情報処理学会論文誌 Vol.61 No.12 1859–1870 (Dec. 2020)

ブロックチェーンを用いた透明性のある抽選システムの提案と実装

廣澤 龍典[1,†1,a)] 上原 哲太郎[2,b)]

受付日 2020年3月9日, 採録日 2020年9月10日

概要：インターネットやスマートフォンの普及により，オンラインサービスにおいてゲームや懸賞などにおける抽選の機会が増えている．しかし，抽選に用いられる乱数の生成手法は非公開であることが多く，抽選結果もその乱数の生成結果を正しく反映しているか確認は難しい．そのため，ユーザは抽選の結果に納得できない場合がある．そこで我々は，ブロックチェーンを用いた透明性のある抽選システムを提案する．スマートコントラクトによって乱数の生成手法は公開され，乱数の生成結果としての賞品などの対応付けはブロックチェーン上に記録・公開されるため，ユーザは抽選の正当性を確認することが可能となる．Ethereum 上に実装したプロトタイプシステムによって，本提案手法が実用的な時間およびコストで実装可能であることを示した．

キーワード：ブロックチェーン，Ethereum，乱数，抽選

Transparent Loot Box System Using Blockchain

Tatsunori Hirosawa[1,†1,a)] Tetsutaro Uehara[2,b)]

Received: March 9, 2020, Accepted: September 10, 2020

Abstract: With the widespread use of the Internet and smartphones, today we often open the loot boxes to win lotteries for games and prizes in online services. However, the methods used to generate random numbers to decide prizes are not disclosed in many cases and it is also difficult to verify whether the result of the lottery reflects the random number generation accordingly. It might cause users' dissatisfaction with the results of the lottery. In this paper, we propose a blockchain-based transparent lottery system. Smart contracts make the algorithm of random number generation available to the public, and the correspondence of resulting prizes can be recorded and published on the blockchain, so users can verify the validity of the lottery. The prototype system on Ethereum shows that our proposed method can be implemented in a practical time and cost.

Keywords: blockchain, Ethereum, random number, loot box

1. 序論

近年，インターネットやスマートフォンの普及により，オンラインサービスにおける抽選の機会が多くなっている．たとえば，SNS を利用した懸賞では，ユーザはシステムと擬似的なジャンケンやトランプによって勝敗を決め，勝てば賞品を受け取ることができる．また，オンラインゲームにおいては，そのゲーム内において使用できるアイテムを抽選で配布している．しかし，オンラインサービスにおける抽選は，現実の抽選より不正を行いやすい．

このような抽選では，希少度の高いアイテムを獲得する

[1] 立命館大学大学院情報理工学研究科
Graduate school of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525–0058, Japan
[2] 立命館大学情報理工学部
College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[†1] 現在，株式会社 NTT データ
Presently with NTT DATA Corporation
a) hirosawa@cysec.cs.ritsumei.ac.jp
b) t-uehara@fc.ritsumei.ac.jp

ためにユーザが自身の想定以上に高額[*1]の金銭を使わされることがある．よって，サービス運営者が不正を行い，生成された乱数やその乱数と賞品の対応付けを偽っていないかどうか，その正当性を担保する必要がある．しかし，ユーザは抽選に用いられる乱数の生成手法や，アイテムが公表された確率どおりに配布されているかは，基本的に確認できない．そのため，ユーザは抽選の結果に納得できない場合がある．そこで本稿では，ブロックチェーンを用いた透明性のある抽選システムを提案する．

2. 研究背景

2.1 オンラインゲームの概要

オンラインゲームには，ユーザがアプリケーションストアからダウンロードする際に料金が発生しないものがある．ユーザにとってゲームを購入する際に料金が発生しないことは魅力的であると同時に，サービス運営者にとってもユーザ誘引のための手段となっている．

ユーザはゲーム内で使用できる仮想のキャラクタやアイテム（以下「賞品」と記述する）を「ガチャ」[*2]と呼ばれる抽選で獲得することができる．一般的にガチャによる抽選を行うことを「ガチャを引く」と表現することがある．

ユーザがガチャによって獲得できる賞品は基本的にランダムである．ユーザがガチャを引いて賞品を獲得する場合には，そのゲーム内でのみ使える通貨（以下「ゲーム内通貨」と記述する）を用いる必要がある．ユーザは法定通貨を支払ってゲーム内通貨を購入するほか，ゲーム内で一定の条件をクリアすることで無償で手に入れる場合もある．**図 1** にゲーム内通貨の購入画面の例[*3]を示す．後述する「10 連ガチャ」にはゲーム内通貨が 500 個必要となっており，その購入価格は 3,060 円となっている．

本ゲームのガチャには「10 連ガチャ」と呼ばれる，ユーザによる 1 度の操作で 10 回連続でガチャを引き，10 個の賞品を獲得する方法がある．他のゲームにおいても 10 連ガチャに類似する，ユーザによる 1 度の操作によって連続でガチャを引く機能が用意されている場合がある．

図 2 にガチャを引く際の画面を示す．図 2 において，右下にある「10 回引く」ボタンが 10 連ガチャに該当する．本ゲームの 10 連ガチャでは，ユーザが「1 回引く」を 10 回行う場合と比べて必要なゲーム内通貨の割引はない．しかし，サービス運営者はユーザが 10 連ガチャを引いた際に，高いレアリティ（希少度）が割り当てられている賞品を必ず 1 個獲得できることを保証している．具体的には，

図 1 ゲーム内通貨を購入する際の例
Fig. 1 An example to buy in-game currency.

図 2 ガチャを引く際の例
Fig. 2 An example to open a loot box.

図 3 ガチャにおける賞品の提供確率の表示例
Fig. 3 An example of the probability of winning prizes in a loot box.

「10 回引く」ボタンの上に「★4 以上 1 枚確定」という表記がある．オンラインゲームにおいては，星のイメージとその数でレアリティを表すことがある．

また，サービス運営者はユーザがガチャで獲得できる賞品の一覧と各賞品の提供確率を表示している．**図 3** にサービス運営者が示した，ガチャにおける提供確率の表示画面を示す．図 3 の上部にある「ガチャ出現率」における，星が 5 つ並んでいるイメージとその下にある 5.0%とある表記は，レアリティ 5 に属する賞品すべての提供確率を足し合わせたパーセンテージが 5.0%であることを表す[*4]．図 3

[*1] 文献 [1] によると 90 万円以上の課金額を支払ったプレイヤがゲーム運営側を訴えた例がある．
[*2] JOGA ガイドライン [2] においては「ランダム型アイテム提供方式」と定義される．
[*3] 画像は株式会社ディー・エヌ・エー（DeNA）が提供している「歌マクロス　スマホ De カルチャー」（公式サイト：https://uta-macross.jp）である．画像の取得は 2019 年 10 月 18 日に行った．図 2・図 3 も同様である．
[*4] 本ゲームの確率表示は小数第 4 位以下を切り捨てているため，確率を合算しても 100%にならない場合がある．

の中央部にある「ガチャ出現一覧」には，各レアリティごとに賞品名とその提供確率が示されている．賞品名の右側のパーセンテージより，サービス運営者は賞品「Halloween Candy ★ Night」を1.501%の確率で提供していることが分かる．また，サービス運営者は一部の賞品について，1%を下回る提供確率を設定している．

2.2 業界団体やプラットフォーマによる自主規制

サービス運営者が前述のように，賞品の提供確率を表示している理由として，業界団体やプラットフォーマがガイドラインを定めていることがあげられる．

一般社団法人日本オンラインゲーム協会および一般社団法人コンピュータエンターテインメント協会はオンラインゲームにおけるガイドライン [2], [3] を定めている．これらのガイドラインは，ガチャで獲得できる賞品とその提供確率の表示，提供確率の不適切な変更の禁止などを定めている．

また，Apple・Google はそれぞれのアプリケーションストアで提供するアプリケーションに対するガイドライン [4], [5] を定めている．いずれのガイドラインも賞品の入手確率の明記を義務付けており，サービス運営者はガイドラインを満たさないアプリケーションをアプリケーションストアで配信できない．

2.3 法律による規制

不当景品類及び不当表示防止法[*5]（以下「景品表示法」と記述する）は商品および役務の取引に関連する不当な景品類および表示による顧客の誘引を防止するために施行されている．景品表示法の第五条は，事業者が供給する商品または役務の取引において不当表示（優良誤認表示や有利誤認表示）の禁止について定めている．

また，消費者庁はインターネット上の取引に対する景品表示法の適用を解説した資料として，Q&A 集 [7] を公開している．Q&A 集にはオンラインゲームにおける景品表示法の適用について，様々な仮定においての検討がなされている．Q&A 集の Q12 には “オンラインゲームには，「カード合わせ」に該当するいわゆる「コンプガチャ」以外にも射幸心を著しくあおる課金の仕組みがたくさんあります．これらの課金方法も景品表示法上問題なのではないですか” という質問がある．これに対する回答の後半に “オンラインゲーム上で販売されるアイテム等が実際のものよりも著しく優良であると示す表示や当該アイテム等の取引条件などが実際のものよりも著しく有利であると誤認される表示がなされた場合にも，景品表示法上問題となります” との一文がある．これはサービス運営者が，ガチャで提供する賞品の提供確率を偽ったり，ガチャで提供していない賞品を提供しているかのように表示したりした場合，不当表示に該当することを意味する．

2.4 ガチャおよび規制の課題点

ガチャの課題点として，サービス運営者による不正が現実の抽選に比べて行いやすいという点がある．比較のため，実物の抽選機が存在する抽選としてガラガラ抽選とぱちんこ遊技機（以下「パチンコ」と記述する）を例に説明する．ガラガラ抽選では，サービス運営者が箱の中に玉を入れておく．抽選参加者は抽選機をまわし，まわしている間に出てきた玉の色によって異なる賞品を獲得できる．この方法では，抽選機をまわそうとする抽選参加者は，不正がないかを確認することができる．具体的には，抽選参加者は抽選機をまわす前に抽選機内にある玉の色と大きさを確認し，玉を排出する穴の大きさを確認する．一方，パチンコは保安通信協会によって，型式試験が行われている．型式試験では，試験を受けようとするパチンコが「遊技機の認定及び型式の検定等に関する規則」（昭和 60 年国家公安委員会規則第 4 号）の別表第 4「ぱちんこ遊技機に係る技術上の規格」に示されている性能や規格を満たしているかが確認される．パチンコは上記の形式試験への合格を含めた所定の手続きを行わなければ，パチンコ店でサービスを提供することができない仕組みとなっている．

しかし，ガチャは抽選機の実物がなくゲームサーバ内で動作するため，ユーザがガチャの実際の動作内容（プログラムや賞品ごとの提供確率）を確認することができない．仮に，ユーザがゲームサーバにおけるガチャの動作内容を確認できた場合でも，実際にガチャの動作がユーザの端末において同様に行われており，サービス運営者が表示している賞品の提供確率が守られていることを確認できなければ，ユーザにとってガチャの正当性が確保されたとはいえない．つまり，ユーザはサービス運営者が自ら表示しているガチャの説明・賞品の提供確率や，2.2，2.3 節で述べた規制を守っているかを確認できないという課題が生じる．サービス運営者による不正事案として 2018 年 1 月，希少な賞品の提供確率を実際より高く見せかけたとして，ゲーム会社「アワ・パーム・カンパニー・リミテッド」が消費者庁より再発防止のための措置命令を受けた [8]．特定の賞品が 3%の確率で当たると告知していたが，実際は 0.33%であった．

また，各ユーザが他のユーザのガチャ結果を確認することは難しい．そのためユーザは「自分だけがガチャの結果が悪い」と思い込み「自分だけが損をさせられている」と感じる恐れがある．その結果，抽選結果やサービス運営者に対して不信感を持つことがある．これらの問題を解決するためには，ガチャに対する規制を加えたり，関係機関による試験を義務化したりすることが必要であるといえる．

しかし，関係機関による試験については，試験を義務付

[*5] 昭和 37 年法律第 134 号 [6].

ける法律が存在しないため実施は困難である．仮に実施できた場合でも前述のとおり，試験を行い合格したシステムが，各ユーザの端末で動作していることを関係機関がすべて確かめることは現実的ではない．つまり，実物の機械が存在するパチンコと異なり，オンラインゲームについては試験を行っても意味をなさない可能性がある．

また，2.2 節で述べたガイドラインは法的拘束力を持たない．そのうえ，景品表示法による不当表示の禁止については，サービス運営者が明らかな不当表示を行っていない限り適用は難しい．そのため，ユーザがサービス運営者に対して，ガイドラインや法律に基づいて表示の是正や支払った金銭の返金などの対応を求めることは困難である．具体的には「星のドラゴンクエスト」というゲーム内のガチャにおいて説明に問題があり，これが不当表示にあたるとして複数のユーザによって集団訴訟が行われた．しかし，東京地方裁判所[*6]および控訴審を取り扱った東京高等裁判所[*7]はいずれも不当表示には該当しないとして，原告（ユーザ）側の請求を棄却している．

一般的にはガチャにおける賞品の提供確率が表示されているとはいえ，ガチャは乱数を用いている．そのため，ガチャにサービス運営者による不正があり，レアリティが高い賞品が表示されている確率どおりに出ない場合でも，サービス運営者は「そのユーザが不利に感じているのは乱数の偏りが原因であり，偶発的に発生したものにすぎない」と主張する可能性もある．よって，ガチャの透明性を確保するための技術の使用が必要である．

2.5 ブロックチェーン

2.5.1 概要

ブロックチェーンとは，P2P ネットワークを用いて世界中にデータを分散させて保存する仕組みである．ブロックチェーンは Nakamoto の論文 [9] において提唱された「Bitcoin」と呼ばれる暗号資産（仮想通貨）の仕組みとして発表された．Bitcoin においては，取引のデータの最小単位のことをトランザクションと呼び，トランザクションを複数個まとめたものをブロックと呼ぶ．そして，Bitcoin はブロックをハッシュチェーンでつなぐ仕組み（ブロックチェーン）を利用することで取引データを保存している．従来の分散データベースと比べたブロックチェーンの大きな特徴は，ブロックチェーンに書き込まれたデータの改ざんがきわめて困難である点と，ユーザ間において中央集権的なサーバを必要とせずに全体データの整合性を維持できる点である．ブロックチェーンの構造およびブロックが持つ情報を**図 4** に示す．ブロックチェーンは名称どおり，ブロックを時系列に従ってチェーンのようにつないだものである．ブロックチェーンは Bitcoin に限らず，他の暗号資産においても利用されている．

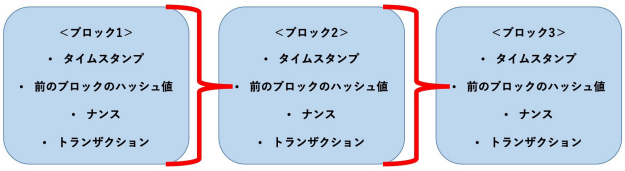

図 4 ブロックチェーンの構造
Fig. 4 The structure of a blockchain.

2.5.2 コンセンサスアルゴリズム

コンセンサスアルゴリズムとは，P2P ネットワークにおける参加者の間で，ネットワーク全体の合意（コンセンサス）をとる手続きのことである．Bitcoin においては，一定の条件を満たせばだれでも「マイナー」と呼ばれるブロックの作成者になることができる．しかし，マイナーが制約なしにブロックを作り，ブロックチェーンにつないでしまうと二重支払い問題をはじめとする不正を防止できない．そこで Bitcoin ではネットワーク参加者らが「Proof of Work」と呼ばれるコンセンサスアルゴリズムを用いることで，ブロックチェーンにつなぐ新たなブロックの合意をとっている．

具体的には，マイナーはブロックのデータに「ナンス」と呼ばれる任意の値を加えてハッシュ値を求める．Bitcoin においては，ブロックのハッシュ値は先頭に「0」が一定数含まれなければならない[*8]．ハッシュ関数は一方向性を持つため，規定を満たすハッシュ値からナンスを逆算することはできない．つまり，マイナーは規定を満たすハッシュ値を見つけるためにはナンスを総当たりしなければならない．よって，ナンスおよび完成するブロックのハッシュ値の予測操作は困難である．

2.6 Ethereum と Solidity

Bitcoin は暗号資産をやりとりすることに主眼を置いているため，カスタマイズ可能な領域が少ない．この問題を「Ethereum」と呼ばれる “分散アプリケーション向けのグローバルなオープンソース・プラットフォーム” [10]（ブロックチェーン基盤）が解決した．Ethereum においては暗号資産「Ether」（単位は Ether）が実装されており，Bitcoin と同じように，ブロックチェーンを利用することで取引の正当性を保証している．Ether をやりとりする際や後述のスマートコントラクトを扱う際には「アドレス」と呼ばれる，0〜9 および a〜f を用いた英数字の羅列を用いる．

Bitcoin と異なり Ethereum は，専用のプログラミング言語を用いることで，スマートコントラクトを記録・実行で

*6 東京地方裁判所平成 29 年（ワ）第 40855 号．
*7 東京高等裁判所平成 30 年（ネ）第 4535 号．

*8 ただし，一部の暗号資産（たとえば後述する Ethereum）においては，ブロックのハッシュ値の先頭に「0」が一定数含まれていなくても正当なブロックとして認められる．Ethereum におけるブロックのハッシュ値の例：0x012b9700ffce43787cdcc89da2806（以下省略）．

きる．Ethereum における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実行されるプログラムのことを指す．スマートコントラクトは，暗号資産のやりとり同様にマイナーがマイニングを行うことで，ブロックチェーンに記録される．ブロックチェーンに記録する際には作成者を表すアドレスも一緒に記録される．このため，ブロックチェーンを閲覧するツールを用いることで，だれでもスマートコントラクト作成者のアドレスや内容の正当性，動作内容の検証を行うことが可能である．また，スマートコントラクトを実行する過程でブロックチェーンへ情報を記録した場合は，記録内容と記録者のアドレスがブロックチェーンに記録され，だれでもその内容を確認できる．

「Solidity」は Ethereum においてスマートコントラクトを記述するためのプログラミング言語である．Solidity はスマートコントラクトを扱うための特有の設計や実装もあるものの，uint や string などの変数をはじめとし，順次処理・分岐処理・反復処理といったプログラムの基礎的な動作が可能である．

Solidity によって記述されたスマートコントラクトは，Ethereum Virtual Machine と呼ばれるスマートコントラクトの実行環境において，バイトコードに変換したうえで実行される．ユーザがスマートコントラクトを実行する際や，スマートコントラクトをデプロイする際には「Gas」（単位は gas）と呼ばれる実行手数料をマイナーに対し Ether で支払わなければならない．支払う Gas は，1 gas あたりの価格とスマートコントラクトの実行に必要なリソース量から計算される．ただし，ブロックチェーンに情報を書き込まない（状態遷移をともなわない）プログラムについては，実行手数料は不要である．

2.7 関連研究

佐古らは SCIS2017 において「ブロックチェーンを用いたオンラインゲーム用公平性を検証可能な乱数発生」[11] という論文を発表している．この論文では，バックギャモンと呼ばれるオンラインゲームにおいて，生成される乱数を Bitcoin のブロックチェーン上に記録している．そして，ブロックチェーンに記録された乱数を検証することで，ユーザに対して生成された乱数が公平なものであると納得してもらおうとしている．しかし，この手法はサーバを介した通信を行っているため，ユーザとサーバが結託することで不正が成立してしまうという課題点がある．

江原らは CSS2017 において，前述した問題の克服を目指し「ブロックチェーンによる乱数生成の透明性確保」[12] という論文を発表している．この論文では，ブロックチェーンを利用しトラストレスな状況でも正当な方法で乱数生成を行い，その乱数列を検証する方式を提案している．また，抽選の透明性を確保し，抽選の結果に納得してもらえるよう，乱数を生成する際のシード値にユーザおよびサービス運営者がそれぞれ作成した乱数を含めている．

Teranishi らは SCIS2019 において，“Dice Rolls for Online Games using Blockchain with Provable Properties” [13] という論文を発表している．この論文では，佐古らの論文 [11] において，第三者が抽選結果の正当性を検証できないことを課題と考えている．そして，トランザクション間に不整合がある場合に，だれでもその不整合の責任者を決定できるような手法を提案している．

渡辺らは「はじめてのブロックチェーン・アプリケーション Ethereum によるスマートコントラクト開発入門」[14] という書籍の第 6 章において，スマートコントラクトを用いた乱数生成について考察を行い，公平な乱数生成手法を考案している．

Bonneau らは “On Bitcoin as a public randomness source” [15] という論文を発表している．この論文では，ビットコインを用いて乱数生成を行う手法が提案されている．また，セキュリティの向上に関する考察を行っている．

Bünz らは “Proofs-of-delay and randomness beacons in Ethereum” [16] という論文を発表している．この論文では，ブロックチェーンを用いた乱数生成における課題としてマイナーによる操作をあげている．そして，操作への対策として遅延関数を検証するための新しいマルチラウンドプロトコルを導入し，性能を解析するためプロトタイプを実装しコスト評価を行っている．

既存研究の課題点として，特に江原らの手法に注目する．江原らの手法では納得感の向上のために，乱数生成におけるシード値に，ユーザとサービス運営者がそれぞれ生成した乱数を使用している．しかし，サービス運営者が乱数であってもシード値に含める行為は，ユーザに無用な不信感をいだかせることになりうる．サービス運営者はシステムを制作する立場であることに加えて，資本力がユーザに比べて大きく，ブロックハッシュの操作に成功する可能性はユーザより高い．また，2.4 節で述べたとおり，制作したシステムに関する規制は不十分である．よって，サービス運営者が不必要にシード値に関与する機会は取り除くべきであると考え，シード値の見直しが必要であると結論づけた．また，江原らの研究をはじめとして上記関連研究はいずれもスマートコントラクトで乱数の生成のみを行っており，乱数と賞品の対応付けを行っていない．乱数の生成に加えて乱数と賞品との対応付けを行うことで，ガチャを模した抽選システムを実現することができる．そして，いずれの既存研究においても，1 度生成された乱数は除外されず，同じ乱数が複数回生成される可能性がある．生成された乱数を除外しない手法は，一定数ある賞品をユーザが山分けするような抽選には不適切である．以上のことから，シード値を見直した「既存手法よりユーザが関与できるガチャを模した抽選システム」（以下「ガチャシステム」と記述する）および「1 度生成された乱数が除外される抽選シ

ステム」（以下「重複除外システム」と記述する）の提案を行う．

3. 提案手法

江原ら・渡辺らは，Ethereum ブロックチェーンを用いて，ある時点における未採掘ブロックのハッシュ値を乱数生成のシード値に含めていた．上記ハッシュ値を使う理由は，ユーザとサービス運営者がともに，生成される乱数を予測操作困難にすることで抽選結果の操作を防ぐためである．

そのほか，関連研究では同じタイミングであってもユーザごとに結果が変わるよう，ユーザごとに異なるものとして「ユーザアドレス」，同じユーザであっても抽選 1 回ごとに結果が変わるよう「抽選を行った数」をシード値に含めていた．以上をふまえてガチャシステムおよび重複除外システムのコンセプトを述べる．

3.1 ガチャシステムのコンセプト

ガチャシステムでは，上で述べた関連研究で用いられた手法を公平さを保つために踏襲する．本システムにおける既存手法との相違点は

(1) ユーザは自身の好きな文字列を乱数のシード値に含められる，
(2) サービス運営者は乱数のシード値に関与できない，
(3) 抽選システムは生成した乱数と賞品の対応付けをスマートコントラクト内で行う，

の 3 点である．

相違点 (1) について，ユーザはサービス運営者と比べて不利な立場にあるため，任意の文字列を乱数のシード値に含められるようにする．シード値に未採掘ブロックのハッシュ値も含んでいるため，ユーザが乱数のシード値に好きな文字列を加えた場合でも特定の抽選結果に操作することはできない．

相違点 (2) についてサービス運営者はサービスを運営・管理する性質上，有利な立場にある．そのため，サービス運営者はユーザが抽選結果を操作できない限り，過度にシード値に関与する必要はない．そして，提案手法では未採掘ブロックのハッシュ値を乱数のシード値に含めることで，ユーザによる抽選結果の操作は不可能である．よって，サービス運営者はシード値に関与できないようにする．

相違点 (3) について 2.1 節で述べたガチャを模することを目的とする．スマートコントラクト内で乱数と賞品の対応付けを行うことにより，サービス運営者もしくは抽選システムが乱数を正しく生成したが，乱数と賞品の対応付けにおいて抽選結果を偽るという不正を防ぐことができる．また，ユーザは 1 回の操作で，複数回分の抽選結果を得られるようにする．これは 2.1 節で述べた「10 連ガチャ」に相当する機能である．10 連ガチャに相当する機能を実装することで，本システムにおけるガチャの再現度を向上させる．

3.2 重複除外システムのコンセプト

重複除外システムはガチャシステムと異なり，1 度生成された乱数を除外し，同じ乱数が生成されない抽選システムとする．具体的には，サービス運営者が上記のガチャのように無限に賞品を提供するのではなく，一定数の賞品を提供しようとするとき，異なる複数のユーザに賞品を割り振る場面を想定している．これは，オンラインゲームにおいて，ミッションクリアなどの進行の区切りに到達したユーザらが一定数の賞品を山分けする場面に相当する．ただし，本システムはプロトタイプとして開発したため，生成した乱数と賞品の対応付けは行わず，乱数の生成のみを行う．本システムのポイントは

(1) ユーザらは自身の好きな文字列を乱数のシード値に含められる，
(2) サービス運営者は未採掘のブロックの番号を決める，
(3) ユーザ間において公平性が保たれるよう対策を行う，

の 3 点である．ポイント (1) は 3.1 節で述べた理由と同じ理由で採用した．ポイント (2) について，サービス運営者は乱数生成に用いる未採掘のブロックの番号を決定する．ポイント (3) に記述した公平性について，本提案手法ではスマートコントラクトを用いるため，抽選に関する情報や処理が公開される．また，複数のユーザ（仮にユーザ A・ユーザ B とする）が抽選に参加する場合を想定している．そのため，ユーザ A は自身がいらないと考える賞品を，ユーザ B が獲得したことをブロックチェーンで確認することが可能となる．そのため，ユーザ A はユーザ B が抽選した後で抽選を行うことで，自身が欲しい賞品の獲得確率を上げることができる．

しかし，同じ立場であるユーザ間で公平性が損なわれてはならない．そのため，抽選に参加するユーザらの各抽選をすべて同じタイミングで行うようにする．

3.3 エンティティ

本抽選システムにおける登場エンティティについて述べる．

ユーザ
一意の識別子（2.6 節で述べたアドレス）を持ち，サービス運営者が提供する抽選システムを用いて抽選を行う．ユーザ間の通信は発生せず，ユーザは抽選システムに対してのみ通信を行う．

サービス運営者
一意の識別子を持ち，ユーザに対して抽選システムを提供する．

スマートコントラクト
サービス運営者によって作成されたプログラムであ

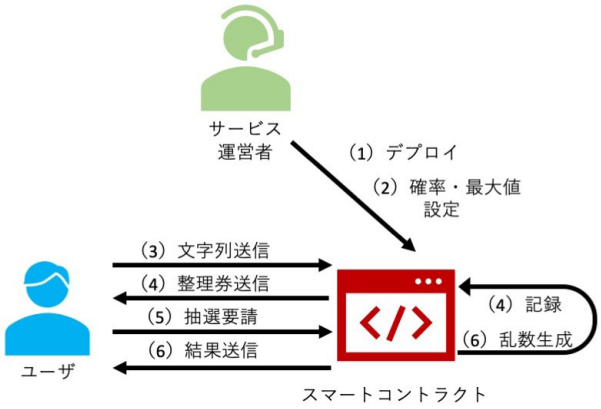

図 5 ガチャシステム
Fig. 5 The proposed loot box system.

る．ユーザおよびサービス運営者から独立した信頼できるブロックチェーン上で動作する．ユーザ・サービス運営者を含め，だれでも閲覧できるものとする．

3.4 ガチャシステムの詳細

ガチャシステムにおける乱数生成および賞品との対応付けの手法を**図 5** および下記の箇条書きに示す．図中の番号と箇条書きの番号は対応している．

(1) サービス運営者は，スマートコントラクトをデプロイする．
(2) サービス運営者はスマートコントラクトが生成する乱数の最大値を設定し，各賞品の確率を設定する．
(3) ユーザは，自身の好きな文字列 P をスマートコントラクトに送信し，整理券番号 N をリクエストする．
(4) スマートコントラクトは「ユーザが送信した文字列 P・リクエスト送信時点における最新のブロック番号」を記録し，これらの情報を紐付けた整理券番号 N をユーザに返す．
(5) ユーザは整理券番号 N と抽選を行う回数をスマートコントラクトに送信する．
(6) スマートコントラクトはユーザから受信した整理券番号 N に紐付いた情報に基づいて「手順 4 時点では未採掘だったブロックのハッシュ値 BH・ユーザのアドレス A・整理券番号 N・ユーザが指定した文字列 P」をまとめてハッシュ化したものをシード値 S として乱数を生成する．生成した乱数がどの賞品に該当するか確認し，ユーザに賞品名を返す．

シード値 S の生成を計算式で表すと

$$S = keccak256(abi.encodePacked(BH, A, N, P))$$

となる．S は本来 BH，A，N，P から得られる暗号学的ハッシュ関数であれば任意の関数から得られるが，ここでは文献 [17] に定義されたハッシュ関数 keccak が Solidity の関数として提供されているため，これを用いて表現した．Solidity の関数 abi.encodePacked を使って複数の入力

```
maxNumber,normal,rare,ultimate
numberedTicket=0
record[][]

function 確率と最大値の登録 (賞品 Normal の範囲, 賞品 Rare の範囲, 賞品 Ultimate
の範囲, 生成される乱数の最大値){
  normal=賞品 Normal の範囲
  rare=賞品 Rare の範囲
  ultimate=賞品 Ultimate の範囲
  maxNumber=生成される乱数の最大値
}

function 文字列の登録 (ユーザの好きな文字列){
  record[numberedTicket][ユーザのアドレス]=ユーザの好きな文字列, 最新のブロック番号
  return numberedTicket++
}

function ガチャの実行 (numberedTicket, 抽選回数){
  randomNumber[],prize[],i,seed

  seed=keccak256(abi.encodePacked(record 内のブロック番号の次のブロックのハッ
シュ値, ユーザのアドレス,numberedTicket,record 内のユーザの好きな文字列))
  Xoshiro256**(seed)
  for(i<抽選回数,i++){
   randomNumber[i] = Xoshiro256**() % maxNumber
   if(randomNumber[i]<=normal) prize[i]="Normal"
    elseif(randomNumber[i] > normal && randomNumber[i] <= rare)
prize[i]="Rare"
    elseif(randomNumber[i] > rare && randomNumber[i] <= ultimate)
prize[i]="Ultimate"
  return randomNumber[],prize[]
}
```

図 6 ガチャシステムの疑似コード
Fig. 6 The pseudocode of loot box system.

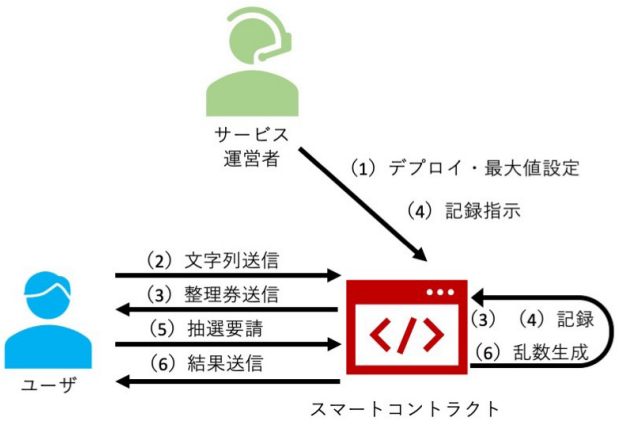

図 7 重複除外システム
Fig. 7 The deduplicated loot box system.

を単一のバイト列に変換したうえで，ハッシュ化したものをシード値 S としている．ここまでの手順を**図 6** に疑似コードとして示す．

ここで Xoshiro256** は，文献 [18] による疑似乱数であり，計算量が小さいため本提案の実装で用いている．ただし，疑似乱数の計算量はスマートコントラクトの実行コストのためには小さく抑えられた方が良い．

3.5 重複除外システムの詳細

重複除外システムにおける乱数生成の手法を**図 7** および下記の箇条書きに示す．図中の番号と箇条書きの番号は対応している．

(1) サービス運営者はスマートコントラクトをデプロイし，スマートコントラクトが生成する乱数の最大値を設定する．
(2) ユーザは自身の好きな文字列 P をスマートコントラクトに送信する．

```
maxNumber,blockNumber
numberedTicket=0
record[]

function 最大値の登録 (生成される乱数の最大値){
  maxNumber=生成される乱数の最大値
}

function 文字列の登録 (ユーザの好きな文字列){
  record[ユーザのアドレス]=ユーザの好きな文字列
  return numberedTicket++
}


function ブロックの確定 (){
  blockNumber=最新のブロック番号
  return numberedTicket++
}

function 乱数の生成 (numberedTicket){
  randomNumber[],seed
  i=0
  checkFlag[]=False

  seed=keccak256(abi.encodePacked(blockNumber のブロック番号の次のブロック
のハッシュ値,record 内のユーザらの好きな文字列をすべて結合させた文字列))
  Xoshiro256**(seed)

  while(i<maxNumber){
   randomNumber[i] = Xoshiro256**()) % maxNumber
   if(checkFlag[ randomNumber[i] ]==False) {
    checkFlag[ randomNumber[i] ]=True
    i++
   }
  }
  return randomNumber[numberedTicket]
}
```

図 8 重複除外システムの疑似コード
Fig. 8 The pseudocode of deduplicated loot box system.

(3) スマートコントラクトは「ユーザが送信した文字列 P・ユーザのアドレス」を記録し整理券番号をユーザに返す．
(4) サービス運営者は，抽選に参加するユーザが全員手順 (2) を行ったことを確認してから，スマートコントラクトに，手順 (4) 時点における最新のブロック番号を記録させる．
(5) ユーザは整理券番号をスマートコントラクトに送信する．
(6) スマートコントラクトは「手順 (4) 時点では未採掘だったブロックのハッシュ値 BH・ユーザらが指定した文字列 P を結合したもの P'」をまとめてハッシュ化したものをシード値 S として抽選に参加したユーザごとの乱数を一括で生成する．具体的には，生成された乱数を添字とするブーリアン型配列を作っておき，乱数が生成された時点で該当する配列の中身を True に変えてその乱数が生成されたことを記録しておくことで，重複する乱数を生成しないようにする．生成した乱数のうち整理券に該当する乱数をユーザに返す．

シード値 S の生成を計算式で表すと

$$S = keccak256(abi.encodePacked(BH, P'))$$

となる．提案手法を図 8 に疑似コードとして示す．

3.6 実装方法

3.6.1 システム全体

ブロックチェーンおよびスマートコントラクトのプラットフォームは Ethereum を選択した．Ethereum スマートコントラクト開発用の総合開発環境として Remix [*9] を用いた．テスト環境用ブロックチェーン（プライベートネットワーク）の構築には Geth [*10] を用いた．Geth のバージョンは 1.8.27 である．また，スマートコントラクトを記述する際のプログラミング言語は Solidity を使用し，そのバージョンは 0.4.26 とした．乱数生成のアルゴリズムとして「Xoshiro256**」を用いた．Xoshiro256** アルゴリズムの詳細は文献 [18] を参照されたい．両システムのソースコードは Github [*11] に掲載している．

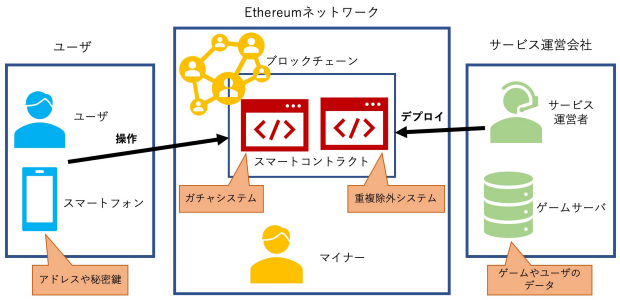

図 9 システム想定図
Fig. 9 Proposed system.

本稿において想定したシステムの全体像を図 9 に示す．ユーザはスマートフォン内のゲームアプリケーションでゲームをプレイする．また，今回提案したスマートコントラクトを，ユーザはスマートフォンによって操作することを想定した．そのためにスマートフォンは，スマートコントラクトを操作するために必要な「ユーザアドレス」や「ユーザの秘密鍵」を保有する．

サービス運営会社では，サービス運営者がゲームを管理している．そして，ゲームデータやユーザデータを保管するゲームサーバがある．サービス運営者が作成したスマートコントラクトをデプロイすることで，ユーザはスマートコントラクトを使用可能になる．

Ethereum ネットワークにはマイナー（ネットワーク参加者）によって正当性が担保されているパーミッションレスタイプのブロックチェーンがある．ブロックチェーンにはスマートコントラクトを記録することができる．

本稿では，提案手法で述べた 2 つの抽選システムのスマートコントラクトのみを実装した．前述した統合開発環境を用いることで，ユーザによる操作やサービス運営者によるスマートコントラクトのデプロイや設定を，擬似的に実現している．

3.6.2 ガチャシステム

ソースコードは Github 上に掲載しているが，後のコスト評価に用いるため，ガチャシステムが持つ関数の概要を

[*9] http://remix.ethereum.org/
[*10] https://geth.ethereum.org/
[*11] https://github.com/cysec-lab/usableSecurity2020

示す．

setProbability 関数 本関数は 3.4 節の手順 (2) に対応し，サービス運営者が賞品ごとの提供確率を設定する際に利用する．

request 関数 本関数は 3.4 節の手順 (3)・(4) に対応し，ユーザから文字列を受け付けて，その時点の最新のブロック番号とともにブロックチェーンに記録し，整理券番号を返す．

3.6.3 重複除外システム

ソースコードは Github 上に掲載しているが，後のコスト評価に用いるため，重複除外システムが持つ関数の概要を示す．

registInfo 関数 本関数は 3.4 節の手順 (2)・(3) に対応し，ユーザから受け付けた文字列とユーザのアドレスをブロックチェーンに記録し，整理券番号を返す．

decideBlock 関数 本関数は 3.4 節の手順 (4) に対応し，関数実行時における最新のブロック番号を記録する．

reset 関数 本関数はシステム再利用のために，システムが生成する乱数の最大値の再設定および整理券番号を初期化する関数である．

4. 実験

4.1 ガチャシステムへの実験

実装したガチャシステムが，提案手法に沿った動作を行えるか確認した．また，本システムをデプロイする際に発生したコストと，本システムを実行する際に発生したコストについても確認した．

4.1.1 実験内容

ガチャシステムが生成する乱数の最大値を 1,000 に設定した．乱数と賞品の対応付けとして，3 種類（Normal・Rare・Ultimate）の賞品の提供確率がそれぞれ 50%，30%，20%となるよう設定した．そして，1 度の操作で 10 回分の抽選結果を獲得できるかを確認した．ユーザが整理券を要請する際に，本システムに送信する文字列は “Ritsumeikan” とした．

4.1.2 実験結果

ガチャシステムによる乱数生成結果を評価するため，実際に Remix 上で関数を実行した結果を**図 10** に示す．関数 GetPrize を実行した結果このような値が得られた．図中の箇条書きにある 0 はシード値に含めたブロックハッシュで型は bytes32，1 は乱数の生成結果で型は uint64 の配列，2 は 1 で生成された乱数と紐付けられた賞品名で型は string であることを示している．今回の抽選では乱数「133, 374, 863, 932, 253, 565, 865, 492, 146, 93」が生成され，それに対応する賞品として「Normal, Normal, Ultimate, Ultimate, Normal, Rare, Ultimate, Normal, Normal, Normal」が獲得できた．

また，本実験においてスマートコントラクトを実行する際に発生したコストを**表 1** に示す[*12]．

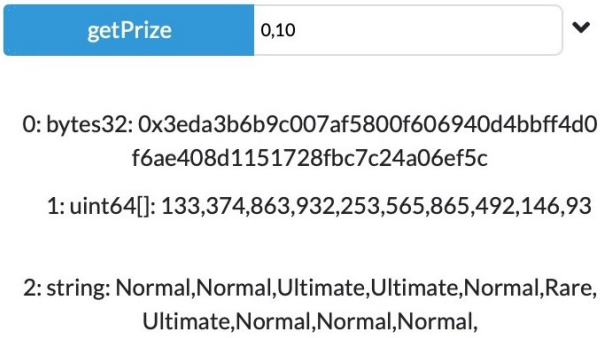

図 10 実験結果（ガチャシステム）
Fig. 10 Experimental result (loot box system).

表 1 ガチャシステムのコスト
Table 1 Execution cost (loot box system).

	Gas	Ethereum 換算	日本円換算
デプロイ	1,054,358 gas	0.0132364113 Ether	374.5 円
setProbability	50,335 gas	0.00063190563 Ether	17.9 円
request	86,771 gas	0.00108932321 Ether	30.8 円

実験時の為替レート[*13]は 0.00000001255400092 Ether/gas，および 28290.6872 JPY/Ether であった．

4.2 重複除外システムへの実験

実装した重複除外システムが，提案手法に沿った動作を行えるか確認した．また，本システムをデプロイする際に発生したコストと，本システムを実行する際に発生したコストについても確認した．

4.2.1 実験内容

重複除外システムが生成する乱数の最大値を 20 に設定した．ユーザが本システムに送信する文字列は「ritsX」とした（ただし，X には順に 1〜20 の数を入れる．例：rits1, rits2, ..., rits20）．実験後には reset 関数を用いて整理券番号を 0 に戻し，スマートコントラクトが生成する乱数の最大値を 20 に再設定した．

4.2.2 実験結果

重複除外システムによる乱数生成結果を評価するため，実際に Remix 上で関数を実行した結果の一部を**図 11** に示す．関数 seeNumber を実行した結果このような値が得られた．図中の箇条書きにある 0 はシード値に含めたブロックハッシュで型は bytes32，1 は乱数の生成結果で型は uint256，2 はユーザらが入力した文字列をすべて結合したもので型は string であることを示している．今回の抽選では乱数が「20」が獲得できた．スペースの都合上すべての実験結果を掲載できないが，重複する乱数は存在しな

[*12] 日本円換算に関しては小数第 2 位を四捨五入した．算出は 2020 年 2 月 16 日のレートを用いて行った．次の実験においても同様である．

[*13] https://etherscan.io/chart/gasprice および https://coinmarketcap.com/currencies/ethereum/より算出した．

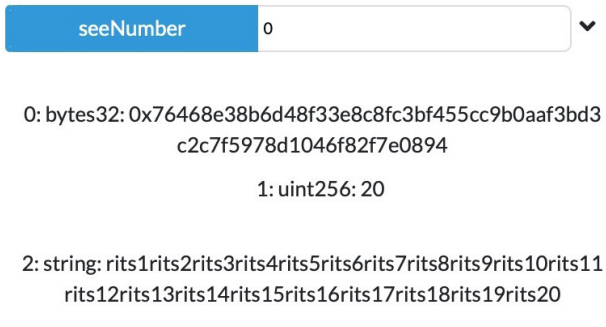

図 11 実験結果（重複除外システム）
Fig. 11 Experimental result (deduplicated loot box system).

表 2 重複除外システムのコスト
Table 2 Exection cost (deduplicated loot box system).

	Gas	Ethereum 換算	日本円換算
デプロイ	922,609 gas	0.01158243423 Ether	327.7 円
registInfo	86,268 gas	0.00108300855 Ether	30.6 円
decideBlock	27,993 gas	0.00035142414 Ether	9.9 円
reset	33,713 gas	0.00042323303 Ether	12.0 円

かった．

また，本実験においてスマートコントラクトを実行するにあたって発生したコストを**表 2** に示す．ここで，registInfo 関数のみ 20 回実行したため，表に示した Gas の約 20 倍[*14]が必要であった．実験時の為替レートは 4.1.2 項と同様である．

5. 考察

5.1 ガチャシステム

図 10 で示した実験結果より，本システムは 2.1 節で示した 10 連ガチャと同じ挙動を再現できた．関連研究では，本実験の内容と同じ動作を行うためには 10 個分のブロックハッシュが必要である．なおかつ，乱数と賞品と対応付けについても別の手法が追加で必要となる．本提案手法では 1 個のブロックハッシュで 10 個すべての乱数が得られ，なおかつ乱数と賞品との対応付けもスマートコントラクト内で行うので，時間の短縮が図れたといえる．また，賞品の決定をスマートコントラクトで行うことにより，既存研究では不可能であった「抽選システムによる乱数と賞品の対応付けにおける不正の防止」が可能となった．

そして，ユーザはスマートコントラクトが持つ性質によって，本システムの動作内容やサービス運営者が設定した確率設定などを確認することができる．加えて乱数のシード値はすべて確認できるようにしているため，ユーザは生成された賞品の検証が可能である．

また，ユーザが文字列をシード値のもとに含められるようにしたことで，サービス運営者とマイナーが結託し，ユーザが文字列を決めた後にシード値を操作しようとした場合，ユーザが文字列を決めた後から賞品が確定するまでの時間は限られている．つまり，ユーザが文字列を加えることができる本手法を用いることで，サービス運営者による不正の機会はより困難になっている．そのため，ユーザからみて透明性あるガチャを実装できたといえる．

本システムでは，スマートコントラクトを実行する際に発生したコストについて，サービス運営者はユーザに合計 423.2 円を課金しなければならない．

ただし，setProbability 関数を用いて，乱数の最大値と各賞品の確率を再設定することにより，サービス運営者は本システムを再利用することが可能である．再利用時はデプロイを行わないため，初回の実行時と比べてコストが 374.5 円低くなる．関連研究 [12] と比較すると，本研究ではサービス運営者がスマートコントラクトをデプロイし直すことなく何度も再利用ができるため，コストを抑えることができ，優位性があるといえる．ただし，機能やソースコードが異なるため，コストについて単純に比較して論じることはできない．

5.2 重複除外システム

本システムもガチャシステム同様に，ユーザはスマートコントラクトが持つ性質によって，本システムの動作内容やサービス運営者がシード値に含めるブロックを決めたタイミングなどを確認することができる．加えて乱数のシード値はすべて確認できるようにしているため，ユーザは生成された乱数の検証が可能である．

本システムは，ゲームにおける賞品の山分けを想定したうえで実装したが，コンサートなどの座席抽選においても使用可能だと考えている．座席抽選もゲームにおける賞品の山分けと同様に，欲しいユーザらが有限個の座席を分ける．座席には会場内における位置や，壇上を見る角度など様々な要因から優劣があるため，コンサート参加者の中で不満が発生することがある．本システムはこのような場面における問題解決も可能ではないかと考えている．

本システムでは，スマートコントラクトを実行する際に発生したコストについて，サービス運営者はユーザに合計 380.2 円を課金しなければならない．ただし，reset 関数を用いて，乱数の最大値と整理券番号を再設定することにより，サービス運営者は本システムを再利用することが可能である．再利用時はデプロイを行わないため，初回の実行時と比べてコストが 327.7 円低くなる．

5.3 リスク評価

本システムは，抽選における乱数発生および賞品割当ての透明性確保は達成されている．そこで，サービス運営者が自らに有利な乱数値を選択できるリスク，およびユーザが自らに有利な乱数値を選択できるリスクについて検討する．

[*14] 2 回目以降の実行では必要な Gas が減少したことにより正確には 20 倍より少ない．

サービス運営者が自らに有利な乱数値を選択できるためには，乱数のシード値 S を自らの有利な値にする必要がある．シード値 S を決定するために必要なパラメータのうち A，N，P が決定するのはユーザが整理券番号 N をリクエストした時点であり，ここからユーザが実際に抽選を依頼するまでの時間にサービス運営者はブロックハッシュ値 BH を何らかの手段で自らの有利となる値になるように調整する必要がある．ユーザは一般に整理券番号を得てから間を置かずに抽選を行うと考えられる．パブリックなブロックチェーンにおいてはサービス運営者がブロックチェーンの新規ブロックを決定するコンセンサスに対して大きな影響力を持つことはできず，本システムにおいて BH を自らの有利な値になるようにはできないと考えられる．

次に，各ユーザが自らに有利な乱数値を選択するリスクについて検討する．これも各ユーザが自らに有利な乱数値を得るためには，BH の決定に関与する必要がある．一般に各ユーザはサービス運営者に対して大きな資本力なども持たないことから，パブリックなブロックチェーンにおいて新規ブロックを決定できる可能性は無視できると考えられ，仮に文字列 P によって乱数値に影響することができても，BH を自由に決定できない以上最終的な乱数値を左右できる可能性は無視できると考えられる．

よって以上より，乱数の決定の過程においてもサービス運営者とユーザ双方にとって不公平の生じないシステムになっていると評価できる．

6. 結論

6.1 まとめ

本稿では，ブロックチェーンを用いた透明性のある抽選システムの提案と実装について述べた．オンラインゲームのガチャには，景品表示法や業界団体・プラットフォーマによる自主規制が適用されている．しかし，ガチャを利用する立場であるユーザの視点では，サービス運営者が管理するサーバ内で動作するガチャは，その動作内容を確認できず不信感をいだくことが多い．そこで，スマートコントラクトを用いた透明性のある抽選システムを 2 種類提案し実装した．

実装したシステムにおいて，ユーザはスマートコントラクトの動作内容を確認でき，乱数生成のシード値もすべて確認できるため，生成された乱数の検証が可能である．よって，ユーザからみて透明性のある抽選システムを実装できたといえる．システムを実行するうえでのコストについては，サービス運営者にシステムの再利用を可能とさせる機能を実装することで，2 回目以降の実行時にコストが下がるようにした．

6.2 今後の課題

今後の課題について「様々な抽選を模してシステムの適用可能場面を増やすこと」と「本番ネットワークにおける動作確認」があげられる．

前者に関して，たとえばオンラインゲームには，2.1 節で述べたレアリティの保証のほか，一定回数のガチャを引くと賞品や賞品の提供確率が変更されるガチャがある．また，座席抽選では複数の座席が隣どうしになることを期待される場合がある．これらの場合でも抽選システムを適切に動作させることで，抽選システムを適用可能な場面を増やすことが考えられる．

後者に関して，今回の実験ではプライベートネットワーク内でのみデプロイを行い実験を行っている．そのため，本番環境の Ethereum ネットワークにデプロイし，正しく動作するか確認を行う必要がある．特に，乱数のシード値に未採掘ブロックのハッシュ値を使っているため，本番環境の Ethereum ネットワークにおけるブロックの採掘速度で，システムが滞りなく動作するかは重要である．また，両システムは簡略化のため，最低限の機能のみを実装している．誤作動を防ぐために，各関数の実行順序を制御する機能などを実装する必要がある．

参考文献

[1] ねとらぼ：スクエニ「星のドラゴンクエスト」ガチャ不当表示で集団訴訟に　1 人で 90 万円以上課金したユーザーも，入手先 ⟨https://nlab.itmedia.co.jp/nl/articles/1801/26/news113.html⟩（参照 2020-01-29）．

[2] 一般社団法人日本オンラインゲーム協会ガイドラインワーキンググループ：ランダム型アイテム提供方式を利用したアイテム販売における表示および運営ガイドライン，入手先 ⟨https://japanonlinegame.org/wp-content/uploads/2017/06/JOGA20160401.pdf⟩（参照 2020-01-29）．

[3] 一般社団法人コンピュータエンターテインメント協会：ネットワークゲームにおけるランダム型アイテム提供方式運営ガイドライン，入手先 ⟨https://www.cesa.or.jp/uploads/2016/release20160427.pdf⟩（参照 2020-01-29）．

[4] Apple：App Store Review ガイドライン – Apple Developer，入手先 ⟨https://developer.apple.com/jp/app-store/review/guidelines/⟩（参照 2020-01-29）．

[5] Google Play：デベロッパーポリシーセンター，入手先 ⟨https://play.google.com/intl/ja/about/developer-content-policy/⟩（参照 2020-01-29）．

[6] 電子政府の総合窓口 e-Gov：不当景品類及び不当表示防止法（景品表示法），入手先 ⟨https://elaws.e-gov.go.jp/search/elawsSearch/elaws_search/lsg0500/detail?lawId=337AC0000000134⟩（参照 2020-01-29）．

[7] 消費者庁：インターネット上の取引と「カード合わせ」に関する Q&A，入手先 ⟨https://www.caa.go.jp/policies/policy/representation/fair_labeling/faq/card/⟩（参照 2020-01-29）．

[8] 消費者庁：アワ・パーム・カンパニー・リミテッドに対する景品表示法に基づく措置命令について，入手先 ⟨https://www.caa.go.jp/policies/policy/representation/fair_labeling/pdf/fair_labeling_180126_0001.pdf⟩（参照 2020-01-29）．

[9] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System，入手先 ⟨https://bitcoin.org/bitcoin.pdf⟩（参照 2020-01-29）．

[10] Ethereum Project: Home – Ethereum, 入手先 ⟨https://www.ethereum.org⟩（参照 2020-01-29）.
[11] 佐古和恵，井口圭一：ブロックチェーンを用いたオンラインゲーム用公平性を検証可能な乱数発生，2017 年暗号と情報セキュリティシンポジウム（SCIS2017），1F2-4 (2017).
[12] 江原友登，多田　充：ブロックチェーンによる乱数生成の透明性確保，コンピュータセキュリティシンポジウム 2017（CSS2017），pp.915–922 (2017).
[13] Teranishi, I. and Sako, K.: Dice Rolls for Online Games using Blockchain with Provable Properties, *2019 Symposium on Cryptography and Information Security* (*SCIS2019*), 2G2-1 (2019).
[14] 渡辺　篤，松本雄太，西村祥一，清水俊也：はじめてのブロックチェーン・アプリケーション Ethereum によるスマートコントラクト開発入門，翔泳社 (2017).
[15] Bonneau, J., Jeremy C. and Steven G.: On Bitcoin as a public randomness source, *IACR Cryptology ePrint Archive* (2015).
[16] Bünz, B., Goldfeder, S. and Bonneau, J.: Proofs-of-delay and randomness beacons in Ethereum, *Proc. IEEE Security and Privacy on the Blockchain* (*IEEE S&B*) (2017).
[17] Bertoni, G., Daemen, J., Peeters, M. and Van Assche, G.: The Keccak reference, SHA-3 competition (round 3) (2011).
[18] Vigna, S.: xoshiro / xoroshiro generators and the PRNG shootout, 入手先 ⟨http://prng.di.unimi.it⟩（参照 2020-06-23）.

廣澤 龍典

2018 年立命館大学情報理工学部卒業．2020 年同大学大学院情報理工学研究科情報理工学専攻博士課程前期課程修了．在学中はビットコインやブロックチェーンに関する研究に従事．2018 年 CSEC 優秀研究賞受賞．

上原 哲太郎　（正会員）

1995 年京都大学大学院工学研究科博士後期課程研究指導認定退学．同大学院工学研究科助手，和歌山大学システム情報学センター講師，京都大学大学院工学研究科附属情報センター助教授，同大学学術情報メディアセンター准教授，総務省情報通信戦略局通信規格課標準化推進官を経て，2013 年より立命館大学情報理工学部教授．京都大学博士（工学）．デジタル・フォレンジック，システムセキュリティ等の研究に従事．共著に『基礎から学ぶデジタル・フォレンジック』（日科技連出版）等．