Designing blockchain-based applications a case study for imported product traceability

Article *in* Future Generation Computer Systems · October 2018

DOI: 10.1016/j.future.2018.10.010

Xiwei Xu
The Commonwealth Scientific and Industrial Research Organisation
95 PUBLICATIONS **1,953** CITATIONS
SEE PROFILE

Qinghua Lu
The Commonwealth Scientific and Industrial Research Organisation
101 PUBLICATIONS **927** CITATIONS
SEE PROFILE

Yue Liu
UNSW Sydney
13 PUBLICATIONS **95** CITATIONS
SEE PROFILE

Liming Zhu
The Commonwealth Scientific and Industrial Research Organisation
191 PUBLICATIONS **3,543** CITATIONS
SEE PROFILE

Future Generation Computer Systems 92 (2019) 399–406

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Designing blockchain-based applications a case study for imported product traceability

Xiwei Xu [a,b], Qinghua Lu [a,b,*], Yue Liu [c], Liming Zhu [a,b], Haonan Yao [c], Athanasios V. Vasilakos [d]

[a] *Data61, CSIRO, Sydney, Australia*
[b] *School of Computer Science and Engineering, UNSW, Sydney, Australia*
[c] *College of Computer and Communication Engineering, China University of Petroleum (East China), Qingdao, China*
[d] *Lulea University of Technology, Sweden*

ARTICLE INFO

Article history:
Received 12 June 2018
Received in revised form 23 August 2018
Accepted 5 October 2018
Available online 16 October 2018

Keywords:
Blockchain
Smart contract
Adaptability
Software architecture

ABSTRACT

Blockchain technology enables decentralization as new forms of distributed software architectures, where components can reach agreements on the shared system states without trusting on a central integration point. Since blockchain is an emerging technology which is still at an early stage of development, there is limited experience on applying blockchain to real-world software applications. We applied blockchain application design approaches proposed in software architecture community in a real-world project called originChain, which is a blockchain-based traceability system that restructures the current system by replacing the central database with blockchain. In this paper, we share our experience of building originChain. By using blockchain and designing towards security, originChain provides transparent tamper-proof traceability data with high availability and enables automated regulatory-compliance checking and adaptation in product traceability scenarios. We also demonstrate both qualitative and quantitative analysis of the software architecture of originChain. Based on our experience and analysis, we found that the structural design of smart contracts has large impact on the quality of the system.

1. Introduction

Blockchain technology provides a distributed shared data store and a computational infrastructure. As a data structure, blockchain only allows inserting data without updating or deleting any existing data on the blockchain to prevent tampering and revision. Blockchain enables decentralization as new forms of distributed software architectures [1], where components can reach agreements on the historical shared states for decentralized and transactional data sharing across a large network of participants without relying on a central integration point.

Many banks are involved in trials of the blockchain technology, including through the global R3 consortium[1] which is applying blockchain to trade finance and cross-border payments. Financial transactions are the first, but not the only use case being investigated for blockchain. A blockchain implements a distributed ledger, which can verify and store any kind of transactions for general purpose [2]. Many startups, enterprises, and governments [3,4] are currently exploring blockchain applications in areas as diverse as supply chain, electronic health records, voting, energy supply, ownership management, identity management, and protecting critical civil infrastructure. Blockchain has some unique properties, including immutability, non-repudiation, data integrity, transparency, and equal rights, as well as limitations, for example, privacy and scalability. However, there is a lack of a systematic and holistic view on applying blockchain to design applications for general purposes. To better leverage the unique properties of blockchain, an architectural formalism of blockchain-based applications is needed.

Previous work [1] from software architecture community has characterized blockchain from a software architecture perspective as a software connector [5], providing a shared infrastructure to store data and execute programs (known as smart contracts). However, due to the properties and limitations of blockchain, the technology does not fit all the use cases. First, blockchain has limited storage capability since it contains a full history of all the transactions across all participants of the blockchain network. Second, blockchain applications might have sensitive data, but the information on blockchain is designed to be accessible to all the participants. Third, since all the smart contracts can be called by all the blockchain participants by default, a permission-less function might be triggered by unauthorized users accidentally, which becomes a vulnerability of blockchain-based applications.

[*] Corresponding author at: Data61, CSIRO, Sydney, Australia.
E-mail addresses: xiwei.xu@data61.csiro.au (X. Xu), qinghua.lu@data61.csiro.au (Q. Lu).

[1] http://www.r3cev.com/.

https://doi.org/10.1016/j.future.2018.10.010
0167-739X/© 2018 Elsevier B.V. All rights reserved.

Our motivation of this work is to apply the software design approaches for blockchain-based applications [1,6] into a real world project, originChain, and illustrate how originChain is designed and implemented to fulfill the requirements. We share our experience of building originChain, which restructures a traceability company's current system by replacing the central database with blockchain. The contribution of this work is the system design of originChain in terms of a set of design decisions we made during implementing the system, and the rationale for our decisions. originChain uses blockchain as a software component in a system, which interacts with other components and databases within the system. By using blockchain and a set of design decisions towards adaptability, originChain provides transparent tamper-proof traceability data with high availability, and enables automated regulatory-compliance checking and adaptation in product traceability scenarios. We have briefly discussed the architecture of orginChain and some lessons learned in [7], this paper presents a more comprehensive architecture of originChain with more detailed design and lessons learned. We also demonstrate qualitative analysis that shows the structural design of smart contract can affect the adaptability of the system, and quantitative analysis that shows writing on blockchain is longer than central database while reading from blockchain is efficient.

The remainder of the paper is organized as follows. Section 2 introduces background of blockchain and blockchain-based applications and related work about design with blockchain. Section 3 discusses the suitability of using blockchain in the context of product traceability. The originChain system is presented in Section 4. Section 5 analyses the architecture of originChain. Section 6 summarizes the lessons learned from this study. Section 7 concludes the paper and outlines the future work.

2. Design with blockchain

2.1. Blockchain and smart contract

Blockchains that provide a public ledger to store cryptographically-signed financial transactions, are regarded as the first generation of blockchain, like Bitcoin [8]. Such blockchains have very limited capability to support programmable transactions and only allow tiny pieces of data being embedded in the transactions to serve other purposes, for example, representing digital assets or physical assets The second generation of blockchains provide a general-purpose programmable infrastructure, and a public ledger that stores the computational results generated from the programmable infrastructure.

Smart contracts [9] are programs deployed and run on blockchain. Smart contracts can express triggers, conditions and business logic [10] to enable transactions with more complex programmable conditions. The signature of transaction authorizes not only the money transfer, but also the data payload of a transaction to create a smart contract or execute a function defined in smart contract.

2.2. Blockchain applications

Globally, financial services organizations, many governments, enterprises and startups are exploring the applicability of blockchain in their domains. Application examples include but are not limited to digital currency, international payments, securities registration and settlements from financial sector, registries, identity and taxation as government services, and Internet of Things (IoT) storage, compute and management and supply chain from industry. Supply chains are believed as a highly promising area for the application of blockchain [3,4].

Various academic work has applied blockchain as a trusted and secure infrastructure into different domains, For example, a blockchain-based personal data management system is built to ensure the users own and control their data in a decentralized way [11]. A public key infrastructure is built on a blockchain-based cryptocurrency, called Namecoin [12]. Blockchain is further used to build a public key infrastructure (PKI) enhancement that offers automatic responses to the misbehavior from certificate authority (CA) and incentives for those who detect misbehavior [13]. Blockchain has been also used to provide data provenance information for cloud-based data analytics [14].

2.3. Design of blockchain-based applications

In software architecture community, blockchain has been characterized as an architecture component, which can be used as data storage with computational capability [1]. Trade-off analysis of using blockchain against conventional databases and different blockchain configurations towards non-functional properties are discussed in [15].

There are some work that summarize the reusable solutions in blockchain-based applications as design patterns. Bartoletti et al. conducts an empirical study on smart contracts supported by different blockchain platforms, and identifies nine common programming patterns in Solidity-based smart contract [16]. Zhang et al. applies four existing object-oriented software patterns to smart contract programming in the context of a blockchain-based health care application [17]. Eberhardt and Tai proposes four patterns for blockchain-based applications, mainly focusing on what data and computation should be on-chain and what should be off-chain [18].

3. Product traceability using blockchain

3.1. Product traceability and traceability system

In supply chain, product traceability is the connection of all the processes involved in generating and distributing goods, from raw material to completed products and to consumers in the end. Product suppliers and retailers usually use government-certified independent traceability companies to inspect the products throughout the supply chain. If everything satisfies the requirements and fulfill government standards, the company issues inspection certificates that verify the quality and originality of products. A traceability system is employed to expose relevant information (e.g., originality, ingredient and locations) and issue certificates. The traceability system enables the users to track products during production and distribution. In this context, security of the system is important for accountability and forensic information. The traceability system normally stores information in databases controlled by the company. Such a centralized data storage runs the risk of being tampered by the company and becomes a potential single point of failure from system perspective.

3.2. Suitability of applying blockchain in traceability system

Supply chain is a promising area for applying blockchain technology [4]. According to Deloitte survey, 42% of the companies in consumer goods and manufacturing planed to spend at least $5 million on the blockchain technology in 2017 [19].

According to the evaluation framework [6], traceability system is a multi-party system that spans across different participants such as product suppliers, traceability companies, and other service providers, like labs for quality testing. The operations on the system need synchronization among participated organizations. Data transparency and immutability is desired because the consumers want to check the originality and authenticity of the products. Due to the low performance requirement of a typical traceability system discussed in Section 4.1 (number of suppliers and products etc.), the performance issues of blockchain can be neglected in this use case. So that a traceability system can benefit from digital nature of blockchain and is not affected by its current limitations.

4. originChain

4.1. Background of originChain

We have an industrial partner, which is an independent third party traceability company. Their traceability system provides information for products imported from overseas to China. The system has been integrated with a large e-commerce company in China and several public service agencies. So far, their traceability services have been used by hundreds of product suppliers and retailers to manage their products, and millions of product consumers to access the traceability information. Each of the product suppliers has about 20 products in the system on average. The granularity of the traceability information is rather large as it corresponds to the packages of products rather than individual product.

4.2. Methodology

In this case study, we first extracted a traceability process with adaptable scenarios from the current traceability system of our partner. Based on the process, we restructured their current system by introducing blockchain to work together with a conventional database and redesigned the architecture to support better adaptability. After restructure, originChain can provide transparent tamper-proof traceability information, enhance availability of the data and automate regulatory compliance checking. We have implemented originChain and tested it under realistic conditions using real data of the system users and products.

4.3. Traceability process of originChain

An extracted simplified traceability process is illustrated in Fig. 1 using BPMN (Business Process Model Notation[2]). The traceability company is the organization that coordinates multiple service providers to conduct services and issues certificates according the information provided by the service providers.

The process starts with a product supplier lodging an enterprise registration application to the traceability company. The administrator checks the paper work (e.g. official enterprise license) associated with the application and signs a long-term service agreement with the valid supplier.

Every batch of products from the product supplier triggers the application of traceability services. The supplier needs to submit an application for each batch with related documents including trading contracts, invoices and order forms. The application is validated based on this paper work. If the application is valid, the traceability company and the supplier sign a traceability service agreement, which defines what services are provided (factory inspection and/or sample testing and/or freight yard examination) to a specific batch, additional conditions and payments.

After the batch-specific service agreement is signed, a freight yard examiner is assigned to examine the products on freight yard and supervise on-site loading. During the on-site loading supervision, the examiner attaches lead seals to the containers with the products if the loading process meets requirements defined in the agreement. If sample testing is covered by the agreement, a product sample is sent to a third party inspection lab for sample testing. Similarly, if factory inspection is covered by the agreement, a factory examiner is assigned to inspect if manufacturing process meets required qualification standards. If the batch of products passes all the inspections and testing, the traceability company issues the supplier a *Traceability Certificate of Commodity* that certify the quality of the batch.

Adaptability Requirement. During this process, the sequence of traceability services can be dynamic due to the customization of inspection process. First, some of the services can be outsourced to other companies/organizations. For example, the labs to do sample testing are bound dynamically based on the availability of the labs and time of recent testing, and the characteristics of the products and the requested test. Second, suppliers and retailers can request a specific combination of services based on their need and budget. Besides, the release of new traceability related regulation could affect the way to conduct certain traceability services. For example, the Food Safety Law of China, which took effect in October 2015, set out new requirements for formulating national food safety standards and food safety traceability systems (what information should be provided at when). Thus, adaptability is one of the main non-functional requirements of traceability systems.

4.4. Design of originChain

4.4.1. Overall architecture

The restructured architecture of originChain with is illustrated in Fig. 2, which consists of UI layer, management layer, data layer (off-chain), and blockchain layer (on-chain). Blockchain is used as a software connector [1] that contains both data and business logic.

There are three types of users in originChain, including service users, traceability providers and blockchain administrators. The service users can be product suppliers, retailers or consumers. Product suppliers use *Product & Enterprise Management* module to manage their enterprise information and products. Product retailers and consumers check the quality and origin information of the products through the front-end of originChain. Suppliers apply for traceability services through the system and aim to receive certificates as evidences of compliance with traceability regulation. Such certificates also show the quality and origin of their products to retailers/consumers.

The traceability company offer various traceability services including factory examination, sample testing, product checking, on-site loading supervision or sealing. The traceability company uses *Sample Test Management* module to manage the results of the sample testing. and uses *Traceability Management* module to manage traceability information, certificates and on-site photos.

Blockchain administrators uses *Smart Contract Management* module to generate and deploy smart contracts, uses *Permission Control* module to control permission of smart contracts. and uses *Blockchain Management* module to manage the settings of blockchain network.

4.4.2. On-chain vs. off-chain

When designing applications on blockchain, the first and most important consideration is what data should be put on-chain. (Application design decision 1 in [1]). Blockchain contains a full copy of all the historical transactions that have ever occurred in the blockchain network. Such data remains on the blockchain permanently. The ever-growing size of blockchain and full replication is a challenge to store "non-small" data on blockchain.

In originChain, only the sensitive and small data is stored on-chain, including the hashes of certificate files and photos of on-site freight yard, other traceability information and permission control information. Traceability certificates and photos are critical to the end users so that storing the corresponding hashes on blockchain

[2] http://www.bpmn.org/.

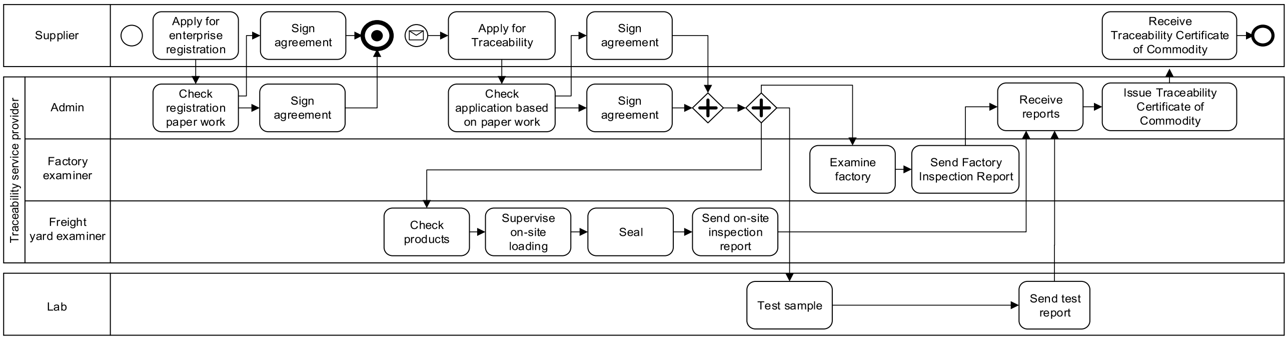
Supplier
Apply for enterprise registration
Sign agreement
Apply for Traceability
Sign agreement
Receive Traceability Certificate of Commodity
Traceability service provider
Admin
Check registration paper work
Sign agreement
Check application based on paper work
Sign agreement
Receive reports
Issue Traceability Certificate of Commodity
Factory examiner
Examine factory
Send Factory Inspection Report
Freight yard examiner
Check products
Supervise on-site loading
Seal
Send on-site inspection report
Lab
Test sample
Send test report

Fig. 1. Simplified Traceability Process Illustrated in BPMN.

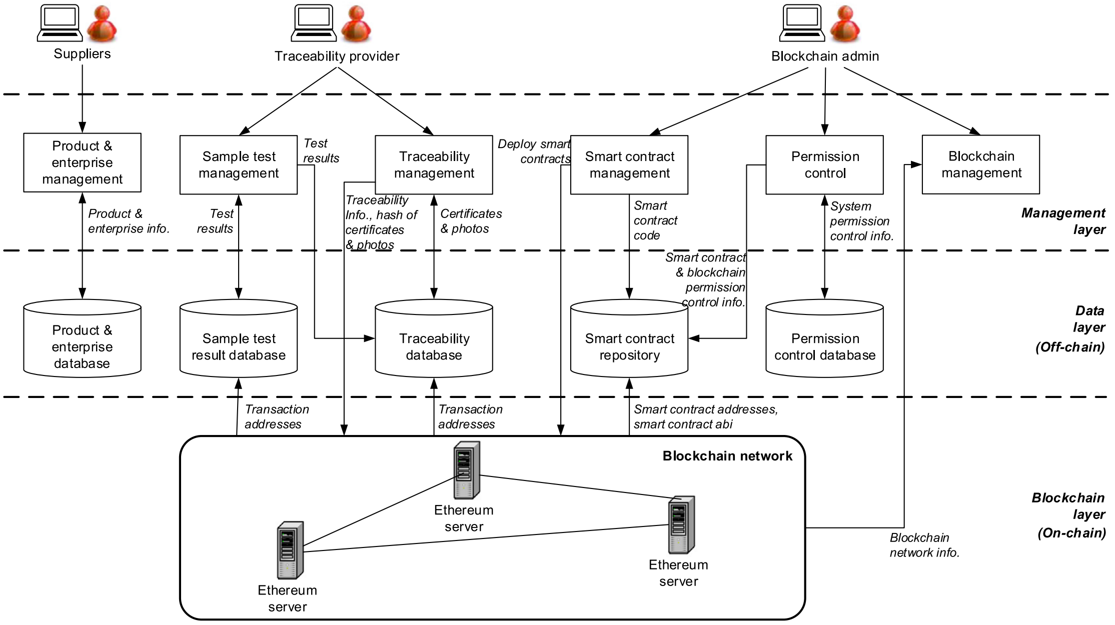

Fig. 2. Architecture of originChain.

can guarantee data integrity of such files. The hash of a file is a digest of the file. The transaction including the hash value is a proof-of-existence of the source file. Traceability information includes the information about product, such as batch number, traceability result and number, place of origin, manufacturer, manufacture date, and expiration date, and information of freight yard, such as loading port, and inspection date. This information is stored in variables of smart contracts. The information about blockchain-level permission control is stored in a separate smart contract.

The data off blockchain includes smart contract addresses, product/enterprise information, source traceability certificates and photos, and the code of smart contracts. The information displayed at front-end consists of product/enterprise/traceability information and blockchain information. The addresses of all the smart contracts that contain traceability data, product/enterprise information is stored off-chain as well due to their non-sensitivity. Traceability certificates and photos are large and not suitable to store on-chain. Therefore, originChain keeps these raw file (in .pdf/.jpg format) off-chain. The smart contracts deployed on blockchain are in a binary format rather than source code. Thus, the human-readable source code of smart contracts in high-level programming languages are stored off-chain in *Smart Contract Repository* for the blockchain administrators to manage and upgrade smart contracts.

The benefits of separately storing data on blockchain and off blockchain is to better leverage the properties of blockchain and avoid the limitations of blockchain. Blockchain can guarantee the integrity and immutability of the critical data on chain. The *non-tiny* files are stored off-chain so that the size of the blockchain would not grow so fast. Storing the hashes of off-chain files can further ensure the integrity of the off-chain files.

However, applying blockchain into the existing system makes the architecture of originChain more complex. More components were implemented to enable the interaction between blockchain and the conventional database. For data stored in transactions or log events of smart contracts, query on blockchain is not easy. To support easier query, all the data on-blockchain is replicated in conventional database as well with the corresponding transaction ID that adds the data into blockchain.

4.4.3. Design of smart contracts

As software connector, Blockchain provides coordination service for components within the system to coordinate their computations through using smart contracts [1]. The structural design of the smart contracts can affect the quality of the software. Fig. 3 illustrates the high-level design of smart contracts used in originChain.

There is a *factory contract* deployed on blockchain before running the system as a contract template to generate smart contracts corresponding to the agreements between originChain, service providers and product suppliers. The *factory contract* has a set

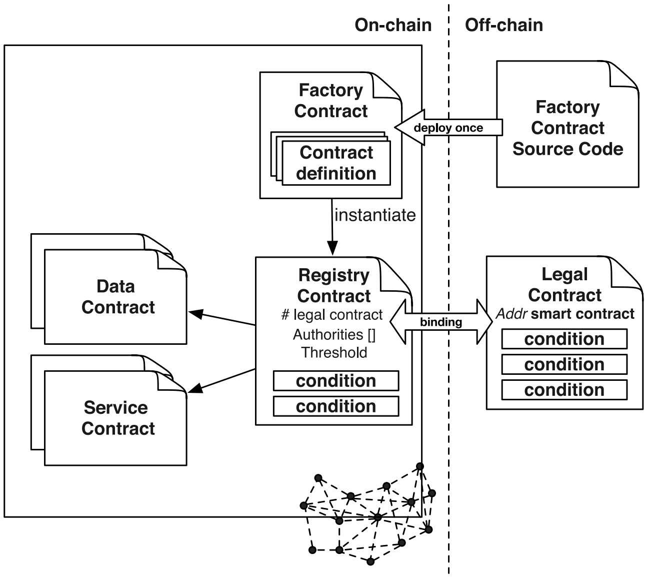

Fig. 3. Structural Design of Smart Contracts.

of contract templates, which represent different combinations of traceability services. The supplier submits a Web form through the front-end UI to call the *factory contract*. The conditions, e.g. what traceability services are selected, are defined in the web form are passed to the *factory contract* through parameters to instantiate new smart contracts. Other than the conditions defined in the corresponding legal agreement, the generated smart contract also implements the function to do compliance checking, which ensure that all the information required by regulation is provided. Other regulatory rules, for example, procedures, can be implemented and enforced by the smart contract as well. A *factory contract* provides a more flexible way to create smart contracts, and improves the confidence that the source code of smart contract is not modified by unauthorized people. When the *factory contract* is called, a *registry contract*, a *service contract* and a *data contract* are created.

A *registry contract* represents a legal agreement. The off-chain legal agreement is linked with the on-chain smart contract through adding the address of the smart contract into the legal agreement, and adding the hash of the legal contract back to a variable of the smart contract. So that a bridge between the legal agreement and the smart contract is established. The smart contract codifies the conditions defined in the legal agreement. These conditions can be checked and enforced automatically by the smart contract. The smart contract also enables automated regulatory compliance checking in terms of the required information and process.

A *registry contract* contains the addresses of a *service contract* and a *data contract*. Separating data and control is a basic principle in software development. Such separation allows the logic to be updated without affecting data. The data contract and control contract can be updated with different frequency.

The smart contracts running on blockchain can be accessed and called by all the blockchain participants by default, since there is no privileged user and every participant can join the network to access all the data and code (as a type of data) on blockchain. A function without proper permission control might be triggered by unauthorized users accidentally. The empirical study conducted in [20] shows that 7% smart contracts on public Ethereum can be terminated without authority. Thus, originChain uses an embedded mechanism for smart contract to check permission (Application design decision 5 in [1]) of every caller that invoke the operations defined in the smart contract.

4.4.4. Smart contracts enabled adaptability

In originChain, an activity (represented by a transaction) might be based on multiple authorities (represented by multiple account addresses). A smart contract specifies a list of possible addresses, which can authorizes the invocation of certain functions. A threshold is also defined as the minimum number of addresses required to authorize the invocation.

An authority is a off-chain validation oracle (Application Design Decision 4 in [1]) that signs the valid transactions based on state from the external world. This might block the transaction processing till a condition over the external state is verified by the validation oracle who owns one or multiple of the pre-defined addresses.

An additional reason to introduce multiple authorities is because blockchain does not provide any mechanism to recover a compromised private key. Losing key will result in permanent loss of control over an account and smart contracts. The mechanism of multiple authorities allows one participant to control smart contracts using more than one blockchain addresses, which reduces the risk of losing control over their smart contracts due to a lost or compromised private key. The list of the authority addresses pre-defined in a smart contract can be also updated through authorization from more than the minimum number of addresses.

4.4.5. Blockchain management

The current deployment of blockchain layer (as shown in Fig. 2) is a geographically distributed private blockchain (Application Design Decision 2 in [1]) within the company, which has three sites in different countries. The vision is to establish a platform that covers other organizations, like labs certified by government, big suppliers and retailers that have long term relationship with the company (e.g. e-commerce companies that has already built their reputation among the customers).

The permission of data on blockchain is defined in a permission control smart contract that defines permission of content management, permission of smart contract management, and permission of joining the blockchain, etc. To join the traceability platform, a company needs to send request off-line to get approval to join the blockchain network. The traceability company updates the permission control smart contract before the company can join the blockchain network and synchronize the historical transactions.

5. Analysis

5.1. Qualitative analysis: adaptability

The structural design of smart contracts affects the upgradability of smart contracts and the adaptability of the whole system. Adaptation scenarios in originChain include adding or removing traceability services from the legal agreement after the legal agreement is signed initially, or dynamic lab binding based on their availability. Such scenarios are supported by smart contracts.

5.1.1. Upgradability

As discussed in Section 4.3, the sequence of the traceability services can be dynamic because the legal agreements signed between the product supplier/retailer and the traceability company can change due to the customized traceability process. Customization is supported by *factory contract*. The function of a *service contract* can be updated by replacing the address of the old version of smart contract with the address of a new version. As long as the interface between the *service contract* and the *data contract* is the same, the updated *service contract* can still use the data stored in *data contract*.

In the case of having new requirements, for example, new types of information required by a new regulation or new services provided by the service providers, the *factory contract* can be upgraded to fulfill the new requirements. In this case, the old contract factory needs to be disabled, and the configuration of other modules is updated accordingly.

5.1.2. Dynamic binding

Dynamic lab binding is enabled by the mechanism of multiple authorities. In Bitcoin, multi-signature is a mechanism that requires more than one private key to authorize a Bitcoin transaction. In Ethereum, multi-signature mechanism is implemented as smart contract. More specifically, an M-of-N multi-signature defines that M out of N public keys are required to authorize a transaction. We call M the threshold of authority.

multiSignature contract in Listing 1 provides the multi-signature mechanism. In originChain, users send a message to originChain to request a certain certificate. Issuance of the certificate requires the approval from both the corresponding service provider and originChain. *issueCert* contract inherits *multiSignature* to use the mechanism.

The address of the authorities are pre-defined in *multiSignature* contract. *agreeSignature()* is a modifier that wraps additional code to the function *issue()* to ensure that certain conditions are met before proceeding to execute the rest of *issue()*. An authority invokes *issue()* to approve the request. *agreeResult()* checks whether the number of signatures is above the threshold every time when *issue()* is invoked. If so, the certificate can be issued. The request can by withdrawn by invoking the function *cancelAgreeRequest()*.

```
contract multiSignature{
  uint total;
  address[] authorities;
  uint agreeThreshold;
  address agreeRequester;
  mapping(address => bool) agreeState;
  modifier agreeSignature(){
    agreeState[msg.sender] = true;
    if(agreeResult()){_;}
  }
  function agreeResult() internal returns
  (bool signatureResult){
    uint k = 0;
    for(uint i = 0; i < total; i++){
      if(agreeState[participants[i]] == true)
          k++;
    }
    if(k >= agreeThreshold)
      return true;
    else
      return false;
  }
  function initialAgree() internal{...}
  function cancelAgreeRequest(){
    if(msg.sender == agreeRequester)
      initialAgree();
  }
  ...
}
contract issueCert is multiSignature{
    string temID;
    bytes32 temCertHash;
    mapping(string => bytes32) certificate;
    function set(string ID, bytes32 certHash){
        temID = ID;
        temCertHash = certHash;
    }
    function issue() agreeSignature(){
        certificate[temID] = temCertHash;
    }
    ...
}
```

Listing 1: multiSignature Contract

```
contract dynamicBinding{
  struct hashSecret{
    bytes32 hashKey;
    bool init;
  }
  mapping (address => hashSecret) secret;
  /distinguish the struct initiated by different
      address/
  function initial(bytes32 key){
    hashSecret a = secret[msg.sender];
    if(a.init != true){
      a.hashKey = key;
      a.init = true;}
  }
  function changeKey(string oldKey, bytes32 newKey){
    hashSecret a = secret[msg.sender];
    if(a.init == true)
      if(a.hashKey == sha256(oldKey))
        a.hashKey = newKey;
}
  modifier verify(address initiator, string inputKey
      ){
    hashSecret a = secret[initiator];
    if(a.hashKey == sha256(inputKey)){_;}
    }
...
}
contract bindingLab is dynamicBinding{
    ...
    function sampleTest(address initiator, string
        key)
    verify(initiator, key){...}
    /passing the parameter to the modifier/
}
```

Listing 2: dyanmicBinding contract

dynamicBinding contract in Listing 2 provides a flexible permission control, which allows a user to authorize the execution of a smart contract by using a hash secret initiated by originChain. The hash secret is generated off-chain and verified on-chain. Similar as above, *bindingLab* contract uses the mechansim by inheriting *dynamicBinding* contract. An originChain employee invokes the *initial* function to initialize a hash secret, which links the hash key with the address of the employee so that only the employee has permission to change the hash secret using the *changeKey()* after the secret is revealed. The hash secret needs to be exchanged off-chain to the authorized labs. The modifier *verify()* verifies secret key. If the result is true, the lab providing the secret key can proceed the rest of *sampleTest()* to upload the result of a sample test. For verification, The secret is sent through a transaction in plain-text to the smart contract. Thus, the hash key is revealed after *verify()* being invoked. The hash secrets are applicable to permission control of both off-chain and on-chain data.

5.2. Quantitative analysis: latency of writing and reading

We conducted experiments to test the performance of originChain. As discussed above, the hash of traceability certificates and photos is stored on-chain to guarantee data integrity. Thus, the main operations on data by originChain and the end users include generating and storing the hash value on blockchain, and querying and comparing hash values. Our experiments compare the latency of writing and reading hash value by using blockchain and central database respectively. The traceability company currently maintains a central database at one of their site. We conducted three groups of experiments on a local database (for the site hosting the database), a physically distributed blockchain, and a remote database (for the other branch offices that access the database remotely).

We deployed a Ethereum-based blockchain. The *difficulty* of the blockchain is $0x4000$. On the public Ethereum blockchain,

Table 1
Writing latency (ms).

	Blockchain		Database	
	Inclusion	Commitment	Local	Remote
Minimum	1348	72 870	1	418
First quartile	15 971	152 749	8	435
Median	25 494	176 332	10	439
Third quartile	35 666	204 159	11	446
Maximum	106 374	592 270	20	542
Average	29 453	187 938	10	441

Table 2
Reading latency (ms).

	Blockchain	Database	
		Local	Remote
Minimum	8	1	422
First Quartile	10	12	437
Median	11	15	443
Third Quartile	13	17	449
Maximum	129	33	485
Average	17	15	444

the difficulty dynamically adjusts to generate blocks every 12 s on average. On a consortium/private blockchain, difficulty can be configured according to the desired throughput of the system. In our experiment, the average block interval is 13.3 s, the maximum block interval is up to 58 seconds, and the minimum block interval is 1 second. For each type of the latency, we ran an experiment for 200 times.

Table 1 shows the result of writing latency. The time unit is millisecond. Two times of blockchain is reported, including *inclusion* time and *commitment* time. Inclusion time is the time spent for the transaction to be included first time into a block on the blockchain. To confirm that a transaction is appended to blockchain and nearly impossible to be modified, a commonly used security strategy is to wait for a number of blocks to be generated after the transaction is included into a block. Commonly used value is 6 in public Bitcoin, and 12 in public Ethereum. The value on a consortium/private blockchain is decided by the participants, and the latency of a committed write is calculated based on the block interval time. In our experiment, we also use 12 blocks for the commitment time. Writing latency of remote database is longer than local database due to additional network latency. Writing latency of blockchain is much longer than remote database because it includes network latency for propagating transactions and blocks and latency introduced by the consensus process.

Table 2 compares the reading latency of the three configurations. The time unit is millisecond. The reading latency of blockchain is comparable to the reading latency of local database because reading blockchain is processed locally without sending transaction to the blockchain network. Thus, it can response immediately.

6. Discussion

6.1. Architectural design of blockchain-based systems

Due to the unique properties of blockchain, there are some blockchain applications specific design considerations, for example, the consideration of on-chain and off-chain, as discussed in the next subsection. On the other hand, because smart contracts are programs running on blockchain, some of the existing architectural design principles might be applicable directly to smart contracts. However, in the case of using a public blockchain, the structural design of the smart contract has large impact on the cost . The cost of deploying a smart contract depends on the size of the smart contract(s) since the code of smart contract is stored on blockchain, which costs data storage fee that is proportional to the size of the data. Thus, a structural design with more lines of code costs more. Although monetary cost is not an issue for a consortium/private blockchain, blockchain size is still a design concern because the size of the ledger is keep growing due to immutability and full replication of blockchain on every participant. The constraints on the size of transaction and block also restrict the complexity of smart contracts. Besides, a more structural design may affect performance because more transactions may be required.

6.2. Separation between on-chain and off-chain

The first and most important consideration for designing application on blockchain is what data and computation should be put on-chain. The two main limitations of blockchain, including performance and privacy, need to be considered. Performance of the system depends on deployment of the blockchain. For example, a consortium/private blockchain can be configured to perform much better than a public blockchain. In originChain, due to low writing throughput because of the large granularity of traceability information, limited throughput of blockchain is not the main concern. However, the data on blockchain is publicly accessible to all the participants of the blockchain network, the private data (for example, the information of customers) should not be stored on-chain. In the context of traceability, large sized sensitive raw data (for example, traceability certificates and photos) are required to be immutable. Thus, the hash of raw data is stored on-chain while the corresponding source files are placed off-chain.

6.3. Adaptability of blockchain-based systems

Adaptability is a quality attribute required by many industrial projects that are inherently dynamic. So far, adaptability is rarely discussed in the existing work about blockchain-based systems. We view blockchain as one component of a large distributed system. In originChain, some business logic is implemented as smart contracts. Thus, the structural design of smart contracts can affect the adaptability of the whole system. However, if blockchain is used as data storage only, not much can be done to affect adaptability of the whole system. Moving some logic to the blockchain can leverage the trustworthiness (achieved by interoperability and transparency of the operation as well as data) of blockchain as a computational platform.

7. Conclusion

Design of blockchain-based applications is challenging to developers. In this paper, we share our experience of building orginChain, a blockhain-based product traceability system. We applied the software design approaches we previously proposed into originChain and discussed how originChain is designed using blockchain taking into account the properties and limitations of blockchain. Our analysis shows that the structural design of smart contracts can improve the adaptability of the system, and the experiment demonstrates that using blockchain only affects write operations due to the consensus process while has no impact on read operations since every participant hosts a local full copy of the blockchain.

References

[1] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A.B. Tran, S. Chen, The blockchain as a software connector, in: The 13th Working IEEE/IFIP Conference on Software Architecture, WICSA, Venice, Italy, 2016.

[2] F. Tschorsch, B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies, IEEE Commun. Surv. Tutor. 18 (3) (2016) 464.

[3] Distributed ledger technology: Beyond blockchain, in: Technical Report, 2016. UK Government Chief Scientific Adviser.

[4] M. Staples, S. Chen, S. Falamaki, A. Ponomarev, P. Rimba, A.B.T.I. Weber, X. Xu, J. Zhu, Risks and opportunities for systems using blockchain and smart contracts, in: Technical Report, Sydney, 2017 Data61(CSIRO).

[5] N.R. Mehta, N. Medvidovic, S. Phadke, Towards a taxonomy of software connectors, in: ICSE, 2000.

[6] S.K. Lo, X. Xu, Y.K. Chiam, Q. Lu, Evaluating suitability of applying blockchain, in: The 22nd International Conference on Engineering of Complex Computer Systems, ICECCS, Fukuoka, Japan, 2017.

[7] Q. Lu, X. Xu, Adaptable blockchain-based systems: A case study for product traceability, IEEE Softw. 34 (6) (2017) 21–27.

[8] M. Swan, Blockchain: Blueprint for a New Economy, O'Reilly, US, 2015.

[9] S. Omohundro, Cryptocurrencies, smart contracts, and artificial intelligence, AI Matters 1 (2) (2014) 19–21.

[10] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in: BPM, Springer, Rio de Janeiro, Brazil, 2016, pp. 329–347.

[11] G. Zyskind, O. Nathan, A. Pentland, Decentralizing privacy: Using blockchain to protect personal data, in: SPW, 2015.

[12] M. Ali, J. Nelson, R. Shea, M.J. Freedman, Blockstack: A global naming and storage system secured by blockchains, in: USENIX ATC, Santa Clara, CA, 2016.

[13] S. Matsumoto, R. Reischuk, Ikp: Turning a pki around with decentralized automated incentives, in: IEEE SSP, San Jose, CA, US, 2017.

[14] X. Liang, S. Shetty, D. Tosh, et al., ProvChain: A blockchain-based data provenance architecture in cloud enviornment with enhanced privacy and availability, in: CCGrid, 2017.

[15] X. Xu, I. Webber, M. Staples, et al., A Taxonomy of Blockchain-based Systems for Architecture Design, in: IEEE International Conference on Software Architecture, ICSA, Gothenburg, Sweden, 2017.

[16] M. Bartoletti, L. Pompianu, An empirical analysis of smart contracts: Platforms, applications, and design patterns, (2017) ArXiv e-prints.

[17] P. Zhang, J. White, D. Schmidt, G. Lenz, Applying software patterns to address interoperability in blockchain-based healthcare apps, (2017) ArXiv e-prints.

[18] J. Eberhardt, S. Tai, On or off the blockchain? Insights on off-chaining computation and data, in: ESOCC 2017(6th European Conference on Service-Oriented and Cloud Computing), Springer International Publishing, Oslo, Norway, 2017, pp. 3–15.

[19] O. Kharif, Walmart tackles food safety with trial of blockchain, (2016).

[20] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A.B. Tran, P. Rimba, On availability for blockchain-based systems, in: SRDS, Hongkong, China, 2017.

Sherry (Xiwei) Xu is a senior research scientist in Architecture & Analytics Platforms (AAP) team at Data61, CSIRO (based in ATP, Sydney). She is also a Conjoint Lecturer at the School of Computer Science and Engineering (CSE) of the University of New South Wales (UNSW). She has a PhD from UNSW. Her main research interest is software architecture. She also does research in the areas of service computing, business process, cloud computing and dependability. She started working on blockchain since 2015. She is doing research on blockchain from software architecture perspective, for example, trade-off analysis, decision making and evaluation framework etc.

Dr. Qingghua Lu is s a senior research scientist at CSIRO, Australia. Before she joined CSIRO, she was an associate professor at China University of Petroleum. She formerly worked as a researcher at NICTA (National ICT Australia). She received her PhD from University of New South Wales in 2013. Her research interest includes architecture design of blockchain applications, blockchain as a service, model-driven development of blockchain applications, reliability of cloud computing, and service engineering. She has published more than 70 peer-reviewed academic papers in international journals and conferences. She is an IEEE member and serves on the Program Committees of a number of international conferences in blockchain, cloud computing, big data and software engineering community. E-mail: qinghua.lu@data61.csiro.au.

Dr/Prof. Liming Zhu is a Research Director at Data61, CSIRO. He is also a conjoint full professor at University of New South Wales (UNSW). He is the chairperson of Standards Australia's blockchain and distributed ledger committee. His research program has more than 200 people innovating in the area of big data platforms, computational science, blockchain, regulation technology, privacy and cybersecurity. He has published more than 150 academic papers on software architecture, secure systems and data analytics infrastructure.