IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 6, 2020, accepted December 24, 2020, date of publication December 29, 2020, date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048004

PABC: A Patent Application System Based on Blockchain

SHUYU BIAN[1], GUOHUA SHEN[1,2,3], ZHIQIU HUANG[1,2,3], YANG YANG[1], JINGHAN LI[1], AND XIAOYU ZHANG[1]

[1]College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210093, China
[3]Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

Corresponding author: Guohua Shen (ghshen@nuaa.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2018YFB1003902, in part by the National Natural Science Foundation of China under Grant 61772270, and in part by the Funding of the Key Laboratory of Safety-Critical Software under Grant 1015- XCA1816403.

ABSTRACT Under the protection of the patent system, an inventor can patent his or her invention, to make an economic profit from the patent, thus encouraging further invention. However, a patent is territorial. To get an international patent, the invention should be patented several times in all target patent offices, even with the help of international conventions like PCT, leading to inefficiency, expensiveness, and uncertainty. Therefore, we construct an international patent application system based on a permissioned blockchain named PABC. Patent applications can be approved or rejected without relying on one patent office. Each patent office in a country or region manages a peer node of the system, reaching agreements of all the applications. First, we build the model of the system and detail the three-layer architecture of PABC Core. Then, we implement PABC on a permissioned blockchain platform — Hyperledger Fabric. In addition, experiments about its throughput, transaction latency, and failure rate were carried out. The results show that PABC can meet current needs.

INDEX TERMS Patent, blockchain, smart contract, Hyperledger fabric.

I. INTRODUCTION

Patent has a history of more than 500 years since the first patent law in the world was established in the Republic of Venice in 1474 [1]. Under the protection of the patent system, an inventor can patent his or her invention, in order to exclude others from making, using, selling, and importing the invention for a limited period of years. In this way, the inventor can make an economic profit from the patent, thus encouraging further invention [2]. According to World Intellectual Property Organization (WIPO), PCT applications increased from 39,994 in 1995 to 265,307 in 2019 [3]. A growing number of people are willing to patent their inventions.

However, patents are territorial. For example, a patent that was given by the United States Patent and Trademark Office (USPTO) is invalid in China. Existing international conventions, such as PCT, can help the inventor apply for an international patent, but the procedure is still complex — a patent application should be reviewed by WIPO and several target patent offices in different countries or regions, leading to low efficiency, high price, and uncertainty.

Blockchain technology may provide a solution to the above problem. A number of nodes, connected within a distributed network, maintain a special database that records all the operations on a specific business object. Because any operation on this object will be encapsulated in blocks, and linked as a chain one by one, much like transactions recorded in a ledger, the database can be called **blockchain** or **ledger**. If patent offices are considered as nodes, and patents as a business object, a patent application system based on a blockchain network. Due to the consensus mechanism and security features on blockchain, all the records of operations on a patent application are admitted by all patent offices instead of a specific one. These records cannot be distorted once written into the ledger either. In other words, patents that have been applied for in this network become credible across patent offices. Thus, an invention is ''Patent Once, be Valid Anywhere (POVA),'' which reduces time and examiner costs.

The associate editor coordinating the review of this manuscript and approving it for publication was Asad Waqar Malik.

Because the ledger is stored physically in all the nodes, single points of failure can also be avoided. The failure of a few nodes will not affect the normal operations of the network.

Therefore, we construct a patent application system based on blockchain (PABC). The main contributions are:

1. We propose the system model of PABC. It includes users, blockchain network, external decentralized storage, external certificate authority, and clients. Unlike permissionless blockchain platforms such as Bitcoin and Ethereum, certificate authorities are introduced into the system to authenticate both the nodes and clients connected to the blockchain network. Malicious clients or nodes which do not have valid certificates cannot access the network.

2. The proposed system is implemented on Hyperledger Fabric [6]. By analyzing the business logic of apply for and review a patent, we describe the network structure, smart contracts, and the client of the patent application system.

3. We run benchmarks on PABC and analyze the performance of the system. Three metrics: throughput, transaction latency, and failure rate were evaluated in different conditions and the results meet the requirements of patent application.

The remainder of this paper is organized as follows. Section II summarizes the existing related work. Then we propose the system model and implement PABC in Section III and Section IV. Finally, benchmarks and the analysis of the results are shown in Section V.

II. RELATED WORK

A. BLOCKCHAIN AND SMART CONTRACT

The concept of blockchain, and Bitcoin, the first platform of blockchain, was proposed by Satoshi Nakamoto in his paper about electronic payment in 2008 [4]. In Bitcoin, a proof-of-work (PoW) consensus algorithm is implemented to ensure that every transactions are valid (i.e. not double-spent). All the nodes should make efforts to find a value that gives the block's hash which begins with a number of zero bits (generally speaking, mining), in order to generate a new block and obtain its remuneration — several bitcoins. However, there are some problems with Bitcoin. In order to keep an average speed to generate a block under the background of the sharp increase of total CPU power, the difficulty to find the correct value is also increasing. In this case, the verification of transaction is becoming both time-consuming and power-consuming [19].

Therefore, a number of new blockchains came into being. Ethereum [5] and Litecoin [20] are two typical cases that present different PoW algorithms to improve the performance of generating and verifying a block. Compared with Bitcoin, both Ethereum and Litecoin raise the time efficiency and reduce the power consumption [21].

Other blockchains try to design consensus algorithms which are different from PoW. For example, Hyperledger Fabric [6] uses Practical Byzantine Fault Tolerance (PBFT) algorithm that does not require a native cryptocurrency to incent costly mining and can be deployed with roughly the same operational cost as any other distributed system. However, Fabric is a permissioned blockchain, requiring that only permitted nodes can join in the blockchain. In addition, too many nodes make significantly negative effect on its performance, including throughput and latency [22]. It supports fewer nodes than blockchains mentioned above.

Smart contract is a digital contract that can be programmed by Turing complete language. By using smart contracts, blockchain platforms, including Ethereum, Hyperledger Fabric, etc., much more complex transactions are supported and further extend the domain of blockchain applications. In the domain of crowdsourcing, Li, *et al.* proposed a blockchain-based framework for crowdsourcing called CrowdBC [7]. In Industrial Internet of Things, He, *et al.* implement a status monitoring system based on blockchain to prevent unauthorized software updating [8]. In the domain of food safety, Tao designed a hierarchical blockchain network and implemented a food safety supervision system [9].

In order to design PABC, we should consider the throughput, the cost, and the support of smart contracts to enable complex program logic of the blockchain. For the throughput, data from [3] shows that more than 700 inventions were applied for PCT patents per day in 2019. If we put national patents into account, the number of patent applications will be much larger. For the cost, it should be stable and lower than that of the existing patent application systems. However, the prices of BTC (Bitcoin) and ETH (Ethereum) are high and keep on fluctuating [23]. The support of smart contracts is also necessary because of the complexity of the procedure of patent application. According to the above, we design PABC based on Hyperledger Fabric.

B. BLOCKCHAIN RESEARCHES ON INTELLECTUAL PROPERTY

Intellectual property (IP), very broadly, means the legal rights which result from intellectual activity in the industrial, scientific, literary, and artistic fields [14]. It can be divided into four forms: trade secrets, copyright, trademarks, and patents [15], but most of the existing researches focus on how to protect copyright and trademarks. For example, Liang, *et al.* developed a dual-chain digital copyright registration and transaction system based on blockchain [10]; Ouyang, *et al.* made an attempt to combine blockchain technology with copyright protection [11]; [18] designed a blockchain-based system for trademarks with Hyperledger and outlined workflow for registering, distributing, and validating trademarks.

However, it is difficult to apply these methods to protecting patents because the only user type in their proposed systems is applicant. Compared with copyright and trademarks, examiners are necessary in patent applications because patents are usually technical, but the introduction of examiners who have technical knowledge also makes the system more complex. We have to design a more suitable system model based on blockchain on patent application.

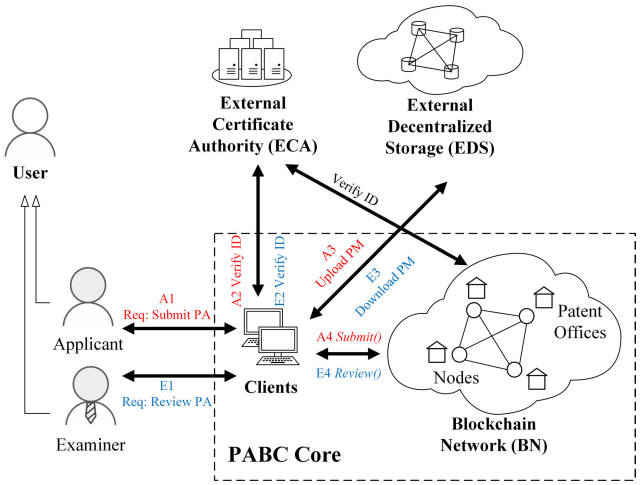


FIGURE 1. The system model of PABC.

III. SYSTEM MODEL

A. OVERVIEW

Traditional patent application system is based on C/S architecture. Both applicants and examiners should use client software to apply for or review patents, and all the patent data are stored in a server which is managed by the local patent office. In PABC, patent data from all over the world should be synchronized by patent offices from different countries and regions, so we replace the traditional server with the decentralized blockchain network (to store summary information and states of patent applications) and the decentralized storage (to store patent materials). We also introduce certificate authorities to ensure that patent offices, applicants, and examiners have valid and legal identities.

Therefore, we put forward the system model of PABC (see Fig. 1). The entities in the model will be detailed firstly, and their interactions will be introduced by presenting two examples. PABC Core is the most important part of the system, so we divide it into three layers, which will be detailed in the next subsection.

The system model of PABC consists of five entities: User (including Applicant and Examiner), Clients, Blockchain Network, External Certificate Authorities, and External Decentralized Storage.

Users, are the people who interact with PABC. They can be divided into two groups: applicants and examiners, according to the difference in permissions to use the system. Applicants can apply for and disclose the patents, while examiners can receive and review patent applications.

Clients, are programs which users (both applicants and examiners) install in their computer. They encapsulate all the very specific operations to interact with EDS, BN, and ECA, and provide an easy-to-use interface to users for the convenience of receiving user's requests (e.g., submitting a patent application).

Blockchain Network (**BN**), is a peer-to-peer (P2P) network, constructed by connecting nodes (presented as circles in the figure) that are managed by patent offices (presented as houses) all over the world. All the nodes are running the same smart contracts about patent application and the same consensus protocol to maintain the blockchain that records all the transactions.

External Certificate Authorities (**ECA**), are organizations which provide legal identities. To avoid malicious nodes, all the patent offices must get Patent Office's Certificates from ECA to connect to BN. ECA also helps to divide users into applicants and examiners, because examiners should have qualifications to review patent applications while applicants should not. An examiner can use PABC only when he or she passes the qualification examination by patent offices and gets Examiner's Certificate from CA. Compared with examiners, applicants can easily register Applicant's Certificates by providing real-name information to ECA.

External Decentralized Storage (**EDS**), is used to store patent materials. The amount of data of patent materials is often large, which may make a significant negative impact on the performance of blockchain (To prove this viewpoint, related experiments have been performed and will be detailed in Section V). Some famous decentralized storage platforms can solve the problem. For instance, IPFS [17]. It calculates a hash string of the data that a user uploads, and people can download the data by providing the hash string. Moreover, once uploaded, the data cannot be deleted. Modifying the data is permitted, but a new hash string will be given as the key of the modified data, while the old data is still accessible by using the old hash string. As the characteristics of IPFS above, we use IPFS as EDS. The hash string, instead of all the patent materials, is stored in the blockchain.

In Fig. 1, two examples demonstrate how these five entities interact. The red texts show the simplified sequence of interactions when an applicant applies for a patent. Firstly, the applicant sends a request of submitting a patent application (PA) with all the information about the PA, including patent materials (PM) to the client. Secondly, the client verifies the applicant's identity by interacting with ECA. If the applicant has a valid Applicant's Certificate, the client will upload patent materials to EDS and call the smart contract method *Submit()* in BN to finish the applicant's requests. Similarly, the blue texts show the progress of reviewing a PA. The difference between submitting and reviewing a PA is operations when the five entities interact with each other. For example, the client should download the PM instead of upload the PM when an examiner reviews a PA. However, no matter what the request is, nodes in BN always interact with ECA, in order to verify each other's identity. More details about user's requests and interactions between the entities will be presented in Section IV.

B. THREE-LAYER ARCHITECTURE OF PABC CORE

In order to further detail the PABC Core, we divide it into three layers: network layer, contract layer, and application layer, as shown in Fig. 2.

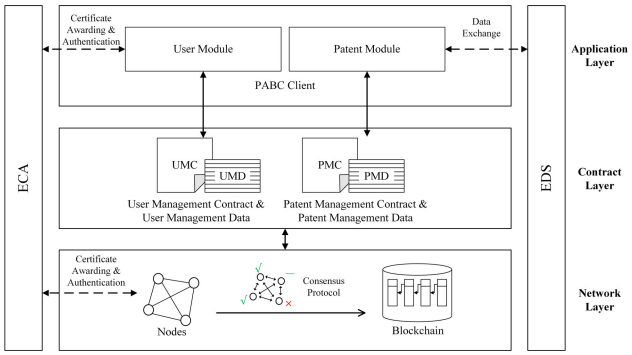


FIGURE 2. The three-layer architecture of PABC Core.

1) NETWORK LAYER

Network layer mainly focuses on how to construct and maintain a decentralized network. It includes three elements: nodes, consensus protocol, and blockchain.

Nodes are the basic units to construct the network. They are managed by patent offices in different countries and regions, running consensus protocol all the time to synchronize the blockchain. Generally speaking, one patent office manages only one node in the network, because more peer nodes mean more power to control the network, based on the consensus protocol PBFT that will be introduced next. Nodes' legal identities are guaranteed by ECA.

Consensus protocol defines the way to let nodes reach an agreement on a transaction. The most commonly used protocol in both Bitcoin and Ethereum, Proof of Work (PoW), encourages nodes to consume their computing power (in other words, mining) to verify transactions by giving an amount of virtual currency like BTC and ETH. However, virtual currency is useless in PABC, for verifying applications itself is of benefit to patent offices. Therefore, PBFT is used as the consensus protocol. PBFT can tolerate one-third of fault nodes in the network and nodes do not need to waste much computing power on verifying. More details about PBFT can be achieved from [16].

Blockchain records all the transactions from the genesis of PABC to the present. Once approved by most of the nodes, transactions will be recorded into a block, and the block will be added to the head of the blockchain. To prevent transactions from being tampered with, each block has a head which contains metadata (like timestamp, sequence number, etc.) and the hash values of the previous block and the current block. By reading the blockchain in order, the current state of the whole blockchain can be inferred.

Network layer is the base layer of PABC Core. Smart contracts execute on nodes. Then, the nodes reach a consensus on the results of executed methods defined in contracts and write the results into the blockchain.

2) CONTRACT LAYER

Contract layer focuses on the detail of smart contracts and corresponding data structure. In PABC, two smart contracts are proposed: User Management Contract (UMC) and Patent Management Contract (PMC). They manage User Management Data (UMD) and Patent Management Data (PMD) respectively. Specifically, a set of methods are defined in both contracts. These methods include a series of operations on reading and writing business data from and to the blockchain. Once a method is called, a transaction will be proposed. The transaction contains the method name, input parameters, previous blockchain state, and new blockchain state after modifying the business data in the blockchain. If most of the nodes approve the transaction, it will soon be added to the blockchain, i.e., the business data will be modified.

3) APPLICATION LAYER

Application layer encapsulates all the very specific operations to interact with contract layer and external entities (ECA and EDS) and provide an interface to users. PABC Client operates at this layer. It contains two modules: User Module and Patent Module.

a: USER MODULE

Users can register or verify certificates or patent accounts by using this module. This module calls methods from UMC, as well as interacts with ECA to complete the user's request.

b: PATENT MODULE

Users can apply for or review patent applications by using this module. This module calls methods from PMC to complete the user's request. It also connects with EDS in order to upload or download patent materials. However, these are transparent to users, which means user do not need to know where the patent materials are stored.

IV. THE IMPLEMENTATION OF PABC

In this section, the implementation of the system model of PABC on Hyperledger Fabric will be introduced. Hyperledger Fabric is an open-source permissioned distributed ledger technology platform, designed for use in enterprise contexts. It has a highly modular and configurable architecture and supports general-purpose programming languages such as Java, which enables great convenience for developers to develop blockchain applications [6].

We first detail the business logic of the patent application. Then, we implement PABC by following the three-layer architecture of PABC Core.

A. BUSINESS LOGIC

There is little difference between the processes of patent application in different countries and regions. For example, when an applicant submits a patent application to USPTO, the application will first be preliminarily examined by an examiner to check whether it has been filled in completely. After that, the examiner will review the content and determine whether the application can be approved or not. If the applicant wants to get a patent in China, he or she will follow similar steps like do in the U.S. The only difference between U.S. and China may be the default time to disclose the patent.

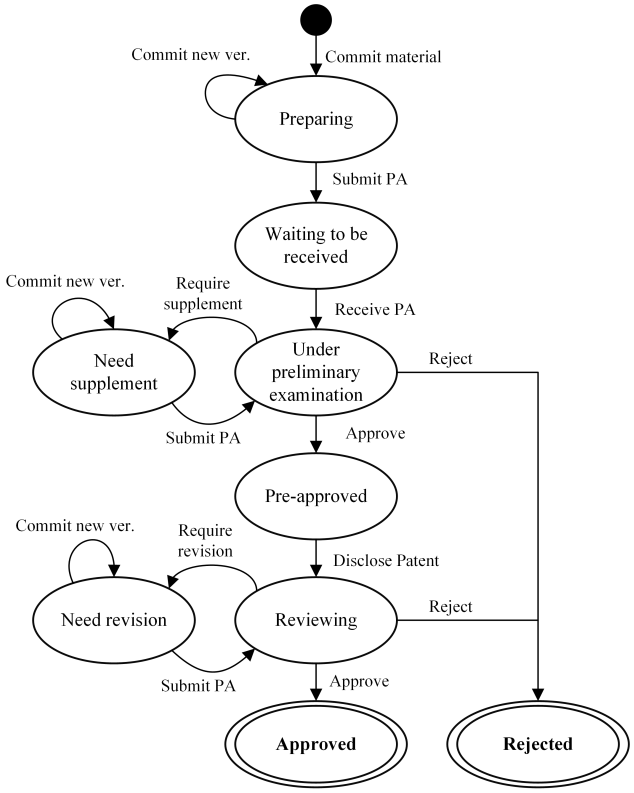


FIGURE 3. The state machine of patent application in PABC.

To simplify the whole system, a unified business logic of patent application is designed. Suppose an applicant U_a conducts a patent application *PA* via our system PABC, and U_e is an examiner, the state machine is shown in Fig. 3.

There are 9 PA states as following:

Preparing. In this state, the applicant U_a is preparing for the patent application *PA*. U_a has already committed at least one version of *PA* and required materials to PABC, and can still commit a new version as U_a wishes. If U_a decides to submit *PA*, the state will change to **Waiting to be received**.

Waiting to be received. In this state, U_a is waiting for an examiner to receive *PA*. U_a cannot commit a new version. If an examiner U_e receives *PA*, the state will change to **Under preliminary examination**.

Under preliminary examination. In this state, *PA* is under preliminary examination. If U_e approves *PA*, the state will change to **Pre-approved**. If *PA* is incomplete, U_e will require U_a to supplement *PA* and change the state to **Need supplement**. If U_e rejects *PA*, the state will change to **Rejected**, which means this *PA* is failed.

Need supplement. In this state, U_a has the second chance to modify *PA* and required materials. U_a can commit a new version several times as U_a wishes, but once U_a decides to submit *PA*, the state will change to **Under preliminary examination**.

Pre-approved. In this state, the preliminary examination of *PA* has passed, and U_a should disclose *PA* at a proper date. If U_a discloses *PA*, the state will change to **Reviewing**.

Reviewing. In this state, *PA* is disclosed and U_e is reviewing the content of *PA*. This state is very similar to **Under preliminary examination**. If U_e approves *PA*, the state will change to **Approved**, which means U_a successfully patents his or her invention. If U_e rejects *PA*, the state will change to **Rejected**. If U_e considers that *PA* requires revision, the state will change to **Need revision**.

Need revision. In this state, U_a has the last chance to modify *PA* and required materials. U_a can commit a new version several times as U_a wishes, but once U_a decides to submit *PA*, the state will change to **Reviewing**.

Approved. *PA* has been approved.

Rejected. *PA* has been rejected and U_a has no chance to modify it.

B. NETWORK ARCHITECTURE

Fabric network is working in the Network Layer of PABC Core. To establish a Fabric network, these entities should be implemented: node, consensus protocol, state database, and CA.

1) NODES

Fabric uses a new architecture called three-phase *execute-order-validate* architecture. So nodes connected to the Fabric network take up one of three roles (Client, Peer, and Orderer) for high performance. Because peers execute a smart contract and endorse the validity of a transaction, one patent office can manage only one peer in the network for the sake of fairness, just as mentioned in Section III-B. The number of orderers is not limited.

2) CONSENSUS PROTOCOL

Fabric supports pluggable consensus protocol, including PBFT, which was mentioned in the previous section. PBFT is used in PABC.

3) STATE DATABASE

State database is a part of Fabric Ledger, used to store the current state of blockchain (see Fig. 4). Fabric supports LevelDB and CouchDB. LevelDB is the default database, storing data as simple key-value pairs. However, CouchDB models data as JSON and supports to issue rich queries against data values, further supports advanced operations on the database. Therefore, we adopt CouchDB as the state database of PABC.

4) CA

CA is used to authenticate patent offices' identities. Fabric supports external CA, so it is possible to use existing CAs to take charge of this entity.

C. USER MANAGEMENT CONTRACT (UMC)

UMC is in charge of managing user data that recorded in UMD. There are four fields, including *uid* (user ID), *name* (user's real name), *role* (applicant or examiner), and *certificate*, see Table 1.

There are two methods defined in UMC.

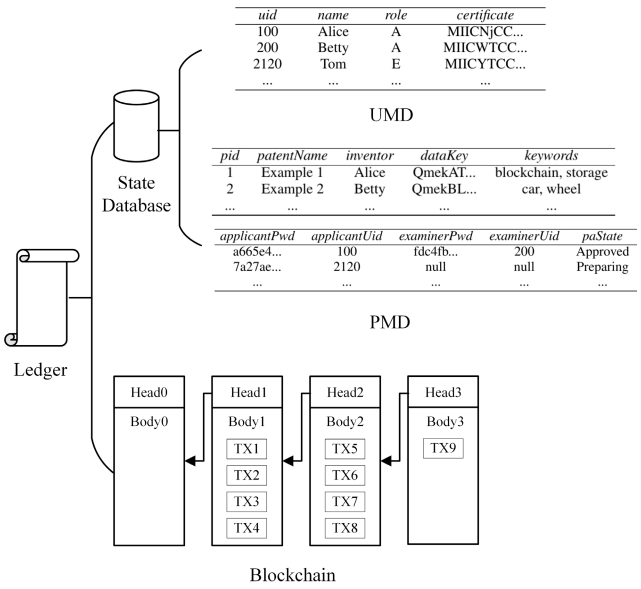


FIGURE 4. The architecture of Fabric Ledger.

TABLE 1. The structure of User Management Data (UMD).

uid	*name*	*role*	*certificate*
100	Alice	A	MIICNjCC...
200	Betty	A	MIICWTCC...
2120	Tom	E	MIICYTCC...
...	...	...	...

1) REGISTER

See Algorithm 1. Given a digital certificate (meeting the X.509 standard), the method will first validate it. If the certificate is issued by a legal CA and is not out of date, the method will extract the user's real name and role from it, generate a unique *uid*, and add a record into UMD.

2) VERIFY

See Algorithm 2. Given enough information, the method will check whether there is one user who satisfies the information that has been registered. The parameters can be *uid*, *name*, *role*, *certificate*, information contained in the certificate, or a combination of the former. The method only returns **Yes**, **No**, or **Insufficient Information**, so that no one can achieve the user's private information unless he/she has already had.

This method can be used to verify the identity of an applicant who wants to apply for a patent which has been accepted in a regional office (like USPTO), because an accepted patent usually provide enough information about its applicant.

D. PATENT MANAGEMENT CONTRACT (PMC)

PMC is in charge of managing patent data recorded in PMD. As is shown in Table 2., there are 10 fields, including *pid* (patent ID), *patentName*, *inventor* (may not be the same as applicant), *dataKey* (hash value or other types of link to download patent material from EDS), *keywords*, *applicantPwd*, *applicantUid*, *examinerPwd*, *examinerUid*, and *paState*. All the passwords are protected by using SHA-1, a cryptographic algorithm.

There are six methods defined in PMC.

Algorithm 1 Register

Input: *certificate*
Output: *uid* or ERROR

```
if certificate is valid then
    user ← UMD.getUser(certificate);
    if user = NULL then
        user ← new User(certificate);
        user.name = certificate.getRealName();
        if certificate.getRole() = Role.Applicant then
            user.role ← A;
        else if certificate.getRole() = Role.Examiner then
            user.role ← E;
        else
            return ERROR;
        end if
        UMD.create(user);
        return user.uid;
    end if
else
    return ERROR;
end if
```

Algorithm 2 Verify

Input: *uid*, *name*, *role*, *certificate*
Output: Yes, No or Insufficient_Information

```
users ← UMD.getUser(uid, name, role, certificate);
if users is NULL then
    return No;
else if users.size() > 1 then
    return Insufficient_Information;
else
    return Yes;
end if
```

1) SUBMIT

See Algorithm 3. Given *patentName*, *inventor*, *dataKey*, *keywords*, *applicantPwd*, and *applicantUid*, the method will generate a unique *pid* and add a record for this new *PA*. This method also used when the applicant modifies the *PA*, while he or she should provide *pid* instead of *patentName* and *inventor*.

The *paState* will not change until the applicant provide another parameter *confirm*. Once confirmed, it will be changed to **Waiting to be received**, **Under preliminary examination** or **Reviewing**.

2) RECEIVE

See Algorithm 4. Given *pid*, *examinerPwd*, and *examinerUid*, the method will search for the *PA* that matches the *pid* and has the state **Waiting to be received**. If succeed, the method will record the *examinerPwd* and *examinerUid* on this record. Then, the *paState* will be changed to **Under preliminary**

TABLE 2. The structure of Patent Management Data (PMD).

pid	*patentName*	*inventor*	*dataKey*	*keywords*	*applicantPwd*	*applicantUid*	*examinerPwd*	*examinerUid*	*paState*
1	Example 1	Alice	QmekAT...	blockchain, storage	a665e4...	100	fdc4fb...	200	Approved
2	Example 2	Betty	QmekBL...	car, wheel	7a27ae...	2120	null	null	Preparing
...	...	...	...	...	...	...	...	...	...

examination. The examiner is able to get the *dataKey* so that he or she can download the material from EDS.

3) PreExam

See Algorithm 5. Given *pid*, *examinerPwd*, and *decision* (**Approve**, **Reject** or **Require supplement**), the method will search for the *PA* that matches the *pid* and *examinerPwd* and has the state **Under preliminary examination**. If succeed, the *paState* will be changed to **Pre-approved**, **Rejected**, or **Need supplement**, depending on the parameter *decision*.

4) DISCLOSE

See Algorithm 6. Given *pid* and *applicantPwd*, the method will search for the *PA* that matches the *pid* and *applicantPwd* and has the state **Pre-approved**. If succeed, the *paState* will be changed to **Reviewing**.

5) REVIEW

See Algorithm 7. Given *pid*, *examinerPwd*, and *decision* (**Approve**, **Reject** or **Require revision**), the method will search for the *PA* that matches the *pid* and *examinerPwd* and has the state **Reviewing**. If succeed, the *paState* will be changed to **Approved**, **Rejected**, or **Need revision**, depending on the parameter *decision*.

6) QUERY

See Algorithm 8. Given enough information, the method will return all the *PA*s which satisfy the information. The parameters can be *pid*, *patentName*, *inventor*, one or more keywords, *paState*, or a combination of the former. To protect privacy, the method will never return passwords. If the patent is not disclosed, *dataKey* will not be returned either.

E. PABC CLIENT

PABC Client works in the application layer. It takes charge of interacting with User, BN, EDS, and ECA, as we have introduced in Section III. As is shown in Fig. 5, PABC Client receives user's requests (such as requesting a certificate from ECA, uploading or downloading material from EDS, sending transactions with method name and parameters defined in UMC or PMC to BN), and returns results to the user.

PABC Client itself do not engage in the progress of blockchain consensus, so users are allowed to use or even develop their own clients, as long as the clients are able to interact with BN, EDS, and ECA with proper roles or APIs. Fig. 5 only shows a simplified implementation of PABC Client. In this example, users do not verify any identities when submit, receive, and review a PA, etc. In practical use, the client can be much more complex to ensure the functionality and security of the system. For instance, an examiner's client may support duplicate checking (by calling *Query()* method), so as not to accept a patent that has been applied for by a former applicant.

Algorithm 3 Submit

Input: *patentName*, *inventor*, *dataKey*, *keywords*, *applicantPwd*, *applicantUid*, *pid*, *confirm*
Output: *pid* or ERROR

```
if pid is NULL then
    PA ← new PatentApplication(patentName, inventor);
    PA.dataKey ← dataKey;
    PA.keywords ← keywords;
    PA.applicantPwd ← applicantPwd;
    PA.applicantUid ← applicantUid;
    PA.paState ← Preparing;
    pid ← PMD.create(PA);
    return pid;
else
    PA ← PMD.getPA(pid, applicantPwd);
    if PA is NULL then
        return ERROR;
    end if
    PA.dataKey ← dataKey;
    PA.keywords ← keywords;
    if confirm then
        if PA.paState = PAState.Preparing then
            PA.paState ← PAState.Waiting_to_be_received;
        else if PA.paState = PAState.Need_supplement then
            PA.paState ← PAState.Under_preliminary_examination;
        else if PA.paState = PAState.Need_revision then
            PA.paState ← PAState.Reviewing;
        else
            return ERROR;
        end if
    end if
    PMD.update(PA);
    return pid;
end if
```

V. EVALUATION

In this section, we evaluate the performance of PABC. Due to the fact that different implementation of PABC Clients (e.g., Java app, Python app, and web app, etc.) may be an interference factor of the evaluation, we only use Hyperledger Caliper as simplified clients. Because PMC is the most

Algorithm 4 Receive

Input: *pid*, *examinerPwd*, *examinerUid*
Output: *dataKey* or ERROR

```
PA ← PMD.getPA(pid);
if PA is NULL then
    return ERROR;
else if PA.paState = PAState.Preparing then
    return ERROR;
else if PA.paState = PAState.Waiting_to_be_received then
    PA.paState ← PAState.Under_preliminary_examination;
    PA.examinerPwd ← examinerPwd;
    PA.examinerUid ← examinerUid;
    PMD.update(PA);
    return PA.dataKey;
else
    if PA.examinerPwd = examinerPwd && PA.examinerUid = examinerUid then
        return PA.dataKey;
    else
        return ERROR;
    end if
end if
```

Algorithm 5 PreExam

Input: *pid*, *examinerPwd*, *decision*
Output: SUCCESS or ERROR

```
PA ← PMD.getPA(pid, examinerPwd);
if PA is NULL then
    return ERROR;
else if PA.paState = PAState.Under_preliminary_examination then
    if decision = Decision.Approve then
        PA.paState ← PAState.Pre_approved;
    else if decision = Decision.Reject then
        PA.paState ← PAState.Rejected;
    else if decision = Decision.Require_supplement then
        PA.paState ← PAState.Need_supplement;
    else
        return ERROR;
    end if
    PMD.update(PA);
    return SUCCESS;
else
    return ERROR;
end if
```

Algorithm 6 Disclose

Input: *pid*, *applicantPwd*
Output: SUCCESS or ERROR

```
PA ← PMD.getPA(pid, applicantPwd);
if PA is NULL then
    return ERROR;
else if PA.paState = PAState.Pre_approved then
    PA.paState ← PAState.Reviewing;
    PMD.update(PA);
    return SUCCESS;
else
    return ERROR;
end if
```

frequently used contract, we perform tests on PMC. First, the environment will be introduced. After that, we run a benchmark of transactions defined in PMC and analyze their performance.

A. ENVIRONMENT

In order to get closer to reality, the Fabric network was constructed with 7 nodes (including 5 peer nodes P1-P5 and 2 orderers O1-O2) managed by 5 patent offices (PO1-PO5, corresponding to the five largest IP offices in the world), as is shown in Fig. 6. Both PO1 and PO2 manage a peer node and an orderer, while PO3, PO4, and PO5 manage only a peer node respectively. ECAs were installed in all the nodes. The network was running on 5 computers, representing PO1 to PO5. These computers all have 8GB RAM, 4-Core CPU, 128GB SSD, and 100 Mbps network connection. We did not deploy IPFS nodes as our EDS because the Fabric network does not store or read data from EDS directly, which means the performance of EDS has no effect on the Fabric network.

Hyperledger Caliper is used to run the benchmark. Caliper is a blockchain benchmark tool designed to allow users to measure the performance of a specific blockchain implementation [13]. Caliper was installed on another computer (not shown in Fig. 6). Once Caliper starts working, it will continuously send transactions to all the peer nodes in the network according to the configure file. In other words, Caliper plays the role of several clients. When the benchmark ends, a report will be generated.

B. THROUGHPUT AND LATENCY WHEN SEND RATE VARIES

In order to evaluate the PABC's concurrent processing capacity, we measured the throughput R and latency t_L when send rate R_s varies. We run 6 rounds, with R_s from 10 up to 60 TPS (Transactions Per Second) respectively. In each round, 6 methods defined in PMC were tested, including *Submit*, *Receive*, *PreExam*, *Disclose*, *Review* and *Query*. Each method was called 300 times per round, and we calculated the average throughput.

As is shown in Fig. 7, most of the methods can achieve maximum $R = R_S$ when R_s is lower than 30. However, R starts to saturate when R_s keeps on increasing and reaches the maximum (about 35 TPS). Besides, As in Fig. 8, the latency is at a low level (less than 1 second) when R_s is lower than maximum throughput. If R_s is higher than 35 TPS, the latency t_L becomes much longer, which means PABC needs more

Algorithm 7 Review

Input: *pid*, *examinerPwd*, *decision*
Output: SUCCESS or ERROR

```
PA ← PMD.getPA(pid, examinerPwd);
if PA is NULL then
    return ERROR;
else if PA.paState = PAState.Reviewing then
    if decision = Decision.Approve then
        PA.paState ← PAState.Approved;
    else if decision = Decision.Reject then
        PA.paState ← PAState.Rejected;
    else if decision = Decision.Require_revision then
        PA.paState ← PAState.Need_revision;
    else
        return ERROR;
    end if
    PMD.update(PA);
    return SUCCESS;
else
    return ERROR;
end if
```

Algorithm 8 Query

Input: *pid*, *patentName*, *inventor*, *keywords*, *paState*
Output: *PAs*

```
PAs ← PMD.getPA(pid, patentName, inventor, keywords, paState);
if PAs is NULL then
    return NULL;
else
    for PA in PAs do
        PA.applicantPwd ← NULL;
        PA.examinerPwd ← NULL;
        if !(PA.paState = PAState.Reviewing || PA.paState = PAState.Need_revision || PA.paState = PAState.Approved) then
            PA.datakey ← NULL;
        end if
    end for
    return PAs;
end if
```

time to deal with transactions if R_s is beyond the processing capacity.

Unlike the other 5 methods, the method *Query* can achieve R more than 50 when R_s is 60, and keep a much low latency t_L (0.04 seconds), because *Query* does not make any changes on databases and does not need to be executed and validated by all peers. Each peer dependently runs *Query* and returns patent information to clients.

C. LATENCY AND FAILURE RATE WITH HIGH SEND RATE

The previous sub-section has discussed about the throughput and the latency of PABC. In this sub-section, we evaluate

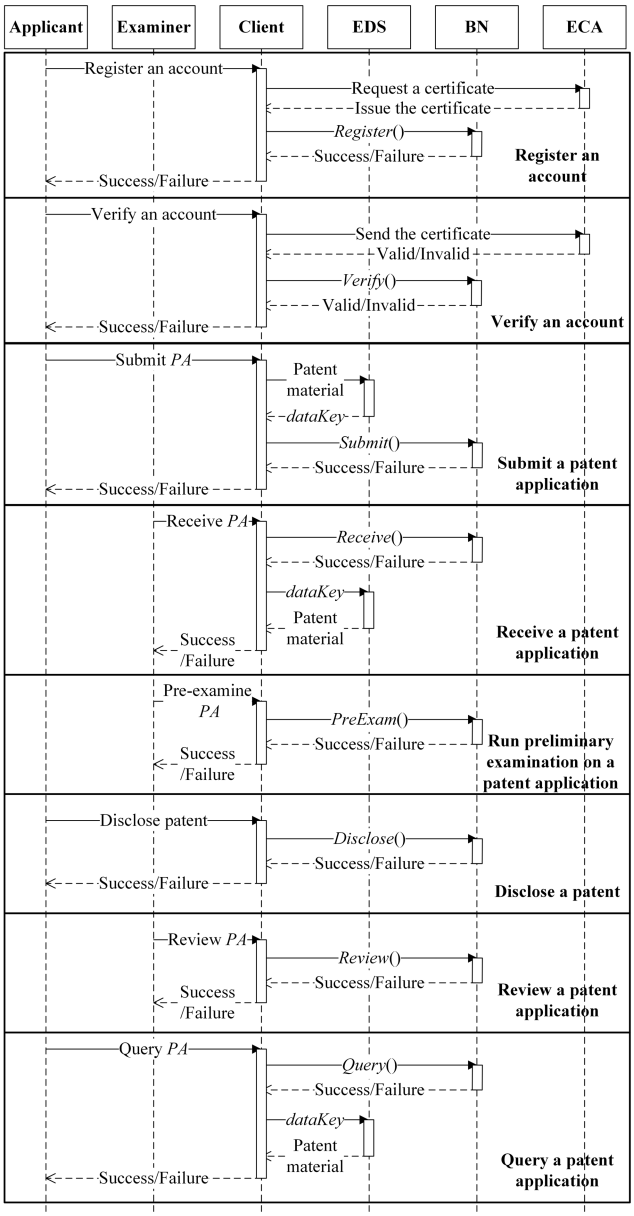


FIGURE 5. Sequence diagram of PABC.

the stability of the system. The latency t_L and the failure rate F were measured with a fixed high R_s (60 TPS) and a send duration t_d varying from 20 seconds to 60 seconds. F is defined as timed-out transactions divided by all the transactions, and we set the threshold of the transaction executing time t_h to 70 seconds.

As is shown in Fig. 9 and Fig. 10, when t_d increases, t_L and F rise. We can predict that if clients send transactions at a high rate with a longer duration, PABC will not work properly. According to this and previous experiments, it is recommended that the system should work at the R_s of 35 or lower.

Due to the same reason that has been mentioned in part B, *Query* has low t_L and F at the R_s of up to 60.

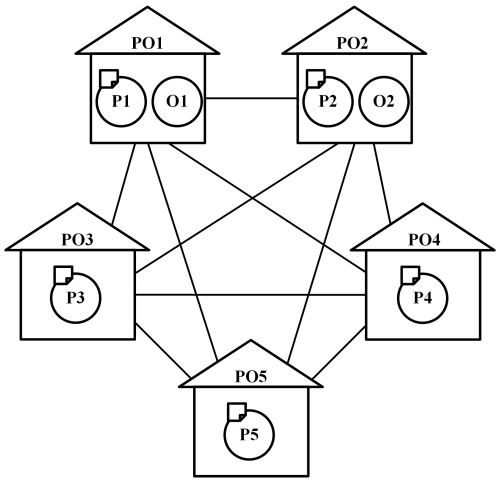


FIGURE 6. Topology of PABC experimental network.

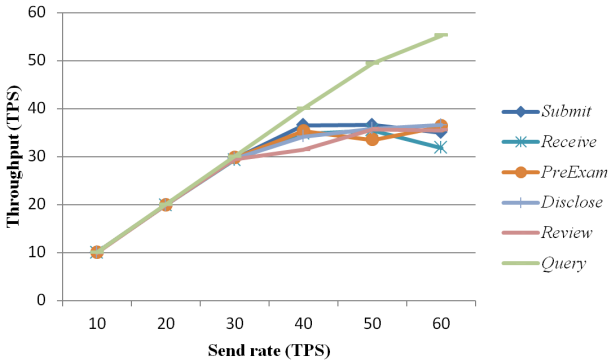


FIGURE 7. Throughput (TPS) when send rate varies.

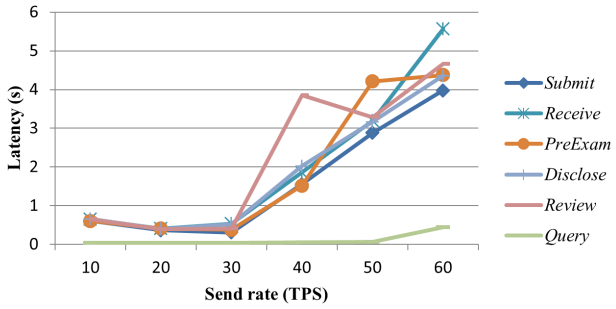


FIGURE 8. Latency (seconds) when send rate varies.

Actually, we can predict the latency t_L and failure rate F of the system with varying R_s and t_d, when R_s is higher than the maximum throughput.

Given the maximum throughput R_{max}, send duration t_d, and send rate R_s, we can find that the minimum latency

$$t_{Lmin} = \frac{R_s t_d}{R_{max}} - t_d. \tag{1}$$

However, (1) is workable if t_{Lmin} is lower than the threshold t_h. The transaction will fail if the latency of it reaches t_h.

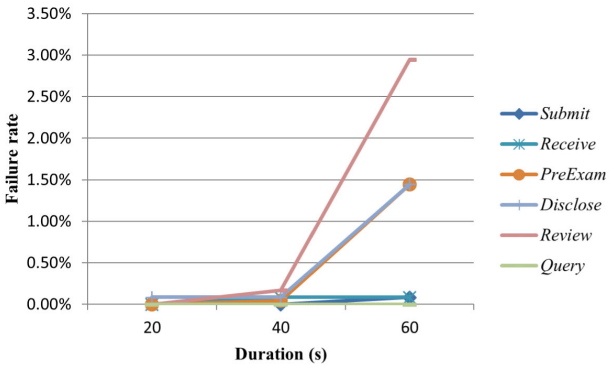


FIGURE 9. Failure rate when duration varies.

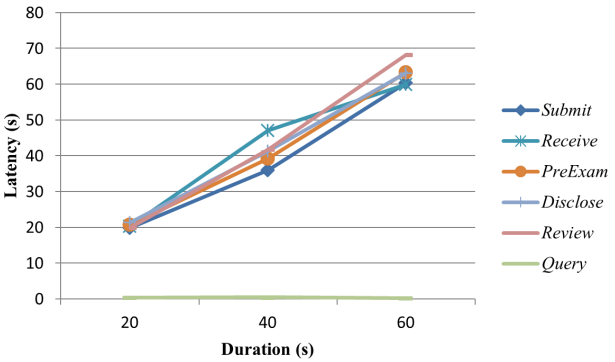


FIGURE 10. Latency (seconds) when duration varies.

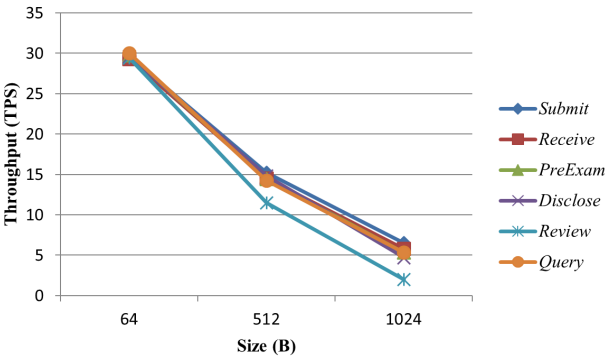


FIGURE 11. Throughput when data size varies.

We can also calculate the failure rate

$$F = max\{0, 1 - \frac{R_{max}(t_d + t_h)}{R_s t_d}\}. \tag{2}$$

We can see in (2) that $F = 0$ when

$$t_d \leq \frac{R_{max} t_h}{R_s - R_{max}}. \tag{3}$$

If t_d gets higher, the failure rate will rise. Therefore, a break out of R_s is permitted, but PABC can work properly only for a very short time according to the above analysis.

D. THROUGHPUT WHEN DATA SIZE VARIES

We have mentioned in Section III that EDS is used to store patent material for higher performance. If all the material, instead of the *dataKey*, is stored in the PMD, what will happen? To answer this question, we measured the throughput with varying data sizes. 3 random strings, of which the

sizes are 64 Bytes, 512 Bytes, and 1024 Bytes, respectively, were filled into each field *datakey* in the ledger. Caliper sent transactions at 30 TPS for 10 seconds.

As is shown in Fig. 11, when the data size becomes larger, the throughput of PABC decreases. When the data size increases to 1024 Bytes, the throughput drops to one-sixth of that when the data size is 64 Bytes. In reality, the patent material is much larger than 1024 Bytes. If all the material is stored in the ledger, PABC will work with a very low throughput. That is why the material (several KB to MB) should be stored in an EDS and the hash value (50 to hundreds of Bytes) should be stored in the ledger.

VI. CONCLUSION

In this paper, we propose a patent application system based on blockchain called PABC. We build the system model and then implement PABC on Hyperledger Fabric. We further evaluate the throughput, latency, and failure rate of PABC in different conditions, and find that PABC can run stably at a send rate of not more than 35 TPS, which can satisfy the current requirement of international patent application. Compared with existing patent application systems applied in different patent offices, our proposed system connects all patent offices in the world. All the patent data are shared among all patent offices, with high security supported by blockchain. Thus, it is possible to realize POVA.

However, the experimental results also show that the state database limits the maximum throughput and average latency. We will further do some research on the performance of the database used in blockchain.

REFERENCES

[1] S. P. Ladas, *Trademarks, and Related Rights: National and International Protection*, vol. 1. Cambridge, U.K.: Harvard Univ., 1975, pp. 6–7.

[2] Fields of Intellectual Property Protection, *WIPO Intellectual Property Handbook*, 2nd ed. Geneva, Switzerland: WIPO, 2004, p. 17. [Online]. Available: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_489.pdf

[3] *WIPO IP Statistics Data Center*. Accessed: Jul. 31, 2020. [Online]. Available: https://www3.wipo.int/ipstats/index.htm

[4] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] G. Wood. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. [Online]. Available: http://ethereum.github.io/yellowpaper/paper.pdf

[6] *Hyperledger Fabric*. Accessed: May 15, 2020. [Online]. Available: https://github.com/hyperledger/fabric

[7] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jun. 2019, doi: 10.1109/TPDS.2018.2881735.

[8] S. He, W. Ren, T. Zhu, and K.-K.-R. Choo, "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 948–959, Feb. 2020, doi: 10.1109/JIOT.2019.2947339.

[9] Q. Tao, X. Cui, X. Huang, A. M. Leigh, and H. Gu, "Food safety supervision system based on hierarchical multi-domain blockchain network," *IEEE Access*, vol. 7, pp. 51817–51826, 2019, doi: 10.1109/ACCESS.2019.2911265.

[10] W. Liang, X. Lei, K.-C. Li, Y. Fan, and J. Cai, "A dual-chain digital copyright registration and transaction system based on blockchain technology," in *Proc. BlockSys*, 2019, pp. 702–714, doi: 10.1007/978-981-15-2777-7_57.

[11] Y. Ouyang, X. Zheng, X. Lu, L. Xiaowei, and S. Zhang, "Copyright protection application based on blockchain technology," in *Proc. IEEE Intl Conf Parallel Distrib. Process. with Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Xiamen, China, Dec. 2019, pp. 1271–1274, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00182.

[12] J. Wang, S. Wang, J. Guo, Y. Du, S. Cheng, and X. Li, "A summary of research on blockchain in the field of intellectual property," *Procedia Comput. Sci.*, vol. 147, pp. 191–197, 2019, doi: 10.1016/j.procs.2019.01.220.

[13] *Hyperledger Caliper*. Accessed: Aug. 3, 2020. [Online]. Available: https://hyperledger.github.io/caliper

[14] The Concept of Intellectual Property, *WIPO Intellectual Property Handbook* 2 nd ed. Geneva, Switzerland: WIPO, 2004, p. 3. [Online]. Available: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_489.pdf

[15] D. Poticha and M. W. Duncan, "Intellectual property—The foundation of innovation: A scientist's guide to intellectual property," *J. Mass Spectrometry*, vol. 54, no. 3, pp. 288–300, Mar. 2019, doi: 10.1002/jms.4331.

[16] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[17] B. Juan. (2014). *IPFS—Content Addressed, Versioned, P2P File System (DRAFT 3)*. [Online]. Available: https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf

[18] G. J. Showkatramani, N. Khatri, A. Landicho, and D. Layog, "A secure permissioned blockchain based system for trademarks," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPCON)*, Newark, CA, USA, Apr. 2019, pp. 135–139, doi: 10.1109/DAPPCON.2019.00026.

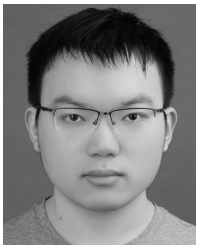
[19] J. Becker, D. Breuker, T. Heide, J. Holler, H. P. Rauer, and R. Bãhme, "Can we afford integrity by proof-of-work? Scenarios inspired by the bitcoin currency," in *The Economics of Information Security and Privacy*. Berlin, Germany: Springer, 2013, pp. 135–156, doi: 10.1007/978-3-642-39498-0_7.

[20] *Litecoin*. Accessed: Nov. 12, 2020. [Online]. Available: https://litecoin.org/

[21] A. Gervais, G. O. Karame, K. Wäst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 3–16, doi: 10.1145/2976749.2978341.

[22] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, Chicago, IL, USA, May 2017, pp. 1085–1100, doi: 10.1145/3035918.3064033.

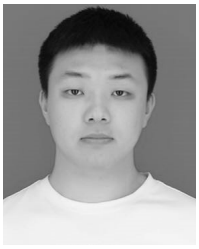
[23] *Top Cryptocurrency Prices*. Accessed: Nov. 12, 2020. [Online]. Available: https://www.blockchain.com/prices



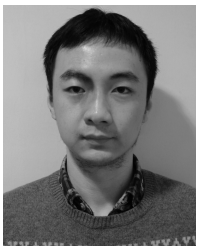
SHUYU BIAN received the B.S. degree in electrical engineering and its automation from the Nanhang Jincheng College. He is currently pursuing the M.S. degree with the Computer Science Department, Nanjing University of Aeronautics and Astronautics, Nanjing. His current research interests include blockchain, the Internet of Things, and privacy preservation.

GUOHUA SHEN received the M.S. and Ph.D. degrees in computer science from the Nanjing University of Aeronautics and Astronautics, China. He is currently an Associate Professor with the College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics. His research interests include requirement traceability, fog computing, description logic, semantic web, and web services and ontology.

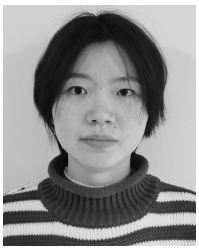
ZHIQIU HUANG received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics. He is currently a Professor with the College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics. He is also the Director of software safety in computer science with the Nanjing University of Aeronautics and Astronautics. His research interests include formal method, cloud computing, edge computing, web security, and privacy preservation.



YANG YANG received the M.S. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, where he is currently pursuing the Ph.D. degree. His current research interests include blockchain, privacy preservation, and access control.



JINGHAN LI received the B.S. degree in information security from Sichuan University. He is currently pursuing the M.S. degree with the Computer Science Department, Nanjing University of Aeronautics and Astronautics, Nanjing. His current research interests include blockchain, access control, and privacy preservation.



XIAOYU ZHANG received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, where she is currently pursuing the M.S. degree with the Computer Science Department. Her research interests include crowdsensing, blockchain, and data privacy.