A Thesis

For the Degree of Master of Science

Interworking Mechanism of Blockchain Platforms for Secure Tourism Service

Lin Chao Zhang

Department of Computer Engineering

Graduate School

Judo National University

Dec 2019

안전한 관광 서비스를 위한 블록체인 플랫폼의 연동 메커니즘

지도교수 김 도 현

Zhang LinChao

이 논문을 컴퓨터공학 석사학위 논문으로 제출함

2019 년 12월

Zhang LinChao의 컴퓨터공학 석사학위 논문을 인준함

심사위원장 안 기 중 ㉑ [illegible]

위 원 송 왕 철 ㉑ [illegible]

위 원 김 도 현 ㉑ [illegible]

제주대학교 대학원

2019년 12월

Interworking Mechanism of Blockchain Platforms for Secure Tourism Service

Zhang LinChao
(Supervised by professor Do-Hyeum Kim)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Computer Engineering

2019. 12

This thesis has been examined and approved.

[illegible]

Thesis Committee Chair, Khi-jung Ahn,Prof. of Computer Engineering

송 왕철

Thesis Committee Member, Wang-Cheol Song,Prof. of Computer Engineering

Do Hyeum kim

Thesis Supervisor, Do-Hyeum Kim,Prof. of Computer Engineering

December 2019

Department of Computer Engineering
GRADUATE SCHOOL
JEJU NATIONAL UNIVERSITY

Table of Contents

List of Figures

List of Table

Acknowledgment

There are many critical points in life, and there are many people in life who are grateful to remember. This thesis was completed under the careful guidance and careful teaching of **Prof. Do-Hyeun Kim.** From the selection of the topic, the method of argumentation, and the collection of primary materials, the teacher's hard work, and sweat were all poured out, which enabled me to grasp the writing direction of the thesis and completed it. The writing of the thesis expresses the most sincere gratitude here! The tutor's teachings and cordial care are vivid and vivid, and the profound knowledge, rigorous academic attitude, high professionalism, broad mind, and sunshine life philosophy of the instructor will surely be in my future life trajectory. Play a guiding role. On the occasion of the completion of this thesis, I would like to extend my sincere respect and blessings to the teachers.

Thanks to all the teachers, friends, brothers, and sisters during my study period, thank you for your care and help in my work and life. My life is more beautiful and meaningful because of you. The teacher's teachings will be remembered in the heart; the friendship between students will last forever.

I need a special thanks to my parents. Parents' parenting has no reason to report. They are the strong backing on my way to school. When I am faced with the confusion of life choices, I will solve my problems. Their selfless love and care for me is the driving force for my continuous advancement.

Finally, thanks to the help of the **Lei Hang** seniors, help me answer and point the direction when I have problems. Allow me to save more time to study monographs and write essays.

Abstract

Since the launch of Bitcoin in 2009, research on blockchain technology has rapidly raised in the field of study and application. Blockchain technology shows the great potential to address interoperability challenges in current IT systems and is hoped to become a part of the core technical standard, enabling individual stakeholders to share data more securely. However, blockchain technology is still in its infancy and requires more improvement. At present, the common problems of blockchain are low efficiency and poor scalability. Most of the existing blockchain systems are built on single chains, which lacks the solution for enterprise applications.

In this paper, we present an interworking mechanism to enable cross-domain communication for multiple chains. This proposed architecture combines private and public chains to form the network infrastructure, and the interaction is achieved through a routing management application. This application acts as the intermediate between multiple chains, which transmits the data from one to another chain. The blockchain network can be extended to support more parallel chains according to different business requirements. This paper applies the multiple chain architecture schemes to secure tourism services and verifies the feasibility of the proposed architecture. The proposed secure tourism service platform consists of three main components: the main-chain, sub-chain, and interworking app. To demonstrate the usability of the proposed approach, a proof of concept is implemented using WindingTree and Hyperledger Fabric. The efficiency of the designed approach is demonstrated through a series of experimental tests using different performance metrics.

Chapter1. Introduction

The concept of the blockchain[1] is proposed by Satoshi Nakamoto, which is also known as the decentralized ledger technology that each participant maintains a replica of a shared append-only ledger of digitally signed transactions. Since Bitcoin[2] appeared, blockchain technology has developed rapidly, and more and more financial services have been developed on the characteristics of Bitcoin[3]. Blockchains are generally divided into the following three categories[4]: public blockchain, private blockchain[5], and consortium blockchain. The three main types of blockchain are different: a public blockchain network is entirely open, and anyone can join and participate in the network; a private blockchain requires an invitation and must be validated by either the network starter or by a set of rules put in place by the network starter, who places restrictions on who is allowed to participate in the network, and only in certain transactions; a consortium blockchain is not granted to a single entity as a private blockchain; rather, it is granted to a group of approved individuals.

Up to now, most of the existing blockchain platforms are built on single networks. As one of the popular blockchain platform, Ethereum can only process a dozen transactions in 1 second at the initial stage, and after optimization, it can reach a maximum of 400[6] transactions in 1 second. Applications built on single chains are inefficient for enterprise requirements. However, the performance of a single-chain base network still cannot fill the needs of enterprise applications, which can generate millions of transactions in 1 second. Enterprises are still used as an example. As an example, for the tourism industry, there is a large amount of business and privacy information from markets and customers. To deal with such an amount of data, a single-chain network cannot systematically provide services on the premise of ensuring efficiency.

Moreover, the cost of a single-chain network is high, and the transaction throughput is insufficient. As a result, a single-chain network is not an appropriate solution. Multi-chain architecture can solve more problems from performance to service than single-chain. Multi-chain architecture can divide an original chain into multiple chains, each chain is responsible for part of the calculation and storage business, and has scalability, that is, the number of chains can increase with the increase of business volume and data. As the number of chains increases, the overall performance of the system improves, and the storage space of the system can also expand as the number of chains increases.

However, the principles of the current multi-chain architecture can only solve part of the problem (increasing the scalability of the network, improving the throughput of the network), and cannot address the cost of the multi-chain architecture and the single transaction efficiency[7] of network transactions. For example, multi-chain is not satisfied with the benefits brought by a single blockchain system.

We propose a configurable multichain, which is easy to configure and can work with different blockchains. The proposed multi-chain architecture is based on a private chain and can be used in parallel with other blockchains. When users in the private chain trade with the main chain, the information transmitted from the main chain are recorded in the private chain to implement the private chain. Privacy control and comprehensive management of user rights for users in the private chain solve mining, privacy, and openness issues. For instance, the size of the block, who can connect transactions, and who can publish transactions. The multi-chain structure proposed in this paper is not only to solve the scalability problem but also to improve the performance of the network. It has different requirements for specific functions and is implemented through various blockchain networks. Each single-chain implements a local service and these single chains are combined into a multi-chain network according to application requirements.

In this paper, the theory and practice of the multi-chain architecture[8] are applied to the tourism industry. We present an interworking mechanism of blockchain platforms based on multi-chain for secure tourism services. The tourism service platform is divided into business systems and transaction systems. A public chain based on WindingTree[9] is implemented to store the information from markets and customers. More precisely, WindingTree in the business system is to allow the connection between customers with service providers such as airlines, hotels, and tour guides for reducing service costs. This is achieved by using smart contracts, and the ERC827 protocol to save costs for all stakeholders in the travel industry. Meanwhile, a private chain based on Hyperledger Fabric is implemented to provide functions such as privacy control and role management. The characteristics of a private chain are utilized to solve the problems of privacy protection and user permission control. Applications can call and operate the specified blockchain through various APIs provided by the routing management service.

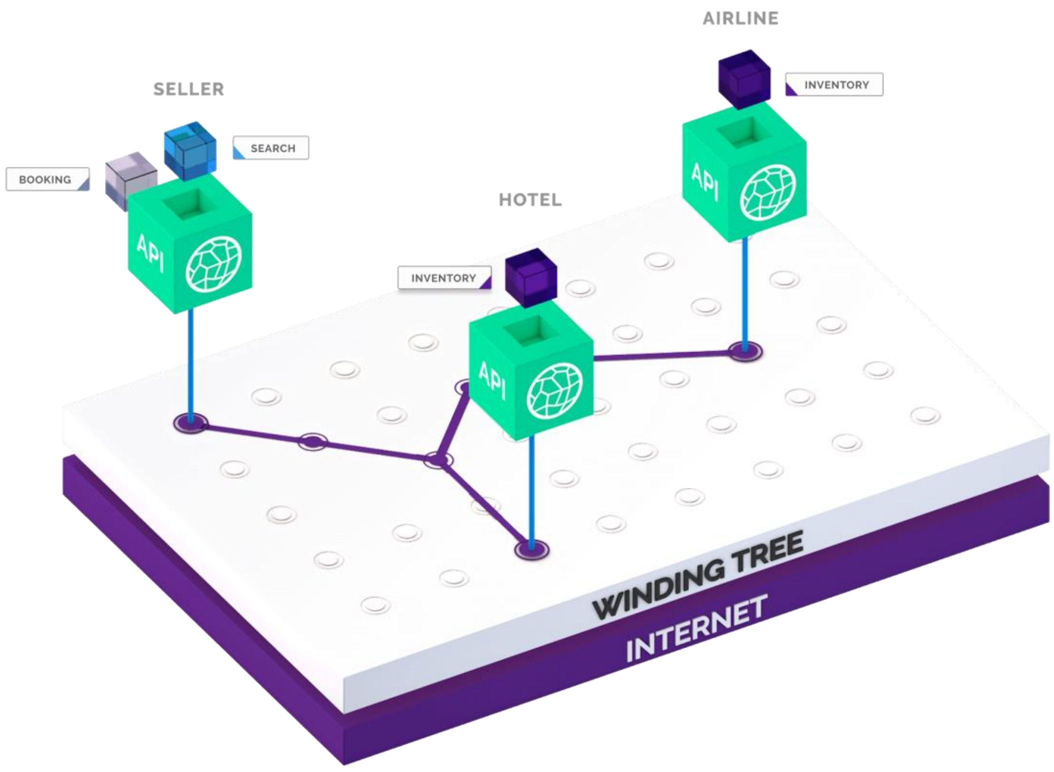


Figure 1: WindingTree Network Model Diagram

The contributions of this paper can be summarised as follows: first, a multi-chain architecture model is proposed, where the public chain is used as the information sharing chain and the private chain is responsible for privacy control. Second, a proof of concept for tourism service is implemented on the top of the proposed architecture. Third, the performance of the proposed approach is evaluated through various performance indexes.

This rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 describes the principle of multi-chain architecture. Section 4 presents the tourism service platform based on the proposed multi-chain architecture. Section 4 illustrates the design and implementation of main-chain based on WindingTree. Section 6 details the design and implementation of sub-chain based on Hyperledger Fabric. Section 7 evaluates the performance of the proposed approach. Section 8 concludes the paper and discusses the future research directions.

Chapter2. Related Work

Most of the blockchains in the travel market are Single private chains, and they structure all the business in the travel market in a Single-chain blockchain network. Such a structure makes the current tourism blockchain in the market very bloated, and the network performance is low (the same time and data volume, the fewer network functions, the higher the network performance). With the expansion of the tourism market, the transaction volume increases. The processing speed of a Single-chain is far less than a request for a transaction Will cause transaction delays and network crashes. The design of the Multi-chain architecture has resulted in multiple medium-type network designs, such as parallel chain, side chain, and cross-chain designs. At a large number of Single-chain tourism, platforms appear in the Single-chain tourism market, and the deficiencies of Single chains in tourism platforms are addressed through Multi-chains.

In the Single-chain market [10], the development of shared and private chains occupies the majority, and they have outstanding performance in most specific functions, but in actual business environments, the demand is often higher than the characteristics of Single-chain networks. Forcing the implementation of practical business functions is mostly achieved by a method that consumes network performance, resulting in low overall Single-chain network performance. For example, the networks developed by Bitcoin[11] and Ethereum are public chains. They are completely decentralized and correctly solve the problem of trust. The existing public chain also shows many shortcomings:

Malicious nodes. Although the public chain is very secure, imagine that it is difficult to reach consensus with so many nodes that randomly enter and exit, because some nodes may go down at any time, and hackers may forge many fake nodes. The public chain has a stringent consensus mechanism. The consensus problem directly leads to the speed of public chain data processing. Bitcoin transfers can take a long time to arrive.Privacy issues. At present, the data transmitted and stored on the public chain are publicly visible, and the parties to the transaction implement privacy protection through the "pseudo-anonymity" mechanism. For specific business scenarios involving a large number of trade secrets and benefits, the exposure of data does not meet business rules and regulatory requirements.

The advantages of private chains are:

1. Enterprise permissions: Enterprises control resources and access to the blockchain.

2. Faster transaction speed: There are nodes distributed locally, but fewer nodes are participating in blockchain activities, and the transaction speed will be faster.

3. Better scalability: Add nodes and services as needed to provide enterprises with more scalability.

4. Get support for compliance: Service providers may need to comply with compliance requirements so that the control network can more easily meet compliance requirements.

5. The consensus is more efficient (fewer nodes): enterprises or private chains have fewer nodes, usually using different algorithms

The above d concludes that the disadvantages of the public chain and the advantages of the private chain are complementary.

This thesis proposes a Multi-chain architecture according to functional requirements, which can also be referred to as a parallel architecture. Different business function requirements are solved through different blockchains, which explains the problem of insufficient functionality on a Single-chain.

1. Solve the problem of insufficient Multi-chain functionality. Single-chain networks cannot meet functional requirements in practical applications and usually can only deal with technical issues by reducing the performance of the network.

2. Provide privacy management between businesses. In a market environment, private information between companies is essential. Increasing communication between enterprises through public information is also an important channel.

3. Multiple chains increase capacity issues. The information stored in the blockchain is particularly expensive, and a certain amount of data cannot be stored in the chain, increasing the blockchain capacity through a Multi-chain architecture.

4. Business diversification, growing requirements for blockchain scalability. Multi-chain architecture can expand the chain according to business needs. The vast majority of blockchain platforms on the market are based on the two most popular blockchain platforms, Hyperledger Fabric and Ethereum, but they are not mature enough and may cause unforeseen problems in blockchain deployment.

2.1 Blockchain

Traditional Internet transactions need to rely on trusted third-party institutions to process electronic payment information, and both parties to the transaction rely on third-party institutions. However, third-party involvement also has many disadvantages:

-High transaction costs (charging specific fees)

-Exposure of privacy (in order to verify information, third parties need to provide information about both parties to the transaction)

Imagine if the third-party organization [12] is removed and the two parties in the transaction deal directly, how can this ensure that the transaction takes effect. The emergence of blockchain technology is to solve such problems. Blockchain is based on cryptographic principles[13] and not based on trust mechanisms so that the two parties who reach an agreement can directly trade and publish to all witnesses. It is a new application model of computer technology such as distributed data storage, point-to-point transmission[14], consensus mechanism, and encryption algorithms.

In a narrow sense, a blockchain is a type of chain data structure that sequentially combines data blocks in chronological order, and it is a cryptographically immutable and unforgeable distributed ledger. Blockchain[15] technology uses blockchain[16] data structures to verify and store data, uses distributed node consensus algorithms to generate and update data, uses cryptography to ensure the security of data transmission and access, and uses automated scripting. A new type of distributed infrastructure and computing method for smart contracts composed of code to program and manipulate data.

A blockchain system consists of a data layer, network layer, consensus layer, incentive layer, contract layer, and application layer. The data layer encapsulates primary data and basic algorithms such as the underlying data blocks and related data encryption and timestamps; the network layer includes distributed networking mechanisms, data propagation mechanisms, and data verification mechanisms; the consensus layer mainly encapsulates network nodes Various types of consensus algorithms[17]; the incentive layer integrates economic factors into the blockchain technology system, mostly including the issue mechanism and distribution mechanism of financial incentives; the contract layer mostly encapsulates various scripts, algorithms, and smart contracts, and is a blockchain The basis of programmable characteristics; the application layer encapsulates multiple application scenarios[18] and cases of the blockchain. In this model, the chain block structure based on timestamps, the consensus mechanism of distributed nodes, economic incentives based on consensus computing power, and flexible and programmable smart contracts are the most representative innovation points of blockchain technology.

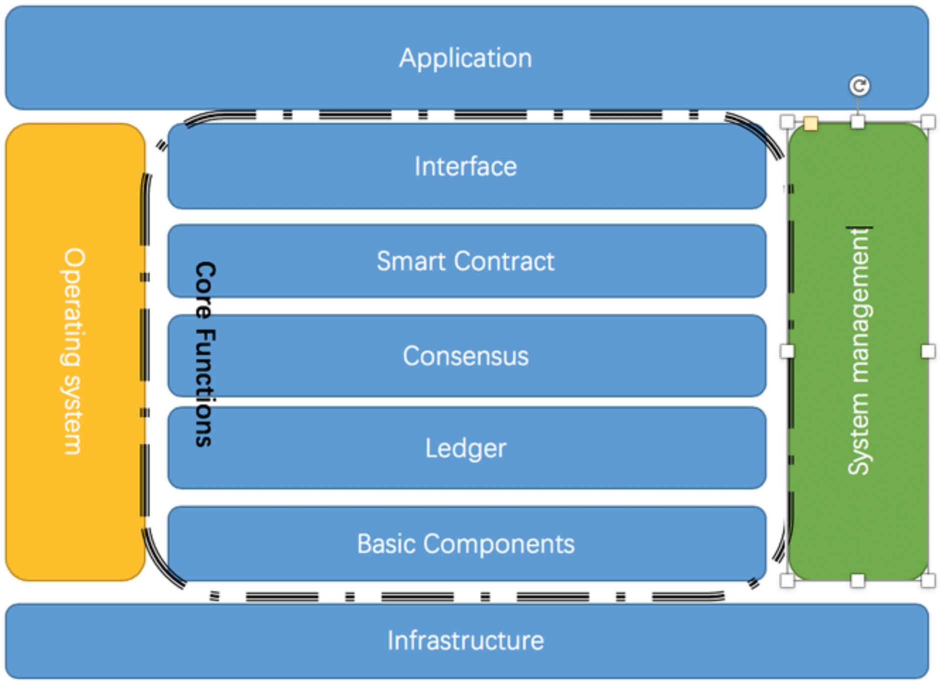


Figure 2: Blockchain Architecture Design

Blockchain mainly solves the problem[19] of trust and security of transactions, so it proposes four technical innovations for this problem:

(1) A distributed ledger[20], that is, transaction accounting is completed by multiple nodes distributed in different places, and each node records a complete account, so they can participate in monitoring the legality of the transaction, and can also jointly Its testimony. Different from the traditional distributed storage, the uniqueness of the distributed storage of the blockchain is mainly reflected in two aspects: First, each node of the blockchain stores

complete data by the blockchain structure. Traditional distributed storage generally, data is divided into multiple shares for storage according to specific rules. Second, the storage of each node of the blockchain is independent and equal. Relying on a consensus mechanism to ensure the consistency of storage, traditional distributed storage generally synchronizes data to other backup nodes through the central node. No Single node can record ledger data separately, thereby avoiding the possibility of a Single bookkeeper being controlled or bribed to keep false accounts. Because there are enough accounting nodes, theoretically speaking, unless all nodes are destroyed, the accounts will not be lost, thereby ensuring the security of the account data.

(2) Asymmetric encryption and authorization technology. The transaction information stored on the blockchain is public, but the account identification information is highly encrypted and can only be accessed with the authorization of the data owner, thereby ensuring the data's integrity. Security and personal privacy.

(3) The consensus mechanism is how to reach a consensus among all bookkeeping nodes to determine the validity of a record. This is both a means of identification and a method of preventing tampering. The blockchain proposes four different consensus mechanisms that are suitable for different application scenarios and strike a balance between efficiency and security.

(4) Smart contracts. Smart contracts are based on these credible, tamper-resistant data and can automatically execute some predefined rules and terms. Take insurance as an example. If everyone's information (including medical information and information about risk occurrence) is real and credible, it is easy to automate claims in some standardized insurance products.

2.2 Multi-Chain Architecture Status

U2U Trust employs the Ethereum Sidechain for its primary Blockchain Network. The main goal of blockchain technology has always been decentralization and security. The main obstacle now for scalability is the fact that every node in network processes every transaction. Hence to improve the scalability, the implementation of the Sidechain is introduced. Sidechains are decentralized, peer-to-peer networks that provide useful enhancements (such as security, risk, and performance) for global systems of value exchange that do not need any other third parties. They enable developers to develop new applications without a risk safely[21].

Key points of the U2U TrustChain network layer are listed as follows:

1. Fully Compatible with Ethereum Protocol with Plasma support: 7 seconds Block Time with 8M Gas Limit allows larger and faster transactions compared to the ETH Mainnet.

2. PoA Consensus: The Proof of Authority requires authorized validators that need computational resources hence no mining.

3. Identifiable DApps: The Applications built on the U2U Infrastructure would be identifiable as verified by the foundation from a regulation perspective.

4. Governance: The validators control addition or Removal of validators in the Network by the rule of N/2 + 1 (where N is the total number of validators).

5. Security: The Authorized Validators are incentivized with U2U Trust Tokens to provide an up-to-date and secure node on the network. The on-chain voting also eliminates any compromised node in the system.

Sidechain is a blockchain that runs in parallel to the main blockchain, which extends functionality through interoperable blockchain networks allowing a decentralized way of transferring/synchronizing your tokens between the two chains. In other words, one can move their cryptocurrency to the sidechain and then back to the main chain. This allows existing tokens from one blockchain (ETH Mainnet) to be securely used within a completely separate blockchain (U2U TrustChain) but still moved back to the original chain if necessary. This helps integrate various third-party DApps existing on the ETH Mainnet to leverage the U2U Trust integration and offer their services with more Trust. Plasma is a proposed framework for incentivized and enforced the execution of smart contracts, which is scalable to a significant amount of state updates per second (potentially billions), enabling the blockchain to be able to represent a substantial amount of decentralized financial applications worldwide.

The Applications on the U2U TrustChain would be validated and identifiable for regulations and compliance with the banking and insurance industry. Decentralized Identity: Decentralized identity services and business entities would be the primary integration strategy to verify the users. Decentralized Data Storage: U2U Trust DApp would be using IPFS to store the DApps Data and related documents.

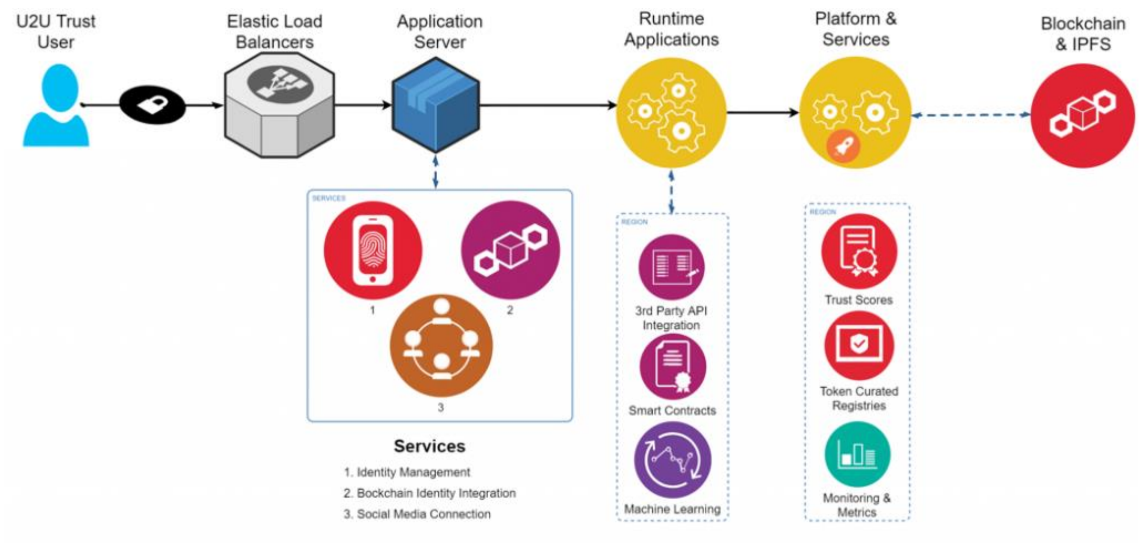


Figure 3: U2U Multi-Chain Architecture Diagram

Trust Score – Logic to create the score

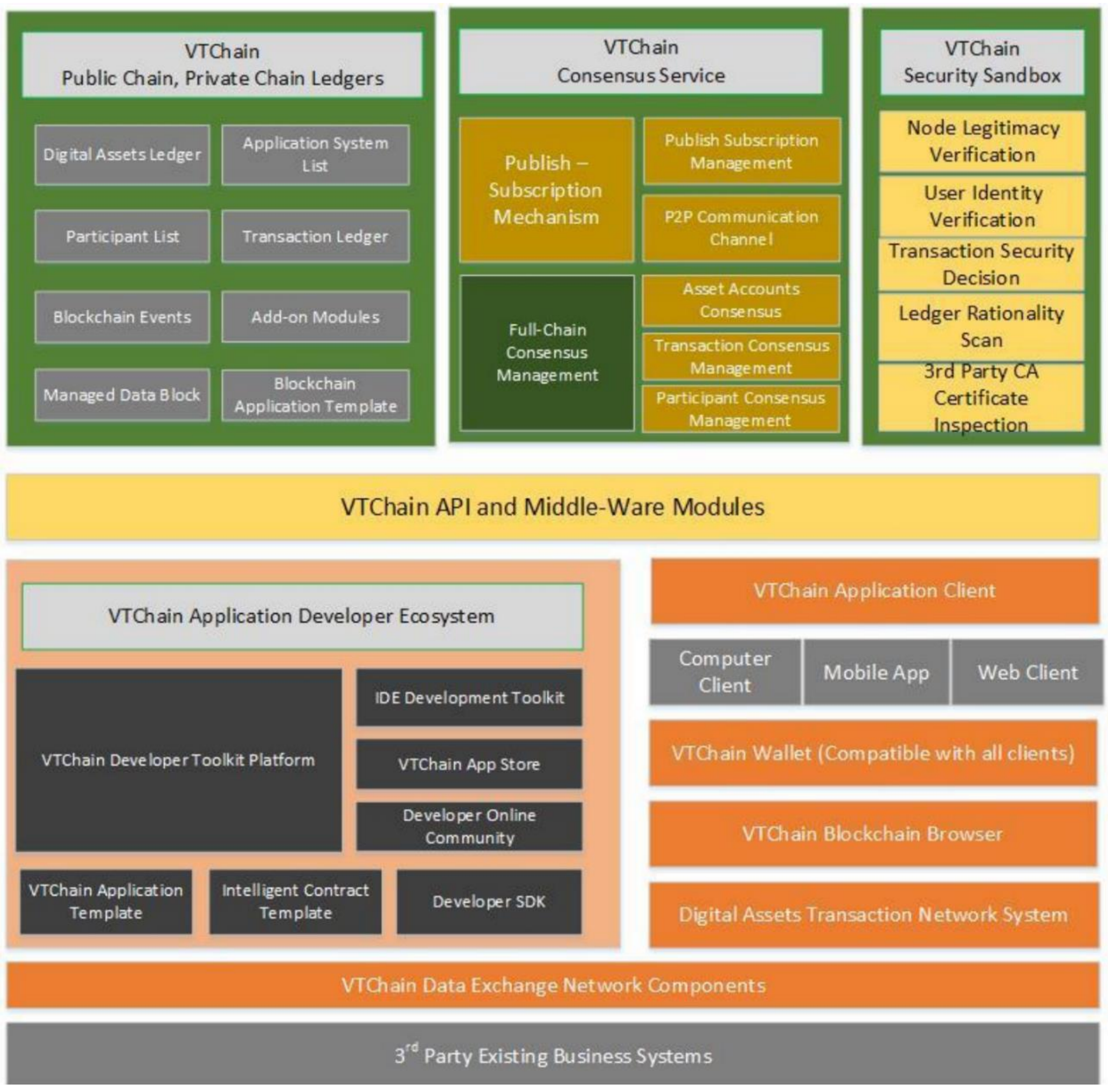
1. Finance – Experian, Equifax, Dun & Bradstreet etc. inspired analytical approach
2. Social – Social Index acquired from linked social media accounts
3. Peers/Co-workers – Peer verification within the same Employment/Geographical area
4. Friends/Family – Authentication & Verification via Friends or Family
5. 3rd party Business Entities

Holders of U2U would be able to suggest functionality upgrades, features integration, and even curate a list of trusted businesses around the world. The curation will be powered by TCRs (Token Curated Registries), which has decentralized voting to add participants (corporations, companies, NGOs.) to the trusted list. This helps create decentralized business rankings without any centralization. As a user of the U2U Trust platform, the motivation is to maintain an easily accessible, high-quality list to attract list applicants who wish to add data to the list.

VTChain is a distributed, production-grade open ecosystem that builds enterprise-level blockchain applications. It is committed to promoting the close connection between blockchain technology and enterprise-level product applications, making full use of the advantages of blockchain technology, and solving application system practices. Centralized systems are becoming increasingly apparent in terms of cost and security issues. VTChain adopts a model of 1 + N Multi-chain structure, a combination of the static ledger and dynamic storage technology, polymorphic nodes, and a variety of consensus mechanisms. Based on the requirements of business application systems, it provides a domestic IDE development environment to build new applications for blockchain 3.0. Ecology. The hot climate of the blockchain has brought the prosperity of the application ecology, and most DAPP applications are facing the same problem: the existing blockchain functions and performance cannot meet the needs of high-concurrency, large-scale cluster applications.

The core nature of VTChain's 1 + N Multi-chain structure is a public chain + N sub-chains. Public chains and sub-chains belong to business logic and data partition processing and are not a distinction between public and private chains that are physically isolated. There is only one public chain, and theoretically, there can be numerous sub-chains, and each sub-chain can run one or more DAPP systems. The sub-chain is similar to the new sharing technology introduced by Ethereum V God and supports multiple transactions in parallel processing. After the transaction is completed, it is asynchronously written into the public chain transaction ledger. DAG (Directed Acyclic Graph) will be one of the essential technologies of VTChain's underlying data structure. DAG changes the technical bottlenecks and defects of the traditional blockchain chain structure, which can effectively improve the speed of transaction confirmation and reduce the resources occupied by the public chain ledger.

Figure 4, VTChain adopts the method of Chain structure + DAG structure compatibility, which is suitable for the public and sub-chains. According to the design requirements of Multi-chain, CSL ledger, and IPFS technology, the use of polymorphic nodes in the VTChain network can effectively complete block transaction consensus quickly and reduce costs. The polymorphic node mechanism divides the nodes of the VTChain network into consensus nodes, data nodes, and ordinary nodes. The consensus and data nodes can be set according to the wishes of the miners, choosing one or both. Ordinary nodes are used as light nodes for end users. The implementation of ordinary nodes eliminates the need for end-users to care about VTChain's full ledger data, but only through the publish-subscribe mechanism, using the DAPP application channel that they care about, to thoroughly complete the "what you see is what you get, ready to use" a goal without synchronizing block data. Multi-consensus parallel mechanism and witness strategy



VTChain Public Chain, Private Chain Ledgers
Digital Assets Ledger
Application System List
Participant List
Transaction Ledger
Blockchain Events
Add-on Modules
Managed Data Block
Blockchain Application Template
VTChain Consensus Service
Publish – Subscription Mechanism
Publish Subscription Management
P2P Communication Channel
Full-Chain Consensus Management
Asset Accounts Consensus
Transaction Consensus Management
Participant Consensus Management
VTChain Security Sandbox
Node Legitimacy Verification
User Identity Verification
Transaction Security Decision
Ledger Rationality Scan
3rd Party CA Certificate Inspection
VTChain API and Middle-Ware Modules
VTChain Application Developer Ecosystem
VTChain Developer Toolkit Platform
IDE Development Toolkit
VTChain App Store
Developer Online Community
VTChain Application Template
Intelligent Contract Template
Developer SDK
VTChain Application Client
Computer Client
Mobile App
Web Client
VTChain Wallet (Compatible with all clients)
VTChain Blockchain Browser
Digital Assets Transaction Network System
VTChain Data Exchange Network Components
3rd Party Existing Business Systems

Figure 4: VTchain Network Architecture Diagram

Aiming at the Multi-chain structure, VTChain uses a Multi-consensus parallel mechanism to increase the speed of transaction confirmation quickly. For the asset chain of the public chain, POW (Proof of Work) or PBFT (Practical Byzantine Agreement) can be used to improve the enthusiasm and contribution incentive of the primary network provider (miner). For the parallel transactions in the sub-chain, the DPOS mechanism is used to complete the transaction confirmation and business execution, which can avoid high transaction costs and quickly reach the DAPP application request submission. Each consensus node in the sub-chain may be randomly selected as a witness to confirm the transaction.

2.3 Existing Blockchain Framework

When we hear the word "Ethereum," we typically associate it with a cryptocurrency – like Bitcoin. While that definition is not entirely incorrect – it is essential to understand that Ethereum is much more than just a pure cryptocurrency. It is an open software platform built on blockchain technology that enables developers to build and deploy decentralized applications.Within the Ethereum platform, is a cryptocurrency called ether that is used to power applications built on the Ethereum blockchain.

The advantages of the Ethereum network are as follows: First and foremost, Ethereu m[22] offers far more functionality than Bitcoin, which makes it attractive to developers f or a variety of reasons. Ethereum is turing complete and stateful, which means that these smart contracts pick up where they left off, which is unlike the Bitcoin blockchain. The fact that the Ethereum blockchain can remember data makes it popular among developers. While Bitcoin transactions are Multi-signature transactions that only execute when two p arties or more consent to it, Ethereum, on the other hand, allows anyone to run programs and applications on top of it, developers can run decentralized code and create new pro ducts or even a new cryptocurrency on top of Ethereum. We see this play out with the I nitial Coin Offerings (ICO's), which are creating new products on top of the Ethereum bl ockchain, and issuing their very own tokens, or cryptocurrencies. Ethereum is both a plat form and an ecosystem, as opposed to Bitcoin, which primarily only runs bitcoin.

Like Bitcoin, Ethereum is open-source. Every node in the blockchain stores the same data. Because anybody can build applications and products on top of the Ethereum blockchain—it is the most robust value proposition—Their contracts can use Ether as a currency, and they can create and issue their currencies.

With smart contract executions, every program executed on the blockchain is by a signed application, and no third party or middle-man is needed. Smart contract executions on the blockchain remove third-parties and middle-men. So that makes it 'trustless,' which makes it even more **trusted**. However, that contract data sits "on-chain" with Ethereum, meaning that the program is stored in the Ethereum blockchain itself, rather than just pointing to code that is sitting somewhere else.

The disadvantages of the Ethereum network include the following: While speed is an issue for most blockchains, Ethereum's speed is particularly important because of the many programs that are trying to run on top of it. Speed and capacity become essential issues. Because Ethereum calculates every Single transaction on any DApp, the more transactions that run on the blockchain will eventually slow it down and prevent it from scaling. That is precisely what we are seeing play out now. To date, I believe the only real application we have seen commercially deployed on Ethereum thus far is CryptoKitties, which took down the Ethereum blockchain with less than 100K users. Because Ethereum is open-sourced—anyone can build a program on top of Ethereum—it is excellent for democratizing access to the blockchain, but the downside is that each new node will continue to work longer and longer which affects the speed of the software that makes it challenging to scale.

There are no fully functioning blockchains to date that can offer real decentralized apps and run faster than Ethereum. Some blockchains may compete with Ethereum like NEO & Stellar, and other post-blockchain protocols like Hashgraph and Dag that offer faster transactions per second, but to knowledge, they may not have much on-chain smart contract usability

First, understand the following keywords of Hyperledger Fabric[23]. Channel is a data isolation mechanism to ensure that transaction information is visible only to transaction participants. Each channel is an independent blockchain, which allows multiple users to share the same blockchain system without worrying about information leakage. Chain code Also called a smart contract, it encapsulates asset definition and asset processing logic into an interface. When a user requests it, it changes the state of the ledger.Ledger: Blockchain ledger, which holds transaction information and smart contract codes.Network: A P2P network between transaction processing nodes, used to maintain the consistency of the blockchain ledger.Ordering service: uses Kafka, SBTF, and other consensus algorithms to sort and package all transaction information into blocks, sends them to the committing peer's node, and writes them into the blockchain.World state: Shows the current state of asset data. The bottom layer organizes asset information in the blockchain through the LevelDB and CouchDB databases, providing an efficient data access interface.The membership service provider (MSP): Manages authentication information and provides authorization services for clients and peers.

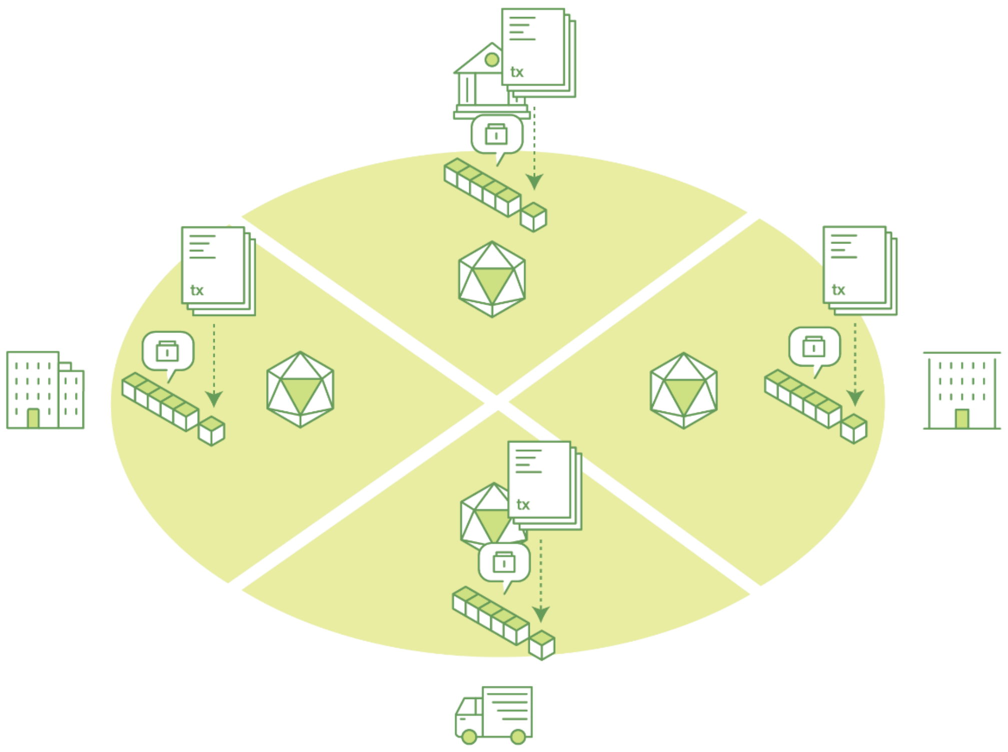


Figure 5: Hyperledger Fabric Blockchain Distributed Ledger Diagram

The core concept of the blockchain[24] is a distributed ledger. As shown in the figure below, the same ledger is available on any node (excluding the client). Therefore, the advantage is that the data is difficult to forge, and legal liability can be investigated through retrospective records after fraud. The disadvantage is a colossal waste. All data related to traditional services should be as small as possible. Even if three copies are stored, many anomalies are taken into account, and service availability has reached its limit. However, this characteristic of the blockchain also causes another problem that the ledger cannot be too giant, at least it cannot exceed the storage and processing capacity of the smallest node in the blockchain network. Therefore, this limits the number of total transaction data and also affects the size of Single transaction data that can be written into the blockchain.

It is worth adding that Hyperledger Fabric, the most popular development tool at present, releases different versions according to the situation, but the principle is the same. The bottom layer of the blockchain uses the same data structure. The functional modules are slightly different between different versions. We can see the following four critical fields in a block:

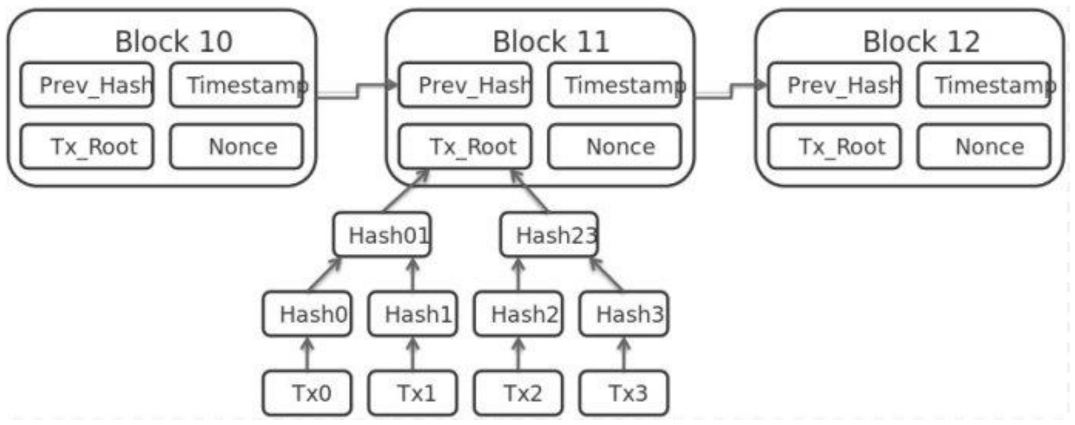


Figure 6: Hyperledger Fabric Blockchain Data Structure

The hash value of the previous block is used to connect the previous block, the previous block also has this field, and the previous block can also be connected. Forming a chain is also the meaning of the blockchain.Timestamp: Standard time, through time sequence, allows transactions to be traced through the time dimension.Nonce : Random numbers, when talking about random numbers, we must talk about another critical field in the block, "difficulty value." The difficulty value is the mining standard. Random numbers reflect the mining process. The hash value of the generated block satisfies the defined "difficulty value."Tx_Root: Meck tree, a summary hash of all transactions. From the picture, we can see that each transaction has a hash value, and every two adjacent hash values will generate a hash until the top hash value is generated.

Smart contracts[25], also called chain codes, are a piece of code that runs on the blockchain. It is middleware that interacts with the bottom layer of the application system. Various complex applications can be achieved through smart contracts[26]. Currently, in addition to using Go to develop smart contracts, you can also use Java and node.js to develop. The best support is Go. The smart contract is installed on a node (Peer), runs in a Docker container, and performs data interaction with Peer through GRPC. The interaction steps are as follows:

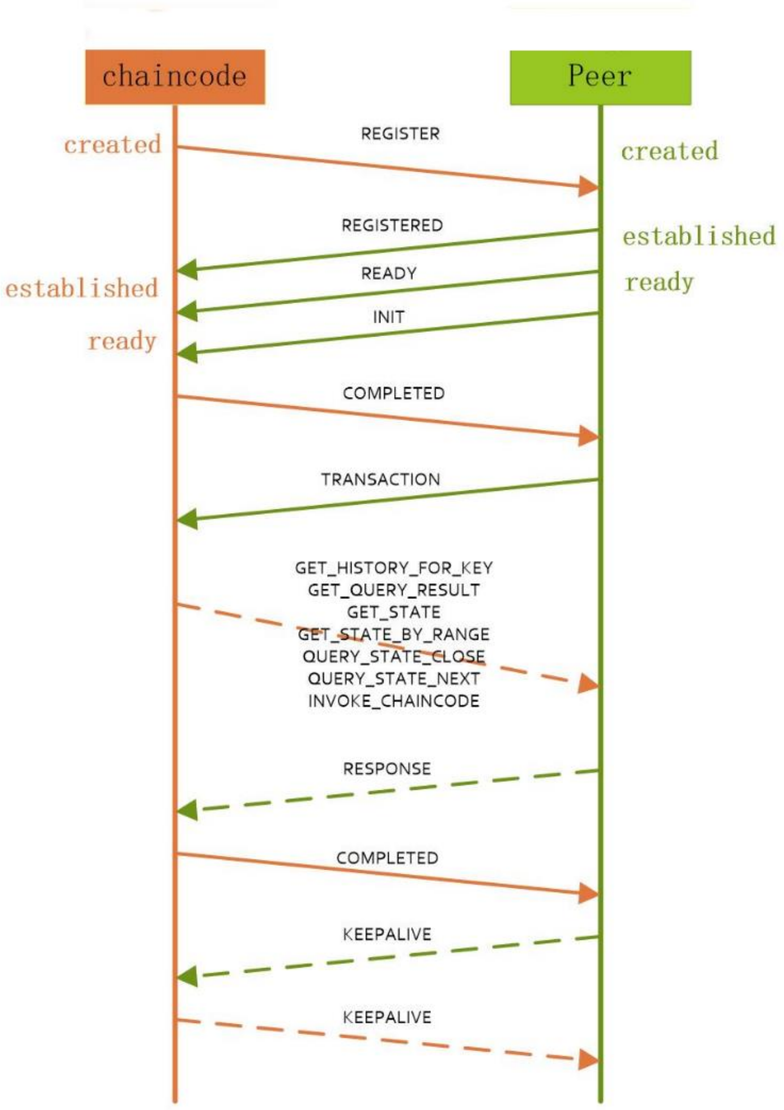


Figure 7: Hyperledger Fabric Chain Code Interacts with Node Messages

1) After the ChainCode calls the shim. Start () method, and it sends a ChaincodeMessage_REGISTER message to the peer to try to register. The client chain code is waiting to

receive messages from the node (Peer). At this time, the status of the chain code and the peers are initially created.

2) After receiving the ChaincodeMessage_REGISTER message from the chain code container, the node (Peer) will register to a local handle structure and return a ChaincodeMessage_REGISTERED message to the chain code container. Furthermore, the update status is established, and then a ChaincodeMessage_READY message is automatically sent to the chain code, and the update status is ready.

3) After receiving the ChaincodeMessage_REGISTERED message, the chain code completes the registration step without performing any operations. The update status is established. Update the status to ready after receiving the ChaincodeMessage_READY message.

4) The node (Peer) sends a ChaincodeMessage_INIT message to the chain code container to initialize the chain code.

5) After receiving the ChaincodeMessage_INIT message, the chain code container calls the user code Init () method for initialization. After success, it returns a ChaincodeMessage_COMPLETED message. At this point, the chain code container can be called.

6) When the chain code is called, the peer sends a ChaincodeMessage_TRANSACTION message to the chain code.

7) After receiving the ChaincodeMessage_TRANSACTION message, the chain code will call the Invoke () method. According to the user's logic in the Invoke method, the following message can be sent to the node (Peer):

8) After receiving these messages, the node (Peer) will process them accordingly and reply to the ChaincodeMessage_RESPONSE message. Finally, the chain code (chain code) will respond to the completed message ChaincodeMessage_COMPLETE to the peer (Peer).

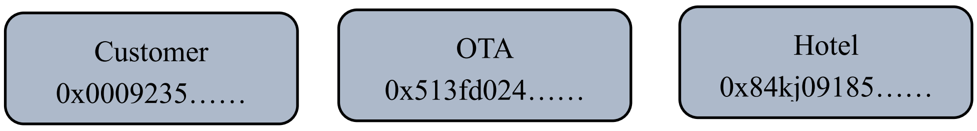
9) After the above message interaction process is completed, the node (Peer) and chain code (chain code) will periodically send ChaincodeMessage_KEEPALIVE messages to each other to ensure that each other is online.

2.4 Existing Blockchain Technology in Tourism

At present, there are many private Single-chain blockchains in the tourism industry. They have designed corresponding functions based on the needs of the tourism market, resulting in different tourism blockchain[27] networks, mostly from architecture and development to Object to achieve. Since the blockchain technology is in its early stages, the complexity of the travel market is far higher than the performance of the blockchain.

Winding Tree Foundation is a non-profit organization incorporated in Switzerland. Its purpose is to develop software projects, data exchange standards, and infrastructure to advance the travel industry. In the WindingTree network architecture, identity design is the key to the network. The market is formed by individual groups buying and selling goods. Since the traveler blockchain uses Ethereum, Ethereum accounts represent these individual participants. The Ethereum account is used as the market participant ID to ensure uniqueness and transaction payments can also be made.

Then the ID of the tourist blockchain can use the Ethereum account ID, for example:



Ethereum account is a simple structure; it has no other properties except ETH balance, and it can do nothing but sign messages and send transactions to other accounts and smart contracts. We can use smart contracts to represent the organizations. Unlike Ethereum accounts, smart contracts do not have private keys but can contain data and execute code.

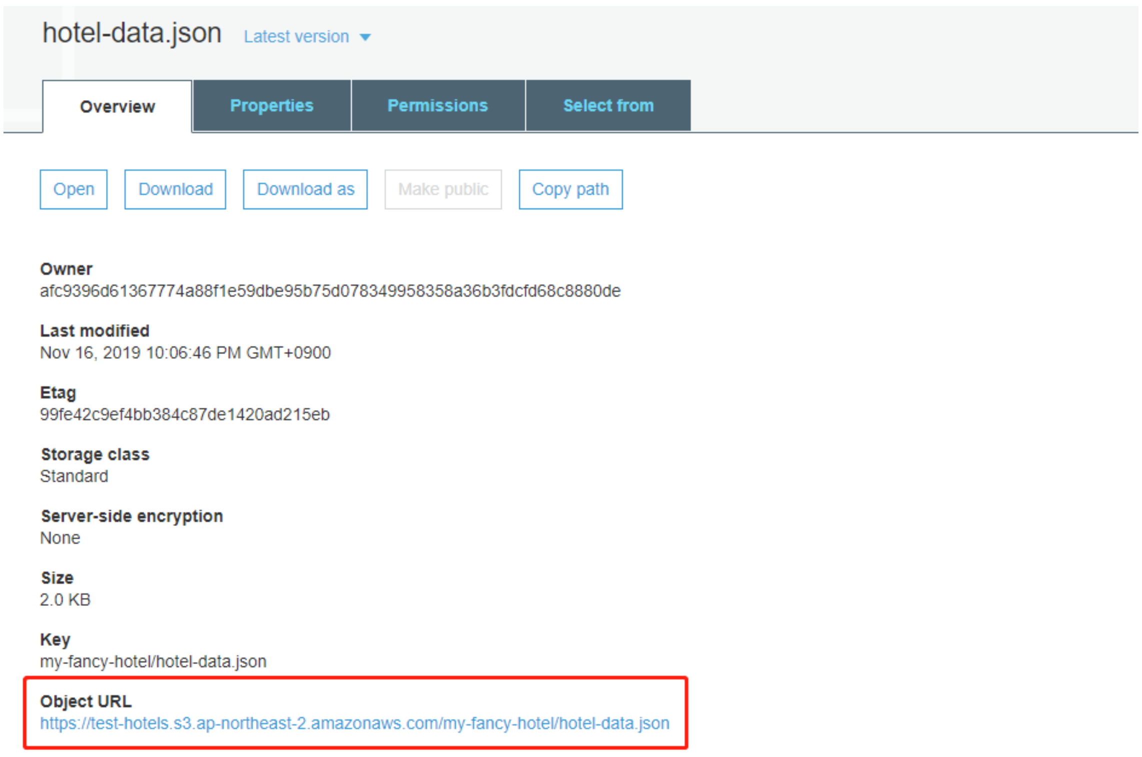
When the consumer registers, it is the behavior of the individual. It is easy to use the Ethereum account as the user ID, but as a service provider (Hotel), two individuals are corporate legal persons and legal representatives. In this case, to establish a relationship between the company's authorized person and the legal representative, we introduce the ORG.ID concept.ORG.ID consists of the following two points: 0xORG Smart contract interface ORG.JSON, A file (name, address.) containing organization (hotel) information.

The 0xORG function is essential for business systems and networks: it provides a unique identifier for each organization (For example,0xc98c5g1r5h1rfd12512cde4h84r, this address is also called ORG.ID). It is obtained by processing the content of the JSON file through a hash algorithm. Since the address in the Ethereum network starts with 0x, we add 0x to the hash value to form the Ethereum address. For example, hotel information cannot be stored on the blockchain. We will store it off-chain and point it to the ORG.JSON location (because storing large amounts of data on the blockchain is too expensive, please store it off-chain). Track ORG, JSON changes with each other (0xORG must contain a valid ORG.JSON hash. Otherwise, it means that the identity has been destroyed) By tracking each other, the organization can verify the storage keys (association keys) of the identity of other companies on the platform.

From the above description, we apply the entire file to the actual development environment. The data is stored off-chain so that the information can be queried through ORG.JSON, and the file is stored in the distributed system and encrypted by the hash algorithm, which ultimately guarantees the security of the information. Edit the service provider information into a JSON file and publish it to an external server. After preparing the JSON file, we will store the file in a network location that is easy to access. We store the file in the AWS (S3) file system. (Ensure that the services you use have appropriate access controls.)

Distributed file storage is an excellent solution to off-chain storage. It will not be necessary to store data on the chain and data with extensive information capacity. Off-chain storage will significantly increase the processing speed and operating speed of the network. The off-chain storage device selected below is the S3 bucket of AWS. The specific operation method will be described in detail below。

Figure 8, Get the storage URL of the JSON file (orgJsonUri): https://test-hotels.s3.ap-northeast-2.amazonaws.com/my-fancy-hotel/hotel-data.json. To prevent fraudulent hotels, we will adopt the method of synchronizing hotel information. We utilize the hash encryption of hotel information, which is to encrypt the JSON data into a hash value (orgJsonHash) using the keccak256 encryption method, and orgJsonHash and orgJsonUri jointly publish to the district. On the blockchain, whenever the hotel information is updated, the hash value needs to be updated, which can effectively reduce the hotel fraud.


hotel-data.json Latest version

Owner
afc9396d61367774a88f1e59dbe95b75d078349958358a36b3fdcfd68c8880de

Last modified
Nov 16, 2019 10:06:46 PM GMT+0900

Etag
99fe42c9ef4bb384c87de1420ad215eb

Storage class
Standard

Server-side encryption
None

Size
2.0 KB

Key
my-fancy-hotel/hotel-data.json

Object URL
https://test-hotels.s3.ap-northeast-2.amazonaws.com/my-fancy-hotel/hotel-data.json

Figure 8: Company Information Off-Chain Storage

Tripio is a decentralized tourism service market platform based on blockchain. Directly linking global travel service providers (companies or individuals) and consumers through the decentralized blockchain network and using travel accommodation booking as the starting point to build future travel based on trust, incentives, and zero commission service status. Tripio's goal is to reconstruct the incentive mechanism and credit mechanism of the tourism industry, reduce economic costs, and enable users to harvest better services at lower prices.

Technology model: Tripio project: smart contract development based on Ethereum. Based on Ethereum's built-in Turing complete virtual machine technology, Tripio redefines the transaction mode and state transition function rules and constructs various smart contracts in the travel service. As a fully redundant distributed system, Ethereum has limitations such as calculating costs and not being able to save significant amounts of data. Tripio will build a complete and usable system by combining on-chain and off-chain. For example, the content of the files showing the pictures and comments will be stored off-chain through the IPFS distributed system, and a hash string will be created for the connection to the smart contract.

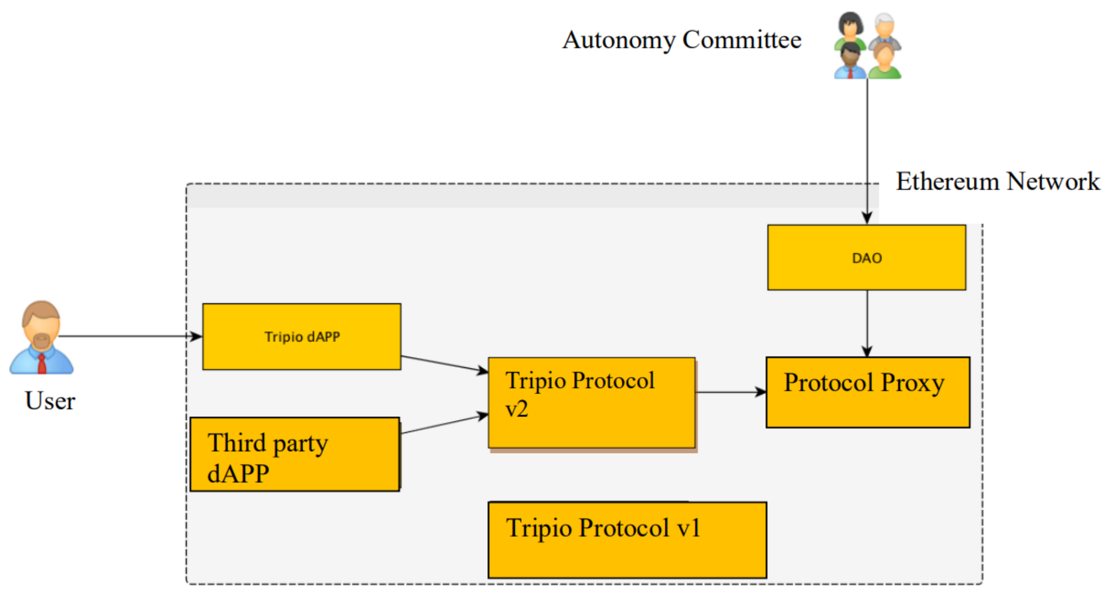


Figure 9: Upgrading Tripio Application in the way of Community Autonomy

Platform architecture: Tripio's business model is implemented based on the Ethereum's underlying public chain, through smart contract writing services and identity registration systems and payment and transaction systems. Tripio will automatically save the data to the distributed file system supported by the IPFS blockchain and generate a corresponding hash string as the service identification code of the smart contract. Tripio uses the DPOS consensus mechanism to maintain community status. The DPOS mechanism will dynamically establish a security committee for community maintenance based on the time and quantity, trustworthiness, and due diligence of participants in the system.

ZatGo is a family of technology companies engaged in technology development and system development in the tourism industry. It has rich experience in business travel, travel, and air ticket systems. ZatGo introduces blockchain technology and Token payment scenario mode. Trying to build a state-of-the-art decentralized alliance based on the ZatGo payment platform. Technology model: ZatGo will connect different public chains in the application layer for smart contract development. For upper-layer applications, it will shield the differences between different public chains. All of them involve digital identity authentication, digital currency transaction ledger, risk control, and credit information. The logic using evaluation and other technologies will be solved in the ZatGo blockchain unified payment platform, and users can query all recorded data in real-time through the client access. ZatGo will issue ZAT rewards for members who have contributed to the state. These contributions include data contribution, usage contribution, and consumption contribution.

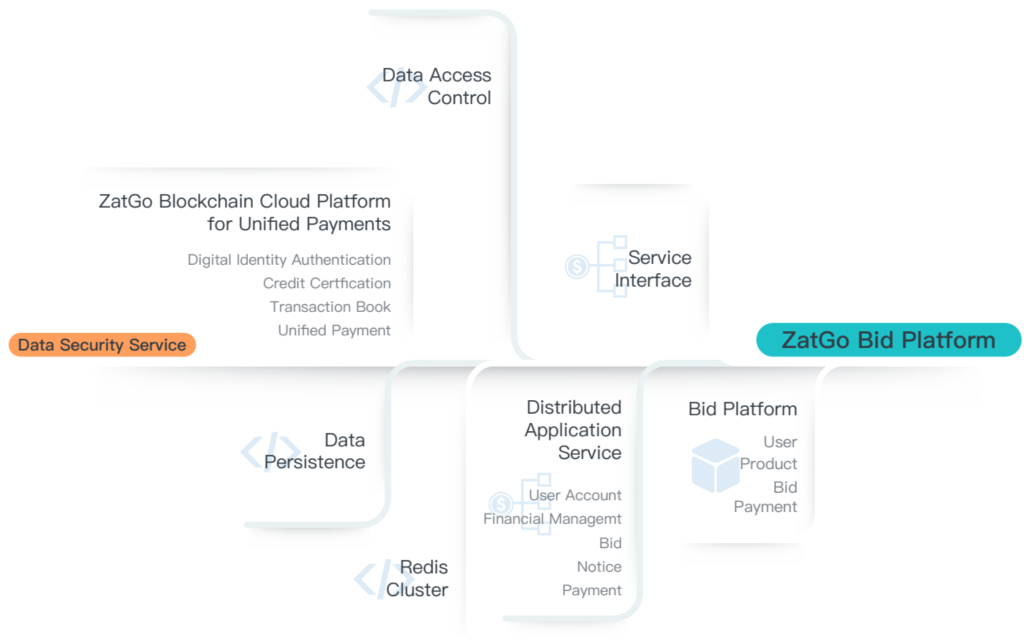


Figure 10: System Architecture Diagram

Platform architecture: ZatGo has built a blockchain-based unified payment model and a business travel service platform for smart applications. Develop the underlying digital currency payment platform to provide digital identity, credit authentication, transaction ledger, and digital currency payment services for upper-level applications. The blockchain layer is connected to Multiple public chains to implement operations such as inquiries and transactions on the chain.

The product service layer implements the client's wallet function and the payment docking of related platforms and realizes the virtual currency payment function of each third-party platform. The business travel cloud platform mainly adopts a high-level architecture: application layer, security gateway layer, application service interface layer, distributed service layer, data access control layer, cache layer, data persistence layer, and application layer.

Chapter3. Multi-Chain Based Interworking Mechanism

This thesis proposes the design of Multi-Chain[28] applications in the industry based on the existing Multi-chain architecture model. Multi-chain architecture has Multiple architecture solutions (side face, cross-chain), but the Parallel chain expansion network is more straightforward, more intuitive, and more powerful. It is not only an application of DApp[29]but directly owns its blockchain ecology. It is more efficient and more straightforward than cross-chain trading. In general, it is more scalable, more efficient, and safer. In the parallel chain, the public chain is used as the main chain, and the private chain is the function extension chain. Fang, Bitcoin) and other technologies. The particular chain network development tools are developed with Hyperledger Fabric as the center.

3.1 Proposed Multi-Chain Based Interworking Architecture

The above figure is designed based on the defects of the Single-chain blockchain in actual business. Through the functional architecture of the main chain, based on the main chain as the core blockchain, the design is based on the principle of public chain technology, and shared information is stored on the main chain. When the main chain is responsible for a Single network task, the network performance is compared to a Single Multi-tasking chain Greatly improved during processing. Other business functions can determine the structure of private or public chains through the selection and determination of the network. Private chains allow Multiple chains to run concurrently. Theoretically, according to the functional requirements of the business, the parallel chain can be expanded infinitely, which also illustrates the scalability of the Multi-chain architecture network. The private chain and the public chain use Application as a medium to pass data to each other through the route return value. For example, when a user sends a login request to the private chain using a client, the private chain verifies the legitimacy of the information and carries the return value (hash value). Return to the client. At the same time, the client sends the return value (hash value) of the private chain to the public chain. After the main chain verifies the information, it sends the return value to the private chain through the client. In this way, through the Application as the central medium, the data is encrypted and processed by the hash algorithm and then routed[30][31], and the main chain and private chain can exchange data.

Aiming at the performance of the network[32] as the starting point, the blockchain applied in the enterprise is used as a reference to design a ping-pong parallel chain that solves privacy and permission control. Based on business functions, it is a chain-chain architecture model.

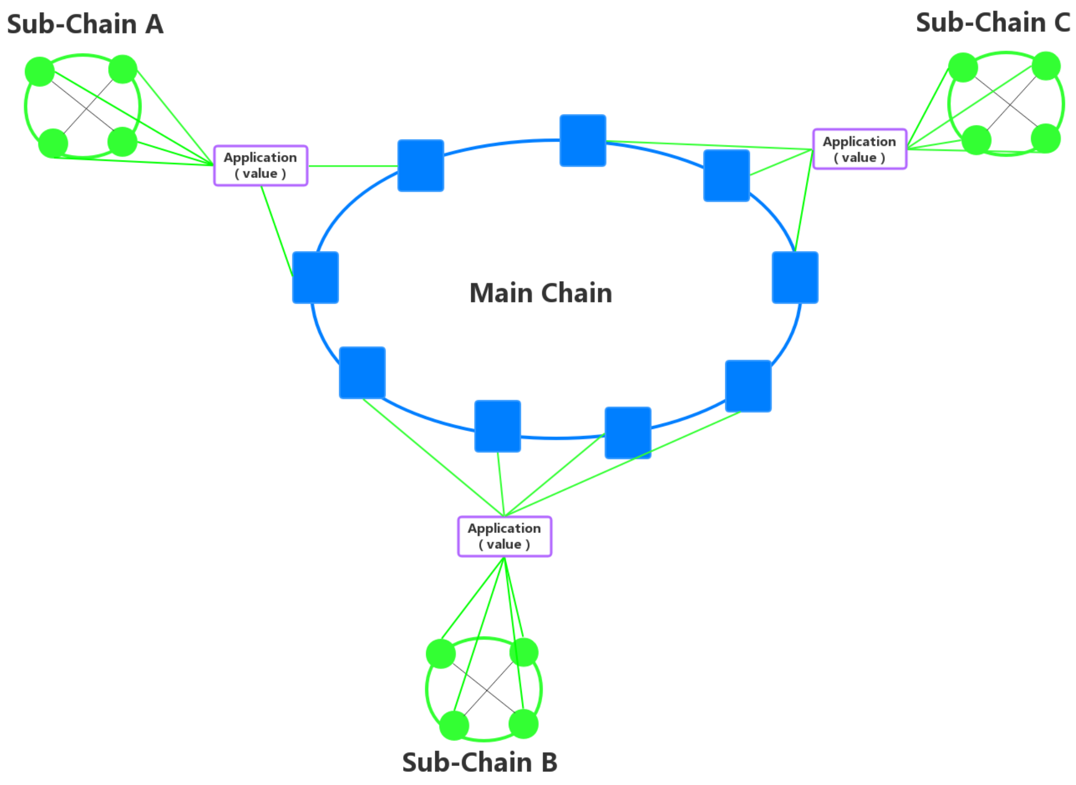


Figure 11: Proposed Multi-Chain Configuration

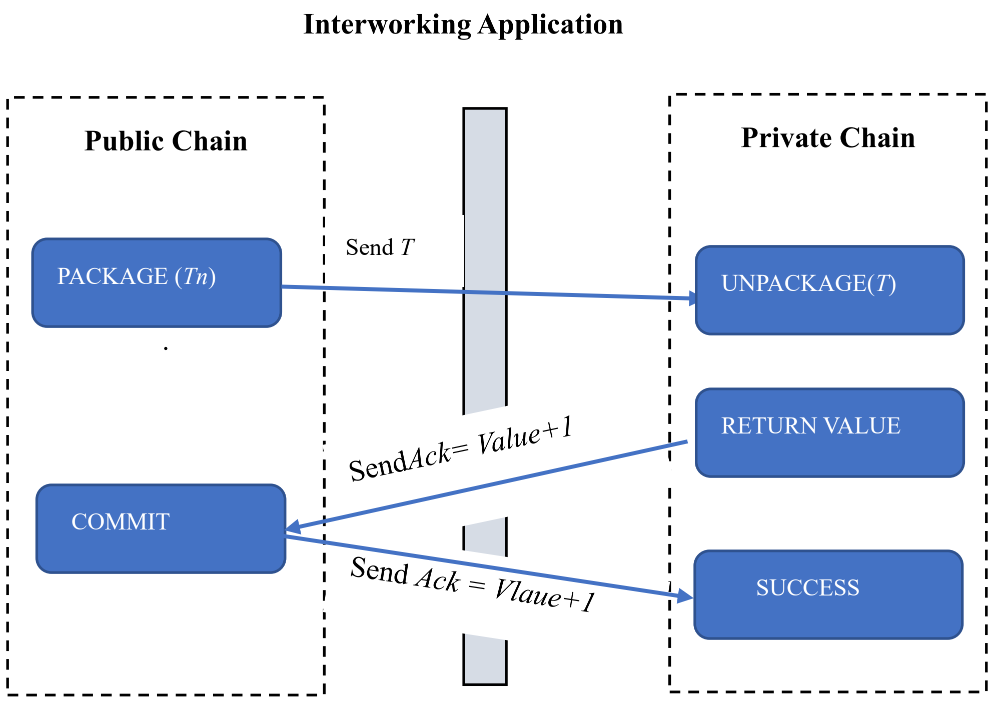


Figure 12: Public and Private Chain Functional Communication Process

Multi-chain architecture enables the public and private chains to communicate through the Application repeater. When there is a request for authenticity on the public chain, the client sends a T function. After the private link is received, it performs verification, consensus, and recording operations, and will confirm. The information is returned to the public chain through Ack[33]. At this time, the public chain operation is successful. At the same time, a prompt is sent to the private chain through Ack to complete the mutual confirmation process. During the process, Ack is equal to 1 to ensure that the peer has received the information. Encrypt the data information, complete the consensus of the peer-to-peer chain, and ensure the security of the information.

The first chapter briefly introduces the Multi-Chain architecture of this thesis to design a Multi-chain architecture through functional classification, select the blockchain according to the required function, and directly extend it to the main chain. The client directly calls the function of the blockchain through API routing. In the actual production environment, under the premise of ensuring performance, low-cost environments are often used for research and development, which leads to confusion in the research and development architecture. The functions that the blockchain is responsible for are defined by integrating APIs, and the verification fields are set to call the front-end API automatically. Way to produce.

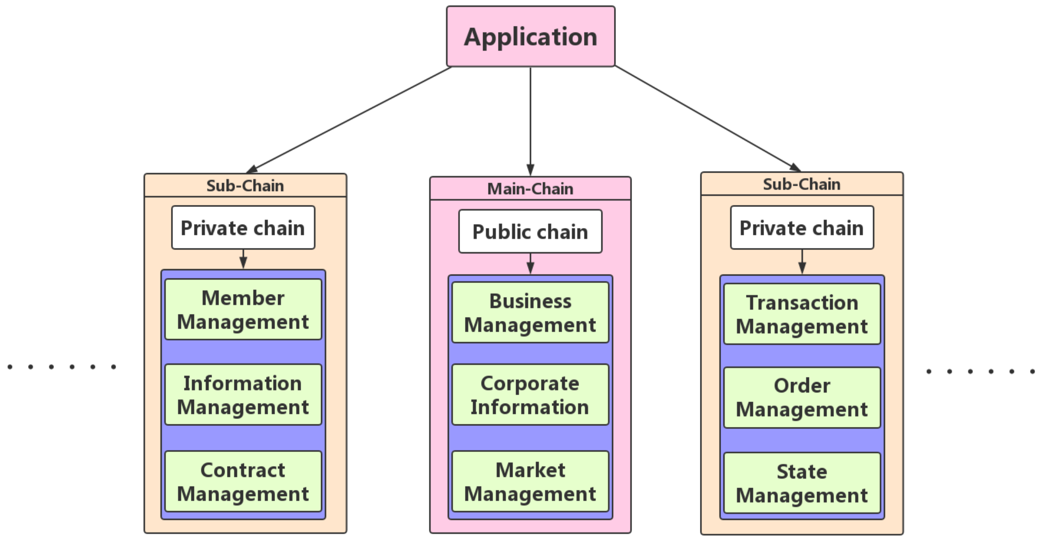


Figure 13: Proposed Multi-Chain-Functional Architecture

The focus of the Multi-chain architecture is on the functional requirements. Building one or two business requirements on the private chain is an essential test for the concurrency pressure on the blockchain network. When facing large-scale projects, the demand for concurrency pressure may Function alone as a private chain to solve concurrency issues. In the above figure, the private chain model is responsible for several business requirements. Through communication with the client, data can be routed to other networks. Passing data passes the value of the hash algorithm through the return value. Of course, this increases the operating pressure on the client. Large, the client needs to continuously pass back the return value to ensure the consensus of other chains, and there are specific requirements for network communication bandwidth to meet the needs of data transmission and to ensure the integrity of the information after mutual agreement. As mentioned above, the routing of the data API is used in architecture. Here, the internal management will be designed in the form of an API, and the business functions will call each other to carry parameters for communication.

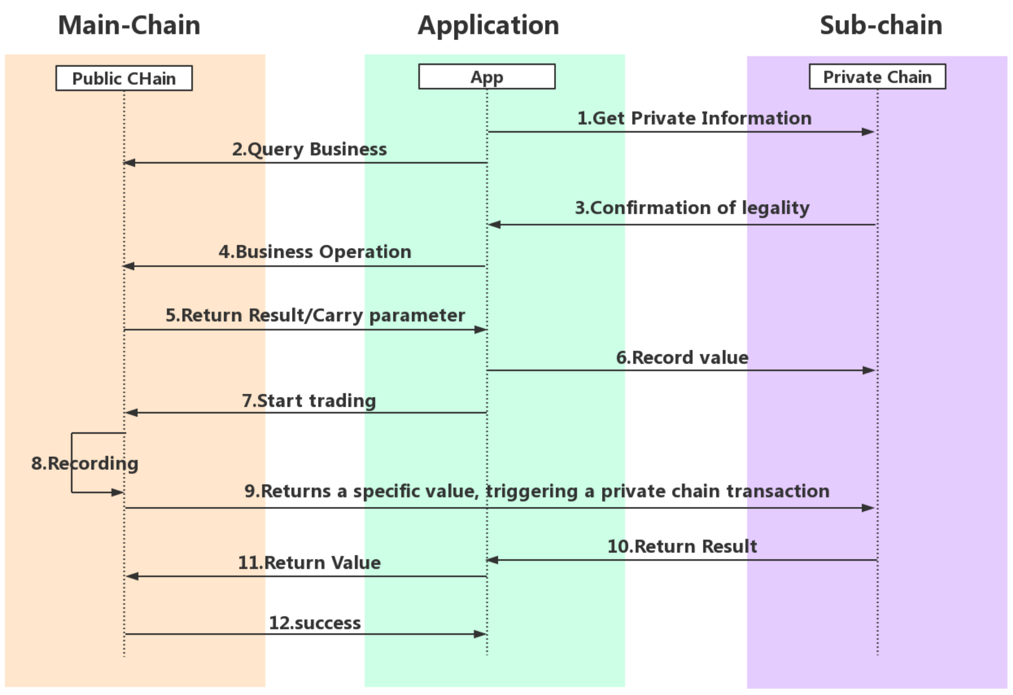


Figure 14: Multichain Flowchart based on Mutual Communication

The above illustrates the flow of Multi-chain architecture by using examples of interaction between enterprises:

1. The client verifies the information on the private chain. This step is usually used to authenticate the user when logging in and confirm the information of the private chain.

2. The application queries and searches the information on the public chain to realize the everyday function of information sharing between enterprises.

3. Query the information and legitimacy of this company to achieve the information verification of the peer-to-peer chain.

4. Transaction, request, and other operations to provide the implementation of requirements for the application.

5. Return the operation result and send it to the private chain for operation.

6. Record public chain information or activate a smart contract.

7. The transaction starts at this time, and the data of the application is used for consensus on the public chain.

8. The public chain records part of the transaction information.

9. The remaining part uses the return value method (Hash encryption) to record on the private chain or execute the smart contract (if a function needs to be executed after successful verification)

10. After the transaction record on the private chain is saved, and the smart contract is completed, the result is returned.

11. The application sends the result (return value) to the public chain for saving, triggering a broadcast.

12. The process of all transactions and information recording is complete. Return the prompt message to the application to display success.

The above describes the architecture of parallel chains in a Multi-chain blockchain and how to trade and control network performance. The Multi-chain architecture model addresses the lack of co-energy and privacy control in practical application environments. At the same time, it also provides excellent support for the expansion of the business environment and adds blockchains according to business needs to address most production needs.

3.2 Interworking Architecture Based on Multi-Chain for Secure Tourism Service

The cross-chain blockchain network designed in this paper uses functions to divide and use different networks. In the design, the cross-chain network can ensure the integrity of network functions. The WindingTree network is responsible for the business system. At this time, this chain is a Single-chain network, but it is only responsible for some functions of the entire tourism blockchain. The Hyperledger Fabric network is responsible for the transaction system part and uses the tools provided by Hyperledger Fabric to develop a function that only executes transactions, which maximizes the utilization of the network. Because this network has only one function, there are fewer smart contracts in the network, the overall performance of the Single-chain is less affected, and the network runs relatively quickly.

As a new network product, the tourism blockchain will replace the traditional tourism operation mode. The travel blockchain connects OTAs directly to travel providers. It is an open-source ecosystem that includes smart contracts, interoperable API standards, and coordinated data structures that enable distributed distribution and discovery of travel products and services. This is an entirely decentralized platform that allows suppliers to showcase their products and services without intermediaries. OTA can also access these products and sell them to customers. It is an open market, a marketplace where buyers and sellers can trade without an intermediary. This is useful for OTAs because they can quickly connect to thousands of vendors, which is suitable for vendors because thousands of OTAs can now access and purchase their products and services. The blockchain network consists of two parts: a service management network (inventory, hotel information.) and a trading network, that is, Winding Tree network and the Hyperledger Fabric network, the two-part design concept strengthens the security of the network blockchain. As mentioned earlier, due to the high cost of storing information in the blockchain, we store essential transaction information separately. The two parts build a network of tourism blockchains.

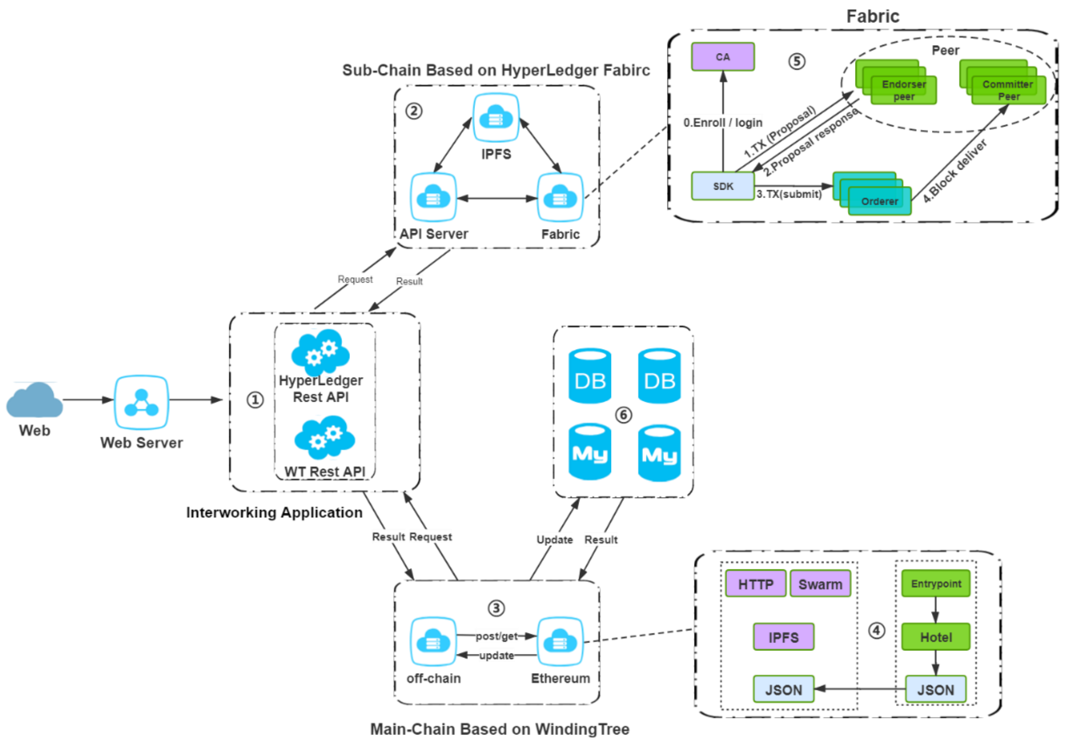


Figure 15: Interworking Architecture Based on Multi-Chain for Secure Tourism Service

1. API Router: Use the API to manipulate the database.
2. Hyperledger Fabrir: Responsible for transactions and permission assignments.
3. WindingTree： Responsible for product inventory, update management.
4. Architectural process of Ethereum and off-chain storage in the WindingTree network
5. Hyperledger Fabric: Security Control and Transaction Centering Control

The process of trading in the actual market is complicated. For example, as a hotel, the sales channels are diverse, and the hotel can be sold on its website so that it is directly traded with the consumer without a complicated booking process. Another channel is the mainstream sales channel in the market. It is sold through travel agencies and OTA platforms (starting now referred to as agents). In this case, agents need to obtain the sales qualification of the hotel first and then carry out related sales. Among them, the WindingTree network proposes rich management APIs for service providers, including WriteAPI, ReadAPI, SerachAPI, ContractAPI, and BookingAPI, which can achieve all the requirements for the management of service providers. Although the Hyperledger Fabric network is only responsible for transaction processing, it is more critical for the network. It completely solves the main problems in the tourism market. Room information (room type, room rate, inventory.).

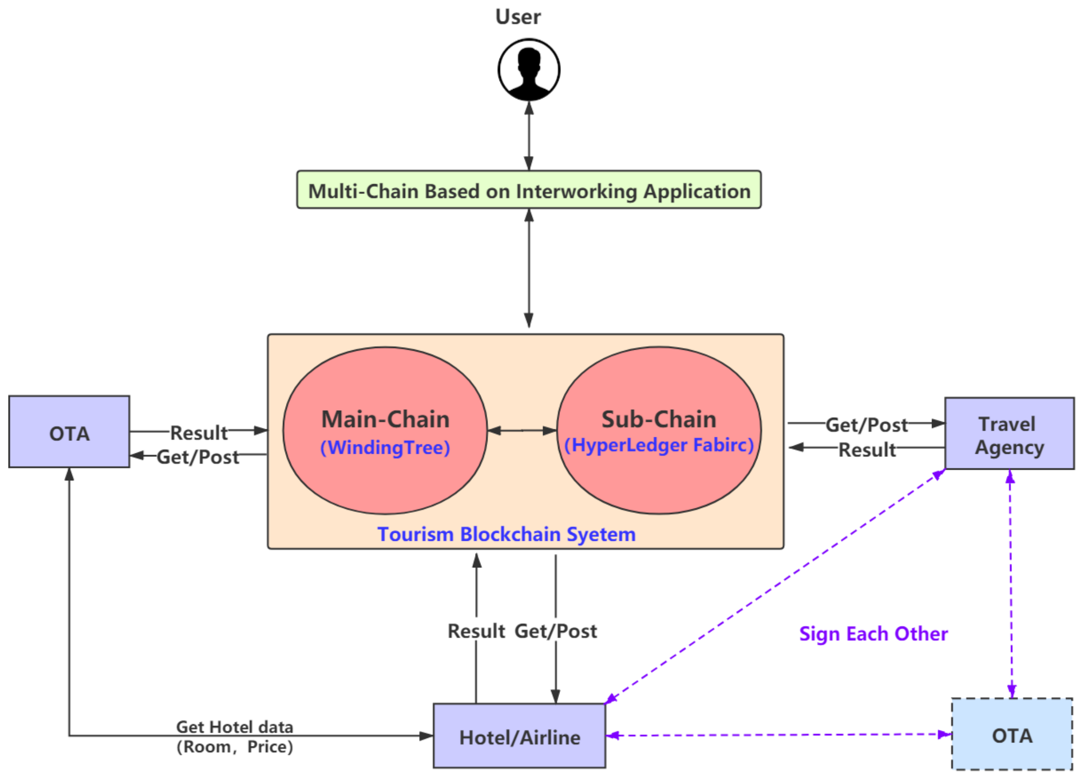


Figure 16: Tourism Commodity Market Relationship Based on Multi-Chain for Secure Tourism Service

The above picture depicts the relationship between service providers in the entire tourism market environment. As a source of goods service, hotel companies need to manage the room management and sales of the hotel. However, the WT network provides management functions for the entire network environment. The hotel can perform hotel sales, inventory, price, planning, and other functions through WT. Hyperledger Fabric will be responsible for OTA, travel agency, hotel, consumer transaction management, and control Multiple transaction processes in the blockchain.

The uncertainty of business scenarios in a market environment is variable. If the above structure can be reused with the increase of distribution organizations, this can ensure that it is reused only at a certain structure point, and will not affect the operation of this business network. In more detail, the mutual call of data is only an increase in the number of structural points, as long as the organization that joins is always classified into our pre-defined organization. The above structure can be reused.

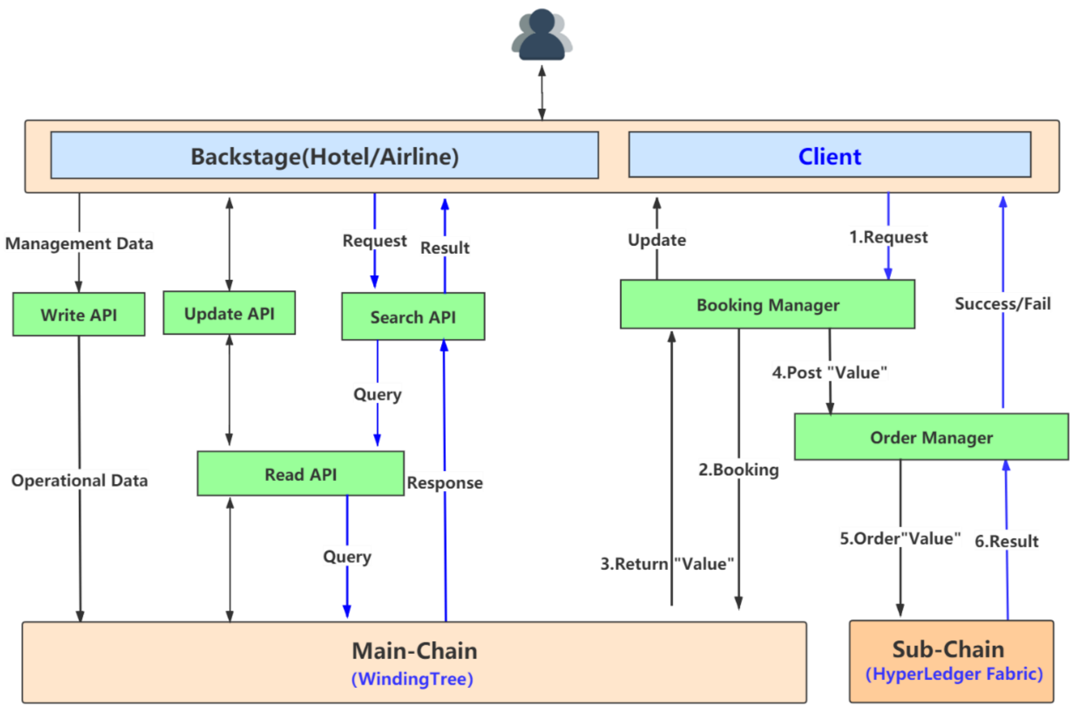


Figure 17: Interworking process Based on Multi-Chain for Secure Tourism Service

Hotel/Airline: Use the WriteAPI tool to manage the inventory of goods, and the operational data goes through the blockchain network so that the inventory records are real and cannot be changed at will. The user uses the Client to query and browse products. After the user searches for goods, the whole process, Winding Tree provides query, inventory status update, and so on. The Hyperledger Fabric records the transaction, and the identity of the user is confirmed. In more detail, the network in the figure above only involves the WindingTree network, only analyzing the entire process of the service provider from the time the service is published to the end of the service. The publishing service is just that the service provider uploads the service to the business network through the API. Generally, WriteAPI and ReadAPI are used to write and update data. The service user searches for services (hotels, air tickets) in the business network through the client. The business network transmits data back to the client, which can reflect the business network's management of the entire network.

Chapter4. Interworking Application based on WindingTree and Hyperledger Fabric

The above two chapters introduce the WingdingTree network and Hyperledger Fabric network, how to design and architecture, and implement. We can know that the two networks can well implement all current services. This chapter will explain and explain the overall architecture of the network, as well as some problems encountered.

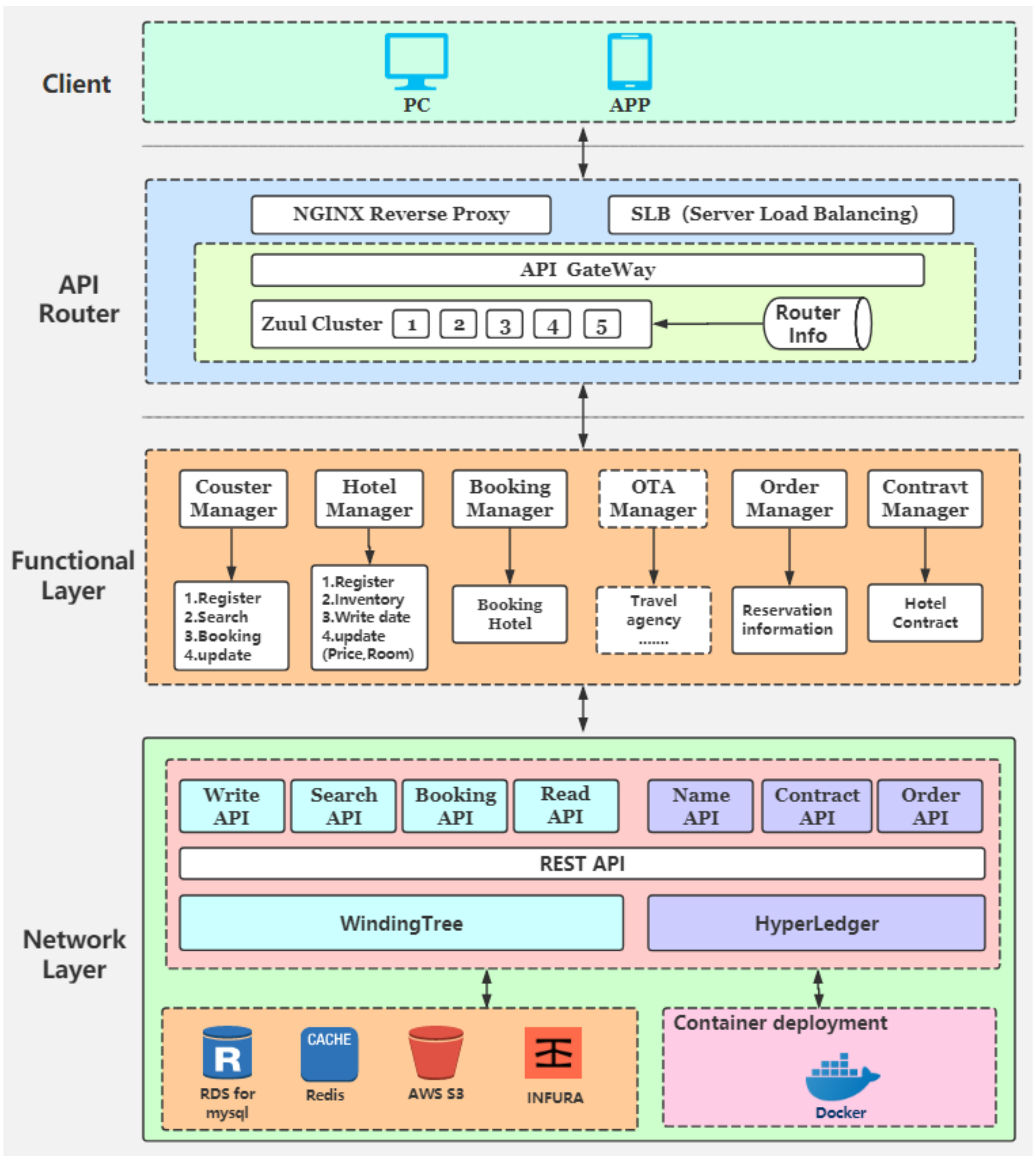


Figure 18: Network Layered Architecture Interworking Architecture Based on Multi-Chain for Secure Tourism Service

This figure describes the entire network architecture, and the client routes to the corresponding function management center through the server API. Each function management center performs its task. When receiving a request, the manager sends a request to the target network. In the Functional Layer, the manager contains specific functions. The request will call the corresponding function to use the API here, and the Rest API uniformly operates the data in the relevant network. At the same time, some tools and databases are used in the entire network structure, and the entire network in the test environment runs in the Docker environment.

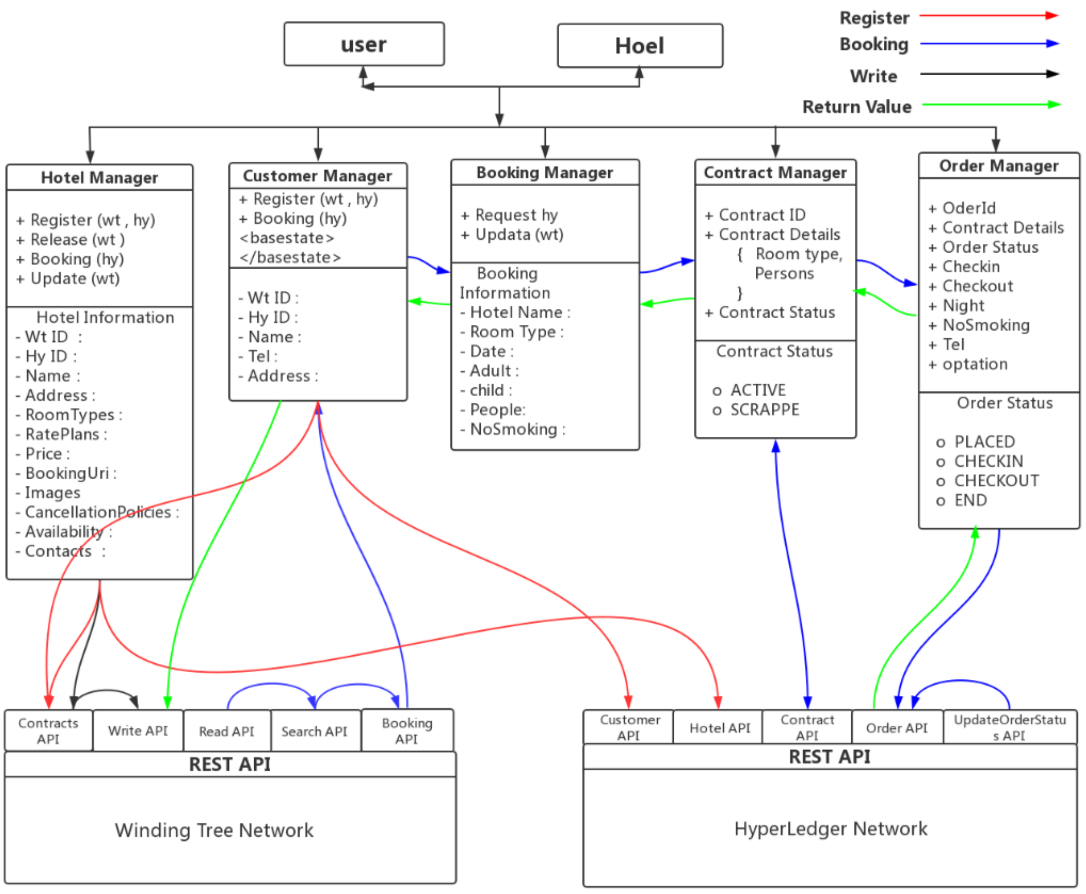


Figure 19: Interworking Network Business Flow Chart

4.1 Design of Interworking Application

From the perspective of the overall operating environment of the network[34], the division of labor between the main functions of the network is clear, which reduces the operating pressure of the network. The specific workflow of the network is roughly divided into registration, writing, querying, booking, updating, and all of the network structure can be realized. The implementation process still completes all business requirements by calling APIs with each other.

As shown in the figure above, the service provider first writes the service content into the network, and the example is illustrated in the figure for illustration. In the WindingTree network, the service content is registered to the network by calling WriteAPI. The client queries the service content (product information) through SearchAPI. The API transmits the information required by the client. After completing the query, the client makes a reservation. When the reservation is made, the reservation information is sent to In the Hyperledger Fabric network, a new order (Order) is created, and at the same time, the signing of the contract is completed. The hotel side processes the order information and completes the process to return the reservation success information. At this point, the entire booking process runs on the network and is recorded.

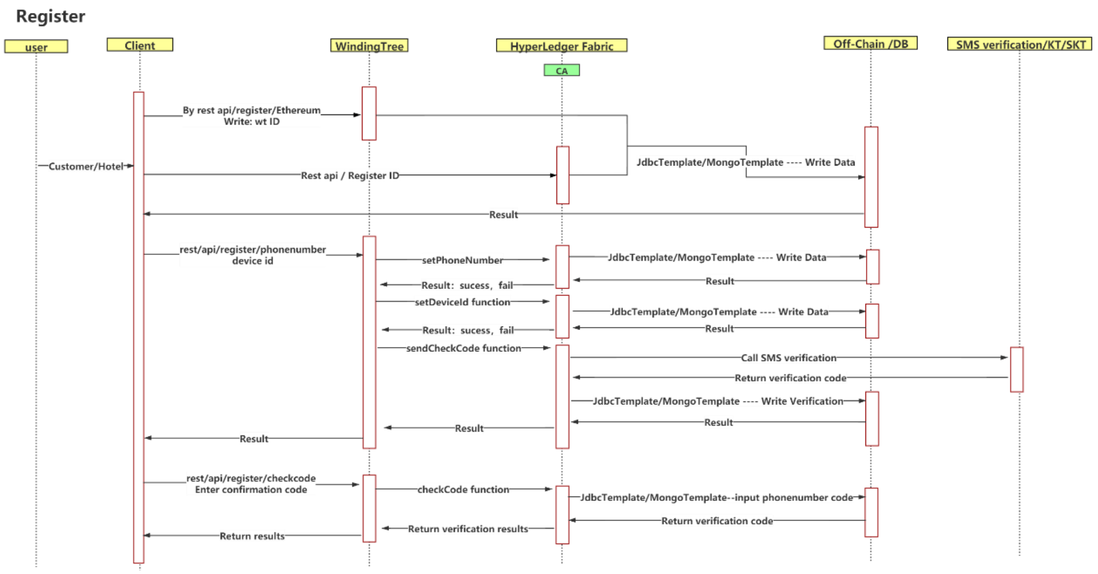

Register
user
Client
WindingTree
HyperLedger Fabric
CA
Off-Chain /DB
SMS verification/KT/SKT
By rest api/register/Ethereum Write: wt ID
Customer/Hotel
Rest api / Register ID
JdbcTemplate/MongoTemplate ---- Write Data
Result
rest/api/register/phonenumber device id
setPhoneNumber
JdbcTemplate/MongoTemplate ---- Write Data
Result
Result: sucess, fail
setDeviceId function
JdbcTemplate/MongoTemplate ---- Write Data
Result: sucess, fail
Result
sendCheckCode function
Call SMS verification
Return verification code
JdbcTemplate/MongoTemplate ---- Write Verification
Result
Result
Result
rest/api/register/checkcode Enter confirmation code
checkCode function
JdbcTemplate/MongoTemplate--input phonenumber code
Return verification code
Return verification results
Return results

Figure 20: Interworking Network Registration Flowchart

How to implement functions in the interworking network, such as registration, login, and which components of the transaction pass through the network. This section describes in detail the sequence diagram of the interworking work between business functions in the interworking network and components in the network.

First, explain how the client registers in the interworking network(Figure 20)

The client sends a registration request through the client and uses the ContractAPI in the WindingTree network to check in the user information on the network, and at the same time, sends a registration application to the Hyperledger Fabric network. In the Hyperledger Fabric network, the identity is first created by the CA and issued to the registration applicant. The user authenticates the user and sends a verification request to a third-party organization via text message or other methods. After verification, the two networks will return the registration information to the DB after the registration is completed. In the Hyperledger Fabric Network, the information will be stored in Level DB. To achieve registration of the entire network.

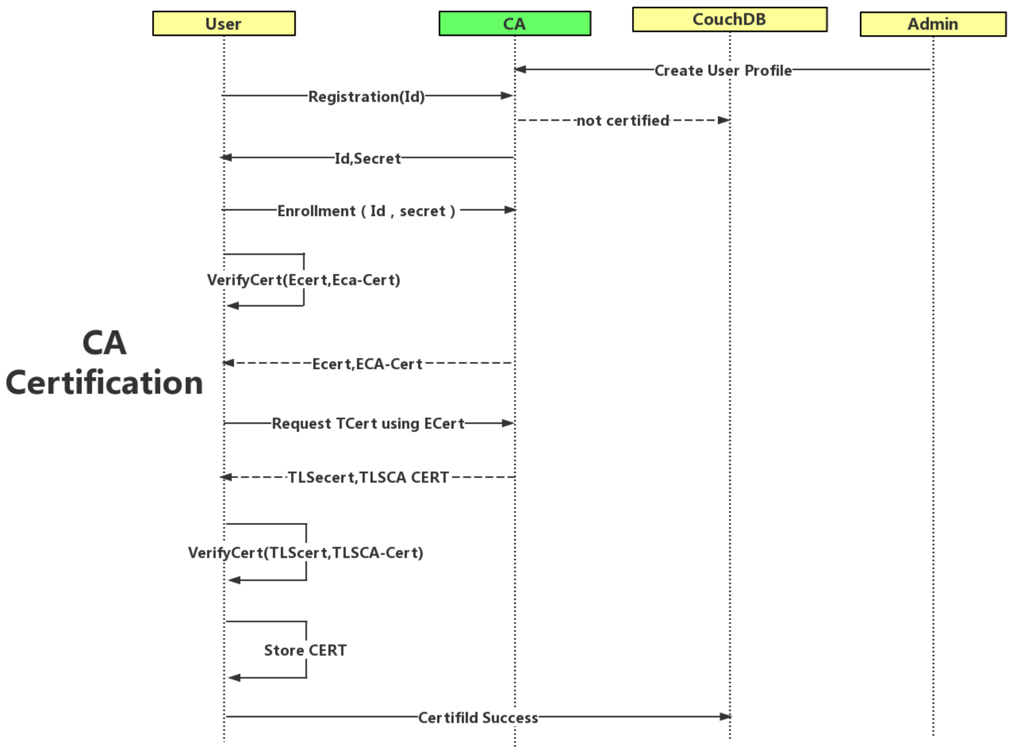


Figure 21: Interoperable Network CA Verification

1. When registering in the Hyperledger Fabric Network, users need to verify and issue certificates through a CA.
2. Enrollment (Id, Secret): Send the id and secret to the CA centre, and self-verify at the same time.
3. The client will receive the ECA-cert (Ecert), request the Court using Ecert, and return Tlsecert and TLSCA-cert upon success.
4. Authenticated and stored locally. Also stored in the DB.

Finally, the sequence diagram when the user logs in the interworking network:

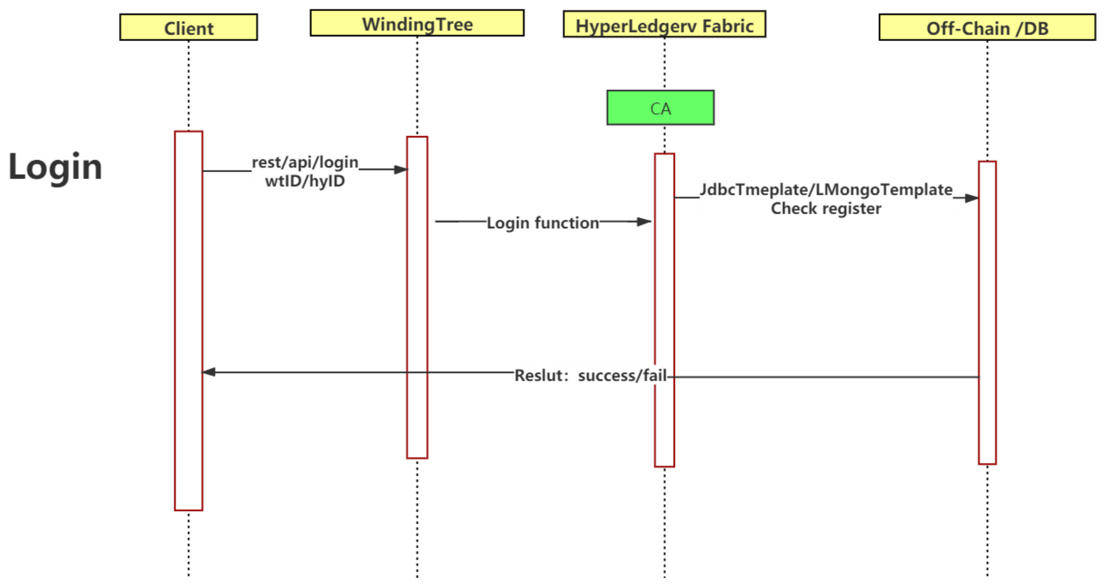


Figure 22: Interworking Network Login Authentication

Login on to the network through the client. RestAPI sends WindingTree ID and Hyperledger ID to the corresponding network. The Hyperledger Fabric Network is signed by CA certification. During the internal process, a verification request is sent to the DB to verify the existence of the information. The WindingTree network directly verifies the data in the DB. If there is data, it returns success. If it does not exist, It returns to Fail and prompts to register.

4.2 Interworking Application Implementation

The transaction of the two networks is the focus of this thesis. The transactions are implemented through the architecture of the two blockchain networks, reducing network pressure and ensuring data security. There is a problem with the storage of data in the process of network interoperability transactions. In the actual development process, the plan can be determined and formulated in advance.

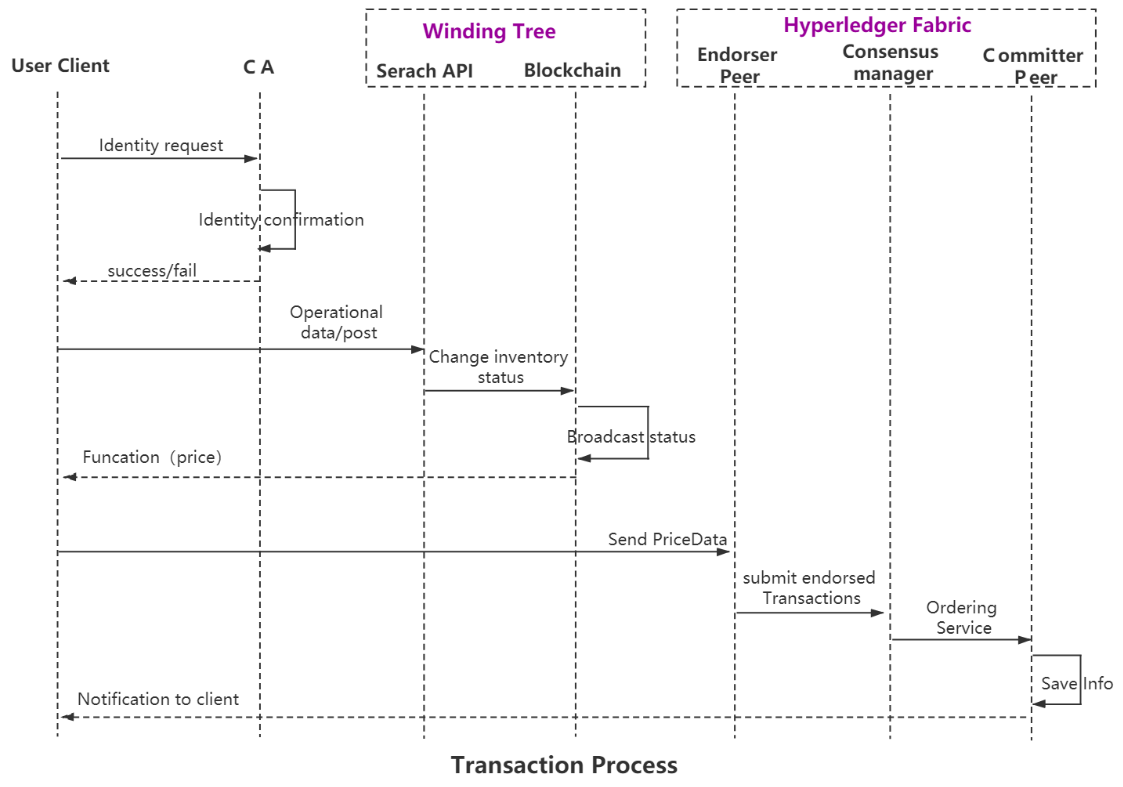
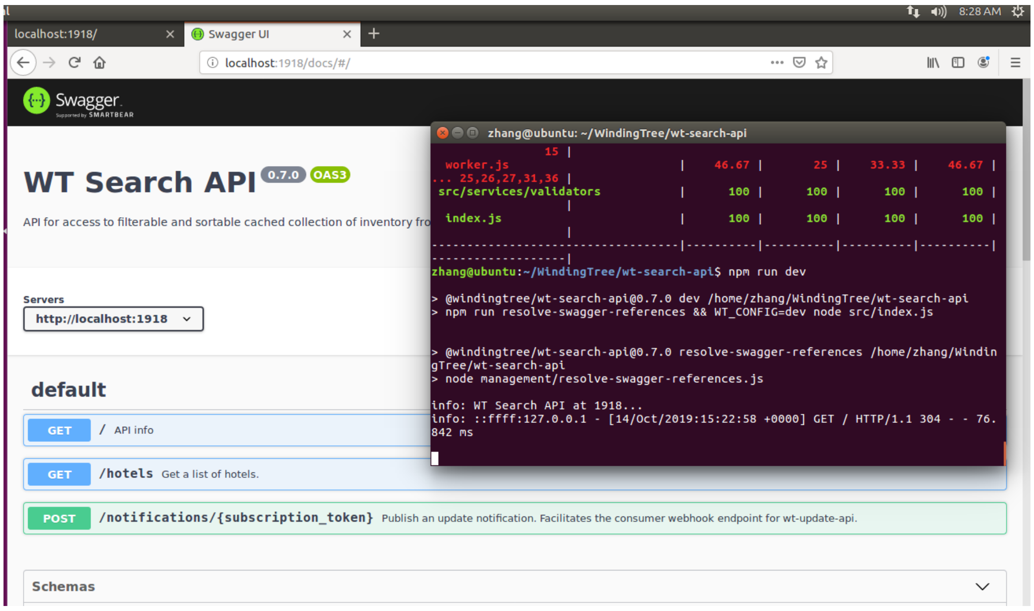


Figure23: Interworking Network Transaction Process

After the user verifies the login name on the client, he will search for product information through the SearchAPI in WindingTree network and return the data to the client. The data is sent to the Hyperledger Fabric network during the transaction, and J stores the data and adds it to the block and broadcasts it after confirming the transaction and adds it to the main chain. At the same time, the value is returned to the network, the transaction is successful, and the inventory is reduced. The detailed process of the transaction has been described in detail in the previous chapter. If you do not understand it, please return to the previous chapter

The API interface is integrated into the application to form the client. Through the development of WindingTree, upload complete hotel data information. In this way, the hotels in the pool have a large number of hotel IDs, and we can get hotel in-formation by ID. Clients can search for hotel information through development. At the same time, the OTA company can also sell for hotel rooms. Two client programs are used to demonstrate the operation of the client by the client and the OTA management terminal. The client implements the necessary operations on the WindingTree network, which are three operations: registration, query, and booking. A detailed explanation of each operation through different pages. Below I will introduce the implementation of all client functions through the API interface:



localhost:1918/docs/#/

WT Search API 0.7.0 OAS3

API for access to filterable and sortable cached collection of inventory fro

Servers: http://localhost:1918

default

GET / API info

GET /hotels Get a list of hotels.

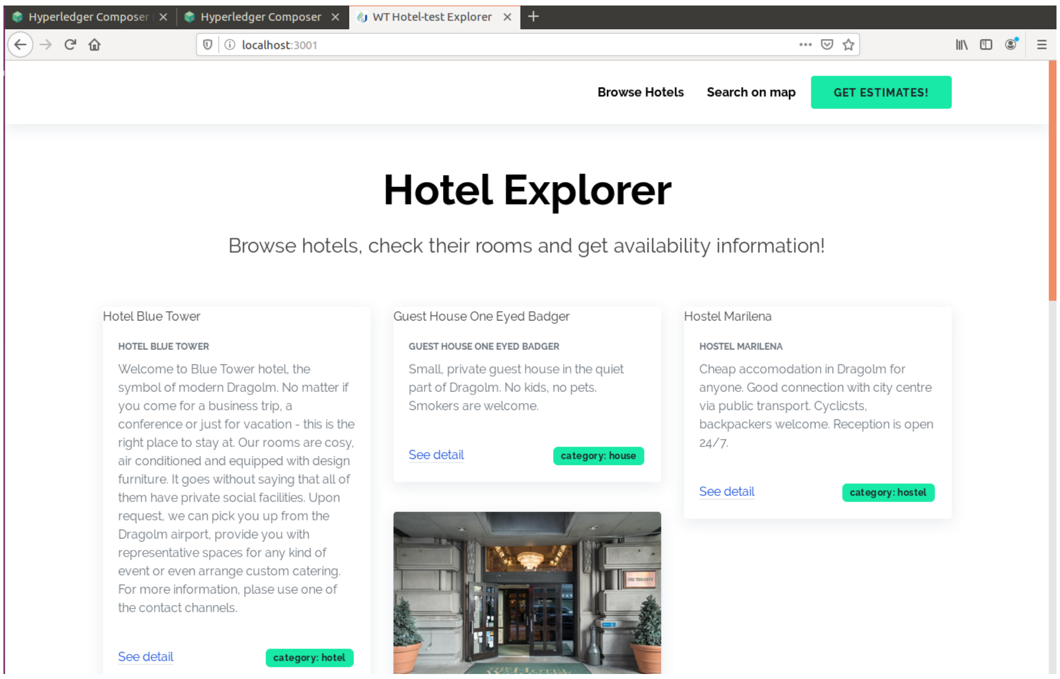
POST /notifications/{subscription_token} Publish an update notification. Facilitates the consumer webhook endpoint for wt-update-api.

Schemas

```
zhang@ubuntu: ~/WindingTree/wt-search-api
                   15 |
  worker.js                              |    46.67 |       25 |    33.33 |    46.67 |
... 25,26,27,31,36 |
 src/services/validators                 |      100 |      100 |      100 |      100 |
                      |
  index.js                               |      100 |      100 |      100 |      100 |
                      |
-----------------------------------------|----------|----------|----------|----------|
-------------------|
zhang@ubuntu:~/WindingTree/wt-search-api$ npm run dev

> @windingtree/wt-search-api@0.7.0 dev /home/zhang/WindingTree/wt-search-api
> npm run resolve-swagger-references && WT_CONFIG=dev node src/index.js


> @windingtree/wt-search-api@0.7.0 resolve-swagger-references /home/zhang/Windin
gTree/wt-search-api
> node management/resolve-swagger-references.js

info: WT Search API at 1918...
info: ::ffff:127.0.0.1 - [14/Oct/2019:15:22:58 +0000] GET / HTTP/1.1 304 - - 76.
842 ms
```

Figure 24: WindingTree Network SearchAPI Implementation

Most of the functions in the client are demonstrative functions, and other functions will not be displayed until the predetermined stage. Therefore, all information about the hotel can be read through SearchAPI and displayed on the page through front-end technology. WindingTree Application calls are provided in the API interface. In the bottom layer of the blockchain, other APIs are used to implement functions through API routing technology.


Hyperledger Composer | Hyperledger Composer | WT Hotel-test Explorer

localhost:3001

Browse Hotels | Search on map | GET ESTIMATES!

Hotel Explorer

Browse hotels, check their rooms and get availability information!

Hotel Blue Tower

HOTEL BLUE TOWER

Welcome to Blue Tower hotel, the symbol of modern Dragolm. No matter if you come for a business trip, a conference or just for vacation - this is the right place to stay at. Our rooms are cosy, air conditioned and equipped with design furniture. It goes without saying that all of them have private social facilities. Upon request, we can pick you up from the Dragolm airport, provide you with representative spaces for any kind of event or even arrange custom catering. For more information, plase use one of the contact channels.

See detail | category: hotel

Guest House One Eyed Badger

GUEST HOUSE ONE EYED BADGER

Small, private guest house in the quiet part of Dragolm. No kids, no pets. Smokers are welcome.

See detail | category: house

Hostel Marilena

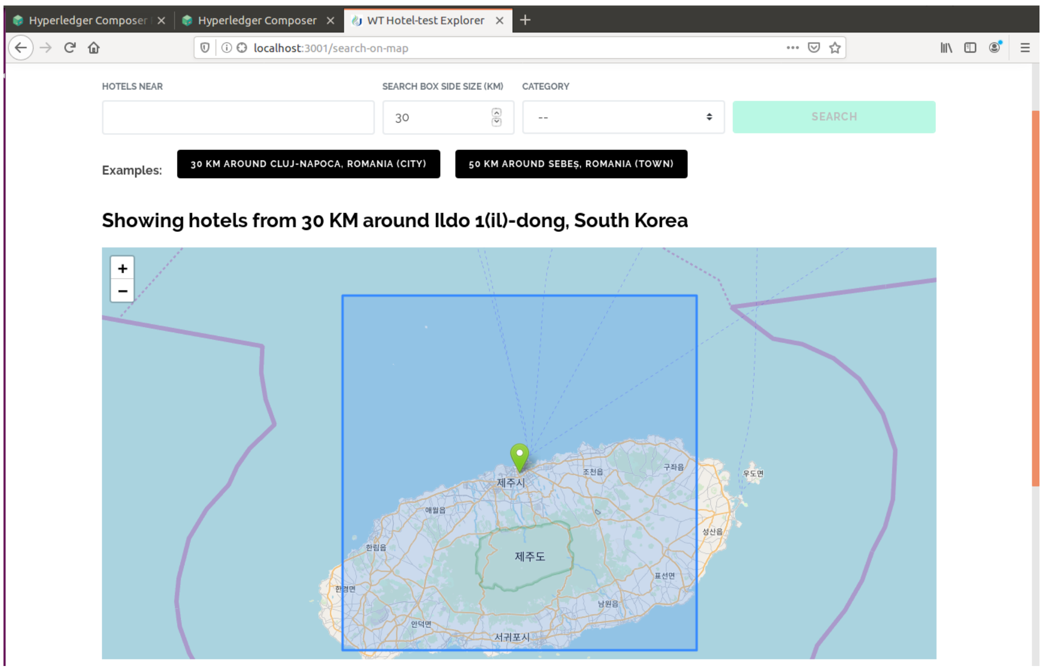
HOSTEL MARILENA

Cheap accomodation in Dragolm for anyone. Good connection with city centre via public transport. Cyclicsts, backpackers welcome. Reception is open 24/7.

See detail | category: hostel

Figure 25: WindingTree Network Client Implementation

After connecting the application to the API, a web page is developed. The client automatically displays the current hotel information, the page contains the hotel name and a detailed description of the hotel, and the service tag provided by the hotel. The blockchain network information is obtained and displayed on the page, including the hotel's detailed information. Users who initiate related operations on the blockchain on the client will be recorded in the public chain network.



HOTELS NEAR

SEARCH BOX SIDE SIZE (KM): 30

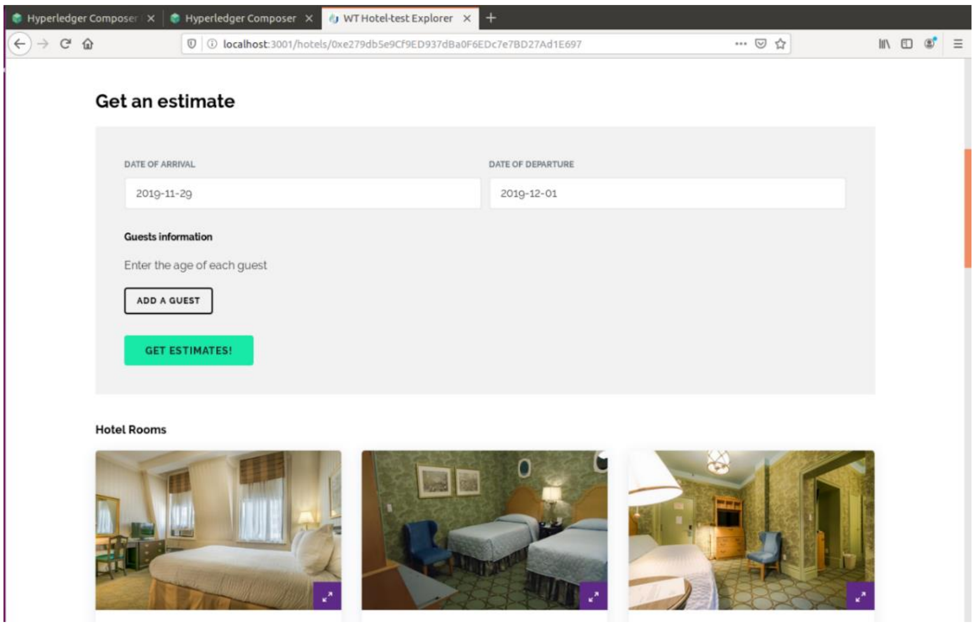
CATEGORY: --

Examples: 30 KM AROUND CLUJ-NAPOCA, ROMANIA (CITY) 50 KM AROUND SEBEȘ, ROMANIA (TOWN)

Showing hotels from 30 KM around Ildo 1(il)-dong, South Korea

Figure 26: WindingTree Network Client Map Search Implementation

The image above shows using location information to get the current location to find nearby hotels. Display all hotel information in the network within the specified area. The information includes the hotel name and address. Click to show the hotel to display detailed information. The user can quickly book the hotel. Use the map to show that the hotel is a friendly UI service and function for the booking service. Click the icon to get hotel information, query hotel room information, and make hotel reservations.


Hyperledger Composer
Hyperledger Composer
WT Hotel-test Explorer
localhost:3001/hotels/0xe279db5e9Cf9ED937dBa0F6EDc7e7BD27Ad1E697

Get an estimate

DATE OF ARRIVAL: 2019-11-29

DATE OF DEPARTURE: 2019-12-01

Guests information

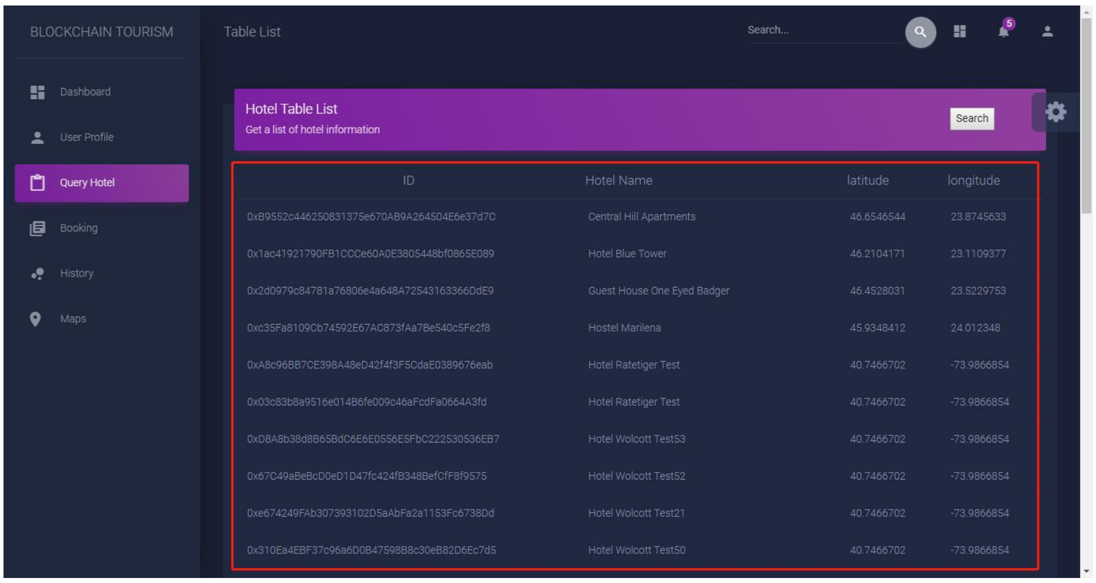
Enter the age of each guest

Hotel Rooms

Figure 27: WindingTree Network Booking Implementation

Query the hotel's room availability by entering the date. In some cases, Alai will be automatically displayed on the list, along with all the information such as price and room name, room preference label, and so on. At this time, you only need to make a reservation through the reservation button to complete the order process.So far, the above interface has been developed for client users. The hotel and OTA belong to the server.

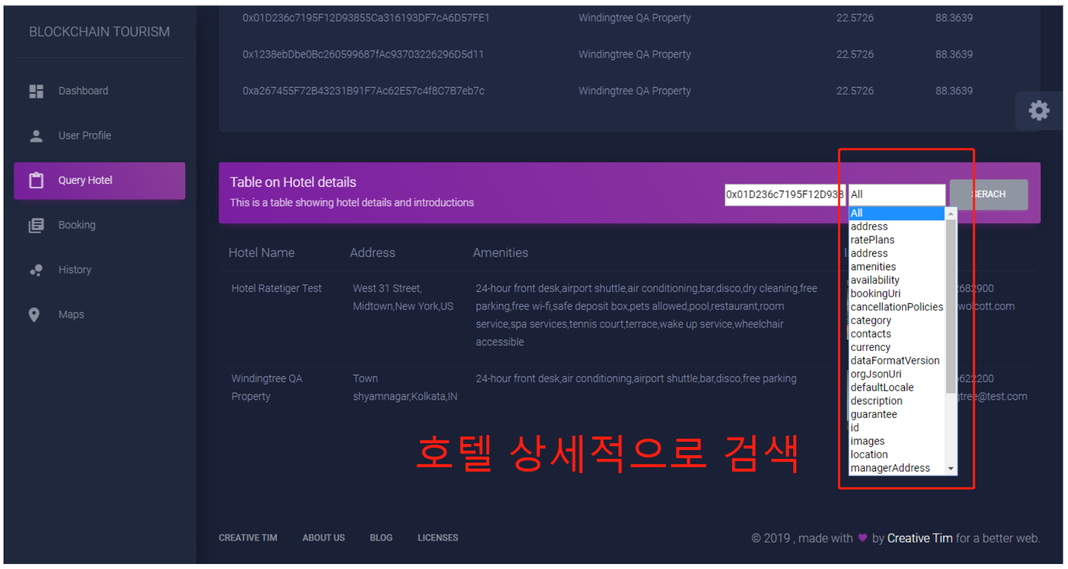
For the service provider, the management of the hotel is critical. The entry and reading of all hotel information and the provision of hotel data to agents need to be completed by the server. Simple development of the hotel server to achieve its functions.



ID	Hotel Name	latitude	longitude
0xB9552c446250831375e670AB9A264504E6e37d7C	Central Hill Apartments	46.6546544	23.8745633
0x1ac41921790FB1CCCe60A0E3805448bf0865E089	Hotel Blue Tower	46.2104171	23.1109377
0x2d0979c84781a76806e4a648A72543163366DdE9	Guest House One Eyed Badger	46.4528031	23.5229753
0xc35Fa8109Cb74592E67AC873fAa7Be540c5Fe2f8	Hostel Marilena	45.9348412	24.012348
0xA8c96BB7CE398A48eD42f4f3F5CdaE0389676eab	Hotel Ratetiger Test	40.7466702	-73.9866854
0x03c83b8a9516e014B6fe009c46aFcdFa0664A3fd	Hotel Ratetiger Test	40.7466702	-73.9866854
0xD8A8b38d8B65BdC6E6E0556E5FbC222530536EB7	Hotel Wolcott Test53	40.7466702	-73.9866854
0x67C49aBeBcD0eD1D47fc424fB348BefCfF8f9575	Hotel Wolcott Test52	40.7466702	-73.9866854
0xe674249FAb307393102D5aAbFa2a1153Fc6738Dd	Hotel Wolcott Test21	40.7466702	-73.9866854
0x310Ea4EBF37c96a6D0B47598B8c30eB82D6Ec7d5	Hotel Wolcott Test50	40.7466702	-73.9866854

Figure 28: WindingTree Network OTA Backend Read Implementation

The above picture shows all the hotels that have registered on the Internet. The information on the hotel is obtained through ReadAPI. From the picture, it can be seen that the hotel has a unique ID, hotel name, address, telephone, and other necessary information. All the hotel information in the hotel directory is obtained through the background. If the OTA needs a proxy, it only needs to know the hotel ID to query the specific information of the hotel through the development page below.


BLOCKCHAIN TOURISM

0x01D236c7195F12D93855Ca316193DF7cA6D57FE1	Windingtree QA Property	22.5726	88.3639
0x1238ebDbe0Bc260599687fAc93703226296D5d11	Windingtree QA Property	22.5726	88.3639
0xa267455F72B43231B91F7Ac62E57c4f8C7B7eb7c	Windingtree QA Property	22.5726	88.3639

Table on Hotel details

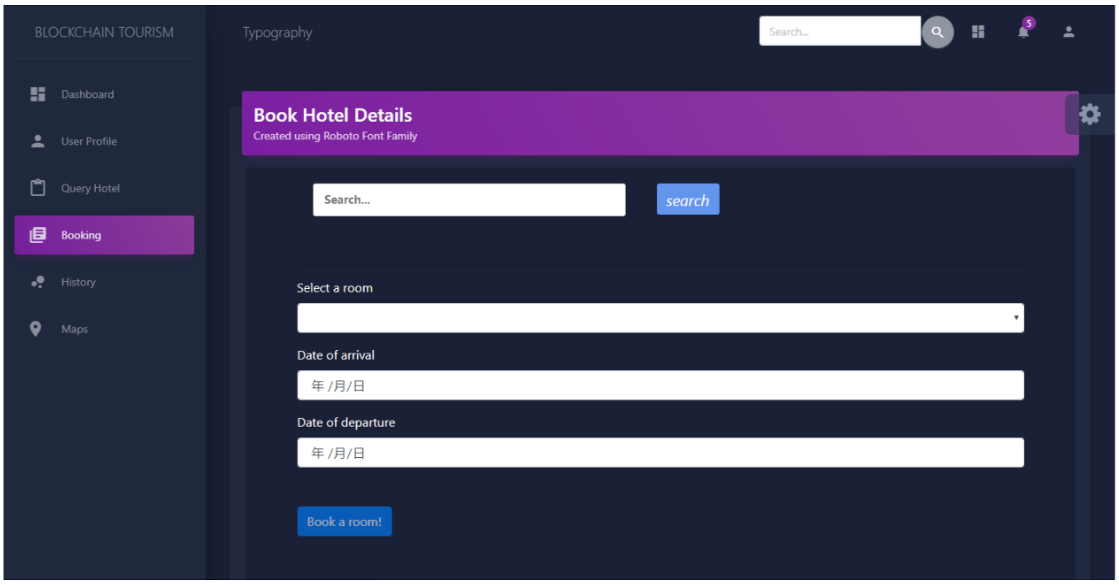
This is a table showing hotel details and introductions

0x01D236c7195F12D938 All SERACH

Hotel Name	Address	Amenities
Hotel Ratetiger Test	West 31 Street, Midtown,New York,US	24-hour front desk,airport shuttle,air conditioning,bar,disco,dry cleaning,free parking,free wi-fi,safe deposit box,pets allowed,pool,restaurant,room service,spa services,tennis court,terrace,wake up service,wheelchair accessible
Windingtree QA Property	Town shyamnagar,Kolkata,IN	24-hour front desk,air conditioning,airport shuttle,bar,disco,free parking

Figure 29: WindingTree Network OTA Backend Query Implementation

The operation of obtaining the detailed information of the hotel by entering the ID of the hotel is as shown in the figure above. Select the required hotel information through the filter box. As the OTA needs the hotel's detailed information, such as all hotel information such as room rate, availability date. When an agent is needed, only one transaction (agent) contract needs to be sent to the hotel party.


BLOCKCHAIN TOURISM

Typography

Book Hotel Details

Created using Roboto Font Family

Select a room

Date of arrival

年 /月/日

Date of departure

年 /月/日

Figure 30: Sub-Chain OTA Backend Booking Implementation

When a hotel party needs to represent another hotel, it only needs to query the hotel information through the search box, obtain the room information, select the agent date, and send the reservation transaction. The agent can get the room. The actual purchase process is just the process of signing a contract. Most of the pages developed in the figure above have achieved the required functions. In the actual development process, the overall framework has been developed. For the business system, WindingTree network has completed the hotel business management function well. In the next chapter, we continue to talk about the network of trading systems.

Chapter5 Main-Chain Based on Wind-ingTree for Secure Tourism Service

In the previous chapter, we introduced the design of Multi-chain architecture. For the main chain, we use the tourism business management network provided by WindingTree. It puts the tourism business on the public chain network and provides APIs to operate the needs of the tourism business. This thesis uses Take the hotel as an example. The hotel information and inventory are written into the blockchain network, and an API interface is provided to connect to the Application for use. It serves as a public chain for consumers to search and query in the market. All travel business information is publicly displayed, and anyone can query the hotel list information.

5.1 Design ofMain-Chain Based on WindingTree

In the WindingTree network, design and development are mainly around three functions: identity, trust, and aggregation. The three functions are mainly designed according to the actual market demand, which is in line with market demand and has high market applicability. Below we explain these three commonalities:

Identity: Provide users and service providers with a unique identifier that uses the Ethereum account as the account ID and detailed identity design for the service provide(Hotel, Airline). As explained below.

Trust: For example, a hotel vendor and you want to control who can access your inventory. As an OTA, make sure that the hotel booked for the customer exists. The ORG.ID (this concept will be mentioned in later chapters) standard defines an agreement that helps buyers and sellers build trust between each other in a completely decentralized manner.

Aggregation: With thousands of online travel agencies now, it is essential to have an OTA database in one place. An API interface is also provided to call this information. In a decentralized market, any organization can show its identity to others, prove that it is a trustworthy company, and list the aggregations found by other companies.

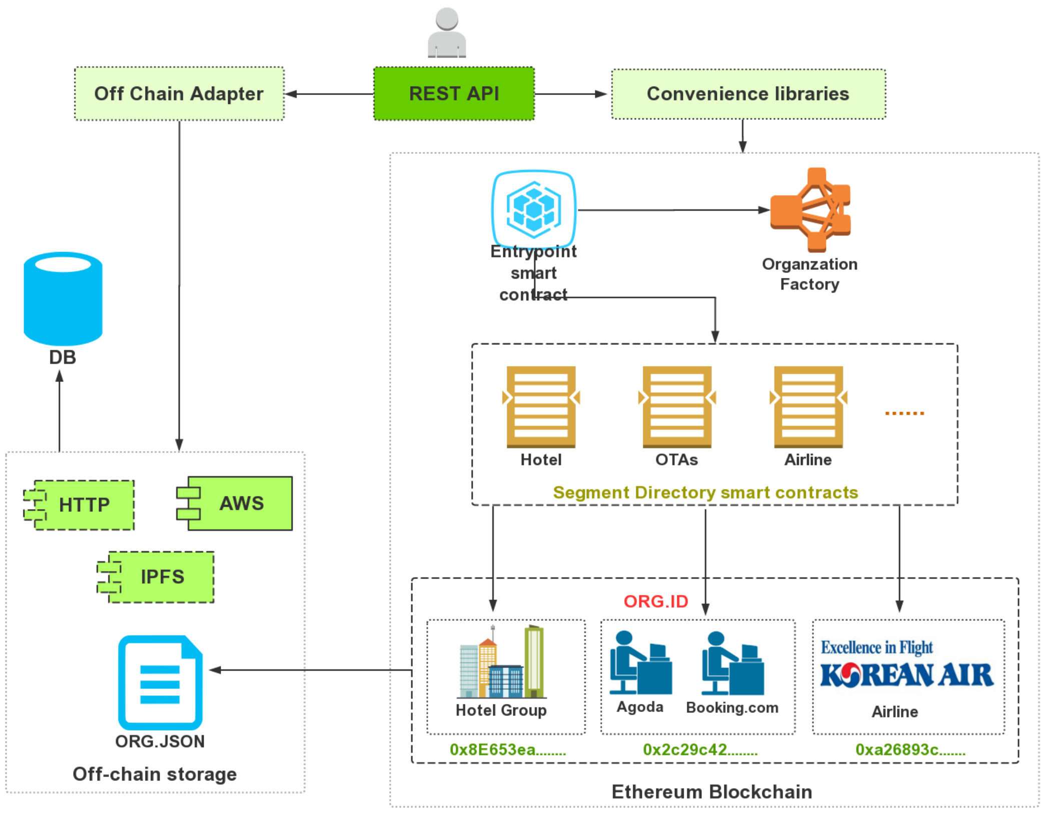


Figure 31: Main-Chain Operation Architecture

The figure above shows the architecture of the WindingTree network in Main-chain.

1.Entrypoint： A list of all segment directories and their Ethereum addresses provided by the platform instance.

2. Segment Directory：A collection of organizations within a business unit, such as a hotel. They are the primary source of discovery for the organization.

3. Organization Factory：A smart contract can create a new instance of the ORG.ID application.

4. Organization.ID(ORG.ID) consists of the following two parts.0xORG is the smart contract interface, and ORG.JSON is a file that contains the organization's information (name, address).

5. Off-chain storage: Distribute the file.

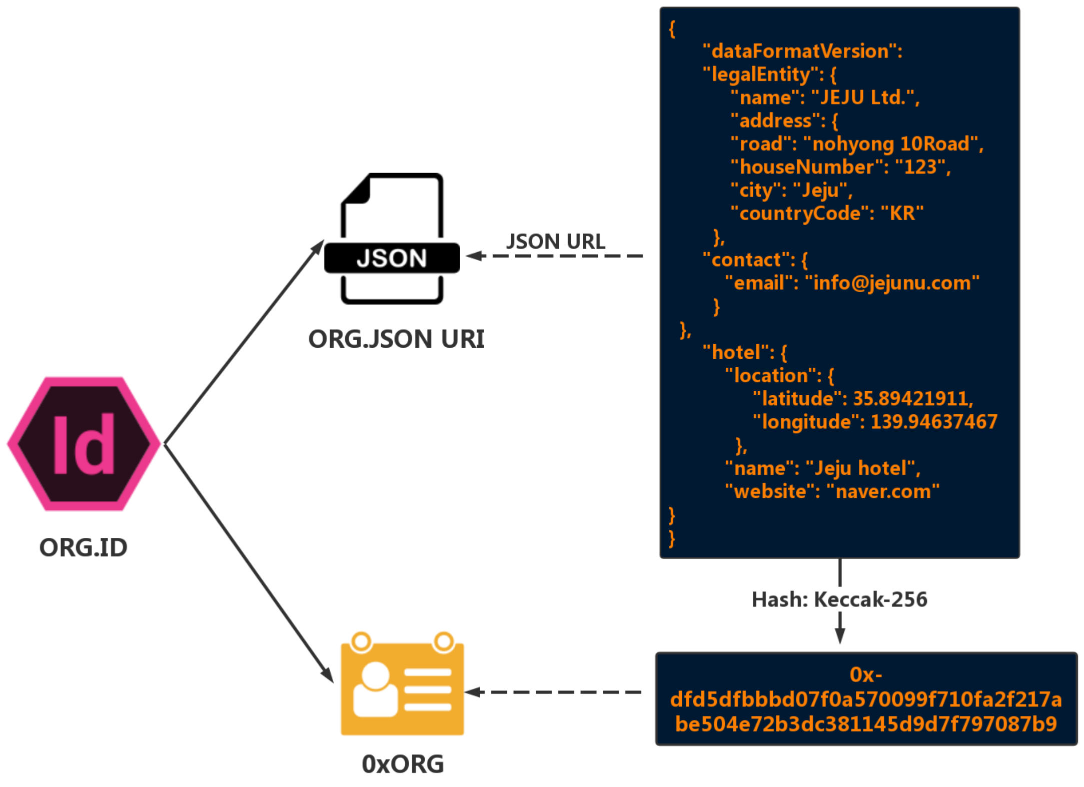
6. HTTP, Swarm, IPFS three file storage methods.

7. DB: Storage database.

The above points can roughly understand the components in the network structure. The WindingTree network controls the business system. The network is roughly divided into two parts: Ethereum Blockchain and distributed file storage system (off-chain storage).

The Ethereum Blockchain is responsible for the organizational management and operation of the network structure. First, the network enters the network from an Entrypoint entry point. The actual Entrypoint is a smart contract. The smart contract contains three parts: Hotel, OTA, and Airline. As a service provider, if you need to register, you first choose a directory as the directory entry point (Segment Directory). It must be clear that each directory has a unique 0xORG on the network (described in detail below). The actual 0xORG is an Ethereum address. When accessing the directory, you only need to connect to the Ethereum address to enter the directory, and then register the hotel. After success, you will get an ORG.ID. There will be a hotel in the online hotel directory. The API provided by the network can access the specific information of the hotel. Hotel information is a JSON file. If it is stored on the chain, it is quite expensive, so that we will store some data off-chain.

The internal use of smart contracts as IDs in Ethernet has uniqueness in the network structure. The first problem in using the WindingTree network is to design the ORG.ID, which is the only proof for hotels in the network. First, you need to make an ORG.JSON file, put the hotel and legal representative information in the file, and some relevant hotel certificates. JSON files need to be stored off-chain. Reducing network costs can also improve network performance.


JSON
JSON URL
ORG.JSON URI
Id
ORG.ID
Hash: Keccak-256
0x-
dfd5dfbbbd07f0a570099f710fa2f217a
be504e72b3dc381145d9d7f797087b9
0xORG

```
{
    "dataFormatVersion":
    "legalEntity": {
        "name": "JEJU Ltd.",
        "address": {
        "road": "nohyong 10Road",
        "houseNumber": "123",
        "city": "Jeju",
        "countryCode": "KR"
    },
    "contact": {
      "email": "info@jejunu.com"
    }
 },
    "hotel": {
      "location": {
          "latitude": 35.89421911,
          "longitude": 139.94637467
        },
      "name": "Jeju hotel",
      "website": "naver.com"
}
}
```

Figure 32: WindingTree Network ORG.ID Composition

0xORG is obtained by hashing the ORGJSON file. This processing can ensure that the information between 0xORG and the company entity is immutable, preventing fraud, and will be mentioned later if you need to modify the company information. Each time you perform a HASH operation on the JSON file to ensure uniformity, you can see that the obtained HASH (df5dfbb ..... 087b9) cannot be used as 0xORG because all values in the Ethereum environment start with 0x. For example, 0x246gd5 .... e24g2d4, therefore, each time the HASH value obtained after being encrypted through the JSON file needs to be prefixed with 0x, the rewritten HASH value can be used as the 0xORGd value in the Ethereum environment.

Write the hotel's ORG.JSON to register the hotel information in the travel network. If 0xORG is a functional part of ORG.ID, ORG.JSON is the actual information record. It can be stored anywhere that other market participants can access (e.g., via HTTP, FTP, Swarm, IPFS.). For the company, the information of the service provider needs to provide legal and legal representative parts.

```
{ // Legal representative information
  "dataFormatVersion": "2.3.1",
  "updatedAt": "2018-11-03T22:10:05.428Z",
  "legalEntity": {
    "name": "kim huming .",
    "address": {
      "road": "10road Yeong-dong",
      "houseNumber": "110",
      "city": "Juju",
      "countryCode": "KR"
    },
    "contact": {
      "email": "test@jejunu.ac.kr"
    }
  },
  // Hotel Information
    "hotel": {
      "location": {
        "latitude": 21.5412,
        "longitude": 48.5641223
      },
      "name": "kim-hotel",
      "website": "https://kimhotel.com",
      "apis": [
        {
          "entrypoint": "api.kim-hotels.com",
          "docs": "developers.kim-hotels.com",
          "format": "OTA",
          "version": "1.9"
        }
      ]
    }
}
```

Smart contracts in the network only have the function of executing programs. Using this to design subdivided directories, through smart contract program execution, we can find the corresponding directories. This step is an excellent solution to the Multi-business situation in the real market environment. Different departments and responsibilities can be set up according to different businesses. It also shows that this paper responds well to the market environment. Hotel information needs to be registered in a directory on the network; It is registered through the following process. As mentioned earlier, smart contracts are a series of hash values on the chain. Through the hash values, you can query the details of smart contracts on the chain. After the network nodes are set up, a smart contract is generated on the chain as the network's entry point.

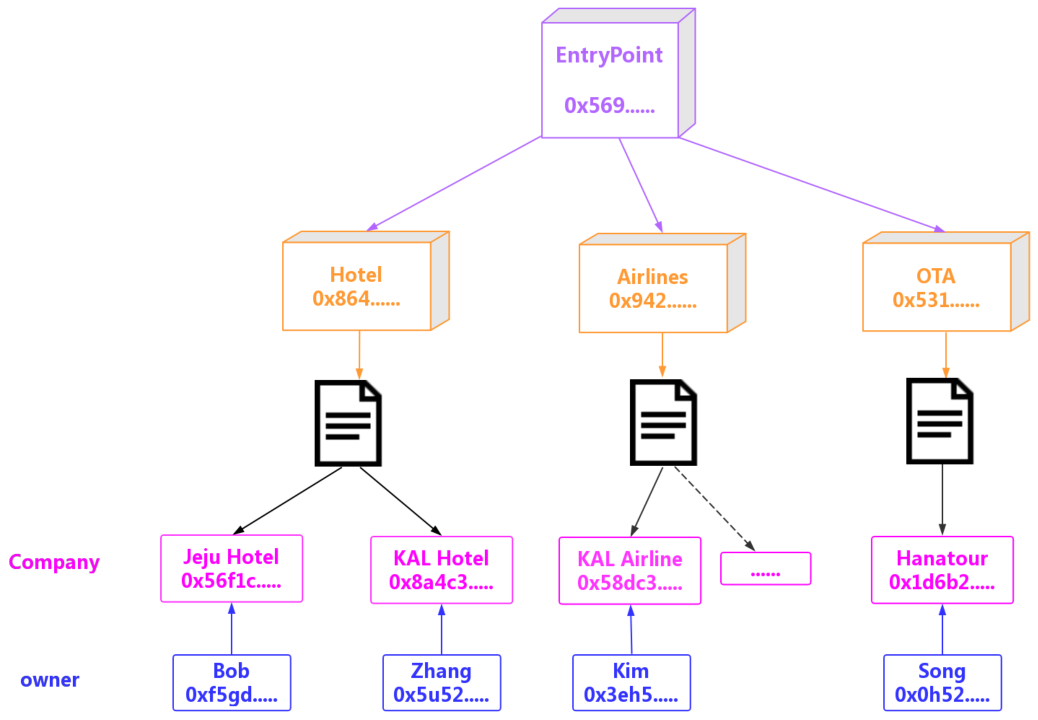


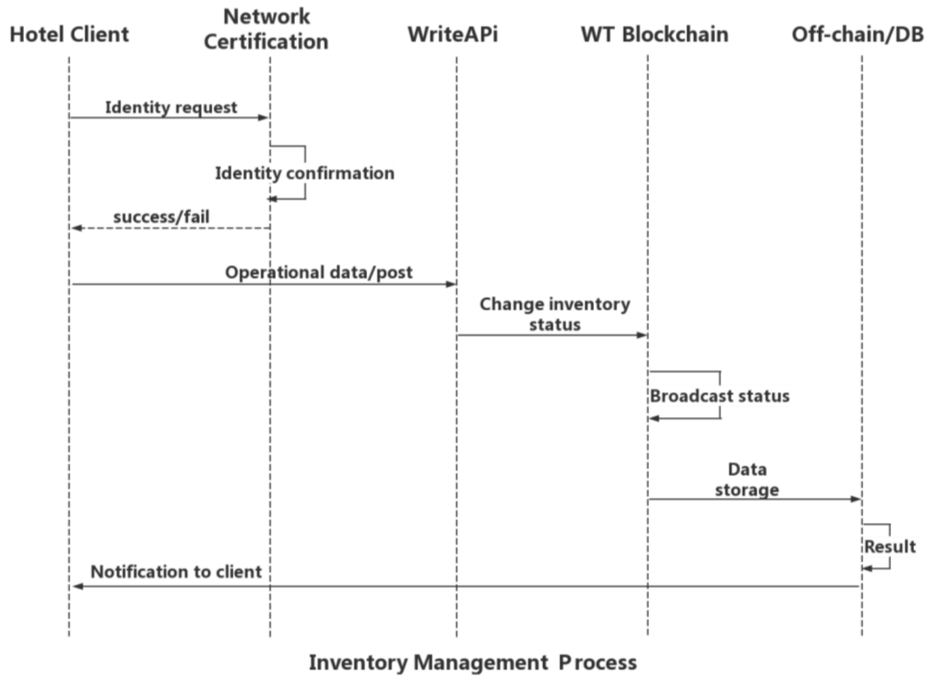
Figure 33: WindingTree Smart Contract Directory Composition

As shown in the figure above, generate a smart contract on the network as the entry point of the network environment (EntryPoint Smart Contract). At this time, the hash value is 0x568... How do you want to verify whether the entry point is generated? Section 4.3.2 will explain. Then, through the EntryPoint Smart Contract, three sub-directories (business) are created, namely Hotel, Airline, and OTA. At this time, the three sub-directories will each have a hash value as the unique value of each business. Because the three business hash values are generated through EntryPoint Smart Contract, if you want to access any of them, you need to connect to EntryPoint first, and then connect to the corresponding business through the business hash value. Using the Hotel Directory as an example, the Hotel Directory naturally manages the hotel business. If you are a hotel supplier and apply for registration in the Hotel Directory, the registration method is described in Chapter 3. After registration, you can get the information of registered hotels in the Hotel Directory pool. At the same time, the registered hotel (JejuHotel) will get the unique ID of the hotel: 0xf5gd ..., when registering, it will be registered through the legal representative of the hotel, and the reason for registration along with the hotel owner information, in order to prevent the hotel from being authentic Effectiveness.After registering company information, companies can mutually access business information through the entry points of each business. The following will mention the agency (OTA) for Hotel, Airline, agency sales.

So far, when querying hotel information through the client, it can be real and practical. The advantage of the design of the catalog is that there is no confusion caused by too many types of business.

5.2 Main-Chain Working Process

There are multiple functions on the main chain that need to call the API to implement operations. In this section, we do not explain in detail how the API works. We share how standard functions are transmitted on the main chain and which nodes pass through the network. Detailed description. The leading chain network is a public chain that discloses all information, and anyone can access it. Use the public chain to ensure that the information in the tourism market is correct and valid.



Inventory Management Process

Figure 34: Sub-Chain Inventory Management Process

In the business management system, how WindingTree manages the hotel through the network, and through the process of the network architecture, a diagram of the inventory management of the service provider on the service side is obtained.

Figure 34, Use the write API in WindingTree to manage and update the inventory status of the blockchain. Sellers first pass network certification. After successful verification, the seller changes the goods inventory and the WT blockchain network broadcasts to complete the agreement and save the data. After the broadcast is completed, there will be detailed inventory information on the main chain. Of course, only the HASH value of the commodity is recorded on the main chain. The specific data is stored on the DB outside the chain. Every operation record of the service provider will be recorded on the chain.

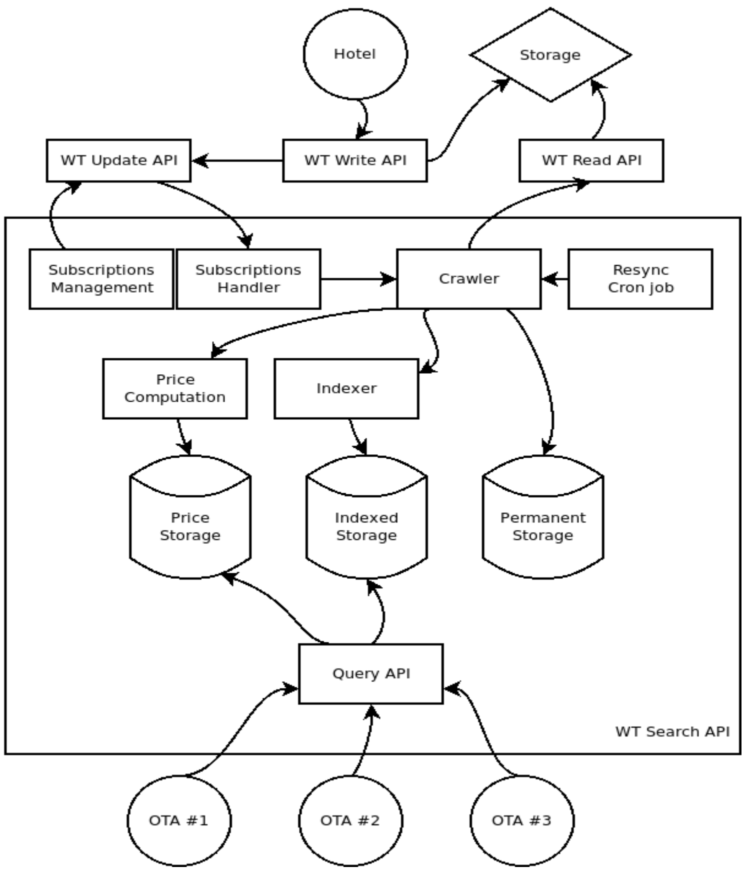


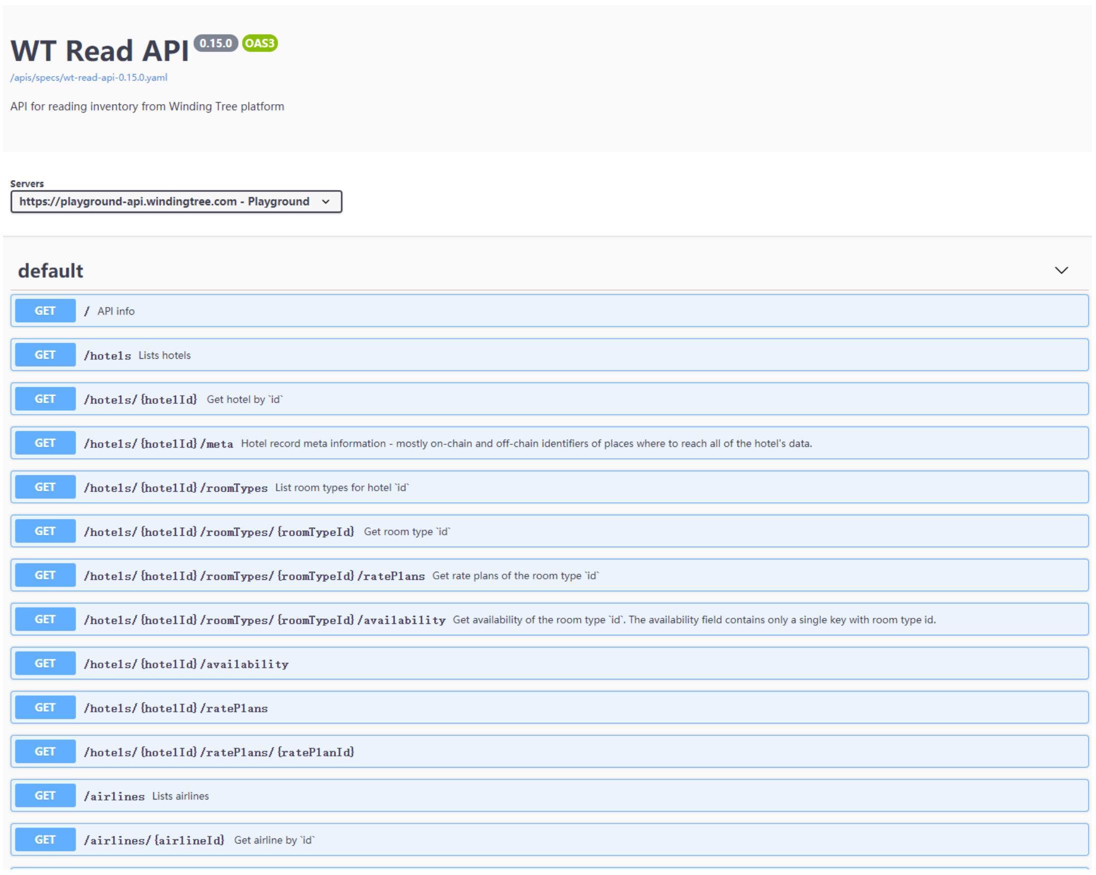
Figure 35: WindingTree Network Search API

The Search API is connected to the blockchain network via API routing conversion in the network to implement the query function. The query operation is implemented in two parts of the architecture. The first is the client's query, and the second is the server's query. The server's query is the query function provided when the OTA service is involved. Inquire. Both aspects of the business function use the same SearchAPI, which also reflects the importance of using API interfaces in website applications. All tasks can be completed by calling the interface.

Figure 35, The exact functional architecture between the service provider and the OTA can be seen in the figure. First, the Hotel uses WriteAPI to connect to the UpdateAPI to update and write information about the hotel. The subscription management passes the updated data to Crawl, and then sends them to different storages after classification. At this time, the information in the entire Storage is written. The OTA uses the QueryAPI to query the data in the DB. In the actual middle process, it uses the SearchAPI and ReadAPI to query. In the whole process, the mutual conversion between different functions is realized through the mutual calls of APIs.

5.3 Main-Chain Implementation

In the previous section, we used WindingTree to implement the network architecture and design a tourism blockchain network that meets the business. In this section, we implement business requirements by integrating API operations. The application operates the network by calling the API. It is open to the outside world and has a great test of network security and pressure. The routing mechanism can significantly reduce risks. The following is the application of Main-chain, which is applied to the tourism blockchain.


WT Read API 0.15.0 OAS3

/apis/specs/wt-read-api-0.15.0.yaml

API for reading inventory from Winding Tree platform

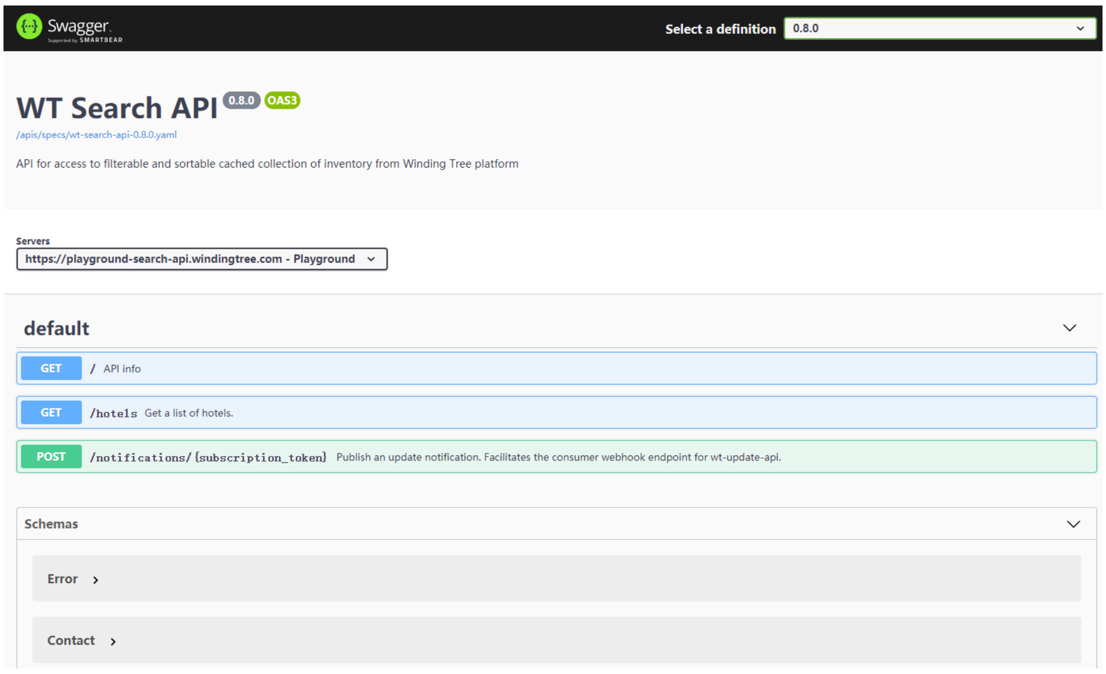
Servers
https://playground-api.windingtree.com - Playground

default

```
GET  /  API info
GET  /hotels  Lists hotels
GET  /hotels/{hotelId}  Get hotel by `id`
GET  /hotels/{hotelId}/meta  Hotel record meta information - mostly on-chain and off-chain identifiers of places where to reach all of the hotel's data.
GET  /hotels/{hotelId}/roomTypes  List room types for hotel `id`
GET  /hotels/{hotelId}/roomTypes/{roomTypeId}  Get room type `id`
GET  /hotels/{hotelId}/roomTypes/{roomTypeId}/ratePlans  Get rate plans of the room type `id`
GET  /hotels/{hotelId}/roomTypes/{roomTypeId}/availability  Get availability of the room type `id`. The availability field contains only a single key with room type id.
GET  /hotels/{hotelId}/availability
GET  /hotels/{hotelId}/ratePlans
GET  /hotels/{hotelId}/ratePlans/{ratePlanId}
GET  /airlines  Lists airlines
GET  /airlines/{airlineId}  Get airline by `id`
```

Figure36: Read API on Main-Chain

Opening as a public chain to users requires API integration of business functions in the Main-chain to form a unified interface. The above picture is the ReadAPI in the travel network. This API can read data on the network. ReadAPI belongs to the network. Low-level API, there are other APIs that need to call ReadAPI to operate the network. In this implementation, you can read the necessary information about the hotel and the various attributes of the hotel's room information. The above query function first needs to query the ID of the hotel, and obtain the various attribute values inside the hotel through the ID. In this application, the hotel and aviation products For development and implementation, more business needs are developed through the same operation. At this time, the application displays the hotel information to the user through ReadAPI, completing the necessary process of ReadAPI.



WT Search API 0.8.0 OAS3
/apis/specs/wt-search-api-0.8.0.yaml

API for access to filterable and sortable cached collection of inventory from Winding Tree platform

Servers
https://playground-search-api.windingtree.com - Playground

default

GET / API info

GET /hotels Get a list of hotels.

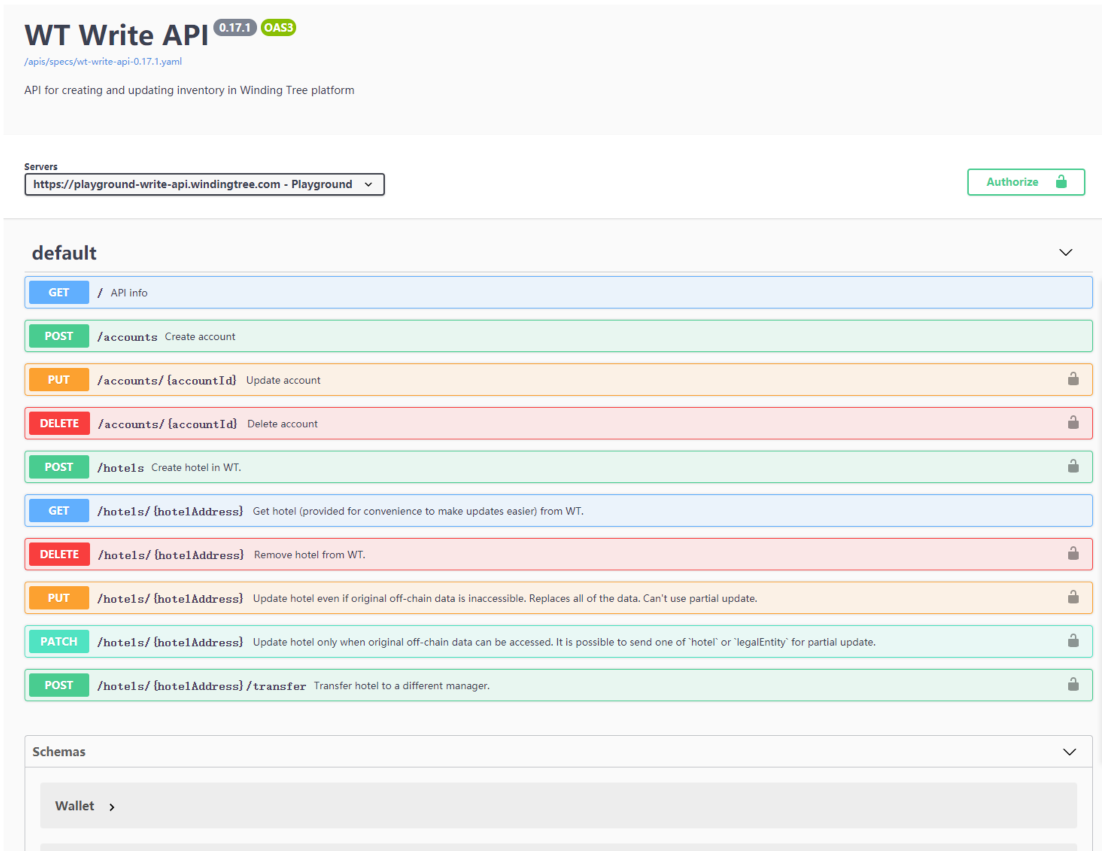
POST /notifications/{subscription_token} Publish an update notification. Facilitates the consumer webhook endpoint for wt-update-api.

Schemas

Error

Contact

Figure 37: Search API on Main-Chain



WT Write API 0.17.1 OAS3
/apis/specs/wt-write-api-0.17.1.yaml

API for creating and updating inventory in Winding Tree platform

Servers
https://playground-write-api.windingtree.com - Playground

default

GET / API info

POST /accounts Create account

PUT /accounts/{accountId} Update account

DELETE /accounts/{accountId} Delete account

POST /hotels Create hotel in WT.

GET /hotels/{hotelAddress} Get hotel (provided for convenience to make updates easier) from WT.

DELETE /hotels/{hotelAddress} Remove hotel from WT.

PUT /hotels/{hotelAddress} Update hotel even if original off-chain data is inaccessible. Replaces all of the data. Can't use partial update.

PATCH /hotels/{hotelAddress} Update hotel only when original off-chain data can be accessed. It is possible to send one of `hotel` or `legalEntity` for partial update.

POST /hotels/{hotelAddress}/transfer Transfer hotel to a different manager.

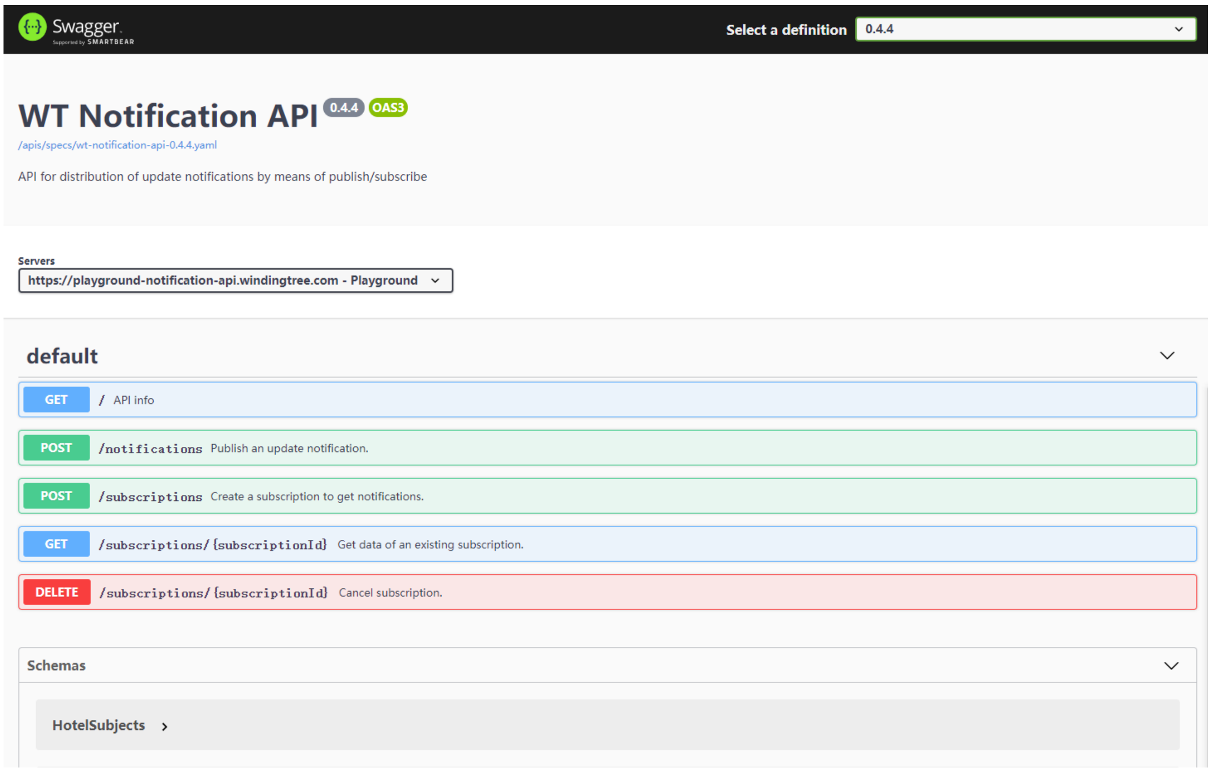
Schemas

Wallet

Figure 38: Write API on Main-Chain

Figure 37, When the user searches for information through the search box on the client, the ReadAPI is first called. At this time, ReeadAPI will connect to the ReadAPI for the query. Keywords usually query the client application query. The API is searching for the hotel ID. After arriving at the hotel, all the information in the hotel will be displayed to the client. The actual SearchAPI inside the API router performs a query by calling ReadAPI.

Figure 38, Where there are consumer services in the tourism chain, there are service providers. When the service provider operates the service information (the above is the operation of the hotel information), the data is uploaded to the Main-chain through the background. WriteAPI can write the data to the blockchain network. It is also required to modify and update the data. WriteAPI to operate. In the picture above, the hotel registers the hotel information (address, room type, phone.) in the Main-chain, and then stores the data in the database. WriteAPI provides functions for adding, deleting, and modifying. WriteAPI is responsible for the return value processing of the Sub-chain in the client. Make Main-chain and Sub-chai perform data association.



WT Notification API 0.4.4 OAS3

/apis/specs/wt-notification-api-0.4.4.yaml

API for distribution of update notifications by means of publish/subscribe

Servers

https://playground-notification-api.windingtree.com - Playground

default

GET / API info

POST /notifications Publish an update notification.

POST /subscriptions Create a subscription to get notifications.

GET /subscriptions/{subscriptionId} Get data of an existing subscription.

DELETE /subscriptions/{subscriptionId} Cancel subscription.

Schemas

HotelSubjects ›

Figure 39: Notification API on Main-Chain

The above Notification API provides a notification function. In the actual travel market, when a hotel needs to publish some preferential policies and other operations, it sends the message to the client through the Notification API to implement this function. At the same time, the reminder function of some clients also uses. The Notification API also performs functions for adding, deleting, and modifying. If you need more notification functions in your business, you only need to develop the Notification API.

In this section, the architecture design of the Main-chain is implemented. The underlying information only needs to realize business performance through APIs. In the architecture of the network, not all of this thesis has been implemented. Necessary business requirements in the tourism market have been developed and implemented — realization, realizing the feasibility of the network architecture. The functional design in Main-chain is publicly accessible and available for the query at any time. Although Main-chain is only responsible for the public access part of the network, it still has high requirements for the pressure of the blockchain network. Compared with the previous Single-chain network, Multi-chain architecture is particularly evident in theory.

Chapter6 Sub-Chain Based on HyperLdger Fabric for Secure Tourism Service

Through the above definition and description of Hyperledger Fabric, we have defined three roles and two assets in the trading system, so that the entire business network (WindingTree Network) and the trading network can be correctly connected. According to business needs, we can further develop the trading network to meet actual production needs.

It is especially important to design a model that works properly with the trading system. First, you need to understand the following concepts: **Assets**: Property list **Participants**: buyers and homeowners **Transactions (transaction process)**: complete the sale of houses, settlement list.

Participants can access transaction data but are limited by their roles. Real estate agents can create an application that provides a simple user interface for buyers and sellers to see the progress of the transaction. Commercial networks can also integrate existing inventory systems to complete the transfer of house assets. Also, other related roles can register to join this network as participants. For example, the Land Bureau may participate in the transaction and complete the transfer of land ownership.

6.1 Design of Sub-Chain Based on HyperLdger Fabric

Sub-chain is developed using Hyperledger Fabric tools. This thesis first designs the role of the Sub-chain network. In actual business scenarios, different roles are given different functions. Facing the tourism market, it is divided into three roles: hotel, customer, and manager. Participants are members of a business network and can own assets or initiate transactions. Participants are also modeled. Like assets, they must have unique identifiers and can also contain other optional attributes. A participant can have one or more identities. The modeling [35] language can be recommended to query and learn from the official website.

Through the above introduction, we design a model that meets the needs of the business. For the participants, we have established three role customers, companies, and regulators. Each role performs a different task.

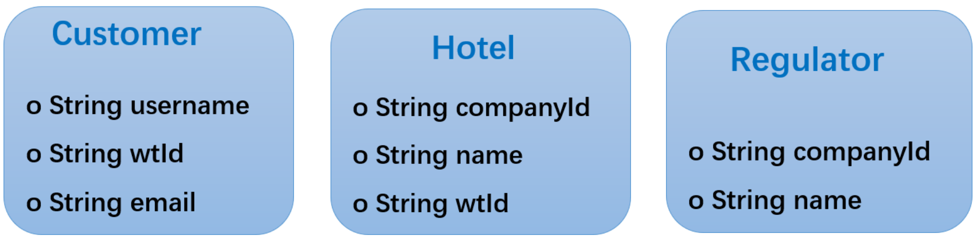


Figure 40: Hyperledger Fabric Participant Attributes

In the figure above, a username is set for the customer for the name in the network, wtId is used as the ID in the Main-chain for cross-chain data access verification, and email is used as the contact information for the guest. The compangId and wtId are used in the Hotel as the ID authenticator when the IDs in the Sub-chain and Main-chain are used for cross-chain communication. The regulator acts as a network administrator, in charge of different directories. Participants define the attributes they need in the business. The attributes define some important data for each type of business occasions that can carry or pass information. This information will be stored in the database in the chain.

Sub-chain needs to act as a trading system, and orders need to be processed when transactions occur. Therefore, orders are used as smart contracts in the tourism blockchain. The contract contains the order ID, reservation date, order status, room information, contact information, and Booking preferences, and more. After the order is formed, the status of the order can be modified through API calls. There are four states: PLACED, CHECKIN, CHECKOUT, and END. Assets here can refer to any tangible and intangible assets, and all tradable commodities that can be chained can be used as assets. Assets must have unique identifiers. Also, some additional information can be added to associate other assets or participants.

According to the business requirements, two assets, Order and Contract in the business network, are designed. Order records the order request sent by the client and also carries a write attribute and data. Ontrack's primary role is to sign the contract between the two parties.

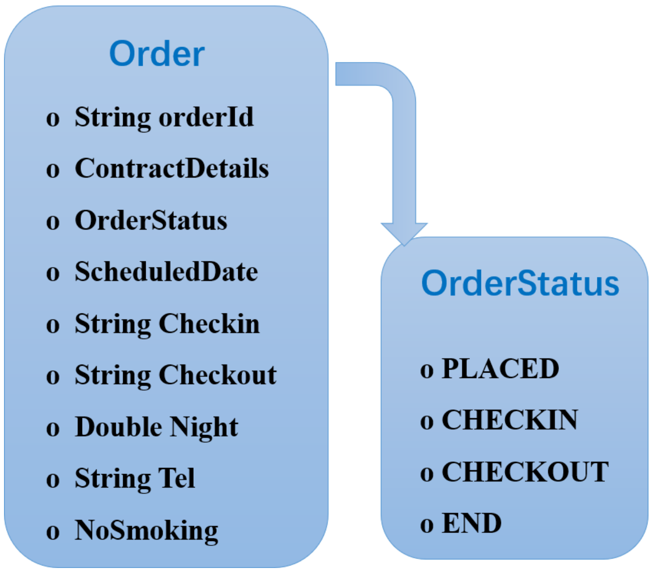


Figure 41: Hyperledger Fabric Asset: Order Property Design

The Order attribute contains all the necessary reservation information for the order. The Order will also generate its ID for broadcasting to the unique ID of the identity on the main chain. The client and server can query the details of the order through the query API, and also, OrderStutas is designed for Order to control the status of the order, and the business system (WindingTree Network) makes corresponding plans based on the status of the order. For example, check-in status, check the status through the system and change bed linen for customers.

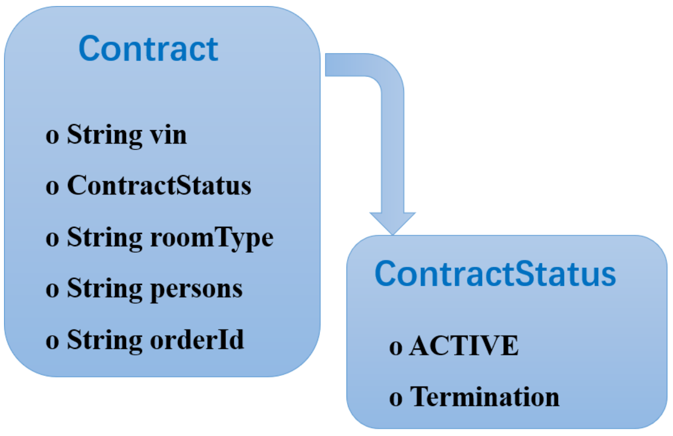


Figure 42: Hyperledger Fabric Asset: contract design

When the two parties trade, a contract is created in advance. The contract contains the content of the contract and the order number. All information about this order will be queried when inquired by the client. At the same time, the contract also set ContractStutas to indicate the status of the contract. Guest check-out in a production environment is equivalent to the end of the validity of this contract. However, this record always exists on the blockchain.

6.2 Sub-Chain Design Based on Hyperledger Fabric

In the previous section, participants and assets were designed with the primary purpose of better operation in the overall network. Starting from sending a transaction application, the client will carry much data to the transaction network and record it into the network as input data. This section mainly describes how the Sub-Chain Hyperledger Fabric network conducts transactions.

First, we need to know the concept of a channel. A channel is a private space for information communication between two nodes (Peer) or Multiple nodes. The transaction data in the channel is isolated from the outside of the channel[36] to ensure the security of the data in the channel. Transactions on the network must be performed on a channel, and each member participating in the mutual transaction needs to be authenticated and authorized in order to be processed on the channel. Fabric is a Multi-channel design. The system can create multiple channels. A node (Peer) can be added to different channels. Each channel has its genesis block and instantiated smart contract. Each channel has its anchor node. Through the anchor node, information can be exchanged with other channels, but the ledger in the channel itself will not be transmitted to another channel through one channel, and the channel ledger is separated.

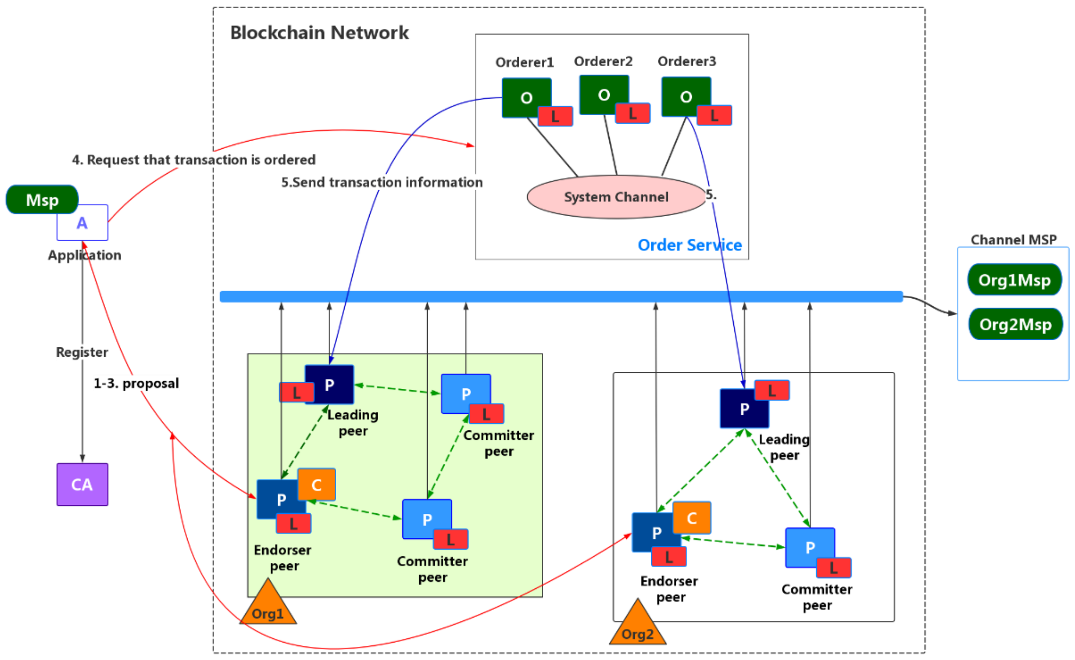


Figure 43: Hyperledger Fabric Blockchain Network Structure Diagram

The CA issues a certificate to Application.

1) After the application successfully connects to the node (Peer), it calls the chain code to propose the node (Peer);

2) The node (Peer) calls the chain code according to the proposal information;

3) Query and update the chain code, and then return the proposal information to the application;

4) The application sends transaction information to the Order

5) Orderer sends blocks containing transaction information to nodes (Peer);

6) The node (peer) updates the transaction block to the ledger, and finally completes the processing;

Nodes are the communication subject of the blockchain. There are many types of nodes related to the blockchain network: clients (applications), Peer nodes, order nodes, and CA nodes.In the sub-chain, the privacy and permission control of the network by the CA guarantees the security of the network. CA is the certificate authority (CA) of Hyperledger Fabric, and it is an optional membership service component in Hyperledger Fabric. It manages the identity certificates of various entities in the network. It mainly implements:

1. Responsible for the registration of the identity of all entities in the Fabric network.

2. Responsible for the issuance of digital certificates, including ECerts (identity certificates), TCerts (transaction certificates). Certificate renewal or revocation.

Fabric CA's role in the Hyperledger Fabric network: Access to the Fabric CA server can be achieved through the Hyperledger Fabric CA client or one of the Fabric SDKs, and all communication with the Hyperledger Fabric CA server is through the REST API.

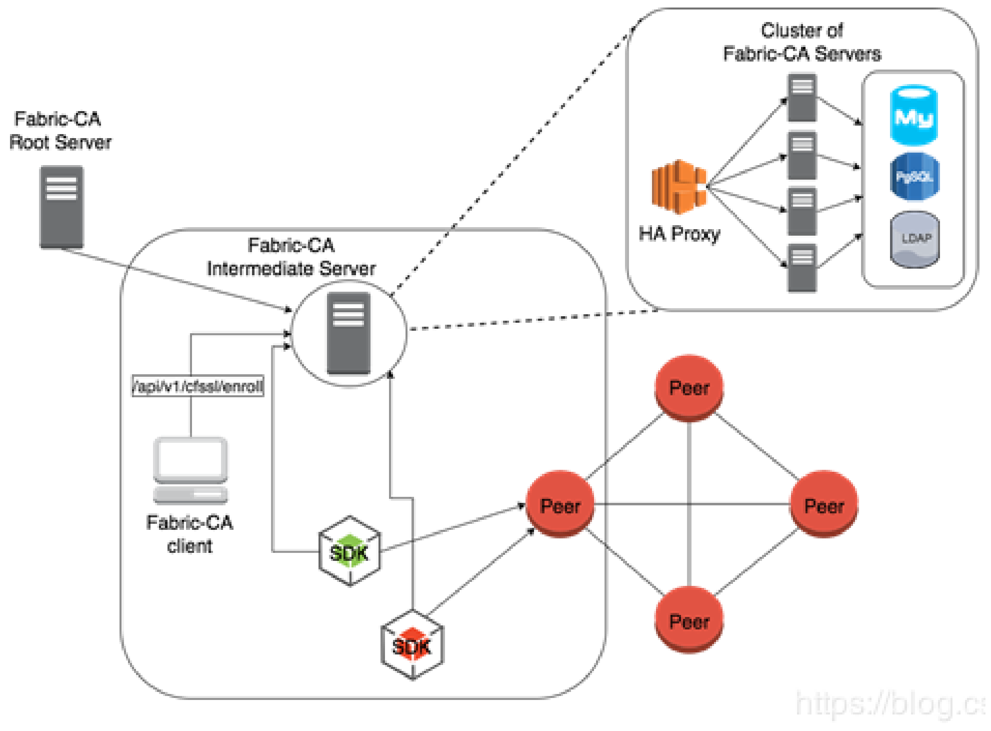


Figure 44: Hyperledger Fabric CA certificates Issuing Architecture Diagram

The Hyperledger Fabric CA client or SDK can connect to a Hyperledger Fabric CA server cluster, and the cluster is load-balanced by HA Proxy. The server may contain Multiple CAs, each CA is a root CA or an intermediate CA, and each intermediate CA has a parent CA. Hyperledger Fabric CA's identity information is stored in a database or LDAP. Currently, Fabric CA supports databases such as MySQL, PostgreSQL, and SQLite; SQLite databases are used by default. If LDAP is configured, identity information is retained in LDAP and not in the database.

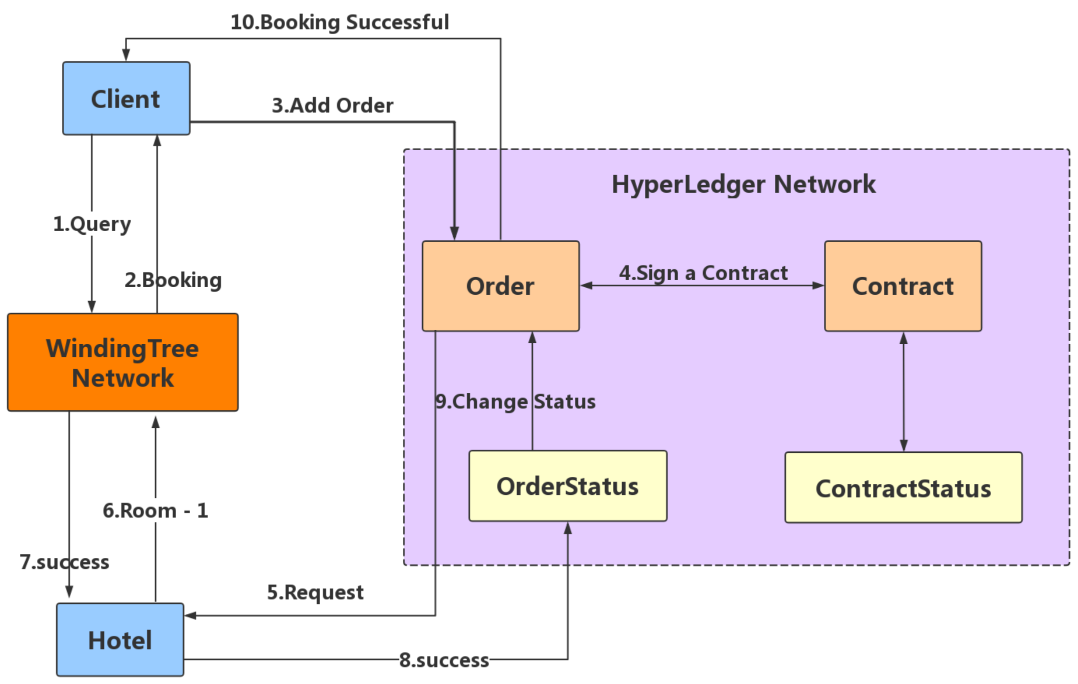


Figure 45: Hyperledger Fabric Network Order Processing Flow

The above diagram describes the workflow when a transaction occurs in the network:

1. The client queries the hotel via Windingtree network to return to the client and make a reservation.

2. The Windingtree network returns the booking information of the hotel to the client through booking API and starts booking.

3. Send an addOrder request to the Hyperledger Fabric Network, which now carries all orderlies.

4. The order generates a contract to sign with the client.

5. Initiate an order confirmation request to Hotel after signing the contract.

6. The hotel sends a request to the WindingTree network to reduce the number of rooms.

7. Successfully reduced inventory.

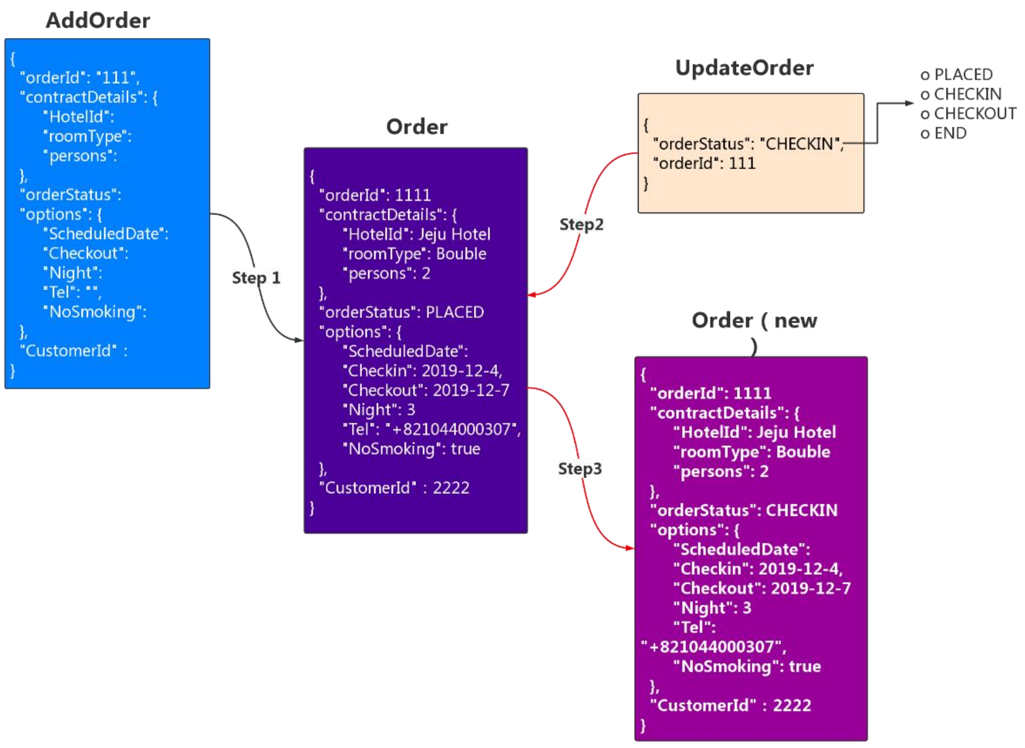
8. Change the order status.

9. If you need to send a verification to the contract when you modify the status for the second time, the contract will automatically end when Order status becomes CHECKOUT.

10. Return the message that the booking was successful.

The above process describes the complete transaction flow in the Sub-chain. The focus is on the data exchange between the Main-chain and the Sub-chain. It usually carries a return value when the function is called. After the above steps are successfully executed, the network will be notified through the return value. The data has changed. For example, in step 6, the number of wine list rooms after receiving the return value is -1. When the Main-chain network processing is completed, the return value is sent to the Sub-chain through the application. Hash encrypts the return value at this time. It is then transmitted to the other party's network to execute the smart contract.

The order status change process is implemented by calling API modification. It is worth noting that the value of the modified state is fixed in the Hyperledger Fabric network. You need to enter an exact value to modify the status. This part of the operation will be done manually by the application. When the check-out status is activated in the hotel's back office, it will be confirmed by the front desk before being written into the Sub-chain network, and the status value will be automatically changed.



```
{
  "orderId": "111",
  "contractDetails": {
     "HotelId":
     "roomType":
     "persons":
  },
  "orderStatus":
  "options": {
     "ScheduledDate":
     "Checkout":
     "Night":
     "Tel": "",
     "NoSmoking":
  },
  "CustomerId" :
}
```

```
{
  "orderId": 1111
  "contractDetails": {
     "HotelId": Jeju Hotel
     "roomType": Bouble
     "persons": 2
  },
  "orderStatus": PLACED
  "options": {
     "ScheduledDate":
     "Checkin": 2019-12-4,
     "Checkout": 2019-12-7
     "Night": 3
     "Tel": "+821044000307",
     "NoSmoking": true
  },
  "CustomerId" : 2222
}
```

```
{
  "orderStatus": "CHECKIN",
  "orderId": 111
}
```

```
{
  "orderId": 1111
  "contractDetails": {
     "HotelId": Jeju Hotel
     "roomType": Bouble
     "persons": 2
  },
  "orderStatus": CHECKIN
  "options": {
     "ScheduledDate":
     "Checkin": 2019-12-4,
     "Checkout": 2019-12-7
     "Night": 3
     "Tel":
"+821044000307",
     "NoSmoking": true
  },
  "CustomerId" : 2222
}
```

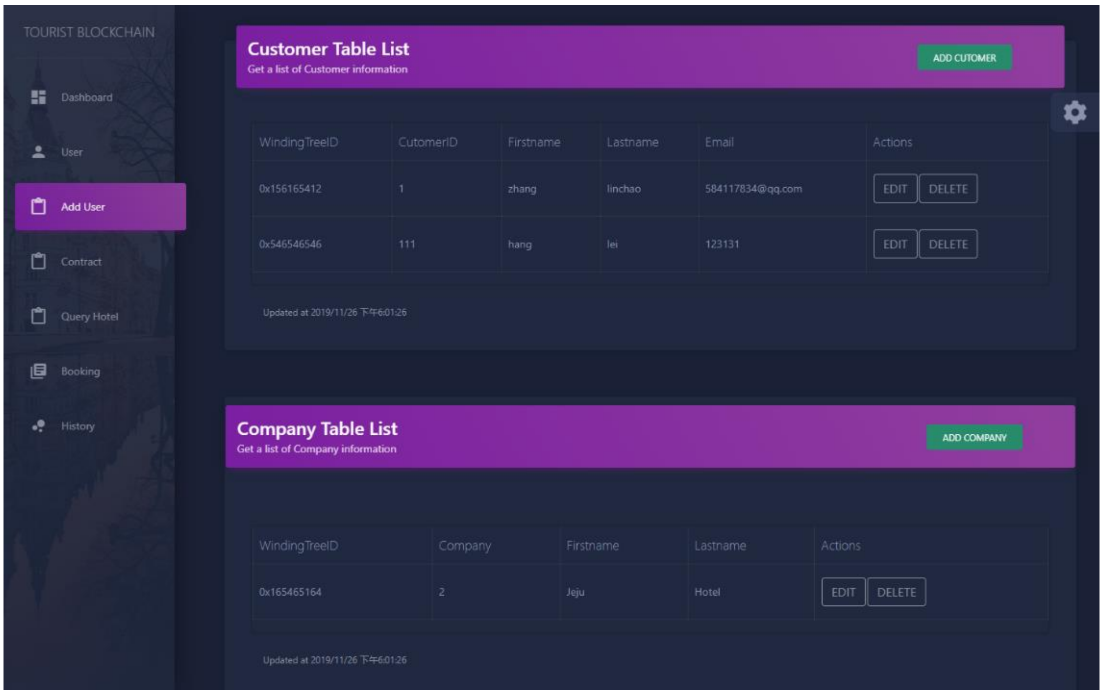
Figure 46: Hyperledger Fabric OrderStatus Status Change Process

As can be seen from the above figure, the status of Order status is modified through the operation of UpdateOrder. After the modification is successful, a new order list is returned. It is worth noting that UpdateOrder will have records in the blockchain system events. Oder and UpdateOrder are two different smart contracts. After the order status is completed, the order status change is automatically triggered, and at the same time, the status change information is automatically sent to the service provider in the Main-chain through the application, and then the contract will be canceled. So far, the entire process from booking to check-out has been entirely recorded by the two networks.

Through the above structure, we can construct a trading system in the Hyperledger Fabric Network. Can record order information, order status, and contract status. It can well complete the privacy control and transaction processing functions of the Sub-chain, solve the shortcomings in the Main-chain, and reflect the advantages of the multi-chain architecture far superior to the advantages of the Single-chain.

6.3 Sub-Chain Implementation Based on Hyperledger Fabric

In actual development, you need to pay attention to Hyperledger Fabric role control permissions when generating the network. If the permissions of all parties have been designed at the beginning of the network design, you can turn on role permission control. Otherwise, the information on the blockchain cannot be read due to insufficient permissions. Permission control is by requesting access from the CA administrator.



Customer Table List
Get a list of Customer information
ADD CUTOMER

WindingTreeID	CutomerID	Firstname	Lastname	Email	Actions
0x156165412	1	zhang	linchao	584117834@qq.com	EDIT DELETE
0x546546546	111	hang	lei	123131	EDIT DELETE

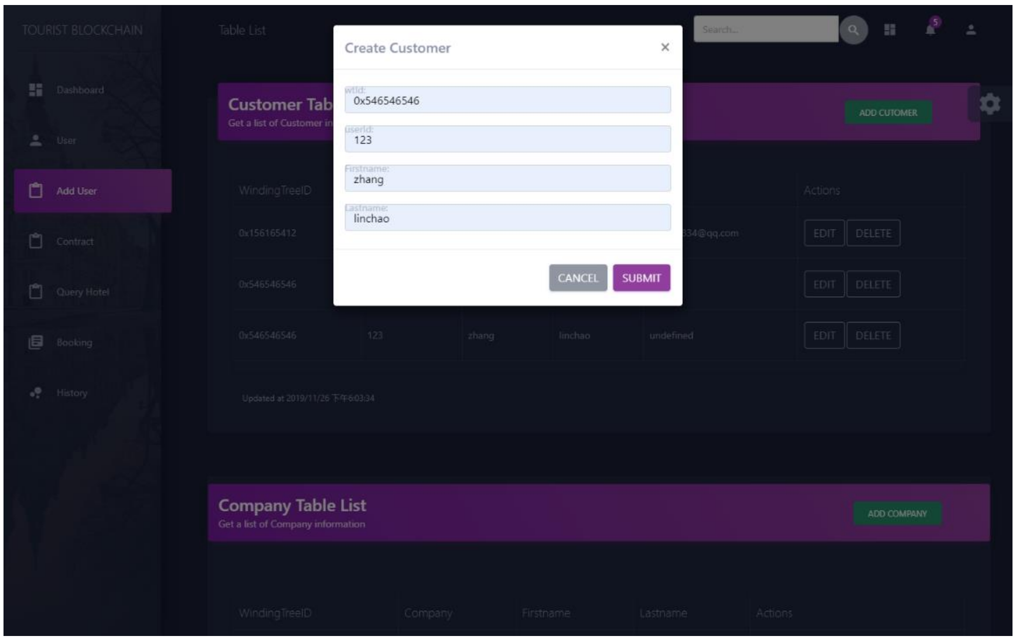
Updated at 2019/11/26 下午6:01:26

Company Table List
Get a list of Company information
ADD COMPANY

WindingTreeID	Company	Firstname	Lastname	Actions
0x165465164	2	Jeju	Hotel	EDIT DELETE

Updated at 2019/11/26 下午6:01:26

Figure 47: Hyperledger Fabric Participant Query Interface



Create Customer

wtId: 0x546546546
userId: 123
Firstname: zhang
Lastname: linchao

CANCEL SUBMIT

Customer Tab
Get a list of Customer in
ADD CUTOMER

WindingTreeID					Actions
0x156165412				34@qq.com	EDIT DELETE
0x546546546					EDIT DELETE
0x546546546	123	zhang	linchao	undefined	EDIT DELETE

Updated at 2019/11/26 下午6:03:34

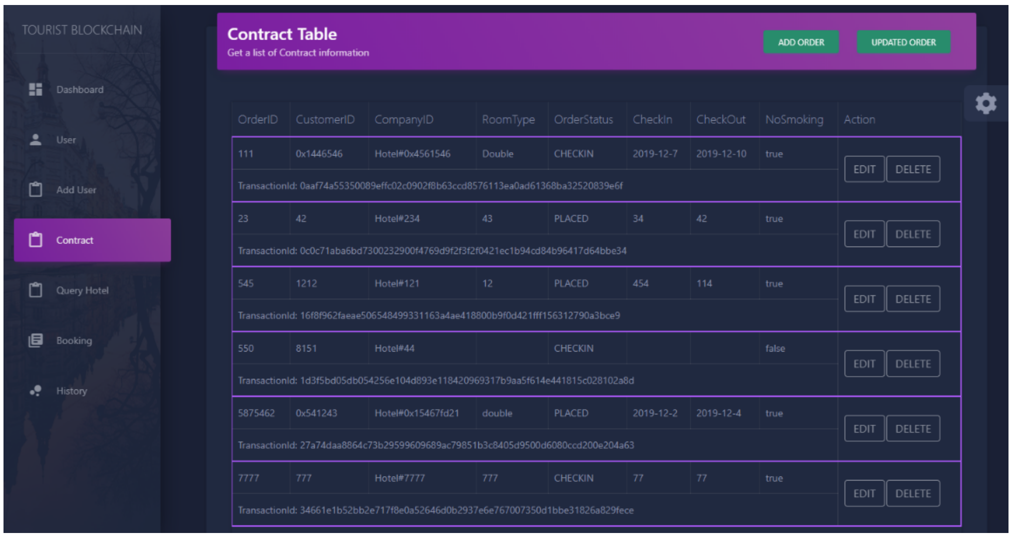
Company Table List
Get a list of Company information
ADD COMPANY

WindingTreeID	Company	Firstname	Lastname	Actions

Figure 48: Hyperledger Fabric Participant Creation Interface

Figure 47, The above picture uses the admin account to obtain the client's customer information and hotel information. The user obtains user information when logging in through the client, and supports modifying the user information. The hotel back office uses the application to modify registration information and hotel room information. When the information is modified, the records will be modified in the blockchain network process. At the same time, customers and hotels can also be created. They will register directly to the network as part of the network operation. OTA companies and hotels can use this background at the same time. Used to manage hotels and issue agency rights to OTAs. Make it in line with the market environment. Participants can also edit new information to support updates.

Figure 48, Through the application, hotels and users can register information, and the information will be registered in the network. Enter different role attributes to register participants directly in the network. The administrator issues a CA certificate to the newly registered hotel to obtain different permissions. This step also applies to the client. Because during the registration process, users are registered in the Main-chain network and the Sub-chain network through the client and obtain different usage rights. Users in the Main-chain have query and reservation permissions. Users in the Sub-chain Create orders, wake up and modify orders; these operations will be defined when issuing a certificate.



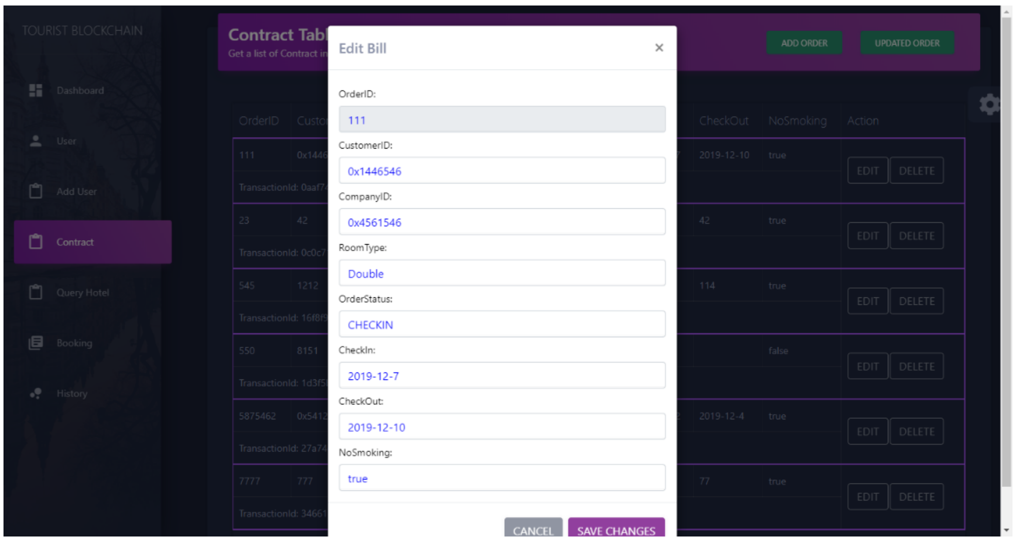
Contract Table

Get a list of Contract information

OrderID	CustomerID	CompanyID	RoomType	OrderStatus	CheckIn	CheckOut	NoSmoking	Action
111	0x1446546	Hotel#0x4561546	Double	CHECKIN	2019-12-7	2019-12-10	true	EDIT DELETE
TransactionId: 0aaf74a55350089effc02c0902f8b63ccd8576113ea0ad61368ba32520839e6f								
23	42	Hotel#234	43	PLACED	34	42	true	EDIT DELETE
TransactionId: 0c0c71aba6bd7300232900f4769d9f2f3f2f0421ec1b94cd84b96417d64bbe34								
545	1212	Hotel#121	12	PLACED	454	114	true	EDIT DELETE
TransactionId: 16f8f962faeae506548499331163a4ae418800b9f0d421fff156312790a3bce9								
550	8151	Hotel#44		CHECKIN			false	EDIT DELETE
TransactionId: 1d3f5bd05db054256e104d893e118420969317b9aa5f614e441815c028102a8d								
5875462	0x541243	Hotel#0x15467fd21	double	PLACED	2019-12-2	2019-12-4	true	EDIT DELETE
TransactionId: 27a74daa8864c73b29599609689ac79851b3c8405d9500d6080ccd200e204a63								
7777	777	Hotel#7777	777	CHECKIN	77	77	true	EDIT DELETE
TransactionId: 34661e1b52bb2e717f8e0a52646d0b2937e6e767007350d1bbe31826a829fece								

Figure 49: Hyperledger Fabric Asset Interface

The above figure realizes the creation of assets. The client obtains the hotel data information from the Main-chain network through the Booking API, sends it to the Sub-chain to create an order, and supports the modification of the order. The hotel receives the new order information through the notification API and obtains the information of the booker in the order. As an administrator, the creation and management of Order and contract. The order contains the details of the reservation information and the system event id when the order is generated. It can confirm the creation of the order. Who created the details and when, and also supports asset updates.



Edit Bill

OrderID: 111

CustomerID: 0x1446546

CompanyID: 0x4561546

RoomType: Double

OrderStatus: CHECKIN

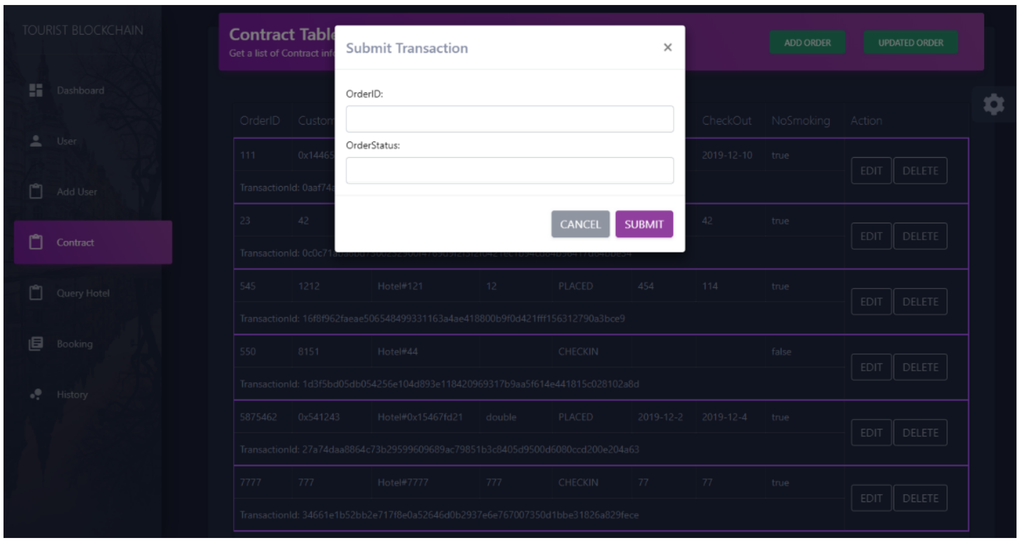
CheckIn: 2019-12-7

CheckOut: 2019-12-10

NoSmoking: true

Figure 50: Hyperledger Fabric Order Update

In the above figure, the steps to modify the order information. When creating an order, the client will use a text contract. After the user uniformly selects a new order, the user can modify the order information according to the contract rules. At the beginning of the order creation, the order status is PLACED. After checking in, the network will update the status of the order CHECKIN. After checking out, the status of the order changes to CHECKOUT. According to the actual situation, the status of the destruction contract will be END after a few days of check-ing out. The order ID (OrderID) does not support modification when modifying an order, be-cause id is automatically generated to record this order when the order is created. The ordered cannot be modified from beginning to end to prevent malicious counterfeiting.



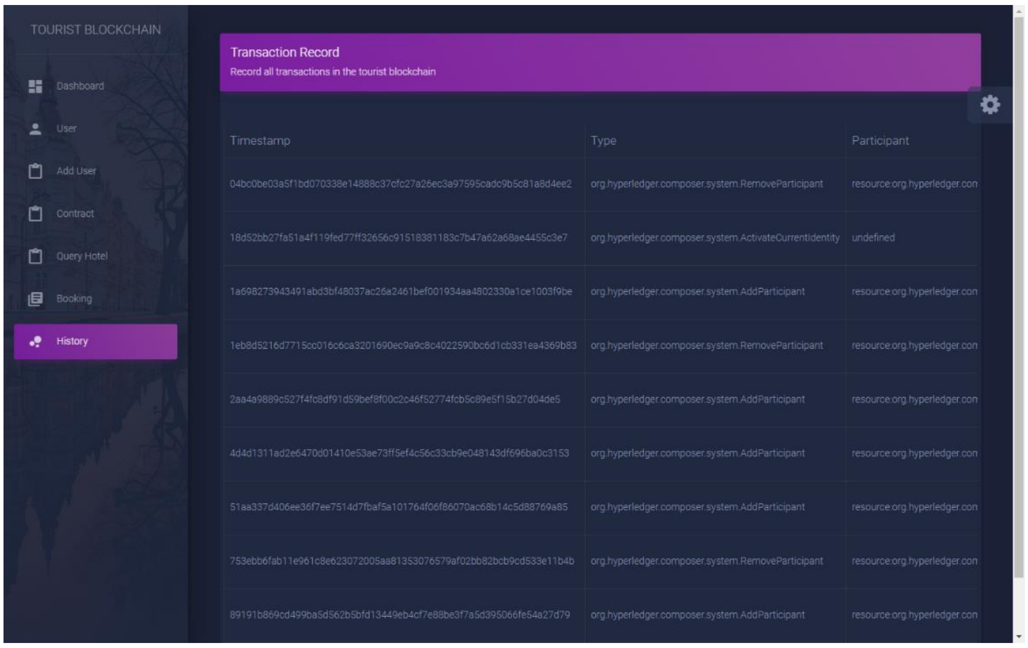
OrderID:

OrderStatus:

OrderID	Custom					CheckOut	NoSmoking	Action
111	0x14465					2019-12-10	true	EDIT DELETE
TransactionId: 0aaf74								
23	42					42	true	EDIT DELETE
TransactionId: 0c0c71								
545	1212	Hotel#121	12	PLACED	454	114	true	EDIT DELETE
TransactionId: 16f8f962faeae506548499331163a4ae418800b9f0d421fff156312790a3bce9								
550	8151	Hotel#44		CHECKIN			false	EDIT DELETE
TransactionId: 1d3f5bd05db054256e104d893e118420969317b9aa5f614e441815c028102a8d								
5875462	0x541243	Hotel#0x15467fd21	double	PLACED	2019-12-2	2019-12-4	true	EDIT DELETE
TransactionId: 27a74daa8864c73b29599609689ac79851b3c8405d9500d6080ccd200e204a63								
7777	777	Hotel#7777	777	CHECKIN	77	77	true	EDIT DELETE
TransactionId: 34661e1b52bb2e717f8e0a52646d0b2937e6e767007350d1bbe31826a829fece								

Figure 51: Hyperledger Fabric Network Update OrderStatus Status

The above figure is the status update of the order. At the same time, the order supports manual updates. For the abnormal order, directly modify the order status to destroy the order end contract. Fill in the order number and order status, respectively, and you can modify the order to the Corresponding status. The client checks the notification information through the client, and each operation on the two blocks will be recorded. As an administrator can clearly understand the details of the operation, Sub-chain can browse the history of the order through permission allocation.



Transaction Record

Record all transactions in the tourist blockchain

Timestamp	Type	Participant
04bc0be03a5f1bd070338e14888c37cfc27a26ec3a97595cadc9b5c81a8d4ee2	org.hyperledger.composer.system.RemoveParticipant	resource:org.hyperledger.con
18d52bb27fa51a4f119fed77ff32656c91518381183c7b47a62a68ae4455c3e7	org.hyperledger.composer.system.ActivateCurrentIdentity	undefined
1a698273943491abd3bf48037ac26a2461bef001934aa4802330a1ce1003f9be	org.hyperledger.composer.system.AddParticipant	resource:org.hyperledger.con
1eb8d5216d7715cc016c6ca3201690ec9a9c8c4022590bc6d1cb331ea4369b83	org.hyperledger.composer.system.RemoveParticipant	resource:org.hyperledger.con
2aa4a9889c527f4fc8df91d59bef8f00c2c46f52774fcb5c89e5f15b27d04de5	org.hyperledger.composer.system.AddParticipant	resource:org.hyperledger.con
4d4d1311ad2e6470d01410e53ae73ff5ef4c56c33cb9e048143df696ba0c3153	org.hyperledger.composer.system.AddParticipant	resource:org.hyperledger.con
51aa337d406ee36f7ee7514d7fbaf5a101764f06f86070ac68b14c5d88769a85	org.hyperledger.composer.system.AddParticipant	resource:org.hyperledger.con
753ebb6fab11e961c8e623072005aa81353076579af02bb82bcb9cd533e11b4b	org.hyperledger.composer.system.RemoveParticipant	resource:org.hyperledger.con
89191b869cd499ba5d562b5bfd13449eb4cf7e88be3f7a5d395066fe54a27d79	org.hyperledger.composer.system.AddParticipant	resource:org.hyperledger.con

Figure 52: Hyperledger Fabric Network System Log

The above figure reflects all operations in the entire Hyperledger Fabric network from the early stage of the system. This is also the reason why the blockchain cannot be counterfeited. Each contract running on the network will be recorded from the beginning to the end of the entire cycle. In the Sub-chain network, each operation generates an eventID, which records all the information at that time. Usually, this part of the operation is viewed by the network administrator, which can be achieved through rights management. The historical record contains event ID, operation object, modification result, identity record, and event creation event. It contains the history of the entire cycle of the blockchain.

Chapter7 Performance Evaluation

This section demonstrates the performance evaluation results of the Sub-chain based on Hyperledger Fabric by varying the number of clients and the number of transactions. The number of clients is varied from 5 to 10, while the number of transactions is varied from 5000 to 10,000. The back-end performance of the blockchain network is tested through open and query functions. Table 7-1 presents the send rate, network latency (seconds), and the throughput (transaction per second) when the number of clients is 5.

For the first-round test, 5000 open transactions are performed in the send rate of 349.3tps, the minimum time taken was recorded to be 0.48s, averaging at 2.01s, and the maximum delay was recorded to be 3.51s. For the second-round test, 5000 open transactions are performed at the send rate of 493.8tps, the minimum time taken was recorded to be 0.33s, averaging at 4.82s, and the maximum delay was recorded to be 8.06s. For the third -round test, 10000 open transactions are performed at the send rate of 399.2tps, the minimum time taken was recorded to be 0.61s, averaging at 4.37s, and the maximum delay was recorded to be 7.44s. For the fourth-round test, 10000 open transactions are performed at the send rate of 498.3tps, the minimum time taken was recorded to be 0.31s, averaging at 8.64s, and the maximum delay was recorded to be 15.37s.

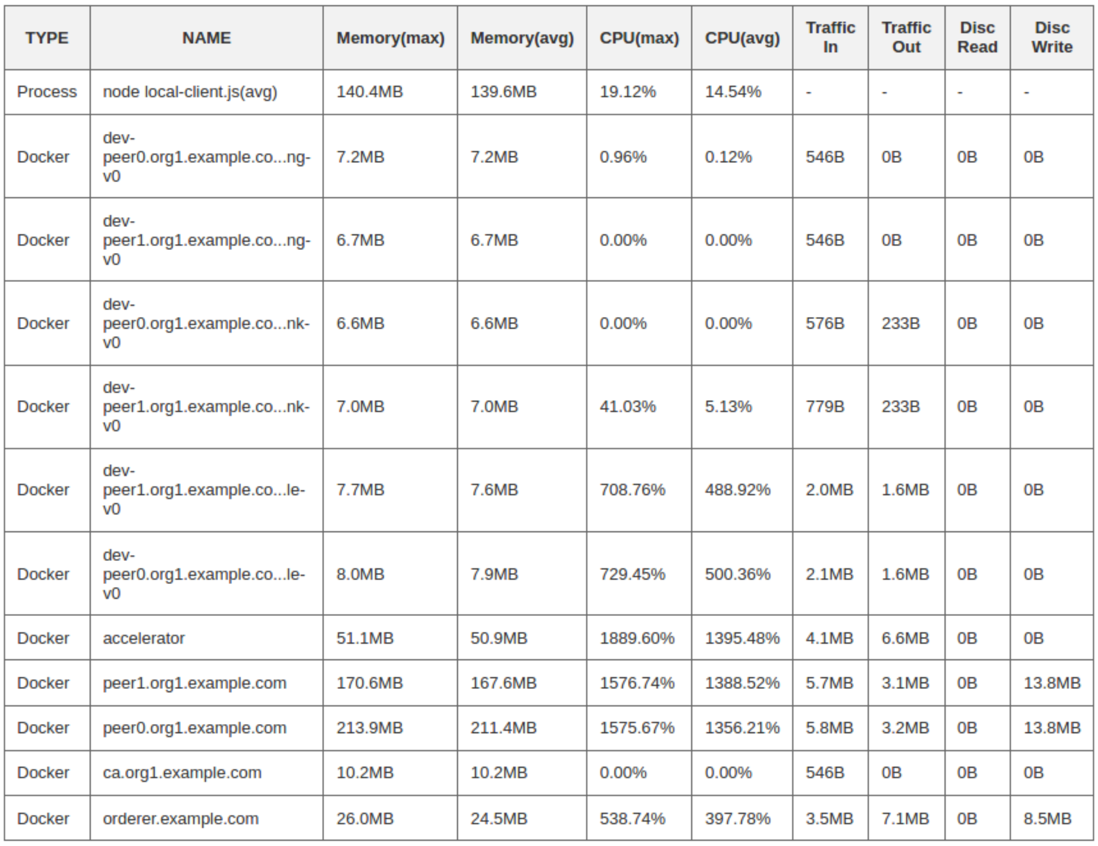
For the fifth-round test, 5000 query transactions are performed in the send rate of 871.8tps, the minimum time taken was recorded to be 0.05s, averaging at 0.85s, and the maximum delay was recorded to be 2.70s. For the sixth-round test, 5000 query transactions are performed at the send rate of 913.4tps, the minimum time taken was recorded to be 0.05s, averaging at 1.96s, and the maximum delay was recorded to be 4.71s. For the seventh-round test, 10000 query transactions are performed at the send rate of 880.5tps, the minimum time taken was recorded to be 0.10s, averaging at 1.73s, and the maximum delay was recorded to be 7.01s. For the eighth-round test, 10000 query transactions are performed at the send rate of 979.4tps, the minimum time taken was recorded to be 0.06s, averaging at 1.43s, and the maximum delay was recorded to be 5.17s.

Table 7-1: Performance Analysis Summary on Clients (Client = 5)

Name	Succ	Fail	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput
open	5000	0	349.3 tps	3.51 s	0.48 s	2.01 s	328.1 tps
open	5000	0	493.8 tps	8.06 s	0.33 s	4.82 s	350.8 tps
open	10000	0	399.2 tps	7.44 s	0.61 s	4.37 s	350.0 tps
open	10000	0	498.3 tps	15.37 s	0.31 s	8.64 s	363.2 tps
query	5000	0	871.8 tps	2.70 s	0.05 s	0.85 s	798.8 tps
query	5000	0	913.4 tps	4.71 s	0.05 s	1.96 s	747.0 tps
query	10000	0	880.5 tps	7.01 s	0.10 s	1.73 s	826.7 tps
query	10000	0	979.4 tps	5.17 s	0.06 s	1.43 s	910.1 tps

Table 7-2 describes the resource utilization test results in various performance indexes. This table presents the maximum, average memory, and central processing unit (CPU) utilization rate by the blockchain network when the number of clients is 5. Each entity of the blockchain network runs in the Docker environment, and the utilization of memory and CPU are shown in the table. For the local-client, the maximum memory allocation taken in the ten iterations was recorded to be 140.4 MB, averaging at 139.6 MB, and the maximum CPU utility was recorded to be 19.12%, averaging at 14.54%. For each peer, the average maximum memory allocation taken in the ten iterations was recorded to be 192.25 MB, averaging at 189.5 MB. For the orderer node, the average maximum memory allocation taken in the ten iterations was recorded to be 26.0 MB, averaging at 24.5 MB. For the CA node, the average maximum memory allocation taken in the ten iterations was recorded to be 10.2 MB, averaging at 10.2 MB, and the CPU utilization was recorded to be 0%.

Table 7-2: Network Consumption of Hardware (clients = 5).



TYPE	NAME	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Process	node local-client.js(avg)	140.4MB	139.6MB	19.12%	14.54%	-	-	-	-
Docker	dev-peer0.org1.example.co...ng-v0	7.2MB	7.2MB	0.96%	0.12%	546B	0B	0B	0B
Docker	dev-peer1.org1.example.co...ng-v0	6.7MB	6.7MB	0.00%	0.00%	546B	0B	0B	0B
Docker	dev-peer0.org1.example.co...nk-v0	6.6MB	6.6MB	0.00%	0.00%	576B	233B	0B	0B
Docker	dev-peer1.org1.example.co...nk-v0	7.0MB	7.0MB	41.03%	5.13%	779B	233B	0B	0B
Docker	dev-peer1.org1.example.co...le-v0	7.7MB	7.6MB	708.76%	488.92%	2.0MB	1.6MB	0B	0B
Docker	dev-peer0.org1.example.co...le-v0	8.0MB	7.9MB	729.45%	500.36%	2.1MB	1.6MB	0B	0B
Docker	accelerator	51.1MB	50.9MB	1889.60%	1395.48%	4.1MB	6.6MB	0B	0B
Docker	peer1.org1.example.com	170.6MB	167.6MB	1576.74%	1388.52%	5.7MB	3.1MB	0B	13.8MB
Docker	peer0.org1.example.com	213.9MB	211.4MB	1575.67%	1356.21%	5.8MB	3.2MB	0B	13.8MB
Docker	ca.org1.example.com	10.2MB	10.2MB	0.00%	0.00%	546B	0B	0B	0B
Docker	orderer.example.com	26.0MB	24.5MB	538.74%	397.78%	3.5MB	7.1MB	0B	8.5MB

Table 7-3 presents the send rate, network latency (seconds), and the throughput (transaction per second) when the number of clients is 10. For the first-round test, 5000 open transactions are performed in the send rate of 302.2tps, the minimum time taken was recorded to be 0.29s, averaging at 4.61s, and the maximum delay was recorded to be 7.32s. For the second-round test, 5000 open transactions are performed at the send rate of 494.0tps, the minimum time taken was recorded to be 0.91s, averaging at 7.36s, and the maximum delay was recorded to be 10.19s. For the third -round test, 10000 open transactions are performed at the send rate of 393.1tps, the minimum time taken was recorded to be 1.53s, averaging at 12.43s, and the maximum delay was recorded to be 16.67s. For the fourth-round test, 10000 open transactions are performed at the send rate of 480.7ps, the minimum time taken was recorded to be 0.93s, averaging at 11.67s, and the maximum delay was recorded to be 16.93s.

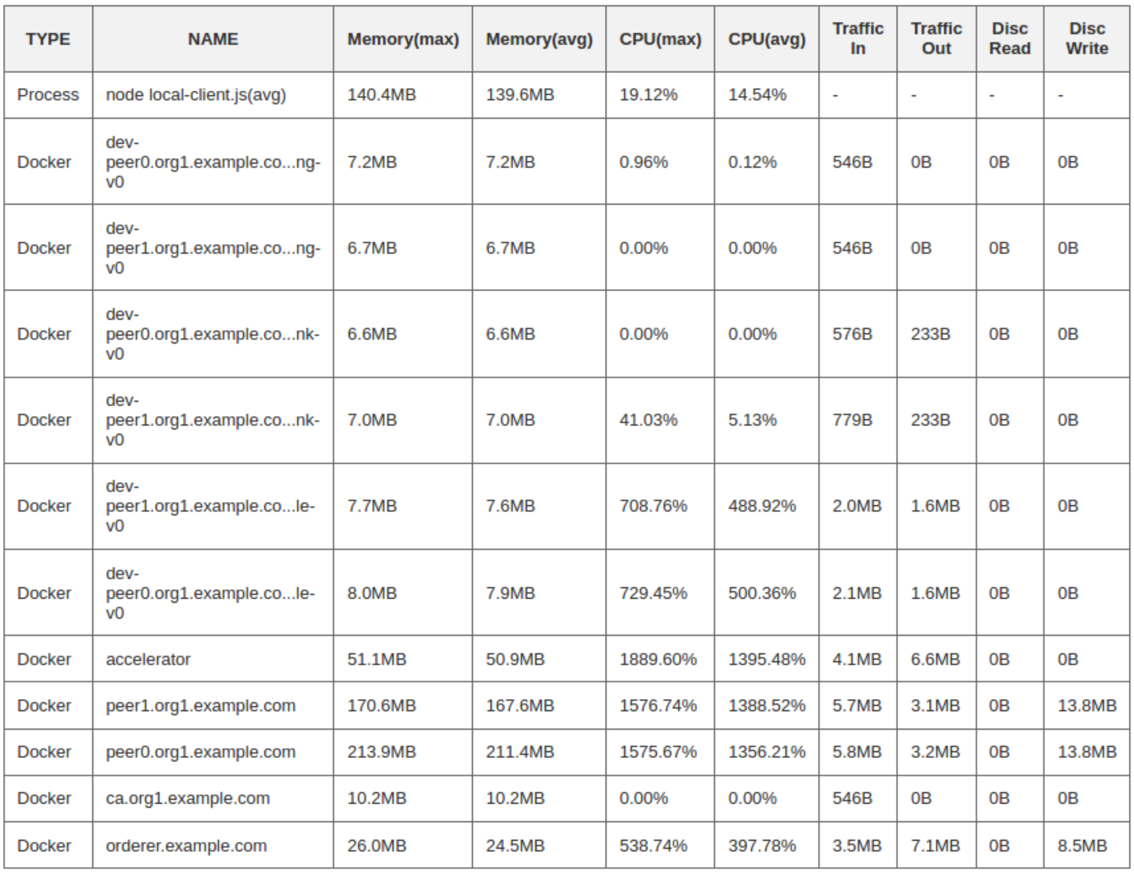
For the fifth-round test, 5000 query transactions are performed in the send rate of 740.0tps, the minimum time taken was recorded to be 0.14s, averaging at 2.66s, and the maximum delay was recorded to be 5.65s. For the sixth-round test, 5000 query transactions are performed at the send rate of 826.7tps, the minimum time taken was recorded to be 0.03s, averaging at 2.32s, and the maximum delay was recorded to be 5.83s. For the seventh -round test, 10000 query transactions are performed at the send rate of 859.1tps, the minimum time taken was recorded to be 0.04s, averaging at 2.68s, and the maximum delay was recorded to be 8.14s. For the eighth-round test, 10000 query transactions are performed at the send rate of 955.7tps, the minimum time taken was recorded to be 0.06s, averaging at 3.86s, and the maximum delay was recorded to be 10.70s.

Table 7-3: Performance Analysis Summary on Clients (Client = 10)

Name	Succ	Fail	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput
open	5000	0	302.2 tps	7.32 s	0.29 s	4.61 s	250.7 tps
open	5000	0	494.0 tps	10.19 s	0.91 s	7.36 s	289.6 tps
open	10000	0	393.1 tps	16.67 s	1.53 s	12.43 s	278.0 tps
open	10000	0	480.7 tps	16.93 s	0.93 s	11.67 s	321.0 tps
query	5000	0	740.0 tps	5.65 s	0.14 s	2.66 s	588.6 tps
query	5000	0	826.7 tps	5.83 s	0.03 s	2.32 s	695.0 tps
query	10000	0	859.1 tps	8.14 s	0.04 s	2.68 s	752.4 tps
query	10000	0	955.7 tps	10.70 s	0.06 s	3.86 s	766.5 tps

Table 7-4 describes the resource utilization test results in various performance indexes. This table presents the maximum, average memory, and central processing unit (CPU) utilization rate by the blockchain network when the number of clients is 10. Each entity of the blockchain network runs in the Docker environment, and the utilization of memory and CPU are shown in the table. For the local-client, the maximum memory allocation taken in the ten iterations was recorded to be 140.4 MB, averaging at 139.6 MB, and the maximum CPU utility was recorded to be 19.12%, averaging at 14.54%. For each peer, the average maximum memory allocation taken in the ten iterations was recorded to be 192.25 MB, averaging at 189.5 MB. For the orderer node, the average maximum memory allocation taken in the ten iterations was recorded to be 26.0 MB, averaging at 24.5 MB. For the CA node, the average maximum memory allocation taken in the ten iterations was recorded to be 10.2 MB, averaging at 10.2 MB, and the CPU utilization was recorded to be 0%.

Table 7-4: Network Consumption of Hardware (clients = 10).



TYPE	NAME	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Process	node local-client.js(avg)	140.4MB	139.6MB	19.12%	14.54%	-	-	-	-
Docker	dev-peer0.org1.example.co...ng-v0	7.2MB	7.2MB	0.96%	0.12%	546B	0B	0B	0B
Docker	dev-peer1.org1.example.co...ng-v0	6.7MB	6.7MB	0.00%	0.00%	546B	0B	0B	0B
Docker	dev-peer0.org1.example.co...nk-v0	6.6MB	6.6MB	0.00%	0.00%	576B	233B	0B	0B
Docker	dev-peer1.org1.example.co...nk-v0	7.0MB	7.0MB	41.03%	5.13%	779B	233B	0B	0B
Docker	dev-peer1.org1.example.co...le-v0	7.7MB	7.6MB	708.76%	488.92%	2.0MB	1.6MB	0B	0B
Docker	dev-peer0.org1.example.co...le-v0	8.0MB	7.9MB	729.45%	500.36%	2.1MB	1.6MB	0B	0B
Docker	accelerator	51.1MB	50.9MB	1889.60%	1395.48%	4.1MB	6.6MB	0B	0B
Docker	peer1.org1.example.com	170.6MB	167.6MB	1576.74%	1388.52%	5.7MB	3.1MB	0B	13.8MB
Docker	peer0.org1.example.com	213.9MB	211.4MB	1575.67%	1356.21%	5.8MB	3.2MB	0B	13.8MB
Docker	ca.org1.example.com	10.2MB	10.2MB	0.00%	0.00%	546B	0B	0B	0B
Docker	orderer.example.com	26.0MB	24.5MB	538.74%	397.78%	3.5MB	7.1MB	0B	8.5MB

Chapter8 Conclusion

This paper proposes a multi-chain architecture, with a public chain as the main-chain and a private chain as the sub-chain. We demonstrate the disadvantages of the existing single-chain network (low performance and poor scalability) and elaborate on the advantages of a multi-chain architecture model. To improve network performance, each single-chain is implemented as a local service that fills the business requirements. These single chains are combines together to form a multi-chain network in parallel.

This paper applies multi-chain architecture technology to the tourism industry. The public chain is implemented on WindingTree, and the private chain is built on Hyperledger Fabric. The WindingTree network is responsible for adding, deleting, modifying, and reading operations in the business system, and providing corresponding functions for different business roles. The Hyperledger Fabric network is responsible for recording transaction information and signing contracts in the transaction system. The data is encrypted transmitted through the routing management application between different chains. A series of experiments are conducted and the results indicate the applicability and feasibility of the multi-chain architecture. To east the development, this paper does not apply the use of token in the transaction. Due to the different development directions of different blockchain networks, there is a demand for a unified token to construct the blockchain ecosystem in the future.

References

1. Wood, G. (2016). Polkadot: Vision for a heterogeneous multi-chain framework. White Paper.
2. G.Maxwell,Deterministicwallets, 2011,BitcoinTalkpost,https://bitcointalk.org/index.php?topic=19137.0.
3. G. Andresen, BIP16: Pay to script hash, Bitcoin Improvement Proposal, 2012, https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki
4. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. bitcoin.org, 2008.
5. Chen Z. Research on Private Blockchain Based on Crowdfunding[J]. Journal of Information Security Research, 2017, 3(3): 227-236.
6. www://usa.visa.com/merchants/industry-solutions/retail-visaacceptance.jsp
7. Castro M. Practical byzantine fault tolerance and proactive recovery [J]. ACM Trans on Computer Systems (TOCS), 2002, 20(4):398-461.
8. Hu K, Zhang T, Yang Z, et al. Exploring AADL Verification Tool through Model Transformation[J]. Journal of Systems Architecture, 2015, 61(3–4):141-156.
9. WindingTree https://windingtree.com/
10. Chinese Blockchain Technology Application and Development White Paper
11. G. Andresen, BIP16: Pay to script hash, Bitcoin Improvement Proposal, 2012, https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.
12. Iansiti, M., & Lakhani, K. R. (2017). The truth about blockchain. Harvard Business Review, 95(1), 118-127.
13. Zyskind, G., & Nathan, O. (2015, May). Decentralizing privacy: Using blockchain to protect personal data. In 2015 IEEE Security and Privacy Workshops (pp. 180-184). IEEE.
14. Pilkington, M. (2016). 11 Blockchain technology: principles and applications. Research handbook on digital transformations, 225.
15. Gupta, S. S. (2017). Blockchain. John Wiley & Sons, Inc.
16. Tapscott, D., & Tapscott, A. (2017). How blockchain will change organizations. MIT Sloan Management Review, 58(2), 10.

2. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017, June). An overview of blockchain technology: Architecture, consensus, and future trends. In 2017 IEEE International Congress on Big Data (BigData Congress) (pp. 557-564). IEEE.

3. Rodrigues, B., Bocek, T., Lareida, A., Hausheer, D., Rafati, S., & Stiller, B. (2017, July). A blockchain-based architecture for collaborative DDoS mitigation with smart contracts. In IFIP International Conference on Autonomous Infrastructure, Management and Security (pp. 16-29). Springer, Cham.
4. Biswas, K., & Muthukkumarasamy, V. (2016, December). Securing smart cities using blockchain technology. In 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS) (pp. 1392-1393). IEEE.
5. Hileman, G., & Rauchs, M. (2017). Global blockchain benchmarking study. Cambridge Centre for Alternative Finance, University of Cambridge, 122.
6. Gupta, S. S. (2017). Blockchain. John Wiley & Sons, Inc.
7. Sousa, J., Bessani, A., & Vukolic, M. (2018, June). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN) (pp. 51-58). IEEE.
8. A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine fault tolerant systems tolerate Byzantine faults. In Proc. 6th Symp. Networked Systems Design and Implementation (NSDI), pages 153–168, 2009.
9. Hyperledger fabric architecture explainedhttps://blog.csdn.net/russell_tao/article/details/80459698
10. Explanation of key components of Hyperledger Hyperledge https://www.cnblogs.com/si812cn/p/9723442.html
11. Sukhwani, H., Martínez, J. M., Chang, X., Trivedi, K. S., & Rindos, A. (2017, September). Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS) (pp. 253-255). IEEE.
12. Polukhina, A., Arnaberdiyev, A., & Tarasova, A. (2019, May). Leading technologies in tourism: using blockchain in TravelChain project. In 3rd International Conference on Social, Economic, and Academic Leadership (ICSEAL 2019). Atlantis Press.
13. Schulte, S., Sigwart, M., Frauenthaler, P., & Borkowski, M. (2019, September). Towards Blockchain Interoperability. In International Conference on Business Process Management (pp. 3-10). Springer, Cham.

14. Dong, Z., Lee, Y. C., & Zomaya, A. Y. (2019). Proofware: Proof of Useful Work Blockchain Consensus Protocol for Decentralized Applications. arXiv preprint arXiv:1903.09276.
15. Ann-Router.[EB/OL].2017[2017-0301].http://7i7ifh.com1.z0.glb.clouddn.com/whitepaper.pdf
16. Mcquillan J, Richer I, Rosen E. The New Routing Algorithm for the ARPANET [J]. IEEE Transactions on Communications, 1980, 25(5):711-719.
17. L. Lamport, Constructing digital signatures from a one-way function, Tech. Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
18. L. Kan, Y. Wei, A. Hafiz Muhammad, W. Siyuan, G. Linchao and H. Kai, "A Multiple Blockchains Architecture on Inter-Blockchain Communication," 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, 2018, pp. 139-145.
19. J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In Advances in Cryptology: Eurocrypt 2015, volume 9057 of Lecture Notes in Computer Science, pages 281–310. Springer, 2015.
20. Hyperledger modeling language. https://hyperledger.github.io/composer/latest/reference/cto_language.html
21. Analysis of the Blockchain Protocol in Asynchronous Networks Rafael Pass and Lior Seeman and abhi shelat Eurocrypt 2017 http://eprint.iacr.org/2016/454