Design Patterns for Blockchain-assisted Accountable Data Dissemination between IoT Devices and Edge Server

CHUN-AN LIN, Department of Computer Science, National Chengchi University
CHUN-FENG LIAO, Department of Computer Science and Program in Digital Content and Technologies, National Chengchi University
KUNG CHEN, Department of Management Information Systems and Department of Computer Science, National Chengchi University

There is an increasing number of software developers that want to take advantage of blockchain technology in their projects. Among various kinds of applications, IoT (Internet of Things) is recognized as one of the most promising domain to employ blockchain technology due to the highly distributed nature of blockchain. Therefore, many blockchain-driven IoT services (B-IoT) have been developed in recent years. Unfortunately, when running a blockchain, a lot of computational power and storage is used. Owning to the limitation of the power and hardware capacity, IoT devices are usually implemented with none or only part of the functionalities of a full blockchain endpoint, resulting in the advantages of the B-IoT not fully leveraged. With the increase in network bandwidth, advancements in hardware capability, and new blockchain endpoint protocol for low-capacity environments, it is now feasible to locate more functions of a blockchain endpoint on an IoT device. We present the empirical lessons of the authors from building several B-IoT systems in the last few years. We observe that there are several design issues and design decisions to be considered. In this paper, we report two patterns related to the design disseminating accountable data from IoT devices to the blockchain in the edge environments. To explain how these patterns work, we also introduce an "Intelligent Refrigerated Shipping Containers" scenario.

Categories and Subject Descriptors: **D.2.11 [Software Engineering]**: Software Architectures—*Patterns;*

General Terms: Design

Additional Key Words and Phrases: blockchain, IoT, Edge computing, Design pattern

ACM Reference Format:

C.A. Lin, C.F. Liao, K.Chen "Design Patterns for Blockchain Assisted Accountable Data Dissemination between the IoT Device and the Edge Server", 2019. Procs. Asian PLoP'20, March 4- 6, Taipei, Taiwan. 16 pages.

1. INTRODUCTION

Blockchain is the core underlying technology of digital cryptocurrency systems such as Bitcoin (Nakamoto 2008). A blockchain is a decentralized platform with an immutable, transparent and traceability ledger. The trustless peers in a blockchain network can reach consensus based on pre-determined algorithms (e.g., Proof-of-Work or Proof-of-Stake). Recent development of the blockchain platform, such as Ethereum(Buterin 2014), also brings about a new idea called Smart Contract, which is essentially a piece of executable programming logic used to verify or to settle the transactions among parties. Moreover, to retrieve and filter the state change of smart contracts from the blockchain platform efficiently, Ethereum provides a logging mechanism for the smart contract, known as Event. Due to the highly distributed and scalable nature, IoT (Internet of Things) is recognized as one of the most promising domain to employ the blockchain technology. It is reported that there are already 50 billion IoT devices interconnected over the world, and this number is increasing. The rapid growth of the number of connected things brings about new issues on scalability, security, and privacy (Evans 2011). Hence, blockchain can be used as a highly scalable peer-to-peer message exchanging and distributed transaction processing infrastructure. Besides, the blockchain can also provide a secure billing layer so that it is straightforward to build a peer-to-peer marketplace among things being interconnected by the blockchain

This work is partially supported by Ministry of Science and Technology, Taiwan, under grant 108-2221-E-004-002, 108-2221-E-004-003 and 108-2218-E-004-001.
Author's address: C.A. Lin, Dept. Computer Science, National Chengchi University, Taipei, Taiwan; email: 107753020@nccu.edu.tw; C.F. Liao, Dept. Computer Science, National Chengchi University, Taipei, Taiwan; email: cfliao@nccu.edu.tw ; K.Chen, Dept. MIS, National Chengchi University, Taipei, Taiwan; email: chenk@nccu.edu.tw

network (Christidis and Devetsikiotis 2016). As a result, many Blockchain-driven IoT services (B-IoT) have been proposed in recent years.

Nevertheless, owing to blockchain-related technologies still under active development and rapidly changing, many design issues are not mature. Therefore, the construction of a high-quality B-IoT system is difficult. When developing a B-IoT system, the strategy for deploying blockchain nodes differs according to the desired degree of decentralization. To join a blockchain network, an edge server (or an IoT device) has to arm with a node, referred to as a blockchain endpoint (e.g., Go-ethereum (Viktor Trón 2014) or Parity (Parity Technologies 2016)) in the sequel, which realizes the underlying consensus mechanism, synchronizes and verifies the block data, manages the smart contracts, and processes the transactions. However, a blockchain endpoint is burdened with computation (transaction processing and mining) and storage loads (block data). Unfortunately, computation and storage are the most limited and precious resources of typical IoT devices. Thus, the IoT device usually implements none or part of the functionalities of a blockchain endpoint, and delegates the remaining tasks to an adaptation node located on the edge or cloud server (Liao, Hung et al. 2019). Nevertheless, it is worth mentioning that with the advancements of technology in the recent year, a blockchain endpoint is more feasible to be located on an IoT device due to the following reasons: (1) with the maturity of 5G, network coverage and transmission speed increases significantly (Andrews, Buzzi et al. 2014); (2) with the advancement of wafer processing technology, the cost of high-performance computing hardware and storage for IoT data storage has been decreased significantly; (3) To allow the low-capacity environments(e.g., smartphones, embedded smart property environments) to maintain high-security assurance about the current state of blockchain network or verify the transaction data, a new type of lightweight blockchain endpoint called the Light client (Antonopoulos and Wood 2018) has been proposed. IoT devices serving the Light client is possible to directly participate in the blockchain network without having to synchronize the whole block data and participate in the mining process. (i.e., validating transactions and adding new blocks to the blockchain) In generally, a Light client has to rely on a trusted full node, namely, a fully functional blockchain endpoint) for synchronizing the block data and emitting the transactions. From software architecture's perspective, this design of B-IoT system belongs to the *Distributed Things* (Liao et al. 2019) style (see Figure 1).

More design issues appear when constructing a B-IoT system. For instance, because most of the consensus algorithm of the blockchain, like Proof-of-Work or Proof-of-Stake (Antonopoulos and Wood 2018), lead to considerable resource consumption, different endpoints deployment strategies or transmission methods between devices have a significant impact on non-functional qualities of B-IoT system (Sun, Hua et al. 2018). Therefore, developers need to spend a lot of time to handle these design issues in the early stages of blockchain technology. In response to the above challenges, it has been pointed out that a design pattern helps developers to solve the problems that are often encountered in the software design process in a more efficient and high-quality way. Moreover, it can also reduce development time and improve the quality of the system when faced with similar problems. Several blockchain patterns have been reported (Wöhrer and Zdun 2018) (Eberhardt and Tai 2017) (Xu, Pautasso et al. 2018). However, few of them concentrate on the blockchain-IoT integration issues. In the previous work (Liao, Hung et al. 2019), we have proposed various styles(including *Fully Centralized*, *Pseudo Distributed Things*, *Distributed Things* and *Fully Distributed*) for the B-IoT systems form a software architecture's perspective. As mentioned, among the four architectural styles, the *Distributed Things* style is probably the most viable as it strikes a balance between constrained resources and the coverage of blockchain endpoints. In *Distributed Things*, Edge is mainly responsible for collecting the sensing data obtained by IoT devices. On the other hand, by deploying a blockchain light client as an endpoint on a high-end IoT device, it can communicate directly with the edge server through the blockchain network. Compared with the less decentralized architecture style(i.e., *Fully Centralized* and *Pseudo Distributed Things*), in *Distributed Things*, the edge server can directly use the blockchain transmission mechanism to collect the data of IoT devices to reduce the risk of malicious third-party attacks.

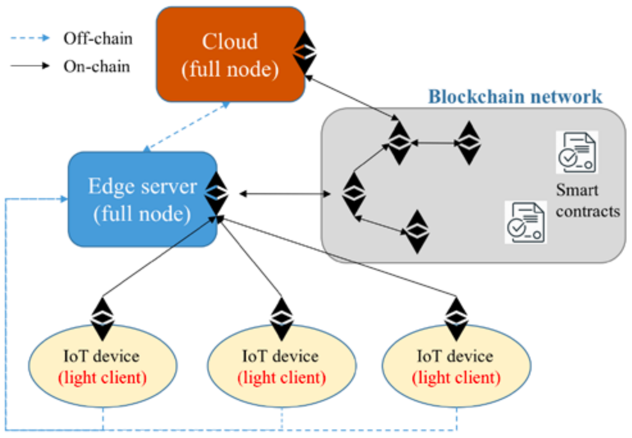

Figure 1: A B-IoT system with the Distributed Things style

At design level, there are a couple of issues when constructing a B-IoT system following the *Distributed Things* style. The objective of this paper is to present two patterns at the design level for such a style. Specifically, how the blockchain-assisted accountable data dissemination mechanism works between the IoT device and the edge server serving as blockchain endpoints. This paper presents two design patterns: *On-chain Edge-initiated Invocation (OEI)* and *OFF-chain Edge-initiated Invocation (OFEI).* In each pattern, we explore the problems and forces faced by the edge server when collecting accountable data from IoT devices in different contexts and consider not only the features of on-chain (dispatching and transmitting data using the blockchain) and off-chain (dispatching and transmitting data without blockchain) transmission method but also transmission data. Then, we propose a concrete solution to deal with the problems and to balance the forces. Patterns provided by this paper could be useful for B-IoT developers under specific contexts when designing B-IoT systems following the *Distributed Things* architectural style.

2. SCENARIO

In order to illustrate how design patterns provided in this paper works in the real world, we adopt the use case of "Intelligent Refrigerated Shipping Containers" (Dittmer, Veigt et al. 2012). The goods are kept in a ship hauling intelligent "reefers" (refrigerated containers) and their environmental factors are tracked to ensure the condition of goods during the shipping process. However, when shipping "high-priced goods", it is necessary to ensure the security of transmission data. The scenarios used in this paper are detailed below.

In a shipping process, there are two cooperative but trustless organizations, namely freight forwarder, and shipping company (see Figure 2). They need to ensure that the environmental factors (e.g., temperature and humidity) in the reefer where the "high-priced goods" are deposited are in compliance. While shipping, the goods are stored in a reefer with an IoT device which is equipped with an array of sensors. In the beginning, when the ship is offshore, due to the weak network signal, the freight forwarder (cloud) needs to authorize the crew to collect the sensing data in the reefers through the ship's server (edge server) and ensure the security of data. Then, the crew sends a request to an IoT device through the edge server once he/she wants to access the IoT device's service to collect sensor data. After receiving the callback data, Edge starts processing and then store it. Finally, when the ship is docked at the port, the freight forwarder needs to synchronize the sensing data detected by IoT devices through the edge server on the ship.

In the scenario, following the *Distributed Things* style, each component (i.e., the cloud and edge server, the IoT devices) is assumed to arm with a blockchain endpoint. The IoT device in the reefer is serving a light client. On the other hand, the cloud and edge server are serving full nodes. In practice, the light node serving on the IoT devices have to synchronize the block data from the full node serving on the edge server. Technically, all the components have to connect with the network to synchronize the data. If there are some network problems that lead to the blockchain endpoint serving on the component out of synchronizing, it is recommended to make a buffer (e.g., transaction queue) for the updating data or transactions (common solution in typical IoT system) locally until the synchronization is continuing. Besides, it is worth mentioning that the B-IoT system following the *Distributed Things* style is more robust than the typical IoT system. In the typical IoT system, while the network link between the cloud and edge server is disconnected, the synchronization would be failed. However, in the B-IoT system, all the components serve as peers. It means that it is possible for the edge server to synchronize data with the other edge servers rather than the cloud server. (B-IoT system has more than one network link while the typical IoT system just has a single link.)

In the following, we will use the scenario mentioned above to illustrate the patterns presented by this paper. We consider whether the pattern's context and problem are related to a specific case in the scenario, and apply the pattern's solution to solve the problem in the example. The forces base on the scenario that is considered to affect solutions of each pattern is listed below.

- **The privacy and immutability of the data, as well as the cost and scalability of the B-IoT system**, could be considerably affected when considering the transmission method between the Edge server and IoT devices.

- **The way of the edge server and IoT devices exchanging encryption keys** could be affected when considering using the different types of cryptography. (e.g., symmetric and asymmetric cryptography)

- **The authenticity of the system** could be affected considering different roles and components of the scenario. (e.g., crew, freight forwarder, edge server, IoT device) Generally speaking, the blockchain infrastructure (e.g., Ethereum) provides a public-key-cryptography-based account system. Thus, the account managed by the blockchain infrastructure can be used to represent the unique ownership of the individual participants in the system. In this way, each request is verified by the blockchain account to protect the smart contract from accessed by the illegal parties.

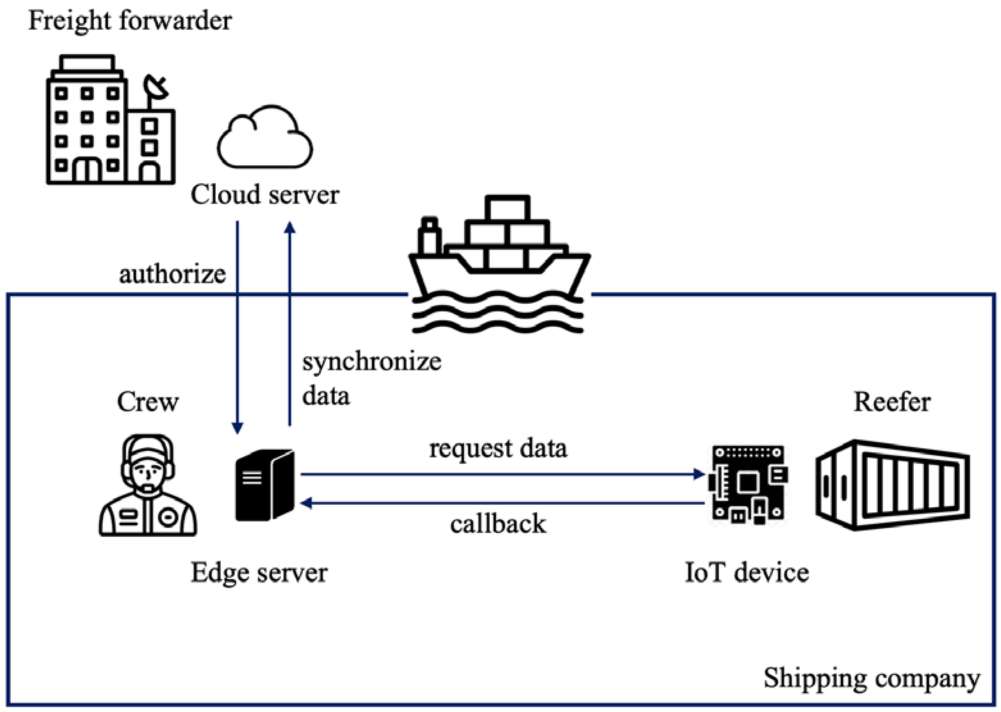

Figure 2: Example of data collection between the edge server and the IoT device

3. ON-CHAIN EDGE-INITIATED INVOCATION, OEI

3.1 Context

In a typical IoT system, when the edge server needs to collect sensing data, it needs to send a request to the IoT device. For instance, client, namely, the control program in the edge server, needs to utilize the API provided by the edge server in order to drive the IoT device to start collecting the data. Then, the IoT device returns the data via a pre-determined transferring mechanism such as MQTT or COAP. However, the data may be exposed to or tampered by a malicious third party sniffing the network links between the edge server and the IoT device, either during or after the transmission process. Therefore, how to transmit the accountable data in a secure way becomes an important concern.

3.2 Problem

In the B-IoT system, how to make sure the security of the transmission between the IoT device and the edge server during accountable data dissemination process?

3.2.1 Forces

- **Immutability.** Compared with transmitting the data off-chain, the on-chain transmission method can ensure that the data are stored in blockchain without tampering. However, due to the consensus algorithm of blockchain such as Proof-of-Work, storing data on-chain can lead to a considerable cost and low throughput.

- **Extensibility.** Typically, a smart contract serves as the logical shared buffer for transferring data among blockchain endpoints. In the B-IoT context, to initialize a data collecting session, the edge server first needs to send a (data) request to the target IoT device via a smart contract and then the IoT device responds to the request by transmitting its data via the same smart contract. In this case, the IoT device needs to know the reference of the smart contract in advance. Apparently, there exists a locational coupling (i.e., the reference of the smart contract) between each data request/response pair (the edge server and the IoT device). Whenever a new smart contract is created for a data transmission task, the IoT devices need to know and store the contract reference in advance, which is infeasible in practice. Obviously, the situation mentioned above leads to an inextensible system.

- **Authenticity.** To avoid malicious requests or attacks, it is necessary to validate the access request send by the edger server and make sure that only the specific IoT Device is allowed to update the related data. If there is no access control mechanism implemented in the smart contract, according to the transparency feature of blockchain, all the participants in the blockchain network can access the smart contract.

3.3 Solution

In order to ensure **integrity** and **immutability**, the on-chain transmission method is a great choice in order to leverage the advantage of the decentralization feature of the blockchain. In the blockchain, there is a lack of the peer-to-peer transfer mechanism. All data transmission on-chain needs to invoke the function of the smart contract by sending a transaction to the blockchain network. In practice, the data disseminated on-chain between endpoints can make good use of the Event mechanism in smart contracts (see Figure 3).

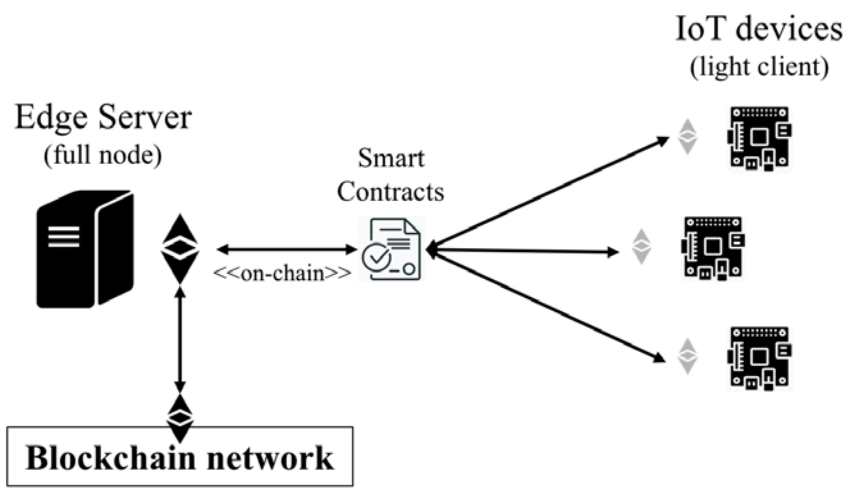

Figure 3: Structure of the *OEI*

3.3.1 Structure

In this pattern, the mechanism for registration, data storage and event of data requests need to be implemented as a method in smart contracts. First, to solve the issue of system extensibility, a centralized registration mechanism implemented in the smart contract, called RequestRegistry contract (see Figure 4), is needed (Logically, the RequestRegistry contract is a centralized design. However, it is realized using a smart contract. As a result, there is a copy of the RequestRegistry contract in each blockchain node. In this view, the RequestRegistry contract is physically decentralized.) In this way, instead of storing the reference of every new smart contract, the IoT devices only need to store the reference of one smart contract, namely, the RequestRegistry contract. The **extensibility** can thus be increased because now that the reference stored in the IoT devices doesn't need to be updated upon contract deployment. In addition, the RequestRegistry contract registers all access requests from other smart contracts (invoked by the edge server) and plays the role of the IoT devices' owner to protect the IoT data from illegal access by verifying the blockchain account on behalf of allowed individual, such as IoT devices and the Consumer contract (see Figure 4), to make sure the **authenticity.** The RequestRegistry contract can label all the access requests with sequence numbers to keep tracks of the ordering of all access requests. The RequestRegistry contract also takes charge of **recording all data** sent from the IoT devices. Each Consumer contract instance represents a data accessor acquiring data from an IoT device. For instance, in the example mentioned above, goods served by different (Consumer) smart contracts are allocated in the same reefer, but each reefer is equipped with only one IoT device. Besides, the Consumer contract also implements an access control mechanism (e.g., verifying the blockchain account) that makes sure the transaction is sent by the edge server.

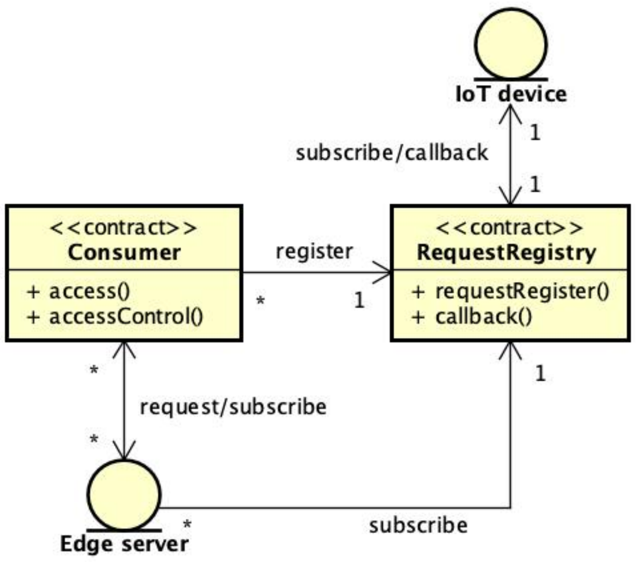

Figure 4: Class diagram of the *OEI*

3.3.2 Dynamics

The process of OEI can be divided into two stages. The first stage is **Registration** (see Figure 5). In this stage, data accessors (typically the edge server) express the intentions to access the sensor data by registering the "data request" to RequestRegistry contract. Technically, this process is realized by sending a blockchain transaction to invoke the requestRegister method of RequestRegistry contract. Firstly, the data accessor invokes the access method of the corresponding Consumer contract. Then, the Consumer contract verifies the identity (blockchain account) of the data accessor. If the identity is valid, the Consumer contract registers the request to the RequestRegistry contract (by invoking the requestRegister method). Meanwhile, RequestRegistry contract verifies the address of the Consumer contract to ensure the invocation is legal. Otherwise, the request is rejected and a rollback operation is performed. Next, the RequestRegistry contract emits an AccessRequest event. The AccessRequest event contains the desired data types and the identifier of the request. Finally, the event logs are stored in the blockchain.

The second stage is **Data Delivery and Recording** (see Figure 6). After the AccessRequest event being received, the corresponding IoT device starts to collect the desired data. The data are delivered by means of invoking the callback function of the RequestRegistry contract. Besides, the identifier of the request is also presented to RequestRegistry contract so that it can be stored as an index to the data delivery record. Finally, the requested data are sent to the data accessor by means of a contract event of the Consumer contract.

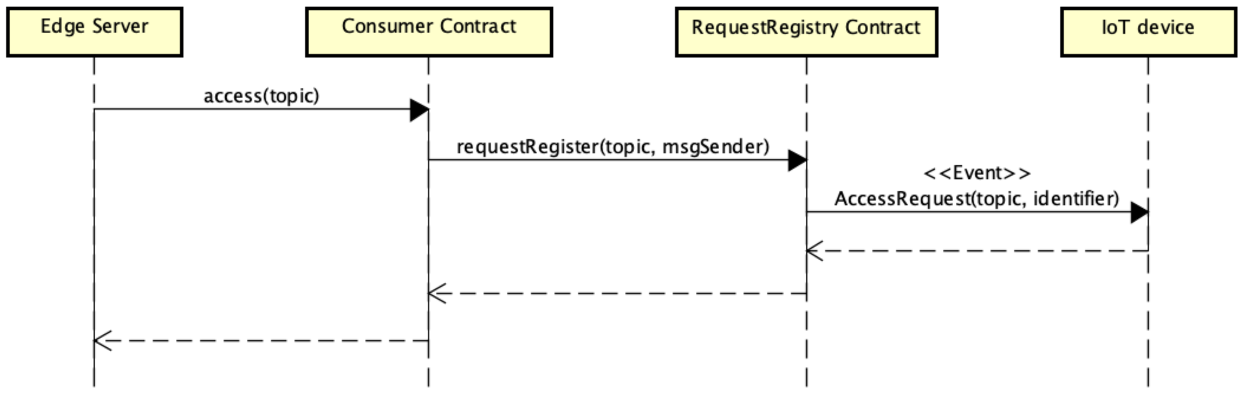

Figure 5: Sequence diagram of the Registration stage in *OEI*

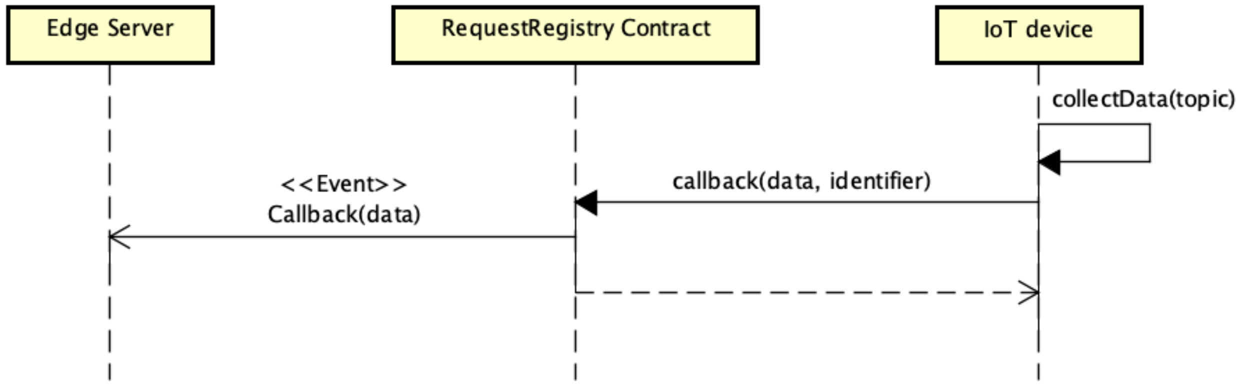

Figure 6: Sequence diagram of the data delivery and recording stage in *OEI*

3.4 Implementation

- **Deploy blockchain endpoints on the IoT device and edge server.** In this pattern, the two-way transmission is executed on-chain. First, blockchain endpoints must be served on both the edge server and the IoT device. Because the blockchain endpoint served by the IoT device is assumed as a Light client, it must synchronize the block data from the full node served by the edge server. If the edge server and the device are trustless, the edge server must be verified by the IoT device before starting to synchronize the block data.

- **Deploy the RequestRegistry contract (see Figure 7) and the Consumer contract (see Figure 8).** If the edge server and IoT devices cannot be trusted, the smart contract should be managed or deployed by a neutral third party. Besides, the edge server registers the access request and emits the AccessRequest event to drive the IoT device through the RequestRegistry contract after the edge server invokes the access function of the Consumer contract. Therefore, while deploying the Consumer contract, the reference (RequestRegistryAddress) of the deployed RequestRegistry contract must be registered to the contract.

- **Define access permission in the RequestRegistry and the Consumer contract.** In order to prevent malicious users from accessing smart contracts, such as DDoS attacks, it is necessary to validate the access request sent to the smart contract. In the blockchain network, each participant can own at least a key pair on behalf of his or her identity. Therefore, in a smart contract, the contract owner can pre-establish a granted-address array. In this pattern, the Consumer contract can be invoked by valid edge servers and the

RequestRegistry contract can be invoked by valid IoT devices.

- **The edge server must obtain the resource information explored by the IoT device in advance.** The edge server must confirm the access method of the resource provided by the IoT device to correctly send an access request to the specific resource. Moreover, the edge server also needs to know the information (e.g., resource topic) that needs to be included when sending an access request to prevent the IoT device from returning redundant data.

- **Determining the transmission method according to the features of the transmission data.** Although the data returned by the IoT device is transmitted on-chain based on the solution provided by this pattern, the raw data can still be transmitted off-chain while it is not appropriate to upload to the blockchain (e.g., huge size or confidential data). In practice, the hash value of the raw data can be returned to the smart contract by the IoT device on behalf of the raw data. Therefore, the edge server can make sure the integrity after receiving the raw data transmitted off-chain by comparing it with the hash value placed on the smart contract.

- **Subscribe the events from smart contracts in advance.** Before the data collection process starts, both the IoT device and the edge server have to subscribe to the events (AccessRequest and CallbackEvent) emitted by the smart contract to receive the event at runtime.

```
contract RequestRegistry {
    bytes32 identifier;
    mapping(bytes32 => bool) validateQueries;
    event AccessRequest(string topic, bytes32 identifier);
    event CallbackEvent(string data);

    function requestRegister(string memory _topic) public returns(bytes32){
            //implement the access control mechanism and provide an identifier for the request
       emit AccessRequest(_topic,identifier);
            return identifier;
    }

    function callback(string memory _data, bytes32 _identifier) public {
            //implement the access control that allow specific device to access
            delete validateQueries[_identifier];
            emit CallbackEvent(_data);
    }
}
```

Figure 7: Simple contract of the *RequestRegistry*

```
contract Consumer{
    address public requestRegistryAddress;

    constructor(address _requestRegistryAddress) public{
        requestRegistryAddress = _requestRegistryAddress;
    }

    //implement the access control function

    function access(string memory _topic) public returns(bytes32){
        //implement the access control mechanism
             RequestRegistry requestRegistry = RequestRegistry(requestRegistryAddress);
             bytes32 identifer = requestRegistry.requestRegister(_topic);
       return identifer;
    }
}
```

Figure 8: Simple contract of the *Consumer*

3.5 Example

In the process of goods shipping, because the shipping company must prove the integrity of the goods to the freight forwarder, the crew must request the photography service from the IoT device in reefer through the edge server during the shipping process. In this scenario, in addition to preserving each access request and validating the identity of the edge server, it is also necessary to confirm the integrity of the photos.

The above scenarios are exactly in line with the problem described in the context of *OEI*. However, owing to the large size of the photos, it is not suitable to transmit on-chain. Therefore, by hashing the photos and obtaining the hash value, the IoT device returns the hash value to the smart contract to leverage the immutability of the blockchain; on the other hand, the raw data of the photos are transmitted by means of a security channel off-chain. Finally, the shipping company or the freight forwarder can verify whether the photos have been tampered with during the transmission process by using the raw data of the photos received by the edge server and the hash values placed in the smart contract.

3.6 Known uses

- ***Electric Vehicle Battery Refueling*****(Sun, Hua et al. 2018).** The electric vehicle battery company verifies the process of battery exchange through the smart contract. In the study, both the edge server and IoT devices exchange relevant data directly through the blockchain endpoints.

- ***Chainlink*****(Chainlink Ltd SEZC 2019).** As a middleware for transmission and verification of data in the blockchain ecosystem, Chainlink's off-chain mechanism aggregates and validates the data requested by the customer, uploads it to the Chainlink's contract, and then disseminates to the customer's contract through the Chainlink's contract.

3.7 Consequences

3.7.1 Benefits

- **Integrity and immutability.** Through the transmission on-chain, the blockchain mechanism can effectively ensure the integrity and immutability during and after the transmission process.

- **Transparency.** Due to the features of the blockchain, the data transmitted on-chain is stored in a transparent way. All the blockchain participants can track the logged historical data through the blockchain endpoint.

3.7.2 Drawbacks

- **Cost.** If the public blockchain is used, it costs real money while storing data on the blockchain. When the transmission frequency is high and the size of data is large, the cost increases. On the other hand, because of the consensus mechanism of the blockchain, such as Proof-of-Work, the block data are replicated into multiple pieces and thus the cost of storage also increases.

- **Privacy.** All the participants of the blockchain network can access the historical transaction information. Therefore, the data transmitted on-chain needs to be secured by encryption mechanisms.

- **Scalability.** Due to most of the consensus algorithm (e.g., Proof-of-Work) of the blockchain, the transaction per second (TPS) are not significantly improved with the increase of miners. It means the throughput compared to the off-chain transmission are significantly lower.

3.8 Related patterns

- ***Oracle*(Antonopoulos and Wood 2018).** The off-chain Oracle mechanism subscribes to the event from the smart contract. When receiving a specific event, the Oracle starts to collect the relevant information and then returns it back to the smart contract.

- ***Ownership*(Wöhrer and Zdun 2018).** Authorizing the smart contract's owner to modify the data in a smart contract by setting the ownership.

- ***Adapter*(Gamma 1995).** Converting an object with the incompatible interface into an expected one to collaborate with.

4. OFF-CHAIN EDGE-INITIATED INVOCATION, OFEI

4.1 Context

In some cases of data collection between the edge server and the IoT device, the frequency of the edge server sending the access request is high or the data returned by the IoT device is low-value(trivial) and private. If the data is transmitted on-chain, the efficiency is reduced and the cost considerably increases because more and more transactions are taking place in the blockchain.

4.2 Problem

In the B-IoT system, how to disseminate accountable data between the edge server and the IoT device in a cost-effective and secure way?

4.2.1 Forces

- **Cost.** Owing to the features of the blockchain, if the data is collected through the blockchain (such as the OEI), it gives rise to a considerable cost, including hardware and energy resource consumption as well as blockchain verification fee. However, the data transmitted through the blockchain can take full advantage of the blockchain such as immutability and traceability.

- **Security and privacy.** In the process of private data transmission, it is important that the data must be transmitted in a secure way (e.g., secure transmission channel) and should not leak to third parties. However, implementing a secure transmission channel will increase the complexity of B-IoT system

- **The exchange of encryption keys.** If the developer is considering transmitting the accountable data by means of the off-chain method, the secure transmission channel is needed. Therefore, before the data collection process, both sides of the parties could exchange the keys which are used to encrypt the accountable data. There are two general cryptography methods. The asymmetric cryptography is suitable for sharing a key in the public, but it is leak of scalability. In contrast, the symmetric cryptography is the opposite to the asymmetric cryptography.

4.3 Solution

When starting the data collection process, the edge server should send the access request to the IoT device through the smart contract to validate the edge server and also leverage the immutability of the blockchain; On the other hand, due to the considerable cost, it is not appropriate to transmit low-value or trivial data by means of the on-chain transmission method. On the contrary, utilizing the secure transmission method off-chain can not only help to improve the scalability but also reduce the **cost** of data transmission (see Figure 9).

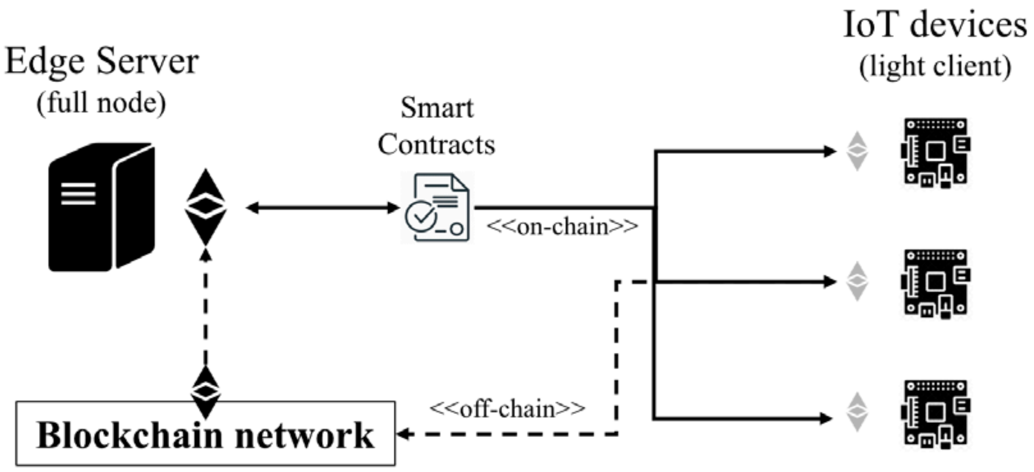

Figure 9: Structure of the *OFEI*

4.3.1 Structure

In order to leverage the advantages of the decentralization of the blockchain, this pattern validates and manages the access requests through the smart contract (see Figure 10). The access requests sent by the specific blockchain's user account or contract address are allowed to access the service of the IoT device. Consequently, the service of the IoT device can be protected from being accessed by malicious requests through the unknown blockchain endpoint.

To ensure the **security and privacy** of the accountable data dissemination, the use of the peer-to-peer secure network link is needed. On the other hand, to establish a secure off-chain channel, the edge server should inform the IoT device about which transmission method and format are used. It is important to note that the IoT device has to make sure that only the specific edge server is allowed to obtain the (plaintext) information. Taking the transparency of blockchain and the above reason into account, the use of **Asymmetric Cryptography** as the encryption method of the off-chain transmission is an effective solution.

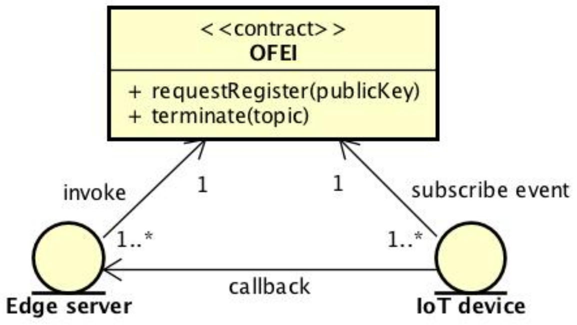

Figure 10: Class diagram of the *OFEI*

4.3.2 Dynamics

In the data collection process (see Figure 11), the edge server sends an access request which includes the public key representing the edge server to the IoT device through the smart contract. The smart contract first checks the identity of the edge server by validating the blockchain account and then emits an AccessRequest event. When receiving the event emitted from the smart contract, the IoT device drives the process to collect the data requested by the edge server and establishes the off-chain secure channel with the edge server. At last, the IoT device returns the accountable data encrypted by the received public key to the edge server constantly through the secure channel until the edge server invokes the terminate function of the smart contract.

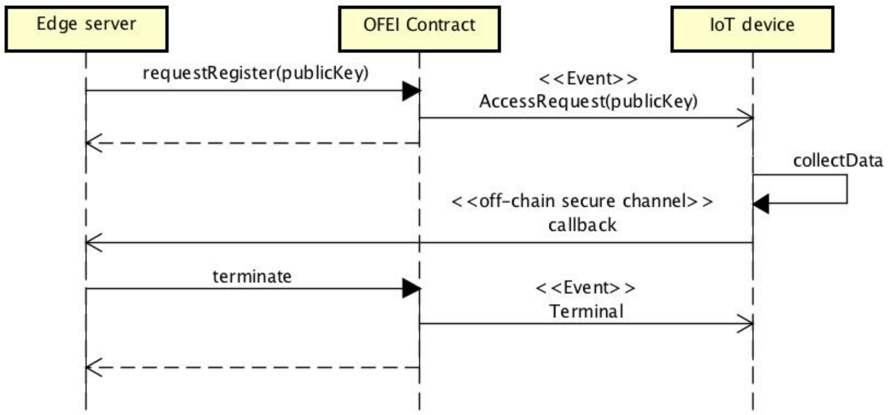

Figure 11: Sequence diagram of the *OFEI*

4.3.3 Implementation

- **Determine a secure transmission method between the IoT device and the edge server.** Before implementing and deploying smart contracts, the developer has to choose an off-chain transmission method that can ensure the privacy and security of the data. In this case, the P2P Messaging protocols(e.g., Ethereum Whisper (Antonopoulos and Wood 2018), Hyperledger fabric Gossip Protocol (Androulaki, Barger et al. 2018)) provided by the blockchain ecosystem is recommended.

- **Deploy the smart contract.** In the smart contract (see Figure 12), an access control mechanism must be implemented. Whenever the edge server invokes requestRegister function, the smart contract should register and validate the request and then emit the AccessRequest event to drive the IoT device. Moreover, the developer can implement a public key exchange mechanism for the transmission off-chain. In addition, the terminate function is responsible for terminating the data collection process.

- **Establish the off-chain secure channel between the edge server and the IoT device.** In order to disseminate the accountable data to the edge server through the off-chain transmission method, it is necessary for the IoT device to establish a secure channel which is chosen by the developer in advance.

```
contract OFEI{

    event AccessRequest(uint indexed deviceID,string topic,string publicKey);
    event Terminal(uint deviceID,string topic);

    function requestRegister(uint _deviceID,string memory _topic,string memory
       _publicKey) public{
        //implement the access control and  the access request registering mechanism
        emit AccessRequest(_deviceID,_topic, _publicKey);
    }

    function terminate(uint _deviceID,string memory _topic) public{
        //implement the access control and the registred request deleting mechanism
        emit Terminal(_deviceID, _topic);
    }
}
```

Figure 12: Simple contract of the *OFEI*

4.4 Example

In the shipping process, in order to monitor the sensing data (e.g., temperature, humidity, navigation data, etc.) promptly, the IoT device in the reefer should return the sensing data constantly. As a result of the high-frequency dissemination of the sensing data, the cost of the transmission is a significant increase while the on-chain transmission method is used. Consequently, the off-chain secure channel is an appropriate choice to substitute for the on-chain transmission.

4.5 Known uses

- ***Status*(Status Research & Development GmbH 2017).** Through the Whisper Protocol provided by the Ethereum ecosystem, a private peer-to-peer message channel is established between two blockchain endpoints to transmit messages off-chain mutually.

- ***Raiden network*(brainbot labs Est 2017).** Raiden network is an off-chain scaling solution underlying the Ethereum blockchain. The endpoints establish payment channels that enable transferring value off-chain to improve the scalability issue of the blockchain.

4.6 Consequences

4.6.1 Benefits

- **Privacy and security of the transmission.** As disseminated through the off-chain secure channel, the transmission will not leak to the network. Therefore, the accountable data returned by the IoT device can avoid being intercepted by malicious third parties.

- **Cost.** The data transmitted through the off-chain secure channel can reduce the transaction cost on the blockchain.

- **Scalability.** Since the transmission off-chain does not need to be verified by the blockchain consensus algorithm, the throughput of the data transmission is significantly increased.

4.6.2 Drawbacks

- **Authenticity.** In this pattern, the IoT device returns the data through off-chain transmission method.

According to the above reason, the edge server cannot effectively identify the IoT device. To solve the above problem, the off-chain transmission method can be utilized with the digital signature or another authentication mechanism.

- **Immutability.** The accountable data disseminated off-chain cannot ensure that the history of the data collection process is stored, not tampered or lost without using the on-chain transmission method.

- **Traceability.** Because the data cannot be preserved in the blockchain through the off-chain transmission method, participants of the blockchain are not feasible to track the history of transmitted data.

4.7 Related patterns

- ***Off-Chain Signatures*(Eberhardt and Tai 2017).** Before determining the final trade result and sending the result to the blockchain, the communication between the two parties is achieved by the establishment of the private message channel off-chain.

- ***Off-Chain Data Storage*(Xu, Pautasso et al. 2018).** The smart object transmits the data which is infeasible to upload to the blockchain through the off-chain transmission method. On the other hand, the smart object places the reference information (e.g., hash value) on-chain to verify the integrity of the raw data transmitted off-chain.

5. CONCLUSION

In order to build a high-quality B-IoT system, developers must carefully evaluate the design issues that were discussed and make the right decision. This paper presents two design patterns that focus on the context of the blockchain-assisted accountable data dissemination between the edge server and the IoT devices. In these patterns, we consider the forces such as security, scalability, privacy, and cost. Although these patterns are helpful for developers to justify the design decisions of a B-IoT system in a given context, it is important to mention that there are still other constraints and optimization issues in such systems. We expect the limitation of constructing the B-IoT system is alleviated with the development of blockchain technology. In the future, we plan to discuss the B-IoT system when integrating with the cloud server. Also, we will continue to identify and evaluate more patterns of interactions between the B-IoT components.

REFERENCES

Andrews, J. G., S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong and J. C. Zhang (2014). "What will 5G be?" IEEE Journal on selected areas in communications **32**(6): 1065-1082.

Androulaki, E., A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman and Y. Manevich (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. Proceedings of the Thirteenth EuroSys Conference, ACM.

Antonopoulos, A. M. and G. Wood (2018). Mastering ethereum: building smart contracts and dapps, O'Reilly Media.

brainbot labs Est. (2017). "Raiden network." from https://raiden.network/.

Buterin, V. (2014). "A next-generation smart contract and decentralized application platform." white paper **3**: 37.

Chainlink Ltd SEZC. (2019). "Chainlink." from https://chain.link/.

Christidis, K. and M. Devetsikiotis (2016). "Blockchains and smart contracts for the internet of things." Ieee Access **4**: 2292-2303.

Dittmer, P., M. Veigt, B. Scholz-Reiter, N. Heidmann and S. Paul (2012). The intelligent container as a part of the Internet of Things. 2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), IEEE.

Eberhardt, J. and S. Tai (2017). On or off the blockchain? Insights on off-chaining computation and data. European Conference on Service-Oriented and Cloud Computing, Springer.

Evans, D. (2011). "The internet of things: How the next evolution of the internet is changing everything." CISCO white paper **1**(2011): 1-11.
Gamma, E. (1995). Design patterns: elements of reusable object-oriented software, Pearson Education India.
Liao, C.-F., C.-C. Hung and K. Chen (2019). Blockchain and the Internet of Things: A Software Architecture Perspective. Business Transformation through Blockchain, Springer**:** 53-75.
Nakamoto, S. (2008). "Bitcoin: A peer-to-peer electronic cash system."
Parity Technologies. (2016). "Parity." from https://www.parity.io/.
Status Research & Development GmbH. (2017). "Status." from https://status.im/.
Sun, H., S. Hua, E. Zhou, B. Pi, J. Sun and K. Yamashita (2018). Using Ethereum Blockchain in Internet of Things: A Solution for Electric Vehicle Battery Refueling. International Conference on Blockchain, Springer.
Viktor Trón, F. L. (2014). "Go-ethereum." from https://github.com/ethereum/go-ethereum.
Wöhrer, M. and U. Zdun (2018). Design patterns for smart contracts in the ethereum ecosystem. 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE.
Xu, X., C. Pautasso, L. Zhu, Q. Lu and I. Weber (2018). A Pattern Collection for Blockchain-based Applications. Proceedings of the 23rd European Conference on Pattern Languages of Programs. Irsee, Germany, ACM**:** 1-20.