DEVELOPING A BLOCKCHAIN-BASED DECENTRALIZED APPLICATION FOR VEHICLE TRACKING

[1]VEDAT COSKUN, [2]BUSRA OZDENIZCI KOSE, [3]MIKAIL ABDULLAH ORAL, [4]OZGUR COSKUN

Istanbul Atlas University , Gebze Technical University
Turkcell Technology, Istanbul Atlas University
E-mail: [1]vedat.coskun@atlas.edu.tr, [2]busraozdenizci@gtu.edu.tr, [3]mikail.oral@turkcell.com.tr, [4]ozgur.coskun@atlas.edu.tr

Abstract - In today's fast-paced and interconnected world, reliable vehicle tracking systems are essential for efficient management and monitoring. Blockchain technology has emerged as a powerful tool for enhancing data security and transparency across various applications. This study presents the development of BC-4Track, an advanced vehicle tracking system that leverages Ethereum blockchain technology and modern web development frameworks to provide a secure, transparent, and decentralized solution. The system integrates a React-based web application with custom hooks to facilitate seamless interaction with the blockchain, and enables users to perform vehicle and driver registration, as well as manage and query accident data. The smart contracts, written in Solidity, automate data interactions and ensure data integrity across the decentralized network. By combining blockchain technology with a user-friendly interface, this research demonstrates how modern web technologies can create a reliable and interactive decentralized application for vehicle tracking, and contributes to improved data security, transparency, and trustworthiness in the transportation industry.

Keywords - Blockchain, Ethereum, Smart Contract, Decentralized Application, Vehicle Tracking, Transactions.

I. INTRODUCTION

In today's fast-paced world, the importance of reliable vehicle tracking systems cannot be overstated. These systems are critical for ensuring the efficient management and monitoring of vehicles in various industries, including logistics, transportation, and personal vehicle management [1-4]. Reliable vehicle tracking enables real-time monitoring, which helps in enhancing operational efficiency, reducing costs, and ensuring the safety and security of vehicles.

However, the current vehicle tracking systems are predominantly centralized, which brings about several limitation [5-7]. Centralized systems are susceptible to single points of failure. If the central server fails, the entire tracking system becomes inoperative, leading to potential loss of critical tracking data and operational downtime. Additionally, centralized databases are attractive targets for cyberattacks. Unauthorized access to these systems can lead to data breaches, compromising sensitive information about vehicle locations and usage patterns. The centralized nature of these systems often results in a lack of transparency. Stakeholders, such as vehicle owners, fleet managers, and insurance companies, may find it difficult to access real-time and accurate information due to the control exerted by the central authority. Furthermore, data manipulation is a significant risk in centralized systems. Since all data is stored in a single repository, it can be easily altered or tampered with, leading to disputes among stakeholders regarding the accuracy and authenticity of the data.

Given these limitations, there is a pressing need for a more secure, transparent, and reliable solution. Blockchain technology offers a promising alternative by providing a decentralized, tamper-proof ledger for recording and sharing vehicle tracking data [5-7]. By leveraging blockchain, it is possible to create a vehicle tracking system that eliminates single points of failure by distributing data across multiple nodes. This enhances security through cryptographic methods that ensure data integrity and prevent unauthorized access. Moreover, blockchain improves transparency by allowing all stakeholders to access real-time, immutable records of vehicle data. It also ensures data authenticity and reduces the risk of manipulation, fostering trust among stakeholders.

This paper aims to present the development issues of an advanced vehicle tracking system called BC-4Track that leverages blockchain technology to address the inherent limitations of existing centralized systems. This innovative solution aims to create a secure, transparent, and reliable platform for vehicle tracking that effectively manages the data and interactions of various stakeholders in the vehicle ecosystem. By utilizing blockchain technology, the proposed system seeks to eliminate single points of failure, ensuring continuous data accessibility and operational resilience. This decentralized approach enhances security through cryptographic methods, safeguarding data integrity and preventing unauthorized access, thereby mitigating the risks associated with centralized databases.

Additionally, the system aims to improve transparency by providing all stakeholders with real-time access to accurate and immutable records of vehicle data, crucial for fostering trust among vehicle owners, car rental companies, insurance companies, and other

authorities. By recording every transaction on a tamper-proof blockchain ledger, the system ensures data authenticity and reliability, and reduces potential disputes over data accuracy. The proposed design will also incorporate smart contracts to automate and enforce agreements and processes within the vehicle tracking ecosystem, facilitate seamless and secure interactions between stakeholders, streamline operations, and ensure compliance with predefined rules and conditions.

The rest of this paper is organized as follows: Section II describes a brief information about proposed system context and architecture. Section III presents the system development issues of BC-4Track model in detail. Finally, the study is concluded, and further work is explained in Section IV.

II. SYSTEM ARCHITECTURE

Blockchain technology as a decentralized, distributed ledger technology, securely records transactions across multiple nodes [8, 9]. It operates on a peer-to-peer network where every participant has access to the entire ledger, ensuring transparency and immutability. This technology is well-suited for applications requiring high levels of trust, security, and data integrity.

Key features of blockchain technology include decentralization, transparency, immutability, and security [8-11]. Decentralization eliminates the need for a central authority, thereby reducing the risk of single points of failure and enhancing system resilience. Transparency ensures that all participants have access to the same data, promoting trust and reducing the likelihood of disputes. Immutability guarantees that once data is recorded, it cannot be altered or deleted, ensuring a permanent and tamper-proof record. Security is maintained through cryptographic techniques that protect data from unauthorized access and ensure that transactions are securely recorded. Together, these features make blockchain a robust and trustworthy solution for various applications, including vehicle tracking systems.

The proposed BC4-Track system leverages blockchain technology to create a secure and transparent vehicle tracking solution, which is described in detail in the previous stage of our research study [7]. As presented in Figure 1, the system aims to address the limitations of centralized tracking systems by providing a decentralized platform where vehicle data can be recorded and accessed by various stakeholders, including vehicle owners, insurance companies, and car rental firms. This research primarily focuses on the development of the blockchain components to ensure data integrity, security, and transparency.
The Smart Kit component, to be installed on vehicles, will facilitate the collection and transfer of vehicle and driving information to the blockchain network via the Internet. Utilizing components such as a Raspberry Pi, the Smart Kit will periodically collect data related to the vehicle. Additionally, it will record sensor data, such as engine temperature and oil level, onto the blockchain network. The software for the Smart Kit will be developed using the Python programming language. This component will be studied and developed at a later stage of the research.

Within this stage of the research, the developed system includes following essential components:

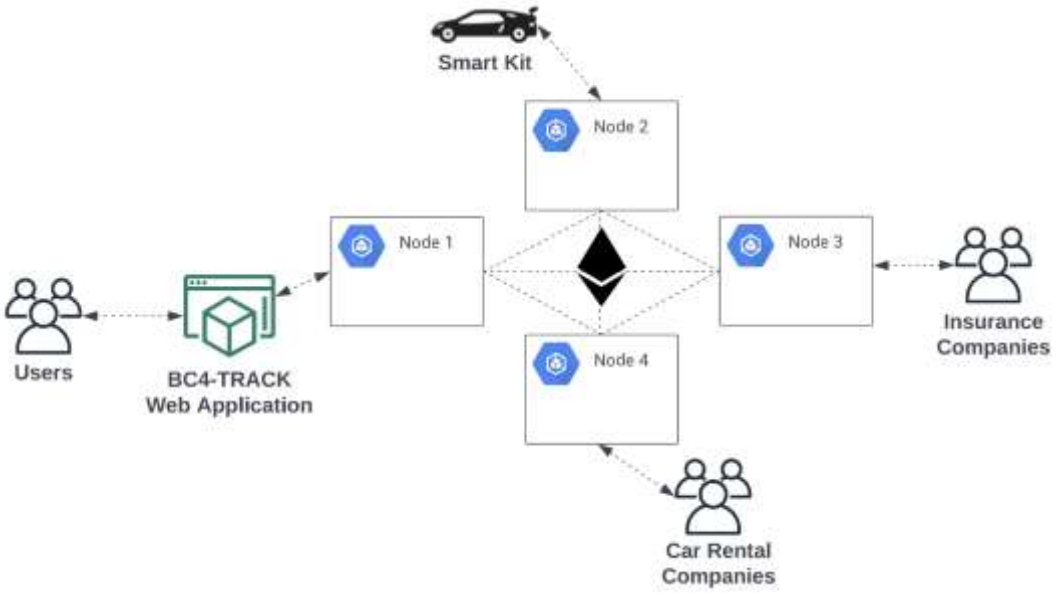


Figure 1: System Context

- Blockchain Network: The blockchain network consists of multiple nodes, distributed across various stakeholders. These nodes maintain the blockchain ledger and validate transactions. Each node represents a stakeholder, such as insurance companies, car rental companies, and vehicle owners. The Ethereum-based blockchain network uses a consensus mechanism to ensure data integrity and prevent tampering. Each node stores a copy of the ledger, ensuring redundancy and resilience against failures.
- Smart Contracts: Smart contracts are developed using Solidity and deployed on the Ethereum blockchain. These contracts automate and enforce rules, ensuring reliable data interactions. They handle the recording and retrieval of vehicle data, automate agreements, and manage transactions between stakeholders without human intervention. The smart contracts manage vehicle data, enforce agreements, and handle transactions between stakeholders without human intervention. They ensure that all actions are recorded immutably on the blockchain, providing a transparent and tamper-proof record of all activities.
- Backend Server: The backend server, developed using .NET Core, acts as an intermediary between the smart kits and the blockchain network. It processes incoming data from the smart kits and writes it to the blockchain. The server ensures secure data transmission and processing, protecting against unauthorized access and data breaches. The backend server ensures secure data transmission and processing, protecting against unauthorized access and data breaches. It manages the interaction between the web application and the blockchain network, facilitating the writing and reading of data to and from the blockchain.
- BC4-Track Web Application as User Interface: The user interface is built using React and other modern web technologies. It provides a user-friendly platform for stakeholders to interact with the system. The web application allows users to enter data, retrieve information, and monitor vehicle status in real-time. It also facilitates interaction with smart contracts and the blockchain network, enabling stakeholders to perform transactions and access vehicle data securely.

The integration of these components ensures that the BC4-Track system is robust and efficient and addresses the key challenges of traditional centralized vehicle tracking systems. By focusing on the development of the blockchain aspects, this research demonstrates the potential of decentralized technology in creating a more reliable and transparent vehicle tracking solution.

III. SYSTEM DEVELOPMENT

The development process begins with detailed planning and design of the system architecture, components, and their interactions, followed by the creation of design documents and diagrams to guide development. Next, smart contracts are written and tested using Solidity, then deployed on the Ethereum blockchain to ensure they function as intended. The backend server is developed using .NET Core to facilitate seamless communication between the smart kits, backend server, and blockchain network. Concurrently, the frontend development involves building the BC4-Track web application using React, focusing on creating an intuitive and user-friendly interface for stakeholders. Rigorous testing is conducted at each stage to ensure functionality, security, and performance, validating the integrity and reliability of the system through various test scenarios. This iterative and feedback-driven approach ensures that the system meets the needs of its users and stakeholders.

A. Development of Smart Contracts

Ethereum as a decentralized platform, enables the execution of smart contracts. It offers a broad range of applications for creating decentralized applications (dApps). Solidity, an object-oriented, high-level programming language designed specifically for Ethereum, is used for developing on this platform. Solidity is utilized to write, compile, and deploy smart contracts, with Remix IDE serving as a web-based development tool for these tasks.

As highlighted, the smart contracts for the BC4-Track system are developed using Solidity. The key components and functionalities of the Solidity code developed for the system include various data structures, functions, and relationships to manage vehicle data securely and transparently. Data structures such as Owner, Driver, Vehicle, Accident, Company, Policy, Service, and Repair form the foundational data types of the smart contract. Each structure contains fields relevant to its category, such as IDs, names, and dates. Separate arrays for each structure type (owners, drivers, vehicles, accidents, companies, policies, services, repairs) are defined to store collections of data within the contract.

```
…
  function contains(string memory what, string memory where)
      private
      pure
      returns (bool)
  {
      bytes memory whatBytes = bytes(what);
      bytes memory whereBytes = bytes(where);
```

```
        require(whereBytes.length >= whatBytes.length);
        bool found = false;
        for (uint256 i = 0; i <= whereBytes.length - whatBytes.length; i++) {
            bool flag = true;
            for (uint256 j = 0; j < whatBytes.length; j++)
                if (whereBytes[i + j] != whatBytes[j]) {
                    flag = false;
                    break;
                }
            if (flag) {
                found = true;
                break;
            }
        }
        return found;
    }
    function setOwner(string memory _ownerName)
        external
    {
        Owner storage owner = owners.push();
        owner.ownerId = Strings.toString(owners.length-1);
        owner.ownerName = _ownerName;
        require(bytes(_ownerName).length > 0,"Null");
        require(
            contains(" ", _ownerName) == false
        ,"Whitespaces");
    }

    function getOwner() external view returns (Owner[] memory) {
        return owners;
    }
    function setDriver(
        string memory _driverName,
        string memory _ownerId
    ) external {
        Driver storage driver = drivers.push();
        driver.driverId = Strings.toString(drivers.length-1);
        driver.driverName = _driverName;
        driver.ownerId=_ownerId;
        require(
            bytes(_driverName).length > 0
        ,"Null");
        require(
            contains(" ", _driverName) == false
        ,"Whitespaces");
    }

    function getDriver() external view returns (Driver[] memory) {
        return drivers;
    }

    function setVehicle(
        string memory _plate,
        string memory _price,
        string memory _ownerId,
        string memory _driverId
    ) external {
        require(owners.length > 0 && drivers.length > 0);
        Vehicle storage vehicle = vehicles.push();
        vehicle.plate = _plate;
        vehicle.price = _price;
        vehicle.ownerId = _ownerId;
        vehicle.driverId = _driverId;
        require(
            bytes(_plate).length > 0 &&
                bytes(_price).length > 0
        ,"Null");
        require(
            contains(" ", _plate) == false &&
                contains(" ", _price) == false
        ,"Whitespaces");
    }

…
```

Program 1: A Sample Excerpt from the Solidity code

For each category, set functions are used to add new data, and get functions are used to retrieve existing data. For example, setOwner adds a new owner, while getOwner lists all owners. The smart contract also includes a contains function, which checks if a string exists within another string, useful for input validation processes. Specific functions have extra controls to ensure data integrity. For instance, the setVehicle function requires at least one owner and driver to be present. Similarly, setPolicy and setRepair functions verify the existence of relevant IDs (vehicles, accidents, etc.) before adding new data. Relationships are established within the smart contract, linking vehicles to their owners and drivers, and associating accidents and services with specific vehicles and policies.

B. Interaction with Ethereum Blockchain

In the context of programming and software development, a hook is a mechanism, or a point of functionality used to intervene, extend, or customize the operation of a program or system. Hooks allow predefined pieces of code (functions) to be executed at specific points in the software, enabling the software to respond to events, be extended, or modified. Hooks increase the modularity and flexibility of the program, enhancing its reusability.

For the BC4-Track project, custom React hooks are developed to facilitate interaction with the Ethereum blockchain. The useConnection and useContract hooks enable users to easily connect to the Ethereum network and interact with smart contracts, thereby allowing the development of secure and efficient decentralized applications.

- Connecting to the Ethereum Wallet: The useConnection hook (Program 2) allows users to connect to the web application using Ethereum wallets like MetaMask. It utilizes the Web3Provider class from the ethers.js library to obtain the Ethereum network provider and the user's wallet address. These details are used within the application once the user successfully connects to the Ethereum network.
- Interacting with Smart Contracts: The useContract hook (Program 3) facilitates interaction with specified smart contracts. The address and ABI (Application Binary Interface) of the smart contract are provided as parameters to this hook. Using the ethers.js library, a contract object is created, allowing the functions of the smart contract to be called and transactions to be executed on the blockchain.

```
import { ethers } from "ethers";
import { useEffect, useState } from "react";

const useConnection = () => {

   const [signer, setSigner] = useState(undefined)
   const [provider, setProvider] = useState("")
   const [address, setAddress] = useState("")
   const [auth, setAuth] = useState(false)
   const [isConnecting, setIsConnecting] = useState(false)
   const connect = async () => {
      if (!window.alert) {
         alert("metamask is not installed!");
         return;
      }
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      try {
         setIsConnecting(true);
         await provider.send("eth_requestAccounts", []);
         const signer = await provider.getSigner();
         const address = await signer.getAddress();
         setSigner(signer);
         setProvider(provider);
         setAddress(address);
         setAuth(true);
         setIsConnecting(false);
      } catch (error) {
         console.log(error);
         setIsConnecting(false);
      }
   }
   return {
      connect,
      signer,
      provider,
      address,
      auth,
      isConnecting
   };
}

export default useConnection;
```

Program 2: useConnnection Hook

```
import { useState, useEffect } from 'react'
import { ethers } from "ethers";

const useContract = (_contractAddress, _contractAbi) => {

   const [contract, setContract] = useState(null);

   useEffect(() => {
      const provider = new ethers.providers.Web3Provider(window.ethereum)
      const singer = provider.getSigner();
      const _contract = new ethers.Contract(_contractAddress, _contractAbi, singer)
      setContract(_contract);
   }, [])

   return contract
}

export default useContract
```

Program 3: useContract Hook

By leveraging these custom React hooks, the BC4-Track system ensures seamless and secure interaction with the Ethereum blockchain. Users can connect their wallets, interact with smart contracts, and perform transactions efficiently, enhancing the overall functionality and user experience of the decentralized application.

C. Interaction with Ethereum Blockchain

The developed BC-4Track web application is a JavaScript-based user interface designed to interact with the blockchain-based vehicle tracking system. This web application supports decentralized application functionality, enabling users to engage with the Ethereum blockchain through a web-based interface (Figure 2). Users can interact with smart contracts, perform transactions, and query blockchain data through this application. Developed using modern web technologies such as React, the BC-4Track web application integrates JavaScript libraries to facilitate communication with the Ethereum network.

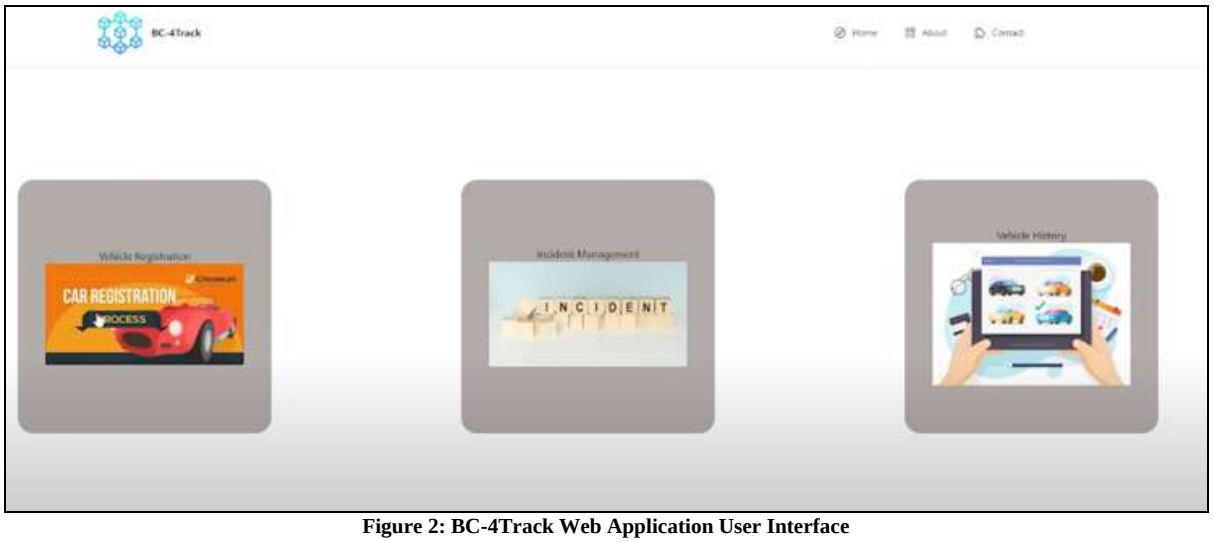


Figure 2: BC-4Track Web Application User Interface

The web application components are designed to manage and display vehicle and driver information retrieved from the blockchain. The "Vehicle" component, for example, utilizes custom hooks like useConnection and useContract to interact with a specified smart contract, retrieving vehicle details such as the plate number, price, owner ID, and driver ID, and displaying them in a table format. Similarly, the "Driver" component connects to the Ethereum network, fetches driver information from the smart contract, and presents details such as the driver's ID, name, and associated vehicle owner ID to the users.

These components allow users to select and manage driver-related data efficiently (Figure 3).

Another key feature of the BC-4Track web application is its capability to record and display accident data. Using custom React hooks that interact with Ethereum smart contracts, users can connect their Ethereum wallets to the application, record accident details, and query previously recorded accident reports on the blockchain. The user interface provides fields for entering accident information such as plate number, date, time, location, reason, and value. The setAccident function records the user-entered accident information onto the blockchain smart contract, while the viewAccident function retrieves and displays all recorded accident data from the blockchain. Dynamic data updates are facilitated by the useEffect hook, which refreshes the accident list whenever there is a change in the Ethereum wallet connection.



Figure 3: Vehicle and Driver Registration Processes

Through these features, the BC-4Track web application offers a seamless and intuitive interface for users to interact with the blockchain-based vehicle tracking system.

IV. CONCLUSION

In this research, the development efforts are presented which includes the integration of Ethereum blockchain technology with a React-based web application. The web application, designed to be a user-friendly decentralized application, enables secure and transparent interaction with smart contracts on the Ethereum network. Custom React hooks have been developed to facilitate seamless connectivity and interaction with the blockchain, while various React components and JavaScript libraries have been utilized to enhance the user interface.

The development process emphasizes the combination of modern web technologies with blockchain to create an interactive and reliable vehicle tracking system. Although the Smart Kit component, which will be built on advanced sensor integration and IoT technology to collect real-time vehicle data, is still under development, the current progress demonstrates the potential of blockchain-integrated solutions in providing secure, transparent, and efficient data management. This research aims to contribute to the broader field by showcasing how blockchain technology can be effectively employed in practical applications, and ultimately enhances the capabilities and trustworthiness of vehicle tracking systems.

ACKNOWLEDGMENT

This work is funded by Turkcell Technology A.S. and TUBITAK (The Scientific and Technological Research Council of Turkey) under 1505 Program, Project no 5220118.

REFERENCES

[1] Atzori, L., Iera, A., Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805.

[2] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.

[3] An, S. H., Lee, B. H., Shin, D. R. (2011, July). A survey of intelligent transportation systems. In 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks (pp. 332-337). IEEE.

[4] Zhang, J., Wang, F. Y., Wang, K., Lin, W. H., Xu, X., Chen, C. (2011). Data-driven intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, 12(4), 1624-1639.

[5] Alsadi, M., Gülseçen, S., Kara, S., Ozdenizci Kose, B., Coskun, V. (2019). A Blockchain Based Data Management

Model. Journal of Information Systems and Management Research, 1(1), 31-36.

[6] Alsadi, M., Yildirim, S., Gülseçen, S., Ozdenizci Kose, B., Coskun, V. (2019, October). Akıllı Araç Ekosistemlerinde Blockchain Tabanlı Güvenli Veri Yönetim Modeli. In 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), (pp. 1-5). IEEE.

[7] Coskun, V., Alsadi, M., Kose, B. O., Mantar, H. A., & Oral, M. A. (2023, November). BC-4Track: A Blockchain based System for Vehicle Tracking Management. In 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS) (pp. 1-5). IEEE.

[8] Zheng, Zibin, et al. Blockchain challenges and opportunities: A survey. International journal of web and grid services 14.4 (2018): 352-375.

[9] Zheng, Z., Xie, S., Dai, H. N., Chen, X., Wang, H. (2018). Blockchain challenges and opportunities: A survey. International journal of web and grid services, 14(4), 352-375.

[10] Casino, F., Dasaklis, T. K., Patsakis, C. (2019). A systematic literature review of blockchain-based applications: current status, classification and open issues. Telematics and Informatics, 36, 55-81.

[11] Wang, X., Zha, X., Ni, W., Liu, R. P., Guo, Y. J., Niu, X., Zheng, K. (2019). Survey on blockchain for Internet of Things. Computer Communications, 136, 10-29.

★★★