A Flexible Electronic Data Exchange Framework Based on Consortium Blockchain

Nai-Wei Lo, Sheng-Chiang Chen, Shou-Chun Chang

Department of Information Management, National Taiwan University of Science and Technology, Taiwan
nwlo@cs.ntust.edu.tw, D9916910@mail.ntust.edu.tw, M10509101@mail.ntust.edu.tw

Abstract

With the rise of Internet-based services and platforms to address and support system scalability, data security, service trustability and user privacy protection, it is hard for traditional electronic data exchange systems based on centralized architecture and private network to serve enterprises and organizations well. In this study, an electronic data exchange framework based on consortium blockchain is proposed to support Internet-based flexible decentralized electronic data exchange service. By adopting blockchain and decentralized public key infrastructure technologies, the proposed framework natually achieves system scalability, service trustability, data security and user privacy protection. A system prototype is constructed to evaluate the performance of the proposed framework. The experimental results show that the prototype can generate 60 data exchange contracts per second; it is around 5 million data exchange volume in 24 hours.

Keywords: Consortium blockchain, Smart contract, EDI, Quorum, Decentralized Public key infrastructure

1 Introduction

The earliest study regarding EDI [1] began in the 1970s. Because information systems used by organizations vary, the formats of generated data differ. Thus, manually converting the data format during data exchange is necessary. After the data are converted, they are manually imported into the organization's information system. These duplications and cumbersome tasks create major obstacles in business processes. Therefore, organizations seek to optimize such processes and reduce data exchange costs through EDI. With the development of EDI applications, the American National Standards Institute (ANSI) developed a more versatile standard in 1979, namely ANSI X12, which is mainly used by domestic organizations in the United States (US). Other regions use the United Nations/ Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT) standard developed by the UN Economic Commission for Europe (ECE) in 1986, which has become the international standard ISO 9735:1988 [2].

Electronic Data Interchange (EDI) has not been widely used since its initial development in the 1990s because the cost of creating exchange systems and mechanisms is high, and the subsequent maintenance costs are considerable. Basically, only large organizations have the ability to create exchange systems [3]. Together with their centralized architecture, the stability and data security of such systems are greatly doubted.

One example is the electronic exchange of official data between government agencies. Take the electronic data exchange system of the government of the Republic of China for example; through the establishment of 58 exchange centers in various counties and cities, more than 20,000 units are provided for the exchange of electronic data [4]. In addition, the identity of each is determined by the certificate issued by the Government Certificate Authority based on the Government Public Key Infrastructure [5-6]. However, the process of electronic data exchange is not fully encrypted [4], which means that official data can be stolen easily. In 2016, the government began to promote a new generation of national, shared, official electronic data exchange systems to establish a fully encrypted system [7]. The new generation of official electronic exchange systems has been online since 2018 and is expected to fully replace the original generation in 2019. Although the new generation of systems has introduced full encryption and removed the eClient design, the overall framework is based on a centralized design wherein a single point of failure (SPOF) and domain name system hijacking problems remain difficult to solve.

The rapid development of blockchain technology has led to numerous innovative services [8], and the most widely known application is cryptocurrencies. The emergence of Bitcoin [9] in 2008 allowed the public to gain a preliminary understanding of blockchain, the main characteristics of which are decentralization, distributed ledgers and data immutability. The blockchain technology uses hash

*Corresponding Author: Nai-Wei Lo; E-mail: nwlo@cs.ntust.edu.tw
DOI: 10.3966/160792642020092105008

values, timestamps, and digital signatures to ensure that data are immutable and achieve decentralization through distributed ledgers. Each node has a complete ledger, and thus, SPOF and data monopoly problems are not of concern. The blockchain is connected by a bunch of blocks according to the hash values of the previous block. Each block contains the hash value of the previous block and multiple transaction records. In addition, the block contains a timestamp to avoid double-spending attacks, which is used to prove the data validity in a certain time because the calculated hash value must be unique. With these mechanisms, the hash values will not match as long as the data of any block in the chain have been tampered with; thus, the data almost certainly cannot be tampered with once they are saved in the blockchain. After the block containing transaction data is created, it will be released to each node through an end-to-end mechanism. Miners will start to perform block content verification and calculate the hash value. After verification, the block will be added to the blockchain. A fast miner can obtain some rewards (i.e., Bitcoin), a consensus process called "proof of work."

Ethereum [10] emerged in 2014 and was termed Blockchain 2.0 technology. It created a new term, smart contract, which allowed more complex programs to save and execute on a blockchain to form a shared computing environment consisting of many nodes. After a smart contract is written through Solidity, it is converted into bytecode by a Solidity compiler and deployed to the blockchain. Once the contract is successfully deployed, a contract address can be obtained to operate or call functions inside the contract. Regardless of whether it is a deployment contract or contract call function, it will cost a unit called gas, the amount required of which is calculated based on the complexity of the contract or function. Hence, infinite loops can be avoided because the contract will be terminated if the gas is insufficient when the function is executed, and it will not always consume computing resources of the node. In addition, the gas price must be set. The node will first select a high price transaction to compute; if the price is set too low, it may never be executed. The execution environment of smart contracts is called the Ethereum virtual machine. Each node executes a smart contract operation through the Ethereum virtual machine, and thus, the gas mechanism is required to ensure that node computing resources are not abused. When a transaction to operate a smart contract occurs, all nodes work in parallel and obtain a consensus before returning the result. This mechanism ensures that the smart contract has a fault tolerance mechanism and zero downtime, and most crucially, the state change after execution of the smart contract still possesses the data immutability characteristic.

The data on a blockchain are public and each node has a copy, which poses a great threat to data privacy. Under Ethereum's public blockchain, the content of smart contracts is also open, indicating that anyone can retrieve their data and decompile their source code. Kosba et al. [11] proposed a solution to solve the smart contract code disclosure problem, namely Hawk. With this technology, the programming of smart contracts can be divided into public and private contracts: a public contract is executed by the node of the blockchain (similar to Ethereum's smart contract), whereas a private contract is executed by the manager. This means that users must entrust a manager to execute an off-chain private contract to ensure that the data are not disclosed. The open source project Quorum [12] led by J.P. Morgan Chase applied a similar concept by adding private transaction functions on the basis of Ethereum. A transaction manager was used to handle privacy, and public and private transactions were dealt with separately [13].

After the emergence of permissioned blockchains, the transparency problem in the original blockchain was solved. Among numerous applications, data is usually valuable or confidential and unlikely to be shared publicly. Therefore, it is difficult to use in a blockchain that is transparent and immutable. However, through a permissioned blockchain, data disclosure can be avoided and the advantages of data immutability and decentralization are provided. With the development of such technologies, the research direction has begun to extend to more complex intellectual assets and smart contracts. Moreover, the application areas are not limited to the financial industry but extend to governments and the medical, art, and culture industries.

The nature of blockchain technology can alleviate the problems of EDI systems. It can reduce the risk of exchange system downtime through decentralization, avoid inconsistent data exchange through its data immutability characteristics, and save exchange records in the chain to prevent exchange records not being recognized or maliciously falsified or forged, as well as increase credibility for both parties.

When blockchain technology is adopted as system infrastructure, digital identities of users should have a corresponding solution to fit in blockchain architecture. Bakre et al. [14] proposed using blockchain technology to save identity data and allow users self-sovereignty over such data. Diebold [15] proposed a method for managing identity data using Ethereum-based smart contracts, allowing users to deposit their digital identity into the contract and save their actual identity data in the InterPlanetary File System (IPFS) [16]. Jung et al. [17] proposed a blockchain-based name resolution service to provide a query for the correspondence between IoT device IDs and IP addresses. Decentralized Public Key Infrastructure (DPKI) service [18] is proposed to avoid attacks utilizing the weakness of conventional PKI services.

There are discussions on the disadvantages of blockchain technology and the corresponding impact to application systems that adopt blockchain infrastructure [19-21]. The major concerns on blockchain technology are high energy consumption, wasting computing power, requiring huge data replication space (for each network node), insufficient user privacy protection, selfish miner problem, user authentication and user key privacy. However, most of those concerns can be alleviated or ellimated by utilizing consortium blockchain to control the number and quality of mining nodes and adopting distributed or decentralized security mechanisms to achieve user privacy protection, identity protection and user authentication. The proposed framework actually uses consortium blockchain structure along with Quorum package and DPKI service.

The objective of this study was to alleviate the difficulties encountered in conventional EDI (e.g., high establishment costs and no full encryption mechanism leading to data leakage or SPOF problems) through the characteristics of blockchain technology. Exchange operations are executed via a permissioned blockchain network through a privacy mechanism that only allows relevant recipients to obtain the exchanged data in private transactions, as well as ensures that only one physical file exists in the entire exchange system for legitimate recipients to download. This reduces the risk of files being hacked, and only the exchange records are saved on the blockchain to avoid the rapid expansion of node data.

This study proposed a flexible electronic data exchange framework based on consortium blockchain technology, which has the following characteristics that can be used to solve the risks that a centralized exchange system may face:

- Decentralized exchange framework: This reduces the SPOF risk.
- Decentralized identity recognition mechanism: This prevents identity data being controlled by either party and reduces the risk of man-in-the-middle attacks.
- Data immutability: This ensures that the data exchange of either party is not subject to malicious tampering.
- Data confidentiality: This only allows individuals in the exchange and data to be known by the true recipient.
- Flexible exchange framework: This can establish intra- or inter-organizational exchange networks according to requirements, as well as establish mixed-situation exchange networks.
- Data security: Encrypted data will not flow through the entire exchange network and will only exist in the sender's system for legitimate recipients to download, thereby reducing the risk of being hacked.
- Easy management: This framework provides an easy mechanism to manage the exchange network inside and outside the organization.

The exchange framework proposed by this study can not only be used for electronic data exchange. It also allows internal units of general organizations to exchange data and can expand the data exchange between related organizations; for example, data exchange between various government ministries, members of industrial alliance organizations, and large organizations' subsidiaries and parent organizations. In addition, the framework can be integrated into more complex applications, such as direct data exchange between internal units of different organizations.

2 Framework Architecture

The system uses Quorum as its underlying exchange network as well as the eXchange agent (XAgent) proxy service to operate the blockchain nodes for exchange operations. According to the application context, the system can be divided into intra-organizational, inter-organizational, and mixed exchange structures.

The unit structure in each scenario is identical. Each unit has a Quorum node and an XAgent proxy service, and users can send and receive files through XAgent. The physical exchanged file only exists in the XAgent when file exchange is executed, and will not exist in the blockchain network. After receiving the contract, the recipient will call the smart contract to obtain the file address to download it from the sender's XAgent.

- XAgent: This is a file exchange proxy service, which provides users or programs to send and receive files through the blockchain from a web interface or Web API; it manages the exchanged files in a unified manner and is responsible for storing or retrieving them in a file isolation area.
- Quorum node: This is the Quorum blockchain node, which is responsible for executing smart contracts and handling blockchain transactions.
- File manager: This is responsible for the file (or data) management of smart contracts; it is a private smart contract and responsible for saving correspondence between exchanged files' hash values and contract addresses.

Under the exchange scenarios within an organization, all exchange units are assumed to belong to the same organization or a subsidiary, and can be used for data exchange between each business department within the organization or between the parent and subsidiary company.

The management node will be served by the main business group or parent company (Unit 1 in Figure 1), whereas other business groups or subsidiaries are general nodes. The identity between nodes is managed by the eXchange Manager (XM). After an internal network is established, the management node will create an XM and notify all nodes to register their identities. The node uses the XM to confirm the identity of the creator or recipient when files are sent and received. During XM data management, the management node can change or delete all data in the XM to control the members inside the exchange network, whereas a general node can only update its own identity data.

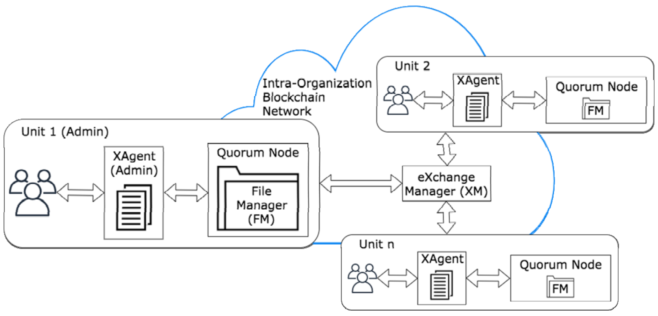

Figure 1. Intra-organizational data exchange structure in the blockchain-based framework

In an inter-organizational context, each node represents the management node of the organization. This framework design only allows management nodes to create exchange nodes between organizations. The identification of each is no longer achieved through an XM; instead, it is achieved using a DPKI service to confirm the identity of each, as shown in Figure 2.

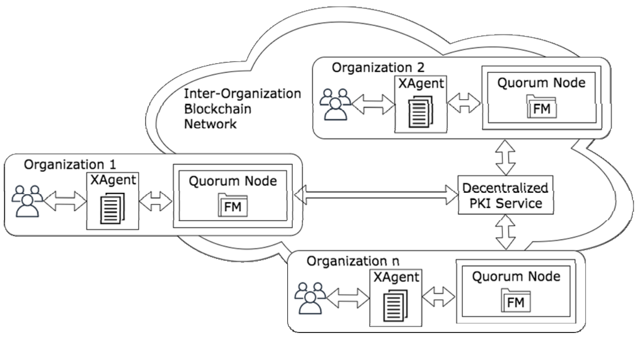

Figure 2. Inter-organizational data exchange structure in the blockchain-based framework

Members of this exchange network are collectively referred to as alliance organizations, and they will recommend an initiating organization to establish an external exchange network according to agreements between them. All organizations first register their own organization data into the DPKI service, and the initiating organization registers an identity for the external exchange network in the service. After registration, it will retrieve the Decentrialized Identifier (DID) of the exchange network and send it to other alliance organizations to join the exchange network. When joining the exchange network, each organization retrieves the DID Descriptor Object (DDO) from the DPKI service through the DID of the exchange network and establishes a connection according to the XAgent service in the DDO.

In the DDO of the exchange network, the public key of all alliance organizations in the exchange network is saved as a limitation of the exchange network's members. If the public key of an organization does not exist in the DDO, it cannot participate in the exchange process because other alliance organizations cannot retrieve it from the DDO for verification; thus, file sending and receiving cannot be performed.

Combining the aforementioned structures can achieve a more complete exchange mechanism. In Figure 3, the XAgent of Organization A has an intra-Quorum node and two inter-Quorum nodes, whereas Organization B only has inter-nodes, and the exchange of internal units may use the old exchange system.

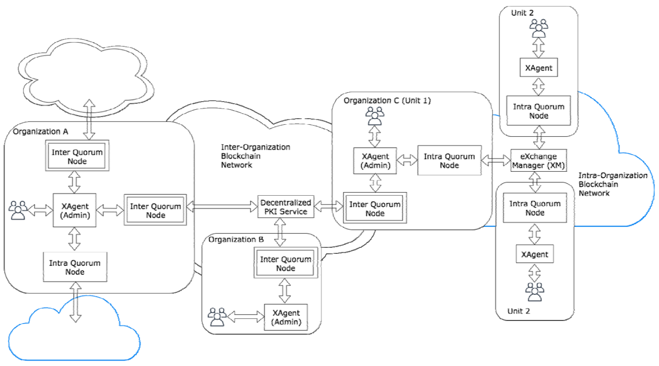

Figure 3. Mixed data exchange structure in the blockchain-based framework

Organization C has an intra-node and an inter-node. The intra-node consists of Units 1, 2, and 3, and the management node is served by Unit 1. When an internal unit wants to send a file to the alliance organizations, an XM can be used to query which external alliance organizations are file-exchangeable and then forward the file to the external alliance organization through the management node's XAgent (Unit 1). The characteristics of this structure are organized as follows.

- The management node's XAgent can only have one intra-node but multiple inter-nodes.
- The general node's XAgent can only have one intra-node.
- The XM of the internal exchange network can provide an internal unit to query which exchangeable external alliance organizations there are.
- When the management node's XAgent creates an inter-node, it synchronically updates the data back to the XM.
- Each blockchain network is independent of each other; therefore, if an organization wants to send files outside of itself, it must forward them through the management node's XAgent.

This framework consists of three smart contracts: Electronic Data eXchange (EDX), File Manager (FM) and eXchange Manager. The EDX contains the hash value, encryption key, file download API location, and recipient lists of files and attachments, which can be used for identity control in the data exchange. When recipients wish to download a file, the contract will confirm whether the recipient exists in the contract's recipient list, and a null value will be returned if he or she does not. When recipients use the file's hash value to call download API to retrieve data, the sender's XAgent will pass these data to the contract for identify confirmation. If the identities do not match, the file will not be returned, whereas if they do match, the physical file and signature are returned, ensuring that the returned file is indeed returned by the sender.

In addition, a function that allows the sender to terminate the exchange is provided. It can only be called by the creator of contract. Once it is terminated, all functions in the contract will only return null values. Therefore, even if the recipient has a file's hash value and a download API location, the sender's XAgent will fail to confirm the identity of the recipient, and thus no data will be returned. Moreover, the contract provides a function to recover the exchange, which also can only be called by the contract creator. After the exchange is recovered, the function of the contract will return to its original value.

Under this mixed structure, a forwarding exchange contract will exist when an internal unit wants to send a file to external alliance organizations or directly to the internal unit of alliance organizations. Because the internal and external exchange networks are independent, the XAgent acts as an intermediary. The forwarding exchange contract indicates that when an internal unit wants to send a file to an alliance organization, the internal unit will create an internal exchange contract and pass it to the management node. The management node's XAgent will then create an external exchange contract and forward it to the alliance organization. The external exchange contract is the forwarding one. After the management node forwards the exchange contract deployment, the address of the forwarding contract will be saved to the internal exchange contract for subsequent use.

File Manager contract is mainly responsible for saving the correspondence between the exchange contract and physical file, allowing the XAgent to query the physical file and corresponding exchange contract address. The internal storage structure is constructed as dictionary, whereas key is the hash value of the exchanged file, and value is the address of the exchange contract. Quorum node manager default is the contract creator, which can delete or recover the corresponding record. In addition, it provides a transfer function for the manager and retains flexibility for subsequent maintenance or account replacement.

Exchange Manager contract is generally used for internal exchange to confirm the identity of each unit and query the public key and account number of the node to which the unit belongs. Mixed structure provides query of which alliance organizations exist. Internal node can directly send a file to the management node, and the management node forwards it to external alliance organizations. In the receiving process, the recipient can also determine whether the sender still exists in the internal exchange network. Files cannot be retrieved if the sender has been removed from the network.

Data creator can maintain the data that he has created to meet the needs of replacing public key or account. In addition, node manager can force deletion or recovery of node data, allowing the management node to have the authority to control the nodes that are internally exchanged. As for the data of external exchange organization alliance, it is also maintained by the management node. After completing the establishment of inter-organizational exchange network, relevant alliance organization data will be saved in the contract for internal nodes to query. When changes occur in the members of alliance organizations, data can be updated to the contract.

XAgent is the core role in the entire exchange framework. It is responsible for transforming complex blockchain operations into simple web operations as well as providing APIs for third-party programming. It is mainly responsible for communicating with the blockchain nodes and sending and receiving data, and moreover, it is responsible for forwarding files under the mixed structure. Furthermore, it is in charge of saving exchange files to a physical file isolation area. The isolation area can be a file server of a completely different computer or a folder on the same computer's hard disk, which is selected according to the needs of organization. Larger organizations may have the ability to establish independent file servers to provide improved security and independence, whereas smaller organizations can only split a small portion of the hard disk to create a file isolation area.

In the entire exchange process, the actual encrypted exchange file will only exist in the file isolation area of the sender's XAgent, and will not be saved in the blockchain network. The recipient downloads the actual exchange file through the API provided by the sender's XAgent. Through this mechanism, preventing files from flowing through the entire exchange network and reducing the risk of brute-force attacks are possible.

System permissions of XAgent are divided into general and management nodes. In addition to the functions of exchange process and file storage, the management node provides the function of establishing and maintaining the exchange network, which allows organizations to create an intraexchange node and an infinite number of interexchange nodes. Moreover, it serves as a bridge for transferring the intraexchange network to an interexchange network, and provides a more complete exchange mechanism.

Originally, intra-organizational identity verification mechanisms were achieved through XM, which is managed by the management node. However, in the context of inter-organizational or mixed structures, a single alliance organization cannot be fully trusted or left alone to manage the XM because each organization may be in a competitive or mutually beneficial relationship. Therefore, in an inter-organizational or mixed structure, the framework uses a DPKI service to handle the identity verification mechanism. In the DPKI service, identity data between the alliance organizations are not controlled by any of them; thus, malicious tampering or falsified identities can be avoided.

This framework does not establish a DPKI service of its own, but rather allows members of the external exchange network to freely agree on which DPKI service to use. The initiating organization of the external exchange network will record the agreed DPKI service used by its alliance organization to identify data of the exchange network; thus, the alliance organizations in the exchange network use the same DPKI service.

The organization's identity data must include the public key for signatures, signature verification algorithm, and XAgent service endpoint. The identity of the exchange network must include the DPKI service endpoint, XAgent service endpoint of the initiating organization, and public key and signature verification algorithm for each alliance organization within the exchange network. When executing the receiving operation, the public key in the identity data can be used to determine whether the alliance organization still exists in the external exchange network, because changes may occur in the alliance organization. When an organization is removed, the identity of the exchange network can ensure it can no longer download any exchange files.

3 Data Exchange Process

The proposed data exchange framework is very flexible. It can support three general scenarios: intra-organizational case, inter-organizational case, and mixed case. To simplify our explanation for the data exchange operation process of the proposed framework, only inter-organizational case is addressed in this study.

3.1 Inter-organizational Data Exchange Initialization

Under an inter-organizational data exchange network, the identity verification mechanism changes from an XM to a DPKI service because each organization must have a physical identity. As shown in Figure 4, All organizations must first register their information with the DPKI service; if it is the initiating organization of the exchange network, it must register the exchange network's identity with the DPKI service and pass the identity DID to other alliance organizations, allowing them to use this DID to retrieve the identity DDO of the exchange network from the DPKI service to join the shared exchange network. After all the members of the exchange network create an inter-Quorum node, they can join the specific exchange network through the service endpoints in the DDO. After joining, they can establish an FM that can only be accessed by its own node for exchange use. The detailed process is as follows:

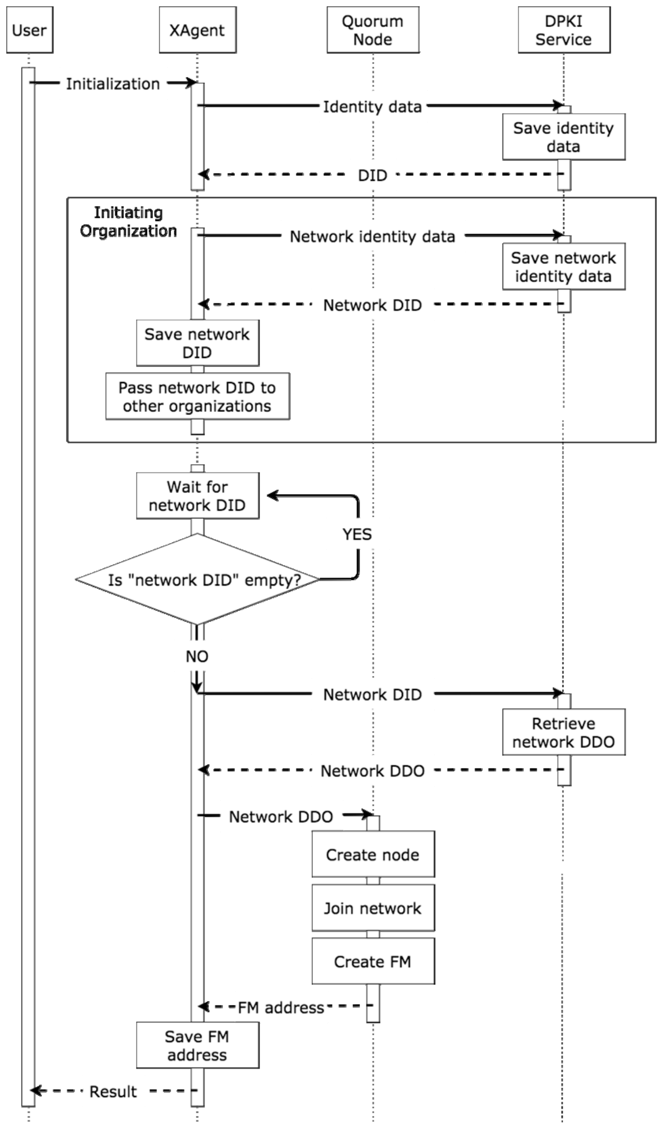

Figure 4. Flow chart of inter-organizational data exchange initialization

(1) Initialize the exchange network: Join the exchange network based on negotiation results of the alliance members.

(2) Register its own organization's information to the DPKI service: Registration must be performed first because the identity of the external exchange network is managed by the DPKI service.

(3) Initiating organization registers the exchange network data to the DPKI service: This organization is also the administrator of the exchange network; thus, the exchange network identity must be registered to the DPKI service.

(4) Retrieve the identity DDO through the DPKI service after obtaining the DID of the exchange

network: Alliance organizations can use this DID to retrieve the identity DDO.

(5) Join the exchange network after interexchange node creation: Join the exchange network according to the service address in the exchange network's identity data.

(6) Establish a private FM: Similar to the function of inter-organizational initialization, which is for exchange purposes.

3.2 Inter-organizational Data Outbound Process

Figure 5 presents the inter-organizational outbound process, i.e., a user sends a file or a set of data through the framework to a receiver located at another organization. Different to intra-organizational outbound process, identity verification uses the DPKI service instead of the XM. The process is similar except that the identity verification process and intra-organizational scenarios are different. Users must first generate an encryption key to encrypt and compress the exchange file through the AES algorithm. Subsequently, they call the XAgent to retrieve the identity DDO of alliance organizations using the DID of the alliance organization to query the DPKI service. Under this scenario, the XAgent may connect to multiple exchange networks of the alliance organizations. However, only one intra-organizational exchange network is required. Therefore, users are required to specify which exchange network DDO is to be retrieved when calling for the XAgent. After the DDO is retrieved, a recipient list is generated according to the requirements for use in generating exchange contracts. After the exchange contract is compiled, it is deployed to the inter-organizational exchange network through the Quorum node, and the address of the contract is retrieved after successful deployment. Subsequently, this address is saved in the node's FM for subsequent file downloading to verify the recipient's identity. The detailed steps are as follows:

(1) Generate an encryption key to encrypt and compress the file: After the user generates the encryption key, the AES algorithm is used to encrypt and compress the file.

(2) Query the recipient data: The user can query the recipient data through the XAgent according to his or her requirements.

(3) Query the DPKI service: The XAgent will query the alliance organizations' identity DDO through the DPKI service according to the DIDs of the exchange network and alliance organizations.

(4) Generate a recipient list: A recipient list can be generated after the user receives the recipient data.

(5) Upload the file and recipient list: The user passes the encrypted exchange file and recipient list to the XAgent for outbound operation.

(6) XAgent executes the outbound contract.

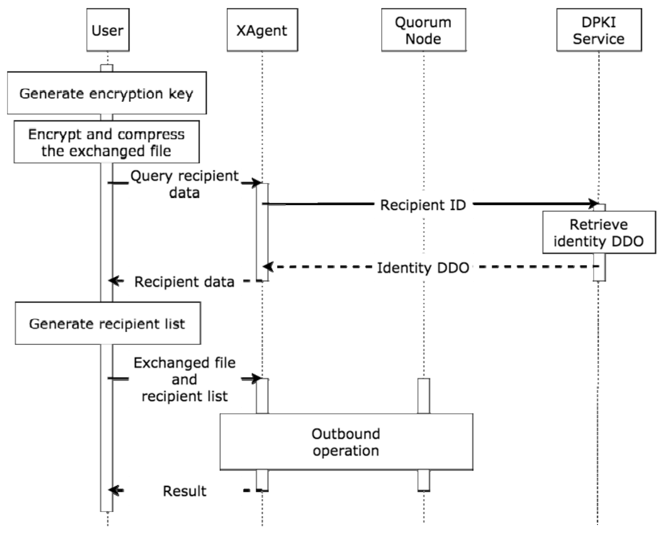

Figure 5. Inter-organizational data outbound process

3.3 Inter-organizational Data Inbound Process

Figure 6 presents the inter-organizational inbound process, i.e., a user receives a file or a set of data through the framework, in which the data is sent by a user located at another organization. Users execute an inbound operation through the XAgent, and the XAgent retrieves EDX data from the Quorum node and adds a signature to the hash value of the exchange file. The signature and hash value are sent to the download API, the XAgent of the sender will perform verification after receiving the request, and the public key of the recipient is retrieved from the signature. After the public key is retrieved, the exchange network DID recorded in its own setting is used to retrieve the exchange network identity DDO through the DPKI service. Moreover, whether the public key exists in the exchange network's identity DDO is checked, and if the identity is confirmed, the recipient is indeed a member of this external exchange network. Subsequently, the file's hash value is inserted into the FM to query the corresponding EDX address, and this address is used to verify whether the recipient exists in the recipient list in the EDX. After verification is passed, the exchange file is retrieved from the file isolation area, and the file's hash value is compared to ensure the file has not been maliciously tampered with. A signature is added to the hash value of the correct file and is returned to the XAgent of the recipient. After the exchange file is received, the recipient's XAgent will perform verification and retrieve the public key of the sender to confirm that the sender still exists in the exchange network. Finally, the file is decrypted to complete the inbound process. The detailed steps are as follows:

(1) Execute an inbound operation: The user executes an inbound operation through the XAgent.

(2) Query the EDX address: The XAgent retrieves the relevant EDX address through monitoring the exchange network.

(3) Execute the "prepare download parameters" step.

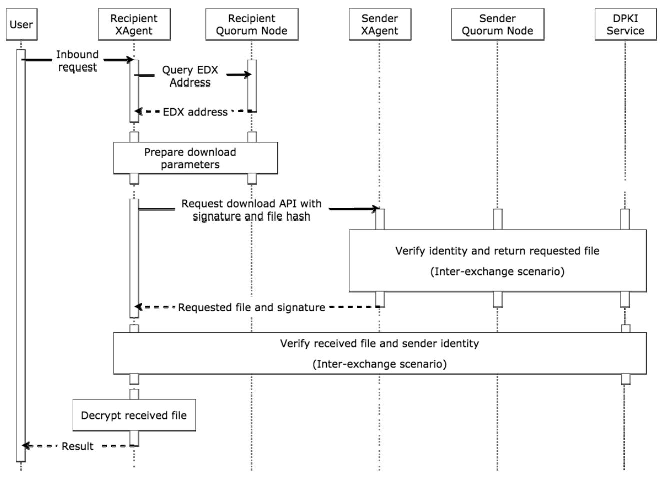
User
Recipient XAgent
Recipient Quorum Node
Sender XAgent
Sender Quorum Node
DPKI Service
Inbound request
Query EDX Address
EDX address
Prepare download parameters
Request download API with signature and file hash
Verify identity and return requested file (Inter-exchange scenario)
Requested file and signature
Verify received file and sender identity (Inter-exchange scenario)
Decrypt received file
Result

Figure 6. Inter-organizational data inbound process

(4) Pass signature and file hash values to the sender's XAgent for file downloading: Call the download API through the Web API.

(5) Sender verifies the signature and extracts the recipient's account: This step confirms the recipient's identity and avoids the forged identity problem.

(6) The sender queries the exchange network DDO through the DPKI service: The sender queries the exchange network identity DDO through the DPKI service using the exchange network DID in the setting.

(7) Query whether the recipient exists through the exchange network DDO: This confirms whether a public key exists in the DDO after retrieving the exchange network identity DDO.

(8) Retrieve the corresponding EDX address through the FM: These data are saved in the FM during outbound, and the sent request is invalid if a record is not found.

(9) Confirm the recipient's identity according to the EDX address: The IsRecipient function in EDX is called for confirmation.

(10) Retrieve the physical file from the isolation area and compare hash values: The file is verified again to avoid tampering after it is saved.

(11) Add a signature to the hash value and return it to the recipient with the exchange file: This ensures that the file is indeed returned by the sender.

(12) Recipient checks and extracts the sender's public key: After the signature verification is passed, the sender's public key is extracted from the signature.

(13) Recipient queries the exchange network DDO through the DPKI service: The is performed using the exchange network DID in the setting.

(14) Query the existence of the sender through the exchange network DDO: This step confirms whether the public key exists in the DDO after retrieving the exchange network identity DDO.

(15) File decryption: Decrypt the file after confirming the file source.

4 Prototype Implementation and Analysis

The framework prototype uses Quorum 2.0.2 as its underlying blockchain network, and underlying Quorum is a core of Ethereum, with additional mechanisms added to handle privacy problems. During the data exchange process, data should only be received by the recipient. However, all data are open on Ethereum, which risks brute-force attacks even if the data are encrypted. Therefore, Quorum's privacy mechanism is required to ensure the data are received by only the true recipient. Another benefit of Quorum is that it is almost compatible with all the existing Ethereum structure; thus, many resources can be shared mutually. The XAgent is a web application written in the ASP.NET Core framework, allowing users to send and receive files directly through the web page. Furthermore, it provides Web API to allow third-party applications to be integrated directly. Because it uses .NET Core [22], it also supports cross-platform application and can be installed on Windows, Linux, and macOS. Communicating components with blockchain uses Nethereum [23], which is also based on .NET Core, allowing users to directly operate Quorum in .NET without having to go through the frontend using Web3.js [24]. In the context of internal exchange, identity management is handled through the XM, and the XAgent allows users to interact with the XM through a web interface. However, in the case of external exchange or a mixed structure, the DPKI service is used instead. To communicate with the DPKI service, it is necessary to retrieve the data from each DPKI service through a Universal Resolver [25], but only Java and Python are used in the currently implemented version.

For the experimental environment, an Ubuntu 16.04 operating system environment with Intel Xeon E5-2620v4 CPU, 4GB memory and 50GB hard disk space was utilized, and the Quorum nodes and XAgents were run internally through Dockers. Data exchange transaction requests were generated by another independent computer, which was directly connected to the framework prototype. The request generation server is built with Windows 10 operating system, Intel i5-6200U CPU, 16GB memory and 240GB SSD. Adopted software packages are Quorum 2.0.2, Docker 17.12.1-ce3, Docker Compose 1.17.1, Node.js v8.10.0 and .NET Core 2.0. Numerous transaction requests were outbound from the request generation server to the framework prototype in the experimental environment through a multithreading asynchronous method to simulate the simultaneous exchange of files in numerous units.

The system experiment was divided into two parts: the first was a prototype stress test, which simulated numerous concurrent exchange contract deployment requests as well as tested the limit of how many requests could be processed per second; the second was a test of the intra-organizational exchange process, which disassembled each process, conducted independent testing, and finally integrated the testing of the entire process.

4.1 Prototype Stress Test

The stress test scenario was designed as follows: 1000 users simultaneously generate different exchange contracts to execute outbound operations. They simultaneously press the outbound button on the XAgent of their own unit to make XAgent issue a request for a deployment contract to the Quorum node. This test contains four scenarios with the number fo recipients increased from one to four gradually.

Each test scenario was conducted five times to get the experimental results in average. The experimental results are shown in Figure 7 to Figure 10. Figure 7 shows the total number of created blocks increases when the number of recipients increases. The increase of the total number of recipients involved in a message exchange has obvious impact on the Quorum processing speed as shown in Figure 8. In Figure 9, it shows that the block creation speed is relatively stable when adding more recipients. In general, approximately 19.16 to 19.47 blocks were created per second. Notice that a transaction is created when deploying a contract into the Quorum blockchain network. Therefore, the number of transactions is equivalent to the number of contract deployments in Figure 10. The average number of transactions per second was 60.82 as shown in Figure 10. In Figure 10, the processing efficiency was considerably reduced when the number of recipients increased. This is because the number of created blocks were increased when the number of recipients increased as shown in Figure 7. However, since the block generation speed was relatively stable as shown in Figure 9, it indicates that more time is required to generate all blocks when the number of recipients increases. As the Quorum network has the same processing capability on verifying and committing transactions through consensus agreement, longer time for committing all transactions is required when the total number of blocks increases. In summary, when the number of recipients increases, Quorum network requires more time to complete the same amount of transactions (i.e., contract deployment). Therefore, the number of transactions per secend will be reduced as shown in Figure 10.

4.2 Intra-organizational Exchange Process Testing

The outbound process can be split into exchange file uploading and exchange contract generation; both were tested separately before testing the entire outbound process.

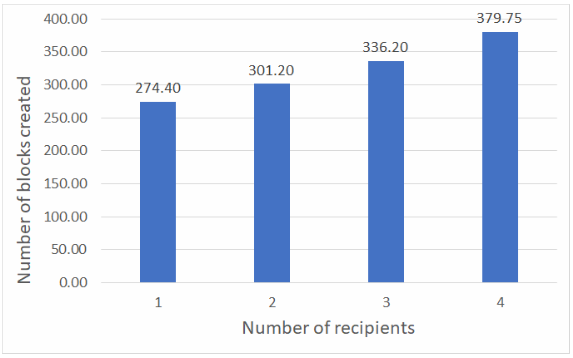

Figure 7. The bar chart of total number of created blocks based on different number of recipients

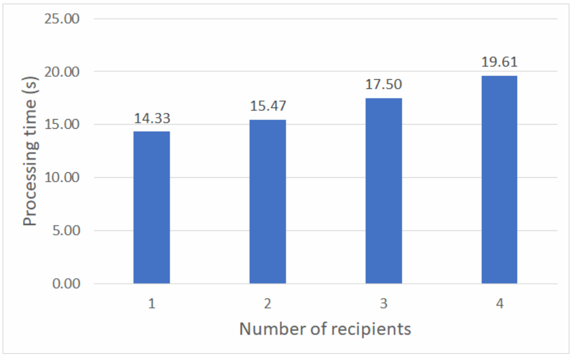

Figure 8. The bar chart of the transaction processing time based on different number of recipients

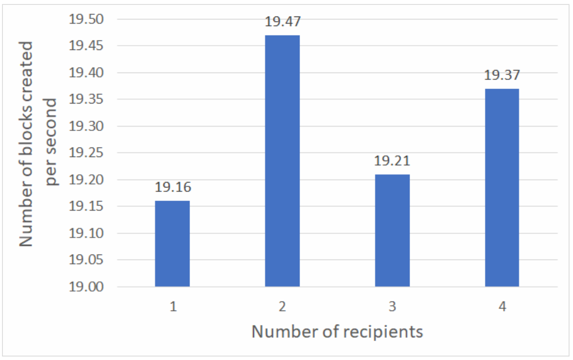

Figure 9. The bar chart of the number of created blocks per second based on different number of recipients

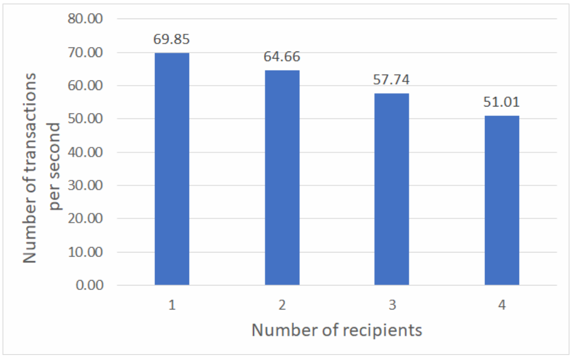

Figure 10. The bar chart of the total number of transactions per second based on different number of recipients

The process of exchange file uploading included the time spent on file uploading to the XAgent, computing time of the hash value, and time spent to save the file in the isolation area. To test the effect of file size on processing time, different file sizes are used: 100 KB, 500 KB, 1 MB, 5 MB, 10 MB, 30 MB, and 50 MB. Each file size was tested five times. The average processing time of file upload for each file size is shown in Figure 11. Notice that most of processing time is spent during the step of file uploading to the XAgent. Therefore, depending on the network bandwidth of the XAgent-installed node, the average processing time for file upload process might vary accordingly. Assume the network bandwidth of the XAgent-installed node is fixed, then the processing time of file upload is longer if the file size is larger in general.

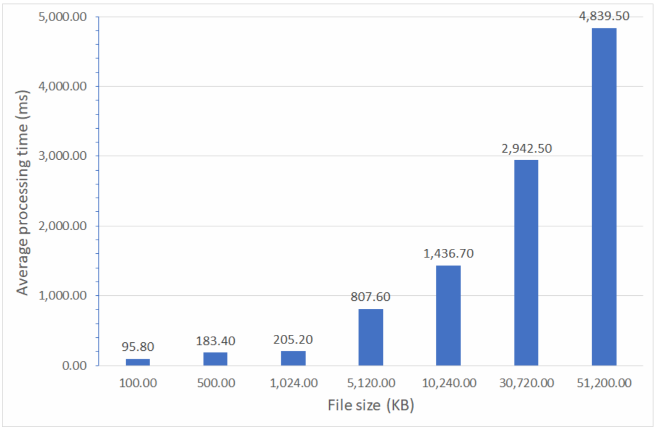

Figure 11. The average processing time for file upload process

The stress test for data exchange contract generation speed is set to different scenarios: 1, 5, 10, 30, 50, 100, 200, 400, 800, 1600, and 3200 recipients. The average results of each scenario after five executions are summarized in Figure 12.

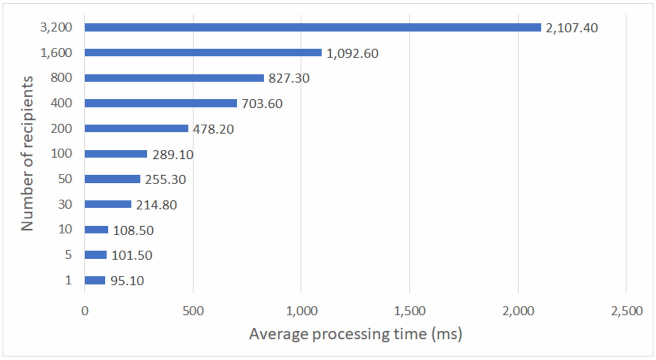

Figure 12. The average processing time for data exchange contract generation

For the outbound process test, timing began from the moment the contract was compiled. The test deployed the contract to the blockchain and retrieved the contract address, and then registered the correspondence between the exchange file and contract address to the FM (two registrations, text and attachment registrations each). Four test scenarios were established depending on the number of recipients, and the settings were similar to the stress tests. Figure 13 presents the test results.

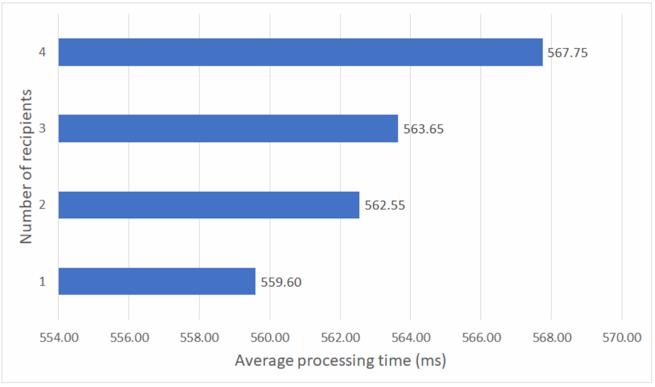

Figure 13. The average processing time of the data outbound process

For the inbound process test, timing began from when the exchange contract was received. The time was only counted upon receipt of the contract until identity verification and exchange file downloading, but did not include the time required to download and verify the file and source. Four test scenarios were established according to the number of recipients, and the settings were similar to the stress tests. Figure 14 presents the test results.

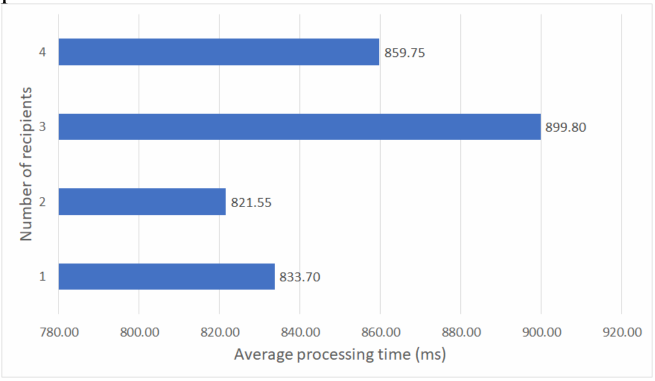

Figure 14. The average processing time of the data inbound process

Figure 11 presents the results of file upload processing. The results indicated that the larger the file size, the longer the processing time that was required. The exchange contract generation test produced the same result as shown in Figure 12, and more recipients required a longer compilation time. Finally, in the whole process of outbound and inbound test results as shown in Figure 13 and Figure14, no apparent difference was found. On average, the outbound and inbound operations could be completed in approximately 563 and 853 ms; that is, if the file upload and download times were not calculated, only 1.4 seconds were required to complete the outbound and inbound of the file (data).

5 Conclusion

Traditional EDI operation between organizations has typically been handled by a centralized system. Although this system can avoid the SPOF problem through a cluster concept, it faces high risks of exchange records being tampered with. In recent years, the emergence of blockchains has led to more research on decentralization. The use of distributed ledgers can prevent data from being monopolized by a particular unit, and also make it possible for both exchange parties to establish a trustable exchange relationship without the witness of a third party.

This study intended to improve the inter- and intra-organizational electronic data exchange system with consortium blockchain framework. Through resolving the shortcomings of the previous centralized framework using blockchain characteristics and Quorum's data confidentiality processing, the physical file (or data) no longer has to flow through the entire exchange network, thereby reducing the possibility of it being hacked and decrypted. Smart contracts can help to ensure that only the real recipient can retrieve the exchange files from the sender. The exchange framework can be divided into the data exchange of intra-organizational, inter-organizational, and mixed structures to meet various exchange requirements. The performance testing results on the system prototype have shown that at least 60 exchange contracts could be deployed per second, and up to 5,000,000 contracts could be deployed per day, which is sufficient for the data exchange requirements between general organizations.

Acknowledgments

The authors acknowledge the support from Taiwan Information Security Center (TWISC) and Ministry of Science and Technology, Taiwan, under the Grant Numbers MOST 108-2221-E-011-063, MOST 108-2221-E-011-065, and MOST 108-2218-E-011-021.

References

[1] National Institute of Standards and Technology, FIPS PUB 161-2, Electronic Data Interchange (EDI), https://web.archive.org/web/20080511043940/https://www.itl.nist.gov/fipspubs/fip161-2.htm, 1996.

[2] International Organization for Standardization, ISO 9735: 1988, Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) – Application Level Syntax Rules, https://www.iso.org/standard/17592.html, 1988.

[3] S. Scala, R. McGrath Jr., Advantages and Disadvantages of Electronic Data Interchange an Industry Perspective, *Information & Management*, Vol. 25, No. 2, pp. 85-91, August, 1993.

[4] K.-W. Lai, Suggestions for Improvement on Information Security of the Electronic Official Document Exchange System, *Archives Semiannual*, Vol. 13, No. 2, pp. 4-17, June, 2014.

[5] C.-W. Wu, H.-L. Shan, W.-C. Wang, D.-M. Shieh, M.-H. Chang, E-government Electronic Certification Services in Taiwan, *International Workshop for Asian Public Key Infrastructures*, Taipei, Taiwan, 2002, pp. 1-8.

[6] C.-M. Ou, H.-L. Shan, C.-T. Ho, Government PKI Deployment and Usage in Taiwan, *Information & Security: An International Journal*, Vol. 15, No. 1, pp. 39-54, 2004.

[7] K.-W. Lai, Research of the Technical Issues for Integration between the Official Document Exchange System and Official Document Management Systems in Government Agencies, *Archives Semiannual*, Vol. 16, No. 2, pp. 84-91, December, 2017.

[8] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain Challenges and Opportunities: A Survey, *International Journal of Web and Grid Services*, Vol. 14, No. 4, pp. 352-375, October, 2018.

[9] S. Nakamoto, *Bitcoin: A Peer-to-peer Electronic Cash System*, http://www.bitcoin.org/bitcoin.pdf, 2009.

[10] Ethereum, *Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform*, https://github.com/ethereum/wiki/wiki/White-Paper, 2018.

[11] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts, *The 37th IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 2016, pp. 839-858.

[12] J.P. Morgan Chase, *Quorum Whitepaper*, https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20White paper%20v0.1.pdf, 2016.

[13] J.P. Morgan Chase, *Quorum: A Permissioned Implementation of Ethereum Supporting Data Privacy*, https://github.com/jpmorganchase/quorum, 2018.

[14] A. Bakre, N. Patil, S. Gupta, Implementing Decentralized Digital Identity using Blockchain, *International Journal of Engineering Technology Science and Research*, Vol. 4, No. 10, pp. 379-385, October, 2017.

[15] Z. Diebold, *Self-sovereign Identity Using Smart Contracts on the Ethereum Blockchain*, Master Thesis, University of Dublin, Dublin, Ireland, 2017.

[16] J. Benet, *IPFS-Content Addressed, Versioned, P2P File System*, https://arxiv.org/abs/1407.3561, 2014.

[17] M.-Y. Jung, W.-S. Kim, S.-H. Chung, J.-W. Jang, A Blockchain-based ID/IP Mapping and User-friendly Fog Computing for Hyper-connected IoT Architecture, *Journal of Information Communication Technology and Digital Convergence*, Vol. 2, No. 2, pp. 12-19, December, 2017.

[18] C. Allen, A. Brock, V. Buterin, J. Callas, D. Dorje, C. Lundkvist, P. Kravchenko, J. Nelson, D. Reed, M. Sabadello, G. Slepak, N. Thorp, H. T. Wood, *Decentralized Public Key Infrastructure*, https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/dpki.pdf, 2015.

[19] J. Golosova, A. Romanovs, The Advantages and Disadvantages of the Blockchain Technology, *IEEE 6th Workshop on*

Advances in Information, Electronic and Electrical Engineering, Vilniaus Apskritis, Lithuania, 2018, pp. 1-6.

[20] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, V. Santamaria, To Blockchain or Not to Blockchain: That is the Question, *IT Professional*, Vol. 20, Issue 2, pp. 62-74, March/April, 2018.

[21] N. Baygin, M. Baygin, M. Karakose, Blockchain Technology: Applications, Benefits and Challenges, *The 1st International Informatics and Software Engineering Conference*, Ankara, Turkey, 2019, pp. 1-5.

[22] Microsoft, *NET Core*, https://docs.microsoft.com/zh-tw/dotnet/core, 2018.

[23] Nethereum, *Ethereum .NET Cross Platform Integration Library*, https://github.com/Nethereum/Nethereum, 2018.

[24] Ethereum, *Ethereum JavaScript API*, https://github.com/ethereum/web3.js, 2018.

[25] Decentralized Identity Foundation, *Universal Resolver Implementation and Drivers*, https://github.com/decentralized-identity/ universal-resolver, 2018.

Biographies

Nai-Wei Lo received his Ph.D. from State University of New York at Stony Brook, USA, in 1998. He is currently a Professor and the Chairman of Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include blockchain security, IoT security, and Web technology.

Sheng-Chiang Chen received his master degree from National Taiwan University of Science and Technology, Taiwan, in 2005. He is currently a doctoral candidate of School of Management in National Taiwan University of Science and Technology. His research interests include supply chain management, information security, and blockchain technology.

Shou-Chun Chang received his master degree in Information Management from National Taiwan University of Science and Technology in 2018. His research interests include blockchain, information security and Internet of Things.