Blockchain-based Attendance Management and Payroll System using Hyperledger Composer Framework

Delphi Hanggoro, Jauzak Hussaini Windiatmaja, Riri Fitri Sari
Department of Electrical Engineering, Faculty of Engineering
Universitas Indonesia
Depok 16424, Indonesia
delphi.hanggoro@ui.ac.id, jauzak.hussaini@ui.ac.id riri@ui.ac.id

Abstract— **Data storage for attendance at a company or agency is usually stored in a local or cloud database. This type of storage has several issues, such as privacy and data integrity, because several parties will fully regulate privacy and data integrity. Therefore, a data storage system is needed that can provide privacy, security, and data integrity to maintain the authenticity of sensitive data. This study wants to develop to implement blockchain technology to store employee attendance data from the HR department in a company. Employee attendance data is taken from the Angular web application integrated with a permissioned blockchain framework called Hyperledger Composer. This study chose Hyperledger Composer because this type of blockchain has a fast validation time. Hyperledger Composer has a Representational State Transfer Application Programming Interface (REST API) named composer-rest-server, which allows Hyperledger blockchain to interact with other components. The implementation results show that Hyperledger Composer can be functionally used as data storage for the attendance management and Payroll system. In addition, Hyperledger Composer performance is measured by evaluating block transaction times. The evaluation of the Hyperledger Composer is done in three ways: directly within the Hyperledger Composer, using the Angular web application through the REST API, and using JMeter through the REST API. As a result, testing for making transaction blocks varies from 1 - 17ms on an experiment directly in Hyperledger Composer, 5 - 296ms through the REST API using JMeter tools, and 1 - 4270ms through the Angular Web Application. The result shows that the performance of the REST API produced by the composer-rest-server is recorded faster than Ethereum. With these results, composer-rest-server can handle systems that require fast transaction times, such as voting systems, health monitoring, and the Internet of Things (IoT) application, because the average time to conduct transactions is still under one second. Thus, Hyperledger as attendance data storage can provide privacy and data integrity.**

Keywords — blockchain, Hyperledger, composer-rest-server, attendance, REST API.

I. INTRODUCTION

Blockchain is a distributed digital ledger technology that Nowadays, the development and use of technology are developing more rapidly in various fields, one of which is information technology. Information technology is a technology used to get, store, organize, process, and manipulate the amount of data to produce relevant and accurate information. The attendance management and payroll system are some of the information systems on financial activities required in a company or agency. The use of this information system can provide an accurate report of attendance records and payroll. The attendance management and payroll system is a system that processes the attendance information to determine the total salary of each employee. Attendance data is sensitive information that must be maintained for authenticity because it can affect the number of salaries to be paid.

In general, data storage for attendance at a company or agency is in a local or cloud database. Local and cloud database is the right storage for information system data [1]. However, these types of storage have several issues, such as privacy and data integrity [2], because several parties regulate this for its privacy and data integrity. Privacy is the user's right to maintain the confidentiality and control of their information when given to the other parties [3]. At the same time, data integrity is data consistency to ensure that a party does not change the data. Therefore, a data storage system that can provide privacy, security and data integrity is needed to maintain the authenticity of the sensitive data.

This study uses blockchain technology to overcome these issues as an alternative for data storage. Blockchain is a distributed ledger technology that cannot be tampered with or modified from a piece of data so that the data presented will always be safe, integrity maintained, and reliable. Blockchain also has a consensus that makes the blockchain has no centralized authority, and the data management controls are distributed among each member of the blockchain network. Every addition or change of data in the block will be recorded and known by all blockchain members. Therefore, all blockchain members will have the exact copy of data and will increase trust in the network [4]. Storing and distributing data in a blockchain can help maintain the privacy and integrity of user data. Moreover, access control settings on the blockchain can also be applied to maintain data privacy.

As one of the solutions to solve the issue of user data privacy threats in the data management centre in the local and cloud database, this study will create an attendance data storage system that is implemented into a blockchain using Hyperledger. Hyperledger is a permissioned blockchain framework. Hyperledger can perform data validation mechanisms in a shorter time than the current blockchain consensus, such as Proof-of-Work on Bitcoin and Proof-of-Stake on Ethereum [5]. In the mechanism of data storage, Hyperledger can be done directly without waiting for the

978-1-6654-6658-5/22/$31.00 ©2022 IEEE

process of solving mathematical puzzles that take a lot of time. Moreover, Hyperledger is also equipped with a smart contract system called chain-code, which briefly is a function that will be executed if several conditions are met. Hyperledger provides several tools that support the implementation of this study. Hyperledger Fabric is used as a runtime blockchain, Hyperledger Composer as a system interface, and composer-rest-server as an API service provider.

This paper will propose an attendance management and payroll system using Hyperledger, including the discussion of Hyperledger, REST API, and web applications design. The goal is to save attendance data into a Hyperledger blockchain through an angular web application. Our contributions are summarized as follows:

- Integrate the Hyperledger blockchain with the Angular web application via the composer-rest-server-generated REST API.
- Provides update attendance and payroll functions on Hyperledger composer business networks, both of these functions can be performed on angular web applications.
- Evaluate the REST API's performance generated by the composer-rest-server for block transactions directly within the Hyperledger Composer, using JMeter through the REST API, and using the Angular web application through the REST API.

The structure of this paper is as follows. Section II will explain and discuss the related works. Next is the explanation of the design from Hyperledger, REST API, and Angular web application in section III. The results and discussion are in section IV. Last but not least, the conclusions in section V.

II. Related Works

This section explains previous research on attendance management and the use of blockchain for storage solutions.

In 2008 Lim et al. [6] examined radio frequency identification (RFID) for the attendance data collection process. RFID was previously used to identify and trace markers embedded into living beings. The system consists of an RFID reader and tag, the RFID reader functions as a translator of the identity in the tag. This system no longer needs to write the name and record the arrival time because it has been recorded when the RFID reader scans the tag. Their system has been recorded in real-time and is also equipped with the display of attendance information by connecting the RFID reader to the computer via Universal Serial Bus (USB).

Furthermore, in 2014 Arbain et al. [7] conducted research that developed an attendance system using RFID for student attendance lists. They developed this system into a web-based system so that all stakeholders can access the system anywhere online. They used the Serial UART 13.56MHZ RFID Reader / Writer Module as the primary hardware. Then it is connected by USB to the server computer. For the software, they used Adobe Dreamweaver and Adobe Fireworks to develop the web and used MySQL for the database. Their system has three authentications with different permissions: admin, staff, and students. Their system succeeds in recording the attendance using RFID and storing it in the database from the test results.

S. Chintalapti et al. [8] developed an attendance system using face detection and recognition to record attendance into the database. Wagh et al. [9] also developed the same system but used a different algorithm. Badejo et al. [10] developed an attendance recording system using fingerprints, which will then be stored in a database. M.M. Islam [11] developed an attendance recording system by marking all Employee smartphones that can access the system online to record attendance. After that, the data will be stored on the database server.

H. Ardina et al. [12] assumed that previous studies [8-11] related to attendance only focused on developing methods to secure attendance authentication so that it could not be manipulated. Furthermore, the system used is categorized as a centralized system that stores data in a central database. Storing databases on a centralized system is easy to manipulate because log entries can be changed. Based on these reasons, Ardina et al. [12] have an idea to develop a blockchain-based attendance system design to become an alternative for data manipulation solutions.

Ayoade et al. [1] have researched integrating blockchain technology with the Internet of Things technology. Their research applies a smart contract using the Ethereum blockchain to manage access rights in managing IoT data. They succeed in managing access rights and recommend using permissioned blockchain to save time to make the system more efficient. Furthermore, Yang et al. [13] have successfully implemented data exchange on smart toys using the Hyperledger blockchain platform to ensure data integrity and consistency.

With the background of the mentioned research, this study wants to develop the design of Ardina et al. [12] using the permissioned blockchain that has been recommended by Ayoade et al. [1] and used by Yang et al. [13].

III. System Design

This section will explain the system's design, which is divided into three parts, e.g. Hyperledger, REST API, and Angular web applications.

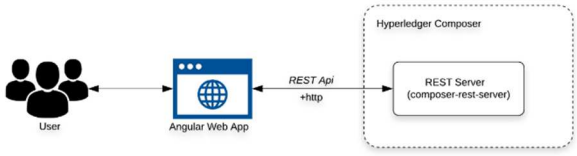

Fig. 1. Design of attendance and payroll systems.

Figure 1 shows an overview of blockchain-based attendance and payroll systems. Managers and Employees can access the system through the Angular web application. Then when the user takes action, all data will be taken or given to the blockchain network using the REST API service via the HTTP protocol so that the data content presented in the Angular web application will always be the same as the data in the blockchain. Two main entities will take action on the Angular web application, Manager and Employee. Managers are actors who can manage all data and pay an employee's salary, while Employees are actors who attend and take absences every day.

1) Hyperledger Composer

The blockchain design in this solution uses one of the popular blockchain infrastructures named Hyperledger. However, this design can be applied to another blockchain framework with a few modifications. The designed blockchain includes three basic elements and permission control rules. The three basic elements are participants, assets, and transactions. Permission control determines who has access to specific resources under certain conditions.

a) Determination of Participants

The determination of participants is based on the needs of actors in the system architecture. Table 1 shows a list of participants and their involved contents in the Attendance and Payroll systems network.

TABLE I. PARTICIPANT LIST

No	*Participant*	*Value*
1	Manager	managerId, name, walletId
2	Employee	employeeId, name, presenceId, walletId, contractId

There are two participants in this system, Manager and Employee. The manager has managerId, name, which is the username and password when logging in on the web page. Then walletId is an id that is connected to the Wallet asset, Wallet in the Manager is used to make `fundFill` transactions. The employee has an employeeId, which becomes the username when performing the `updatePresence` function on the web page. presenceId is an id connected to the Presence asset, walletId is an id connected to a Wallet asset, contractId is an id connected to Contract asset. Managers need data from Employee to be able to execute `employeeSalary` functions.

b) Determination of Asset

Assets are everything that has value and contains data from the transaction process carried out. Table 2 shows the list of assets and contents needed in this system.

TABLE II. ASSET LIST

No	*Asset*	*Value*
1	Fund	fundId, value
2	Presence	presenceId, latestPresence, totalPresence
3	Wallet	walletId, value, latestPaid
4	Contract	contractId, minPresence, mainSalary, supportSalary, salary cut

- **Fund**
  Fund is a place to hold money intended to make it easier for the company to oversee the payroll cycle. Usually, if the employee has different levels, the funds used are also different.

  - fundId: the fund identity to pay employees.
  - value : available money (value)

- **Presence**
  Presence (Attendance) is a place to store attendance data, this asset can be updated through the updatePresence function that employees do every day.

  - presenceId: the Presence identity, which is connected with the Employee participant. Each asset has an id to be called in a transaction.
  - latestPresence: timestamp (date) in the Presence data that indicates when the Presence was last updated, updatePresence function can only be done once on the same date.
  - totalPresence: the amount of attendance that has been done.

- **Wallet**
  Wallet is a place of money (value) from participants. Each participant has a Wallet, the system needs Manager to fill the Fund, and the Employees to get the salary.

  - walletId: the wallet identity that is connected with participant Managers and Employees.
  - value: the available money (value).
  - latestPaid: timestamp (date) in the Wallet data for employee participants indicates when the employee was last paid because employeeSalary functions can only be done in different months.

- **Contract**
  Contract is an asset that determines how much the employee's salary. The Contract can be changed according to the agreement.

  - contractId: the contract identity owned by the employee.
  - minPresence: a minimum number of total Presence so as not to be penalized in the payroll process.
  - mainSalary: the amount of basic salary that is not added to the value of salary Support and salaryCuts.
  - supportSalary: the additional money if attendance is more than minPresence.
  - salaryCuts: the amount of deducted money if attendance is less than minPresence.

c) Determination of Transaction

The transaction is a carried-out process by participants to change the contents of an asset. Table 3 shows a list of transactions carried out in this system.

TABLE III. TRANSACTION LIST

No	*Transaction*	*Participant / Asset*	*Explanation*
1	fundFill	Manager, Wallet	Add value from the Fund that used to pay the Employee
2	updatePresence	Employee, Presence	Update the Presence data
3	employeeSalary	Employee, Fund, Wallet, Presence, Contract	Transferring Funds to Funds to the Employees

Transactions in our system are carried out by participants who need data from certain assets according to their authority. This transaction process aims to input asset data in the blockchain system.

- `FundFill` transaction is a transaction to move the money in the Wallet Manager into the intended Fund. `FundFill` transactions require data from Wallet and Fund.

- `updatePresence` is a transaction to add one of the Presence numbers from an Employee. Employees can only perform this function once a day. `updatePresence` transactions require data from Employees and Wallets.

Algorithm 1. UpdatePresence function

```
1  FUNCTION employeePresence(pres):
2    getParticipant(payroll.Employee);
3    getAsset(payroll.Presence);
```

```
4   IF (pres.employee.employeeId exist in
    employeeRegistry):
5     employeeExists <- true
6   END IF
7   IF (employeeExists == TRUE):
8     (presenceRegistry.totalPresence <-
       presenceRegistry.totalPresence + 1)
9   ELSE
10    error <- 'Employee does not exist'
11  END FUNCTION
```

Algorithm 1 will call participant employee and asset presence. If the employee is not registered, there will be shown the error message "Employee does not exist." If the employee is registered, the function will change the total current presence + 1 to increase, as shown in Algorithm 1, line 8.

- `employeeSalary` is a transaction to pay employees. The amount of the salary is determined by the number of attendance and Contract that has been determined. employeeSalary transactions require data from Employees, Funds, Wallet, Attendance, and Contracts.

```
Algorithm 2. employeeSalary function
1   FUNCTION
2     getParticipant Employee
3     getAsset Presence & Wallet
    IF (es.employee.employeeId exist in
4   employeeRegistry):
      employeeExists <- true
5   END IF
6   IF (employeeExists == TRUE):
7    IF (es.employee.status == 'UNPAID'):
8     IF (es.presence.totalPresence -
          es.contract.minPresence) < 0:
9       penalty = absolute(es.totalPresence -
        es.contract.minPresence)
10    ELSE
        penalty = 0
11    END IF
12  thisMonthSalary = es.contract.mainSalary +
13        (es.contract.supportSalary *
14         es.presence.totalPresence) -
          (es.contract.salaryCuts * penalty)
     IF (thisMonthSalary < es.fund.value):
      es.employee.wallet.value <-
      es.employee.wallet.value +
15    thisMonthSalary;
16    es.fund.value <- es.fund.value -
      es.employee.salary;
      es.employee.status = 'PAID'
17      walletRegistry <- es.employee.wallet
              fundRegistry <- es.fund
18        employeeRegistry <- es.employee
19                    ELSE
20    error <- 'Insufficient balance in funds'
21    END IF
22  ELSE
23    error <- 'You are already paid. Try next
      month!'
24  END IF
25  Else
26    error <- 'Employee does not exist.'
    END FUNCTION
```

`employeeSalary` transaction uses existing contracts on assets to determine employees' salary based on presence, which is shown in Algorithm 2 lines 8-10.

d) Determination of Access Control

Permission control rules aim to control all resources in the network. Access rights are given based on the needs of activities that have been designed on the system. Table 4 shows a list of access rights of each participant.

TABLE IV. ACCESS CONTROL LIST

No	*Participant*	*Wallet*	*Presence*	*Contract*	*Fund*
1	Manager	CRUD	CRD	CRUD	CRUD
2	Employee	R	RU	-	-

Note:
C = Create
R = Read
U = Update
D = Delete

2) REST API

Hyperledger's composer-rest-server will generate the REST API. Representational State Transfer Application Programming Interface (REST API) can be used to service events from business systems that have been created and then display them to other environments/components through API services. The design of the mechanism flow of the interaction between the blockchain and the web application is divided into two. The first is taking data from the blockchain to a web application. The second is storing data from web application inputs to the blockchain.

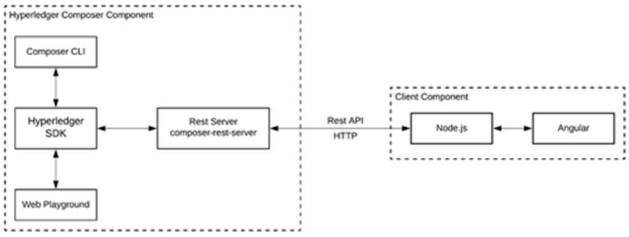

Fig. 2. Hyperledger System Design.

Figure 2 shows the REST API connects the Hyperledger component with Angular web applications. REST API is used to change the business network model to the Open API Definition. When it is running, it can create, read, update, and delete asset and participant data and allow submissions to be processed or taken. These components can interact using the REST API through the HTTP protocol.

3) Angular Web Application

This web design starts from creating a web framework compatible with the existing business network in the Hyperledger Composer and connects the web to the Hyperledger blockchain network using the Rest API via HTTP protocol.

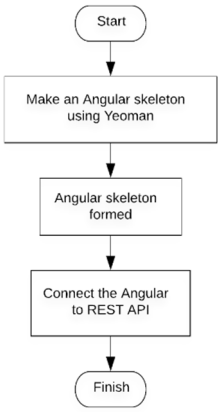

Fig. 3. Angular Web Application making process.

Figure 3 explains the process and relation between Angular web applications and Hyperledger Composer. The design starts by making a framework using Yeoman. Yeoman is a framework code generator for creating Angular web application frameworks that are compatible with the business system in Hyperledger Composer. The framework built by Yeoman can call the REST API that has been generated/produced by the composer-rest-server. So that all activities that are carried out on this web page will be taken and sent to the existing business system in Hyperledger Composer.

IV. Result and Discussion

This study evaluates the performance of the composer-rest-server, which is the gateway for the blockchain to communicate with the Angular web application. To evaluate the performance, this study tests the transaction time into Hyperledger Composer.

The performance test is divided into three scenarios. First, making transactions on the blockchain directly within the Hyperledger Composer. Second, making transactions using the Angular Web using the Google Chrome browser 83.0.4103.97 (64-bit) via the REST API that was created by composer-rest-server. Finally, making transactions using JMeter through REST API created by composer-rest-server. Each scenario will complete 100 transactions to evaluate and compare. The test was carried out on the Ubuntu 18.04 operating system installed on the Oracle VM Virtual Box with six-core processors and eight GB of memory. The VM is installed on a laptop device with I7-9th Gen 2.60 GHz and 32 GB of memory.

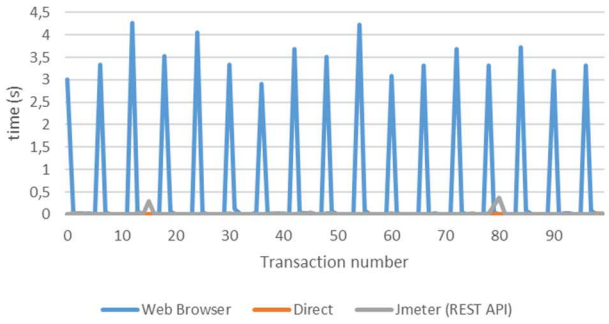

Fig. 4. Transaction time comparison

Figure 4 shows a comparison graph when making transactions into the blockchain system directly from the Hyperledger (direct) system, uses the Angular web application through the REST API (web browser), and uses JMeter through the REST API (JMeter). The time required to make transactions varies between 0.001 - 0.017 seconds for making transactions directly, 0.005 - 0.296 seconds for making transactions using JMeter, and 0.005 - 4.27 seconds for making transactions using a web browser. Transmission via REST API directly using JMeter takes three to four times compared to making a transaction directly. While making transactions using a web browser has quite a fluctuating transaction time, this is because the HTTP protocol limits the number of connections to the same domain. Most modern browsers allow six connections per domain, whereas most older browsers only allow two connections per domain. So the HTTP protocol buffers data that will enter the blockchain system every six transactions.

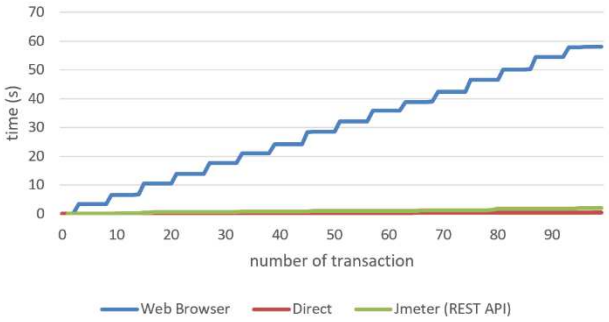

Fig. 5. Total time transaction comparison.

Figure 5 shows a comparison chart of the timestamp (total time) of transactions into the blockchain system. The total time needed to make 100 transactions takes 0.507 seconds for making transactions directly, 1,985 seconds for making transactions using JMeter via REST API, and 58,039 seconds for making transactions using a web browser via REST API. The performance of the composer-rest-server is recorded in a fast time with an average time of 0.198 seconds directly and 0.580 seconds through a web browser for each transaction compared to the Ethereum blockchain, which takes 14 seconds for each transaction [5]. Because the average time to make transactions is still under one second, however, it should be noted that the testing of this system is still in one local network, so the transaction speed can be different when applied in the real world.

V. Conclusion

The blockchain attendance management and payroll system design can be well implemented. All functions of the Hyperledger Composer component and the Angular web application are running properly and according to the design. Hyperledger Composer can communicate with the other components through the REST API provided by the composer-rest-server. The REST API provided by composer-rest-server performs well through JMeter with an average transaction time of 198ms and the total time required for 100 transactions of 1.98 seconds compared to Ethereum Blockchain. Blockchain technology can support the attendance and payroll system, so this system has advantages in data security and data integrity.

ACKNOWLEDGMENT

This work is supported by the Indonesian Government Scholarship PMDSU Grant number NKB-3046/UN2.RST/HKP.05.00/2020 from the Ministry of Research, Technology, and Higher Education (Kemristekdikti).

REFERENCES

[1] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized IoT data management using blockchain and trusted execution environment," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, 2018: IEEE, pp. 15-22.

[2] E. Androulaki *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1-15.

[3] M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014*, 2014: IEEE, pp. 414-419.

[4] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *Journal of Network and Computer Applications,* vol. 125, pp. 251-279, 2019.

[5] G. Ramezan and C. Leung, "A blockchain-based contractual routing protocol for the internet of things using smart contracts," *Wireless Communications and Mobile Computing,* vol. 2018, 2018.

[6] T. Lim, S. Sim, and M. Mansor, "RFID based attendance system," in *2009 IEEE Symposium on Industrial Electronics & Applications*, 2009, vol. 2: IEEE, pp. 778-782.

[7] N. Arbain, N. F. Nordin, N. M. Isa, and S. Saaidin, "LAS: Web-based laboratory attendance system by integrating RFID-ARDUINO technology," in *2014 2nd International Conference on Electrical, Electronics and System Engineering (ICEESE)*, 2014: IEEE, pp. 89-94.

[8] S. Chintalapati and M. Raghunadh, "Automated attendance management system based on face recognition algorithms," in *2013 IEEE International Conference on Computational Intelligence and Computing Research*, 2013: IEEE, pp. 1-5.

[9] P. Wagh, R. Thakare, J. Chaudhari, and S. Patil, "Attendance system based on face recognition using eigen face and PCA algorithms," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015: IEEE, pp. 303-308.

[10] J. A. Badejo, C. C. Eke, S. I. Popoola, T. O. Odu, and A. A. Atayero, "Integrating Automated Fingerprint-based Attendance into a University Portal System," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2017: IEEE, pp. 1016-1020.

[11] M. A. Islam and S. Madria, "A Permissioned Blockchain Based Access Control System for IOT," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 14-17 July 2019 2019, pp. 469-476, doi: 10.1109/Blockchain.2019.00071.

[12] H. Ardina and I. G. B. B. Nugraha, "Design of A Blockchain-based Employee Attendance System," in *2019 International Conference on ICT for Smart Society (ICISS)*, 2019, vol. 7: IEEE, pp. 1-4.

[13] J. Yang, Z. Lu, and J. Wu, "Smart-toy-edge-computing-oriented data exchange based on blockchain," *Journal of Systems Architecture,* vol. 87, pp. 36-48, 2018.