コントラクトウォレットを用いたIoT機器による暗号資産従量課金制の自動決済システム

芳賀 慎也[1,a)]　面 和成[1,2]

概要：近年 IoT 機器の発展，数の増大により，サービスを常時監視することが容易になったことで従量課金制のビジネスに参入する企業が増えてきている．その決済に暗号資産を導入する研究も進んでいるが，秘密鍵を奪取された場合，それに紐づく暗号資産が盗難される恐れがある．本稿では，コントラクトウォレットとサービスを連携させることで，たとえユーザーが秘密鍵を漏洩させたとしても，安全な自動決済の仕組みを維持できるシステムを提案する．インターナルトランザクションを利用し連携させることで，IoT 機器から無人のスマートコントラクト同士による安全な自動決済を可能にした．さらに，本システムを Ethereum ブロックチェーン上で実証し，gas コストの計測も行った．

キーワード：暗号資産，ブロックチェーン，スマートコントラクト，コントラクトウォレット

Cryptographic Assets Mater Charging Payment System with IoT Device using Contractwallet

Shinya Haga[1,a)]　Kazumasa Omote[1,2]

Abstract:
In recent years, more and more 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 IoT devices. Research is in progress to introduce cryptographic assets into the settlement, but if the private key is stolen, the cryptographic assets associated with it are stolen.In this paper, we propose a system that integrates a Contractwallet and a service to maintain a secure automated payment system, even if the user compromises the private key. By using internal transactions, we have made it possible for smart contracts to make secure and automatic payments to each other from IoT devices. Furthermore, we have demonstrated our system on the Ethereum blockchain and measured the cost of gas.

Keywords: Cryptographic assets, Blockchain, Smartcontract, Contractwallet

1. はじめに

近年，スマートシティやスマートホームの台頭により，IoT (Internet of Things) 機器の社会に及ぼす影響は以前に増して大きくなりつつある．IDC の調査 [1] によると，全世界の IoT 機器の普及台数が 2025 年には 416 億台に達するとされており，その後も市場規模は成長する見通しである．それに伴い，機械やサービスの稼働状況をリアルタイムに監視することが容易になったため，安定した収入を得ることができる従量課金制のサービスを提供する企業も増えている．加えて、IoT による支払いにスマートコントラクトを導入する研究も多くある．

しかし、現在において暗号資産の支払いに使用するウォレットはインターネットにつながっていて秘密鍵でアカウントを管理する，いわゆるホットウォレットの状態で管理

[1] 筑波大学
University of Tsukuba
[2] 情報通信研究機構
National Institute of Information and Communications Technology, Japan
[a)] s1711390@s.tsukuba.ac.jp

することが多い．故に秘密鍵が奪取され，暗号資産が盗難にあった場合，その資産を取り戻すことは基本的に不可能になる．コインチェック社の暗号資産 XEM 流出事件も秘密鍵が奪取されて，資産の盗難の被害にあったと言われている．

一方で、秘密鍵盗難のリスクを最小限に抑えるためにコントラクトウォレットと言われるスマートコントラクトが存在する．コントラクトウォレットとは，暗号資産の管理を目的としたコントラクトであり，暗号資産を安全に保持することができる．コントラクトウォレットに資産を預けておけば，コントラクトウォレットの所有者の秘密鍵が奪取されたとしても，送金金額の上限設定など，プログラムで制御できるため，被害を最小限に抑えることができる．代表的なウォレットの例としては，Argent [2]，Dapper [3]，Gnosis Safe [4] などが挙げられる．

本稿では，コントラクトウォレットとサービスを連携させることで，たとえユーザーが秘密鍵を漏洩させたとしても，安全な自動決済の仕組みを維持できるシステムを提案する．従来，サービスと暗号資産の管理を司るコントロールコントラクトとコントラクトウォレットは独立して，互いに関わることはなかった．本稿では，インターナルトランザクションを利用し連携させることで，IoT 機器から無人のスマートコントラクト同士による安全な自動決済を可能にした．さらに，本システムを Ethereum ブロックチェーン上で実証し，gas コストの計測も行った．なお，このようなシステムの提案及び実証は我々が知る限り初めての試みである．

2. 準備

2.1 ブロックチェーン

ブロックチェーンとは，サトシ・ナカモトにより考案された分散型台帳技術であり，このアルゴリズムは Bitcoin を含む多くの暗号通貨の中核をなす技術である．ブロックチェーンは P2P ネットワーク上で発生したトランザクションをブロックにまとめ，そのブロックのハッシュ値を次のブロックに埋め込むことで，鎖の連なりのようにデータを保持する．トランザクションをまとめ，ブロックを生成することができたノードは報酬を得ることができ，その一連の流れをマイニングという．マイニングできるノードは一つと定めれており，そのノードを決める方法として Bitcoin，Ethereum 等では時間のかかる計算を成功したものがマイニングできる PoW (Proof of Work) を採用している．ブロックチェーンの主鎖は最初のブロックから現在のブロックまでの最長のものと定められている．そのため，ブロックに格納されたデータを改竄するためには，それ以降のブロックを全て破棄し，それまで PoW で計算されてきた膨大な計算を行わなくてはならないため，現実的に不可能とされている．故にブロックチェーンは耐改竄性を持つと言われる．

また，パブリックブロックチェーンにおいては，ネットワークに誰でも参加することができ，ブロックに格納されているデータを見ることができるので透明性が高いとも言われる．

2.2 スマートコントラクトと Ethereum

スマートコントラクトとは，ブロックチェーン上に記録されているプログラムであり耐改竄性を持つ．そのプログラムが実行されるとその記録もブロックチェーンに格納されるため，一連の動作の透明性を確保することができる．また，P2P ネットワーク上で実行されるため，単一障害点となるものは基本的に存在しない．この特性故に，第三者機関を介さずに，任意の契約を自動で実行できると言われている．

スマートコントラクトを最初に採用した暗号資産として Ethereum があり，Ethereum ではスマートコントラクトを Solidity，Viper，LLL などの言語で実装できるが，本稿では，Javascript ライクな高級言語でチューリング完全である Solidity を使用する．

Ethereum は ECDSA を採用しており，そのアカウントのアドレスは，公開鍵のハッシュ値をとるなどすることで作られる．このアカウントは EOA (Externally Owned Account) と言われ，ユーザーは EOA を通してスマートコントラクト内の関数を実行する．また，スマートコントラクトにもアドレスが割り当てられ，こちらは，デプロイした EOA とそのナンスを RLP エンコーディンと言われる符号化を施し，そのハッシュをとることで生成される．

スマートコントラクト内の関数を実行するためには，EOA がトランザクションを発行する必要があり，その際に gas と言われる手数料がマイナーに徴収される．一般的に，関数内の処理が複雑であるほど，gas は高くなる．また，スマートコントラクト内の関数が外部のコントラクトの関数を参照，実行する場合は，インターナルトランザクションが発行される．

2.3 コントラクトウォレット

コントラクトウォレットとは，Ethereum での暗号資産を安全に管理することを目的としたスマートコントラクトである．Monika ら [5] によると，Ethereum のメインチェーンにおけるスマートコントラクト全体の 21%がコントラクトウォレットと言われている．

コントラクトウォレットの主な機能としては，暗号資産へのアクセス制御，秘密鍵紛失時のリカバリー機能，ホワイトリストの指定などがあげられる．基本的な使い方として，入金する際は EOA を持つ人なら誰でも入金できるが，引き出す際は，何らかの条件を満たさない限り一定の金額以上を引き出すことができない．この機能により，ウォ

レットの所有者の秘密鍵が盗まれた場合でも，暗号資産の盗難のリスクを大幅に下げることができる．また，ホワイトリストに指定されているアカウントに対する入金はその条件なしでもできるようになっており，ウォレットとしての利便性もあげている．ただし，ホワイトリストに任意のアカウントを指定する際には，何らかの条件を満たさなければならない．その条件としてはマルチシグをが導入されている場合が多い．

一般的に署名に必要なマルチシグの対象としてウォレットの所有者である EOA とウォレットを作成，管理する組織の EOA があげられるが，コントラクトウォレットの種類によっては自分で信頼できる第三者を指定することができる．もしコントラクトウォレットの所有者である EOA の秘密鍵が盗まれた可能性がある場合は、ウォレットをロックし、マルチシグの相手に指定した信頼できる EOA の所有者に連絡をとることで，ウォレットの所有者となる EOA を更新することができる．そこにコントラクトウォレットの所有者が新たに作成した EOA を指定してもらうことで，秘密鍵が盗難された EOA は資産へのアクセスを失い，ウォレットの所有者は資産を守ることができる．

3. 関連研究

Hoang らの研究 [6] では，スマートコントラクトを用いた従量課金制の自動車保険アプリケーションを提案している．データをブロックチェーンに保存することでデータの改竄防止と追跡が可能になり、保険料を明示的に請求できる。

Yanqi らの研究 [7] では，スマートコントラクトと信頼性評価のシステムを用いてセキュアなブローカーレス Pub/Sub モデルを考案している．ブローカーレス Pub/Sub モデルでは，スマートコントラクトと評価システムを組み合わせ，セキュリティ要件である，機密性，認証，スケーラビリティ，完全性，公平性，匿名性を満たすことを実現している．

Thitinan らの研究 [8] では，IoT 機器とスマートコントラクトを組み合わせて，緊急時の通報サービスを提案している．IoT 機器で観測した環境データをトリガーにスマートコントラクトを介して，病院などの公共サービスへの通報，料金の支払いまでを実現している．

しかし，これらの研究はユーザーの EOA の秘密鍵が盗難にあった際の対応については言及されていない．本稿では，そのリスクを踏まえた従量課金制のサービスシステムを提案する．

4. 提案手法

4.1 概要

コントラクトウォレットを使用することで，資産を安全に管理しながら従量課金制の自動決済を行えるシステムを提案する．企業は IoT 機器を使用してユーザーにサービスを提供しつつ，従量課金制のビジネスを展開しようと考えていることとする．提案手法の概要は図 1 に示す．提案手法は次の 4 つから構成される．

(1) 資産の送信先である企業の EOA をコントラクトウォレットのホワイトリストに登録
(2) 料金の計算や IoT 機器の状態を制御するコントロールコントラクトをコントラクトウォレットに登録
(3) IoT 機器からデータの送信と支払い
(4) 企業による IoT 機器の停止

具体的な流れは以下の通りである．まず，資産の送信先である企業の EOA を自分が所有するコントラクトウォレットのホワイトリストに登録する．次に，企業によって公開されたコントロールコントラクトと IoT 機器，企業のアドレスを紐づけてコントラクトウォレットに登録する．IoT 機器が定期的にデータをコントラクトウォレットに送信し，コントロールコントラクトを通して料金を計算，その料金を自動的に指定されている企業の EOA に送金する．また，何らかの原因で IoT 機器の機能を停止したい場合は，企業がコントロールコントラクトにアクセスし，状態を off にするようにする．

前提として，IoT 機器は故障せずに正常に動作し続けるとし，この前提の上で IoT 機器がコントロールコントラクトの状態パラメータに従うために定期的にこのコントラクトにアクセスし，常に状態を把握しているものとする．

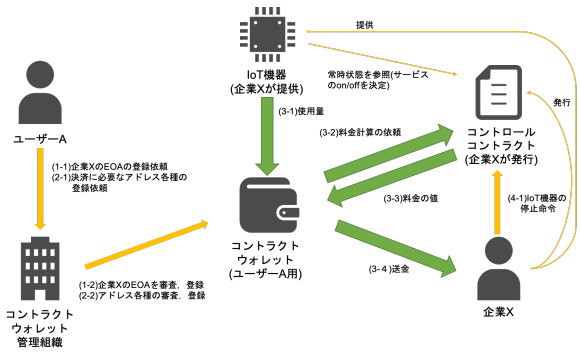


図 1 提案手法の全体図

4.2 システムモデル

- ユーザー
  ユーザーは EOA を持ち，トランザクションを発行することで自身のコントラクトウォレットにアクセス可能である．また，企業から配布される IoT 機器の EOA とコントロールコントラクトのアドレスは企業から知らされているものとする．
- 企業
  ユーザーに対して，IoT 機器やコントロールコントラクトを配布，または管理する組織であり，コントロー

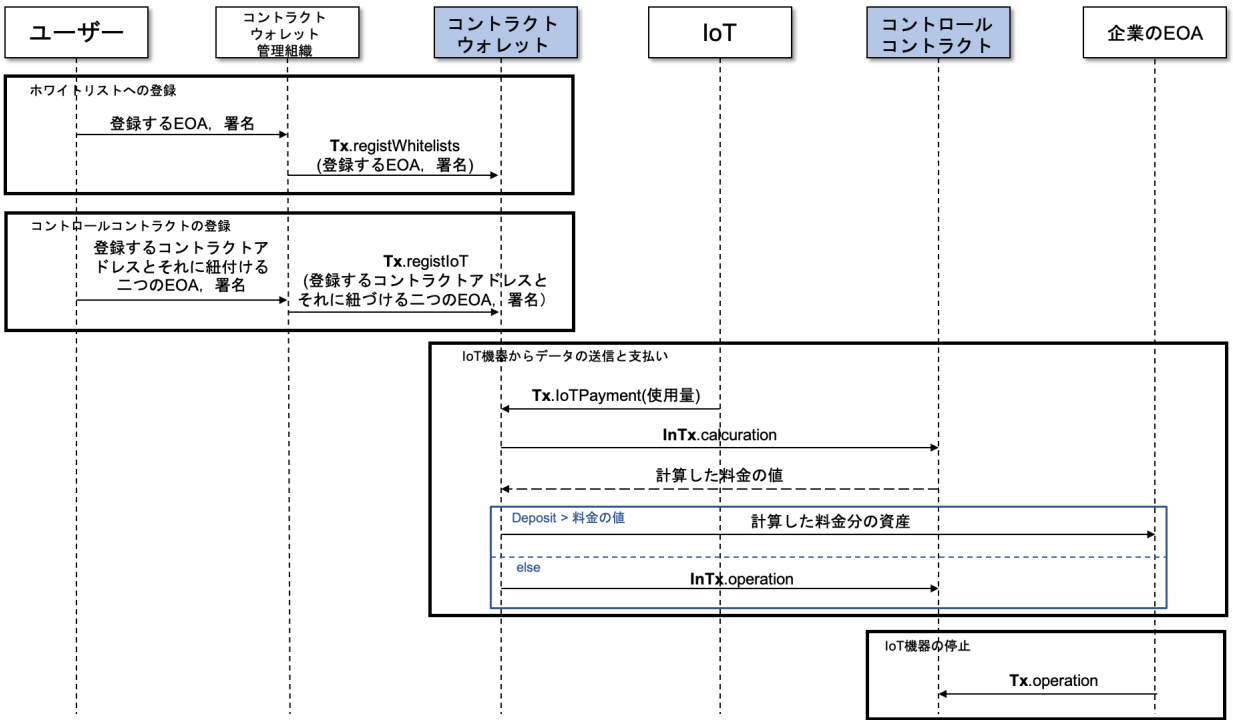


図 2　提案手法の流れ

ルコントラクトへのアクセス権限を持つ．企業側はユーザーが資産の送信先として指定する EOA またはウォレットのアドレスを持ち，これも事前にユーザーに通知しておく．

- コントラクトウォレット管理組織
  ユーザーが所有するコントラクトウォレットを管理する組織であり，ウォレットのマルチシグの対象である．コントロールコントラクトの審査とコントラクトウォレットのアップデートを担当する．一般的な銀行と違い，資産はユーザーが保有したままでコントラクトウォレット管理組織が預かっているわけではないため，中央集権的になることはない．また，この組織は信頼されていて，不正は働かないものとし，コントラクトウォレットのソースコードは公開していることとする．
- IoT 機器
  提供したサービスを使用量のデータに変換したものをコントラクトウォレットに送信するために定期的にトランザクションを発行する．また，定期的にコントロールコントラクトの状態を参照して，サービスの on/off を決定する．今回の実装では，IoT 機器に秘密鍵がハードコードされており，Infura の Web3API を利用して，IoT 機器から直接トランザックションを発行できるものとする．
- コントラクトウォレット
  ユーザーの資産を安全に保管するため，ユーザーはコントラクトウォレット管理組織とのマルチシグで資産へアクセスできる．本稿では，2of2 のマルチシグを採用する．また，登録した IoT 機器からデータが送られてきた場合のみ，マルチシグを必要とせず，ホワイトリストに登録してある企業の EOA に資産の移動をできるものとする．
- コントロールコントラクト
  コントラクトウォレットから送られてくるデータをもとに料金を計算し，その値をコントラクトウォレットに返す関数を持つ．また，IoT 機器の状態パラメータの on/off を切り替える関数を持つ．ソースコードは企業により公開されているものとする．

4.3　システムの流れ

IoT 機器からの自動送金の流れを図 2 に示す．なお，トランザクションを Tx，インターナルトランザクションを InTx と表示する．

4.3.1　ホワイトリストへの登録

(1) ユーザーは予め与えられた料金の送信先となる企業の EOA を自身の秘密鍵での署名とともに，オフチェーンでコントラクトウォレット管理組織に送信する。

(2) ユーザーのマルチシグの相手であるコントラクトウォレット管理組織はその EOA が信頼できるものであるかを精査し，信頼できるならば，関数 **registWhitelists** を実行し，コントラクトウォレットのホワイトリストにマルチシグ のもとで追加する．

4.3.2 コントロールコントラクトの登録

(1) ユーザーは使用する IoT 機器の EOA とそれに関連するコントロールコントラクトのアドレス，支払い先の企業の EOA を自身の署名とともにオフチェーンでコントラクトウォレット管理組織に送信する．

(2) コントラクトウォレット管理組織は同様に，企業により公開されているコントロールコントラクトを精査し，信頼できるならば，マルチシグのもとでユーザーから送られた通りに，IoT 機器の EOA と企業の EOA に紐づけて登録する．

4.3.3 IoT 機器からのデータの送信と支払い

(1) IoT 機器が定期的にそれまで提供したサービスを使用量のデータに変換，コントラクトウォレットの関数 **IoTPayment** を実行して，データをコントラクトウォレットに送信する．

(2) 関数 **IoTPayment** がコントロールコントラクト内の関数 **calcuration** を実行し，料金の値を得る．

(3) デポジットがコントロールコントラクトから得た値より大きい場合は，その分だけ暗号資産を指定されたアカウントに送信する．

(4) デポジットがコントロールコントラクトから得た値より小さい場合は，関数 **operation** を実行し，IoT 機器の状態パラメーターを off にする．

4.3.4 IoT 機器の停止

(1) ユーザーが意図しないトランザクションが送られるなどの異変に気づいた場合，ユーザーは企業に連絡をとり，企業に IoT 機器を止めてもらうように連絡する．

(2) 連絡を受けた企業は，コントロールコントラクトの関数 **operation** を実行し，IoT 機器の状態パラメーターを off にする．

4.4 スマートコントラクトの展開

提案手法では，コントラクトウォレットがすでにブロックチェーン上にデプロイされているものとし，そのコントラクトウォレットに対して，IoT 機器の EOA やコントロールコントラクトを登録していく．なお，ユーザーの署名は自身の EOA のハッシュ (keccak256) を ECDSA で自身の秘密鍵で署名したものと，自身の EOA，IoT 機器の EOA，コントロールコントラクトのアドレスを順に結合したもののハッシュ (keccak256) を同様に署名したものとする．

コントロールコントラクト，コントラクトウォレットの関数は以下の通りであり，アルゴリズム 1,2,3 はコントラクトウォレット，アルゴリズム 4,5 はコントロールコントラクト内の関数とする．

4.4.1 ホワイトリストへの登録 registWhitelists

料金の支払先となる企業の EOA をホワイトリストに指定する関数であり，ユーザーの依頼のもとコントラクトウォレット管理組織が実行する．企業の EOA を指定することで資産の引き出し上限の制限なしに，支払いできるようになる．ユーザーからオフチェーンで送られる電子署名から署名者の公開鍵，EOA を生成，検証することで，自身の秘密鍵と合わせて 2of2 マルチシグ を実現している．

Algorithm 1 ホワイトリストへの登録 registWhitelists

Input: ユーザーの署名，ユーザーの EOA，企業の EOA
Output: void
　企業の EOA のハッシュ (keccak256) を計算
　ハッシュと署名から ECDSA により署名者の公開鍵，アドレスを生成
if 生成したアドレス == ユーザーの EOA **then**
　ホワイトリストに企業の EOA を加える
　イベントの発火
end if

4.4.2 IoT 機器の登録 registIoT

IoT 機器の EOA をホワイトリストに登録した企業の EOA とコントロールコントラクトのアドレスに紐付けて IoT 登録リストに登録する関数であり，ユーザーの依頼のもとウォレット管理組織が実行する．IoT 登録リストに登録することで，IoT 機器が関数 **IoTPayment** を実行することができるようになる．関数 **registWhitelists** と同様の方法でマルチシグで実行する．

Algorithm 2 IoT 機器の登録 registIoT

Input: ユーザーの署名，ユーザーの EOA，IoT の EOA，コントロールコントラクトのアドレス，企業の EOA
Output: void
　ユーザーの EOA，企業の EOA，コントロールコントラクトのアドレスを順に結合したもののハッシュ（keccak256）を計算
　ハッシュと署名から ECDSA により署名者の公開鍵，アドレスを作成
if 生成したアドレス == ユーザーの EOA **then**
　IoT の EOA と，コントロールコントラクト，企業の EOA を紐付けて IoT 登録リストに挿入
　イベント発火
end if

4.4.3 IoT による支払い IoTPayment

IoT 機器から与えられた使用量のデータを元に資産の支払いを実行する関数であり，IoT 機器が実行する．IoT 機器の EOA がコントラクトウォレットに登録されているかを検証し，されている場合は，予め紐づけてあるコントロールコントラクト内にある関数 **calcuration** を実行することで，料金の値を取得し，その値の暗号資産を紐づけてある企業の EOA に送金する．デポジットが少ない場合は，IoT 機器が提供するサービスの on/off を切り替える関数 **operation** を実行する．

4.4.4 料金の計算 calcuration

関数 **IoTPayment** により内部的に実行される関数である．与えられた使用量のデータから料金を計算し，その値をウォレットコントラクトに返す．

Algorithm 3 IoT による支払い IoTPayment

Input: 使用量のデータ
Output: void
if 実行者の EOA in IoT 登録リスト **then**
実行者の EOA に紐付けてあるコントロールコントラクト内の **calucuration** を実行
if **calucuration** で得た値が deposit の値より大きいか判定 **then**
calucuration の値の資産を紐づけてあるアドレスに送金
イベント発火
else
コントロールコントラクト内の **operation** を実行
イベント発火
end if
end if

Algorithm 4 料金の計算 clacuration

Input: 使用量のデータ
Output: 料金の値
使用量のデータから料金の値を四則演算

4.4.5 IoT の状態管理 operation

IoT 機器の状態を管理する関数で IoT 機器のサービスの on/off を司る変数を変化させる．この関数を実行する際，実行者のアドレスが企業の EOA またはコントラクトウォレットであるか検討するため，企業の秘密鍵を入手しない限り，外部からは企業以外が実行することは不可能である．

Algorithm 5 IoT 機器の状態管理 operation

Input: void
Output: void
if 実行者のアドレス == 企業の EOA or IoT の EOA **then**
IoT の状態の on/off を変換する
end if

5. 実装と評価

Ethereum のテストネットワークの一つである Rinkeby において，google chrome の拡張機能で EOA を管理する Metamask と Web3API を提供する Infura，IoT 機器は arduino 言語，スマートコントラクトは solidity で記述し，提案手法のプロトタイプの実装を行うことで．提案手法が実現可能であることを示す．

5.1 構成要素

- ESP32-DevkitC(1 台)
  IoT 機器の役割を果たす．今回の実装では，実際に IoT 機器が提供したサービスを測定し，使用量のデータを送信するのではなく，ランダムな整数を使用量のデータとして使用することとする．IoT 機器の開発環境は，Arduino-IDE，CPU は，Xtensa デュアルコア 32 ビット LX6 マイクロプロセッサ，メモリは，520KiB（SRAM），容量は，4MiB（フラッシュメモリ）である．

表 1 実行環境

名前	バージョン
Arduino-IDE	1.8.13
solidity	0.5.17

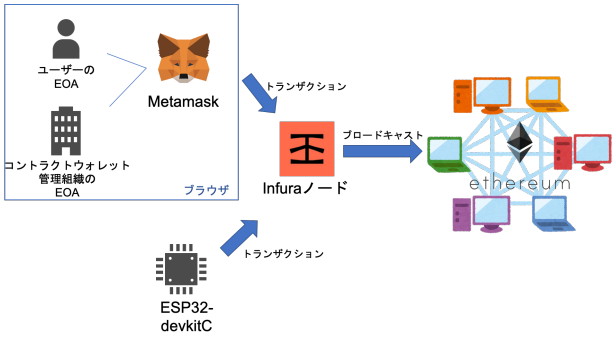


図 3 プロトタイプ実装の構成図

表 2 gas コスト

トランザクションの種類	gas コスト
registWhitelists	29,247wei
registIoT	84,047wei
IoTPayment(通常)	36,741wei
IoTPayment(残高不足)	36,615wei

表 3 アドレスの振り分け

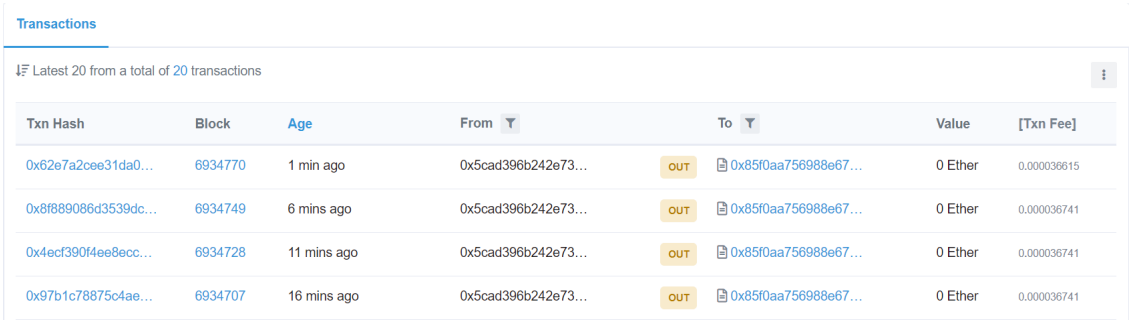
役割	アドレス
企業の EOA	0xa4820181BE90b5570Ce03bbF605b65796d6B6bBA
IoT の EOA	0x5CAd396b242E73BC9f4827F33aa45fdc73061801
コントラクトウォレットのアドレス	0x85f0aA756988E67fE64baaB613C3aA896AA5Ac84
コントロールコントラクトのアドレス	0x41c0Da607bfB6DF20dB79D49d27F6D94c3cdedE9

- Infura
  Ethereum のノードホスティングサービス．自身で Ehtereum のフルノードを立てる必要なしに，このサービスを通して Ethereum ブロックチェーンに参加することができる．Web3 の JSON RPC API が用意されているため，IoT 機器から http または websocket を通じて，トランザクションを発行することができる．
- Metamask
  google chrome の拡張機能であり，ブラウザ上で EOA の管理を容易にしてくれるサービス．使用するノードは指定することができ，今回の実装では，Infura ノードを使用する．

実行環境は表 1，全体の構成図は図 3 に示す．

5.2 プロトタイプ実装

(1) ESP32 から使用したデータを 1000 から 2000 の間のランダムな値として，Infura を通して 5 分ごとにコントラクトウォレットの関数 **IoTPayment** を実行する．図 4 から、トランザクションがマイニングされる時間に誤差はあるが、おおよそ 5 分で実行されていることがわかる．なお，トランザクションの確認には，Etherscan [9] と言われる，Ethereum ブロックチェー

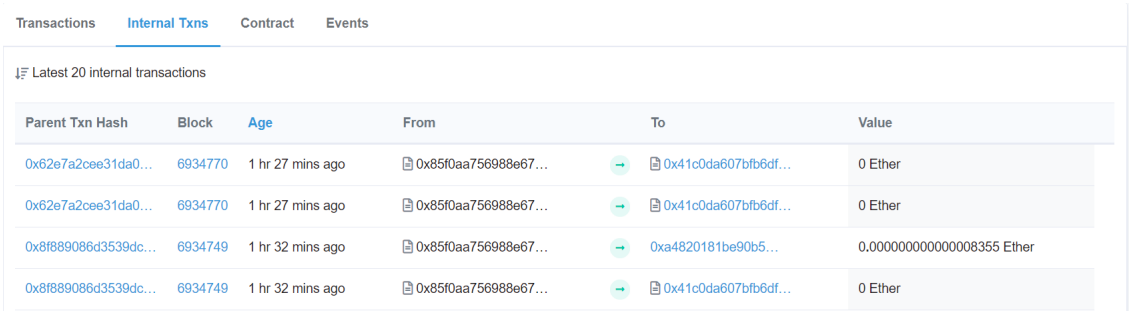


Transactions

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

図 4　IoT からのトランザクション



Transactions　Internal Txns　Contract　Events

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

図 5　コントラクトウォレットからのインターナルトランザクション

ン内のトランザクションを確認できるサイトを用いた．図 4，図 5，図 6 はこのサイトで確認したアドレスごとのトランザクションの記録である．

(2) コントラクトウォレットがインターナルトランザクションを発行し，コントロールコントラクトの関数 **calcuration** を実行し，その値を受け取る．図 5 からインターナルトランザクションが発行されていることがわかる．

(3) この一連の動作が完了すると，企業の EOA に資産が送金されていることを確認できる（図 6 参照）．以上から提案手法である自動送金は正常に動作することを確認できた．

実行した関数毎に発生したガスコストは表 2，IoT，企業，コントラクトウォレットのアドレスは表 3，コード量はコントラクトウォレットは 71 行，コントロールコントラクトは 32 行である．また，2019 年 8 月 10 日現在，gas コストは 10000wei あたりおよそ 0.0000000004 円である．

6. 考察

6.1 セキュリティについて

このシステムに対して考えられる攻撃方法として，ユーザー側，企業側，第三者による攻撃の三つに分類して考えることができる．

6.1.1 ユーザー側からの攻撃

ユーザー側からの攻撃方法としては，サービスを受けながら，そのサービス料を支払わない事が考えられる．前提として決まった時間に，その期間において使用した分の資産を支払うとする．この前提の上でウォレットの資産を予め少なくしておくことで，次の決済が行われるまでにサービスを受けることができ，資産で支払えるより多くのサービスを受け取ることができる．

この攻撃に対しては，IoT 機器がデータを送り，決済する時間間隔を短くすることで対処することができる．データの送信間隔が短いほど，ユーザーが資産で賄えないサービスを受ける量も短くなるので，企業側の損失も少なくなる．ただし，トランザクションを発行する量も増え，その分かかる gas コストも増大するため，送信間隔は検討が必要である．

6.1.2 企業側からの攻撃

サービス提供者からの攻撃方法としては、サービスを提供していないのにもかかわらず、料金を受け取ることがあげられる。

しかし，この攻撃に対しては，ユーザーが使用した量に対して料金を計算，支払いがされるようにスマートコントラクトに記述されているため，事実上不可能であると言える．

6.1.3 第三者からの攻撃

第三者からの攻撃としては，何らかの形で IoT 機器，またはユーザーの秘密鍵を奪われた場合にユーザーが意図しないトランザクションを発行されてしまうことがあげられる．

IoT 機器の秘密鍵が奪取された場合，攻撃者は IoT 機器になりすまし，適当な値をユーザーが使用した量として，トランザクションを発行することができる．この攻撃に

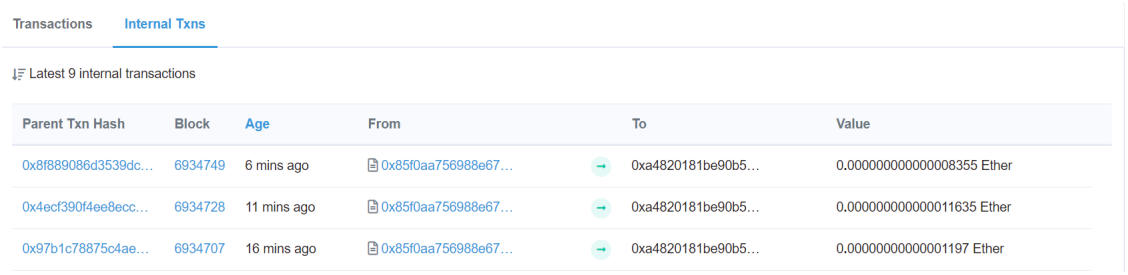


図 6　企業のアドレスが資産を受け取る

対しては，トランザクション発行時にイベントが発火し，ユーザーがブロックチェーンを監視することで，基本的にユーザーは怪しいトランザクションに対しては気づくことができる．そして，企業に連絡をとることでサービスを停止することができるため，被害を最小限に抑えることが可能である．

ユーザーの秘密鍵が奪取された場合，攻撃者はユーザーのコントラクトウォレットへのアクセスを試みることが可能になる．しかし，ユーザーの秘密鍵だけではある一定の金額しか引き落とすことができず，それ以上の金額を引き落とそうとするとマルチシグによる検証が必要になるため，被害を最小限に抑えることができる．また，その際に，イベントが発火し，ユーザーがブロックチェーンを監視することで，意図しないトランザクションに気づいたユーザーはウォレット管理組織にウォレットの所有者の EOA 変更の連絡し，ユーザーが新たに作成した EOA をそのコントラクトウォレットの所有者に指定して貰えば，攻撃者はコントラクトウォレットへのアクセス権を失い，ユーザーは資産へのアクセスを回復することができる．

6.2　プライバシー問題について

提案手法では，使用したデータに対して暗号化を施していない状態でパブリックチェーンに公開されてしまうという問題があげられる．しかし，一般的にユーザーの EOA とユーザーの個人情報は紐づくことはないので，暗号資産の送金履歴が公開されていることと同様に大きな問題ではないと考えられる．

この問題を解決するために，Paillier 暗号などの準同型暗号で暗号化した上でコントラクトウォレットにデータを送信し，コントロールコントラクト内で秘密計算を実行し，料金の値を計算する方法が考えられる．しかし，秘密計算を実行すると，gas コストが大幅に増加し，企業側の負担が増してしまう問題があり，検討が必要である．

また，パブリックブロックチェーンを使用するのではなく，コンソーシアム型，プライベート型のブロックチェーンを使用する方法も考えられる．上記二つのブロックチェーンを用いることで，情報の公開を一部制限することができるが，ブロックチェーンのノードが不正を働く可能性を排除できないため，こちらも検討が必要である．

7.　まとめ

本稿では，コントラクトウォレットと言われるスマートコンタクトを用いることで，秘密鍵が奪取されたことによる暗号資産盗難のリスクを抑えつつ，従量課金制の自動決済システムを実装，評価した．

今回の提案手法において，コントロールコントラクトを自由にプログラムすることで，より複雑な処理が可能になるが，その分 gas コストが増大することも考慮に入れなければならない．

今後の展望としては，暗号化などのプライバシー問題に取り組みつつ，gas コストが小さくなる手法を検討する予定である．

参考文献

[1] IDC japan，2020，「国内 IoT 市場 データエコシステム事業者調査結果を発表」

[2] Argent, “https://github.com/argentlabs/argent-contracts“, 2020

[3] Dapper, “https://github.com/dapperlabs/dapper-contracts “, 2020

[4] Gnosis, “https://github.com/gnosis/MultiSigWallet “, 2020

[5] Monika di Angelo, Gernot Salzer, “Wallet Contracts on Ethereum“, arXiv:2001.06909

[6] Hoang Tam Vo, Lenin Mehedy, Mukesh Mahania, Ermyas Abebe, “Blockchain-based Data Management and Analytics for Micro-insurance Applications “, CIKM ’17: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2539-2542

[7] Yanqi Zhao, Yannan Li, Qilin Mu, Bo Yang, Yong Yu, “Secure Pub-Sub: Blockchain-Based Fair Payment With Reputation for Reliable Cyber Physical Systems“, IEEE Access volume:6, 12295 - 12303

[8] Thitinan Tantidham, Yu Nandar Aung, “Emergency Service for Smart Home System Using Ethereum Blockchain: System and Architecture “, 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)

[9] Etherscan, Ethereum(ETH) Blockchain Explorer, “https://etherscan.io “, 2020