**IoT Inventory Management using Blockchain**

|  |  |  |  |
| --- | --- | --- | --- |
| Joseph Nguyen School of Computer Science The University of Adelaide joseph.nguyen@student.adelaide.edu.au | Nguyen Khoi Tran School of Computer Science The University of Adelaide nguyen.tran@adelaide.edu.au | | Ali Babar School of Computer Science The University of Adelaide ali.babar@adelaide.edu.au |
|  |  |  | |

ABSTRACT

The recent proliferation of Internet of Things (IoT) devices as a means of tracing and interacting with the physical world has found applications in many areas including smart cities, supply chains, and industry 4.0. Traditional IoT-based systems are built upon a centralized structure which is often vulnerable to trust issues such as data tampering and mismanagement. Blockchain technology provides an innovative approach to resolving these issues through its decentralized, distributed ledger that maintains immutable records of transactions whilst removing the need for a central authority. This paper proposes a blockchain-based inventory management system that can monitor the transfer, deployment and maintenance of devices without relying on trust through the inherent security, transparency and immutability properties that the blockchain provides. A prototype of the system will be developed on the Ethereum blockchain and the API needed to interact with the system will be implemented. The system will then be evaluated for any improvements.

1. INTRODUCTION

Supply chain tracking allows companies to monitor the movement and status of their inventory, which can affect supply chain efficiency, product safety and security, management of risks and cost and delivery performance [1]. Some common technologies used to assist in tracking supply chains include barcodes and tags. However, these methods share a problem in that the data collected is all stored in a single source. This raises trust and security issues as it becomes increasingly difficult to establish a single point of trust in a growing company. Tampering or mismanagement of data by those controlling the data source could be possible without the knowledge of the rest of the company. Furthermore, the centralized system has a single point of failure which is more vulnerable to being targeted by attackers.

This problem could be resolved by having a secure inventory management system that maintains an immutable record of transactions without the need of a central authority. This paper proposes such solution through the integration of blockchain technology to implement a trustless inventory management system that can monitor the transfer, deployment and maintenance of devices.

The decentralized system distributes data to every node in the network, minimizing the need for trust towards a central authority and the risk of data tampering. Furthermore, the immutable transaction records stored on the blockchain ensures that data integrity is maintained. The transparent nature of the system makes all transactions recorded in the blockchain visible on the network, enabling all connected users to monitor the transactions at any time and thus reducing the risk of fraudulent activities.

Whilst the proposed inventory management system aims to resolve trust issues with previous methods of supply chain tracking, it also has potential uses for smaller scale cases, such as smart homes and personal device tracking, and for larger scale cases in smart cities and industry 4.0.

2. BACKGROUND

IoT is currently one of the top trending technology worldwide with an estimated growth to more than 5.8 billion connected devices in 2020 [2]. In essence, it is a set of devices, or ‘things’, which uses sensors and embedded software to interconnect over a network, the ‘internet’. This system provides the capability for communication, collection and exchange of data between connected devices over the network, which leads to increased efficiency, automation and control [3]. In recent years, IoT technology has seen a rise in integration with another major technology in blockchains.

A blockchain is a decentralized, public ledger that applies peer-to-peer networking to maintain a list of transaction records linked through cryptography. When a transaction is made, the data is sent to all nodes in the network for validation. Once a consensus has been reached, the data is stored in a block with the cryptographic hash of the previous block, hashed and then added onto the end of the chain. Early implementations of blockchains revolved around the exchange of currency, giving rise to cryptocurrencies such as Bitcoin [4]. However, its scope has now expanded to general applications in market and finance, allowing transactions with assets other than currency and introducing the concept of smart contracts and smart property.

Smart property is property encoded onto a blockchain, enabling ownership to be monitored and controlled through the blockchain via its unique private key. Owners can sell their property by transferring the private key to the new owner. Moreover, smart property can be transacted via smart contracts, pre-defined contracts on the blockchain that execute a set of instructions when certain conditions are met. These properties can be tangible like phones, cars and houses, or intangible like shares, stocks and copyrights [5].

3. LITERATURE REVIEW

Early research and development on property and supply chain tracking often saw the employment of barcodes and tags to aid in monitoring the movement of properties. In a research on agri-food supply chain traceability, Tian [6] demonstrated the applications of Radio Frequency Identification (RFID) tags to support Chinese agri-food markets. These tags were an upgrade over the barcode system as they did not require line-of-sight scanning thus were more flexible for information gathering and real-time tracking [7]. Whilst they were shown to reduce logistics cost and improve the safety and quality of agri-food, the key issue that was presented was the centralized traceability system of RFID tags. Information extraction and sharing across the organization was difficult and trust became an issue. Likewise, Caro et al. [8] described that agri-food supply chains built upon centralized infrastructures leave room for major concerns such as data integrity, tampering and single points of failure.

Since the introduction of Bitcoin and its underlying blockchain technology by Nakamoto [3], blockchains have been garnering attention as the new and innovative approach to solving these issues. Research by Tian [9] also looked into integrating blockchain and IoT technology to further improve supply chain traceability. The decentralized, distributed system for storing and sharing data would not only increase the credibility of the data but also minimize the risk of fraud. Abeyratne and Monfared [10] explored blockchains in a similar research to discuss the potential benefits of such technology in manufacturing supply chains. They highlighted durability, transparency, immutability and process integrity as key advantages that blockchains provide and proposed a system in which products are given unique identifiers linking them to virtual identities on the network that can be exchanged via smart contracts.

Blockchain technology has also been a trending topic of research in other areas of IoT. Kostal et al. [11] introduced a monitoring and management architecture for IoT devices based on blockchains, utilizing the decentralized and distributive properties to improve network security and storage. Research by Liu et al. [12] explored methods to improve data integrity verification for cloud-based IoT. They capitalized on the peer-to-peer networking of blockchain technology to implement a more reliable data integrity verification system without relying on a third party auditor. Similarly, research on a blockchain platform for industrial IoT by Bahga and Madisetti [13] resulted in enhanced cloud-based manufacturing functionalities from the removal of the intermediary party during transactions.

The proposed inventory management system will be similar to the proposed system by Abeyratne and Monfared. By registering properties onto the blockchain as smart property, they can be transacted with via smart contracts that control information such as their ownership, location and status. However, this system will focus more on user usability and will be designed for ease of use whilst maintaining the key functionalities that is expected of an inventory system.

4. METHODOLOGY

The prototype of the inventory management system will contain two key components, the underlying blockchain layer describing the logic behind the system via smart contracts and the API layer for communication and interaction with the system.

The blockchain layer will be implemented using Ethereum [14], an open source, programmable blockchain platform with a range of useful features to support the development of decentralized applications. This will help simplify the creation of a local blockchain network to run the inventory system on. The network will be configured to use proof of authority as its consensus mechanism, which enables authorized nodes to validate and accept new transaction blocks at a given interval. Compared to the proof of work mechanism that Bitcoin uses, which requires complex hashing calculations of each new block, proof of authority removes the need for such calculations thus greatly reducing the CPU usage of the nodes.

Ethereum also comes integrated with web3.js [15], the Ethereum JavaScript API that enables interaction with the smart contracts deployed on the blockchain network. The contracts themselves will be in Solidity [16], an object-oriented programming language designed to support the implementation of smart contracts. Solidity is derived from common programming languages in C++, Python and JavaScript hence contains familiar syntaxes and features and allows programming skills to easily be transferred over. The smart contracts will be developed using Remix [17], a browser-based Ethereum IDE featuring many useful plugins that provide testing, debugging and deploying capabilities. Implemented contracts can be compiled and deployed directly in the browser which enables fast testing and debugging of functionalities during its development.

Once the smart contracts are complete and functioning as expected in the Remix testing environment, they will be transferred to the live blockchain network and deployed using Truffle [18], a development framework for Ethereum. Truffle handles the compiling, linking and deployment of contracts to the blockchain network and provides a console for direct communication with them, which will be used to test the contract to ensure that the system is functional in a real blockchain environment.

The API layer of the inventory management system will be implemented according to the OpenAPI specification [19], which defines a standard interface to RESTful APIs. This will help users to understand how the API works without needing the implementation logic. It will reflect all the functionalities of the smart contracts. The completed API will be used to generate a corresponding server using Swagger [20], a set of tools to simplify API development. The API logic will then be developed in JavaScript, which will allow communication between the server and the contracts. This will ultimately enable the server to interact with the deployed inventory system on the blockchain network via the API.

5. EXPERIMENTAL SETUP

A new ethereum node was generated on a local device to initialize the blockchain network via the ethereum protocol. The first block of the chain in that node, the genesis block, was configured to give the node authority to validate and accept new transaction blocks as part of the proof of authority consensus mechanism. This was set to allow a new block to be processed once every 15 seconds.

The smart contracts were then implemented through the Remix Ethereum IDE. The inventory system was divided into three subcontracts: device, inventory and inventory management interface. The device contract stores information about the name, owner, location and state of the device and has a constructor to set these parameters when a new device is created. It also includes functions to modify the stored information, return the information and to rotate between states.

The inventory contract contains the necessary variables and functions needed to operate the inventory system. It stores the address of the inventory manager, which is used to authorize some functions, a dynamic list of devices in addition to some helper variables to keep track of key information such as device IDs and number of devices in inventory. Its functionalities include listing the information of current devices in the inventory, adding and removing devices, modifying and looking up device information and the ability to check in and out devices to and from the inventory.

The inventory management interface simply defines the key functionalities of the inventory system to provide a readable, abstract view for users to better understand the system whilst hiding the implementation details in the inventory contract. As each function was being developed, they were frequently tested and debugged on Remix to ensure that they were behaving as expected.

The completed contracts were then transferred to the local device where Truffle was used to compile and deploy the system onto the local blockchain network initialized earlier. The console provided by Truffle could then be used to interact with the inventory system via the web3.js library. Each function was tested once more on the live blockchain environment to ensure that the authorized node is being sent the new transaction blocks for validation and that these blocks contained the expected transaction details such as inputs, outputs and any emitted events.

Once the inventory management system was confirmed functional on the blockchain network, the corresponding API was implemented to allow the system to communicate with other applications. Each request in the API was linked to a functionality of the system defined in the inventory management interface. A server was then generated from the implemented API and the business logic behind the API was developed. This was achieved by creating an instance of the inventory system from the blockchain network and calling the required functions to satisfy each request. The transactions would go through to the authorized node for validation, after which the server would be able to send back a response containing the output of the transactions. The API was tested through the Swagger UI tool, a visual testing environment for APIs, to ensure that each request was met with the correct interactions with the inventory system and the correct response from the server.

6. RESULTS

The resulting product is a prototype of the inventory management system with an Ethereum blockchain layer containing the smart contracts that define the logic behind the system and an API layer that defines how the system communications and interacts with other applications. The key functionalities of the system includes the ability to add new devices to the inventory, remove existing devices from the inventory, list the current devices in the inventory, modify the information of a device, look up the information of a device and rotate the state of a device. There are three states in order of ‘inStore’, ‘checkedOut’ and ‘pending’.

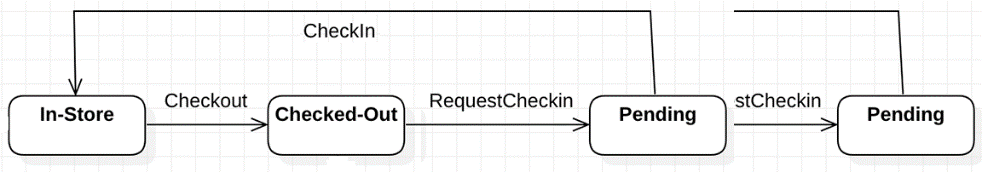
The add request takes three string arguments representing the desired name, owner and location information of the new device. These arguments are passed to the add function of the inventory system which creates a new device in the inventory and initializes it with the given information, in addition to a unique, incrementing device ID and the initial state ‘inStore’. The response is a string representing the status of the transaction, ‘OK’ if the transaction was successful or ‘Error’ if the transaction failed.

The remove request takes an integer argument representing the ID of the device to be deleted. The argument is passed to an ID validator function to ensure that the device ID exists, otherwise the transaction fails. The argument is then passed to the remove function of the inventory system which deletes the device corresponding to the given ID. The response is a string representing the status of the transaction, ‘OK’ if the transaction was successful or ‘Error’ if the transaction failed.

The list request takes no arguments and calls the list function of the inventory system which goes through each existing device in the inventory and creates a JSON object containing the ID, name, owner, location and state information of the device. These JSON objects are then appended to a single array that is returned by the function. The response is the array containing the list of JSON objects representing the information of devices in the inventory, or an empty array if the inventory is empty.

The modify request takes an integer and three string arguments representing the ID and the desired name, owner and location information of the device to be modified. The integer argument is passed to an ID validator function to ensure that the device ID exists, otherwise the transaction fails. The arguments are then passed to the modify function of the inventory system which changes the device information to the given information of the device corresponding to the given ID. The response is a string representing the status of the transaction, ‘OK’ if the transaction was successful or ‘Error’ if the transaction failed.

The look up request takes an integer argument representing the ID of the device to be looked up. The argument is passed to an ID validator function to ensure that the device ID exists, otherwise the transaction fails. The argument is then passed to the look up function of the inventory system which returns the name, owner, location and state information of the device corresponding to the given ID. The response is a JSON object representing the output of the transaction, the device information if the transaction was successful or ‘Does not exist’ if the transaction failed.



**Figure 1: Diagram of device state rotation**

There are three state rotation requests that behave similarly: checkout, request check-in and check-in. They each take an integer argument representing the ID of the device to have its state rotated. The argument is passed to an ID validator function to ensure that the device ID exists, otherwise the transaction fails. The argument is then passed to either the checkout, request check-in or check-in function of the inventory system which first examines the current state of the device corresponding to the given ID to ensure that it is either ‘inStore’, ‘checkedOut’ or ‘pending’ respectively, otherwise the transaction fails. The function then rotates the state of the device to the next state, resulting in ‘checkedOut’, ‘pending’ or ‘inStore’ respectively. This process is illustrated in Figure 1. The response is a string representing the status of the transaction, ‘OK’ if the transaction was successful or ‘Error’ if the transaction failed.

However, the implemented system is still just a prototype hence there are many improvements that could be made. For instance, the system currently only looks up the present information of devices, which could be improved by enabling the function to search through the blockchain and return the history of the device as well to allow tracing of the device back to its origins. Another improvement could be to allow more control over the list function, such as returning devices of a specific state or ID range, instead of always returning all devices. Similarly, the remove, look up and state rotation functions could be enabled to delete, look up and rotate the state of multiple devices at once. These could prove useful as the inventory size grows larger.

A new feature could be implemented to reset the inventory back to its initial state, which could be useful for testing purposes or if the system is required for a different reason. This could be achieved by adding a reset function that deletes all devices from the inventory and reverts all variables back to their initial values. However, this will not affect the information stored on the blockchain due to its immutable nature. Another potential feature is the ability to transfer devices and their history to another inventory on a different blockchain network. This would enable cross blockchain compatibility and allow users to pass a device to different inventories along with its history, which otherwise would only exist in the blockchain of the initial inventory system the device originated from.

7. CONCLUSION

Blockchain technology is an innovative approach to solving key trust issues that exist in many traditional, centralized systems. It provides a distributed, public ledger that maintains transaction records, enabling the development of more secure, transparent and immutable systems for decentralized applications. This paper explores the integration of blockchain with IoT technology to improve on the current supply chain tracking systems and proposes a trustless inventory management system that can monitor the transfer, deployment and maintenance of devices.

A prototype of the inventory system was implemented with the basic functionalities expected of an inventory system such as the ability to add, remove and look up devices. The corresponding API for the system was also developed to enable communication and interaction between the system and other applications. It also enables other applications to integrate the inventory system into their own systems with ease. Further research could look into linking the system to sensors to enable seamless, autonomous interactions. Much research and development is still required before the proposed system becomes practical for industrial use.

REFERENCES

[1] Potts S. 2015. "The Logistics of Tracking Traceability". *https://knowthechain.org/the-logistics-of-tracking-traceability/*

[2] Gartner. 2019. “Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020”. *https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io*

[3] Pauw C. 2018. "https://cointelegraph.com/news/how-significant-is-blockchain-in-internet-of-things". *https://cointelegraph.com/news/how-significant-is-blockchain-in-internet-of-things*

[4] Nakamoto S. 2008. “Bitcoin: A Peer-to-Peer Electronic Cash System”. *https://bitcoin.org/bitcoin.pdf*

[5] Swan M. 2015. "Blockchain: Blueprint for a new economy". O'Reilly Media, Inc.

[6] Tian F. 2016. “An agri-food supply chain traceability system for China based on RFID & blockchain technology”. In *2016 13th international conference on service systems and service management (ICSSSM)* (pp. 1-6). IEEE

[7] Michael K and McCathie L. 2005. “The pros and cons of RFID in supply chain management”. In *International Conference on Mobile Business (ICMB'05)* (pp. 623-629). IEEE

[8] Caro M.P, Ali M.S, Vecchio M and Giaffreda R. 2018. “Blockchain-based traceability in Agri-Food supply chain management: A practical implementation”. In *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)* (pp. 1-4). IEEE

[9] Tian F. 2017. “A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things”. In *2017 International conference on service systems and service management* (pp. 1-6). IEEE

[10] Abeyratne S.A and Monfared R.P. 2016. “Blockchain ready manufacturing supply chain using distributed ledger”. *International Journal of Research in Engineering and Technology*, *5*(9), pp.1-10

[11] Kostal K, Helebrandt P, Bellus M, Ries M and Kotuliak I. 2019. “Management and monitoring of IoT devices using blockchain”. *Sensors*, *19*(4), p.856

[12] Liu B, Yu X.L, Chen S, Xu X and Zhu L. 2017. “Blockchain based data integrity service framework for IoT data”. In *2017 IEEE International Conference on Web Services (ICWS)* (pp. 468-475). IEEE

[13] Bahga A and Madisetti V.K. 2016. “Blockchain platform for industrial internet of things”. *Journal of Software Engineering and Applications*, *9*(10), pp.533-546

[14] Ethereum. 2020. “Ethereum”.  
*https://ethereum.org/ /*

[15] web3.js. 2016. “Ethereum JavaScript API”. *https://web3js.readthedocs.io/en/v1.2.8/*

[16] Solidity. 2020. “Solidity”. *https://solidity.readthedocs.io/en/v0.6.8/*

[17] Remix. 2020. “Ethereum IDE”. *https://remix.ethereum.org/*

[18] Truffle Suite. 2020. “Truffle”. *https://www.trufflesuite.com/truffle*

[19] SmartBear. 2020. “OpenAPI Specification”. *https://swagger.io/specification/*

[20] SmartBear. 2020. “Swagger Tools”. *https://swagger.io/tools/*