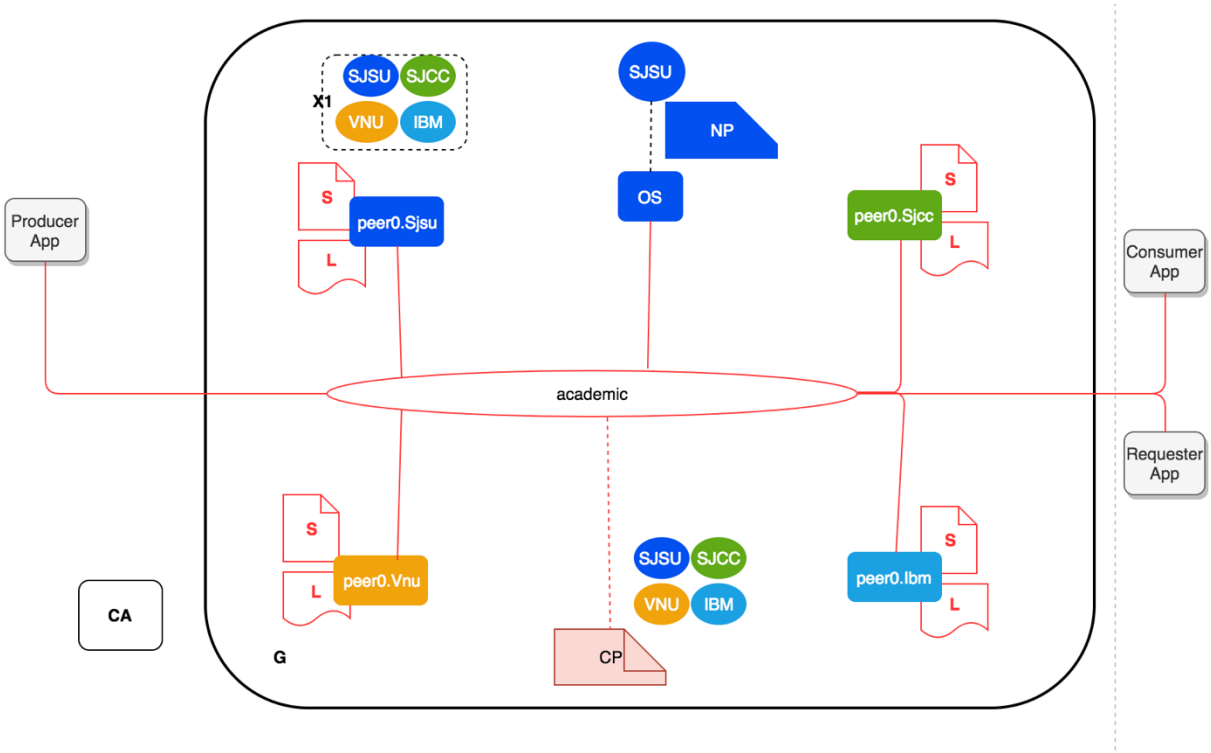


Fig.16. Gradubique diagram.

D. Network development

1) Policies:

Gradubique is built on top of the Hyperledger Fabric framework (HLF) which allows developers design a blockchain network with multiple layers of policies. Because HLF focuses on permissioned blockchain network, it requires members to have identities to join the network and an organization can choose its favor of certificate authority provider. However, to simulate the Gradubique network, we use cryptogen, a built-in tool in HLF, to generate the digital certificates for all the members in the network.

a. HLF Policy

- Network layer policy: This policy describes the administrative capabilities of the network. For example, organization O1 has the initial administrative role to create a blockchain network, then O1 adds organization O2 into the network and grants some

administrative roles to O2. Both O1 and O2 can create a consortium which defines a set of organizations who share a need to transact with one another. Organizations in a consortium communicate to each other via channel which also require a predefined policy.

- Channel layer policy: defines the channel-level administrative privilege of the members. For example, because organization O2 has network-level administrative role, O2 can add organization O3 to network. O2 then create a channel C1 and add O3 to this channel. Even though O1 has the network-layer administrative role, O1 cannot access information in channel C1 unless O2 or O3 grant permission. A transaction is read and written into the blockchain via channel, but it must follow a consensus rule of a channel. This consensus is ruled by HLF protocol and endorsement policy.
- Endorsement policy: indicates how a transaction is considered a valid transaction before it is added into the blockchain. A valid transaction must satisfy three following conditions: a transaction is endorsed by an authorized expected source; the number of transaction endorsements matches the declared requirement; all the endorsements are valid. A simple example of endorsement policy is that both O2 and O3 agree to add T1into the blockchain. In other words, if O2 says that the valid transaction T1 should be added into the blockchain, while O3 only agree to add T1'. This break the endorsement policy, because O2 and O3 can't make an agreement on this transaction, so neither T1 nor T1' is a valid transaction.

b. Gradubique policies

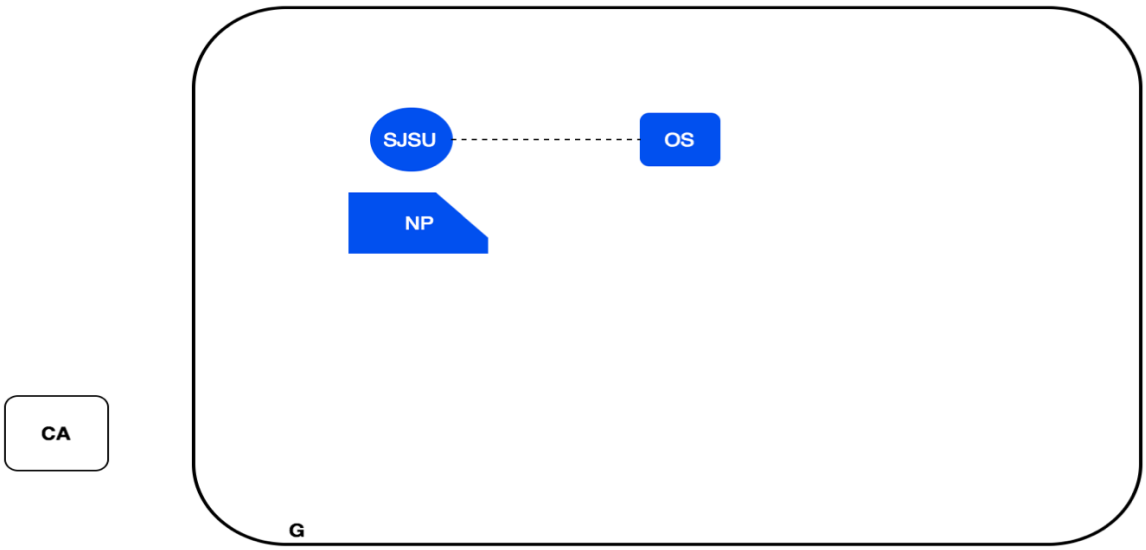
At the network level: SJSU is the initial organization who creates gradubique network and adds SJCC, Vietnam National University (VNU) and IBM as members. Gradubique network may grow if SJSU grant network administrative permission to other organizations, but in the scope of this paper SJSU is the only organization who can create gradubique consortium and academic channel.

An administrator at SJSU will create an academic channel where all the read and write sets of transactions go through. SJSU is also the unique organization who can manage members of the channel.

At first, endorsement policy defined in a simplest form, a valid transaction is determined by at least one endorsement signature of the peer. In other words, peer executes the chaincode to generate a transaction, sign the transaction by its private key to endorse the transaction and send back to client. Therefore, a transaction need at least 1 signature to be a valid transaction in Gradubique.

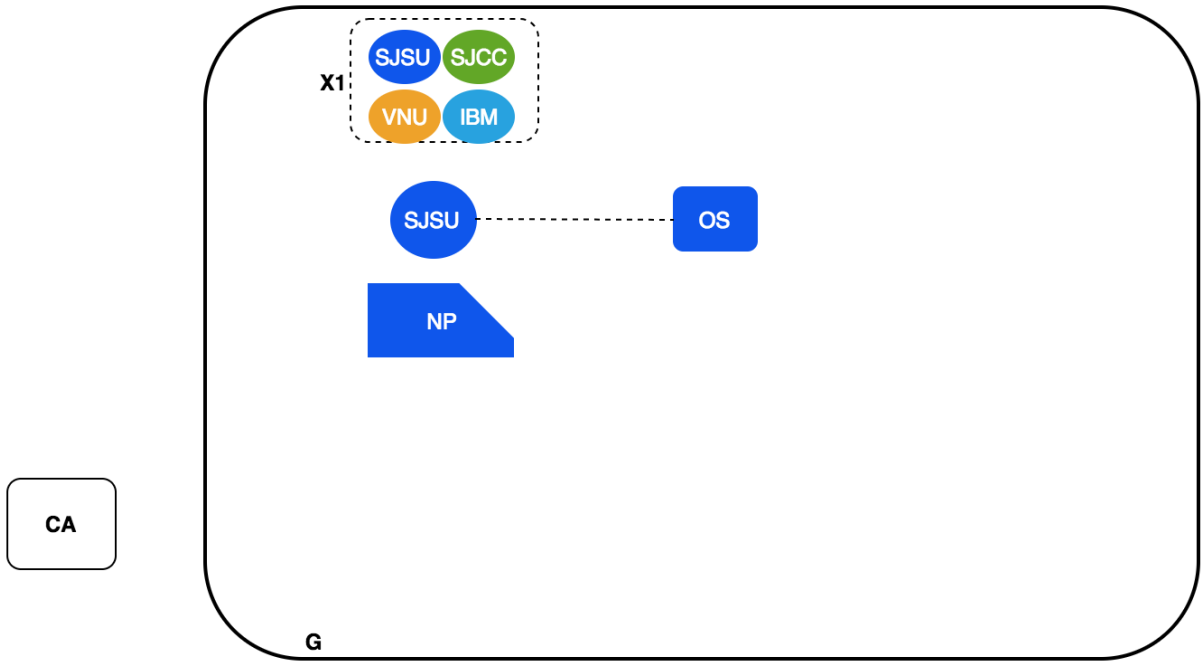
In Gradubique, instructors use Producer app to submit grade transactions and view transcript, while students use Consumer app to view grade and transcript. Requestor app is the place where employers create a request for a student to view his/her transcript. This request need to be approved by student in Consumer app before an employer can view a student transcript

2) Creating gradubique network



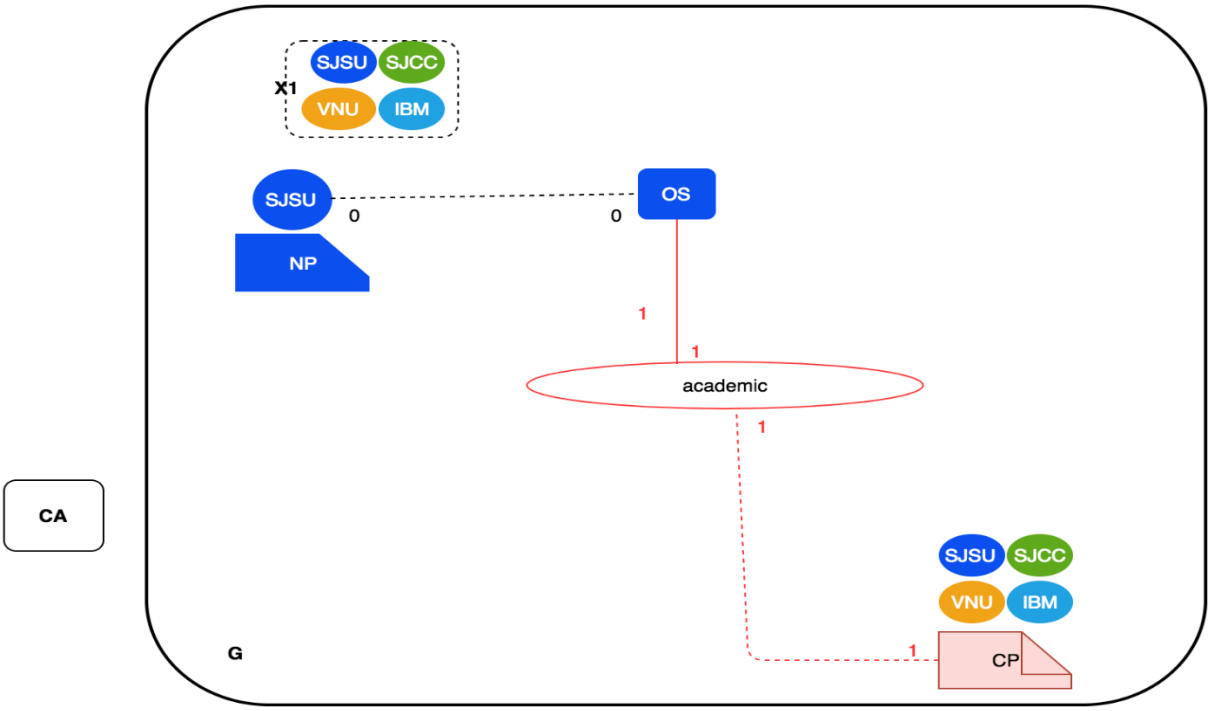
SJSU is the founder organization of Gradubique network, so SJSU defines a network policy NP and controls the ordering service OS. The Network policy NP indicates who has the right to add members to Gradubique. In the scope of this paper, SJSU is the only entity who can control the network and channel. Members participate in Gradubique must be an identifiable entity who has a digital certificate to access the network resource. Organization can pick any certification authority; however, to simplify the complexity of the network, we use cryptogen, a built-in tool in HLF, to generate members' certificates, and cryptogen tool implies the only certificate provider, named CA, in the Gradubique setup

3) Defining a consortium



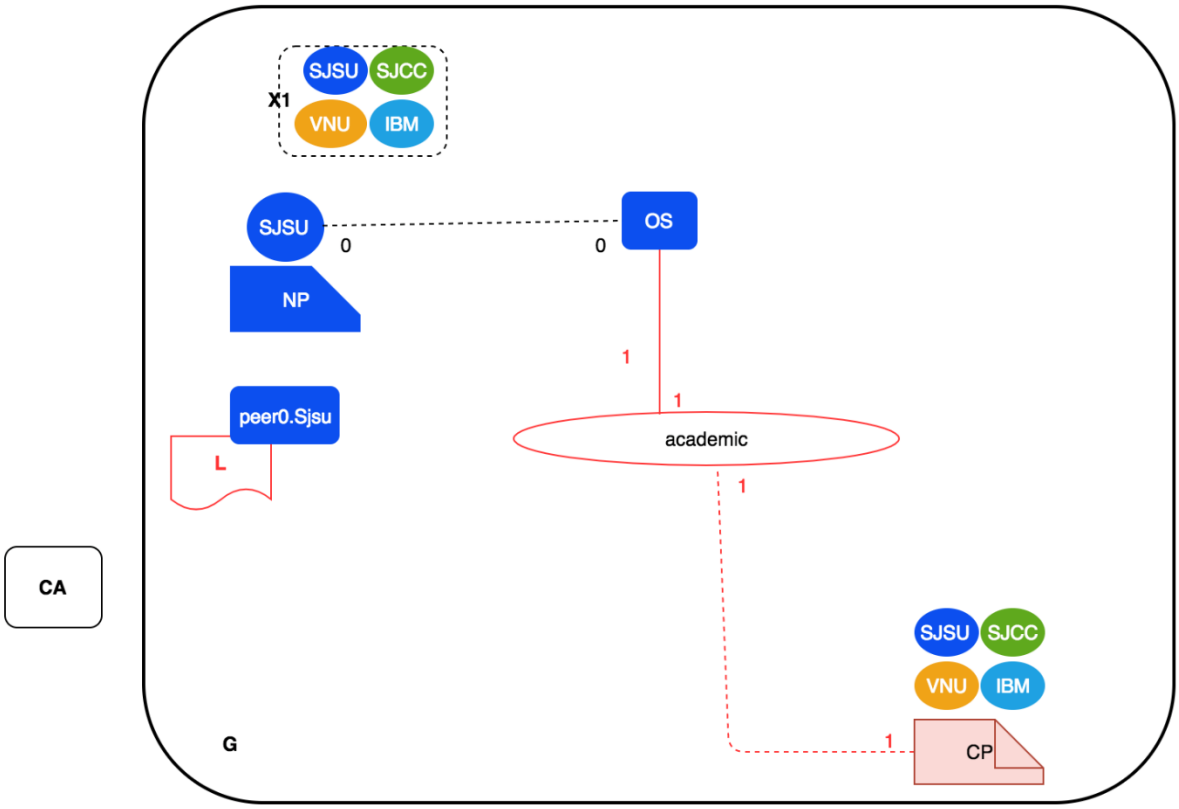
Since SJCC, VNU, and IBM want to join gradubique, a network administrator at SJSU defines a consortium X1 that contains 4 members. The consortium definition is stored in network policy file NP. Simply put, a consortium is a set of organizations which want to transact to each other. In this case, SJSU, SJCC, VNU and IBM want to share the transcript information.

4) Creating a channel



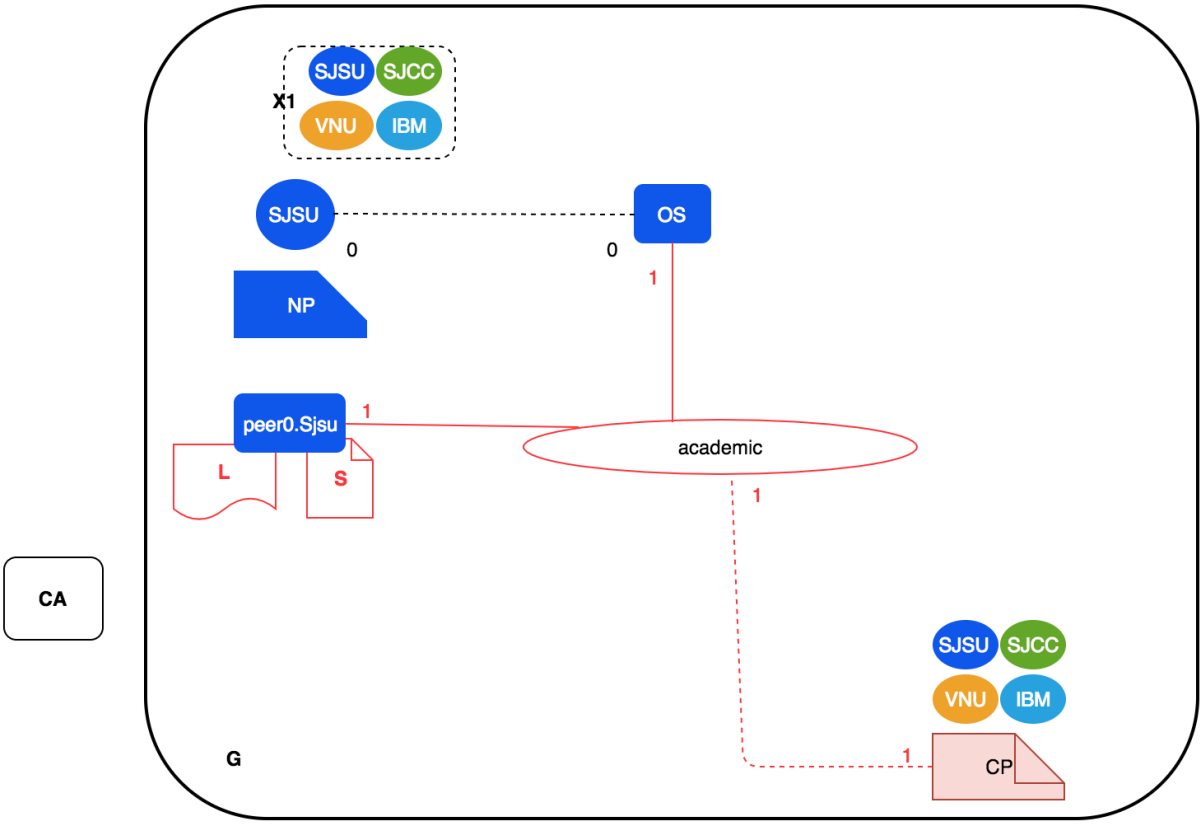
Members in a consortium communicate with each other via a channel. In gradubique use cases, an administrator creates academic channel as a medium for members to exchange information within consortium X1. Ordering service OS acts like a dispatcher in agent-base model, while peers are the agents of the model. “Academic” channel has a completely different policy, CP, to the network policy NP. At this moment, CP contains the policies that govern the rights of SJSU, SJCC, VNU and IBM have over the academic channel. Suppose SJSU administrator adds UC Merced(UCM) into the gradubique network but has not included UCM to academic channel. The resource in academic channel is unexposed to UCM.

5) Peer and Ledger



Peers are the network components where copies of blockchain files are hosted. Blockchain file is also called a ledger in HLF, and each peer can host one or multiple ledgers associate with the channels. If peer0.Sjsu participates in more than one channel, then peer0.Sjsu will host more than 1 ledger. In addition, peer0.Sjsu must own a valid certificate issued by CA to join a channel. When peer0.Sjsu is started, it asks OS to join academic channel, ordering service OS will use the channel policy CP to determine the right of peer0.Sjsu in the channel

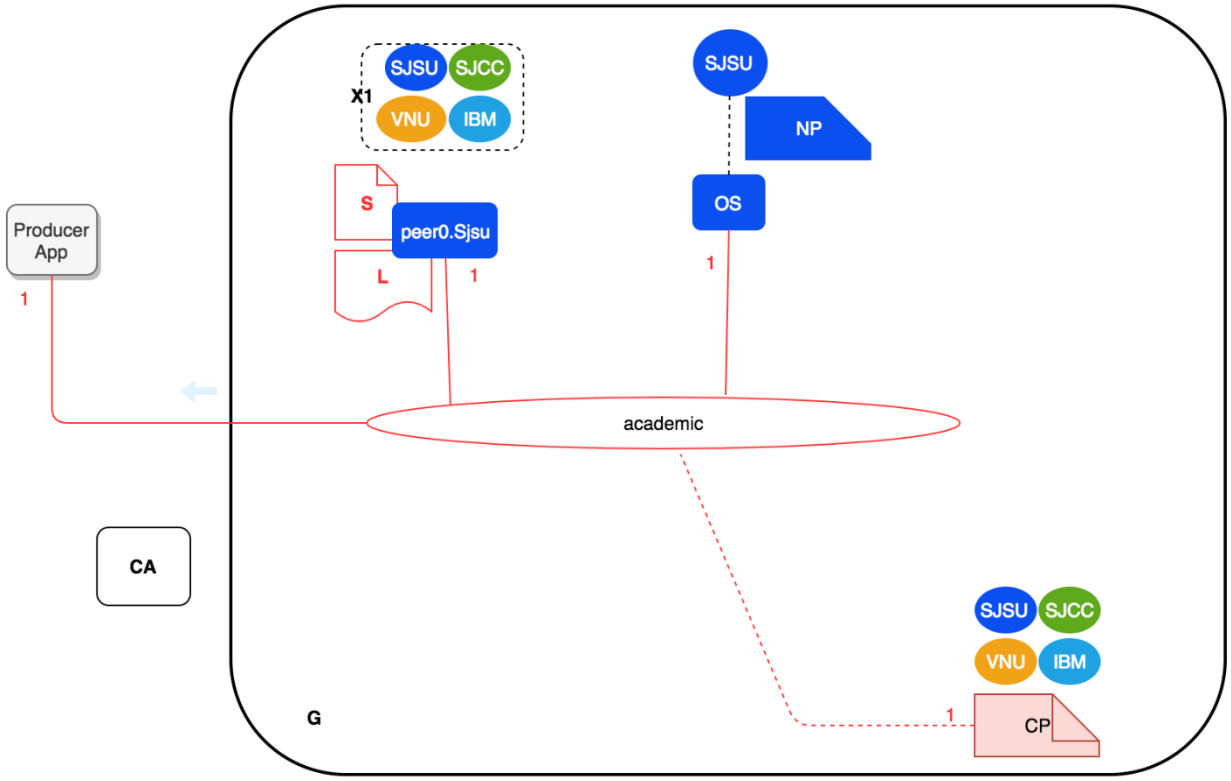
6) Peer and Smart Contract



Smart contract or Chaincode can be created by application developers in each organization to implement the business logic shared by consortium member. In gradubique, Chaincode can insert grades into the ledger, calculate total grade of a course base on a defined grade scheme, query and calculate the total GPA of students. Smart contracts in HLF are supported the programing language of Golang, node.js, Java, and Python.

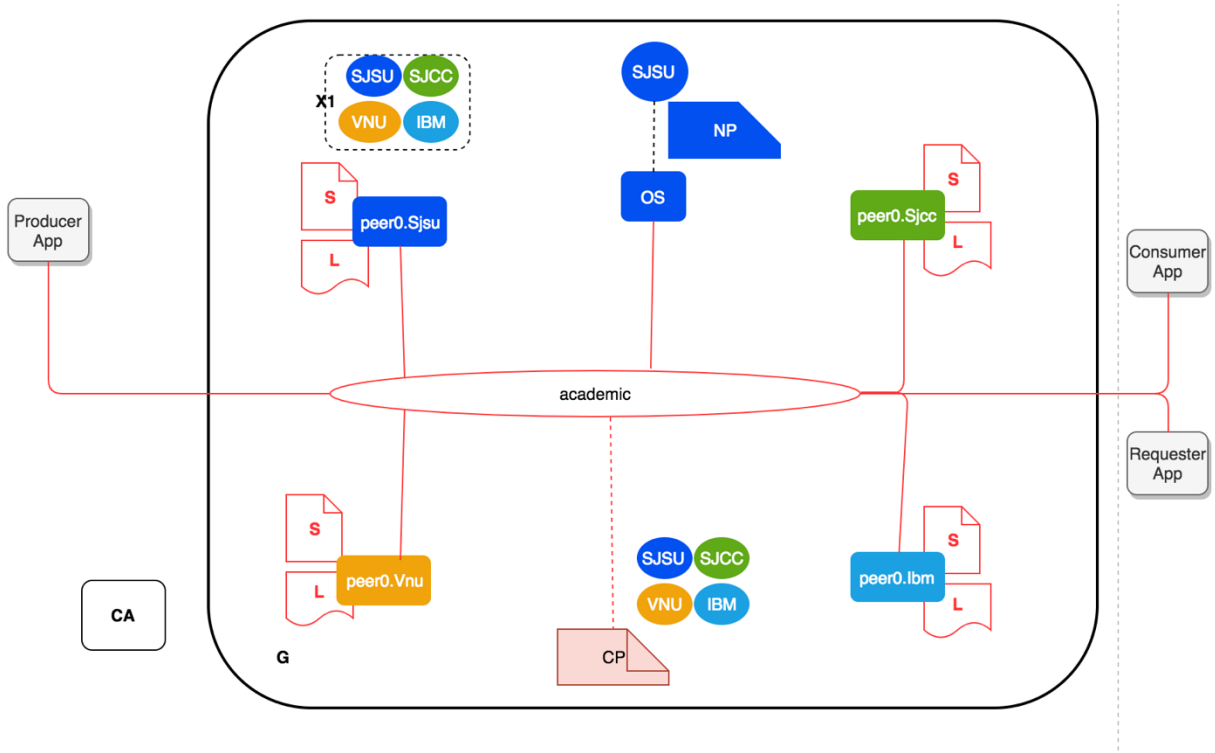
After smart contract S has been developed, an administrator in Sjsu must install it onto peer0.Sjsu. Smart contract S is now exposed as an interface which receive inputs from clients, execute the business logic and return the outputs to clients.

7) Application, peer, smart contract and ledger



A client application now can access transactions from ledger L, or insert transaction into ledger L via smart contract S. For example, an instructor can use Producer app to invoke the chain code S to insert a grade to ledger L. The querying and insertion of a transaction must follow HLF protocol which is described in section 4.

8) Network completion



Each organization has a peer to host a copy of a smart contract and a ledger. These components form a distributed ledger network that require a consensus among the peers before adding a transaction into the ledgers. For example, if we define an endorsement policy that requires 2 out of 3 peers (Sjsu, Sjcc, and Vnu) to sign off a transaction to make it a valid transaction, a grade record submitted by professor Ngo at Vnu is a legitimate record if and only if that transaction is signed by at least 2 peers among Sjsu, Sjcc and Vnu. In addition, consumer app is built to let students view grade and transcript, while Requestor App allow employers to create a request to view a student's transcript.

E. Implementation

We use Docker to simulate the Gradubique network. Docker is a virtualization technology that let developers pack runtime environments and applications into a container. This container works as a virtual machine which is installed all the necessary and lightweight

environments to run applications. So, Docker makes it easy to bundle, ship and run any application as a portable and self-sufficient computer virtually anywhere [41].

HLF provides built-in docker images for peers, orderers, and command line interface (cli). First, we set up organizations and peers as described in system design section. This set up is defined in "*docker-compose-gradubique.yaml*". Below is the snippet of code extracted from "*docker-compose-gradubique.yaml*" to define a peer in Gradubique

```
peer0.Sjsu.edu:
  container_name: peer0.Sjsu.edu
  extends:
    file: ./peer.yaml
    service: peer
  environment:
    - CORE_PEER_ID=peer0.Sjsu.edu
    - CORE_PEER_ADDRESS=peer0.Sjsu.edu:7051
    - CORE_PEER_LOCALMSPID=SjsuMSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.Sjsu.edu:7051
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdbSjsu:5984
    # provide the credentials for ledger to connect to CouchDB.  The username and password must
    # match the username and password set for the associated CouchDB.
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=admin
  volumes:
    - ./crypto-config/peerOrganizations/Sjsu.edu/peers/peer0.Sjsu.edu/msp:/etc/hyperledger/msp/peer
  ports:
    - 7051:7051
    - 7053:7053
  depends_on:
    - orderer.gradubique.net
    - couchdbSjsu
  networks:
    - gradubique
```

Academic channel and organizations participate in Gradubique are defined as following in "*cryptotx.yaml*" file.

```
Profiles:
  GradubiqueOrdererGenesis:
```

```
    Orderer:
      <<: *OrdererDefaults
      Organizations:
        - *OrdererOrg
    Consortiums:
      GRADUBIQUE:
        Organizations:
          - *Sjsu
          - *Sjcc
          - *Vnu
          - *Ibm

  AcademicChannel:
    Consortium: GRADUBIQUE
    Application:
      <<: *ApplicationDefaults
      Organizations:
        - *Sjsu
        - *Sjcc
        - *Vnu
        - *Ibm
```

For development and testing, HLF provides a crypto generator tool, called cryptogen, to create membership's certificates. File "*crypto-config.yaml*" defines the domain name of the organizations and number of certificate for each organization. Members' certificates are created by executing the command "*cryptogen generate –config crypto-config.yaml"*.

```
OrdererOrgs:
  - Name: Orderer
    Domain: gradubique.net
    Specs:
      - Hostname: orderer
PeerOrgs:
  - Name: Sjsu
    Domain: sjsu.edu
    Template:
      Count: 1
    Users:
      Count: 1
  - Name: Sjcc
    Domain: sjcc.edu
    Template:
      Count: 1
```

```
  Users:
    Count: 1
- Name: Vnu
  Domain: vnu.edu
  Template:
    Count: 1
  Users:
    Count: 1
- Name: Ibm
  Domain: ibm.com
  Template:
    Count: 1
  Users:
    Count: 1
```

At this point, developer need to write chain code to process the business rules of the application. Chain code in gradubique provides some following functionalities:

- It allows instructor to define a grading scheme for a course
- An instructor can submit an exam or assignment grade, then the smart contract will calculate the total grade based on the grading scheme of the course
- An instructor can view grade and transcript of a student
- An employer sends a request to view a student's transcript
- A student approves a request to view his/her transcript
- An employer views a transcript from an approved request.

Here is the snippet of chain code in Golang to handle a simple grading scheme submission request from back end

```
//submitGradeScheme puts a grade scheme for a course by semester
func (c *CourseProcessor) submitGradeScheme(stub shim.ChaincodeStubInterface, args
[]string) pb.Response {
    objStr := strings.Join(args, "")
    //fmt.Println("ObjString:", objStr)
    var gradeSchemeJS *GradeScheme
    err := json.Unmarshal([]byte(objStr), &gradeSchemeJS)
    if err != nil {
```

```
        fmt.Println("Error", err)
        return shim.Error("Submitted grade scheme is not a valid json object")
    }
    gradeSchemeJS.ObjectType = "gradeScheme"
    fmt.Println("Grade Scheme Json:", gradeSchemeJS)
    compositeKey := []string{gradeSchemeJS.School, string(gradeSchemeJS.Year),
gradeSchemeJS.Semester, gradeSchemeJS.Instructor,
        gradeSchemeJS.Course}
    schoolYearSemesterInstructorCourse, err := stub.CreateCompositeKey(
        "School~Year~Semester~Instructor~Course", compositeKey)

    // Use JSON to store in the Blockchain
    gradeSchemeAsBytes, err := json.Marshal(gradeSchemeJS)

    if err != nil {
        return shim.Error(err.Error())
    }
    // Use EntryID as key
    err = stub.PutState(schoolYearSemesterInstructorCourse, gradeSchemeAsBytes)
    if err != nil {
        return shim.Error(err.Error())
    }
    return shim.Success(nil)
}
```

On the front end, three applications are built using node.js :

- ProducerApp: instructors use this app to define grading scheme, submit assignment grade. Instructors can also view a student's transcript from this app.
- ConsumerApp: students view their grades and transcript from this app. In addition, a student can also approve requests from employers to view his/her transcript.
- RequestorApp: An employer uses this app to send a request to view a student's transcript. As the request approved, the employer can view the transcript from the request.

Here is the sample node.js code from front-end to process a request by an instructor to submit a grading scheme.

```
app.post('/submitGradeScheme', function(req, res) {
  var exams = new Array();
  var scheme = JSON.parse(JSON.stringify(req.body))
  var i = 1;
  var name = false, point = false, percentage = false;
  var exam = {'name': '', 'point': 0.0, 'receivablePoint': 0.0, 'percentage': 0.0}
  var totalPoint = 0.0
  //tranform data fields in the form to json object
  for (var key in scheme) {
    if (key === ('exam-name-' +i)) {
      exam.name = scheme[key];
      name = true;
    }
    if (key === ('exam-points-' +i)) {
      exam.receivablePoint = parseFloat(scheme[key]);
      totalPoint += exam.receivablePoint;
      point = true
    }
    if (key === ('exam-percentage-' +i)) {
      exam.percentage = parseInt(scheme[key]);
      percentage = true;
    }
    if (name && point && percentage) {
      i += 1;
      exams.push(exam);
      exam = {'name': '', 'point': 0.0, 'receivablePoint': 0.0, 'percentage': 0.0}
      name = false;
      point = false;
      percentage = false;
    }
  }
var gradeScheme = {'instructor': scheme.instructor, 'school': scheme.school, 'course': scheme.course,
'year': parseInt(scheme.year), 'semester': scheme.semester, 'totalPoint': totalPoint, 'exams':exams}
console.log("Grade scheme:", gradeScheme)
//Since chaincode accept string as paramater, we have to treat json object as a string
args = JSON.stringify(gradeScheme);
invokeProtocol.invoke(setting, "submitGradeScheme", args)
  .then(result => {
    res.render('responseMessage', {status: result[0].status, message: result[1]})
  })
  .catch(err => {
    res.status(500).send(err.toString());
  });
```

});

Now, the clients and chain code are ready. The next step is to start Gradubique network by executing the command “docker-compose -f docker-compose-gradubique.yaml up”. As the network is up and running correctly, we run the following script to set up channels and add members to the channels. This script also installs and instantiates our chain code into peers:

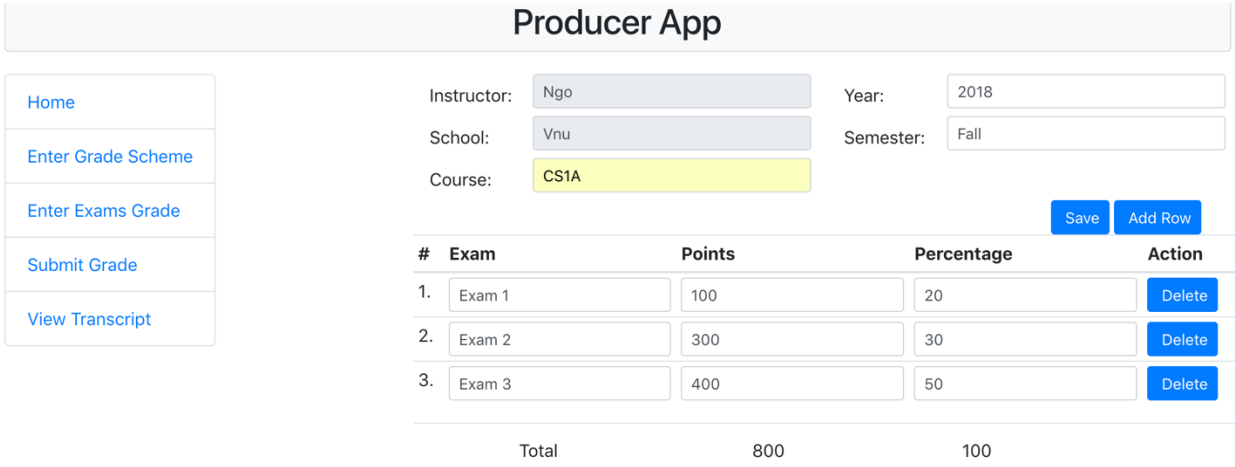
```
configtxgen -profile GradubiqueOrdererGenesis -outputBlock ./orderer/genesis.block

configtxgen -profile AcademicChannel -outputCreateChannelTx ./channels/Academic.tx -channelID academic

configtxgen -profile AcademicChannel -outputAnchorPeersUpdate ./channels/sjsuacademicanchor.tx -channelID
academic -asOrg SjsuMSP
configtxgen -profile AcademicChannel -outputAnchorPeersUpdate ./channels/sjccacademicanchor.tx -channelID
academic -asOrg SjccMSP
configtxgen -profile AcademicChannel -outputAnchorPeersUpdate ./channels/vnuacademicanchor.tx -channelID
academic -asOrg VnuMSP
configtxgen -profile AcademicChannel -outputAnchorPeersUpdate ./channels/ibmacademicanchor.tx -channelID
academic -asOrg IbmMSP

# Create academic channel
docker exec -it cli.Sjsu bash -c 'peer channel create -c academic -f ./channels/Academic.tx -o
orderer.gradubique.net:7050'

# Each organization join the channel
docker exec -it cli.Sjsu bash -c 'peer channel join -b academic.block'

docker exec -it cli.Sjcc bash -c 'peer channel join -b academic.block'

docker exec -it cli.Vnu bash -c 'peer channel join -b academic.block'

docker exec -it cli.Ibm bash -c 'peer channel join -b academic.block'

#Anchor peer
docker exec cli.Sjsu bash -c 'peer channel update -o orderer.gradubique.net:7050 -c academic -f
./channels/sjsuacademicanchor.tx'

docker exec cli.Sjcc bash -c 'peer channel update -o orderer.gradubique.net:7050 -c academic -f
./channels/sjccacademicanchor.tx'

docker exec cli.Vnu bash -c 'peer channel update -o orderer.gradubique.net:7050 -c academic -f
./channels/vnuacademicanchor.tx'
```

```
docker exec cli.Ibm bash -c 'peer channel update -o orderer.gradubique.net:7050 -c academic -f
./channels/ibmacademicanchor.tx'

# Intall and instantiate chaincode
docker exec cli.Sjsu bash -c 'peer chaincode install -p courseprocessor -n courseprocessor -v 0'
docker exec cli.Sjsu bash -c "peer chaincode instantiate -C academic -n courseprocessor -v 0 -c '{\"Args\":[]}'"

docker exec cli.Vnu bash -c 'peer chaincode install -p courseprocessor -n courseprocessor -v 0'

docker exec cli.Sjcc bash -c 'peer chaincode install -p courseprocessor -n courseprocessor -v 0'

docker exec cli.Ibm bash -c 'peer chaincode install -p courseprocessor -n courseprocessor -v 0'
```

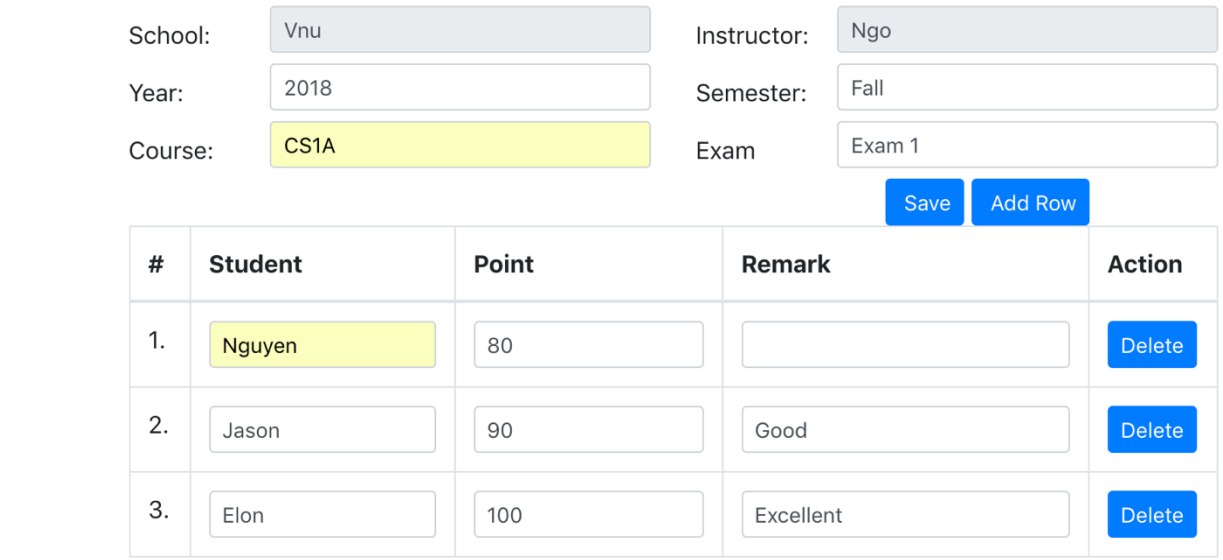
Now, the Gradubique network is up and running, chain codes on the peers are ready to process the requests from clients.

F. Demonstration

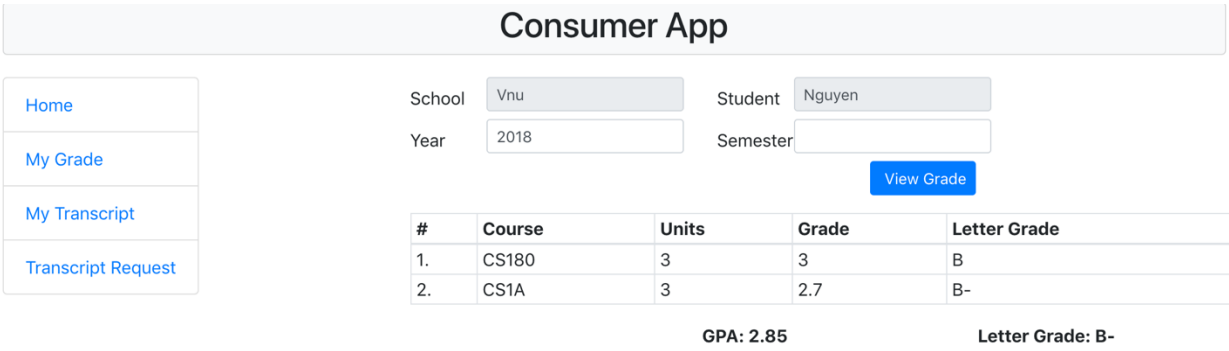
First, an instructor submits a grading scheme for his teaching course



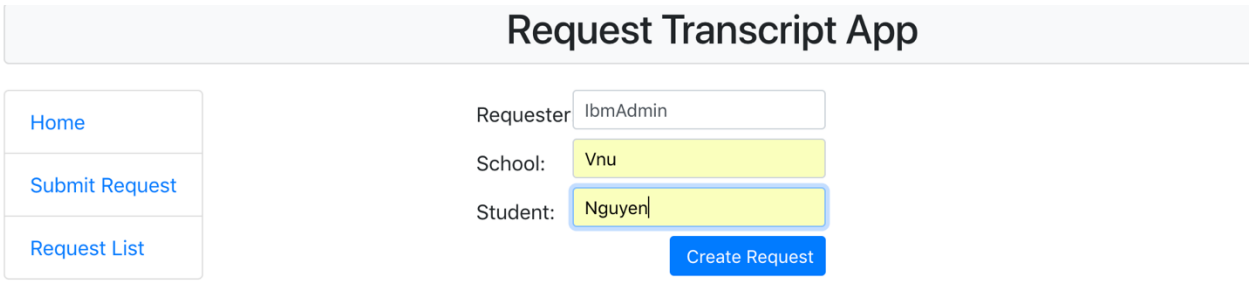
After defining a grading scheme, an instructor will enter an assignment grade or exam grade for students. The chain code will calculate final grade of student base on the defined grading scheme of the course.



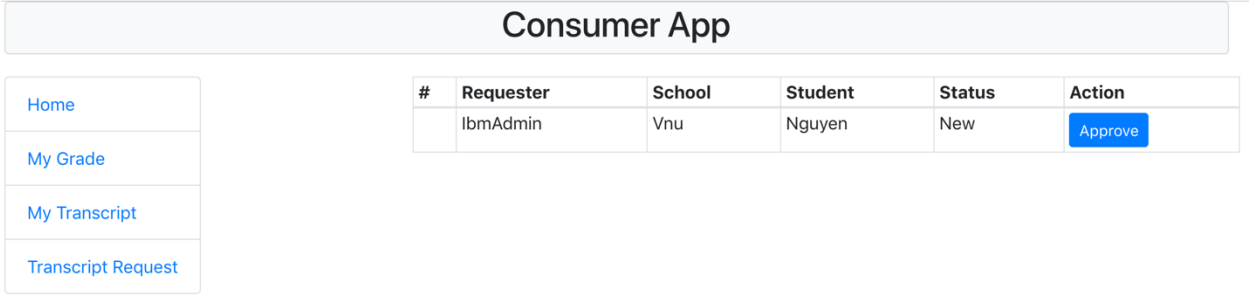
A student uses consumer app to view his/her grade and transcript



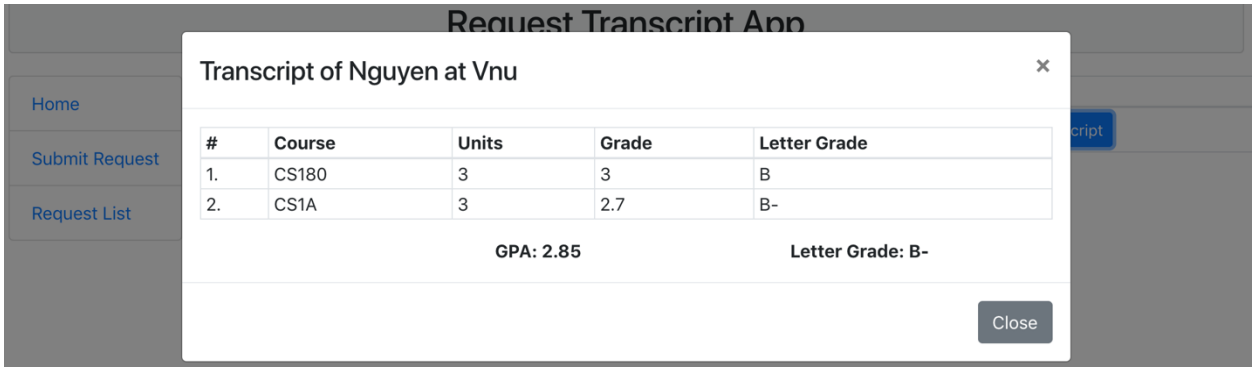
Nguyen's transcript is viewable by instructors or coordinators of the universities, but Nguyen's transcript is not viewable by employers. However, an employer can send a request to Nguyen to view his transcript



A student need to approve a transcript request from employer, so employer can view his/her transcript



An employer can view a student's transcript after its request is approved



FUTURE WORK

Our goal for this project is to study popular blockchain architectures which constitute mainly from 3 technologies: hashing, distributed system, and security. Blockchain is a new space that cultivates a lot of under development projects which attempt to solve the issues of scaling, performance, security, and power consumption. HLF is perfectly suitable for a permissioned blockchain network like Gradubique, because the actors in this domain must be identifiable. Going further from this paper, we will enhance Gradubique to make it not only a distributed academic records tracking, but also a distributed course and certificate management system.

First, grading scheme in Gradubique should be able to manage as multi-level category. For example, a grading scheme has 4 categories: test, homework, project, and final. The test section

can be split out to multiple sub-tests, while homework section may include many home works. In other words, the grading scheme may look similar to the following table:

	Receivable point	**Weighted Value**
Test	**390**	**50%**
Test 1	100	12.5%
Topic 1	*30*	3.75%
Topic 2	*40*	5.0%
Topic 3	*30*	3.75%
Test 2	65	12.5%
Test 3	50	12.5%
Test 4	75	12.5%
Homework	**100**	**10%**
HW1	50	5%
HW2	50	5%
Project	**10**	**10%**
Project 1	10	10%
Final	**150**	**30%**

A described grading scheme in a table above will be useful for an employer or a school to look at a specific skill of a student. For example, Mike has a good grade at agent-base model which is the topic 1 of test 1 in the CS251A class. So, a recruiter can zoom in the detail of Mike's transcript and evaluate this skill as a criterion for the company hiring position. In addition, Gradubique will also be a place to store all the official diplomas and certificates generated by schools for students.

Furthermore, Gradubique can be expanded to store all the standard test scores and professional certificates for students. For example, SAT, TOEFL, GRE, GMAT test scores are kept on Gradubique, so a university can look at students' profiles in one place to evaluate its student candidates. Likewise, professional certificates certified by companies such as Java certificates by Oracle, or CCNA by Cisco can also be added to Gradubique.

REFERENCES

[1] Wikipedia, "Wikipedia," Wikimedia Foundation, Inc, 28 3 2010. [Online]. Available: https://en.wikipedia.org/wiki/Ledger#cite_note-1. [Accessed 20 4 2018].

[2] Wikipedia, "1911 Encyclopædia Britannica/Book-Keeping," Wikimedia Foundation, Inc, 23 11 2016. [Online]. Available: https://en.wikisource.org/wiki/1911_Encyclopædia_Britannica/Book-Keeping. [Accessed 20 4 2018].

[3] S. Haber and S. W. Stornetta, "How to Time-Stamp a Digital Document," in *Lecture Notes in Computer Science*, vol. 537, Berlin , Heidelberg: Springer, 1991, pp. 437-455.

[4] A. Back, "Hashcash," Adam Back, 38 Mar 1997. [Online]. Available: http://www.hashcash.org. [Accessed 11 April 2018].

[5] H. Finney, "Reusable Proofs of Work," IBM Research, [Online]. Available: http://nakamotoinstitute.org/finney/rpow/theory.html. [Accessed 21 4 2018].

[6] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Unknown, Unknown, 2008.

[7] Ethereum, "Ethereum White Paper," Github, [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum-accounts. [Accessed 28 4 2018].

[8] N. Popper, "New York Times," The New York Times, 27 2 2017. [Online]. Available: https://www.nytimes.com/2017/02/27/business/dealbook/ethereum-alliance-business-banking-security.html. [Accessed 23 4 2018].

[9] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382-401, July 1982.

[10] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in *Advance in Cryptology - CRYPTO' 92*, Berlin, 1993.

[11] M. Jakobsson and A. Juels, "Proofs of Work and Bread Pudding Protocols(Extended Abstract)," in *Secure Information Network*, vol. 23, Boston, MA: Springer, 1999, pp. 258-272.

[12] I. Bashir, Mastering Blockchain, L. Subramanian, V. Pagare, A. Menon and N. Sawakhande, Eds., Birmingham, Birmingham: Packt Publishing Ltd., 2017.

[13] N. Community, "NXT Wiki," 3 Feb 2015. [Online]. Available: https://web.archive.org/web/20150203012031/http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt#Blocks. [Accessed 11 April 2018].

[14] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Operating Systems Design and Implementation*, New Orleans, 1999.

[15] Wikimedia, "Fedwire," [Online]. Available: https://en.wikipedia.org/wiki/Fedwire. [Accessed 01 05 2018].

[16] BitPay, Inc, "Bitcore Library," BitPay, 1 1 2013. [Online]. Available: https://bitcore.io/api/lib. [Accessed 19 4 2018].

[17] Bitcoin.org, "Bitcoin Developer Guide," The bitcoin foundation, 18 04 2018. [Online]. Available: https://bitcoin.org/en/developer-guide#block-chain. [Accessed 18 04 2018].

[18] MediaWiki, "WikiMedia Common," Wikimedia, 9 9 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png. [Accessed 18 4 2018].

[19] Bitcoincash, "Bitcoin Cash," Bitcoin Cash, 1 8 2017. [Online]. Available: https://www.bitcoincash.org. [Accessed 18 4 2018].

[20] Ethereum, "Ethereum Wiki," Github, 28 3 2015. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Accessed 19 4 2018].

[21] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger EIP-150 Revision," 2015.

[22] C. Dannen, Introducing Ethereum and Solidity-Foundation of Cryptocurrency and Blockchain Programming for Beginners, Brooklyn, NY: Springer Science + Business Media Finance Inc, 2017.

[23] atomh33ls, "Ethereum Stack Exchange," Stack Exchange, [Online]. Available: https://i.stack.imgur.com/YZGxe.png. [Accessed 7 5 2018].

[24] P. Kasireddy, "Medium," Medium, [Online]. Available: https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369. [Accessed 7 5 2018].

[25] Wikipedia, "Application Specific Integrated Circuit," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Application-specific_integrated_circuit. [Accessed 8 5 2018].

[26] Google, "Google Cloud," Google, [Online]. Available: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu. [Accessed 8 5 2018].

[27] Ethereum Wiki, "Ethash," Github, [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethash. [Accessed 8 5 2018].

[28] V. Pradeep, "Ethereum's Memory Hardness Explained, and the Road to Mining It with Custom Hardware," 28 4 2017. [Online]. Available: https://www.vijaypradeep.com/blog/2017-04-28-ethereums-memory-hardness-explained/. [Accessed 8 5 2018].

[29] Linux Foundation, "Hyperledger White Paper," Linux Foundation, [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf. [Accessed 25 04 2018].

[30] Hyperledger Fabric, "Hyperledger Fabric - Membership," Hyperledger, [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/membership/membership.html. [Accessed 25 04 2018].

[31] Hyperledger Fabric, "Hyperledger Fabric - Chaincode," Hyperledger, [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/chaincode4noah.html?highlight=chaincode. [Accessed 25 04 2018].

[32] Hyperledger, "Hyperledger Fabric - Channels," [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/channels.html?highlight=gossip%20protocol. [Accessed 25 4 2018].

[33] Hyperledger Fabric, "Hyperledger Fabric-Ledger," [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/ledger.html. [Accessed 25 4 2018].

[34] Hyperledger Fabric, "Hyperleder Fabric - Peer," Hyperledger, [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/peers/peers.html. [Accessed 25 4 2018].

[35] Hyperledger Fabric, "Hyperledger Fabric - Gossip Protocol," Github, [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.1/gossip.html?highlight=gossip%20protocol. [Accessed 26 4 2018].

[36] World Education Service, "WES," [Online]. Available: https://www.wes.org/evaluations-and-fees/education/graduate-admissions/. [Accessed 14 5 2018].

[37] Educational Credential Evaluation, "ECE," [Online]. Available: https://www.ece.org/ECE/Individuals/Services--Fees. [Accessed 14 5 2018].

[38] American Education Research Corporation, "AERC," [Online]. Available: http://www.aerc-eval.com/fees.html. [Accessed 14 5 2018].

[39] Homeland Security, "Study in the States," [Online]. Available: https://studyinthestates.dhs.gov/sevis-by-the-numbers/march-2018. [Accessed 15 5 2018].

[40] Hyperledger, "Hyperledger," The Linux Foundation, [Online]. Available: https://www.hyperledger.org/members. [Accessed 30 4 2018].

[41] Docker, "Docker," Docker Inc, [Online]. Available: https://www.docker.com. [Accessed 1 5 2018].

[42] T. D. Joseph Poon, "The Bitcoin Lightning Network: Scalable O -Chain Instant Payments," 2016.

[43] Ethereum, "Sharding FAQ," [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQ. [Accessed 9 5 2018].

[44] National Science Board, [Online]. Available: https://www.nsf.gov/statistics/2018/nsb20181/data/tables. [Accessed 15 5 2018].