IoT 機器とスマートコントラクトを用いた警備用自動監視システム

岩田 琴乃[1,a)]　面 和成[1,2]

概要：近年，IoT 市場規模が急速に拡大し，様々な分野で IoT の活用が行われている．国内の IoT 市場では，セキュリティ分野の高い成長率が見込まれている．しかし，従来の中央集権型の管理では，サイバー攻撃による改竄・単一障害点などのリスク，IoT 機器のデータ量・整合性・信頼性の問題が存在するため，膨大な IoT 機器のログの管理を行うのは困難である．また，ログだけでは，どの程度の異常事態であるかを把握しづらい．本稿では，ブロックチェーンに IoT 機器のログを安全に記録するとともに，そのログを用いて自動的に異常を検知する手法を提案する． また，スマートコントラクトを用いて，ログから異常事態の程度を推定し，警備会社による初期対応までの一連の流れの自動化を可能にする．

キーワード：ブロックチェーン，スマートコントラクト，異常検知，IoT

Automatic Monitoring System for Security Using IoT Devices and Smart Contracts

Kotono Iwata[1,a)]　Kazumasa Omote[1,2]

Abstract: In recent years, the IoT market has been expanding rapidly. The IoT is being utilized in various fields. In the domestic IoT market, a high growth rate is expected in the security field. However, in the centralized management, there are risks such as falsification and single point of failure due to cyber attacks, as well as problems of data volume, integrity, and reliability of IoT devices. Therefore, it is difficult to manage the logs of a huge number of IoT devices and to grasp the degree of anomalies from the logs only. In this paper, we propose a method to securely record the logs of IoT devices in a blockchain and to automatically detect abnormalities by using the logs. Using smart contracts, we can estimate the degree of anomalies from the logs, and automate the sequence of events up to the initial response by the security company.

Keywords: Blockchain, Smart Contract, Anomaly Detection, IoT

1. はじめに

近年，IoT 機器が急速に普及している．IDC によると，全世界の IoT デバイス（エンドポイント）の普及台数は，2018 年の 228 億台から 2025 年には 416 億台に達すると予測されている [1]． それに伴い，国内の IoT 市場規模も増大している．NRI では，IoT の活用が進む主分野はエネルギー，セキュリティ (監視カメラや警備サービスなどを含む)，自動車, 流通, ヘルスケアの 5 分野であると見ており，上記の分野を組み合わせたサービスも合わせて, 2022 年には市場規模は 3 兆円を超えると予測されている [2]．そして，エネルギー，セキュリティ，自動車，その他の分野がそれぞれ 20 %を占めるようになる．その中でも，セキュリティ分野の成長率が高いとされている．また，セキュリティ分野の IoT 機器の発展に伴い，機械警備の対象施設も大きく増加している．機械警備とは，警備対象施設に警備

[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)] s2020525@s.tsukuba.ac.jp

人員を置かず，センサーを配置して異常を検知し，その作動情報を受信すると警備員が出動し，初期対応を行う形態の警備業務のことである [3]．警察庁の調べによると，機械警備の対象施設数は 2005 年から右肩上がりで，2019 年は 311 万施設が対象となっている [4] ．

また，IoT 機器は，工場や興業施設などの大規模施設に多く設置されている．IoT 機器の作動は，施設で発生している異常事態を早急に知らせ，施設の被害や損害を抑える効果がある．そして，IoT 機器のログは，異常事態時の背景・原因を解き明かす重要な証拠になる．

一般的に，機械警備では，IoT 機器のログは警備会社のコントロールセンターに送られる．そこで，その情報を基に機械警備員が指示を与え，対象施設に警備員を出動させる．また，IoT 機器のログの管理はサーバーで行われている．よって，改竄や破損等が起こりづらく，ログを安全に管理するシステムが求められる．

しかし，サーバーのような中央集権型の管理には，サイバー攻撃による改竄・単一障害点などのリスクが存在する [5]．ゆえに，ログに証拠としての信頼性がなくなり, 異常事態時の追跡の妨げになる可能性がある．また，サーバーは膨大な IoT 機器のデータを処理しきれなくなると，サービスが停止し，ダウンタイムが発生する．さらに，破損したデータは復元しづらい．よって，サーバーで膨大な IoT 機器のログの管理を安全に行うのは困難である．また，IoT 機器単体から送られてくるログからでは，対象施設でどの程度の異常事態が発生しているのか分かりづらい．

本稿では，ブロックチェーンを用いて，IoT 機器のログの管理と自動異常検知を行うシステムの提案を行う．具体的には，ブロックチェーンに IoT 機器のログを保存することで証拠としての完全性を維持させ，そのログを用いて自動的に異常を検知する．また，スマートコントラクトを用いて，ログから異常事態の程度を推定し，警備会社による初期対応までの一連の流れの自動化を提供する．さらに，IoT 機器の作動から警備会社が初期対応を開始するまでの動作と要する時間の検証を行い，実現可能性を示す．

2. 準備

2.1 ブロックチェーン

ブロックチェーンは，サトシ・ナカモトによって考案された分散型台帳技術のことである．取引情報が格納されているトランザクションはブロックごとにまとめられ，ブロックは鎖のように連結されている．このブロックにトランザクションを承認してまとめる作業をマイニングと呼ぶ．各ブロックは，1 つ前のブロックのハッシュ値を含んでいるため，ブロックチェーンの改竄を行うためにはそのブロック以降のハッシュ値を計算する必要がある．ゆえに，ブロックチェーンは耐改竄性の特徴を持つ．また，ブロックチェーンはトランザクションの公開範囲・管理者の有無によって，プライベート型・コンソーシアム型・パブリック型に分けられる．本稿では，複数の管理主体によって特定の範囲内でのみ運用されるコンソーシアム型ブロックチェーンを採用する．

2.2 IoT とブロックチェーン

現在では，仮想通貨以外にも IoT とブロックチェーンを組み合わせた研究がさかんに行われている．ブロックチェーンは分散型であるため，単一障害点が存在せず，システムのダウンタイムが無い．他にも，拡張性，データの不変性，相互運用性や整合性などのメリットが存在することから，ブロックチェーンと IoT の組み合わせは親和性があると言える．

2.3 スマートコントラクト

スマートコントラクトとは，ブロックチェーン上に展開されているプログラムで，自動的に契約を実行できる仕組みである．スマートコントラクトが実行されると，そのトランザクションは全てブロックチェーンに格納されるため，取引の流れの透明性とトランザクションの完全性を維持できる．ゆえに，第三者機関の仲介なく，任意に契約を実行できる．これにより，契約に要していた時間の短縮や人件費のコストの削減が可能である．

2.4 PoA

PoA（Proof of Authority）とは，PoS（Proof of Stake）に基づいた高速なコンセンサスアルゴリズムである．バリデーターと呼ばれる承認済みのアカウントによってのみ，トランザクションとブロックは検証される．このブロック生成プロセスは自動化され，信頼できるバリデーターがネットワークのセキュリティと一貫性を維持する．バリデーターは厳しい審査をクリアした者のみが選抜され，自身のアイデンティティはブロックチェーンに格納されているため，不正な行動を行うのが困難である．

2.5 Ethereum

Ethereum はブロックチェーンプラットフォームである．様々な言語でスマートコントラクトを実装できるが，本稿では Solidity を用いる．また，テストネットや Infura や MetaMask などのサービスが用意されている．Infura とは，自身で Ethereum のフルノードをたてることなく，簡単に既存の Ethereum ブロックチェーンやサービスを使用できるノードホスティングサービスである．Web3 の JSON RPC API が提供されているため, パソコンなどからトランザクションの発行が可能である．MetaMask とは，google chrome の拡張機能で，Ethereum のアカウントである EOA (Externally Owned Account) の管理を提供するサービスである．

3. 関連研究

3.1 IoT 機器のログをブロックチェーンに保存・管理するシステム提案

Thitinan らの研究 [6] では，ブロックチェーンを用いた，高齢者や患者向けのヘルスケアサービスとホームセーフティの監視及び自動制御を行うホームサービス用のシステムを提案した．ヘルスケアもしくは侵入検知用の IoT 機器が作動すると，センサー情報を含んだ緊急事態を意味するトランザクションをブロックチェーンに自動で送信し，警備会社が駆けつける．

この関連研究の特徴は 2 点ある．1 点目は，IoT 機器の緊急信号の受信から警備隊出動までの記録をブロックチェーンに格納し，記録の不変性・整合性を維持する点である．2 点目は，スマートコントラクトを使用することで，緊急信号の受信後の対応を自動化した点である．緊急信号の受信後，家主に通知し，警備会社のスタッフに情報を伝え，警備隊を出動するまでの一連の流れをブロックチェーンで一括管理する．これは，人員削減，作業の効率化の効果がある．しかし，各 IoT 機器より送信されるトランザクションからでは，発生している異常事態の程度を推定できない．

3.2 ブロックチェーンを用いた IoT フォレンジックフレームワーク

Suat らの研究 [8] では，3 種類のブロックチェーンを用いて，IoT 機器から収集したデータを保存・管理するフォレンジックフレームワークを提案した．事前に定義された前提条件を基にデータのフィルタリングを行い，フィルタリングにかかったデータのみを IoT 機器を管理している企業のデータセンターに保存し，また，そのデータのハッシュを Stellar と EOS の両方に書き込む．一日の終わりに, 企業のデータセンターは，Stellar と EOS のトランザクションを参照し，Merkle root を計算する．その値を，より安全で信頼性の高い Ethereum に格納する．データセンターにはデータ毎に，IoT のイベントデータ，IoT のイベントデータのハッシュ値，Merkle Path node, Merkle root が保存される．何か異常事態が発生した場合，原因を解き明かすため，捜査官や警察官は企業のデータベースにアクセスする．IoT のイベントデータのハッシュ値を計算し，その値が Stellar もしくは EOS にあれば，そのデータをフォレンジックデータとして信用することができる．なお，この研究では, IoT デバイスがハードウェアセキュリティによって改竄されていないことを前提としている．

3.3 ブロックチェーンのブラックリスト

Yannan らの研究 [9] では, CA 機能を持つブロックチェーンベースの PKI の提案を行った．特徴としては，CA 機能を持つブロックチェーン上の無効にしたいノードをブラックリストに追加することでノードの失効管理を行う提案を行う点である． ブロックチェーンは一度格納した情報を削除できないというデメリットがあるため，ノード情報を削除することができない．そこで，検証者がブラックリストに追加することで，ユーザーの失効管理を行う．

3.4 スマートコントラクトの異常検出

Xinming らの研究 [7] では，Ethereum のスマートコントラクトを防御する最初の侵入検知システム (IDS) である ContractGuard を提案した．スマートコントラクトに対する侵入攻撃を検知するプログラムを埋め込み，従来のプログラムの IDS と同様に，ContractGuard はアノマリー検知ベースで侵入の試みを異常な制御フローとして検出する．また，異常な動作を検出すると，スマートコントラクトに及ぼした不正な行動を管理者にアラームとして通知する．しかし，複数の IoT 機器のトランザクションから犯人の目的や異常事態の原因を予測することには対応していない．

4. 提案手法

4.1 概要

IoT 機器のログの管理と自動異常検知を行うシステムを提案する．本稿では，警備会社が複数の IoT 機器を用いて，大規模施設を警備・監視しているとする．提案手法は 5 段階で構成される．まず，IoT 機器をブロックチェーンに登録する．対象施設に設置された IoT 機器が異常を検知して作動すると，センサー値をブロックチェーンに送信する．ブロックチェーンはログとして保存し，センサー値を基に異常事態の程度を推定する異常レベルを計算する．警備会社は異常レベルを読み取り，異常レベルに応じた初期対応を自動的に開始する．実際に何らかの被害や損失が発生した場合は，ログを証拠として調査機関へ提出する．また，IoT 機器が破損するか，あるいは乗っ取られた場合は，IoT 機器ベンダーが IoT 機器を失効する．

提案手法の全体図を図 1 に示す．

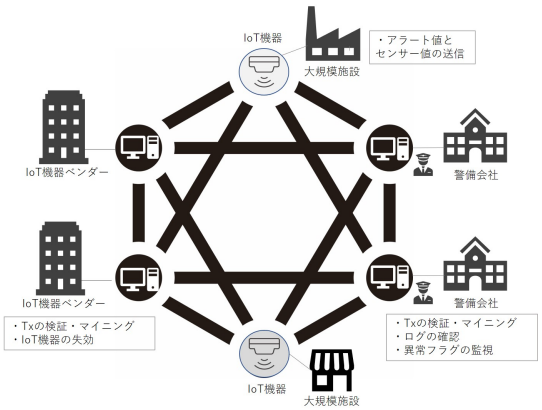

図 1 提案手法の全体図

4.2 エンティティ

- 大規模施設
  警備会社が警備する施設．人の命や資産を守りたい場所に，監視・異常検知を行う IoT 機器を設置する．
- 警備会社
  IoT 機器を監視・管理することで大規模施設の警備を行う． IoT 機器のログの保存と自動異常検知を行うブロックチェーンを管理する．
- IoT 機器ベンダー
  IoT 機器の秘密鍵の登録，IoT 機器の販売及び失効管理を行う．
- IoT 機器
  ブロックチェーンノードであり，ブロックチェーンにセンサー値を送信する．IoT 機器の例として，温度センサー，赤外線センサーなどの機器が挙げられる．

4.3 前提条件

- 警備会社，IoT 機器ベンダー，IoT 機器で構成されるコンソーシアム型ブロックチェーンである．
- コンセンサスアルゴリズムは PoA を想定しており，トランザクションの検証とマイニングを行うバリデーターは複数の IoT 機器ベンダー・警備会社である．
- 警備会社と IoT 機器ベンダーのみ，ブロックチェーンを参照することができる．

4.4 システムの流れ

提案システムの流れを図 2 に示す．まず，IoT 機器の情報をブロックチェーンに登録し，対象施設の警備・監視したい場所に設置する．IoT 機器はセンサー値をブロックチェーンに送信する．ブロックチェーンはスマートコントラクトを用いてログを保存し，同時に，一定時間内に作動したログを基に，自動異常検知関数で異常事態の程度を推定する異常レベルを計算する．異常フラグのレベルが高いほど，被害や損害が大きくなる可能性が高くなる．警備会社はブロックチェーンを常時監視しており，異常レベルを読み取ると，異常レベルに応じた初期対応を行う．実際に事故や事件が発生した場合，原因や背景を解き明かす証拠として，関連のあるログを警察などの調査機関に提出する．また，IoT 機器が破損するか，あるいは乗っ取られた場合，IoT 機器ベンダーが IoT 機器のアドレスをブロックチェーンのブラックリストに追加することで IoT 機器の失効を行う．

4.4.1 IoT 機器の登録

- IoT 機器ベンダーは販売前に，IoT 機器に秘密鍵の登録を行う．その後，購入された IoT 機器は警備対象施設に設置される．
- 警備会社は，関数 RegisterIoT を実行し，IoT 機器の識別子，アカウントアドレス，住所，建物内の詳細な設置場所の情報をブロックチェーンに格納する．同時に，アカウントアドレスは IoT 機器の Whitelist にも登録される．よって，関数 RegisterIoT に登録された IoT 機器のみがブロックチェーンにトランザクションを送信できる．

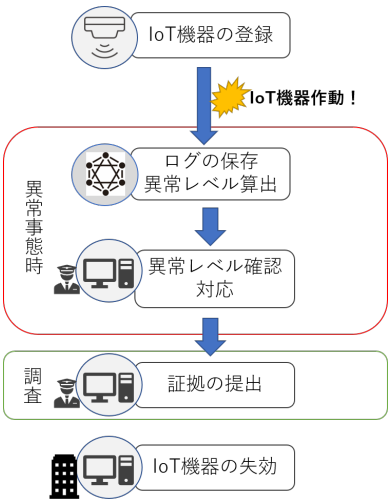

図 2 システムの流れ

4.4.2 ログの保存・自動異常検知

- IoT 機器は作動すると，自動的にセンサー値をブロックチェーンに送信し，関数 AddAlert が実行される．
- 関数 AddAlert はセンサー値，異常レベル，アカウントアドレス，タイムスタンプ，ログの識別子であるアラート ID を格納する．アラート ID は，ブロック高・タイムスタンプ・アドレスを基にしたハッシュ値（Keccak256）である．
- 関数 AddAlert ではまず，センサー値，アドレス，タイムスタンプ，アラート ID を格納し，その後，センサー値を基に異常レベルの算出と格納を行う．異常レベルは，予め設定された異常検出ルールに沿って算出される．具体的には，ある一定時間内にどの IoT 機器が閾値を超えた異常を検知したのかによって，異常レベルが算出される．警備対象施設の警備目的に合わせて，一定時間，IoT 機器の組み合わせ，それに対応した異常レベルを任意に決めることができる．

4.4.3 初期対応

- 警備会社はブロックチェーンを常時監視し，その異常レベルを読み取る．
- 読み取った異常レベルに応じた初期対応を行う．
- 警備員が出動した場合は，実際の現場の状況を確認する．関数 AddContent を実行し，その情報をアラート ID 毎にブロックチェーンに格納する．何らかの事故や事件が発生した場合，格納した情報は，原因や背景を解き明かす調査に利用できる．

4.4.4 調査機関への証拠提出

- 事故や事件などの異常事態が発生した場合，その原因や背景を解き明かす可能性のあるログを検索する．アラート ID，タイムスタンプ，センサー値，異常レベル毎の検索が可能である．本稿では，アラート ID からログの検索を行う関数 SearchAlert を示す．ログは，センサー値，異常レベル，アカウントアドレス，タイムスタンプ，アラート ID で構成されている．
- ログを警察などの調査機関に提出する．

4.4.5 IoT 機器の失効

- IoT 機器が破損するか，あるいは乗っ取られた場合，施設の管理人が IoT 機器ベンダーに報告する．
- IoT 機器ベンダーは関数 BlackList を実行し，アカウントアドレスを Blacklist に登録する．これにより，IoT 機器の失効を行う．

4.5 具体例

提案システムの具体例として食品工場を挙げ，以下に示す．食品工場の品質管理と施設の警備を目的とし，図 3 のように工場内に IoT 機器 3 台が設置されているとする．ここでは，IoT 機器 A を湿度センサー，IoT 機器 B を温度センサー，IoT 機器センサーを熱を感知する赤外線センサーとする．IoT 機器 A・B・C の順で重要度が高いとし，それぞれのセンサーの閾値を T_A，T_B，T_C とする．また，工場内は一定の湿度と温度に保たれているものとし，IoT 機器 A・B がそれぞれ T_A・T_B を超えたセンサー値を検知すると，品質が大きく下がるものとする．

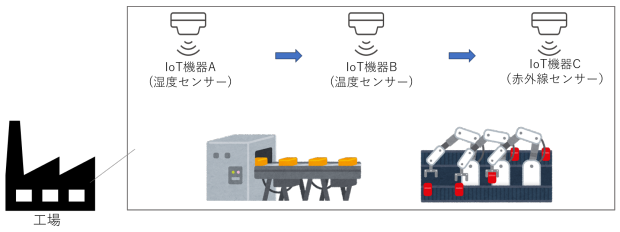

図 3 工場に設置されている IoT 機器

4.5.1 IoT 機器の登録

関数 RegisterIoT を実行し，設置されている 3 台の IoT 機器の識別子，アカウントアドレス，住所，建物内の詳細な設置場所の情報をブロックチェーンに登録する．

4.5.2 ログの保存・自動異常検知

IoT 機器が作動すると，センサー値を自動的にブロックチェーンに送信する．ブロックチェーンがセンサー値を受信すると関数 AddAlert が実行され，異常検知ルールに基づいた異常レベルの算出とログの保存が行われる．異常レベル算出のルール例を表 1 に示す．ここでは，一定時間を 30 分間とする．よって，30 分以内にどの IoT 機器が閾値を超えたセンサー値を検知するかによって，異常レベルが算出される．例えば，30 分間に IoT 機器 A もしくは IoT 機器 B がセンサー値を検知した場合，何者かによる品質悪化を目的とした犯罪行為や火災などの被害が発生している可能性が低いと考え，異常レベル 1 が算出される．また，30 分以内に IoT 機器 A・B が閾値を超えた場合，異常レベル 2 が算出される．これは，何者かが工場の品質を下げるために，湿度と温度を変化させた可能性があるからである．また，IoT 機器 C が閾値を超えると，既に火災などの異常事態が発生している可能性が極めて高いので，一定時間によらず即刻異常レベル 3 が算出される．

表 1 異常検知ルール例

異常レベル	異常検知ルール
異常レベル 3	IoT 機器 C $> T_C$
異常フラグ 2	30 分内 ・IoT 機器 A $> T_A$ 且つ IoT 機器 B $> T_B$
異常フラグ 1	30 分内 ・IoT 機器 A $> T_A$ 且つ IoT 機器 B $< T_B$ ・IoT 機器 A $< T_A$ 且つ IoT 機器 B $> T_B$

4.5.3 初期対応

警備会社は異常レベルを常に監視している．異常レベルを確認すると，レベルに応じた対応を行う．警備会社の初期対応の例を表 2 に示す．異常レベル 1 の場合は警戒を行い，異常レベル 2 の場合は厳重警戒を行い，異常レベル 3 の場合は警備員が出動する．夜間の警備の際は，異常レベル 2 の場合でも，警備員が出動する．警備員が出動した場合，関数 AlertContent を実行し，アラート ID，担当した警備員の名前、実際の現場の状況の情報を格納する．

表 2 異常レベルと対応の例

異常レベル	警備会社の対応
異常レベル 3	警備隊出動
異常レベル 2	厳重警戒（場合によっては出動）
異常レベル 1	警戒

4.5.4 調査機関の証拠提出

事件や事故によって警備対象施設に被害や損害が発生し，調査が行われるとする．関数 SearchALert を実行し，事件に関連のあるログを検索し，警察などの調査機関に証拠として提出する．また，関数 AlertContent で格納した異常発生当時の現場の状況の情報も提出する．

4.5.5 IoT 機器の失効

IoT 機器が破損するか，あるいは乗っ取られた場合，工場の管理人は IoT 機器ベンダーに報告する．IoT 機器ベンダーは BlackList 関数を実行し，IoT 機器のアカウントアドレスを Blacklist に登録することで，IoT 機器の失効を行う．

4.6 関数

本節では，提案手法の主な関数について説明する．スマートコントラクトはブロックチェーン上にデプロイされているものとする．なお，本稿で記載されているアドレスは全て Ethereum アカウントアドレスを指している．

4.6.1 IoT 機器の登録　RegisterIoT

IoT 機器をブロックチェーンのノードとして使用するために，ブロックチェーンに IoT 機器の登録を行う関数である．まず，IoT 機器情報を保存する IoTList に，登録したい IoT 機器のアドレスが既に登録されていないか確認する．登録されていない場合，IoTList に IoT 機器の識別子，アドレス，住所，建物内の詳細な設置場所の情報を格納する．また，アドレスを紐づけしている Whitelist にアドレスを格納し，1 を挿入する．1 が挿入されている場合は登録されていることを表し，NULL を示すデフォルト値 0 の場合は登録されていないことを表す．Whitelist に登録された IoT 機器のみ，センサー値をブロックチェーンに格納できる．

4.6.2 ログの保存・自動異常検知　AddAlert

作動した IoT 機器のセンサー値，アドレス，タイムスタンプ，ログの識別子であるアラート ID をブロックチェーンに格納し，センサー値を用いて異常レベルの算出・格納を行う関数である．まず，ブロック高，タイムスタンプ，アドレスを基にしたハッシュ値（Keccak256）をアラート ID に挿入する．その後，IoT 機器のアドレスが Whitelist に登録されており，且つ，Blacklist に登録されていないかを確認する．前述の条件をクリアしている場合のみ，ログを保存する AlertList に，センサー値，アドレス，タイムスタンプ，AlertID を格納する．その後，IoT 機器のセンサー値が閾値を超えているか確認する．一回目のセンサー値の送信の場合，IoT 機器 A もしくは B ならば異常レベル 1，IoT 機器 C ならば異常レベル 3 を算出する．一回目ではないならば，過去のログを検索し，その過去のログのタイムスタンプが今回作動した IoT 機器のタイムスタンプより一定時間前までに発行されたものか否かによって，異常検知ルールを基に異常レベルの算出される．また，算出された異常レベルは，AlertList に格納される．ここでは，IoT 機器の閾値を T とする．

4.6.3 現場の情報の追加　AddContent

警備員が出動した場合に，現場の状況の情報を保存する関数である．現場の状況の情報を保存する AlertContentLis に，アラート ID，担当した警備員，実際の状況を格納する．

4.6.4 ログの検索　SearchAlert

AlertID 毎に，ログを検索できる関数である．入力したアラート ID と同じアラート ID が含まれるログが存在するか検索する．存在する場合は，アドレス，異常レベル，センサー値，タイムスタンプを出力する．

4.6.5 IoT 機器の失効　Blacklist

IoT 機器が破損するか，あるいは乗っ取られた場合に，IoT 機器を失効させる関数である．IoT 機器の秘密鍵に対応している公開鍵を基に作成されているアドレスを紐づけしている Blacklist にアドレスを格納し，1 を代入する．1 が挿入されている場合は登録されていることを表し，NULL を示すデフォルト値 0 の場合は登録されていないことを表す．

Algorithm 1 IoT の登録 RegisterIoT

```
Input: 機器の識別番号，アドレス，住所，設置場所
Output: void
  if IoTList. アドレス != アドレス then
    IoTList.push(機器の識別番号，アドレス，住所，設置場所)
    Whitelist.[アドレス]=1
  end if
```

Algorithm 2 ログの保存・自動異常検知 AddAlert

```
Input: センサー値，アドレス，タイムスタンプ，
Output: アラート ID，異常レベル
    ブロック高・タイムスタンプ・アドレスを基にしたハッシュ値（Keccak256）の計算，AlertID に挿入
  if WhiteList.[アドレス]==1 && BlackList[アドレス]==0 then
    AlertList.push(センサー値，アドレス，タイムスタンプ，アラート ID)
    if IoT 機器 > T then
      if 1 回目のセンサー値の送信 then
          IoT 機器 A もしくは B ならば異常レベル 1，IoT 機器 C ならば異常レベル 3 を算出
      else
          過去のログを検索
        if 一定時間前までに，閾値を超えた IoT 機器がある then
          異常検知ルールを基に異常レベルの算出
          AlertList.push(異常レベル)
        else
          異常検知ルールを基に異常レベルの算出
          AlertList.push(異常レベル)
        end if
      end if
    end if
  end if
```

Algorithm 3 現場の情報の追加　AlertContent

```
Input: アラート ID，担当した警備員，実際の状況
Output: void
    AlertContentList.push(アラート ID，担当した警備員，実際の状況)
```

Algorithm 4 ログの検索　SearchAlert

Input: アラート ID
Output: アドレス，異常レベル，センサー値，タイムスタンプ
　ログのアラート ID を検索
　if ログのアラート ID == アラート ID **then**
　　アドレス，異常レベル，センサー値，タイムスタンプの出力
　end if

Algorithm 5 IoT 機器の失効　Blacklist

Input: アドレス
Output: void
　　Blacklist.[アドレス]=1

5. 実証実験

5.1 実験目的

IoT 機器がセンサー値を送信し，警備会社が異常レベルを確認するまでの一連の動作実験と要する時間の検証を行い，実現可能性を検証する．

5.2 環境

Infura と MetaMask 用いて，Ethereum のテストネットワークである Rinkeby 上で実証実験を行った．スマートコントラクトは，開発プラットフォームである RIMIX–Ethereum IDE において Solidity（Ver.0.5.16）で記述し，実装を行った．また，Ubuntu（18.04 LTS）上で，IoT 機器を模したセンサー値を送信するプログラムと，警備会社が異常レベルを確認するデーモンプログラムを Python で実装した．本実験では，仮想的に 3 台の IoT 機器と警備会社の異常レベルを確認するデーモンプログラムの役割を果たす．全体の構成図を図 4 に示す．

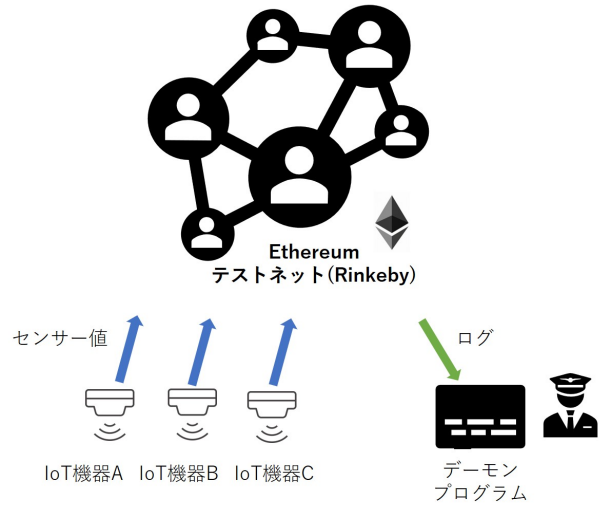

図 4　実験の構成図

5.3 実験の概要

実際の運用を模擬するため，センサー値を送信させるプログラムを実行する度に，IoT 機器を適当に選択し，センサー値をブロックチェーンに送信させる．IoT 機器によってセンター値の閾値は様々であるため，センサー値は 0〜100 までの値をランダムに送信させる．デーモンプログラムでは，時刻，異常レベル，IoT 機器の識別番号，住所，設置場所を表示させ，提案手法の動作検証を行う．また，センサー値を送信した時刻とデーモンプログラムで異常レベルを確認できた時刻から，IoT 機器の作動から警備会社が初期対応を開始するまでにかかる時間を計測し，実現可能性を検証する．

5.4 実験の流れ

前提として，関数 RegisterIoT を実行し，予め IoT 機器 A・B・C の登録を行う．

(1) IoT 機器 A・B・C を適当に選択し、ランダムなセンサー値を 10 回送信する．
(2) 警備会社のデーモンプログラムで異常レベルを確認し、動作検証を行う．
(3) センサー値を送信した送信時刻とデーモンプログラムで異常レベルを確認した時刻から，IoT 機器作動後に警備会社が初期対応を開始するまでの時間を計測する．

5.5 実験の検証

5.5.1 センサー値の送信

センサー値「30」をブロックチェーンに送信させた．図 5 に，センサー値を送信した日時とセンサー値を示す．また，表示されている日時はパソコンのローカルの日時である．

```
iwata@LAPTOP-P9B5AQGM:~$ python SecFirm_check.py
2021年08月21日 14:41:32:958911
AnomalyFlag: 3
IoT機器識別番号：03　住所：A city 1-2　設置場所：B center F1
```

図 5　実験の構成図

5.5.2 デーモンプログラム上での動作検証

警備会社は常時，デーモンプログラムを用いて異常レベルの確認を行っている．デーモンプログラムでは，図 6 のように異常レベルとその参照日時，関数 RegisterIoT で登録した IoT 機器識別番号，住所，設置場所が表示される．

```
iwata@LAPTOP-P9B5AQGM:~$ python SecFirm_check.py
2021年08月21日 14:41:32:958911
AnomalyFlag: 3
IoT機器識別番号：03　住所：A city 1-2　設置場所：B center F1
```

図 6　実験の構成図

5.5.3 時間計測

センサー値の送信時刻とデーモンプログラムで異常レベルを確認できた時刻から，IoT 作動から異常レベルを確認できるまでに要する時間を計測した．その結果，10 回における実験の平均時間は約 7.812 秒であった．センサー値の送信は正常に行われ，デーモンプログラムにおいて異常レベルや異常を検知した IoT 機器の情報の更新を確認することができた．

6. 考察

6.1 IoT 作動から異常レベル確認までの動作と時間

IoT 作動から異常レベルを確認できるまでの平均時間は約 7.812 秒であった．PoA のブロック生成時間は約 5 秒程度であるので，計測時間は妥当であると考えられる．また，一般的な機械警備では，コントロールセンターで機械警備員が作動情報を読み取り，警備員に指示を与えることで，初期対応が行われる．警備業法によると，機械警備業務を実施する上で，センサー作動後 25 分以内に対象施設に警備員を到着させ，速やかに事実確認もしくは警察などへの通報が努力義務として規定されている [3]．ゆえに，早急な判断と初期対応が求められる．提案手法では，センサー発生後に機械警備員の仲介を挟むことなく，自動的に指示が出され，初期対応が行われる．また，IoT 機器作動から初期対応を開始するまでの時間は約 7.812 秒と短い．よって，初期対応が開始される時間を短縮することが可能である．また，コントロールセンターでは 24 時間体制で，規模に応じて複数人で監視する．本提案手法ではブロックチェーンが一連の流れの管理を行うため，従来の体制より必要とする機械警備員の人数を少なくすることができ，人件費の削減を行うことが可能である．ゆえに，本提案手法は実現可能性が高い．

6.2 ログの証拠としての信頼性

ブロックチェーンに格納した情報の改竄や削除を行うことは非常に困難である．さらに，ブロックチェーンは分散型であるため，一部のノードの情報が破損したとしても復元が可能であり，また，サービスが停止することがない．よって，ブロックチェーンに格納された IoT 機器のログの完全性は維持される．ゆえに，警備対象施設で被害や損失が発生した場合，ブロックチェーンに格納されたログは背景や原因を解き明かすための信頼性の高い証拠となる．

7. まとめ

本研究では，従来型の機械警備の管理と中央集権型の IoT 機器の管理の問題に対処するため，ブロックチェーンを用いた IoT 機器のログの管理と自動異常検知を行うシステムの提案を行った．ブロックチェーンに IoT 機器のログを格納することでデータの完全性を維持し，異常事態が発生した場合の原因や背景を解き明かす証拠として利用できる．自動異常検知を実装することで一連の対応の自動化を実現させた．また，Ethereum のテストネット環境で IoT 機器の作動 から警備会社の初期対応を開始するまでの動作と要する時間の検証実験を行い，実現可能性を示した．

提案手法では，IoT 機器が乗っ取られた場合，不正データが書き込まれる可能性がある．不正データは事故や事件の調査を混乱させる危険がある．この問題を解決するために，ログのブラックリストを作成し、不正なログの AlertID を登録をすることで、証拠として使用できないようにする方法が挙げられる．しかし，不正データによる異常レベルの誤計算を解決することはできないため，今後検討が必要である．

謝辞 本研究成果の一部は，JSPS 科研費 19H04107 の助成を受けたものである．

参考文献

[1] IDC japan，2020,「国内 IoT インフラストラクチャ市場予測を発 表」
[2] 野村研究所，拡大する IoT 市場にどう関わるべきか，”https://www.nri.com/-/media/Corporate/jp/Files/PDF/knowledge/publication/itsolution/2017/06/ITSF170603.pdf”，2021.8.16
[3] 警備業法（昭和 47 年法律第 117 号）
[4] 警視庁：平成３０年における警備業の概況（2018）
[5] N. M. Kumar, P. Mallickb, ”Blockchain technology for security issues and challenges in IoT”, Procedia Computer Science Volume 132, 2018, pp 1815-1823
[6] T. Tantidham, Y. N. 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), 2019, pp 888-893
[7] X. Wang, J. He, Z. Xie, G. Zhao and S. Cheung, ”ContractGuard: Defend Ethereum Smart Contracts with Embedded Intrusion Detection”, in IEEE Transactions on Services Computing, vol. 13, no. 2, pp 314-328
[8] S. Mercan, M. Cebe, E. Tekiner, K. Akkaya, M. Chang and S. Uluagac, ”A Cost-efficient IoT Forensics Framework with Blockchain”, 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)，pp 1-5
[9] Y. Li, Y. Yu, C. Lou, N. Guizani and L. Wang, ”Decentralized Public Key Infrastructures atop Blockchain”, in IEEE Network, vol. 34, no. 6, pp 133-139