明治大学総合数理学部

2020 年度

卒　業　研　究

Ethereum による貸し出し承認システムの開発と評価

学位請求者　先端メディアサイエンス学科

高松毅瑠

目次

第 1 章

はじめに

1.1　研究背景

2020 年コロナウイルス感染対策のために出勤をできるだけ減らす試みが進んでおり，在宅でのリモートワークが推奨されている．しかしオンラインで承認することができない手続きが多くあり，承認のためだけに出勤する必要があることもある．

例えば Excel を使った備品を管理する応用を考える．共有されたファイルにはアクセス制御がなく，後から借りた備品の情報を書き換える不正が生じる可能性がある．第三者の承認なく不正に備品を持ち出す事も懸念される．

1.2　研究目的

本研究では，ブロックチェインシステムを用いてこれらの認証の偽装や文章の不正更新を防止し，オンラインで承認と第三者による検証可能なシステムの開発を試みる．安全性と利便性を実現したシステムを実装し，その性能評価を報告する．

第 2 章

貸し出し承認システム

2.1　概要

本研究室ではクラウドのファイル共有システムと Excel を使って，備品の貸し出しの依頼と管理者の承認をすることで備品を管理している．この管理法では管理者に連絡し，承認を受け，自分で Excel 上に記録することで備品を借りる事が出来る．処理の流れを図 2.1 に示す．

この手法の問題点として，

- 管理者になりすましが出来る点
- 別の生徒になりすましが出来る点
- 一旦書き込んだデータを後から改竄できてしまう点
- 管理者の不正を防ぐ事ができない点．

の 4 点が挙げられる

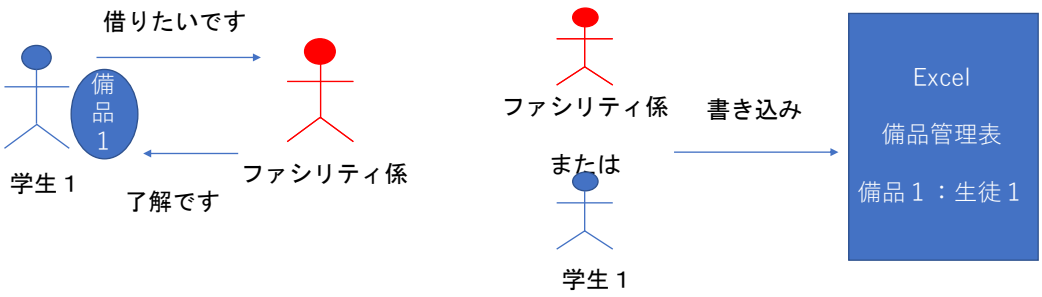

図 2.1　従来の貸し出し処理

第 3 章

要事技術

3.1　ブロックチェイン

ブロックチェインはネットワークに分散されたデータベースである．非中央集権的なシステムである事．記録されているデータを誰でも確認できる事，改竄が困難である事，が特徴である．

ブロックチェインでは取引（トランザクション）をブロックにまとめ記録している．また，ブロックはタイムスタンプ，前のブロックのハッシュ値，ナンス，トランザクションから成る．この構造により，あるブロックを改竄しようとするとそのブロック以降の全てのブロックを改竄する事が必要になり，改竄を困難にしている．

3.2　Ethereum

イーサリアム (Ethereum) はブロックチェイン技術を暗号資産以外の領域で使うために作られた暗号資産である．また，ここで使う仮想通貨をイーサ (Eth) と呼ぶ．

3.3　スマートコントラクト

スマートコントラクトは取引などの契約をブロックチェインに書き込み，自動で実行する機能である．あらかじめ決められた条件に当て嵌まった時のみプログラムが実行される仕組みになっている事で，不正な第三者の介入も，相手を信頼する必要もなく取引を行う事が出来る．

3.4　Solidity

イーサリアムではスマートコントラクトを実装するためのプログラミング言語として Solidity がある．Solidity で書かれたプログラムをブロックチェイン上に配置し，そのプログラムを Gas と呼ばれる手数料を支払って実行する事でスマートコントラクトを動作させる．

3.5　MetaMask

MetaMask は google crome のプラグインとして利用できるウェブウォレットであり，これを利用する事で Eth の支払いを伴うプログラムをウェブ上で実行する事ができる．

第 4 章

提案手法

4.1　概要

本研究では，研究室内での備品の貸し出しをブロックチェイン技術を用いてシステム化し，運用にかかるコストと有用性を評価する．

4.2　システム構成

システム構成図を図 4.1 に示す．ブロックチェインには貸し出し状態を 2 つの情報に分けて記録している．1 つは貸し出しリクエストの状態，2 つ目は備品の状態である．また，この 2 つはシステム利用者の入力によって変化する．処理の流れを図 4.2 に示す．処理の流れは以下 4 つのステップで行われる．

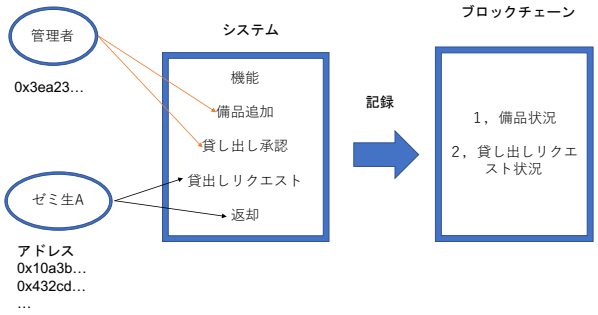

図 4.1　システム構成図

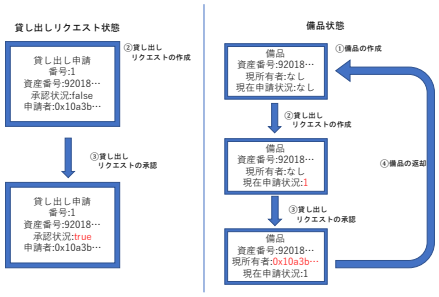

図 4.2　処理フロー

(1) 管理者が資産番号をブロックチェイン上に記録する.
(2) 生徒が管理者に貸し出しリクエストを送信する.
(3) 管理者が貸し出しリクエストを承認する.
(4) 生徒が利用が終わった備品を返却する際にブロックチェイン上に記録する.

備品の登録，貸し出しの承認は管理者のアドレスでのみ行うことができる.

4.3 ソースコード

開発したシステムのソースコードをソースコード 4.1 に示す.

Listing 4.1 check.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.8.0;

contract check {
    address[16] public adopters;
    string defaultMessage;
    address checker;
    event NewRequest(string _name, address _from);
    event AgreeRequest(string _name, address _from);
    event returnRequest(string _name, address _from);

    constructor() public {
        defaultMessage = "Hello World!";
        checker = 0x8A541e657AffE2d2B0f9A48dDB80a620A13c8FDf;
    }

    struct Contracts {
        uint256 id;
        string name;
        bool agree;
        address from;
    }

    struct Facility {
        bool rent;
        address ownerA;
        uint256 last;
```

```
}
Contracts[] public contracts;
mapping(string => Facility) public facility;

function getAdopters() public view returns (address) {
    return checker;
}

function getMessage() public view returns (string memory) {
    return (defaultMessage);
}

function Createcontract(string memory _name) public {
    facility[_name].last = contracts.length;
    contracts.push(Contracts(contracts.length, _name, false, msg.sender));
    emit NewRequest(_name, msg.sender);
}

function viewcontractname(uint256 id) public view returns (string memory) {
    return contracts[id].name;
}

function viewcontractFrom(uint256 id) public view returns (address) {
    return contracts[id].from;
}

function viewcontractAgree(uint256 id) public view returns (bool) {
    return contracts[id].agree;
}

function agreecontract(uint256 id) public {
    require(varifyChecker());
    contracts[id].agree = true;
    ownerFacility(contracts[id].name, contracts[id].from);
    emit AgreeRequest(contracts[id].name, contracts[id].from);
}
```

```
function varifyChecker() public view returns (bool) {
    return (checker == msg.sender);
}

function NewChecker(address New) public {
    require(varifyChecker());
    checker = New;
}

function returnContract(string memory _number) public {
    require(msg.sender == facility[_number].ownerA);
    returnFacility(_number);
    emit returnRequest(_number, msg.sender);
}

function makeFacility(string memory _number) public {
    require(varifyChecker());
    require(facility[_number].ownerA == address(0x0));
    facility[_number] = Facility(false, checker, 0);
}

function ownerFacility(string memory _number, address ownera) public {
    require(varifyChecker());
    facility[_number].ownerA = ownera;
    facility[_number].rent = true;
}

function returnFacility(string memory _number) public {
    facility[_number].ownerA = checker;
    facility[_number].rent = false;
    facility[_number].last = 0;
}

function viewFacility(string memory _number)
    public
    view
    returns (
```

```
            bool,
            address,
            uint256
        )
    {
        return (
            facility[_number].rent,
            facility[_number].ownerA,
            facility[_number].last
        );
    }

    function viewFacilityLast(string memory _number)
        public
        view
        returns (uint256)
    {
        return facility[_number].last;
    }
}
```

第 5 章

実験

5.1　実験目的

- システム運用が可能であるか検証する．
- 提案手法によるシステム導入時間，コストを評価する．

5.2　実験環境

- 12 月 5 日から 15 日までの 10 日間に渡り，菊池研究室内 18 名にシステムの導入から備品の貸し出し，返却まで行った．その際にかかった時間，手数料などを計測した．
- 12 月 22 日システムの全機能を利用し，その手数料を調査した．

5.3　実験結果

被験者は全員システム導入に成功した．システム導入のためにかかった時間の平均は約 15 分であり，最大は 30 分，最短で 3 分 42 秒であった．また，30 分かかった際にはシステムにバグが起こっておりバグの対応に時間がかかってしまった．プログラムを実行した結果を表 5.2 に示す．実験は 12 月 22 日に行い，実験の際は 1Eth ＝ 61,695.57 円であった (2020/12/22 17:01:03)．実験の結果システムの導入には約 2000 円，その後機能を使う毎に手数料 (2 から 6 円) がかかる．

表 5.1　プログラム実行によるコスト

機能名	動作	手数料
デプロイ	プログラムをブロックチェインに配置する．	0.03590924
makeFacility	備品の情報を記録する	0.000049033
CreateContract	貸し出しの申請を行う	0.000116253
viewContractname	申請の情報を見る（書き込みなし）	0.0
agreeContract	申請に許可を出す	0.00005085
returnFacility	備品の返却をする	0.000025

表 5.2　従来手法との比較

	(1) 管理者へのなりすまし	(2) 持ち主へのなりすまし	(3) 書き込み後の改竄	(4) 管理者の不正書込み，改竄
本手法	○	○	○	○
従来手法 (Excel)	×	×	×	×

5.4　従来手法との比較

従来手法の問題点であった (1) 管理者へのなりすまし，(2) 持ち主へのなりすまし，(3) 書き込み後の改竄，(4) 管理者の不正書込み改竄，を比較する．

(1)(2) 本手法で作ったシステムに対してなりすましを行うには管理者の秘密鍵，所有者の秘密鍵を各々持っていなければならないため，Metamask での各々管理する過程の下では，不正は不能である．(3)(4) 不正な取引をブロック上に記録するには攻撃者がネットワーク全体のマイニングの 51 %以上を支配する必要があり，不正は困難なものになっている．書き込み後の修正はブロックチェインの性質上，書き換えが行われるとブロック上にその記録が残り，検出されずに不正な書き換えを行うことはできない．従来手法は以上の不正の全てに脆弱である．以上の比較を表 5.2 に整理する．

第 6 章

おわりに

本研究では Ethereum を用いた貸し出し承認システムを開発した．ブロックチェインの性質である改竄を防止した安全なシステムを開発した．システムの開発，利用にはデプロイが (2020/12/22 時点) 約 2000 円，1 つの貸出しリクエストを承認するまでに約 10 円かかる事を示した．本手法では 1 つの貸し出しに時間がかかり，すぐに持ち帰る事ができないと言ったデメリットがある．安全性の面とともに検討していくことを今後の課題とする．

謝辞

本研究を行うにあたり，多くの方より御指導いただきました．特に明治大学総合数理学部先端メディアサイエンス学科，菊池浩明教授に深く感謝申し上げます．予備実験等に協力してくださった菊池研究室の皆様並びに先端メディアサイエンス学科の方々に深く感謝の意を表するとともに，謝辞とさせていただきます．

参考文献

[1] 廣澤 龍典，上原 哲太郎, “ブロックチェインを用いた検証可能な抽選システムの提案”, *Computer Security Symposium (2019),pp,776-783,2018*,

[2] フォン ヤオカイ，松本 晋一, 穴田 啓晃, 川本 純平, 櫻井 幸一, “次世代暗号通貨プラットフォーム Ethereum の実験的評価”, *Computer Security Symposium (2015),pp,1151-1158,2014*,

付録 A

歩容データからの歩きスマホの検出

A.1　はじめに

A.2　研究背景

歩きながらスマートフォンの操作を行う，いわゆる，「歩きスマホ」が近年問題になっている．歩きスマホを行っている人は注意散漫になり，他の歩行者と衝突してしまう恐れがある．歩きスマホ検出の技術はいくつか提案されている．加藤らは Realtime Multi-Person Pose Estimation を用いた姿勢推定によって得られた姿勢情報をもとに，スマホ使用姿勢検出，歩行検出，把持物体認識の 3 ステップによる歩きスマホ検出手法を提案している，従来の歩きスマホ検出・認識ではなされていない 8 方向を向いた人物の動画による検証を行い，F1-score にして 0.852 という高い精度での検出が可能であることを確認した．本手法は単眼 RGB 画像を用いた歩きスマホの判別であり，高度な画像処理と姿勢推定に大きなコストがかかっていた．そこで本研究では，モーションキャプチャーによって外部から歩き方を観測し，歩きスマホを自動検出することを試みる．歩様データから右手と左肘間の距離の統計量などの「歩きスマホ」に固有の関節の組み合わせによる特徴量を調査し，機械学習アルゴリズムにより，汎用性のある歩きスマホの検出方式を提案するシステムを実装し，121 名の歩容データを用いた検出精度を報告する．

A.3　研究目的

本研究では，歩容を観察し歩きスマホの検出のために有効な特徴量を明らかにすること．提案手法による歩きスマホの判別精度を明らかにすることの 2 点を研究目標とする．

A.4　提案手法

A.5　概要

本研究では，モーションキャプチャデバイス kinect v2 による姿勢推定から通常歩行，歩きスマホのそれぞれについてサイクル切り出しを行い，1 歩分の歩行データと定める．歩行データに対して特徴量を作成，特徴量のランダムフォレストによる学習を行い，歩きスマホの検出器を作成する．

表 A.1　実験データ

被験者	通常歩行	歩きスマホ
人数 [人]	40	40
データ数	200	200

A.6　適合率，再現率，F 値

本研究では，歩きスマホ検出器の精度として，適合率 P と，再現率 R，P と R の調和平均である F 値により評価する．適合率は歩きスマホと予測したデータのうち，正しいものの割合であり，適合率は前歩きスマホのデータのうち検出できたものの割合である．F 値は適合率と再現率の調和平均をとった値であり，各々次の様に定める．

$$P = \frac{\text{歩きスマホと正しく判定したデータ数}}{\text{歩きスマホと判別したデータ数}}$$

$$R = \frac{\text{歩きスマホと正しく判定したデータ数}}{\text{歩きスマホのデータ数}}$$

$$F\ \text{値} = \frac{2}{(1/R) + (1/P)}$$

A.7　特徴量

通常歩行と歩きスマホの判別に使う特徴量を見つけるため，全身 25 個の関節の組み合わせ，関節間の距離を取り，時系列データから，平均，標準偏差，最大値，最小値を用いて，特徴量をそれぞれ 300 個作成する．図 A.1 は関節（1,2），(3,4)，(4,5）間の距離の時系列データにした時の図でありこの時系列データから統計量を用いて算出した値を特徴量とする．

A.8　クロスバリデーション

モデルの汎化性能の評価のため，本研究ではクロスバリデーションを用いて適合率，再現率の計算を行う．クロスバリデーションでは 200 のデータを 50 ずつ 4 つのグループに分割しそれぞれのグループを一度ずつ学習データとし，それ以外のデータをテストデータとして 4 回検証し，適合率と再現率それぞれ平均の値を算出する．

A.9　ランダムフォレスト

特徴量を基にランダムフォレストを用いて判別を行なう．ランダムフォレストはジニ係数が 0.13 以下になるか，1 つのノードに所属するデータの個数が 1 0 個以下になるまで枝を伸ばす．100 個の決定木を作成し，それらの決定木から判別を行う．平均，標準偏差，最大値，最小値などの統計量及び，平均と標準偏差，最大値と最小値，の様に複数個の特徴量を組み合わせる．

図 A.1　特徴量の例とその変化

表 A.2　統計量毎の再現率，適合率，F 値

	平均値	中央値	標準偏差	最大値	最小値
適合率	0.973	0.971	0.864	0.962	0.945
再現率	0.913	0.913	0.846	0.905	0.900
F 値	0.942	0.941	0.855	0.932	0.926

A.10　実験

A.11　実験目的

1. 歩容を観察し，歩きスマホの検出のために有用な特徴量を明らかにする．
2. 提案手法による歩きスマホの判別精度を明らかにする．

A.12　実験環境

データ取得には Microsoft 社のモーションキャプチャデバイス，Kinect v2 を用いる．Kinect v2 には RGB カメラ，深度センサ，マイクなどが搭載されており，体の関節の 3 次元座標を推定し，その変化に基づいて人の動きを認識する．Kinect v2 は体の 25 の関節の 3 次元座標を測定する．

A.13　データ取得

本研究では 2018 年 7 月，明治大学中野キャンパス多目的室において，実験協力の同意を取って，121 名の歩容データを取得した．床から 0.9m の位置に固定した Kinect v2 から 5.5m の離れた位置を被験者の歩行開始地点とし，1m の位置を歩行終了地点とする. 歩容の測定は 4.5m 地点から 2m 地点までの区間で行う．121 名に 5 回歩行してもらい，測定を行った．121 名の内表 1 に示す 80 名の 400 データを実験に用いる．

A.14　実験結果

ランダムフォレストにより作成した決定木の 1 つを図 A.2 に示す．ここで，class2 が「歩きスマホ」1 が通常歩行である．統計量毎の適合率，再現率を表 A.2 に示す．最も F 値が高くなった統計量は平均値であり適

表 A.3　特徴量を組み合わせた場合の F 値

	平均	中央値	標準偏差	最大値	最小値
平均		0.947	0.949	0.945	0.939
中央値	0.947		0.951	0.941	0.940
標準偏差	0.949	0.951		0.938	0.927
最大値	0.945	0.941	0.938		0.938
最小値	0.939	0.940	0.927	0.938	

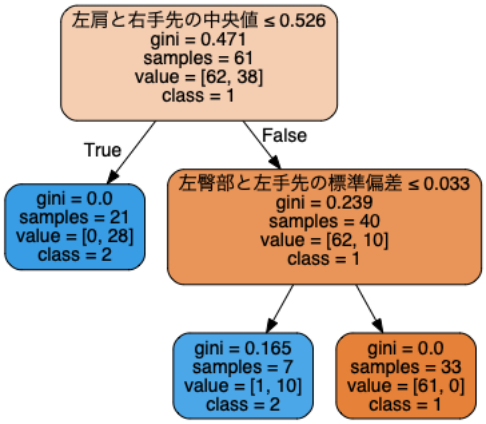

図 A.2　決定木の例

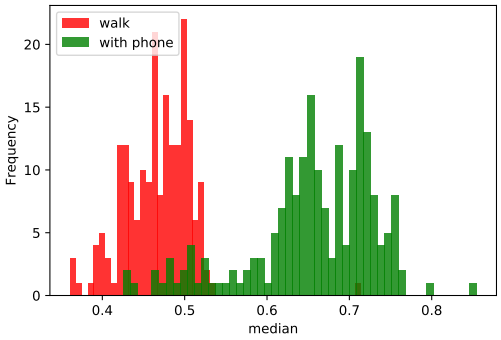

図 A.3　右手と左膝の中央値の分布

合率は 0.973，再現率は 0.913 である．各統計量を組み合わせたランダムフォレストによる判別の結果を A.3 に示す．最も F 値が高くなった統計量の組み合わせは中央値と標準偏差を組み合わせた時で適合率 0.990，再現率 0.915 である．中央値と標準偏差を使ったランダムフォレストの作成において最も重要度が高くなった特徴量は右手と左膝の中央値である．最も重要度が高いと判断された右手と左膝の中央値のヒストグラムを A.3 に示す．図 A.3 より，通常歩行（赤）と歩きスマホ（緑）の分布が明確に分離しているのが明らかである．

A.15　考察

表 A.2，A.3 より，特徴量単体では適合率，再現率が低かった標準偏差と中央値を組み合わせることで精度を上げる事ができた．標準偏差を使うことで特徴量の分散を表すことが出来るため，変動の激しい特徴量を対処出来たと考える．図 A.3 より，多くの人が歩きスマホの検出には右手と左膝の中央値を使って判別することが出来る．誤差の原因は左利きの被験者のデータであり，利き手の情報を含む特徴量の追加が必要である．

A.16　終わりに

本研究では歩きスマホの特徴的な行動を調査，検出を目的に歩きスマホ検出器を作成し，80 名の被験者による実験の結果，中央値と標準偏差を組み合わせたランダムフォレストによって適合率 0.990，再現率 0.915 で歩きスマホが検出可能であることを示した．

歩きスマホの検出には右手と左膝の中央値が有効であり，1 つの特徴量で適合率 0.994 再現率 0.895 で歩きスマホが検出可能であることを示した．

参考文献

[1] 森　駿文, 菊池　浩明, ” 歩容データの DTW 距離に基づく個人識別手法の提案と外乱に対する評価”, マルチメディア，分散，協調とモバイルシンポジウム *(DICOMO 2018)*, pp. 672-680, 2018.

[2] 森　駿文, 菊池　浩明, ” 複数の歩容特徴量の k 近傍による「歩きスマホ」にロバストな個人識別手法の提案”, 暗号と情報セキュリティシンポジウム *(SCIS 2019)*, pp. 1-7， 2019.

[3] 三好駿, 森駿文, 菊池浩明, 歩容データからの属性暴露リスクについて, 暗号と情報セキュリティシンポジウム *(SCIS 2019)*, pp. 1-7， 2019.

[4] 加藤丸君, 渡辺裕, 姿勢情報を用いたカメラ映像からの歩きスマホ検出 *2017* 年度 早稲田大学大学院修士論文