ブロックチェーンを用いた出席管理システムの提案

塚田研究室　218K6078　檜山　祐太

1　はじめに

各種学校や塾などでは、出欠確認や集計処理などの出席管理に関する作業は多くの労力を必要とし、その負担は小さくはない。しかし、学生の出席状況を把握することは必要不可欠であるため、近年では、ICT 技術を用いてより効率よく出欠を確認することができる出席管理システムの導入が数多く進められている。[1]

そこで本研究ではブロックチェーンを用いた出席管理システムのアーキテクチャを設計し、そのプロトタイプを実装する。

2　ブロックチェーン

ブロックチェーンとはデジタルな資産の移転や取引などの履歴データを複数の利用者と管理者によって共有される仕組みのこと。ブロックチェーンはいくつかの取引情報をブロックとしてまとめることで、矛盾した取引情報がないようにしている。また、取引情報をブロックとしてまとめる際に、矛盾した取引情報が同じブロックに含まれないように検証が行われる。ブロックは生成される際、１つ前のブロックのハッシュ値を含むことで連鎖する形態をとる。このブロックの連鎖のことをブロックチェーンという。

ブロックチェーンのモデル図を図１に示す

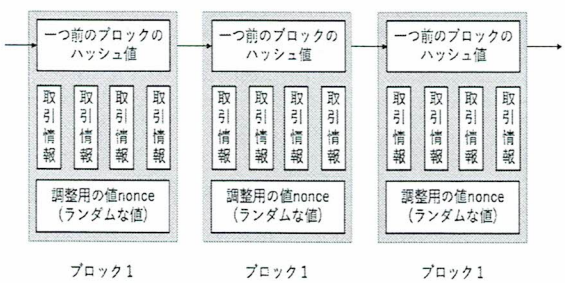

図１　ブロックチェーンのモデル図

2－1　ブロック構造

ブロックチェーンの各ブロックは１つ前のブロックのハッシュ値と複数の取引情報、調整用のランダムな値である nonce によって構成される。

2－2　プルーフオブワーク

ブロックチェーンはビットコインによって生まれた仕組みであり、ビットコインはブロックチェーンの原点と言われている。ビットコインではブロックの生成は多数の作成者の競争で行う方法が採用されている。ブロックを生成するには、大量のハッシュ値計算が必要で、新たなブロックを生成できた者には、新たに発行する暗号通貨とブロックに含まれた取引情報の手数料の合計が与えられる。このハッシュ値計算の競争のことをプルーフオブワークと称されている。

2－3　ブロックの生成

プルーフオブワークはブロック生成競争の基準となるターゲットの値よりも小さなハッシュ値を競争して探すものである。ブロックチェーンに記録されていない取引情報のハッシュ値とハッシュ値を合わせて、ハッシュ関数にかけることでハッシュ値を得るという処理をツリーのような構造で繰り返し、最終的に得られた１つのハッシュ値と、１つ前のブロックのハッシュ値、ランダムな値である nonce を１つ決めてハッシュ値計算する。その結果得られたハッシュ値とターゲットの値を比べ、ハッシュ値の方が小さい場合はそれらのデータを格納し、１つのブロックを生成する。そしてそのブロックを他のノードに送付し、受信したノードはそのブロックの答え合わせを行う。同じ手順で計算し合っていれば、新しいブロックとして受け入れられる。また、ターゲットとなる値は、過去の正解を出すまでの所要時間から導出され、正解が出るまでの時間が約１０分間となるように調整がされる。

2－4　ブロックチェーンの改ざん

ブロックチェーンの改ざんは容易ではない。なぜなら、攻撃対象のブロックを書き換えた場合、その書き換えたブロックのハッシュ値が変化するため、それ以降全てのブロックを作り直す必要がある。それに加え、次々に行われるブロック生成を追い越さ

なければいけないため、ブロックチェーンの改ざんは容易ではない。

3　出席管理システム

出席管理システムとは、学校や塾などにおいて、生徒の出欠状況を管理するシステムのことをいう。出席管理システムには様々な種類があり、中でもICカードを読み取るカードリーダを用いた出欠システムが広く導入が進んでいる。また、各種デバイスを管理端末とする出席管理システムも増加していて、こうしたシステムでは学生が所持しているスマートフォンやパソコンを使用し、クラウドで提供されるものもあるため、専用機器を準備する必要がなく、手軽かつ低コストで出席管理が可能。さらに、出席管理にとどまらず、出席管理システムによって収集した情報を活用し、他のシステムと連携させることで学生指導や学校運営を支援するパッケージシステムとして提供されるものもあります。[1]

4　アーキテクチャ設計

本研究では、生徒の出席情報がサーバに送信された際にサーバを介してブロックチェーンに登録する。学校はブロックチェーンに出席情報の確認申請を行うことで、部屋ごとの出席情報と生徒ごとの出席情報の確認を行うことができる。

4－1　要件定義

本研究で提案するアーキテクチャには次の３つの要件を定義することにした。

(1)　改ざん不可能：ブロックチェーンを用いて、出席者の情報を改ざん不可能にする。

(2)　出席情報の登録：生徒の出席情報がサーバに送られた際、その時点での時刻とともに生徒の出席情報をブロックチェーンに登録する。

(3)　出席情報の公開：学校はブロックチェーンに出席情報の確認申請を行うことで、部屋ごとの出席情報と生徒ごとの出席情報の確認を行うことができる。

4－2　システム概要

本研究で提案する出席管理システムのアーキテクチャを図２に示す。

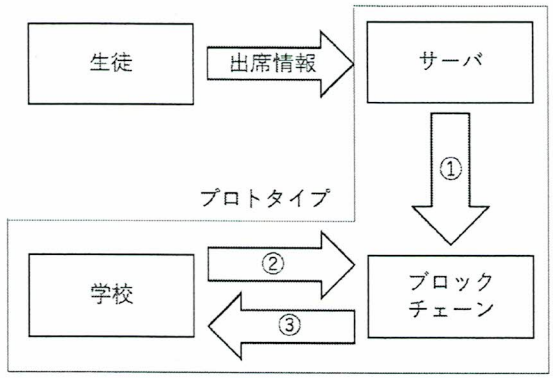

図２　アーキテクチャ図

①サーバに送られてきた生徒の出席情報をブロックチェーンに登録する。

②学校はブロックチェーンに出席情報の確認申請を行うことができる。

③ブロックチェーンシステムに学校から出席情報の確認申請が行われた場合、ブロックチェーンシステムから出席情報を学校に送信する。

今回の出席管理システムでは、生徒の出席情報をサーバに送信する部分以外の基本部分をプロトタイプとして実装する。

5　システム詳細設計

5－1　開発環境

本プロトタイプの作成においては、ブロックチェーンを利用した非中央集権アプリケーションであるEthereum を使用した。また、ソースコードはSolidity 言語を利用して記述し、Solidity コンパイラのバージョンは 0.4.15 を使用した。開発環境はRemix[3]にて実行した。

5－2　プロトタイプ概要

今回の出席確認システムは図２で示したプロトタイプ部分である。本プロトタイプは生徒の出席情報がサーバに送られてきた際に、その出席情報をブロックチェーンに登録し、学校による部屋ごとの出席情報と生徒ごとの出席情報の確認までの動作を目的としたシステムである。

部屋名と生徒名はあらかじめプログラム上に３つずつ登録しておくものとし、そのソースコードを図３に示す。また、部屋名を追加で登録する addRoom 関数のソースコードを図４に示す。生徒名を追加で

登録する addStudent 関数のソースコードを図５に示す。

```
string[] public rooms = ["roomA", "roomB", "roomC"];
string[] public students = ["studentA", "studentB", "studentC"];
```

図３　部屋名と生徒名のソースコード

```
function addRoom(string _room) isOwner public returns (bool) {
    require (roomData[_room].isValue == false);
    rooms.push(_room);
    roomData[_room].isValue = true;
    return true;
}EOF
```

図４　addRoom 関数のソースコード

```
function addStudent(string _student) isOwner public returns (bool) {
    require (studentData[_student].isNumber == false);
    students.push(_student);
    studentData[_student].isNumber = true;
    return true;
}EOF
```

図５　addStudent 関数のソースコード

サーバに生徒の出席情報が送られた際に、その時点での時刻とともにその出席情報をブロックチェーンに登録する Attendance 関数のソースコードを図６に示す。

```
function Attendance(string _room, string _student)
    isExist(_room) isNumber(_student)public {
    uint time = now;
    roomData[_room].attendanceDB.push(
        AttendanceData({
            attendanceTime : time,
            attendanceStudent : _student
        })
    );
    studentData[_student].studentDB.push(
        StudentData({
            attendanceTime : time,
            attendanceRoom : _room
        })
    );
    AttendanceLog(_room, time, _student);
}EOF
```

図６　Attendance 関数のソースコード

学校がブロックチェーンに部屋ごとの出席情報の確認申請を行う際の getAttendanceDB 関数のソースコードを図７に示す。

```
function getAttendanceDB(string _room) isExist(_room) public {
    for (uint i = 0; i < roomData[_room].attendanceDB.length; i++) {
        AttendanceLog(
            _room,
            roomData[_room].attendanceDB[i].attendanceTime,
            roomData[_room].attendanceDB[i].attendanceStudent
        );
    }
}EOF
```

図７　getAttendanceDB 関数のソースコード

学校がブロックチェーンに生徒ごとの出席情報の確認申請を行う際の getStudentDB 関数のソースコードを図８に示す。

```
function getStudentDB(string _student) isNumber(_student) public {
    for (uint i = 0; i < studentData[_student].studentDB.length; i++) {
        AttendanceLog(
            _student,
            studentData[_student].studentDB[i].attendanceTime,
            studentData[_student].studentDB[i].attendanceRoom
        );
    }
}EOF
```

図８　getStudentDB 関数のソースコード

６　実行結果

本プロトタイプを実際に動作させた結果、出席情報の登録や出席情報の確認が正しく動作することが確認できた。また、登録していない部屋名や生徒名が入力された際は、正しく動作し、ブロックチェーンへの登録は行われなかった。

(1)　“roomA” に “studentA” が出席したことを、サーバに生徒の出席情報が送られた際に、その時点での時刻とともにその出席情報をブロックチェーンに登録する Attendanse 関数で実行した結果を図９に示す。

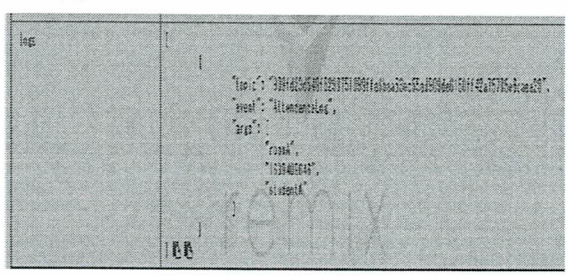

図９　Attendance 関数の実行結果

(2) “roomA” の出席情報を、学校がブロックチェーンに部屋ごとの出席情報の確認申請を行う際の getAttendanceDB 関数で実行した結果を図１０に示す。

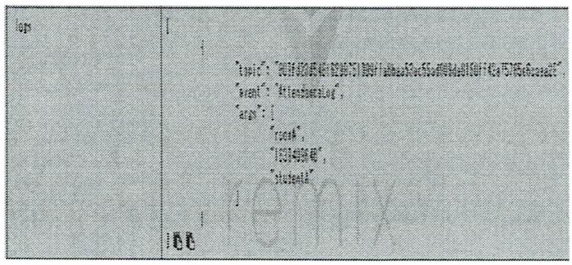

図１０　getAttendanceDB 関数の実行結果

(3) “studentA” の出席情報を、学校がブロックチェーンに生徒ごとの出席情報の確認申請を行う際の getStudentDB 関数で実行した結果を図１１に示す。

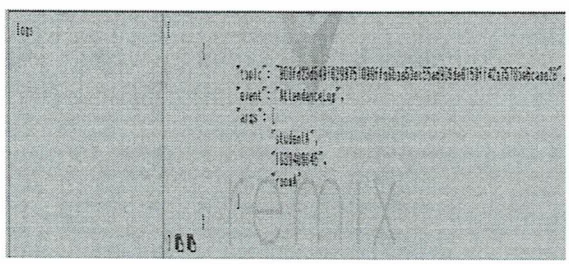

図１１　getStudentDB 関数の実行結果

しかし、今回のプロトタイプでは誰でも関数を実行できるようになっているため、関数を実行できる人に制限を設けたほうがいいと考える。また、今回のプロトタイプでは出席情報を削除する関数を導入していないため、古くなり必要がなくなった出席情報の削除ができないので、出席情報を削除する関数を実装したほうがいいと考える。

7　まとめ

ブロックチェーンを用いた出席管理システムを提案し、そのアーキテクチャを示すとともに、プロトタイプを実装した。その結果、出席情報の登録や出席情報の確認が正しく動作することが確認できた。

今回のシステムでは、管理者が存在せず誰もが全ての関数を実行できるようになっているため、関数を実行できる人に制限を設け、限られた人しか実行できないようにすることが今後の課題だ。また、出席情報を削除する関数を導入していないため、古くなり必要がなくなった出席情報の削除ができないので、出席情報を削除する関数を実装することも今後の課題だと考える。

参考文献

[1] 出席管理システムの概要とおすすめのシステム11選｜HRテック｜ミツカルは、資料が”見つかる”ビジネスメディア ミツカル
https://mitsu-karu.com/article/attendance-management/

[2] 佐藤雅史、長谷川佳祐、佐古和恵、並木悠太、梶ヶ谷圭佑、松尾真一郎：ブロックチェーン技術の教科書、C&R 研究所(2018)

[3] Remix - Solidity IDE
https://remix.ethereum.org