applied sciences

Article

SLA-Based Sharing Economy Service with Smart Contract for Resource Integrity in the Internet of Things

Lei Hang and Do-Hyeun Kim *

Department of Computer Engineering, Jeju National University, Jeju 63243, Korea
* Correspondence: kimdh@jejunu.ac.kr

Received: 5 August 2019; Accepted: 23 August 2019; Published: 2 September 2019

Abstract: Recently, technology startups have leveraged the potential of blockchain-based technologies to govern institutions or interpersonal trust by enforcing signed treaties among different individuals in a decentralized environment. However, it is going to be hard enough convincing that the blockchain technology could completely replace the trust among trading partners in the sharing economy as sharing services always operate in a highly dynamic environment. With the rapid expanding of the rental market, the sharing economy faces more and more severe challenges in the form of regulatory uncertainty and concerns about abuses. This paper proposes an enhanced decentralized sharing economy service using the service level agreement (SLA), which documents the services the provider will furnish and defines the service standards the provider is obligated to meet. The SLA specifications are defined as the smart contract, which facilitates multi-user collaboration and automates the process with no involvement of the third party. To demonstrate the usability of the proposed solution in the sharing economy, a notebook sharing case study is implemented using the Hyperledger Fabric. The functionalities of the smart contract are tested using the Hyperledger Composer. Moreover, the efficiency of the designed approach is demonstrated through a series of experimental tests using different performance metrics.

Keywords: service level agreement; sharing economy; blockchain; smart contract; Hyperledger Fabric

1. Introduction

Due to the popularity of peer-to-peer (P2P) platforms, the business dealings among private individuals on a large scale are increasingly frequent. The sharing economy is a P2P-based economic model, which provides access to goods and services that are often facilitated by a community-based online platform [1]. This new Internet-based rental market allows individuals to get an extra benefit by renting out their unused goods and services to others. For example, Mobike [2] is a fully station-less bicycle sharing system, which enables consumers to pick up and leave a bike at any time or any parking area.

Another typical example is Airbnb [3], in which the individuals are allowed to rent out their spare rooms, apartments, or entire homes. The sharing economy has rapidly exploded over the past decade, and the potential market size is expected to grow from 14 billion in 2014 to 335 billion dollars in 2025 [4]. The advancement of information technologies and data analytics capabilities pave the way for rental firms to match the demand and supply efficiently. Mutual trust is one of the most fundamental prerequisites for such interactions to take place in the sharing economy [5]. For example, the host of the department not only needs to trust potential guests to behave respectfully but also to believe in Airbnb's ability relative to the reservation and payment procedures. However, with reduced social ties compared to other common sharing platforms, providers, and consumers in the sharing economy typically know less of each other than they would in a conventional exchange. As a result, many sharing economy platforms introduced rating systems [6–8] to distribute the cost of adjusting the platform to members. Although the trust represented in user ratings might be significant, individual participants disproportionately bear the risk.

Recently, the decentralized technology represented by blockchain has been introduced as a new paradigm by some researchers [9–11]. In concept, blockchain is a decentralized, distributed ledger consisting of multiple peers, which can execute, track, verify transactions, and record transactions across a large variety of entities [12]. Every block of the ledger is logically linked to the previous block by storing the hash of the parent. As a consequence, the user of the blockchain can easily use the verification process to detect the discrepancy even if a single bit of the ledger has been tampered. This stunning property makes it impossible for any single party to manipulate data on the blockchain. The economic sharing system built on the blockchain technology, makes a transaction trust-free, thereby increasing the efficiency of the sharing service and lowering the operational costs.

Though blockchain started as a core technology of Bitcoin, its use cases have already been expanded to various areas [13], such as healthcare [14–17], intelligent transportation [18,19], and the Internet of Things (IoT) [20–23]. Despite the great efforts made by many researchers to apply the theory of the blockchain technology in the sharing economy, the authors in [24] indicate the existing defects of blockchain-based sharing economy systems by reviewing recent literature and research analysis. The limitations can be summarized as the following points: Firstly, most of the systems primarily concentrate on the online trading and transaction transparency. Moreover, they only provide simple services like transferring the ownership of the property by using cryptocurrencies (e.g., Bitcoins or Ethers). Few works of the literature consider the actual connections between the blockchain with the physical world so far. Most important of all, the major problem is concerned with the reduced operational governance control as the consumer has less control of the actual service level being offered by the provider compared to the on-premise services. In general, businesses offering rental services are always regulated by federal, state, or local authorities. However, in the sharing economy, unlicensed individuals offering rental services may not follow these regulations. For example, there are stories about users who find their bikes provided by the Mobike owners heavily damaged, with little chance of compensation.

To establish an operational governance control in the sharing economy, this paper utilizes the concept of SLA [25], which describes the obligations that service providers must comply with when delivering a specific service to consumers. For instance, in the cloud service, an SLA can state that the network throughput must be higher than 100 Mbps; otherwise, the consumer receives 50% of its payment back. The SLA management is responsible for the SLA template generation, negotiation, configuration, enforcement, maintenance, and evolution. For most of the existing business support systems (BSS), the SLA compensation, still requires manual effort and interaction to be accomplished. For example, if a violation occurs during the term of the SLA contract, the compensation process is bureaucratic, and involves dedicated personnel in the case of a dispute. Moreover, such manual interaction hinders service agility and is prone to errors. This paper proposes a novel mechanism to manage SLA by relying on smart contracts [26] and the blockchain. A smart contract is defined as contractual clauses, for instance, collateral and property, which can be embedded in the hardware and software. With the advance of software technologies, the smart contract is no longer related to the conventional concept of a deal but can be any computer code running on top of the system. In light of this statement, the Ethereum Foundation [27] is the earliest start in a smart contract that takes place on the blockchain. The smart contract contains a set of rules under which the parties to that smart contract agree to interact with each other. The agreement is automatically enforced by the smart contract only when the condition meets the pre-defined rules. This technology brings foreseeable benefits to the SLA management, such as increased business efficiency, better security, task coordination, and improved customer satisfaction.

Overall, the main contributions of this work are multi-fold: Firstly, we apply the concept of SLA to build the self-governance control in the sharing economy system. Moreover, we define the SLA specifications as the smart contract, which automates the SLA lifecycle and brings transparency to the service provisioning since all rules for the SLA management. Market barriers can be lowered as there is no third-party intervention. Lastly, we implement a notebook sharing case study on top of the proposed architecture by using a permissioned blockchain network called Hyperledger Fabric.

The rest of this paper is structured as follows: Section 2 discusses the related work. Section 3 illustrates the proposed system architecture and details each internal component of the system. Section 4 elaborates the implementation of the case study in depth. Section 5 presents the implementation results of the case study with numerous snapshots. Section 6 analyzes the system performances of the case study in different performance indexes. Finally, Section 7 summarizes the full paper and indicates future research directions.

2. Related Work

Recently, there has been considerable interest in the shared economy applications, whereby people can monetize their things [28]. The sharing economy has experienced waves of hyper enthusiasm, but it has yet to reach its full potential. Most of the business model of platforms such as Airbnb and Uber play a role as intermediaries between users and private resources. The blockchain is stated to provide an infrastructure with the potential to organize truly decentralized markets [29]. A recent discussion paper by IBM reported that blockchains have the potential to create a "sharing economy 2.0" by decentralizing trust [30]. Additionally, much of the academic literature suggests the blockchain technology to overcome trust-related issues and hence to contribute to the resolution of one of the fundamental challenges of the peer-to-peer markets and sharing economy activities [31]. Beenest [32] is one startup that aims at improving the house-sharing economy by eliminating the terms of service and costs incurred by the centralized authorities using the blockchain technology. In this new model, those hosts with spare rooms or houses can directly connect with the users looking for accommodation, without intermediary charging or exploiting its user's data. SLOCK.IT [33] is an example that specialized in blockchain and IoT applications to realize Szabo's vision of smart contracts embedded in IoT-enabled devices. The owner of a SLOCK can set a deposit for his property for rental, and thereby the user pays the deposit through the Ethereum blockchain to get access permission to control the smart lock via their smartphone. Helbiz [34] is a critical proponent of this economic model for transportation by combining the car-sharing and blockchain technology. The blockchain serves as a conduit for the services provided by Helbiz's platform, automatically processing transactions between operators, owners, and external service providers. Widely publicized as the "blockchain version of Uber," La'zooz' aims to build an open-source, worldwide, decentralized ride-sharing network, to challenge and revolutionize the established private transpiration systems with large numbers of wasted empty seats and cargo space [35]. Arcade City [36] is another decentralized ride-sharing service built on the Ethereum blockchain. It links up drivers and passengers and provides a ride in the same manner Uber does. Except for these startups, some design science approaches pick up the notion of the blockchain and transfer the concept of intermediary-free platforms to potential application contexts. For instance, [37] proposes a proof of concept to demonstrate how decentralized applications enable every day sharing. The authors built the smart contract on the smart contract, which allows owners to register and rent devices without a trusted third-party involvement. However, a blockchain-based system leaks privacy information of the involved parties due to its openness to the public, and a privacy-respecting approach is proposed to enforce agreements between owners and actual users of goods [38]. The authors in [39] introduce a decentralized trust-free transaction system that allows users to transfer real-world assets without a central registry. Moreover, they propose to utilize the blockchain's publicly shared and immutable record to mitigate the impact of informational asymmetries on bilateral transactions. The authors in [40] propose a blockchain-based infrastructure for the IoT-enabled sharing economy in mega smart cities. The proposed infrastructure leverages the cognitive computing and off-chain based decentralized data storage for a massive crowd. The authors in [41] present a distributed solution to protect the privacy of users and to the security of the vehicle ecosystem. The proposed security architecture is eminently suitable for car-sharing services, which require a trusted communication channel to exchange data including the location of the vehicle, keys to unlock the car, and payment details of the user.

Although considerable works have been conducted, these works fall short of the QoS for service consumers when contracting third-party resources, especially business ones that require guarantees for outsourcing business processes. For example, most of these works only focus on the management of payment but lack the compensation process in the event of default. To address these issues, an automatic approach for service quality assurance and service control automatic by using SLA is presented herein. As far as we know, this is the first study dedicated to exploring the use of the smart contract to automate the SLA management process in sharing economy systems.

3. The Architecture of SLA based Sharing Economy Service

3.1. SLA Based Sharing Economy Service

As shown in Figure 1, the proposed sharing economy service consists of four components: Participants, assets, blockchain network, and off-chain data lake. Participants and assets are the basis of the sharing business model. A participant is an actor with unique roles, and for example, they can be the provider of the assets or the user who wants to search and rent assets according to certain conditions automatically. Assets are goods, services, or properties, which can be controlled by electronics locks or access control systems. Assets can represent almost anything in a business network, for example, a vacant house or an idle desktop. To monitor the status of these assets, we can collect various sensor data from sensors that are attached to assets. Participants can perform operations on the blockchain, for example, creating a new asset or exchange assets with other participants. All the interactions between the participants and the blockchain are encrypted with a digital signature to ensure the security of data transmission and the authenticity of the participant's identity. The blockchain network consists of various trusted peers, which hold the smart contract and a copy of the ledger for the network to maintain the data consistency within the network. The smart contract acts as a trusted distributed application that performs business logic running on the blockchain. It can perform either simple operations such as a data update, or complex operation that requires attached conditions. The blockchain preserves a complete up to data history of all transactions, which would follow a lifecycle of a specific asset. The off-chain data lake is an independent data repository, which holds the current values of participants and assets.

As shown in Figure 2, we insulate the submitting clients from the blockchain network. They are only used to invoke a smart contract and to receive a notification whenever the blockchain network includes a new transaction in the ledger. Participants are members of the business network, with the abilities to have assets and submit transactions. The assets can also invoke smart contracts in the case of sending sensing data collected from them. These data are used to produce a variety of ways for services such as the remote fault diagnosis and maintenance. The participant registry holds the instances of generated participants while the asset registry holds the instances of assets. In this work, we use the enrollment certificates as the mappings of identities to participants, and these certificates are maintained in the identity registry. The blockchain network creates a new mapping to the identity registry whenever the admin bounds an identity to a participant.

As a consequence, the network can verify the identity in the identity registry when a participant uses that identity to submit transactions. After the identity verification, the participant becomes the current participant who can submit the transaction. The access control in the smart contract manages access to resources by associating a policy. These rules in the blockchain network are applied to the current participant, who performs the authorized operations on resources according to access control requirements.

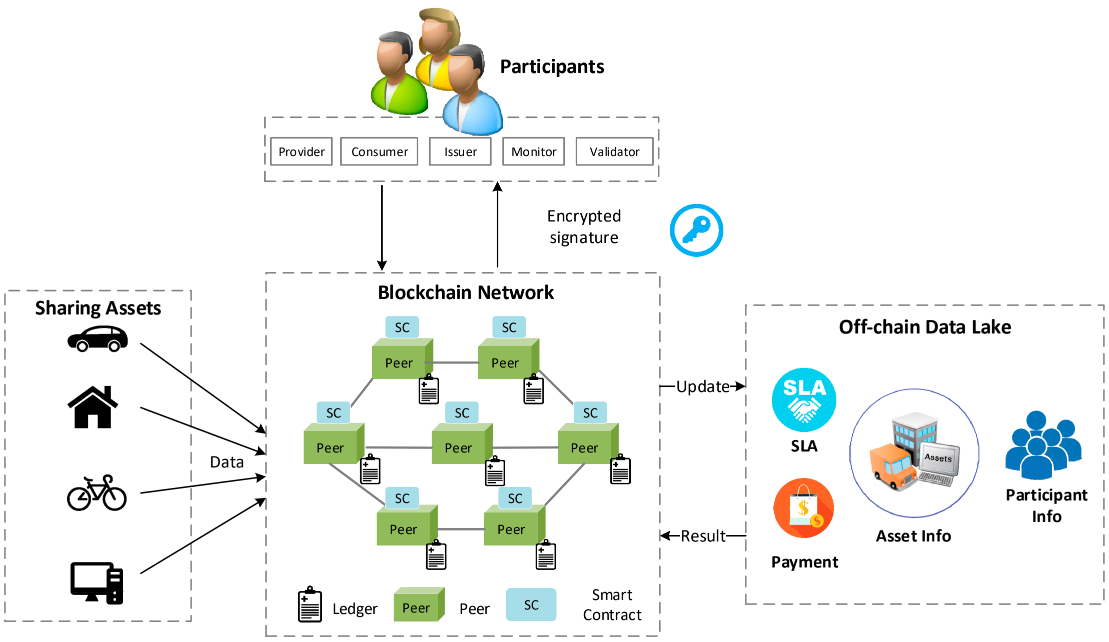


Figure 1. Conceptual model of the service level agreement (SLA)-based sharing economy service with the smart contract.

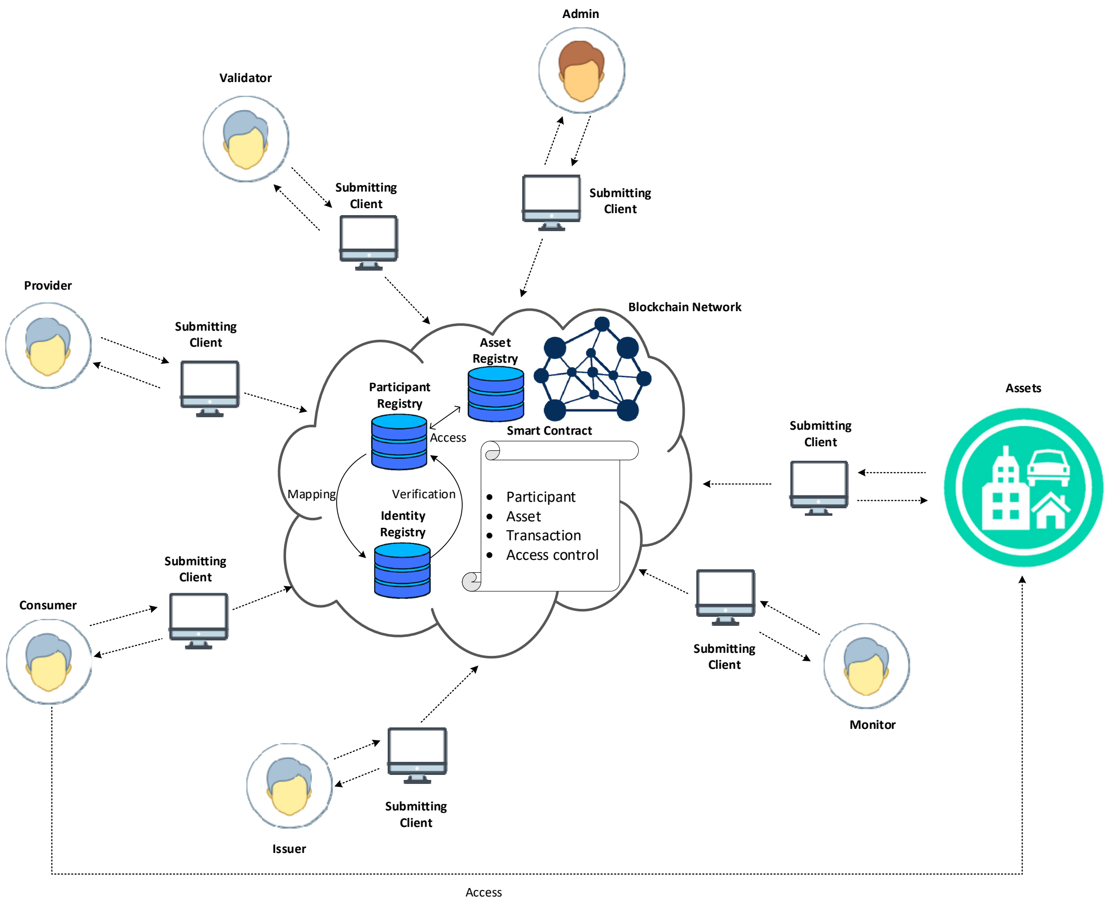


Figure 2. Proposed sharing economy service interaction diagram.

3.2. Participants of the Proposed Sharing Economy Service

The proposed sharing business model is defined as a building agreement based on the SLA specification for concluding the contract between the providers and users. We propose the mechanism of specifying the components of the SLA based on the smart contract. The most characteristic feature of SLA is that the services should be provided to the consumer as agreed upon in the contract. In the proposed sharing economy service model, the financial transactions are services that are automated and intelligently managed by the SLA contract. All the participants in trusted operations in which the SLA contract studies the agreement and the obligations are executed automatically without human involvement. Such a system can report faults, handle payment and refund, monitor performance metrics, perform data analytics, and make decisions. As shown in Figure 3, the proposed system includes many participants in terms of their roles. In this work, five different participants are defined, including the validator, provider, consumer, issuer, and monitor. The provider is a vendor that offers the required solutions or services based on the SLA contract templates. The validator validates the SLA registered to the system and responds to the system with the validation results. The consumer can discover the SLA and associated validation results. negotiation is performed to formulate the right contract content that suits both providers and consumers. Once the service is being established and executed, it needs to be monitored. The monitor continuously monitors the service performances according to the respective guarantees posed in the smart contract. Once these guarantees are violated, the monitor sends the notice to the system. The penalties are assessed to conduct the respective compensation actions associated with them, as dictated by the smart contract. Moreover, charging is automatically executed according to the cost of the service concerned. Once the contract validity period expires, the SLA contract is automatically terminated. However, its specification can remain in the system for accountability and legal reasons for some time. After that period, the contract might be discarded, as in the case of regular contracts. For the reuse, the specification would require to be preserved in the system. The issuer is a legal entity that is authorized to issue its securities or tokens. The created token so-called IoTcoin is a particular coin, which allows providers to tokenize assets. Representing assets as tokens establish the individual state and permission of an item, and reduce the risk and difficulty of transferring assets across multiple parties.

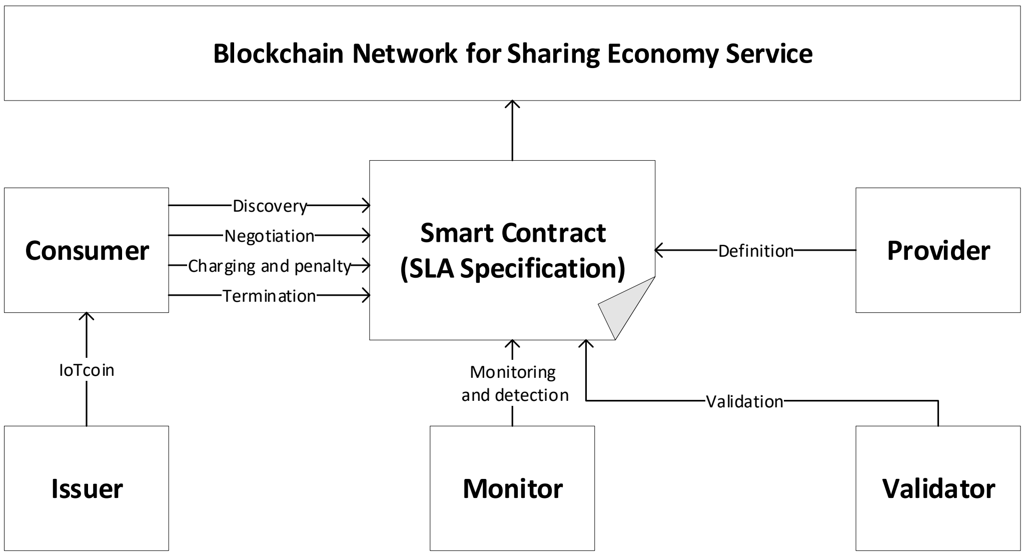


Figure 3. The relation between participants and the smart contract with the SLA specification in the proposed model.

3.3. Assets of the Proposed Sharing Economy Service

In the proposed sharing business model, there are two classes of the assets: Intangible asset and tangible assets. Tangible assets are physical properties (e.g., car, house, desktop) that can be controlled by electronics locks or access control systems. Intangible assets include non-physical properties (e.g., sensor data, access keys) that can be controlled by digital devices. However, these two assets share many standard features; for example, they can be transmitted directly to the network or controlled by smart devices. Moreover, because there are no traditional stages such as storage or logistics, customers can get their assets immediately while the deal is completed. Although participants are related to the assets, the shift of the usage permission can be realized by the token transmission or digital control. In a real environment, cars with the anti-theft system can only be unlocked with the right key that is equipped with the corresponding private key provided by the anti-theft system. So far, the potential of the asset is far from being fully developed. In the above example, the private key is usually kept in a physical container (e.g., SIM card), which is hard to transfer. The blockchain can address this issue as the permission can be accomplished in the network. In the proposed sharing business model, we can use mobile devices equipped with the NFC module as the carrier and use the APP to transfer the usage permission on the blockchain.

3.4. Blockchain Network of the Proposed Sharing Economy Service

As shown in Figure 4, the blockchain network proposed in this paper is built on a permissioned network that only allows access from authorized users. This solution avoids the risk of data exposure since the transaction history that records how a resource is manipulated are invisible to unauthorized users. In a permissioned blockchain network, only authorized users can operate on the blockchain, and a valid block must contain a signature from a subset of users. As a result, no invalid node could modify or insert a transaction in the blockchain. Moreover, the approved smart contract cannot be altered unless it gets signed by the agreement from all users featured on the operation. The blockchain network provides various blockchain features including identity authentication, verification, and P2P communication. Peers are the fundamental entity of the blockchain network since they maintain the ledger and provide the running environment for the smart contracts. The orderer peer provides transaction ordering that packages the transaction in sequence and creates blocks of transactions. The distributed ledger is consensually shared and synchronized across the blockchain network, in which each peer of the network can have their identical copy of it. The practical byzantine fault tolerance (PBFT) algorithm provides a mechanism for peers to correctly reach a consensus despite malicious peers in sending out incorrect information. The smart contract is installed and instantiated onto all peers and runs as a decentralized application to manage access and modifications in the ledger. The event hub generates events to the client whenever a new block is added to the ledger, or the transaction triggers the predefined condition in the smart contract. The API interface exposes the services provided by the blockchain network as open API schemes by which the client application can interact with the smart contract.

This work utilizes the asymmetric cryptography, including a key pair to encrypt and decrypt data, as shown in Figure 5. The public key is often provided by a trusted, designated authority and used to encrypt data. The other key in the pair is the private key that is kept secret and is mathematically linked to the public key. In an asymmetric cryptography, either of the keys can be used to encrypt a message, but whatever is encrypted can only be decrypted by its relevant opposite key.

A trusted authority called the public key institution (PKI) [42] is required to make users and systems aware of that a public key is authentic and it has not been tampered with by a malicious third party. As shown in Figure 6, the PKI architecture in our system consists of certificate authorities (CA) and the certificate database. CA is a trusted party, which provides the root of trust for all PKI certificates and provides services to authenticate, issue, and revoke the identity of individuals. Each CA maintains its own root CA, for use only by the CA. The CA that is certified by a root CA is authorized to issue certificates for specific uses allowed by the root. The certificate database stores information about issued certificates and the validity period and status of each PKI as well. Among various standards of PKI, this work utilizes the Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework (RFC 5280) [43]. The X.509 certificates are widely used in many Internet protocols, for example, the TLS or SSL, and are also used in offline applications like electronic signatures. Although a blockchain network can perform transactions without the involvement of a third-party, it still depends on the PKI architecture to ensure secure communication between various network participants, and to ensure that messages posted on the blockchain are correctly authenticated.

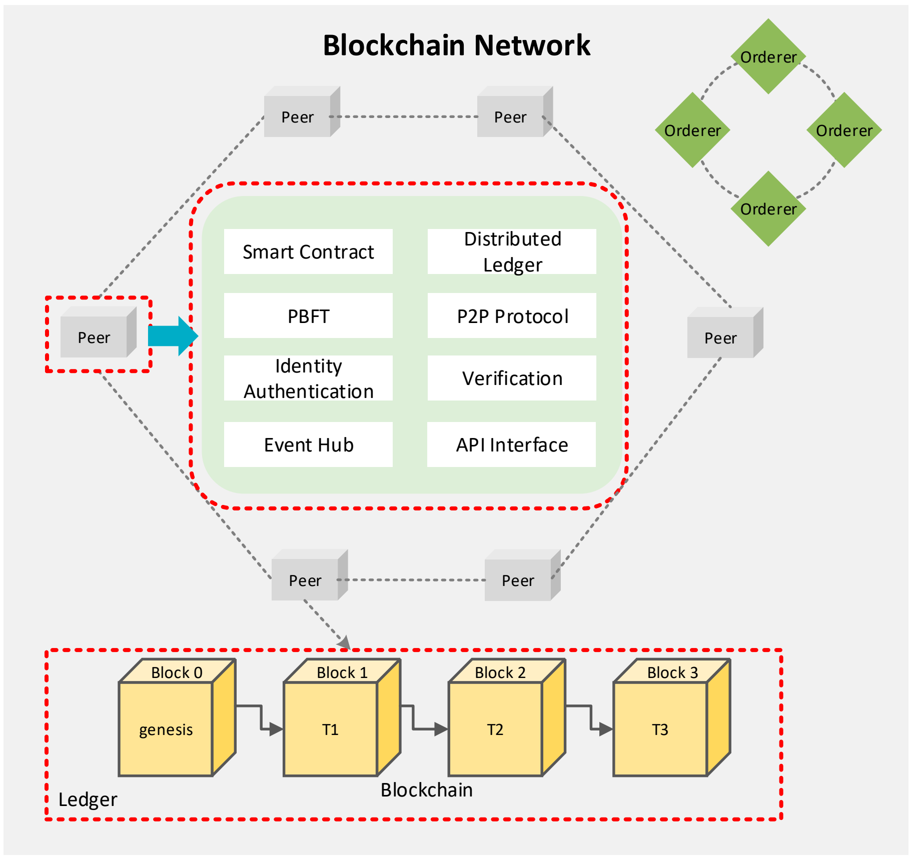


Figure 4. Blockchain network for the proposed sharing economy service.

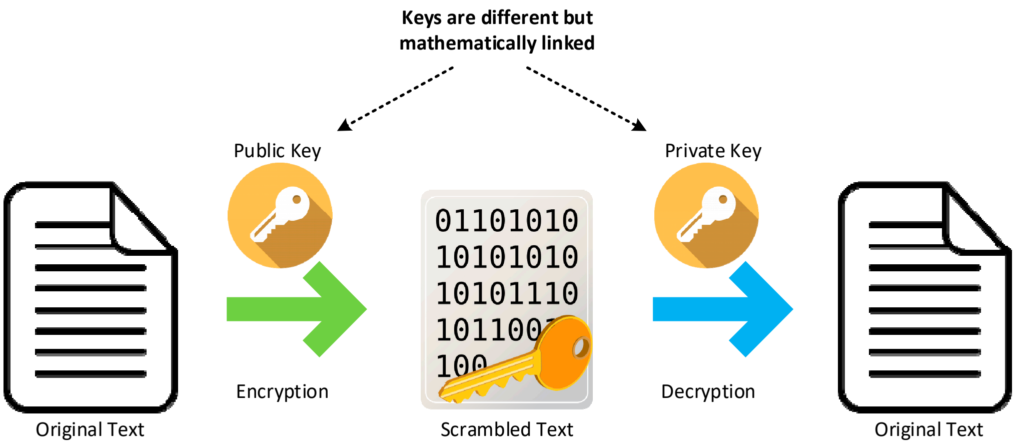


Figure 5. Asymmetric key encryption for the data encryption and decryption.

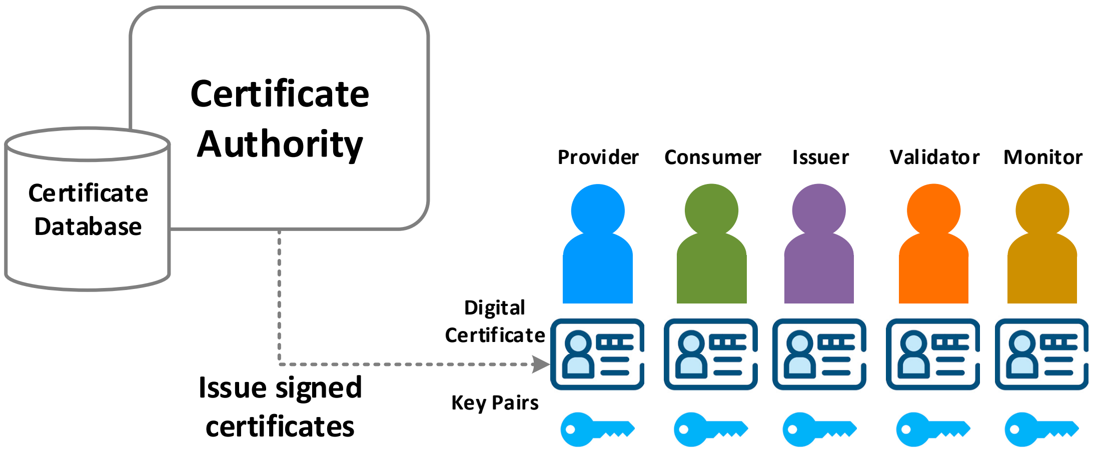


Figure 6. Digital identity issuing using certificate authorities (CA).

Figure 7 presents the transactional workflow during a transaction operation. The client application user has registered and enrolled with the CA and received back necessary, which is used to authenticate to the network. Transaction proposals are constructed by client applications, which leverages an application SDK to send a request to the target peers. The proposal is a request to invoke the smart contract functions with the intent of reading or updating the ledger. The endorser peers take the transaction proposal inputs as arguments to the invoked smart contract function. The smart contract is then executed against the defined business logic to produce the transaction results in their simulated environment. However, the updated transaction results will not be made to the ledger at this point. The set of response values include a response value, read set, and write set. The read set captures the latest read from the current state, and the write set holds the data that will be written to the next state when simulating the transaction. The read set and write set, along with the endorser peer's signature is passed back as a proposal response to the client application. The client application verifies the endorser peer signatures and determines if the predefined endorsement policy has been fulfilled before broadcasting the transaction proposal and response to the orderer. The orderer receives the transactions within the network, orders them into a block and delivers these blocks to all the committer peers. Each committer peer validates the transaction to validate whether the endorsement policy is fulfilled and to ensure whether the read set is matched with the current state. Once the transaction is validated, each peer appends the block to the ledger and commits the write set to the current state. Lastly, these committing nodes generate asynchronous events to notify the client that the transaction has been appended to the blockchain as well as a notification of whether the transaction is validated or not.

3.5. Off-Chain Data Lake of the Proposed Sharing Economy Service

We propose the use of the off-chain data lake to take away the current values of the ledger states from the blockchain. Clients submit transactions that capture changes to the off-chain data lake, and these transactions end up being committed to the blockchain. The most remarkable difference between the blockchain data structure and the off-chain data lake is that the data immutability. The data cannot be modified in the blockchain even by the network admin once it is written into the ledger. However, the data stored in the off-chain data lake updates incessantly whenever the state value changes, such as when the ownership of a car is transferred from one to another. The off-chain data lake serves as a database, which provides a rich set of operators for the efficient storage and retrieval of states. The blockchain network can be configured to use different databases to address the needs of different types of values and the access patterns required by applications. The proposed approach greatly enhances the transaction processing performance since we do not need to traverse the overall transaction log in the blockchain.

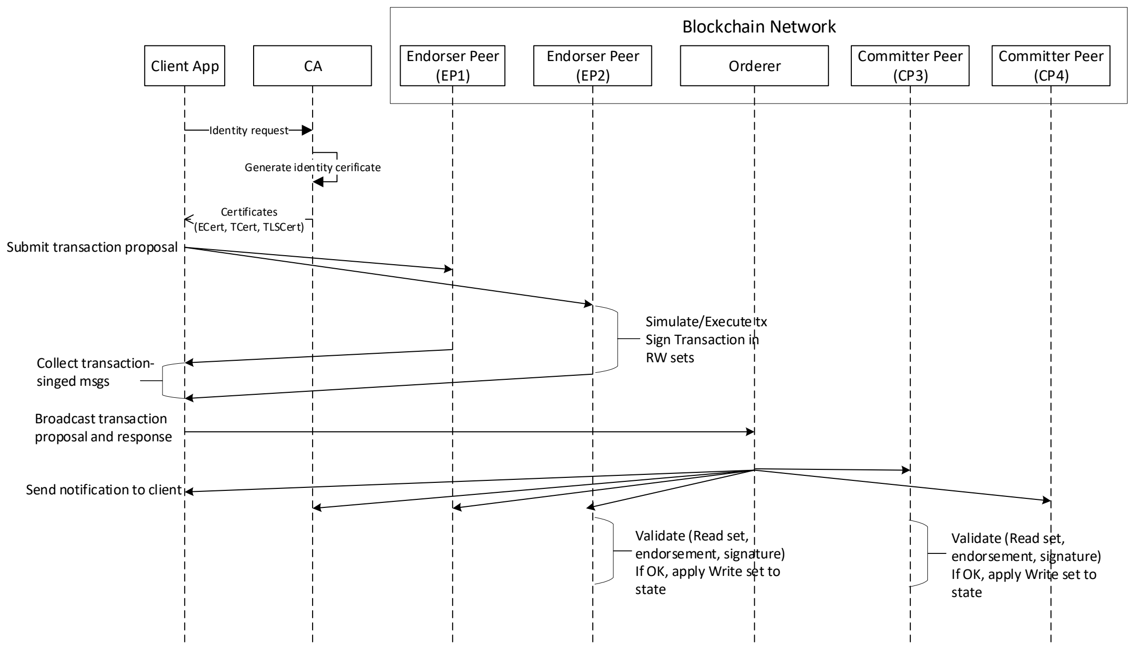

Blockchain Network
Client App
CA
Endorser Peer (EP1)
Endorser Peer (EP2)
Orderer
Committer Peer (CP3)
Committer Peer (CP4)
Identity request
Generate identity cerificate
Certificates (ECert, TCert, TLSCert)
Submit transaction proposal
Simulate/Execute tx
Sign Transaction in RW sets
Collect transaction-singed msgs
Broadcast transaction proposal and response
Send notification to client
Validate (Read set, endorsement, signature) If OK, apply Write set to state
Validate (Read set, endorsement, signature) If OK, apply Write set to state

Figure 7. Transaction operational processes in the blockchain network.

4. Case Study Implementation Using Hyperledger Fabric

This section presents the development technologies and tools to implement the notebook sharing case study. The Hyperledger Fabric blockchain network is deployed in the Ubuntu Linux v18.04.2 LTS with an Intel Core i5-8500 @ 3.00 GHz processor and 12 GB memory. The Docker engine acts as the running environment and provides the tools to configure docker images and containers in the virtual machine. The blockchain client application is implemented using a JavaScript SDK, which provides a set of APIs to manage and interact with the deployed business network. The VSCode is used as the integrated development environment (IDE) to work on the smart contract code. We use the Hyperledger Composer [44] to generate the Representational State Transfer (REST) Server that exposes the blockchain logic to the web or mobile applications. The REST Server converts the smart contract for a business network into an open API definition, which at runtime supports create, read, update and delete (CRUD) to perform operations on the blockchain. To secure the access of the REST network resources, Passport.js is utilized to authenticate requests through an extensible set of strategies, including using a username and password, and third-party services. The CouchDB is used as the off-chain data lake to hold the current values of the ledger states. It provides rich query support, including the get, put, and delete method, which enables the smart contract to access the off-chain data lake via simple APIs. The structure of a participant in the smart contract is modeled, as shown in Table 1.

As mentioned earlier, assets can range from the tangible (real estate and hardware) to the intangible (contracts and intellectual property). The Hyperledger blockchain network provides the ability to modify assets using transactions. Table 2 represents the structure of an asset, which contains various information about the asset.

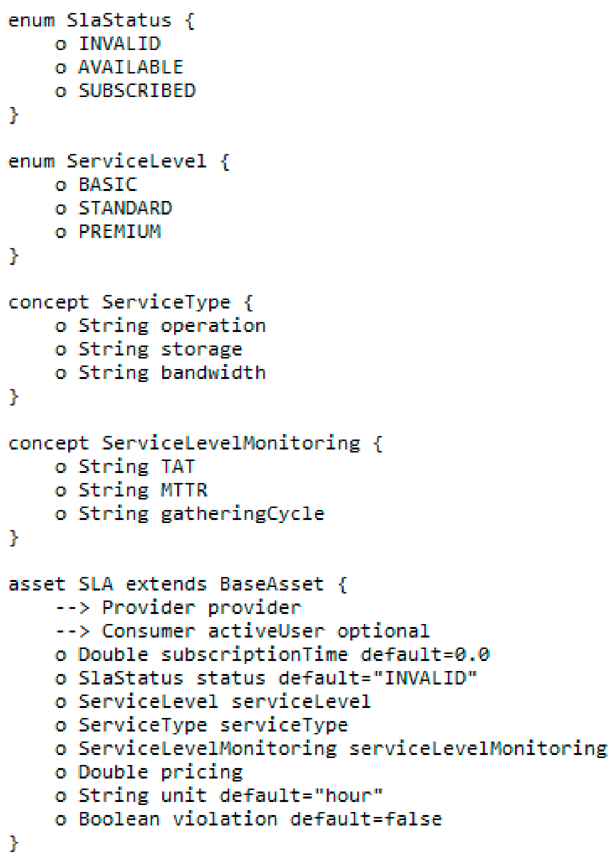
Figure 8 details the SLA specifications, which are specified for the notebook sharing case study. The notebook providers and consumers are used as the basis of the agreement and represent the primary stakeholders associated with SLA. The SLA contains three statuses over the lifecycle of a sharing service that represents the status over the SLA lifecycle. We define three different service levels, which vary from the operation level, data storage and the network bandwidth. The turn-around time (TAT), mean time to cover (MTTR), and gathering cycle are used as performance metrics in the SLA. The subscription time represents the valid period of the SLA between the provider and consumer. The violation field is used to represent whether a violation occurs under terms of the contract, and it can only be modified by the validator.

Table 1. Participant definition in the smart contract.

Name	Property	Type
Issuer	pid	String
	contact info (name, phone, address)	Concept
	token name	String
	type	String
	quantity	String
Provider	pid	String
	contact info (name, phone, address)	Concept
	org name	String
	balance	Double
Consumer	pid	String
	contact info (name, phone, address)	Concept
	active notebook Ids	String [ ]
	issuer	Object
	balance	Double
Validator	pid	String
	contact info (name, phone, address)	Concept
	org name	String
Monitor	pid	String
	contact info (name, phone, address)	Concept
	org name	String

Table 2. Asset definition in the smart contract.

Name	Property	Type
Notebook	Aid	String
	notebook status (instore, available, in use, callback)	Enum
	Provider	Object
	user (active)	Object
	Remark	String
Login password	Aid	String
	user (active)	Object
	Notebook	Object
	Password	String
Service issue	Aid	String
	User	Object
	Provider	Object
	issue status (submitted, in progress, solved, unsolved)	Enum
	issue note	String
Service level agreement (SLA)	Aid	String
	Provider	Object
	service level (basic, standard, premium)	Enum
	service type (operation, storage, connection bandwidth)	Concept
	service level monitoring (TAT, MTTR, gathering cycle)	Concept
	Pricing	Double
	unit (hour)	Double
	SLA status (invalid, available, subscribed)	Enum
	violation (default: false)	Boolean



```
enum SlaStatus {
    o INVALID
    o AVAILABLE
    o SUBSCRIBED
}

enum ServiceLevel {
    o BASIC
    o STANDARD
    o PREMIUM
}

concept ServiceType {
    o String operation
    o String storage
    o String bandwidth
}

concept ServiceLevelMonitoring {
    o String TAT
    o String MTTR
    o String gatheringCycle
}

asset SLA extends BaseAsset {
    --> Provider provider
    --> Consumer activeUser optional
    o Double subscriptionTime default=0.0
    o SlaStatus status default="INVALID"
    o ServiceLevel serviceLevel
    o ServiceType serviceType
    o ServiceLevelMonitoring serviceLevelMonitoring
    o Double pricing
    o String unit default="hour"
    o Boolean violation default=false
}
```

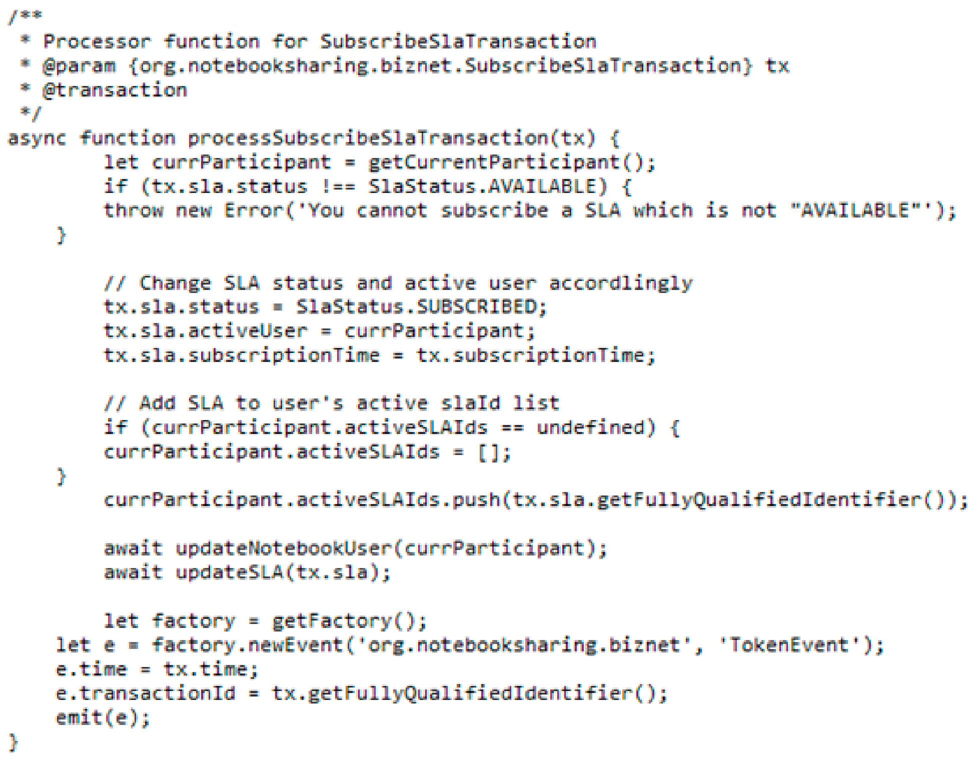
Figure 8. Detailed SLA specification in the smart contract.

As shown in Table 3, we define the list of transactions for the notebook sharing case study. Participants can interact with assets through the transactions. The transaction is restricted to specified participants, and they modify the resources according to the predefined operation. It is worth noting that the operation can equally apply to participants not only to assets according to business requirements. For example, the "Issue token" transaction is used to increase the token balance of a specified consumer participant.

Table 3. Transaction definition in the smart contract.

Transaction	Participant	Operation	Resource (Participant, Asset)
Issue token	issuer	UPDATE	consumer
Redeem token	issuer	UPDATE	consumer
Register notebook	provider	CREATE	notebook
Release notebook	provider	UPDATE	notebook
Callback notebook	provider	UPDATE	notebook
Generate SLA	provider	CREATE	SLA
Validate SLA	validator	UPDATE	SLA
Subscribe SLA	consumer	UPDATE	consumer, SLA
Terminate SLA	consumer	UPDATE	consumer, SLA
Rent notebook	consumer	UPDATE	notebook, consumer, provider
Return notebook	consumer	UPDATE	notebook, consumer
Request login password	consumer	CREATE	login password
Cancel login password	consumer	DELETE	login password
Submit issue	consumer	CREATE	service issue
Confirm issue	provider	UPDATE	service issue
Solve issue	provider	UPDATE	service issue
Unsolve issue	provider	UPDATE	service issue
Detect violation	monitor	UPDATE	SLA
Fee refund	consumer	UPDATE	consumer, provider, SLA, notebook

The transaction process function is the logical operation of a transaction defined in the smart contract as shown in Figure 9. The structure of this function includes decorators and metadata followed by a JavaScript function, both parts are required for a transaction to work. As shown in Algorithm 1, the pseudo code for the transaction processor function relating to the "Subscribe SLA" transaction is to modify the status of both the SLA asset and the current consumer. This transaction will execute only when the status of the SLA is available. Then, it updates the status of the SLA asset, adds the associations between the SLA asset and the current participant, and emits an event.



```
/**
 * Processor function for SubscribeSlaTransaction
 * @param {org.notebooksharing.biznet.SubscribeSlaTransaction} tx
 * @transaction
 */
async function processSubscribeSlaTransaction(tx) {
        let currParticipant = getCurrentParticipant();
        if (tx.sla.status !== SlaStatus.AVAILABLE) {
        throw new Error('You cannot subscribe a SLA which is not "AVAILABLE"');
    }

        // Change SLA status and active user accordlingly
        tx.sla.status = SlaStatus.SUBSCRIBED;
        tx.sla.activeUser = currParticipant;
        tx.sla.subscriptionTime = tx.subscriptionTime;

        // Add SLA to user's active slaId list
        if (currParticipant.activeSLAIds == undefined) {
        currParticipant.activeSLAIds = [];
    }
        currParticipant.activeSLAIds.push(tx.sla.getFullyQualifiedIdentifier());

        await updateNotebookUser(currParticipant);
        await updateSLA(tx.sla);

        let factory = getFactory();
    let e = factory.newEvent('org.notebooksharing.biznet', 'TokenEvent');
    e.time = tx.time;
    e.transactionId = tx.getFullyQualifiedIdentifier();
    emit(e);
}
```

Figure 9. Snapshot of the transaction process function for subscribing the SLA.

Algorithm 1. Pseudo Code for Subscribe SLA Transaction (*Consumer ID, SLA ID, Subscription Time, Timestamp*)

Input (*Consumer ID, SLA ID, Subscription Time, Timestamp*)

```
Begin
  Set Consumer as the Current Participant
  If SLA State = AVAILABLE then
    SLA State changes to SUBSCRIBED
    SLA ActiveUser changes to Consumer ID
    SLA Subscription Time changes to Subscription Time
    If Active SLA ID List of Consumer = undefined then
      Initialize Active SLA ID List of Consumer
    Else
      Push SLA ID to Active SLA ID List of Consumer
      Emit event to Current Participant
  Else
  Reject the transaction and show an error.
End
```

Figure 10 presents the interaction between the blockchain and the off-chain data lake. The sample off-chain data lake includes one SLA asset expressed as key-value pairs. In contrast to the off-chain data lake, the blockchain is a transaction log that records all the changes that reflect the off-chain data lake. Transactions are sorted into blocks in time order, and these blocks are cryptographically linked to each other to form a sequence of the chain, which enables the user to know the historical changes that took place in the off-chain data lake.

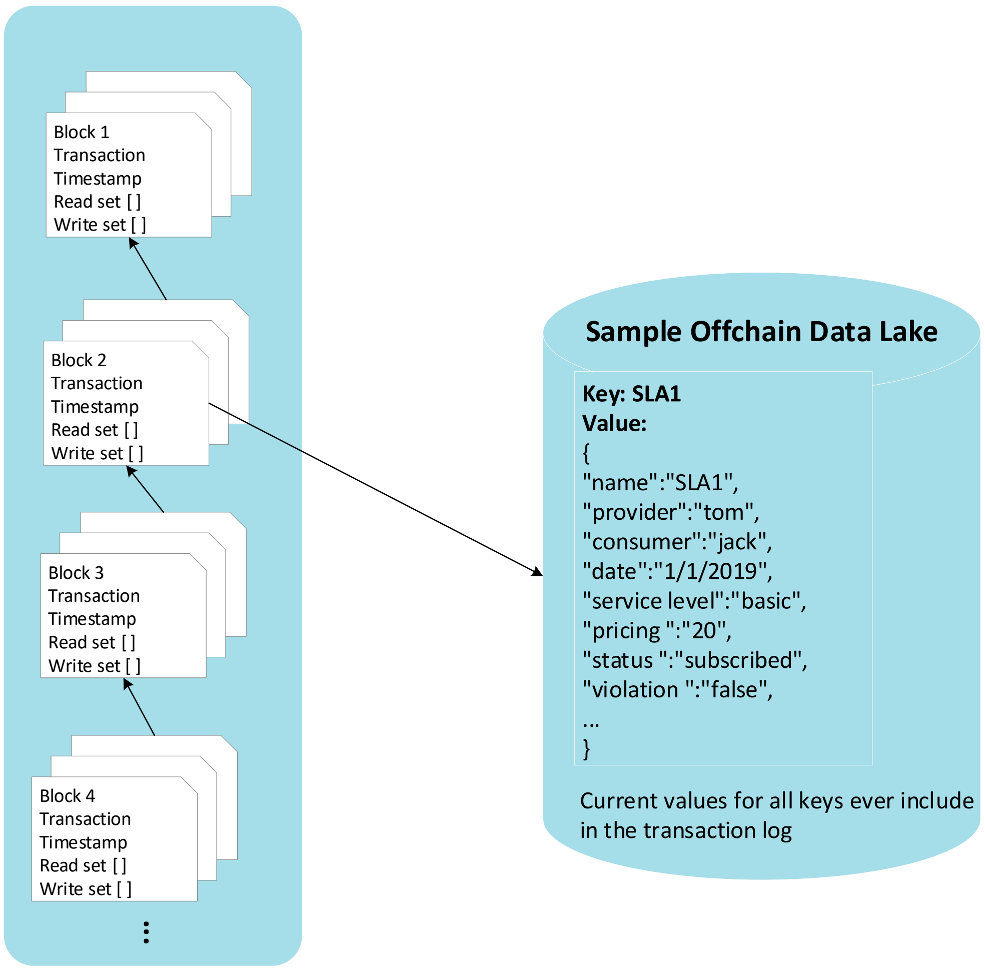


Figure 10. Interaction with the blockchain file system and off-chain data lake in the case study.

In the real product environment, the real-time data from assets are needed to determine whether the violation happened over the life of the SLA contract. In this paper, the monitoring of the assets is not considered to simplify the case study. Figure 11 illustrates the simplified workflow of the notebook sharing case study. In the permissioned network, every participant of the system is required to obtain a certificate that contains the identity information and the connection file before connecting to the system. This certificate is issued through the register and enrollment processes that can be performed only by the system admin. The provider can register the basic information of the notebook, including the model name, manufacturer, and some other specifications. However, in this stage, the notebook is not visible to consumers unless the provider releases it. Then, the provider can define the SLA, and the format of the SLA needs to be confirmed by the validator before the consumer can view the document. The issuer issues tokens to consumers as the further steps require tokens. The consumer subscribes the SLA and rents the notebook. In this step, the consumer transfers a certain number of tokens to the provider according to the cost of service level. The consumer can get the login password that is randomly generated by the network and use the obtained password to access the notebook. During the contract period, the consumer can submit an issue to the network if a fault occurs; for instance, the wireless network is disconnected. The network delivers this issue to the provider who can perform the task to resolve the issue. The provider can modify the status of the submitted issue in terms of the actual condition. The monitor continually monitors the status and makes decisions accordingly. In the case of that, the issue is not resolved according to the task time defined in the SLA, this can be considered as the violation, and the consumer can get the compensation. If no validation is detected during the contract and the lease is expired, the consumer can terminate the SLA, in turn, cancels the login password and returns the notebook.

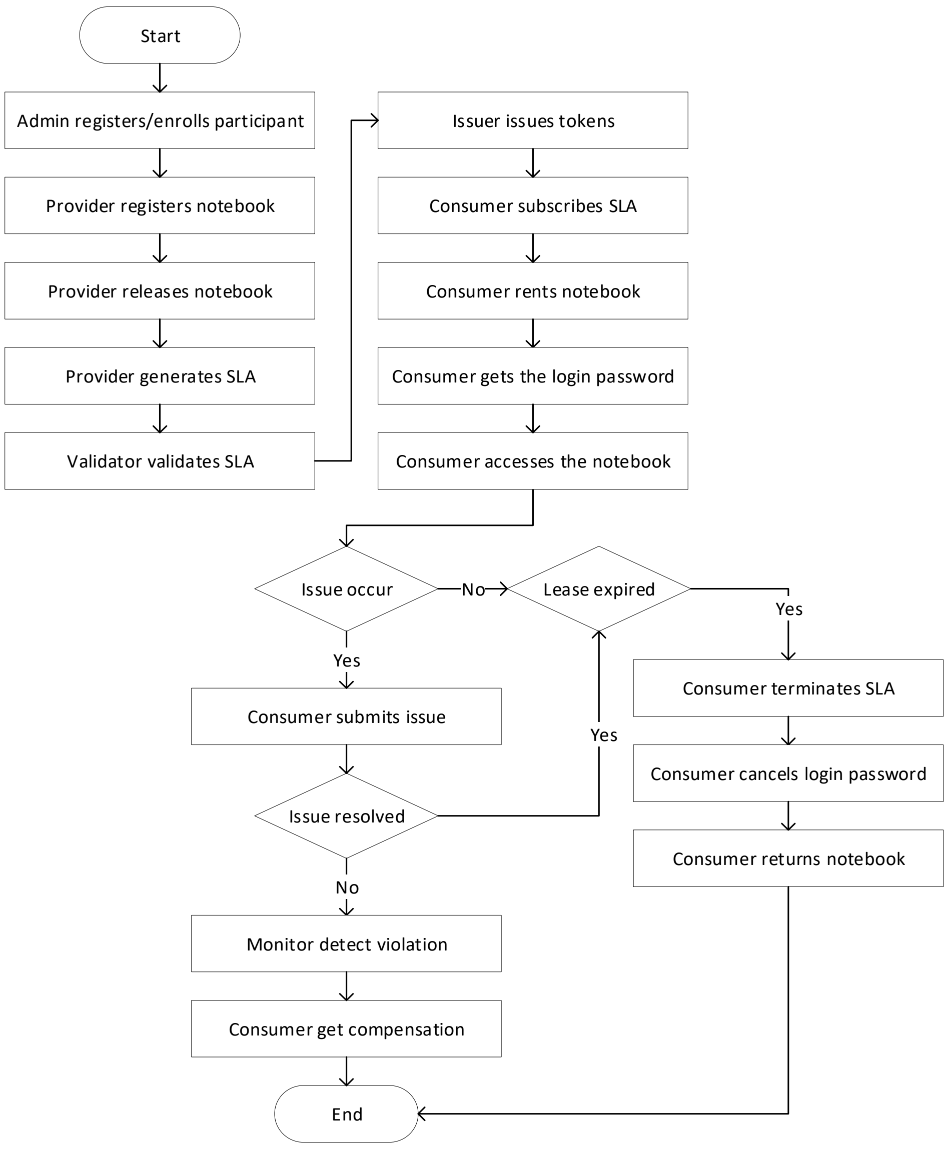


Figure 11. The simplified workflow of the notebook sharing case study.

In this case study, we configure the REST Server to authenticate clients using the Github authentication provider. As shown in Figure 12, the client must authenticate to the REST Server before it is permitted to access to business network resources, which are protected by the OAuth2.0 scheme. The service owner (Github account) can grant consent to the client application. The Github authorization server requests consent of the service owner and issues access tokens to clients. The issued access token is stored in the local storage of the web browser. When the user initiates a subsequent request, the client validates the access token retrieved from the local storage instead of reauthenticating the user. Moreover, the REST Server itself is configured to persist the business network cards that are required to connect to the blockchain network. These business network cards are identities issued by using the Fabric-CA server to register enrollment certificates. Each identity has a standard structure comprising a descriptive label, an X.509 certificate containing a public key, a private key, and some Fabric-specific metadata.

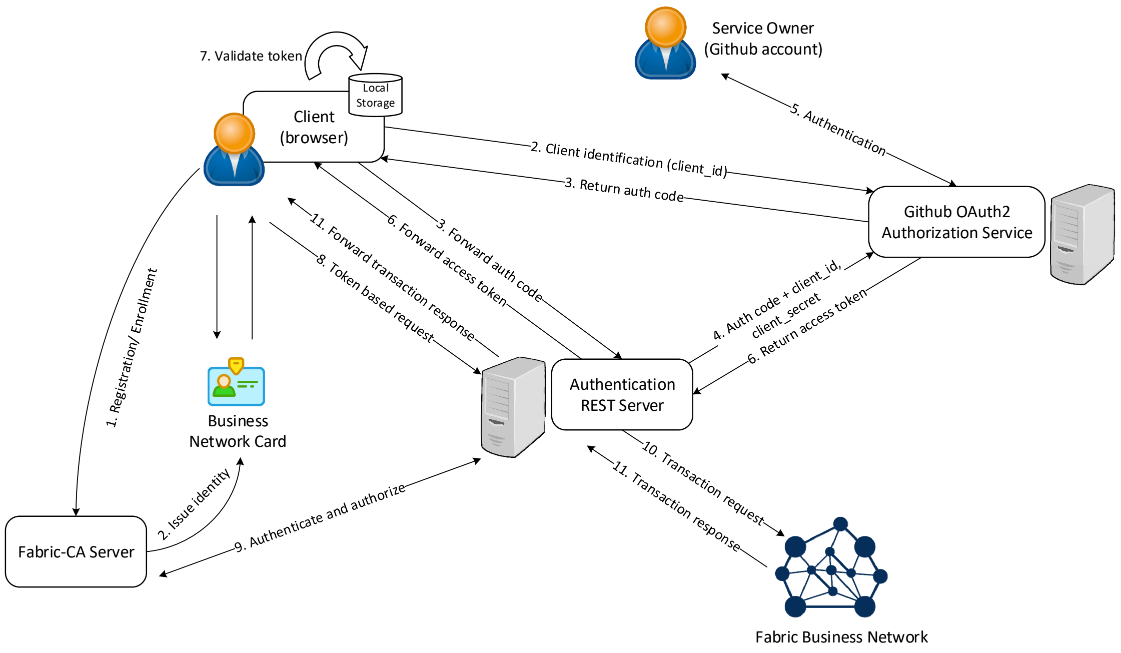
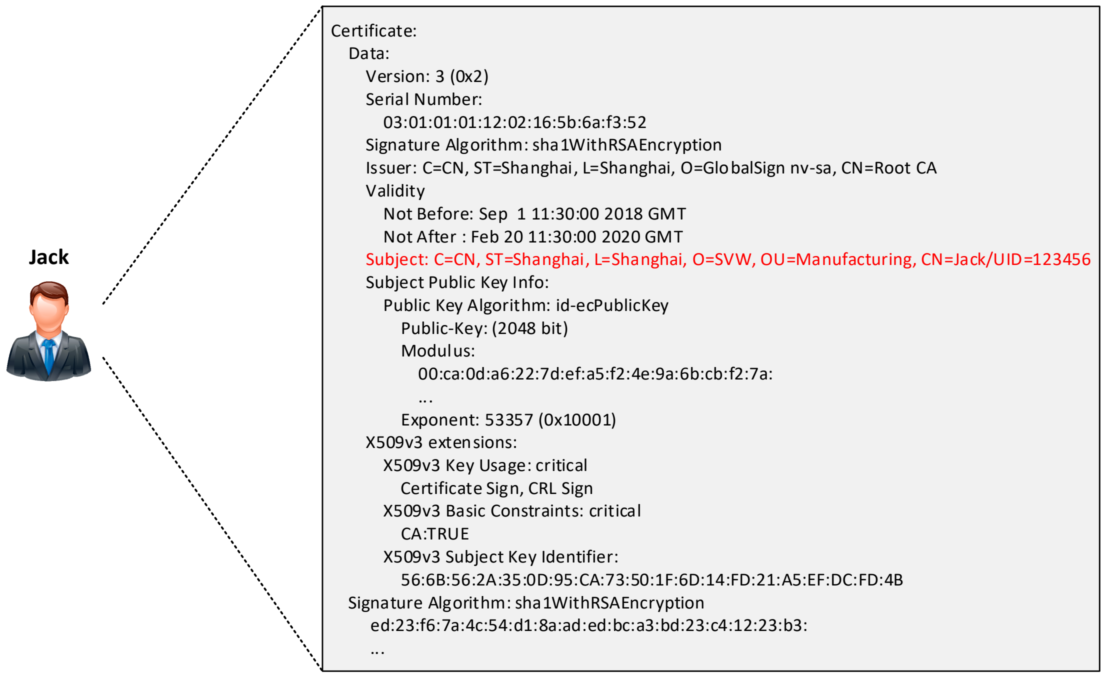


Figure 12. Github OAuth2.0 REST authentication overview.

As shown in Figure 13, the sample certificate contains a digital certificate describing a participant called Jack. This certificate includes identifying information such as the PKI that issued the certificate, the creation date of the certificate, and the certificate validity period. The highlighted subject text provides key facts about Jack, and it also holds many pieces of information. It is noteworthy that Jack's public key is distributed within his certificate. However, his private signing key is kept private.



```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      03:01:01:01:12:02:16:5b:6a:f3:52
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=CN, ST=Shanghai, L=Shanghai, O=GlobalSign nv-sa, CN=Root CA
    Validity
      Not Before: Sep  1 11:30:00 2018 GMT
      Not After : Feb 20 11:30:00 2020 GMT
    Subject: C=CN, ST=Shanghai, L=Shanghai, O=SVW, OU=Manufacturing, CN=Jack/UID=123456
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (2048 bit)
        Modulus:
          00:ca:0d:a6:22:7d:ef:a5:f2:4e:9a:6b:cb:f2:7a:
          ...
        Exponent: 53357 (0x10001)
    X509v3 extensions:
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Subject Key Identifier:
        56:6B:56:2A:35:0D:95:CA:73:50:1F:6D:14:FD:21:A5:EF:DC:FD:4B
  Signature Algorithm: sha1WithRSAEncryption
    ed:23:f6:7a:4c:54:d1:8a:ad:ed:bc:a3:bd:23:c4:12:23:b3:
    ...
```

Figure 13. Sample participant certificate issued by Fabric-CA using X.509.

5. Implementation Results of the Case Study

This section presents the implementation results of the case study in the Hyperledger Composer Playground [45] with various snapshots. It is a web user interface to test the business logic of the smart contract that executes on the Hyperledger Fabric runtime. As shown in Figure 14, the playground wallet holds a set of identities, each issued by the Fabric CA.

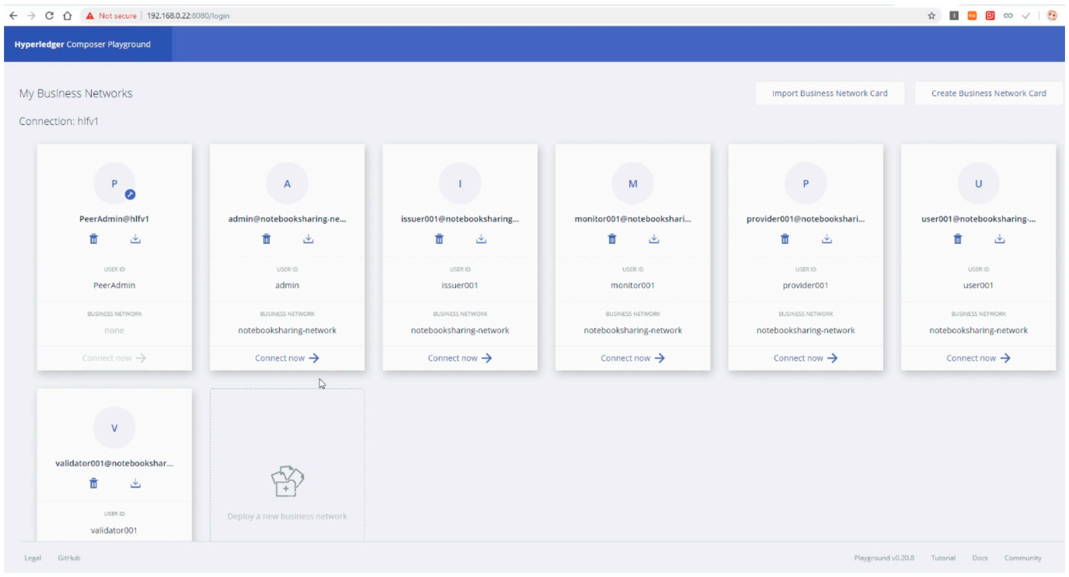
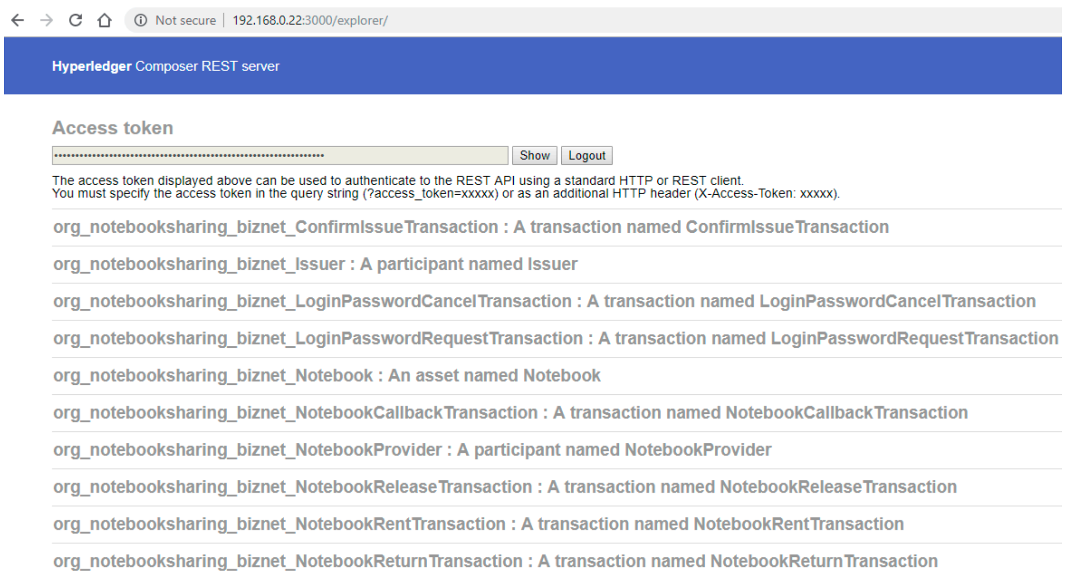


Figure 14. Composer web playground main interface.

Figure 15 presents the Composer REST Server explorer, which is used to inspect and test the generated REST API. The smart contract model for a business network is converted into an Open API definition, and at runtime the create, read, update and delete operation are implemented for allowing transactions to be submitted for processing or retrieved. When the client authenticates to the REST Server, the explorer redirects the request to the Github authentication provider. Afterward, the authentication provider redirects back to the REST Server and shows the access token at the top of the page. The access token can be passed into the REST request to authenticate the client.


Not secure | 192.168.0.22:3000/explorer/
Hyperledger Composer REST server
Access token
Show
Logout

The access token displayed above can be used to authenticate to the REST API using a standard HTTP or REST client.
You must specify the access token in the query string (?access_token=xxxxx) or as an additional HTTP header (X-Access-Token: xxxxx).

org_notebooksharing_biznet_ConfirmIssueTransaction : A transaction named ConfirmIssueTransaction

org_notebooksharing_biznet_Issuer : A participant named Issuer

org_notebooksharing_biznet_LoginPasswordCancelTransaction : A transaction named LoginPasswordCancelTransaction

org_notebooksharing_biznet_LoginPasswordRequestTransaction : A transaction named LoginPasswordRequestTransaction

org_notebooksharing_biznet_Notebook : An asset named Notebook

org_notebooksharing_biznet_NotebookCallbackTransaction : A transaction named NotebookCallbackTransaction

org_notebooksharing_biznet_NotebookProvider : A participant named NotebookProvider

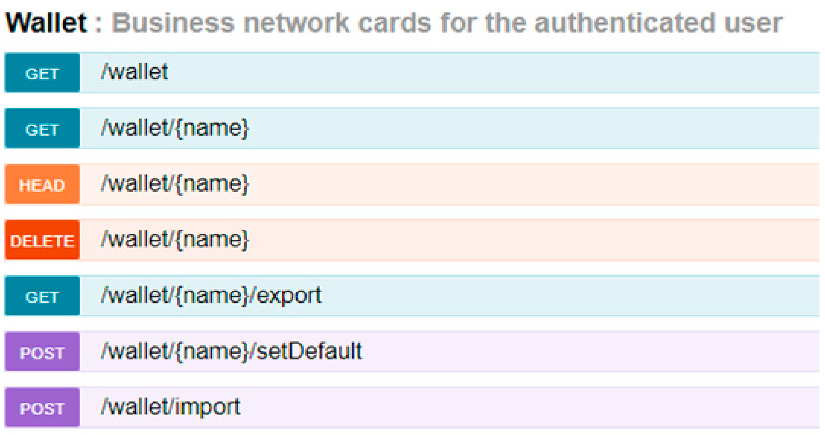
org_notebooksharing_biznet_NotebookReleaseTransaction : A transaction named NotebookReleaseTransaction

org_notebooksharing_biznet_NotebookRentTransaction : A transaction named NotebookRentTransaction

org_notebooksharing_biznet_NotebookReturnTransaction : A transaction named NotebookReturnTransaction

Figure 15. Composer REST Server explorer main interface.

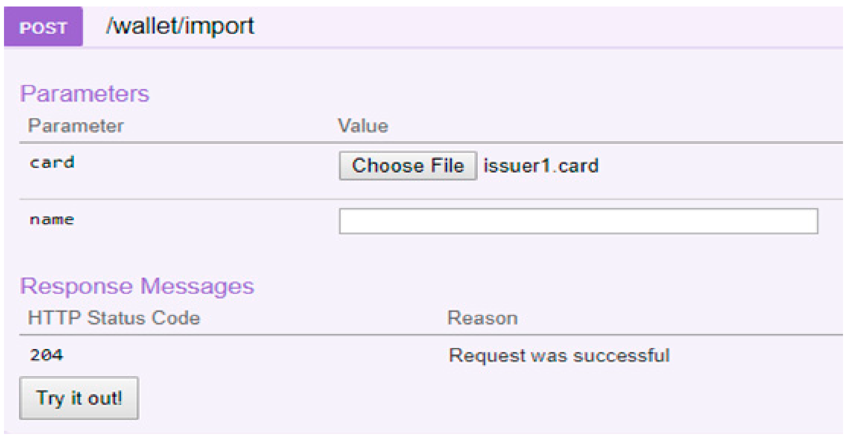
Once a client has authenticated to the REST API, that client can add identities to a wallet. The wallet is private to that client and is not accessible to other clients. When a client requests the REST Server, an identity in the client's wallet is used to sign all transactions made by that client digitally. To add an identity card to the wallet, we should navigate to the wallet APIs by expanding the wallet category, as shown in Figure 16a. The identity card can be imported into the wallet by calling the POST /wallet/import operation, as shown in Figure 16b.



Wallet : Business network cards for the authenticated user

GET	/wallet
GET	/wallet/{name}
HEAD	/wallet/{name}
DELETE	/wallet/{name}
GET	/wallet/{name}/export
POST	/wallet/{name}/setDefault
POST	/wallet/import

a



POST /wallet/import

Parameters

Parameter	Value
card	Choose File issuer1.card
name	

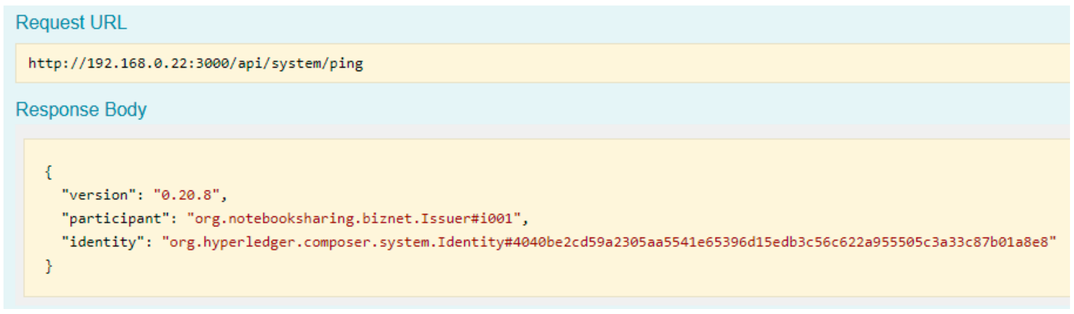
Response Messages

HTTP Status Code	Reason
204	Request was successful

b

Figure 16. Snapshot of the wallet API. (**a**) Wallet API specifications; (**b**) add a business network card to the wallet.

We can check that the participant does connect to the business network by calling the following API. As shown in Figure 17, the current participant is connected to the business network as it returns the information of the participant and related identity in the response body.



Request URL

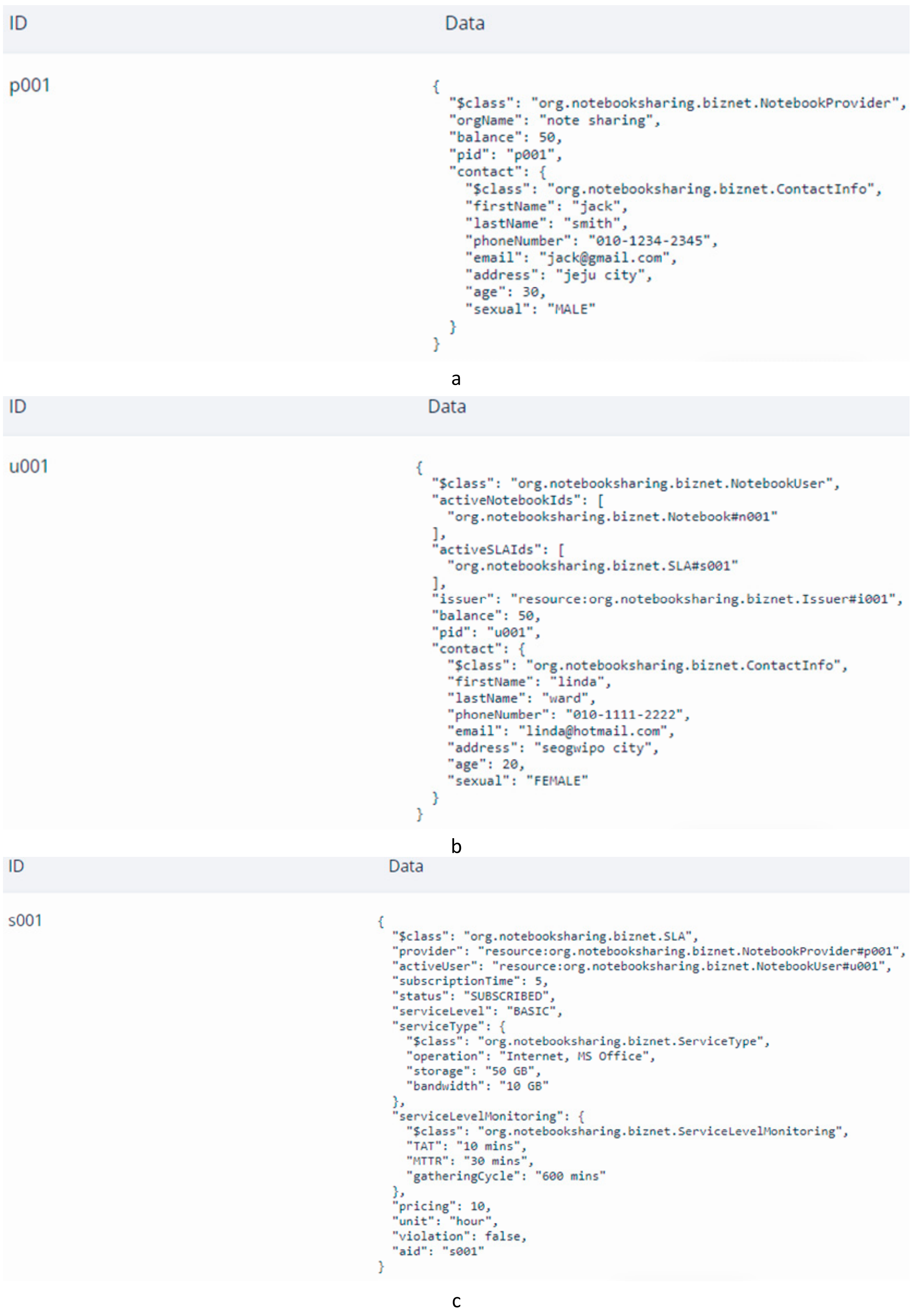
```
http://192.168.0.22:3000/api/system/ping
```

Response Body

```
{
  "version": "0.20.8",
  "participant": "org.notebooksharing.biznet.Issuer#i001",
  "identity": "org.hyperledger.composer.system.Identity#4040be2cd59a2305aa5541e65396d15edb3c56c622a955505c3a33c87b01a8e8"
}
```

Figure 17. Transaction test in the Composer REST Server.

The snapshots in Figure 18 show some of the case study operations results. Figure 18a is a snapshot of the provider profile that is defined in the smart contract. Similarly, the consumer profile is presented in Figure 18b. The specifications of the SLA generated by the provider is illustrated in Figure 18c.



ID	Data
p001	see below

```
{
  "$class": "org.notebooksharing.biznet.NotebookProvider",
  "orgName": "note sharing",
  "balance": 50,
  "pid": "p001",
  "contact": {
    "$class": "org.notebooksharing.biznet.ContactInfo",
    "firstName": "jack",
    "lastName": "smith",
    "phoneNumber": "010-1234-2345",
    "email": "jack@gmail.com",
    "address": "jeju city",
    "age": 30,
    "sexual": "MALE"
  }
}
```

a

ID	Data
u001	see below

```
{
  "$class": "org.notebooksharing.biznet.NotebookUser",
  "activeNotebookIds": [
    "org.notebooksharing.biznet.Notebook#n001"
  ],
  "activeSLAIds": [
    "org.notebooksharing.biznet.SLA#s001"
  ],
  "issuer": "resource:org.notebooksharing.biznet.Issuer#i001",
  "balance": 50,
  "pid": "u001",
  "contact": {
    "$class": "org.notebooksharing.biznet.ContactInfo",
    "firstName": "linda",
    "lastName": "ward",
    "phoneNumber": "010-1111-2222",
    "email": "linda@hotmail.com",
    "address": "seogwipo city",
    "age": 20,
    "sexual": "FEMALE"
  }
}
```

b

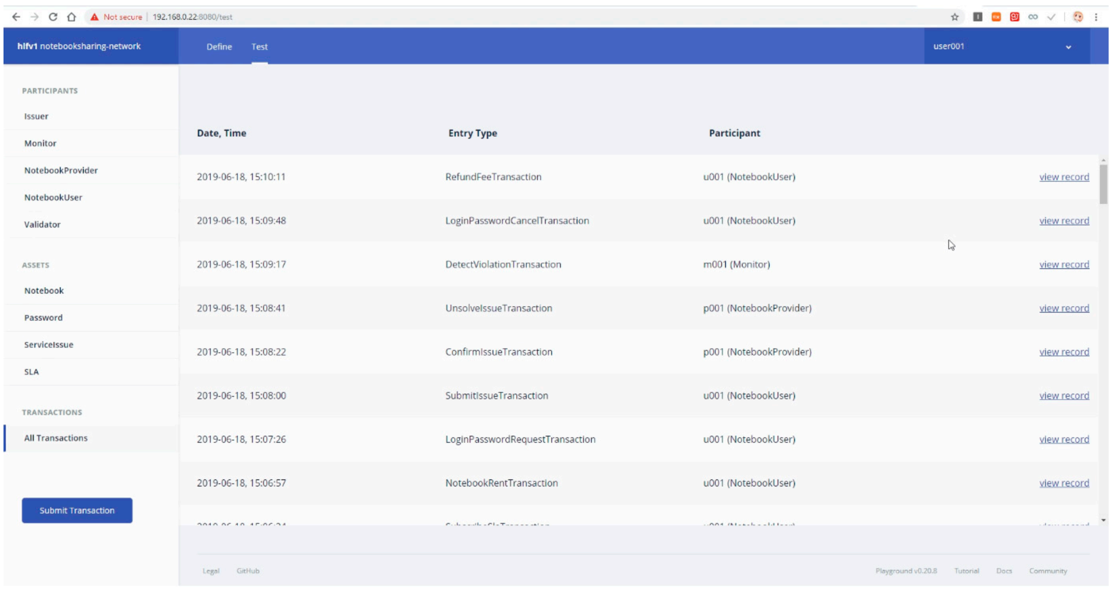
ID	Data
s001	see below

```
{
  "$class": "org.notebooksharing.biznet.SLA",
  "provider": "resource:org.notebooksharing.biznet.NotebookProvider#p001",
  "activeUser": "resource:org.notebooksharing.biznet.NotebookUser#u001",
  "subscriptionTime": 5,
  "status": "SUBSCRIBED",
  "serviceLevel": "BASIC",
  "serviceType": {
    "$class": "org.notebooksharing.biznet.ServiceType",
    "operation": "Internet, MS Office",
    "storage": "50 GB",
    "bandwidth": "10 GB"
  },
  "serviceLevelMonitoring": {
    "$class": "org.notebooksharing.biznet.ServiceLevelMonitoring",
    "TAT": "10 mins",
    "MTTR": "30 mins",
    "gatheringCycle": "600 mins"
  },
  "pricing": 10,
  "unit": "hour",
  "violation": false,
  "aid": "s001"
}
```

c

Figure 18. Snapshot of the case study operation results. (**a**) Provider profile; (**b**) consumer profile; (**c**) SLA specification.

Figure 19 represents the snapshot of the transaction history in the Composer playground, including the date, entry type, and participant. "Date" is an unalterable blockchain ledger record time indicating when the transaction is executed. "Entry Type" presents the transaction type and "Participant" represents the current participant that submits the transaction. The dashboard also provides an entry that shows the detail of a transaction.



Date, Time	Entry Type	Participant	
2019-06-18, 15:10:11	RefundFeeTransaction	u001 (NotebookUser)	view record
2019-06-18, 15:09:48	LoginPasswordCancelTransaction	u001 (NotebookUser)	view record
2019-06-18, 15:09:17	DetectViolationTransaction	m001 (Monitor)	view record
2019-06-18, 15:08:41	UnsolveIssueTransaction	p001 (NotebookProvider)	view record
2019-06-18, 15:08:22	ConfirmIssueTransaction	p001 (NotebookProvider)	view record
2019-06-18, 15:08:00	SubmitIssueTransaction	u001 (NotebookUser)	view record
2019-06-18, 15:07:26	LoginPasswordRequestTransaction	u001 (NotebookUser)	view record
2019-06-18, 15:06:57	NotebookRentTransaction	u001 (NotebookUser)	view record

Figure 19. Transaction history in the Composer web playground.

6. Performance Evaluation of the Case Study

This section offers a comprehensive evaluation to demonstrate the performance of the proposed system in various performance metrics, along with a formula where appropriate. The prototype blockchain network in the experiment consists of four peers and one orderer. The term transactions per second (tps) refers to the number of transactions performed by the blockchain network per second, also called throughput. For this analysis, we used the Hyperledger Caliper [46] as the benchmark simulation tool. We modified the configuration script to simulate the behavior of the case study for the performance evaluation. For the first study, we evaluated the throughput of the proposed system by varying the send rate from 200 tps to 1300 tps. The throughput can be classified into two categories, the read throughput and transaction throughput. The read throughput measures the number of read operations that are completed in a defined period. The transaction throughput measures the number of valid transactions that are committed by the blockchain within the allotted time. Formulas for these two metrics are presented in Equations (1) and (2), respectively. Note that the total number of invalid transactions should be eliminated from the total transactions to yield the total valid transactions. The average read throughput of the proposed system was evaluated by varying the send rate from 500 tps to 3000 tps at a randomly selected system utilization level, and the experimental results are shown in Figure 20. The read throughput grows until it reaches a peak of 2346 tps at the send rate of 2500 tps. Therefore, we get the optimal send rate as 2500 tps for the read throughput in the case of the proposed network. Similar to Figure 21, the optimal send rate for the transaction throughput is 1100 tps as the throughput decreases continuously with the increase of the send rate after that.

$$\text{Read Throughput} = \frac{Total\ read\ operations}{total\ time\ in\ seconds} \tag{1}$$

$$\text{Transaction Throughput} = \frac{Total\ valid\ transactions}{total\ time\ in\ seconds} \tag{2}$$

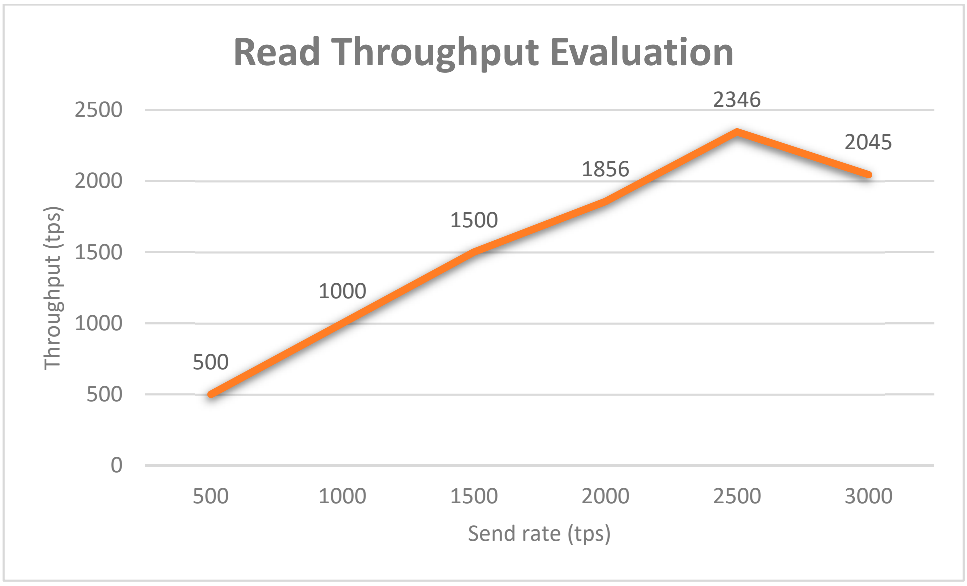


Figure 20. Blockchain network read throughput evaluation.

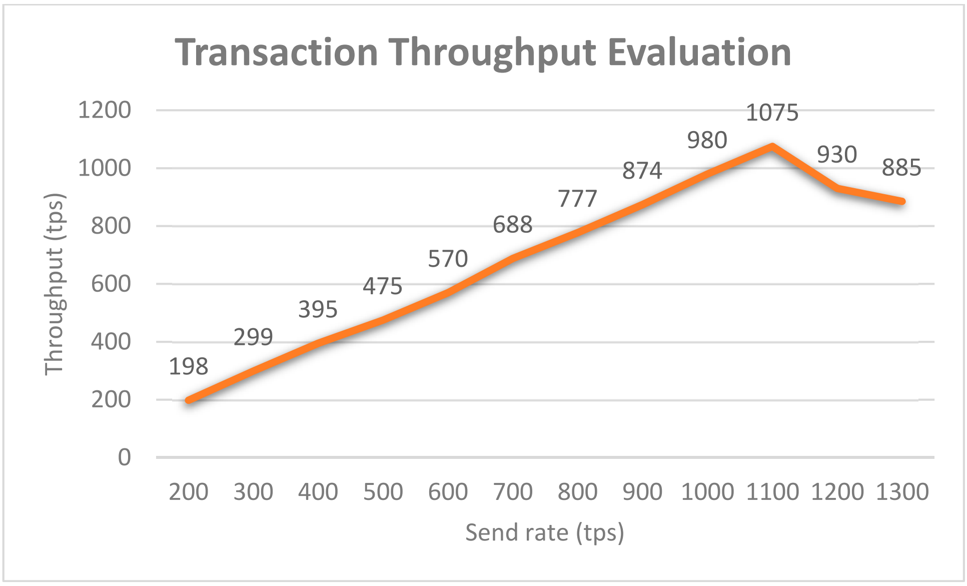


Figure 21. Blockchain network transaction throughput evaluation.

Another study on the network latency was performed to indicate the amount of time taken for a transaction to be executed across the blockchain network. Similar to the first study, the latency also falls into two categories: the read latency and transaction latency. The read latency is the time that the read request is submitted plus the time that the reply is received. The transaction latency is a network-wide view of the amount of time taken for a transaction to be executed across the network. The measurement not only includes the time from the point that it is submitted but also the broadcast time and any correction time because of the consensus mechanism in place. Eyal et al. [47] proposed a useful definition of the network threshold, which is the amount of time for a percentage of the network to commit the transaction. In this study, the percentage of the network was set to 100% since the use of the non-probabilistic protocol like PBFT. General and simplified formulas for computing the read latency and transaction latency are shown in Equations (3) and (4), separately. As shown in Figure 22, the average read latency of the proposed system was evaluated by varying the send rate from 500 tps to 3000 tps 10 times at a random selected system resource utilization level. The read latency has a relatively small increase with the increase of the send rate. However, the latency increases significantly after the send rate of 2500 since the optimal value is exceeded. Similarly, the graph in Figure 23 shows that the average transaction latency increases linearly as the user request rate increases, and this value rises sharply after the optimal send rate of 1100 tps.

$$\text{Read Latency} = \text{Response received time} - \text{submission time} \tag{3}$$

$$\text{Transaction Latency} = \text{Confirmation time} * \text{network threshold} - \text{submission time} \tag{4}$$

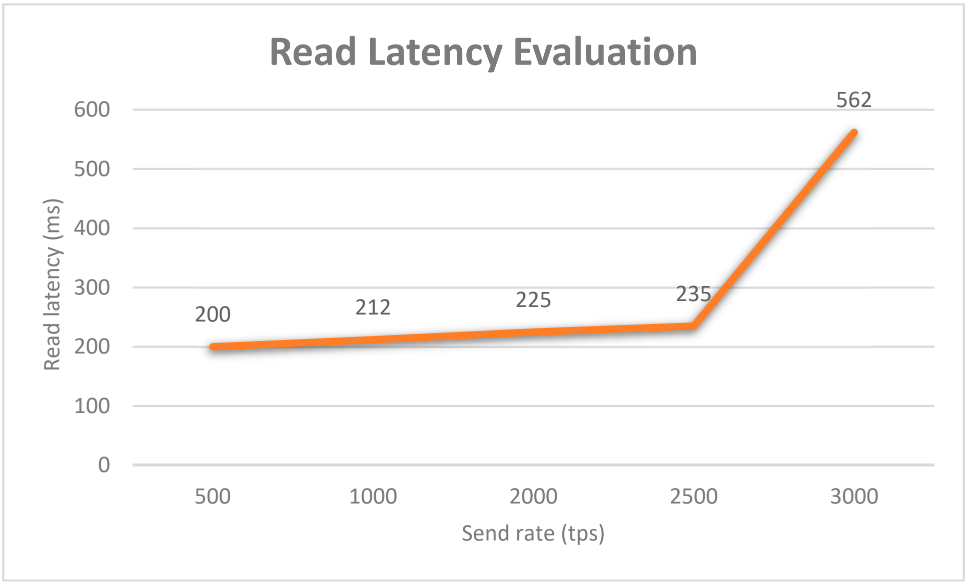


Figure 22. Blockchain network read latency evaluation.

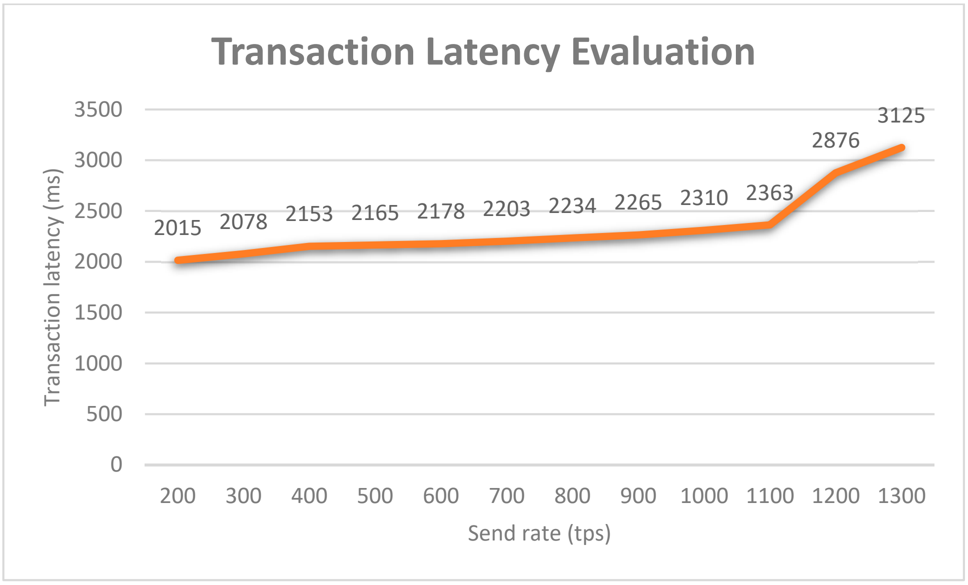


Figure 23. Blockchain network transaction latency evaluation.

In general, Bitcoin takes near 10 min to mine a block, and it can be expected that a transaction takes around an hour on average as at least six confirmations are required before a transaction is finalized. For Ethereum, the average transaction time to mine a block is around 15 s. Moreover, this time cost varies significantly in terms of the network environments. According to Figure 21, the average transaction latency for the proposed blockchain network is around 2.3 s, which races far ahead than most popular blockchain platforms. The reason behind that depends on the nature of the blockchain, which directly impacts the transaction performance. Bitcoin and Ethereum are permissionless blockchains, where anyone can participate, and every participant is anonymous. In order to establish trust among anonymous participants, the permissionless blockchain typically employs a native cryptocurrency or transaction fees to provide economic incentive to offset the extraordinary costs of validating transactions based on the proof-of-work (PoW). On the other hand, the proposed blockchain is built on a permissioned network, where transactions are operated among a set of known, identified participants under a governance model. By relying on the identities of the participants, a permissioned blockchain can use a more traditional PBFT protocol that does not require costly mining. Additionally, a permissioned blockchain can diminish the risk of a participant intentionally introducing a malicious code through a smart contract. Since the participants are known to one another and all actions, whether submitting application transactions, modifying the configuration of the network are recorded on the blockchain following an endorsement policy that was established for the network and relevant transaction type. Additionally, as a consequence, any guilty party can be easily identified and the incident handled in conformity to the terms of the governance model.

Another experiment on the resource utilization of the blockchain network using the Hyperledger Caliper was performed in five iterations. The resource utilization test results describe the maximum, average usage rate of the memory and central processing unit (CPU), as shown in Table 4. For the peer, the average memory allocation was recorded to be 97.35 MB, and the average CPU utility was recorded to be 5.77%. For the orderer node, the average memory allocation was recorded to be 26.4 MB, and the average CPU utility was recorded to be 1.25%. For the CA node, the average memory allocation was recorded to be 5.5 MB, and the CPU utility was recorded to be 0%. The result of the resource utilization experiment demonstrates that the blockchain network has a low rate of the system resources occupation, high reliability, and comfortable user experience.

Table 4. Resource utilization evaluation of the blockchain network.

Type	Name	Memory (Max)	Memory (Avg)	CPU (Max)	CPU (Avg)	Traffic In	Traffic Out
Docker	peer0	107.0 MB	103.2 MB	11.46%	5.53%	4.8 MB	4.5 MB
Docker	peer1	95.6 MB	89.7 MB	13.65%	6.55%	5.2 MB	5.5 MB
Docker	peer2	114.3 MB	108.0 MB	13.92%	5.24%	5.6 MB	11.5 MB
Docker	peer3	92.6 MB	88.7 MB	12.95%	5.75%	5.0 MB	5.6 MB
Docker	orderer	36.6 MB	26.4 MB	5.16%	1.25%	5.1 MB	13.2 MB
Docker	ca0	5.5 MB	5.5 MB	0.00%	0.00%	526 B	0 B
Docker	ca1	5.5 MB	5.5 MB	0.00%	0.00%	456 B	0 B

7. Conclusions

The approach presented herein is based on smart contracts for the automated management of SLAs in the field of the sharing economy. It automatically enforces the payment of the defined subscription fee or the monetary compensation in the case of an SLA violation. All involved parties get to know which entity has breached the SLA, and each party can explore every transaction. Moreover, different facets of the SLA management have been incorporated in the approach, such as the SLA definition, negotiation, and termination. A notebook sharing case study is implemented as the proof of concept on top of the proposed architecture by using the Hyperledger Fabric. We evaluate the system performance in different performance metrics, which indicate a steady level, allowing an effective transaction throughput. Although the coevolution of the blockchain and SLA research studies is still in its infancy, it is the goal of this work to explore the potential applications to improve efficiency and bring automation, to revolutionize robust business solutions in various sharing economy scenarios. The future research direction of this work is to monitor the terms of the SLA and resources while guaranteeing the integrity of the monitored data to verify whether the SLAs are being complied or not. Moreover, businesses considering the blockchain-enabled SLA management have to discuss the security issues related to smart contracts. More standard templates and security-auditing protocols are required to improve the quality of the smart contracts.

Author Contributions: L.H. conceived the idea for this paper, designed the experiments and wrote this paper; D.-H.K. conceived the overall idea of the paper and proofread the manuscript.

Funding: This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2019-2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation). Any correspondence related to this paper should be addressed to Dohyeun Kim.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Horton, J.J.; Stern, L.N.; Zeckhauser, R.J. Owning, Using and Renting: Some Simple Economics of the "Sharing Economy". *NBER Work. Pap. Ser.* **2016**, *42*. [CrossRef]
2. Qi, L.W.; Chao, R.M. Enterprises' Legal Countermeasures to Patent Infringement under the Sharing Economy—Take Mobike's Case as an Example. *J. Tianjin Sino-Ger. Univ. Appl. Sci.* **2018**, *5*, 116–119.
3. Zervas, G.; Proserpio, D.; Byers, J.W. The rise of the sharing economy: Estimating the impact of Airbnb on the hotel industry. *J. Mark. Res.* **2017**, *54*, 687–705. [CrossRef]
4. Yaraghi, N.; Ravi, S. The Current and Future State of the Sharing Economy. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3041207 (accessed on 5 April 2019).
5. Hawlitschek, F.; Teubner, T.; Weinhardt, C. Trust in the Sharing Economy. *Die Unternehm. Swiss J. Bus. Res. Pract.* **2016**, *70*, 26–44. [CrossRef]
6. Katz, V. Regulating the sharing economy. *Berkeley Technol. Law J.* **2015**, *30*, 1067–1126.
7. Cockayne, D.G. Sharing and neoliberal discourse: The economic function of sharing in the digital on-demand economy. *Geoforum* **2016**, *77*, 73–82. [CrossRef]
8. Dyal-Chand, R. Regulating sharing: The sharing economy as an alternative capatilist system. *Tul. L. Rev.* **2015**, *90*, 241–309.
9. De Filippi, P. What blockchain means for the sharing economy. *Harv. Bus. Rev.* **2017**, *15*, 1–6.
10. Sun, J.; Yan, J.; Zhang, K.Z. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financ. Innov.* **2016**, *2*, 26. [CrossRef]
11. Huckle, S.; Bhattacharya, R.; White, M.; Beloff, N. Internet of things, blockchain and shared economy applications. *Procedia Comput. Sci.* **2016**, *98*, 461–466. [CrossRef]
12. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017.
13. Zheng, Z.; Xie, S.; Dai, H.N.; Wang, H. Blockchain Challenges and Opportunities: A Survey. Available online: http://inpluslab.sysu.edu.cn/files/blockchain/blockchain.pdf (accessed on 2 March 2019).
14. Hang, L.; Choi, E.; Kim, D.H. A Novel EMR Integrity Management Based on a Medical Blockchain Platform in Hospital. *Electronics* **2019**, *8*, 467. [CrossRef]
15. Peng, Z.; Jules, W.; Schmidt, D.C.; Gunther, L.; Rosenbloom, S.T. FHIRChain: Applying Blockchain to securely and scalably share clinical data. *Comput. Struct. Biotechnol. J.* **2018**, *16*, 267–278.
16. Dwivedi, A.D.; Srivastava, G.; Dhar, S.; Singh, R. A Decentralized Privacy-Preserving Healthcare Blockchain for IoT. *Sensors* **2019**, *19*, 326. [CrossRef] [PubMed]
17. Khezr, S.; Moniruzzaman, M.; Yassine, A.; Benlamri, R. Blockchain Technology in Healthcare: A Comprehensive Review and Directions for Future Research. *Appl. Sci.* **2019**, *9*, 1736. [CrossRef]
18. Yuan, Y.; Wang, F.Y. Towards blockchain-based intelligent transportation systems. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 2663–2668.
19. Chaudhary, R.; Jindal, A.; Aujla, G.S.; Aggarwal, S.; Kumar, N.; Choo, K.K.R. BEST: Blockchain-based secure energy trading in SDN-enabled intelligent transportation system. *Comput. Secur.* **2019**, *85*, 288–299. [CrossRef]

20. Ouaddah, A.; Elkalam, A.A.; Ouahman, A.A. FairAccess: A new Blockchain-based access control framework for the Internet of Things. *Secur. Commun. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
21. Novo, O. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet Things J.* **2018**, *5*, 1184–1195. [CrossRef]
22. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
23. Hang, L.; Kim, D.H. Design and Implementation of an Integrated IoT Blockchain Platform for Sensing Data Integrity. *Sensors* **2019**, *19*, 2228. [CrossRef] [PubMed]
24. Hawlitschek, F.; Notheisen, B.; Teubner, T. The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy. *Electron. Commer. Res. Appl.* **2018**, *29*, 50–63. [CrossRef]
25. Mubeen, S.; Asadollah, S.A.; Papadopoulos, A.V.; Ashjaei, M.; Pei-Breivold, H.; Behnam, M. Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study. *IEEE Access* **2017**, *6*, 30184–30207. [CrossRef]
26. Szabo, N. The Idea of Smart Contracts. Available online: fon.hum.uva.nl/rob/Courses/InformationInSpeech/ (accessed on 3 March 2019).
27. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
28. Loper, N. 15 Apps That Let You Join the Sharing Economy, Lifehack. 2016. Available online: http://www.lifehack.org/articles/technology/15-apps-that-let-you-join-the-sharing-economy.html (accessed on 3 March 2019).
29. Xu, X.; Pautasso, C.; Zhu, L.; Gramoli, V.; Ponomarev, A.; Tran, A.B.; Chen, S. The blockchain as a software connector. In Proceedings of the WICSA 2016 Proceedings, Venice, Italy, 5–8 April 2016; pp. 182–191. [CrossRef]
30. Lundy, L. Blockchain and the Sharing Economy 2.0. IBM Dev. 2016. Available online: https://www.ibm.com/developerworks/library/iot-blockchain-sharing-economy/ (accessed on 4 April 2019).
31. Glaser, F. Pervasive Decentralisation of Digital Infrastructures: A Framework for Blockchain enabled System and Use Case Analysis. In Proceedings of the HICSS 2017 Proceedings, Waikoloa Village, HI, USA, 4–7 January 2017; pp. 1543–1552.
32. Beenest. Available online: https://www.beenest.com/ (accessed on 5 April 2019).
33. SLOCK.IT Website. Available online: https://slock.it/iotlayer.html (accessed on 5 April 2019).
34. HELBIZ Website. Available online: https://www.helbiz.com/ (accessed on 5 April 2019).
35. La'zooz Website. Available online: http://www.lazooz.org/ (accessed on 5 April 2019).
36. Arcade City Website. Available online: http://arcade.city/ (accessed on 5 April 2019).
37. Bogner, A.; Chanson, M.; Meeuw, A. A Decentralised Sharing App running a Smart Contract on the Ethereum Blockchain. In Proceedings of the 6th International Conference on the Internet of Things—IoT'16, Stuttgart, Germany, 7–9 November 2016; pp. 177–178. [CrossRef]
38. Xu, L.; Shah, N.; Chen, L.; Diallo, N.; Gao, Z.; Lu, Y.; Shi, W. Enabling the sharing economy: Privacy respecting contract based on public blockchain. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, Abu Dhabi, UAE, 2 April 2017; ACM: New York, NY, USA, 2017.
39. Notheisen, B.; Hawlitschek, F.; Weinhardt, C. Breaking Down the Blockchain Hype—Towards a Blockchain Market Engineering Approach. In Proceedings of the ECIS 2017 Proceedings, Guimarães, Portugal, 5–10 June 2017.
40. Rahman, M.A.; Rashid, M.M.; Hossain, M.S.; Hassanain, E.; Alhamid, M.F.; Guizani, M. Blockchain and IoT-Based Cognitive Edge Framework for Sharing Economy Services in a Smart City. *IEEE Access* **2019**, *7*, 18611–18621. [CrossRef]
41. Dorri, A.; Steger, M.; Kanhere, S.S.; Jurdak, R. Blockchain: A distributed solution to automotive security and privacy. *IEEE Commun. Mag.* **2017**, *55*, 119–125. [CrossRef]
42. Axon, L. Privacy-Awareness in Blockchain-Based PKI, Tech. Rep. 2015. Available online: https://ora.ox.ac.uk/objects/uuid:f8377b69-599b-4cae-8df0-f0cded53e63b/datastreams/ATTACHMENT01 (accessed on 12 April 2019).
43. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. *RFC 5280: Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*; IETF: Fremont, CA, USA, 2008.

44. Gaur, N.; Desrosiers, L.; Ramakrishna, V.; Novotny, P.; Baset, S.; O'Dowd, A. *Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer*; Packt Publishing Ltd.: Birmingham, UK, 2018.
45. Hyperledger Composer Playground. Available online: https://composer-playground.mybluemix.net/ (accessed on 8 May 2019).
46. Hyperledger Caliper. Available online: https://www.hyperledger.org/projects/caliper (accessed on 9 May 2019).
47. Eyal, I.; Gencer, A.E.; Sirer, E.G.; van Renesse, R. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the Usenix Conference on Networked Systems Design and Implementation, ser. NSDI'16, USENIX Association, Berkeley, Santa Clara, CA, USA, 16–18 March 2016; pp. 45–59. Available online: https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf (accessed on 10 May 2019).