Hindawi
Security and Communication Networks
Volume 2021, Article ID 8937448, 10 pages
https://doi.org/10.1155/2021/8937448

Research Article

IoT-Based Autonomous Pay-As-You-Go Payment System with the Contract Wallet

Shinya Haga [1] **and Kazumasa Omote** [1,2]

[1]*University of Tsukuba, Tsukuba, Japan*
[2]*National Institute of Information and Communications Technology, Tokyo, Japan*

Correspondence should be addressed to Shinya Haga; s2120545@s.tsukuba.ac.jp

Received 10 June 2021; Revised 22 August 2021; Accepted 27 August 2021; Published 24 September 2021

Academic Editor: Chunhua Su

In recent years, increasingly many companies have entered the pay-as-you-go business because it has become easier to monitor services constantly due to the development and increase in the number of Internet of Things (IoT) devices. Research is in progress to introduce cryptographic assets into the payment, but if a private key is stolen, the cryptographic assets associated with it can be stolen. To address this issue, this paper proposes a secure automated payment system using contract wallets. This method ensures the security of cryptographic assets even if the private key is stolen. Secure automated payments are enabled by issuing transactions from IoT devices and using internal transactions to link contract wallets and smart contracts that handle the payment of cryptographic assets. Furthermore, the effectiveness of the proposed system is demonstrated on the Ethereum blockchain as a proof of concept, and the cost of gas is measured.

1. Introduction

In recent years, the effects of Internet of Things (IoT) devices on our lives have increased due to the rise of smart cities and smart homes. According to a survey conducted by Statista, Germany [1], the global penetration of IoT devices is expected to reach 50 billion units by 2030, and the market size will continue to grow thereafter. Consequently, increasingly many companies are offering pay-as-you-go services that provide income stability because it has become easier to monitor the operating status of machines and services in real time. In addition, many studies have been conducted on the use of smart contracts for the payment by IoT devices.

However, the wallets utilized for cryptographic asset payments are often managed in the hot wallet state, where the account is connected to the internet. If a private key is compromised and cryptographic assets are stolen, it is essentially impossible to recover the assets. The Coincheck incident [2] is said to have been a case of asset theft as the private key was stolen.

There are smart contracts known as contract wallets that minimize the loss of cryptographic assets caused by private key theft. A contract wallet is a smart contract for managing cryptographic assets securely; thus, users can securely hold these assets. If assets are deposited in the wallet, even if the private key of the wallet owner is stolen, the damage can be minimized because assets can be managed programmatically (e.g., an upper limit can be set on the amount of assets transferred). Typical contract wallets include Argent (Argent, "Argent—the best Ethereum wallet for DeFi," 2020, https://www.argent.xyz/), Dapper (Dapper Labs, "Dapper—your account manager for all things flow," 2020, https://www.meetdapper.com/), and Gnosis Safe(Gnosis, "Gnosis Safe: Overview," 2020, https://gnosis-safe.io/).

Contribution: traditionally, smart contracts and contract wallets have been independent of each other. In this study, we developed a system that enables secure automated payments with IoT devices by using internal transactions to link contract wallets and smart contracts that handle the payment of cryptographic assets. In addition, we demonstrated the effectiveness of our system on the Ethereum

blockchain and measured the cost of gas. Our evaluation results show that the proposed system can operate with a reasonable gas cost. This investigation represents the first attempt to propose and demonstrate such a secure payment system, as far as we know.

The remainder of this paper is organized as follows: Section 2 describes some preliminaries, and Section 3 presents related works. Section 4 provides a detailed explanation of our proposed architecture. Section 5 presents the results of implementing the proposed architecture and its evaluation. Section 6 provides a discussion on the security, privacy, and signature scheme of the proposed system. Finally, we conclude the paper in Section 7.

2. Preliminaries

2.1. Blockchain. Blockchain is a distributed ledger technology invented by S. Nakamoto [3] and is the core technology of many cryptographic assets, including Bitcoin. Blockchain stores data by grouping transactions into blocks and embedding the hash value of one block into the next block that occurs on a peer-to-peer (P2P) network.

The nodes put together a transaction, generate a block, and are rewarded. This sequence of events is called mining. In particular, Bitcoin and Ethereum have employed the proof-of-work (PoW) method to determine the mining node, which is time consuming. The main chain of the blockchain is defined as the longest one from the first block to the current block. Therefore, in order to falsify the data stored in a block, all subsequent blocks would have to be revoked, and thus, a large number of computations in the PoW would be necessary. Such falsification is practically impossible.

Blockchain is tamper resistant. In addition, public blockchains are highly transparent because anyone can participate in the network and can see the data stored in the blocks.

2.2. Smart Contract and Ethereum. A smart contract is a program that is recorded on a blockchain and is falsification resistant. When the program is executed, its state is also stored in the blockchain, ensuring the transparency of the sequence of actions. In addition, there is essentially no single point of failure as the program runs on a P2P network. Because of this characteristic, contracts can be automatically executed without the need for a third party.

Ethereum [4, 5] is the first cryptographic asset to employ smart contracts and enables smart contracts to be implemented in Solidity, Viper, LLL, and other languages. Solidity is a JavaScript-like high-level programming language with Turing completeness (albeit with gas limits on execution). Ethereum uses the elliptic curve digital signature algorithm (ECDSA) [6], where the address of an account is created by taking a hash (keccak256 [7]) of the public key. This account is called an externally owned account (EOA), and the user executes a function in the smart contract through the EOA. Addresses are also assigned to smart contracts, which are generated by encoding the deployed EOA and its nonce, called recursive length prefix encoding, and then taking a hash of them.

In order to execute a function in a smart contract, the EOA needs to issue a transaction, and a fee called gas is charged to the minor for that transaction. In general, the more complex the processing within a function is, the higher the gas value is. When a function in a smart contract references or executes a function in another smart contract, an internal transaction is issued.

2.3. Contract Wallet. A contract wallet is a smart contract for securely managing cryptographic assets in Ethereum. According to Angelo and Salzer [8], 21% of all smart contracts in the main chain of Ethereum are contract wallets.

The main functions of a contract wallet include controlling access to cryptographic assets, providing recovery from the loss of a private key, and designating whitelists. Basically, anyone with an EOA can deposit money into the account but cannot withdraw more than a particular amount unless certain conditions are met. This feature significantly reduces the risk of cryptographic asset theft, even if the private key of the wallet owner is stolen. The wallet also allows deposits to be made to whitelisted accounts without the requirement to do so, which increases the convenience of the wallet. However, some conditions must be met when designating an arbitrary account for the whitelist. In many cases, multisig is introduced as a condition for this purpose.

Typically, the EOAs of the wallet owner and organization that create and manage the wallet are the targets of the multisig users required for signatures, but some types of contract wallets allow individuals to designate their own trusted third parties. If it is suspected that the private key of the EOA that owns the contract wallet has been stolen, the private key can be updated by locking the wallet and contacting the owner of the trusted EOA designated to the multisig partner. By having the owner of the contract wallet designate a newly created EOA, the EOA whose private key is stolen loses access to the assets, and the wallet owner can protect the assets.

3. Related Work

Angelo and Salzer [8] analyzed the characteristics of contract wallets deployed in the Ethereum main chain and grouped them into six types. To this end, they presented approaches to identify contract wallets by analyzing the source code, bytecode, and execution traces extracted from transaction data. Moreover, they investigated usage scenarios and patterns. In addition, the following researchers have utilized smart contracts purposefully and have proposed various applications.

Wang et al. [9] attempted to solve the integrity problem, which is a key issue in Cloud storage auditing, by using blockchain technology and smart contracts. Traditional auditing schemes necessitate a third-party auditing and imply a limited pay-as-you-go service as they require clients to pay for services in advance. However, Wang et al. replaced third-party auditing with blockchain technology and

achieved fair payments by performing the payments through smart contracts.

Furthermore, Tam et al. [10] proposed a pay-as-you-go automotive insurance application based on smart contracts. The data are stored in a blockchain, making them tamper-proof and traceable, and premiums can be charged explicitly.

Subsequently, Yanqi et al. [11] proposed a secure brokerless pub/sub model using smart contracts and reputation systems [12]. This model combines these two elements and hybrid cipher based on Elgamal [13] and AES to satisfy the security requirements of confidentiality, authentication, scalability, integrity, fairness, and anonymity.

Finally, Thitinan and Nandar [14] proposed an emergency notification service by combining IoT devices and smart contracts. Using environmental data observed by IoT devices as triggers, this system enables reporting to public services such as hospitals and payment of bills through smart contracts.

However, these researchers did not address the proper course of action when the private key of a user EOA is stolen. Hence, this paper proposes a pay-as-you-go service system that takes into account the risks involved.

4. Proposed Architecture

4.1. Overview. We propose an autonomous pay-as-you-go payment system with secure management of assets using contract wallets. Implementing a multisig-compatible contract wallet for payments ensures that even if the private key of a user is stolen, his or her assets will not be compromised. The assumption is that the relevant company intends to develop a pay-as-you-go business while providing services to users using IoT devices. Figure 1 provides an overview of the proposed architecture. The architecture consists of the following seven components:

(1) Company X deploys a smart contract (i.e., a control contract) for calculating fees and controlling the status of the IoT

(2) User A requests to register an IoT device, the control contract, and the EOA of company X in his or her contract wallet

(3) The contract wallet organization registers the IoT device, control contract, and EOA of company X in the contract wallet of the user

(4) The IoT device refers to its own state regularly from the control contract

(5) The IoT device sends usage data to the contract wallet of the user

(6) The control contract calculates the fee from the usage data and returns the value to the contract wallet of the user

(7) The contract wallet of the user transfers the calculated fee to the EOA of company X

The contract wallet organization is a part of multisig, but it does not have strong authority by itself (e.g., the right to access user assets).

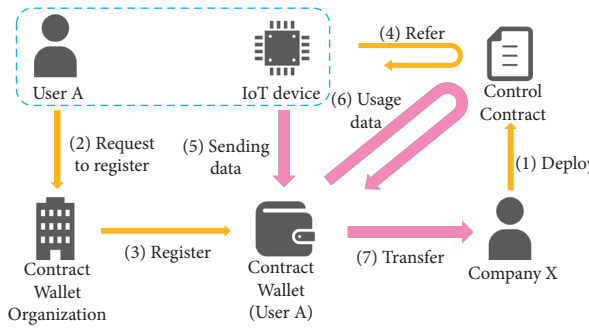


Figure 1: Entire flow of the proposed system.

4.2. Use Cases and Comparison with the Conventional Architecture. Use cases include automatic payments for lifelines such as water and electricity. Lifeline and pay-as-you-go tariffs are compatible as they are essential to life and are expected to be used continuously.

As Figure 2 shows, in pay-as-you-go automated payments using smart contracts, it has traditionally been necessary to deposit assets into a smart contract that exists for each IoT device. Consequently, as IoT devices are added to the system, the burden on the user increases as the amount of time to deposit increases. In the proposed architecture, a control contract for each IoT device is registered in its own contract wallet and is paid from the contract wallet, which greatly improves the convenience because there is no need to make a deposit for each IoT device.

4.3. System Model

4.3.1. User. The user has an EOA and can access his or her contract wallet by issuing a transaction. The addresses of the EOA and control contracts of the IoT devices distributed by the company are assumed to be known by the company. In addition, the user checks each time a transaction is issued to see if it is intended.

4.3.2. Company. The company is an organization that distributes and manages IoT devices and control contracts to users and has access to the control contracts. The company has an EOA or wallet address that the user designates as the destination for the assets and notifies the user in advance.

4.3.3. Contract Wallet Organization. This organization manages user-owned contract wallets and is a part of the multisig of the wallet. It is in charge of reviewing control contracts and updating contract wallets. The assets remain in the hands of the users and are not held by the contract wallet organization, so they are not centralized. Therefore, the organization does not have strong authority because it cannot access user assets by itself.

4.3.4. IoT Device. The IoT device periodically issues a transaction to send the translation of the provided service into usage data to the contract wallet. It also periodically refers to the state of the control contract to determine

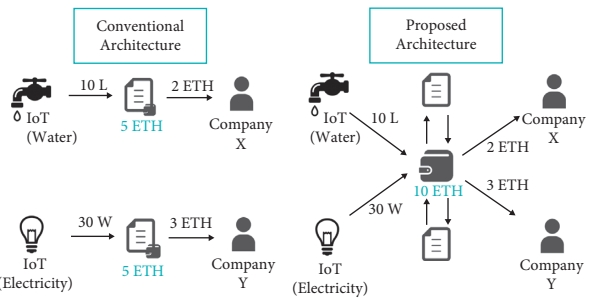


Figure 2: Comparison with the conventional architecture.

whether to turn the service on or off. In this implementation, we assume that the secret key of the EOA is hard coded into the IoT device. In addition, it is possible to issue transactions directly from IoT devices using the Web3API of Infura.

4.3.5. Contract Wallet. To store user assets securely, the user can access assets via multisig with the contract wallet organization. In this study, we adopted 2 of 2 multisig. In addition, only when data are sent from the registered IoT device is it possible to move the assets to the EOA of the whitelisted company without the need for a multisig.

4.3.6. Control Contract. The control contract calculates the fee based on the data sent from the contract wallet and returns the value to the contract wallet. It also has a function to turn the state parameters of the IoT device on and off. The source code must be made available to the public by the company.

4.4. System Flow. Figure 3 depicts the flow of automatic asset transfer from IoT devices. Note that transactions are denoted as Tx and internal transactions as InTx.

4.4.1. Whitelist Registration

(1) The user sends a pregiven EOA of the company to which the fees are to be sent and his or her signature with the corresponding private key to the contract wallet organization. This work is done outside the blockchain.

(2) The multisig counterpart of the user, the contract wallet organization, scrutinizes its EOA for trustworthiness.

(3) If reliable, the organization executes the function Whitelisting in the contract wallet and adds the EOA to the whitelist of the contract wallet under multisig.

4.4.2. Control Contract Registration

(1) The user sends the EOA of the IoT device used, the address of the associated control contract, and the EOA of the company to which the payment is made, along with his or her signature, to the contract wallet organization. This work is done outside the blockchain.

(2) The contract wallet organization similarly scrutinizes the control contracts published by the company.

(3) Then, if the control contract is trusted, it executes the function RegisterIoT in the contract wallet and registers the EOA of the IoT device with the EOA of the company, as sent by the user under multisig.

4.4.3. Transmission of Data from IoT Devices and Automatic Payment

(1) The IoT device periodically converts the services provided by the device into usage data, executes the contract wallet function IoTPayment, and transfers the data to the contract wallet

(2) The function IoTPayment executes the function Calculation in the control contract and obtains the fee value

(3) If the deposit has a value greater than the fee value obtained from the control contract, the cryptographic asset is sent to the designated EOA based on the fee value

(4) If the deposit has a value less than the fee value obtained from the control contract, the function Operation in the control contract is executed, and the state parameter of the IoT device is turned off

4.4.4. Stopping the IoT Device

(1) If a user notices something unusual activity, such as an unintended transaction being sent, the user contacts the company and asks the company to stop the IoT device

(2) When contacted, the company executes the function Operation in the control contract and sets the state parameters of the IoT device to off

4.5. Smart Contract. The proposed architecture assumes that the contract wallet has already been deployed on the blockchain, and the user registers the EOA and control contracts of the IoT devices to the contract wallet. The user signature is assumed to be a hash (keccak256) of his or her EOA signed by ECDSA with his or her own private key, as well as a hash (keccak256) of the combined addresses of the user EOA, EOA of the IoT device, and control contract.

The functions of control contracts and contract wallets are as follows (Algorithms 1–3 are contract wallets, and Algorithms 4 and 5 are functions in control contracts).

4.5.1. Whitelisting. This function whitelists the EOA of the company to which the charges are to be paid and is executed by the contract wallet organization at the request of the user. Specifying an EOA of the company allows for payment without any asset withdrawal limits. By verifying the

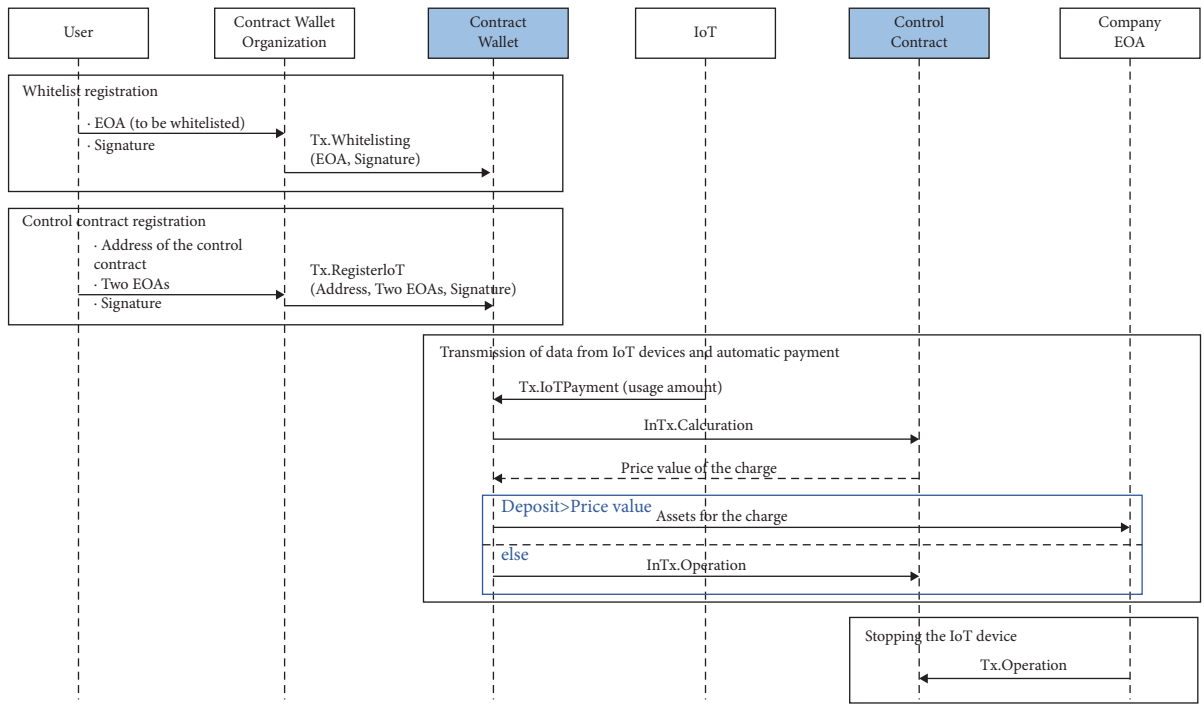


Figure 3: System flow.

Input: user EOA, company EOA, and user signature
Output: none
(1) Generate a hash (keccak256) of the company EOA
(2) Decrypt the public key and EOA of the signer from the hash and user signature using ECDSA
(3) **if** decrypted EOA = = user EOA **then**
(4) Add company EOA to the whitelist
(5) Emit the event
(6) **else**
(7) Error handling
(8) **end if**

Algorithm 1: Whitelisting.

Input: user EOA, IoT EOA, address of the control contract, company EOA, and user signature
Output: none
(1) Generate a hash (keccak256) of the user EOA, company EOA, and control contract address, concatenated in that order
(2) Decrypt the public key and EOA of the signer from the hash and user signature using ECDSA
(3) **if** decrypted EOA = = user EOA **then**
(4) Tie the IoT EOA to the control contracts and company EOA and insert it into the IoT registration list
(5) Emit the event
(6) **else**
(7) Error handling
(8) **end if**

Algorithm 2: RegisterIoT.

Input: usage amount
Output: none
(1) **if** a transaction issuer in the IoT registration list **then**
(2) Run the function **Calculation** in the control contract tied to the issuer EOA
(3) **if** determine if the value obtained by function **Calculation** is greater than the value of deposit **then**
(4) Transfer assets for the value obtained from the calculation to the company EOA, which is linked to the issuer EOA
(5) Emit the event
(6) **else**
(7) Run the function **Operation** in the control contract
(8) **end if**
(9) Error handling
(10) **end if**

Algorithm 3: IoTPayment.

Input: usage amount
Output: charge value
(1) Calculate the charge value from usage amount
(2) **return** value

Algorithm 4: Calculation.

Input: none
Output: none
(1) **if** issuer EOA = = the company EOA **then**
(2) Convert the IoT on/off state
(3) **else**
(4) Error handling
(5) **end if**

Algorithm 5: Operation.

Table 1: Execution environment.

Name	Version
Arduino-IDE	1.8.13
Solidity	0.5.17

signatures sent outside of the blockchain by the user within this function, together with the verification of the transactions themselves, we achieve 2 of 2 multisig.

4.5.2. RegisterIoT. This function associates the EOA of an IoT device with the EOA of the company and address of the control contract and registers them to the IoT registration list. It is executed by the wallet organization at the request of the user. By registering to the IoT registration list, the IoT device can execute the function IoTPayment. This function is executed in multisig in a similar way to the function Whitelisting.

4.5.3. IoTPayment. This function executes asset payment based on the usage data given by the IoT device. It checks whether the EOA of the IoT device is registered in the contract wallet. If so, the function obtains the charge value by executing the function Calculation in the control contract and then transfers the cryptographic assets of the value to the EOA of the company to which it is linked. If the deposit is less than a given charge value, the function Operation in the control contract, which switches the service provided by the IoT device on and off, is executed.

4.5.4. Calculation. This function is executed internally by the function IoTPayment. It computes the fees from the given usage data and returns the value to the contract wallet.

4.5.5. Operation. This function manages the status of the IoT devices and changes the variables that control the turning on and off of the IoT device services. It cannot be executed by anyone other than the company unless the private key of the company is obtained because it checks if the EOA of the transaction issuer is the EOA of the company or contract wallet.

5. Implementation and Evaluation

We demonstrated the feasibility of the proposed architecture by implementing a prototype of the proposed method in

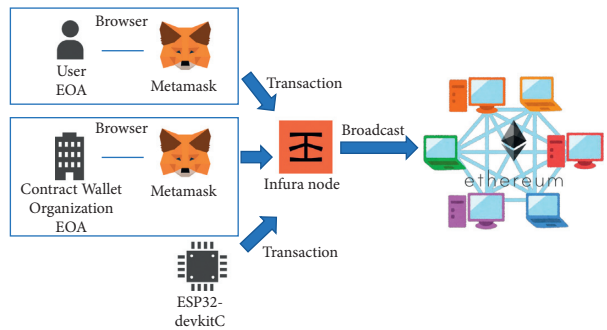
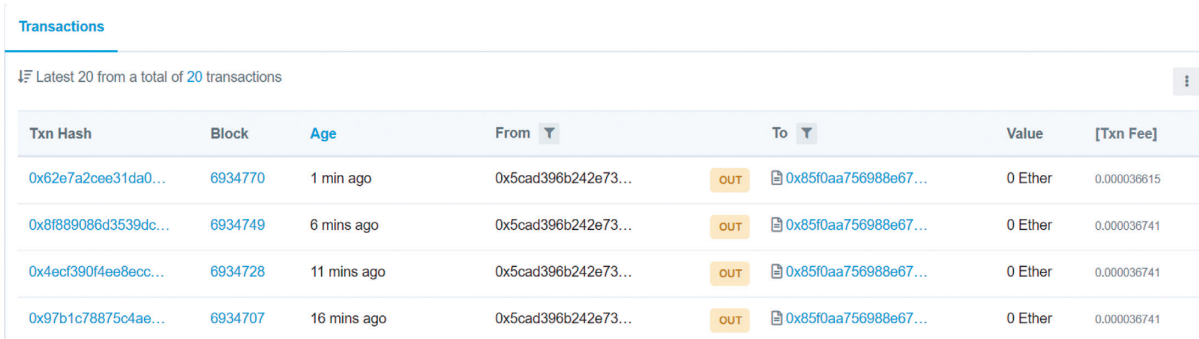


Figure 4: System components.

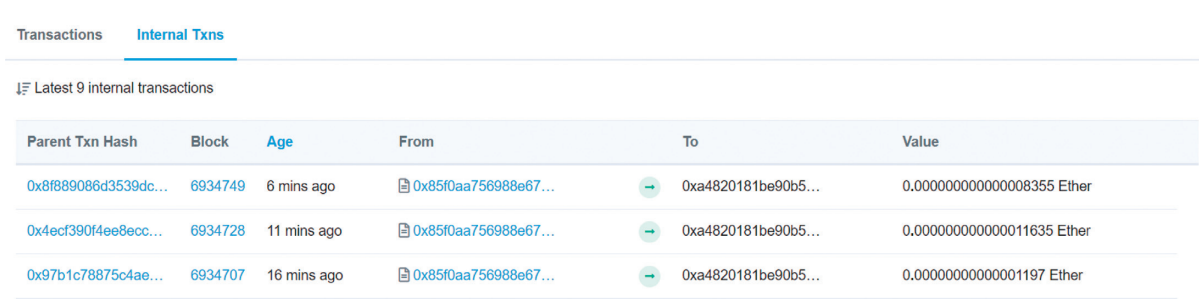


Transactions

Latest 20 from a total of 20 transactions

Txn Hash	Block	Age	From		To	Value	[Txn Fee]
0x62e7a2cee31da0…	6934770	1 min ago	0x5cad396b242e73…	OUT	0x85f0aa756988e67…	0 Ether	0.000036615
0x8f889086d3539dc…	6934749	6 mins ago	0x5cad396b242e73…	OUT	0x85f0aa756988e67…	0 Ether	0.000036741
0x4ecf390f4ee8ecc…	6934728	11 mins ago	0x5cad396b242e73…	OUT	0x85f0aa756988e67…	0 Ether	0.000036741
0x97b1c78875c4ae…	6934707	16 mins ago	0x5cad396b242e73…	OUT	0x85f0aa756988e67…	0 Ether	0.000036741

Figure 5: Transaction from the IoT device.

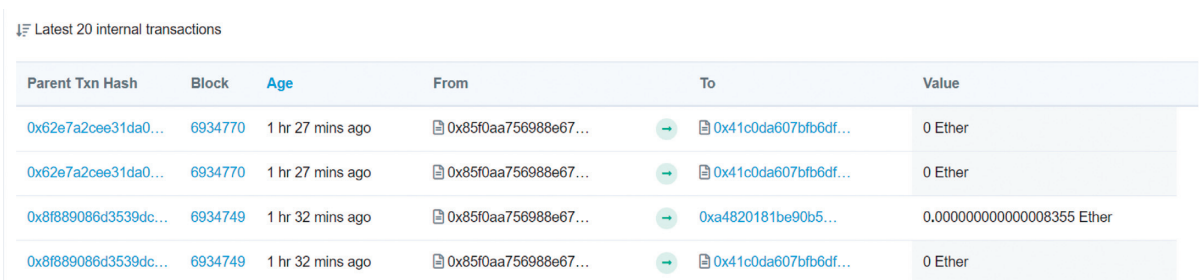


Transactions | Internal Txns

Latest 9 internal transactions

Parent Txn Hash	Block	Age	From		To	Value
0x8f889086d3539dc…	6934749	6 mins ago	0x85f0aa756988e67…	→	0xa4820181be90b5…	0.000000000000008355 Ether
0x4ecf390f4ee8ecc…	6934728	11 mins ago	0x85f0aa756988e67…	→	0xa4820181be90b5…	0.000000000000011635 Ether
0x97b1c78875c4ae…	6934707	16 mins ago	0x85f0aa756988e67…	→	0xa4820181be90b5…	0.000000000000001197 Ether

Figure 6: Internal transaction from the contract wallet.



Latest 20 internal transactions

Parent Txn Hash	Block	Age	From		To	Value
0x62e7a2cee31da0…	6934770	1 hr 27 mins ago	0x85f0aa756988e67…	→	0x41c0da607bfb6df…	0 Ether
0x62e7a2cee31da0…	6934770	1 hr 27 mins ago	0x85f0aa756988e67…	→	0x41c0da607bfb6df…	0 Ether
0x8f889086d3539dc…	6934749	1 hr 32 mins ago	0x85f0aa756988e67…	→	0xa4820181be90b5…	0.000000000000008355 Ether
0x8f889086d3539dc…	6934749	1 hr 32 mins ago	0x85f0aa756988e67…	→	0x41c0da607bfb6df…	0 Ether

Figure 7: Company EOA receives assets.

Table 2: Gas cost.

Transaction	Gas cost	Transaction fee
Whitelisting	29,247 gas	734977.11 Gwei
RegisterIoT	84,047 gas	2112101.11 Gwei
IoTPayment (normal)	36,741 gas	923301.33 Gwei
IoTPayment (insufficient funds)	36,615 gas	920134.95 Gwei

Table 3: Address.

Characteristic	Address
Company EOA	0xa4820181BE90b5570Ce03bbF605b65796d6B6bBA
IoT device EOA	0x5CAd396b242E73BC9f4827F33aa45fdc73061801
Contract wallet address	0x85f0aA756988E67fE64baaB613C3aA896AA5Ac84
Control contract address	0x41c0Da607bfB6DF20dB79D49d27F6D94c3cdedE9

Rinkeby, an Ethereum test network, using MetaMask and Infura. The device and smart contracts were written in Arduino language and Solidity language, respectively.

5.1. System Components

5.1.1. ESP32-DevKitC (1 Machine). This component plays the role of an IoT device. In this implementation, instead of actually measuring the services provided by the IoT devices and transferring the usage data, random integers are utilized as usage data. The development environment for IoT devices is the Arduino-IDE. The CPU is an Xtensa dual-core 32 bit LX6 microprocessor. The memory is 520 KiB (SRAM). The capacity is 4 MiB (flash memory).

5.1.2. Infura. Infura is the node-hosting service of Ethereum. Users can join the Ethereum blockchain through this service without the need to set up a full Ethereum node. Because the JSON-RPC API of Web3 is available, transactions can be issued from IoT devices via http or WebSocket.

5.1.3. MetaMask. This component is a Google Chrome extension, and the service facilitates EOA management on the browser. It is possible to specify the node to be used, and an Infura node is utilized in this implementation.

Table 1 describes the execution environment, and Figures 4 and 5 show the overall configuration diagram.

Figure 6 indicates that an internal transaction is issued from the contract wallet. Here, the contract wallet executes the function Calculation in the control contract and receives the value.

Figure 7 Demonstrates that the assets are transferred from the contract wallet to the company EOA.

These observations confirm that our proposed architecture involving automatic assets' transfer works properly.

Table 2 lists the gas cost incurred for each executed function, and Table 3 provides the addresses of the IoT, company, and contract wallet.

There are 51 and 25 lines of code for the contract wallet and control contract, respectively. The lines of code were measured using the cloc command on macOS Catalina 10.15.7. The gas price was set to 25.13 Gwei, which is the average in the mainnet on July 1, 2021.

5.2. Experiment. ESP32 executed the contract wallet function IoTPayment every 5 min through Infura using random values between 1000 and 2000. To confirm the transactions, we utilized a website called Etherscan (Etherscan, "Etherscan," 2020, https://etherscan.io/), which enabled us to check transactions in Ethereum. Figure 7 presents the transaction records for each of the addresses identified on this site.

Figure 5 shows that the transaction is executed in roughly 5 min.

6. Discussion

6.1. Security Analysis. The possible attacks on our proposed system can be classified into the following four patterns. We discuss them in terms of "purpose of attack," "assumed attack means," and "countermeasures."

6.1.1. Attack from a Third Party to the Private Key of the IoT. The purpose of this attack is for a third party to impersonate the IoT device and force the user to pay an unusual fee.

As a means of attack, by stealing the private key in the IoT in some way, an attacker can issue a transaction with the free value as the amount used by the user.

The countermeasure would be that the contract wallet organization would monitor the blockchain and notify users whenever a transaction related to their EOA is issued. This notification system enables users to provide notice immediately when an unintended transaction is issued. The user can then contact the company to stop the service and request the return of the moved assets. In addition, because the destination of the assets is specified in the contract wallet as the EOA of the company, merely stealing the private keys of the IoT device will not allow the assets to be moved to the wallet controlled by the attacker.

6.1.2. Attack from a Third Party to the Private Key of the Users. The purpose of this attack is for a third party to impersonate the owner of the contract wallet and attempt to access the contract wallet.

As a means of attack, by stealing the private key of the user in some way, an attacker can attempt to issue a transaction that transfers the assets in the contract wallet to the attacker's account.

Against this attack, the contract wallet can minimize the damage by having most of the user's assets moved to the contract wallet. This is because contract wallets are designed so that even the owner of the contract wallet can restrict the withdrawal limit. Therefore, if the owner moves many assets from his EOA to his contract wallet, the attacker cannot move all assets pooled in the contract wallet even if he has the owner's private key. To remove the restriction, a digital signature with the owner's private key and the private key of the contract wallet management organization must be verified in the contract wallet. In addition, when an asset withdrawal occurs, the transaction is issued, and a notification is sent to the user from the contract wallet organization so that he or she is made aware of the unintended transaction. Moreover, by designating a newly created EOA as the owner of the contracted wallet under the multisig of the previous EOA of the user and the EOA of the contract wallet organization, the attacker will lose access to the wallet.

6.1.3. Attack from the User Side. The purpose of this attack is for the user to receive services but not to pay for them.

It is assumed that the user will pay for the assets used in the period at a fixed time. As a means of attack, by reducing the amount of assets in the contract wallet in advance, the user receives more services than he or she can pay for with his or her assets by the time the next payment is made (e.g., even if the assets in the contract wallet are set to zero, the user will still receive the service until the next payment).

The attack can be handled by preparing a security deposit, which we did not implement in this study. If there are not enough assets in the contract wallet to pay at the time of payment, the benefit to the user side can be reduced by collecting the deposit.

6.1.4. Attack from the Company Side. The purpose of this attack is for the company to receive fees but not to provide the service.

This type of attack is virtually impossible because the program is written in the smart contracts in such a way that users are charged and paid for the amount they use.

6.2. Privacy Issue. The proposed method has an issue that the data are exposed to the public chain without encryption. However, this is not severe because the EOA of the user is generally not linked to the user's personal information (e.g., users of cryptographic assets accept that the remittance history is public).

To solve this problem, instead of using a public blockchain, a consortium or private blockchain can be employed. Either of these blockchains can be used to partially restrict the disclosure of information; however, the possibility that the blockchain nodes may act fraudulently cannot be eliminated. Therefore, it is necessary to consider this aspect.

6.3. Private Key Distribution Using the Threshold Signature. The proposed architecture uses a multisig contract wallet to distribute the access rights to the assets to achieve a secure payment method, but there is also a method to distribute the private key itself [15, 16] by using secret sharing [17]. This method is called the threshold signature scheme (TSS) (Binance, "Threshold Signatures Explained," 2020). By applying this signature scheme to a wallet of cryptographic assets, it is possible to create a decentralized wallet.

The main difference between contract wallets and TSS-based wallets is that the former are implemented on smart contracts, whereas the latter are implemented as ordinary software outside the blockchain. The secret share of a private key is normally distributed among various devices. By collecting a threshold number of secret shares synchronously with its devices, TSS-based wallets can decrypt the private key and issue transactions. Therefore, we can distribute the privileges by allowing others to manage the secret share, similarly to multisig.

However, TSS is a relatively new technology, and its communication protocol is more complex than that of other public key cryptographic methods. We are in the process of developing new signature methods, including previously undiscovered attacks. Therefore, TSS is a promising technology, although there is scope for further study.

7. Conclusion

In this study, we implemented and evaluated a pay-as-you-go automatic payment system using contract wallets to reduce the risk of cryptographic asset theft due to private key theft. We implemented a contract wallet and control contract using the Solidity language and connected them by utilizing internal transactions. We have also implemented ESP32 using the Arduino language to demonstrate that the proposed method is feasible as it allows the IoT devices to issue transactions and reference states from the blockchain. This ESP32 only keeps the private key to issue transactions. In addition, we recorded the transactions that occurred and measured the gas cost. In the proposed architecture, the control contracts can be freely programmed to be more complex, but it is necessary to account for the increased gas cost.

In future work, we plan to address privacy issues such as encryption and to consider ways to reduce the gas cost by using the TSS and other methods. We will also consider more deeply about the key management of IoT devices.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Grant-in-Aid for Scientific Research (B) (19H04107).

References

[1] Statista, "Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030," [Online]. Available: https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/, 2019.

[2] Fortune Media Group Holdings, "How to steal $500 million in cryptocurreny," [Online]. Available: https://fortune.com/2018/01/31/coincheck-hack-how/, 2018.

[3] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," [Online]. Available: http://www.bitcoin.org/bitcoin.pdf, 2009.

[4] Ethereum, "A next-generation smart contract and decentralized application platform," [Online]. Available: https://ethereum.org/en/whitepaper/, 2021.

[5] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger petersburg version fa00ff1 – 2021-05-12," [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf, 2021.

[6] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[7] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak, advances in cryptology - eurocrypt 2013," in *Proceeding of the Annual international conference on the theory and applications of cryptographic techniques*, pp. 313-314, Springer, Athens, Greece, May 2013.

[8] M. D. Angelo and G. Salzer, "Characteristics of wallet contracts on ethereum," in *Proceeding of the 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, Sepember 2020.

[9] H. Wang, H. Qin, and M. Zhao, "Blockchain- based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520300621.

[10] V. H. Tam, M. Lenin, M. Mukesh, and A. Ermyas, "Blockchain-based data management and analytics for micro-insurance applications," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2539–2542, Beijing, China, November 2017.

[11] Z. Yanqi, L. Yannan, M. Qilin, Y. Bo, and Y. Yong, "Secure pub-sub: blockchain-based fair payment with reputation for reliable cyber physical systems," *IEEE Access*, vol. 6, pp. 12295–12303, 2018.

[12] K. Bok, J. Yun, Y. Kim, J. Lim, and J. Yoo, "User reputation computation method based on implicit ratings on social media," *TIIS*, vol. 11, no. 3, pp. 1570–1594, 2017.

[13] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[14] T. Thitinan and A. Y. Nandar, "Emergency service for smart home system using ethereum blockchain: system and architecture," in *Proceeding of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 888–893, IEEE, Kyoto, Japan, March 2019.

[15] D. Jack, K. Yashvanth, L. Eysa, and S. Abhi, "Threshold ecdsa from ecdsa assumptions: the multiparty case," in *Proceeding of the IEEE Symposium on Security and Privacy (SP)*, pp. 1051–1066, IEEE, San Francisco, CA, USA, May 2019.

[16] G. Rosario and G. Steven, "Fast multiparty threshold ecdsa with fast trustless setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1179–1194, New York, NY, USA, October 2018.

[17] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.