Archives of Business Research – Vol. 9, No. 10
Publication Date: October 25, 2021
DOI:10.14738/abr.910.11164.
Lim, J. (2021). How Blockchain and IoT-Based Decentralized System Solves ULD Rental Problem of Airlines. Archives of Business Research, 9(10). 331-344.

How Blockchain and IoT-Based Decentralized System Solves ULD Rental Problem of Airlines

Janghyuk Lim
Department of Business Administration, Shantou University, Shantou, China

ABSTRACT

This study is devoted to ULD (Unit Load Device) rental among airlines and air cargo terminals at airports through data exchange. When airlines want to rent their partner airline's ULD, they use it after receiving data through its ULD inventory management system. This paper proposes a de-centralized ULD rental solution by combining blockchain technology and IoT (Internet of Things). The airline that wants to rent a partner airline's ULD checks the available inventory information before use and pays the amount according to the time of service. In addition, the smart contract specifies the remittance status to activate IoT on the ULD. Airlines that supply ULDs for partner airlines first create inventory information to provide services. Airlines that want to rent ULDs can check availability and remit rental costs according to usage time in cryptocurrency. The ULD rental service ends when the usage time is over. This ULD rental solution allows all airlines rent out their ULDs to partner airlines in real-time. Also, when a specific airport runs out of its own ULD, it is possible to overcome the difficulties of inventory shortage by renting the inventory of partner air-lines.

Keywords: Blockchain, decentralization, IoT, smart contract, unit load device.

INTRODUCTION

In recent decades, the airline industry has become increasingly competitive with the advent of large aircraft. As a result, ULDs, essential for air transport, has become crucial for efficient air transport [1]. According to IATA, 1 million aircraft worldwide are in service with ULDs valued at over $1 billion, which expensive assets requiring proper are handling. The total cost of repairing and losing ULDs (Unit Load Devices) every year get estimated at $330 million [2]. Ensuring the right ULDs are available in the correct airport in real-time in the right conditions is essential for airline operations and revenue management. However, ULD operating standards and procedures vary across airlines and airports [2]. Lack of unified mechanisms of watching over the whole process from one viewpoint, culminates transparency. Suppliers and purchasers of ULD do not have visibility on the state of their assets [3]. So, sharing accurate ULD movement and available ULD inventory information between departure and arrival airports and partner airlines is required to improve airlines' operation and profit management. The fourth Industrial Revolution featured new technologies such as IoT, blockchain, RFID, 5G, and intelligent machines. The international transportation sector must keep up with technological advances to increase productivity and competitiveness [4].

The leading technology covered in this paper is blockchain-based IoT, which activates and controls IoT through smart contracts. Blockchain technology can ensure originality,

traceability, dependability, and data security. All these aspects enhance trust among supply chain parties through providing decentralized supply chain information [5]. Combining IoT and blockchain in this paper is most IoT gadgets depend on a primary entity for data transactions but moving this responsibility to decentralized nodes makes these devices smarter. Furthermore, utilizing a blockchain for IoT, the security essentials and measures get significantly increased. It places the messaging and notification functions within each gadget's protocol layer [6].

This paper developed a full-proof strategy of managing the ULD supply chain to inventory management, standard processes, and procedures while utilizing block-chain-based smart contracts and IoT. The objective of blockchain and IoT-based ULD rental is to airlines and air cargo terminals at airports cooperating on a single network rather than their own centralized ULD rental system. Through this, airlines and air cargo terminals can share reliable information in real-time, and based on this, it is possible to rent, repair, exchange, and use the necessary ULDs.

Blockchain ULD Rental Solution – A Brief Overview and Functional Architecture

Requirement of ULD management

The airline must effectually reposition ULDs between airports to balance supply and demand and keep a sufficient and economical fleet size. Therefore, it is vital for an airline to correctly estimate the safety stock level for each operating airport to cope with its ULD repositioning operation [7]. In addition, real-time ULDs visibility and accomplishing optimization are gaining importance due to dynamic loading considerations, including division of hazardous products, aircraft weight balancing, and oversize cargo handling [3].

Table 1. Consortium-Based Decentralized Ambulance Dispatch Model.

Key Feature	Traditional ULD rental System	Blockchain-based Solution
Control	Centralized by ULD managing body (One database copy)	Decentralized (All parties hold copies of transactions)
Database	Central server	A blockchain is a database due to its stores information in blocks
The function of data processing	Create, Read, Update and Delete (CRUD)	Read and write only
Integrity	Both unintentional errors and intentional factors affect the integrity of the data.	Immutable, cannot be edited or deleted ledger provides integrity
Transaction size and speed	Designed for specific data volume and transaction	Fixed block size and transaction speed
Transparency	Transferring data without trust	Encrypted blocks, data available to all nodes, make transparent transactions.

Collaboration between airlines has also strengthened, and ULD leasing has become common. Nevertheless, the existing ULD rental method weakens information transparency, security, reliability, and timeliness due to the centralized system and operations. To illustrate one example of ULD operation, the ULD control center collates the ULD control receipts (in paper or electronic form) from their field operators at air cargo terminals. Then, it transmits the transfer information on these receipts into LUC (ULD exchange control) messages. These ULD control receipts are kept on records for two (2) years after the transfer [8].

Features of ULD rental solution

The blockchain-based ULD rental solution has distinctive features from existing systems. Firstly, no central entity manages the ULD system, but the airlines, air cargo terminals, ULD leasing companies, and repair centers participating in the ULD rental solution share and verify data in one network. Participants can be limited only authorized and limited participants by constructing a secret blockchain network base on specific business logic. Secondly, there is no central server like the existing ULD rental system. Instead, the block can store and retrieve data, so participants participating in the blockchain to manage the ULD do not need to deploy a separate server. Thirdly, data processing of the blockchain is limited to writing and reading only, preventing forgery or falsification of ULD rental data. Fourthly, the blockchain-based system pro-vides integrity with immutability that cannot be modified or deleted. However, in the existing system, various intentional factors such as unintentional human error and business strategies and policies affect the integrity of the database's data. Also, the existing system sets the transaction size and speed according to the user company's data volume; however, blockchain can't change them according to individual business needs due to the fixed length and transaction speed. Lastly, all data is encrypted and stored in blocks, and participants can read it, thus creating trust for data and between participants. However, the existing system is difficult to trust because data management is centralized.

Table 2. Significant challenges for ULD rental and expected effectiveness using blockchain and smart contracts.

Challenges	Components of Digital Technology	Expected effectiveness
Most IoT devices count on a central entity [6]	Blockchain	Moving responsibilities to individual nodes will accord the notification and messaging operations within each IoT gadget's protocol layer
IoT needs integrity control and planning optimization of available Resources [3]		It is unstoppable, incorruptible, and able to make the business agreement, which can be said to execute themselves [9]
Frequently, workers do not know the exact shipment information to load ULD in advance [10]		Creating the user interface and decentralizing data for all participants [11]
ULD allocation to airlines or aircraft are highly empirical [1]	Smart contracts	The data stored in the blockchain gets manipulated to conduct information recording, balances, facts, associations and other information to actual-world contracts and participants [12]
Insufficient authentication, uncontrolled data generation and use, and monitoring without permission [13]		Agreements are automatically secured, continuously executed, and enforced [14]

From the technology point of view, IoT-based solutions impact warehousing and inventory management, among other domains. ULD monitoring and location identification are optimized and realized using IoT technology [15]. Since the function of logistics depends on decision-making results, IoT gets to play an essential part as data-driven systems to enhance decision-making through ac-quiring of effective data, processing, and assessing [16]. However, major challenge concerning IoT lies in confidentiality, integrity, secure authentication, and trust management [17]. The hurdles identified above can get overcome through: Using a powerful combo of critical features of smart contracts and blockchain and how to solve ULD-related problems, Listed in Table 1. This paper explicitly defines the blockchain and IoT-related actual operation process and runs main functions to con-firm it. The ULD rental solution proposed in this paper implements decentralized operation initiated by all airlines as participants in blockchain network range from ULD service contract price negotiation, inventory management, and contract-controlled ULD asset monitoring to contract fulfilment.

Concurrence Algorithm

Since blockchain data gets stored across nodes scattered around the world instead of a centralized server to distribute power horizontally [18], each node needs a process of mutually agreeing that the data recorded in the block is an unforged original. Furthermore, to ensure that all nodes are processing the duplicate transactions and work for the same objectives as a combined group of efforts, all must reach a concurrence agreement [11]. In the blockchain-based ULD solution, airlines and various related companies can participate in the decentralized network shown in Figure 1.

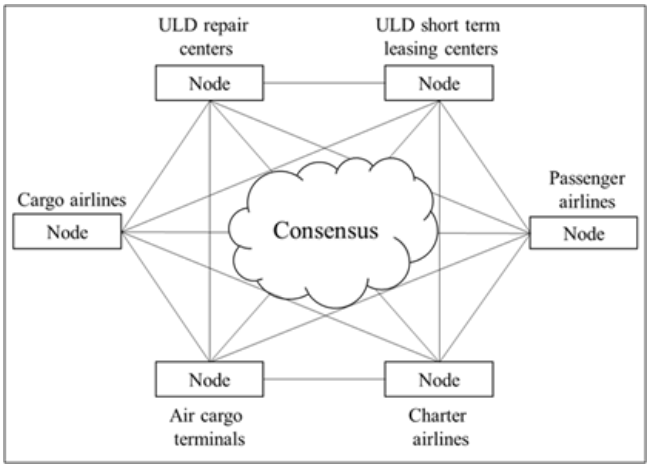

Figure 1: Scheme of a blockchain-based P2P network for ULD supply chain all transaction data is shared, verified, and stored by all nodes.

If a specific node creates a malicious block, stores the manipulated data, and spreads out in the network to disrupt the ULD tradeoff. In that case, the reliability of the entire ULD rental system will decrease. Even if such a spiteful situation occurs, the process and algorithm that generates the correct block through mutual verification by multiple nodes (Miners) who want to conduct the network in the right direction are called concurrence. A concurrence is a vital aspect of the blockchain system. A concurrence is helping to reach a unified ground while keeping intact the decentralization aspect. In contrast, a concurrence intends to yield a system of strict rules lacking a ruler [19]. Various models continuously develop for the concurrence algorithm, but the most basic ones are PoW, PoS, and PoA. Simple concepts are summarized below:

- PoW (Proof-of-Work): It generates random hashes until miners find the one below the target value to make the next block. Miners broadcast hashes to all connected nodes after finding the correct hash to process transactions and create blocks [11].
- PoS (Proof-of-Stake): PoS gets based on a miner's investment capital in the system [20]. Ethereum defined that to become a validator in the network, and participants must stake their cryptocurrency [21]. Unlike PoW, which builds a concurrence based on the amount of computing power miners perform, the owner of the next block is chosen according to a combination of random selection and wealth or stake in the proof-of-stake (PoS) system [22].
- PoA (Proof-of Authority): PoA gets based on a small number of trusted entities. All transactions get validated by pre-permission nodes that work as mediators or leaders, and this trusted entity protects PoA. As a result, PoA gets frequently utilized for supply chain management and its function in various cryptocurrencies [23].

Cryptographic Hash Operation

A cryptographic hash operation is a one-way hash function [19], converting any input data into fixed-length random character data. The generated string data makes it impossible to retrieve or identify the original data. When employing the SHA256 or keccak256 hashing algorithms and function with any input length, the result is always 256 bits [12]. All information gets converted into a fixed output value of 256 bits, 32 bytes, in the ULD transactions or information update process, regardless of the input size. A cryptographic hash makes it easy to determine whether a document has been tampered with, even if the original document is not present. For example, when two airlines in a ULD rent relationship create a rental document and create and share a cryptographic hash of the record, the document's authenticity can get judged without having a copy of the original document.

Smart Contract

The view of smart agreements got coined by Szabo in 1994. However, this idea did not get materialized until the genesis of blockchain technology [24]. Etheruem developed a language called Solidity and write smart contracts using Solidity. Therefore, a smart contract is a language for the Ethereum blockchain, and the deployer must pay a fee called gas to use a smart contract after deployment on the blockchain. Smart contracts operate autonomously, without third-party intervention. In addition, it also acts as an intelligent software agent to ensure that all decision points are achieved as expected [25]. Features of smart contracts also can be described via specific properties [26]:

- The smart contract extends its impact in the blockchain, and the latter sustains the account-based model.
- Smart contract involves expressing logical business with code.
- All blockchain network participants can check the smart contract code.
- When initiated by a transaction delivered to its address, the smart contract gets solely and spontaneously executed on each network node.
- All blockchain network participants can obtain verifiable tracking of smart con-tract activities.
- Smart contract is required to be decisive, which means that the same input will always produce the similar output. Non-decisive smart contracts with unexpected results interrupt the blockchain flow from arriving at a concurrence on the out-come of the agreement.
- Upon writing the code, the smart contract can get regarded as a solo agent with entirely behavior that can get predicted.

Smart contract to control IoT

The popularity and success of the IoT depends on the rapid growth of the applications used. Additionally, smart agreements may play a crucial part in the IoT [27]. Because the implementation of smart contracts, possible in any sector that requires them [14]. Echoing the findings of Christidis & Devetsikiotis, (2016) smart contracts to control IoT have specific properties to describe: this is also expedites property and services sharing in general. It works by unlocking the device and simplifying the billing of the token called Ether. The Ether passes on Ethereum blockchain, an optimized open blockchain connection for smart contracts using its cryptocurrency. Observing from the view of the producer, the centralized model at the moment encompasses heightened maintenance costs, factor in distributing upgrades of the

soft-ware to equipment out of production in many years. Observing from the view of the client, there inadequacy of trust in the device. All of the manufacturer's IoT devices can run on a uniform blockchain connection. The producer initiated storage of the hash value through smart contracts of the current firmware upgrade on the web. In addition, transactions (i.e., a data connection) in blockchain create synergies involving IoT gadgets and smart contracts to collect and exchange data. For example, while carrying out a smart contract, flow of data from IoT gadgets can also point out that the conditions for further workflow steps have been met or verify that contractual commitments have gets fulfilled [27].

Design of blockchain and IoT-based ULD rental solution: Components, functions, and roles

Smart contracts are programs that run on Ethereum. Smart contracts use Turing complete language and can perform more complex processing. The Ethereum block-chain provides the possibility to deploy and execute decentralized smart contracts mainly coded in the Solidity language [28]. In this study, smart con-tracts got developed using a programming language called Solidity. Although the Solidity language is object-oriented, since it got created to develop smart contracts of the blockchain, the characteristics of the constructor are also slightly different from other languages (ex, C++, Java). The difference between the Solidity language and other object-oriented languages is that it is not to create an object on the program indefinitely but to create a contract on the blockchain once and use it.

Remix IDE (Integrated Development Environment) is an open-source web application for writing, debugging, and deploying Ethereum smart contracts. This study used Re-mix IDE to compile and deploy the developed smart contract to verify whether the function implementation is effective. In this study, a proposed ULD rental answer carries out end-to-end tasks from real-time-based inventory updates, close deals, monitored asset control, and contract accomplishment. The contract usage process shows the various participants' interactions while displays events of the function called in the smart contract. Each entity on the connection has an Ethereum link, which allows them to communicate by calling the operations of smart contracts.

ULD rental business logic

In this study, an owner who owns a ULD (e.g., airline, ULD rental company) rents out a ULD and withdraws a rental cost using smart contracts and IoT to users (e.g., airlines and logistics companies) who want to use the ULD. The pros of blockchain in an IoT set up to ULD rental consider the characteristic supply chain that indicates the worth of a blockchain and IoT: update a ULD inventory that is available at the airport or cargo terminal (①), ULD user selects and pays to rent out (②), then ULD confirms payment (③), ULD user can get approval to use ULD for a given period so, when the usage time is over, the status is changed, and service use will terminate (④), ULD owner withdraws service fee what ULD user paid for (⑤), ULD user can use a ULD until the service end time. Figure 2 schematically illustrates the process of using a smart contract.

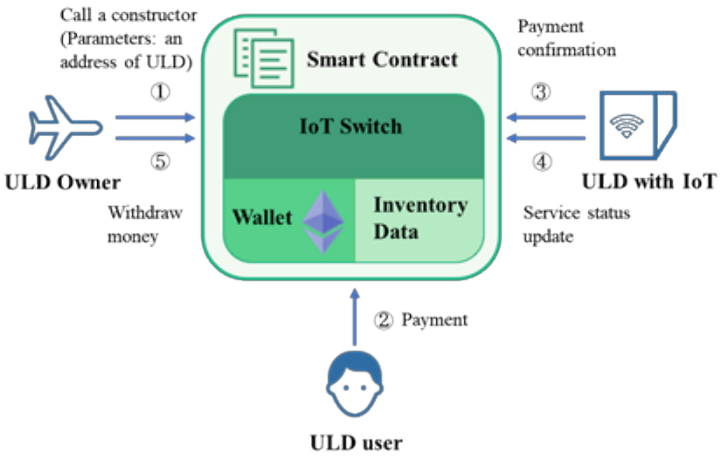

Figure 2: ULDwithIoT contract usage process.

It provides proof-of-work use smart contracts to solve the ULD rental process's data management, agreement compliance, transparency, and data integrity challenges. Smart contract of ULD rental solution contains several primary functions: Key features of ULD trading is implemented by operating four functions, pay to switch, update status, withdraw Funds, and kill.

Configuration of smart contract: ULDwithIoT.sol

As decentralized applications, blockchain gets driven by smart contracts, this study constructs and distributes smart contracts. For the ULD rental solution, a declarative approach defines what smart contracts in this study want to deploy, what dependencies are related to IoT and Cryptocurrency, and how smart contracts need to get initialized to execution. This chapter explores configuration options for developing smart contracts.

Source code: ULDwithIoT.sol

Rental ULD remotely requires mutual trust between multiple parties. In this con-tract, configuration involves only ULD owners and ULD service users. ULD service users want to receive ULD from the ULD owner, and the ULD owner wants money in return. The ULD owner acts as a service provider to set up the IoT equipment: It needs to let the IoT device that the contract address will need be considered its contract. ULD users pay to unlock IoT devices. The IoT device automatically runs a function to stop the contract. ULD owners can withdraw funds from the contract.

The tricky part is the ULD's remote activation. There is no way to determine in real-time for sure whether ULD is available and contract conditions in the centralized ULD management system. However, there is a way to solve this problem; the ULD owner must set the IoT device in this smart contract. Then, when the contract, ULDwithIoT, is first created, service cost, availability of ULD, and service time can get shown publicly. Since Solidity language is an object-oriented language (OOP), it has a constructor, an object-oriented feature. Therefore, the first person to create the contract in this contract is considered the specific ULD owner.

Table 3. ULDwithIoT.sol

```
pragma solidity ^0.8.6;
contract ULDwithIoT {

        struct Switch {
                address addr;
                uint    endTime;
                bool    status;
        }

        address payable public owner;
        address public iot;
        bool public inUse;

        mapping (uint => Switch) public switches;
        uint public numPaid;
        uint public currentBalance;

        event NewUser(address indexed user);

        modifier onlyOwner() {
                require(msg.sender == owner);
                _;
        }

        modifier onlyIoT() {
                require(msg.sender == iot);
                _;
        }

        constructor(address _iot) {
            owner = payable(msg.sender);
            iot = _iot;
            numPaid = 0;
        }

        function payToSwitch() public payable {

            require(!inUse, "currently in use");
            require(msg.value == 300000000000000000);

        currentBalance += msg.value;

             Switch storage s = switches[numPaid++];
             s.addr = msg.sender;
             uint offsetInSeconds = 5;
             s.endTime = block.timestamp + offsetInSeconds;
             s.status = true;
             inUse = true;
             emit NewUser(msg.sender);

        }

        function updateStatus(uint _index) public onlyIoT {
            require(inUse, "device is not currently in use");

            Switch storage s = switches[_index];

           require(s.endTime != 0, "this switch does not exist");
            require(s.status == true, "switch is not currently use is");
            require(block.timestamp > s.endTime,
             "has not finished being used");
             switches[_index].status = false;
             inUse = false;
        }

        function withdrawFunds() public onlyOwner {
            uint balance = address(this).balance;
            require(balance > 0, "balance is not greater than 0");
            require(owner.send(balance),
             "could not send ether from contract to owner successfully");
            currentBalance = 0;
        }


        function kill() public onlyOwner {
                selfdestruct(owner);
        }
}
```

The require Solidity function check certain conditions that must be true before a function gets executed: inputs, state variables, and values returned from calls. For example, the first function in this contract uses 'require' to check whether the ULD is currently in use and confirms the ULD user's 3Ether payment. Once the ULD user pays:

- An event is emitted to let the ULD know that it can get used.
- 3Ether will remain locked in the contract until the ULD owner confirms the return value.
- Ensure ULD users can only use the ULD if it is not already in use.
- Only one person can use it at a time to avoid double payment.

In the second function, when the ULD service time gets finished, it updates the con-tract and makes as follows:

- The ULD is currently being used.
- The 'endTime' exists.
- 'only the IoT' modifier checks the message sender.
- The ULD needs to be currently used by the current switch.
- Set the status to no longer being used and the in-use to be false when the service time gets finished.

The third function verifies that the balance is more than zero and that the contract can successfully send Ether to the ULD owner to retrieve the contract balance. Finally, the ULD owner can terminate the agreement by using the last function. The source code gets shown in Table 3.

Deploy and Run: ULDwithIoT.sol

After creating a smart contract that reflects the primary process and business logic, the next step is to verify the operation of the smart contract through deployment and testing. Another purpose of deploying smart contracts is for users to use them in the blockchain. Deploying a smart contract requires sending a transaction containing the compiled smart contract's code without specifying a recipient. It is executed in an integrated development environment (IDE) that supports smart contract development and construction called Remix for compilation and deployment. The Remix is an integrated development environment that supports Solidity:

- Compilation and deployment: In 'File Explorer, a new file was created, pasted the contract into the editor, and named on the Remix side. Compiling was triggered after clicking the compile button in the 'Solidity Compiler. After compilation got completed, the next step moved to 'Deploy & Run Transactions' and deployed the contract. At this point, the environment dropdown changed to Injected Web3 to instruct Remix using MetaMask as an injected provider. After completed deploying, the deployed contract appears, as figure3 shows. Deploy means that the ULD owner announces on the blockchain network that ULD rental is possible. The deployment complete implies that users can check the ULD available on the blockchain and rent it. Since the contract can only be interacted with in Remix after deploying the agreement, a functionality test is available after the contract gets deployed.
- Verification of contract deployed: When the smart contract deployment gets completed, the main functions appear, as shown in Figure 4: Although there is a fixed period defined in the

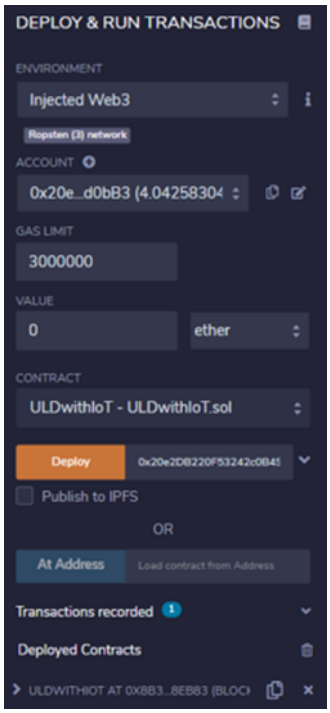

Figure 3: Deploy & Run Transaction.

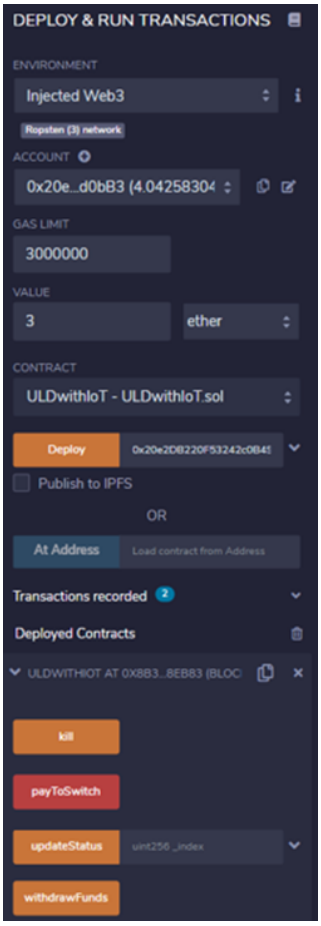

Figure 4: Main functions of ULDwithIoT.sol.

rental contract, there is a 'Kill' function for cancellation for specific reasons, 'payToSwitch' for pay rental cost, which activates IoT when the rental price paid, ‘update status to check ULD availability, and 'withdraw funds function to with-draw paid rental amount. According to the contract terms, the user has to pay 3 Ether to rent ULD, so the user pays 3 Ether using pay to switch. 3Ether gets paid through MetaMask linked to this smart contract. At the same time as payment of 3Ether is completed, other users cannot use the ULD.

- Figure 5 shows the information displayed after the user pays 3 Ether. 'CurrentBalance' confirms 3 Ether payments.

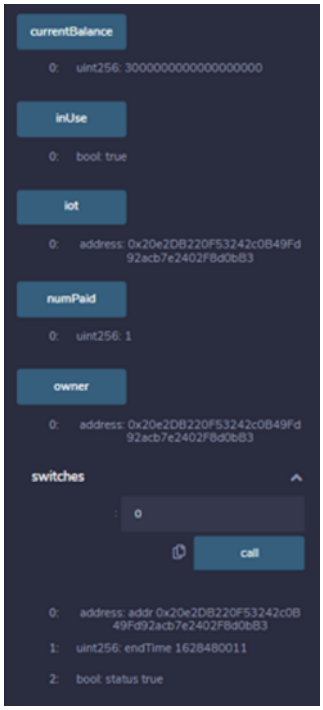

Figure 5: Main functions of ULDwithIoT.sol.

Etherscan allows view data about any pending or confirmed transactions on the Ethereum blockchain. The ULD users can check the details for the payment of 3 Ether on Etherscan as shown in figure 6. ' inUse' has an actual status, indicating that ULD can get used but displayed as false before payment gets made.

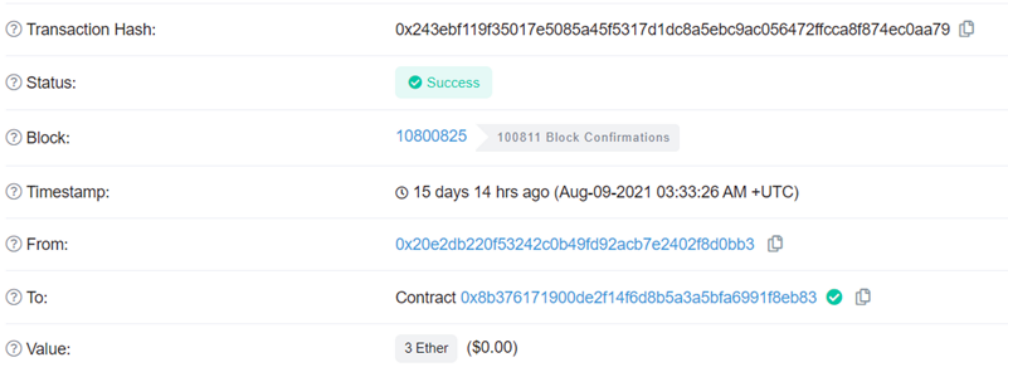

Figure 5: Main functions of ULDwithIoT.sol.

'iot' represents the IoT address of the ULD. 'numPaid' shows the number of rental costs paid, and 'owner' shows the address of the ULD owner. If the user calls switches, the user can check brief information, such as service period time, IoT address, and ULD usage status.

CONCLUSIONS

The blockchain and IoT-based ULD rental solution in this study results in accurate ULD inventory data management in any air cargo terminal worldwide that can grasp, and airlines can either use them for their aircraft or rent them out. Furthermore, this study proposes a blockchain answer utilizing smart contracts with IoT for evenly-distributed manning ULD rent and service data and processes. The blockchain ULD rental solution gets configured to set real-time ULD for rent, conditions, a settlement between participants in the ULD supply chain, and execution of contracts in conjunction with IoT. The most significant advantage of this system is that it moves away from the centralized management method and allows direct information exchange, payment, and operation between network participants in real-time.

The study result can be considered satisfactory as it shows that all participants can perform real-time data sharing and operation using smart contracts. The most significant advantage of this system is that each participant can perform accurate tasks in real-time by protecting against inaccurate, counterfeiting, and fraud data. Further-more, blockchain ULD inventory management protocols, control items, and procedures will be adopted and developed globally in the aviation sector. There are many other areas to discuss blockchain that this paper didn't address due to their broad range of topics. What this paper presents here are just significant facts and experiments for blockchain-based ULD rental solutions from a technical perspective.

References

Bikram, M., et al., Unit Load Devices (ULD) Demand Forecasting in the Air Cargo for Optimal Cost Management. Journal of Automation, Mobile Robotics, and Intelligent Systems, 14, 71-80, 2021, 10.14313/JAMRIS/3-2020/37.

https://www.iata.org/en/programs/cargo/unit-load-devices/

Arumugam, S., et al., IoT Enabled Smart Logistics Using Smart Contracts, 2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS), 2018, 1-6, DOI: 10.1109/LISS.2018.8593220.

Lim, J., The Concept of “Dizistics”- Stay Ahead of Freight Forwarders’ Business Model, International Journal of e-Education, e-Business, e-Management and e-Learning, 9, 331-337, 2019, 10.17706/ijeeee.2019.9.4.331-337.

Issaoui, Y., et al., Smart logistics: Study of the application of blockchain technology. Procedia Computer Science, Volume 160, 266-271, ISSN 1877-0509, 2019, https://doi.org/10.1016/j.procs.2019.09.467.

Rayes, A., et al.,Internet of Things from Hype to Reality, the Road to Digitization (1st Ed.), 293, 2016, Springer Publishing Company Incorporated, ISBN 978-3-319-99515-1.

Lu, H., et al., Safety stock estimation of unit load devices for international airline operations. Journal of Marine Science and Technology, 431-432, 2012, 20. 10.6119/JMST-011-0322-1.

https://www.uldcare.com/pdf/Proposal_Section_8.2.3__20130819.pdf

Diedrich, H., Ethereum, Blockchains, Digital Assets, Smart Contracts, Decentralized Autonomous Organizations, 30, 2021, Wildfire Publishing, ISBN 978-1-52393-049-0.

Emde, S., et al., Scheduling personnel for the build-up of unit load devices at an air cargo terminal with limited space. OR Spectrum 42, 397–426, 2020, https://doi.org/10.1007/s00291-020-00580-2.

Grincalaitis, M., Mastering Ethereum/Implement advanced blockchain applications using Ethereum-supported tools, services, and protocols, 63, 367,382, 2019, Packt Publishing Ltd, ISBN 978-1-78953-137-4.

Modi, R., Solidity Programming Essentials/A beginner's guide to building smart contracts for Ethereum and blockchain, 137-141, 2018, Packt Publishing Ltd, ISBN 978-1-78883-138-3.

Hacıoğlu, Ü., Digital Business Strategies in Blockchain Ecosystems Transformational Design and Future of Global Business, Transformational Design and Future of Global Business, 51, 2020, Springer Nature Switzerland AG, 10.1007/978-3-030-29739-8.

Bashir, Imran., Mastering Blockchain, Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 202, 2018, Packt Publishing Ltd, ISBN 978-1-78883-904-4.

Tripathy, R., et al., Next Generation Warehouse through disruptive IoT Blockchain. International Conference on Computer Science, Engineering and Applications (ICCSEA), 1-6, 2020, DOI: 10.1109/ICCSEA49143.2020.9132906.

Tran, H., et al., The Internet of Things for Logistics: Perspectives, Application Review, and Challenges. 7th IETE Technical Review, 2020, DOI: 10.1080/02564602.2020.1827308.

Patel, C., IoT privacy preservation using blockchain, Information Security Journal, A Global Perspective, 3, 2021, DOI: 10.1080/19393555.2021.1919795.

Jones, K., Blockchain in or as governance? Evolutions in experimentation, social impacts, and prefigurative practice in the blockchain and DAO space. Information Polity, 24, 1-18, 2019, 10.3233/IP-190157.

Antonopoulos., Wood G., et al., Mastering Ethereum/Building Smart Contracts and DApps, 70-71, 319-20, 2018, O'Reilly Media Inc, ISBN-13: 978-1491971949.

Alhejazi, M., et al., Enhancing the blockchain voting process in IoT using a novel blockchain Weighted Majority Concurrence Algorithm (WMCA), 2021, Information Security Journal: A Global Perspective, DOI: 10.1080/19393555.2020.1869356.

https://ethereum.org/en/developers/docs/concurrence-mechanisms/pos/#top

Navamani, T., A Review on Cryptocurrencies Security. Journal of Applied Security Research, 1-21, 2021, DOI: 10.1080/19361610.2021.1933322.

Heckler, N., et al., Crypto-Governance: The Ethical Implications of Blockchain in Public Service, Public Integrity, 1-16, 2020, DOI:10.1080/10999922.2020.1848106.

Alharby, M., et al., Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research, 2018, International Conference on Cloud Computing, Big Data and Blockchain (ICCBB), 1-6, DOI: 10.1109/ICCBB.2018.8756390.

Omar, I.A., et al., Ensuring protocol compliance and data transparency in clinical trials using Blockchain smart contracts. BMC Med Res Methodology 20, 224, 2020, https://doi.org/10.1186/s12874-020-01109-5.

Christidis, K., et al., Blockchains and Smart Contracts for the Internet of Things, in IEEE Access, 2021, 4, 2292-2303, DOI: 10.1109/ACCESS.2016.2566339.

Schmitt, G., et al.,Smart Contracts and Internet of Things: A Qualitative Content Analysis using the Technology-Organization-Environment Framework to Identify Key-Determinants. Procedia Computer Science, 160, 189-196, 2019, ISSN 1877-0509,https://doi.org/10.1016/j.procs.2019.09.460.

Pierro, G., et al., An Organized Repository of Ethereum Smart Contracts' Source Codes and Metrics, Future Internet, 12(11), 1, 2021, Gale Academic OneFile,link.gale.com/apps/doc/A645131126/AONE?u=tnlk&sid=bookmark AONE&xid=5dc50e33.