令和2年度「専修学校による地域産業中核的人材養成事業」
スマートコントラクトを使用したシステム開発人材の育成

スマートコントラクト開発入門
指導マニュアル

学校法人 麻生塾 麻生情報ビジネス専門学校

スマートコントラクト開発入門

1. ブロックチェーンの概要

ブロックチェーンに対して
どんなイメージがありますか?

ブロックチェーン関して持っているイメージ/知識などを生徒に質問する

- そもそもブロックチェーンという言葉を知っているか?
- ブロックチェーン関して知っていることはあるか?
- どのような種類があるか知っているか?

1.1.1

ブロックチェーン

ブロックチェーンは決して
新しい技術ではない

・ブロックチェーンが初めて実用化されたのは2009年1月
・初めてブロックチェーンが用いられたシステムはビットコイン
・大きな注目を集めたビットコインだが、「ビットコイン」、「ブロックチェーン」は決して新しい技術ではない

1.1.1

ブロックチェーン

既存の技術・学問の組み合わせ

データベース	ネットワーク	経済学
アルゴリズム	セキュリティ	暗号学

学問の総合格闘技

・ブロックチェーンは既存の様々な技術の組み合わせによる新しい仕組み
・ITだけでなく、経済学なども利用してブロックチェーン の仕組みが作られている
・このことから、ブロックチェーンは「学問の総合格闘技」とも呼ばれている（実際に、ブロックチェーンを研究している文系の研究室もある）

ブロックチェーン

データを
削除・改ざんされることなく、
半永久的に保存することができる

ブロックチェーンの概要について説明

ブロックチェーンが中立的なデータベースである部分にフォーカスして説明する

ブロックチェーンを利用すると
- データを半永久的に保存することができる
- 一度書き込まれたことは削除/編集することができない

1.1.1

ブロックチェーン

データを

削除・改ざんされることなく、

半永久的に保存することができる

↓

当たり前にできることでは?

「半永久的に保存することができる」と聞いて「現在使っているツールでデータを削除改ざんされることなく保存することは当たり前にできることなのでは?」と思った人がいると思う。

1.1.1

ブロックチェーン

Q. このファイルを100年間保存してください

8

質問を投げかける

皆さんの手元にあるファイル(何でもいいです。word, txt, html, js, etc..)を100年間改ざんや削除されることなく、100年後に自分の子孫が見ることができるようにしてください。どのような方法で保存しますか?

回答例
手元のPCで保存すると答えた場合

- そのパソコンが100年間安全に利用できることを保証することはできますか

クラウドなどを利用すると答えた場合

- グーグルやアマゾンなどのクラウドサービスが100年間利用できることを保証できますか
- 大企業によって都合の悪いことが削除されないという保証はされますか

考えてみると案外難しいことがわかる

1.1.1

ブロックチェーン

「誰か」に管理されている限り、
データが失われる可能性がある

「誰か(個人、企業に関わらず)」の手によって管理されているデータは失われる可能性がある

- 管理者が意図的に破棄する
- 管理者が管理をやめてしまう(管理者がいなくなる)
- 管理者に対する攻撃が行われる

ここで「管理者に依存しない」方法で保存するのがブロックチェーン

1.1.1

ブロックチェーン

データを「みんな」で管理する

・誰でも管理することができる
・特別な権利を持っている人はいない

データを「みんな」で管理することにより安全に保存することができる

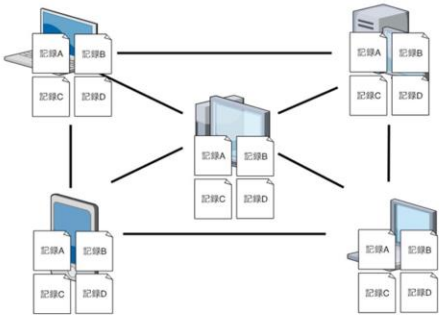
「みんな」とは?

- 不特定多数の人で管理する
- 特別な権限を持っている人(管理者)はいない

1.1.1

分散自律型台帳

同じデータをみんなで保存する



11

ネットワークを構成して、同じデータを共有する

・同じデータをネットワーク内の全てのノードで保存する
・通貨の取引履歴を共有することで通貨システムを作ったのがビットコイン

1.1.1

分散自律型台帳

Q. 悪い人が混じる可能性はないの？

・ データ削除
・ 書き換え

不特定多数の人で管理すると、悪い人が混じることはないのか?
- データの削除
- データの書き換え

などが、行われる可能性はないのか?

1.1.1

分散自律型台帳

Ａ.ある。
ただし、悪いことをしても
経済的に損するような仕組みが作られている

悪いことをするためのお金 ＞ 悪いことをして得るお金

世界中の不特定多数の人で管理を行うので、悪い振る舞いをする人は混ざってくる。しかし、ブロックチェーンでは悪い振る舞いを行う動機をなくす仕組みが内部にある。

ブロックチェーンに対して不正を行う(不正にデータを書き換えたい。消してしまいたい)ためには、コスト(お金)がかかるようになっている。このコストが不正を行って得ることのできる利益よりも大きくなるように仕組みが作られている。(不正をする動機をなくす)

1.1.1

分散自律型台帳

・ 分散→みんなで
・ 自律→管理者の存在しない
・ 台帳→記録

「データ」を安全に管理することができる

「通貨のやり取り記録」を安全に管理することができる

↓

ビットコイン

14

ブロックチェーンは「分散自律型台帳」とも呼ばれる

分散:世界中の不特定多数の人で
自律: 特別な権限を持った管理者の存在しない
台帳: 記録(通帳や、飲食店の伝票のようなイメージ)

この仕組みを利用することにより、「データ」を安全に管理することができる

「データ」を安全に管理することができる
↓
「通貨の取引記録」を安全に管理することができる(通貨の取引記録は銀行の通帳を考えるとわかりやすい)
↓
通貨システムを作ることができる(ビットコイン)

1.1.1

分散自律型台帳

「誰でも参加できる」とは？

ビットコインを例にすると…

・ 取引の審査をすることができる
・ 世界中で行われた取引を見ることができる
・ 仕様変更に意見することができる

誰でも参加できるとは?

ビットコインを例に挙げると
- 取引の審査をすることができる(銀行のような役割)
- 世界中で行われた取引を見ることができる(これまでの取引の記録を参照しないと、次に行われる取引が正当なものであるか判断できない)
- 仕様変更に意見することができる(管理者がいないため、参加者の協議で仕様の変更が行われる)

参加の方法については教材の3ページを参照

1.1.1

ブロックチェーンの種類

パブリックブロックチェーン
→誰でも参加できる
（今回扱うのはこちら）

パーミッションドブロックチェーン
→参加者に制限がある
（企業内などで運営される）

ブロックチェーンの種類

パブリックブロックチェーン

- 誰でも参加することのできるブロックチェーン(ここまで説明してきたのはパブリックブロックチェーン の特徴)
- 「ブロックチェーンとは~~~である！」のような説明はこちらのブロックチェーンを指していることが多い
- 具体的にはBitcoinや、このテキストの後半の演習で利用するEthereumなどがこれに該当する

パーミッションドブロックチェーン

- 参加者が限定されるブロックチェーン(企業や団体などが管理しているブロックチェーン)
- 国内で大手の企業が「ブロックチェーンを利用したシステムを開発」と言っている場合は、こちらのブロックチェーンを指していることが大半

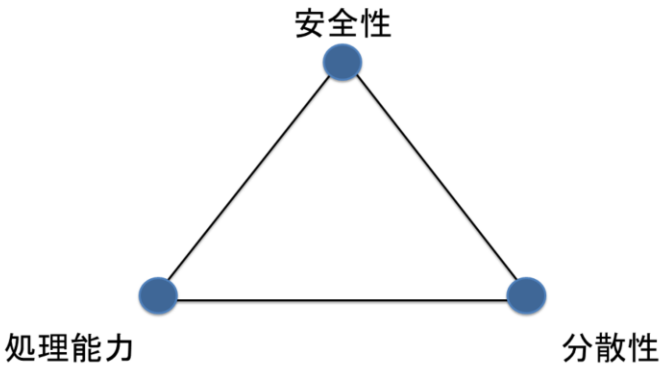
パブリックブロックチェーンとパーミッションドブロックチェーンは、参加者の形式に大きな違いがある。

今回の講義で扱うのは以下の理由のためパブリックブロックチェーン

- 初めて実用化されたブロックチェーンはビットコインであり、パブリックブロックチェーンであるため
- ブロックチェーン の特徴はパブリックブロックチェーンにより濃く出るため

1.1.1

ブロックチェーンのトリレンマ



3つの項目を同時に満たすことはできない

17

ブロックチェーン のトリレンマ

トリレンマ: 3つの項目を同時に満たすことはできない(ジレンマの3つのパターン)

ブロックチェーンでは、「安全性」「分散性」「処理能力」の3つを高いレベルで実現することはできないとされている

具体例
Bitcoin/Ethereum(パブリックブロックチェーン)

- 安全性: Bitcoin/Ethereumの安全性は極めて高いレベルで実現されている
- 分散性: Bitcoin/Ethereumはパブリックブロックチェーンであり、分散性は高いレベルで実現されている
- 処理能力: Bitcoin/Ethereumは高いとは言えない(秒間10数件程度)

1.1.1

ブロックチェーンのトリレンマ

パブリックブロックチェーンと

プライベートブロックチェーンは

分散性(参加者の数)に大きな違いがある

↓

特徴に違いが出る

パブリックブロックチェーンとプライベートブロックチェーンの特徴

この2つのブロックチェーン には分散性に大きな違いがあり(参加者に制限があるか、そうでないか)、この点が異なることで、その他の部分に関しても大きな違いが出てくる

1.1.2

非中央集権

中央集権 ⟷ 非中央集権

管理者が存在せず、
参加者のみでシステムを動かしている状態

非中央集権

- 中央集権の対義語
- 管理者が存在せずに、参加者のみでシステムを動かしている状態

中央集権の例
- 今の日本の政治の仕組み
- クライアントサーバー型のネットワーク

非中央集権の例
- クラスでの多数決(全員が完全に平等)

パブリックブロックチェーンは、これまでの技術ではできなかった、非中央集権的に不特定多数の同意を得るということができるようになった

1.1.2

高い改ざん耐性

通常のデータベースと比べて、改ざん耐性が高い

高い改ざん耐性を実現する要因

・独自のデータ構造(マイニング)
・世界中の不特定多数のコンピュータがデータを保存

高い改ざん耐性

通常のデータベースシステムと比べて、改ざん耐性が高い

高い改ざん耐性を実現する要因

- ブロックチェーン独自のデータ構造(ブロックチェーンの構成要素の説明で詳細を話す)
- 世界中の不特定多数のコンピュータがデータを保存

不特定多数とは?

- 特定の管理者(企業など)の管理下にあるコンピュータではなく、不特定多数の管理者に管理される不特定多数のコンピュータ

1.1.2

高い可用性

世界中の同等な権限を持った多くの
コンピュータで構成されている

↓

単一障害点がない

ビットコインは2009年から一度も停止していない

高い可用性

ブロックチェーンは世界中の同等な権限を持った多くのコンピュータで構成されている
→どのコンピュータが停止しても、システム全体には影響しない
→単一障害点がない

※ビットコインは2009年から一度も停止していない

1.1.2

ブロックチェーンの特徴

パブリックブロックチェーンの特徴

- 非中央集権
- 高い改ざん耐性
- 高い可用性

ブロックチェーンの特徴のまとめ

- 非中央集権: 特定の管理者が存在しない(管理者権限による特別な振る舞いなどはできない)
- 高い改ざん耐性: ブロックチェーン独自の構造と多数の複製の存在
- 高い可用性: 多数のコンピュータによって構成され、かつ単一障害点がない

最低限これらの3つの特徴は頭に入れておいて欲しい

1.2.1

ブロックチェーンの構成要素

ビットコインを例にして
ブロックチェーンの構成と処理の流れを知る

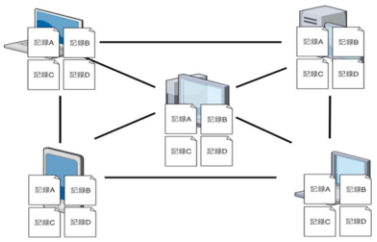
ブロックチェーンの処理の流れを構成要素とともに理解する。その際に理解の助けとしてビットコインを例に挙げる。

1.2.1

ブロックチェーンの構成要素

ブロックチェーンネットワーク

参加者によって
インターネット上に構成される
仮想的なP2Pネットワーク



24

・ブロックチェーンを構成するコンピュータによって作られるネットワークをブロックチェーンネットワークと呼ぶ
・現在多く利用されているクライアントサーバー型のネットワークではなく、インターネット上に構成される仮想的なP2Pネットワーク
・このネットワーク全体で一つのブロックチェーンが機能する

1.2.1

ブロックチェーンの構成要素

トランザクション

ブロックチェーンに要求する処理が記述されたデータ

ここからは、ビットコインでAさんがBさんに1BTC(BTCはビットコインの単位、読みはビットコイン)送ることを例にして、ブロックチェーンの処理の流れを説明する

トランザクション

- ブロックチェーン内での全ての処理の始まりはトランザクションが発行されること
- トランザクションにはブロックチェーンに要求する処理が記述されている
- ブロックチェーンの種類によってトランザクションのフォーマットは決められている

1.2.1

ブロックチェーンの構成要素

ビットコインでは

「AさんがBさんに1BTC送る」などの情報が
トランザクションに記述される

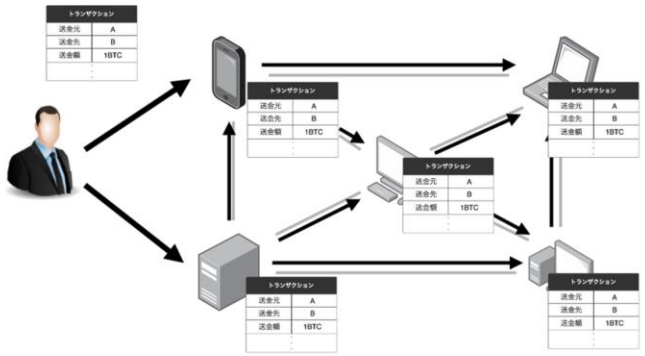
トランザクション	
送金元	A
送金先	B
送金額	1BTC
⋮	

ビットコインで送金を行う際には、送り元、宛先、総金額を記述したトランザクションを発行する。
実際には、これら以外の項目もあるが、ここでは説明を割愛する。

1.2.1

ブロックチェーンの構成要素

発行されたトランザクションは
ネットワークにブロードキャストされる



・発行したトランザクションは、ブロックチェーンネットワークにブロードキャストする。
・トランザクションはP2Pネットワーク内をバケツリレー式に伝達され、全てのノード(参加者)が同じトランザクションを保有した状態となる

1.2.1

ブロックチェーンの構成要素

トランザクションを受け取ったノードは
そのトランザクションが適正であるか検証する

・正しい額の送金であるか
・正しい署名がされているか
・正しいフォーマットで記述されているか
・データサイズは適切であるか

トランザクションを受け取ったノードは、そのトランザクションが正当なものであるか判断(不正な記述はされていないか、不正な送金ではないか)し、正しいと判断したものだけを手元に保存する

1.2.2

ブロックチェーンの構成要素

トランザクションはブロックという まとまりにまとめられる

手元に蓄えられたトランザクションは、一定時間経過すると「ブロック」という単位にまとめられる。この作業はマイニングと呼ばれる。

1.2.2

ブロックチェーンの構成要素

Proof of Work

大量を計算を行い(大量のコストを払い)、
最も早く答えを出した人(最もコストをかけた人)が
ブロックの作成権を得る方法

ブロックを作ると報酬をもらうことができる

ブロックの作成方法

不特定多数の参加者の中で、ブロックを作ることのできる人は一人(ブロックは時間の経過とともに次々に作られるので、各ブロックごとに担当者は変わる)
その一人を選ぶ方法をコンセンサスアルゴリズムと呼び、その一つにProof of Workと呼ばれるアルゴリズムがある

Proof of Work

- ブロックを作成するために、大量の計算が課される
- 計算を行い、最も早く答えを出した人(最もその計算にコストを投じた人)にブロックの作成権が与えられる
- この競争には誰でも参加することができる

Proof of Workに参加するにはコスト(お金)がかかるが、競争に勝ちブロックを作成すると報酬をもらうことができる(ビットコインでは一つブロックを作ると現在12.5BTCもらうことができる)

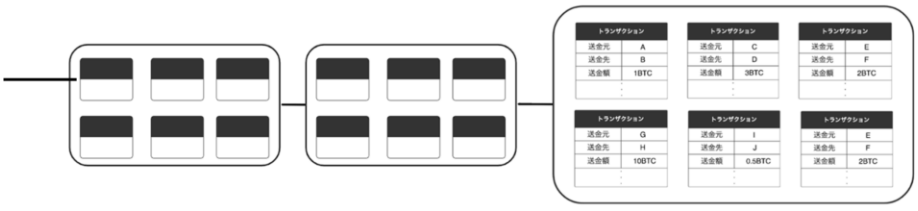
Proof of Workに参加して、ブロックを作成する行為をマイニングという。マイニングを行う人のことをマイナーと呼ぶ。

1.2.3

ブロックチェーンの構成要素

ブロックチェーン

ブロックの連なり



最新のブロック

・作成されたブロックは、数珠状につなげて保存する。(それぞれのブロックは前のブロックの識別子を持つ)
・この形式に着目してブロックチェーンという名前がつけられた。

・トランザクションはブロックに取り込まれた段階で処理が完結したものとみなされる。
・AさんからBさんへの送金もこれで完了となる。

2. スマートコントラクトの概要

2.1.1

広義のスマートコントラクト

- 賢い契約
- 契約の自動化
- プログラム化された契約
- 自動執行権のある契約 etc..

スマートコントラクトには様々な解釈がある
→人が介在しない取引

・スマートコントラクトの解釈はスライドの通り様々な解釈がある。
・大きくまとめると、スマートコントラクトは「人が介在しない取引」のことをいう。

「人が介在しない取引」というと難しく感じるかもしれないが、普段から自分たちが利用している技術

2.1.1

広義のスマートコントラクト

スマートコントラクトの例

・自動販売機
・オンライン決済
・自動改札

日常で使っている技術

身近なスマートコントラクト

- 自動販売機(人が介在しない状態で飲み物などを購入することができる)
- オンライン決済(クレジットカードなどを利用した決済。サブスクリプションなどで権利を購入するなど)
- 自動改札(電車の乗車料金の支払いの自動化)

スマートコントラクトはありふれた技術であり、決して珍しいものではない。

2.1.1

スマートコントラクトとブロックチェーン

なぜ今、
スマートコントラクトが注目されているのか?

ブロックチェーンにより
これまでできなかったことができるようになった

では、なぜスマートコントラクトが注目されているのか?

ブロックチェーンの登場により、これまでできなかった形のスマートコントラクト(決済、契約)を実現することができるようになった。方法としては、スマートコントラクトの機能の一部にブロックチェーンを利用する。

2.1.1

スマートコントラクト

従来のスマートコントラクト

サービスの提供者がサーバーを準備

→ その中で契約処理が行われる

サーバーでは取引の検証作業が行われる

→取引の記録を保存しておかなければならない

従来のスマートコントラクト

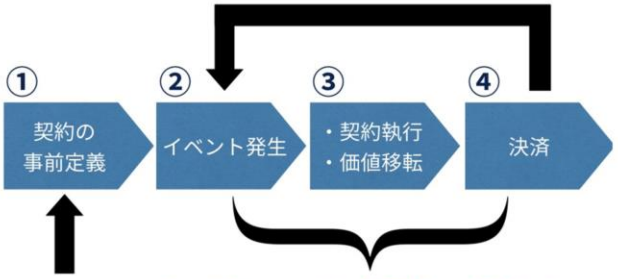
サービスの提供者がサーバーを準備し、このサーバーで契約の処理が行われる
サーバーでは、新たに行われる取引が正当なものであるかの判断をするために、これまでの取引の記録を保存しておく必要がある(銀行で送金をするためには、総金額以上の通貨を持っていることを銀行が確認する必要がある。そのためには、その人のこれまでの全ての取引記録が必要となる)

2.1.2

スマートコントラクトの仕組み

契約の自動化

スマートコントラクトの流れ



管理者が入力　**プログラムにより自動的に実行される**

https://gaiax-blockchain.com/smart-contract より引用

37

スマートコントラクトの仕組み

②から④が自動的に行われる

① 契約の事前定義：スマートコントラクトのプログラムの作成、ハードウェアが必要になる場合には準備する
自動販売機を例にすると、自動販売機本体の準備と設置

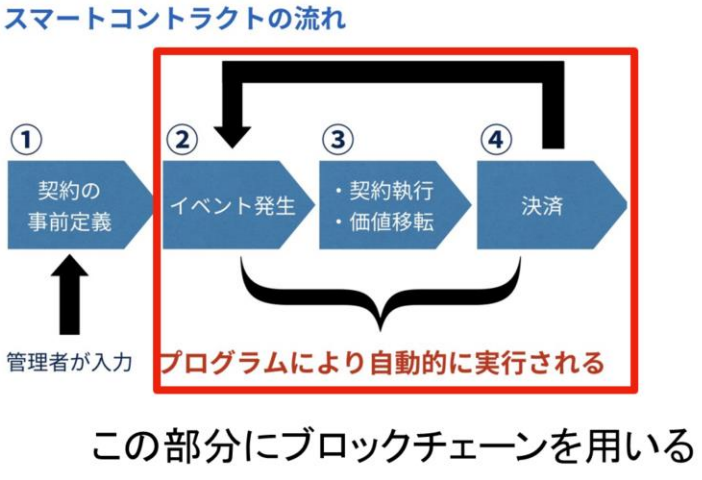
② イベントの発生: 自動販売機では、購入する人がお金を入れてボタンを押す

③ 契約執行・価値移転: 自動販売機内であらかじめ作られたプログラムにより契約行為が行われる

④ 決済: 最後に飲み物が出てきて、支払った通貨が自動販売機の中にチャリンと落ちて契約が終了する

2.1.2

スマートコントラクトとブロックチェーン



この部分にブロックチェーンを用いる

スマートコントラクトにブロックチェーンを取り込むとは?

スマートコントラクトの処理の流れの内、自動的に処理が行われる部分にブロックチェーンを利用する

2.2.1/2.2.2

スマートコントラクトの利点/課題点

スマートコントラクトの利点

・人件費がかからない
・人為的なミスの排除

スマートコントラクトの課題点

・システムの維持管理
・情報の維持管理
・仲介による手数料

スマートコントラクトの利点と課題点

利点
・人件費がかからない: 自動販売機や自動改札を考えるとわかるが、かなりの人件費が削減されていることがわかる

・人為的なミスの排除: 機械が行うことでヒューマンエラーを無くすることができる

課題点
・システムの維持管理: スマートコントラクトの仕組みを維持し続けるには維持費がかかる(サーバー代、ネットワーク代、電気代、システムの更新費用など)

・情報の維持管理: 個人情報などを扱う場合にはセキュリティなどにコストをかける必要がある

・仲介による手数料: 利用者の目線から考えると、仲介者(スマートコントラクトシステムの運営者)などに手数料を支払う必要がある

2.2.1

ブロックチェーンとスマートコントラクト

管理者不在　　耐改ざん性

透明性

ブロックチェーンを利用する代表的な3つのメリットについて

・管理者不在
・耐改ざん性
・透明性

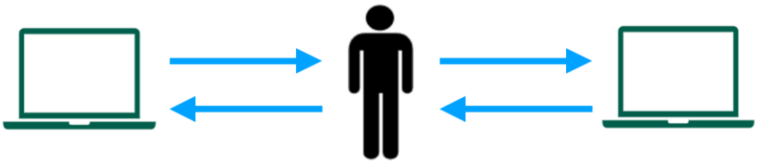
スマートコントラクトでは、ブロックチェーンの特徴を生かした取引を行うことができ、これらの特徴はブロックチェーンと一致している。

2.2.1

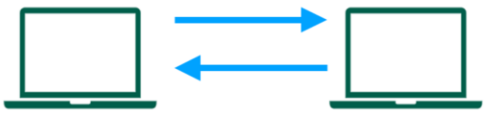
管理者不在

仲介者を介さずに安全な取引を行うことができる

従来のスマートコントラクト



ブロックチェーンを用いたスマートコントラクト



41

管理者不在

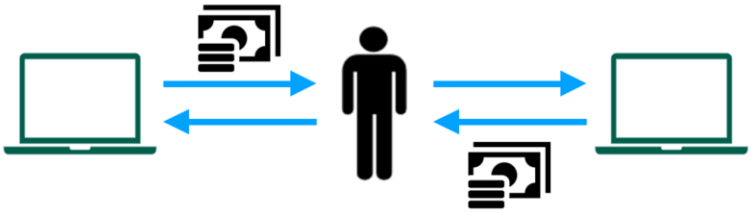
これまでのスマートコントラクトでは、管理者が必要だったが、ブロックチェーンを利用するスマートコントラクトでは管理者が必要ない

管理者が不在であることの利点
・単一障害点がなくなる
・管理者の独断でシステムが利用できなくなるなどといったことがない

2.2.1

手数料の削減

仲介手数料の発生



仲介手数料は必要ない

管理者が不在であることの利点
管理者がいないことで、仲介により発生する手数料が必要なくなる。これによって利用者はより安い手数料でシステムを利用することができる

2.2.1

高い透明性

ブロックチェーン上の記録は誰でも見ることができる
→契約内容・結果を誰でも見ることができる

秘匿性の問題

・暗号化して書き込む
・見られたくないことは書かない

高い透明性

スマートコントラクトにパブリックブロックチェーンを利用する
→取引の内容を世界中に公開する
→透明性の高い取引を行うことができる

秘匿性の問題
非公開で行いたい取引にはパブリックブロックチェーンは向いてない

対策方法
・暗号化して書き込む
・見られたくないことは書かない

ブロックチェーンに書き込んだことは決して消すことができない
→今安全とされている暗号方式で暗号化しても、それが永遠に破られないとは限らない

2.2.1

耐改ざん性

契約内容や結果はブロックチェーンに記録される
→耐改ざん性が高い

ブロックチェーンに記録される内容

- 契約内容を記述したコード
- 契約内容

耐改ざん性

契約の内容や結果をブロックチェーンに記録することで、契約に対してブロックチェーンの高い改ざん耐性を利用することができる
→契約プログラムの改ざんや、契約結果の書き換えなどを防ぐことができる

これまでは、サーバーに保存、実行していたプログラムや、データベースに保存していた契約情報などをブロックチェーンにより管理する

2.2.2

スマートコントラクトの課題点

- スケーラビリティ問題
- 鍵の管理の問題
- 手数料の問題
- 処理速度の問題

スマートコントラクトには利点だけでなく課題点もあり、これらはブロックチェーンの課題点に起因する

以下の課題点について順に説明する。

2.2.2

スマートコントラクトの課題点

スケーラビリティ問題

一秒間に処理できるトランザクション(TPS)は

・Bitcoin: 約7件
・Ethereum: 約15件

大手クレジットカード会社は数千件

スケーラビリティ問題

ブロックチェーンでは、一定時間に処理することのできる取引量が仕様の段階で決まっている
→サーバーを増やす、マシンのスペックをあげるなどで対処することができない

具体的には
・Bitcoin(初めて実用化されたブロックチェーンシステム): 7TPS(Transaction Per Second)
・Ethereum(スマートコントラクトのために作られたブロックチェーン): 15TPS

大手のクレジットカード会社は数千TPSであることから、ブロックチェーンの処理能力は極めて低いことがわかる

2.2.2

スマートコントラクトの課題点

鍵の管理

仮想通貨を利用する際には、

自身の秘密鍵が必要

一般の利用者が

秘密鍵を正しく管理することができるか?

鍵の管理

仮想通貨を利用する際(トランザクションを発行する際)には、秘密鍵での署名が必要になる
→秘密鍵を安全に管理しなければならない

一般の利用者が鍵の管理を正しく行うことができるか?
→秘密鍵を紛失すると、秘密鍵に紐づいた仮想通貨は利用できなくなる(再発行を行う仕組み、機関は存在しない)

・銀行口座のパスワード→銀行で再発行可能
・秘密鍵→再発行不可能

2.2.2

スマートコントラクトの課題点

手数料

トランザクションを発行するには

手数料がかかる

→支払いは仮想通貨

例えるなら

HTTPリクエストを出すたびに手数料がかかる

手数料

ブロックチェーン上で取引を行うためには、トランザクションを発行する必要がある
→トランザクションを発行するためには手数料が必要になる(トランザクション手数料)
→支払いは利用しているうロックチェーンの内部通貨(仮想通貨)

仮想通貨を持っていないとブロックチェーンを用いたサービスを利用することができない
→仮想通貨の保有率が極めて低い(この点がボトルネックとなる)

仮想通貨を保有することなく、サービスを利用できる仕組みは作られつつあるが、まだ普及段階

2.2.2

スマートコントラクトの課題点

処理速度

処理速度が極めて遅い

即時性の求められる処理には向いていない

処理速度

ブロックチェーンは、取引が行われてからの時間の経過とともに、取引の安全性が向上する(取引が書き換えられる、取り消される確率が低下する)
つまり、安全に取引を行うためには時間を必要となるため、処理速度が極めて遅い
このような特徴から、即時性の求められる処理には向いていない

2.2.2

スマートコントラクトの課題点

コントラクトの修正

ブロックチェーンは一度書き込んだデータの

書き換え/削除を行うことができない

一度ブロックチェーンに書き込んだプログラムは修正することができない

コントラクトの修正

ブロックチェーンでは一度書き込んだデータを書き換えることや消すことはできない(このような特徴から石板に刻むと例えられることがある)
→一度、ブロックチェーンに書き込んだプログラムは修正することができない

利点
一度正しいプログラムを書き込めば、誰にも書き換えられることがない

難点
プログラムにバグがあったとしても、書き換えることや修正することができない

このような特徴から過去に多額の通貨が盗まれるといったことも発生している

2.3.1

スマートコントラクトに利用される ブロックチェーン

Q .全てのブロックチェーンで
スマートコントラクトは実現できるか?

BitcoinやEthereumを始めとして無数のブロックチェーンが存在する。
これらを始めとして全てのブロックチェーンでスマートコントラクトを作ることができるのか?

2.3.1

スマートコントラクトに利用される ブロックチェーン

A.スマートコントラクトのために作られた ブロックチェーンが必要

スマートコントラクトのために作られたブロックチェーン が必要

ここからは具体的にスマートコントラクトに利用するブロックチェーンを紹介する

2.3.1

スマートコントラクトに利用される ブロックチェーン

Ethereum

スマートコントラクトを
実現するために生まれた
初めてのブロックチェーン

現在、スマートコントラクトを作成する際の
最もメジャーなパブリックブロックチェーン

Ethereum

スマートコントラクトを実現するために作られた初めてのブロックチェーン

今回の教材では、スマートコントラクトの作成にEthereumを利用する。詳細については3章で説明する

2.3.2/2.3.3

スマートコントラクトに利用される ブロックチェーン

NEO

中国で開発されたブロックチェーン
様々な言語で開発を行うことができる

EOS

トランザクションの処理が高速
完全なパブリックブロックチェーンではない

NEO

・中国で開発されたパブリックブロックチェーン
・スマートコントラクトの開発をPython、Javaなど様々な言語で行うことができる

EOS

・トランザクションの処理が高速である(利用されているDPoSというコンセンサスアルゴリズムに特徴がある)
・完全なパブリックブロックチェーンではない

3. Ethereum

3.1.1

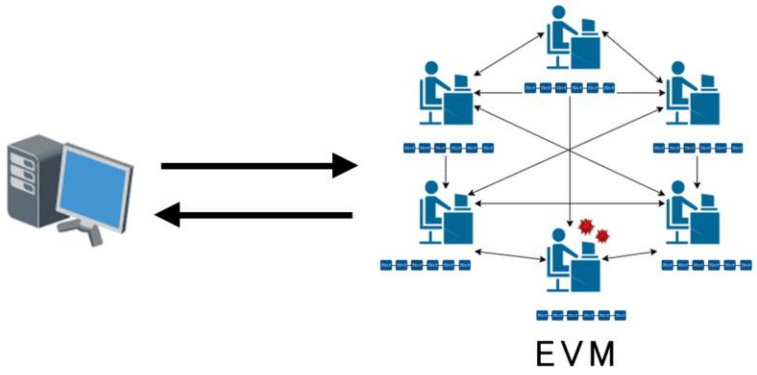
Ethereum

・ パブリックブロックチェーン
・ 管理者のいないシステム上で様々なサービスを実現するためのブロックチェーン

Ethereumは
・パブリックブロックチェーン
・スマートコントラクトのためのブロックチェーン（管理者のいないシステム上で様々なサービスを実現するためのブロックチェーン ）
・当時大学生だったヴィタリック・ブテリンにより作られた

3.1.2

EVM(Ethereum Virtual Machine)



Ethereumはパブリックブロックチェーンの一つであり、ネットワーク全体で一つの仮想的なコンピュータのように振る舞う。このコンピュータのことをEVM(Ethereum Virtual Machine)と呼ぶ。

3.1.2

EVM(Ethereum Virtual Machine)

- 世界中の多数のノードにより構成される
- 一般的な処理を行うことができる
- 状態は一意に保たれる

EVMは
・世界中の多数のノードにより構成される
→ Ethereumはパブリックブロックチェーンであるため、世界中にEVMを構成するノードが存在する

・一般的な処理を行うことができる
→　一般的なプログラムを実行することができる(Bitcoinなどでは繰り返し処理などを行うことができない)

・状態は一意に保たれる
→　EVM内のどのノードが全く同じデータを持っている(変数aの値をどのノードに聞いても必ず同じ答えが返ってくる)

3.1.2

EthereumにおけるContract

辞書では

Contract → 契約, 契約書

Ethereumでは

Ethereum上で動くプログラムは全てContractと呼ばれる （契約を行うプログラムだけではない）

コントラクト(Contract)という言葉は辞書的には、「契約、契約書」という意味

Ethereumにおいては、Ethereum上で動くプログラムは全てコントラクトと呼ばれる

・整数を与えると、その2倍の数を返すプログラム
・変数に文字列を登録するだけのプログラム

これらもEthereumでは「コントラクト」と呼ばれる

3.1.3

BitcoinとEthereum

それぞれのブロックチェーンには目的・特徴がある

Ethereum

- アプリケーションのプラットフォーム
- World Computer

Bitcoin

- 価値(通貨)の移転に特化
- Digital Gold

60

ブロックチェーンには無数の種類があり、それぞれには作られた目的や特徴がある

Ethereum
・スマートコントラクト(ブロックチェーン 上のアプリケーション)を動かすためのプラットフォーム
・「WorldComputer」とも呼ばれる

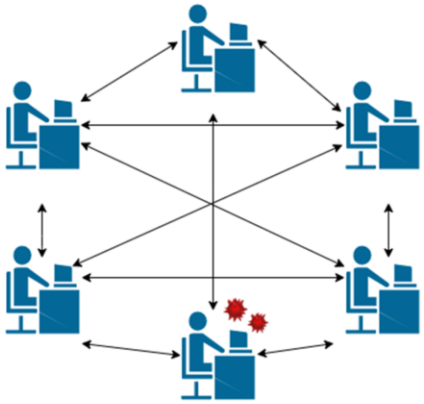
Bitcoin
・価値(通貨)の移転に特化したブロックチェーン
・「Digital Gold」とも呼ばれる

Ethereumの構成要素

3.2.1

Ethereumの構成

Ethereumのネットワーク



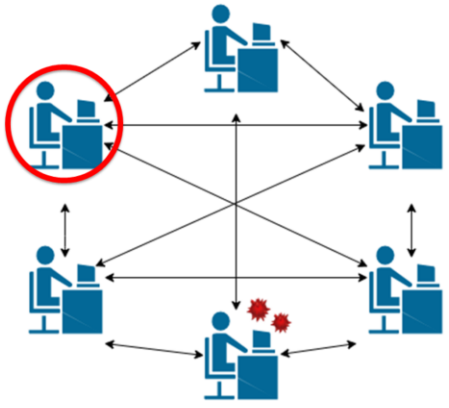
62

Ethereumはパブロックブロックチェーンの一つであり、参加者によりネットワークが構成されている

3.2.1

Ethereumの構成

Ethereumのネットワーク



63

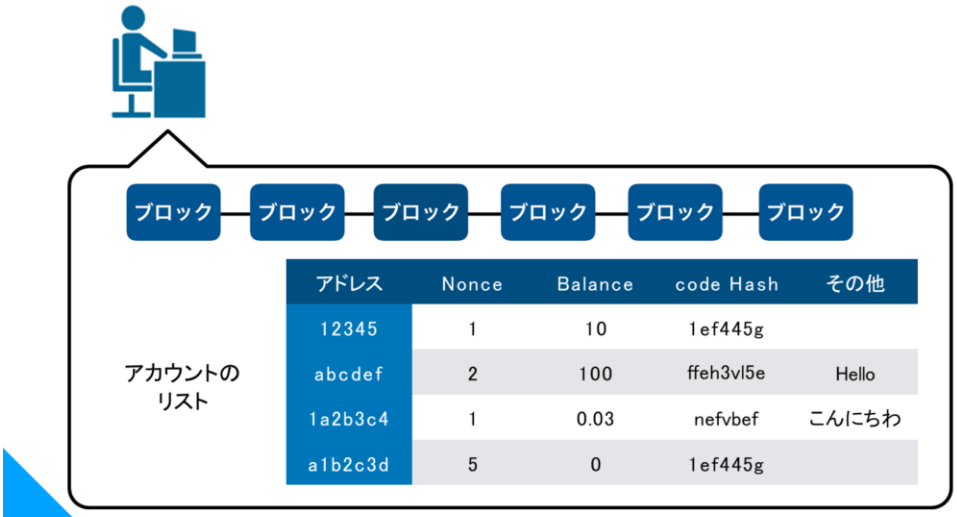
ネットワーク内の一つのノードに注目する

Ethereumのネットワークにノードの一つとして参加するためには、専用のソフトウェアが必要(インターネットで配布されている)
・その中でも最もシェアが高く有名なのがGeth(Go Ethereum)

参加のためのソフトウェアはEthereumの仕様に則っていれば、自作したものでも良い

3.2.2

ノードについて



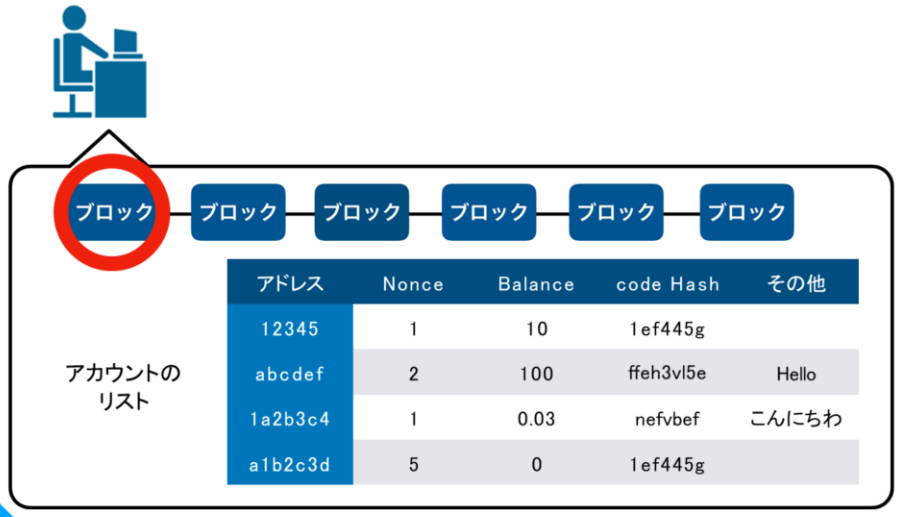
64

ノード

それぞれのノードが保有している情報
・ブロックチェーン(ブロックの連なり)
・アカウントリスト(どのアカウントがどれだけの通貨を保有しているのか?などといった情報が記録されたもの)

3.2.2

ノードについて



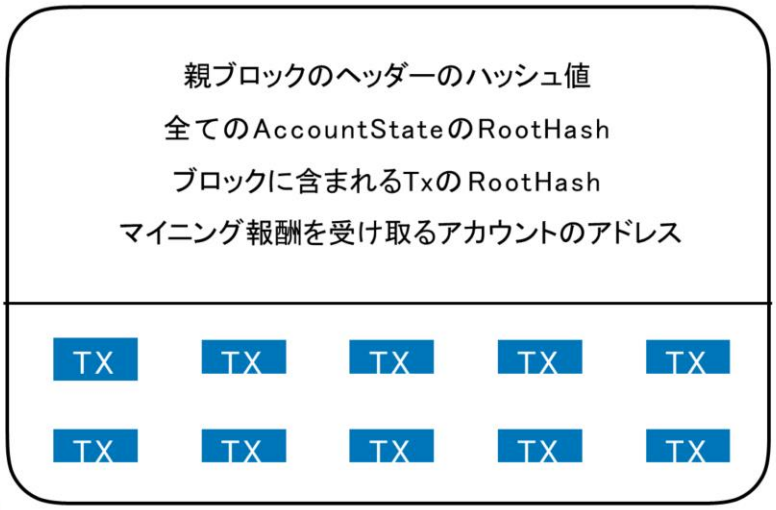
65

ノード

ノードが保有している情報の内、ブロックチェーンに注目してさらに細かく中身を確認していく

3.2.3

ブロックの構成



66

ブロック

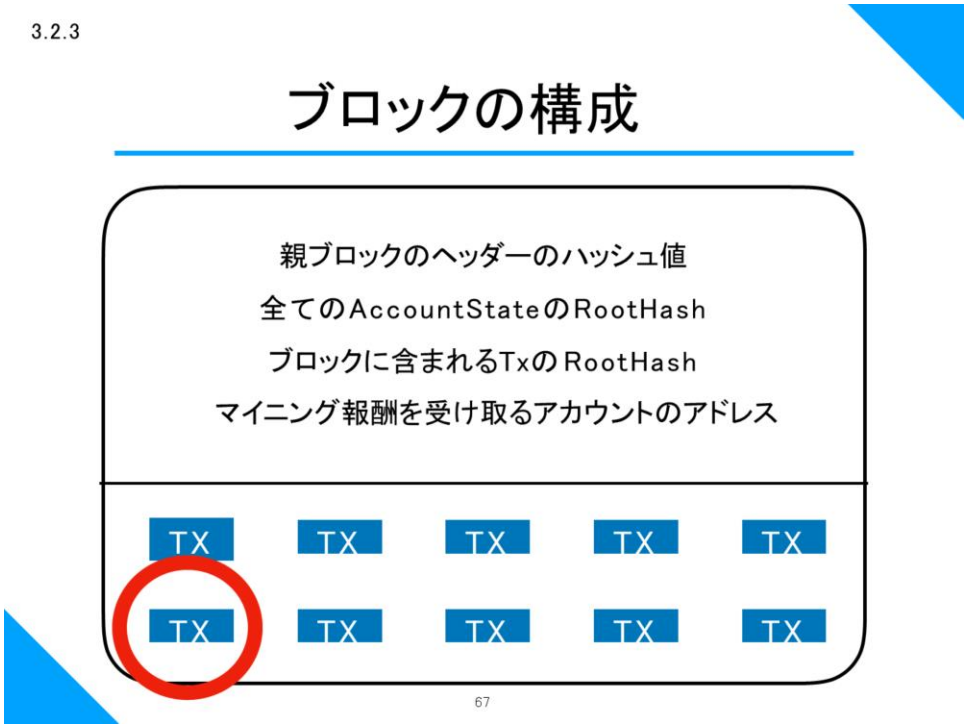
Ethereumのブロックの構成

大きく2つの部分により構成される
・ブロックヘッダー(スライドのブロックの上半分)
・トランザクションリスト(スライドのブロックの下半分)

厳密には3つ目(Ommerブロックのリスト)があるが今回は触れない

・ヘッダーには、そのブロックの固有の情報が記録されている
・トランザクションリストには、そのブロックにまとめられたトランザクションが記録されている

3.2.3



ブロック

ブロックに含まれる情報の内、トランザクションに注目してさらに細かく中身を確認していく

3.2.4

トランザクションの種類

Contract Creation

コントラクトアカウントを作成するためのトランザクション

Message Call

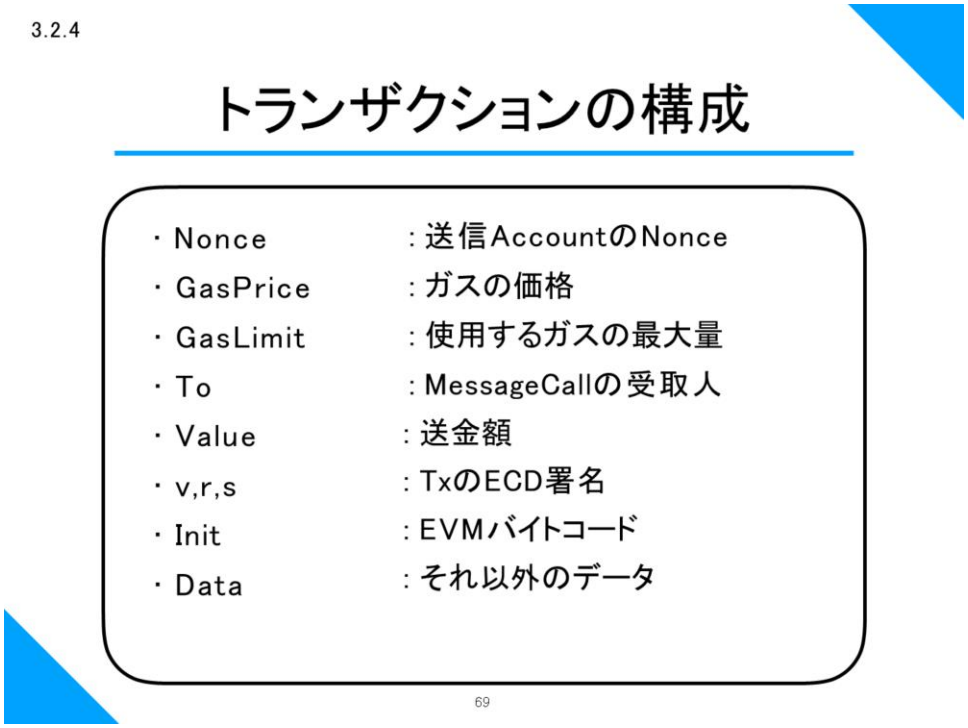
送金やコントラクトの実行のための
トランザクション

トランザクションの種類

Contract Creation
・作成したコントラクトをブロックチェーンに登録するためのトランザクション(コントラクトアカウントを作成するためのトランザクション)

Message Call
・Ethereum上で発行されるContract Creation以外のトランザクション。送金やコントラクトの実行などのためのトランザクション

3.2.4



トランザクション

スライドのような項目からなる。重要な項目のみ説明する。

・Nonce: トランザクションの発行者がこれまでに発行したトランザクションの数。この項目により順序付けてトランザクションを実行することができる

・GasPrice、GasLimit: 次のスライドで説明

・To: 送金のためのトランザクションであれば送金先、コントラクトを実行するためのトランザクションであれば、実行したいトランザクションの識別子を記入

・Value: 総金額

3.2.4

トランザクションの構成

Gas(ガス)

トランザクションを発行するための手数料

実行する処理の複雑さ、書き込みの量により
必要なGasの量が定められる

Gas(ガス)

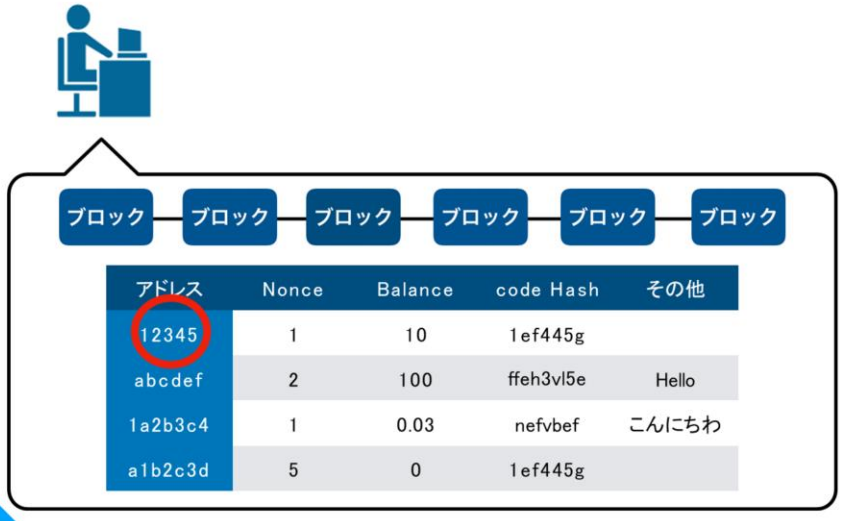
GasはEthereumでトランザクションを発行するための手数料

・実行する処理の複雑さ(複雑であればあるほど、手数料は増える)
・ブロックチェーンに対する書き込みの量(データ量が多いほど、手数料は増える)

Gasの必要性
・DoS攻撃を防ぐ: 不要なトランザクションを発行させない
・ストレージの増加を抑える: 最低限の書き込みを行う
・処理量の抑える: 複雑なトランザクションが大量に発行されると、ネットワークに遅延が発生する

3.2.5

Ethereumの構成



アドレス	Nonce	Balance	code Hash	その他
12345	1	10	1ef445g	
abcdef	2	100	ffeh3vl5e	Hello
1a2b3c4	1	0.03	nefvbef	こんにちわ
a1b2c3d	5	0	1ef445g	

ノードが持っている情報である、アカウントリストについて

3.2.5

アカウント

通貨の管理を行うための単位

全てのアカウントはアドレスと呼ばれる
固有の識別子を持つ

アカウント

・アドレス: 全てのアカウントをユニークに管理するための識別子(Hash関数により作られる)
・銀行の口座番号のようなイメージ

3.2.5

アカウント

Accountの種類

- 外部アカウント
- コントラクトアカウント

アカウントの種類

外部アカウント: 人間が秘密鍵によって管理するアカウント(一般的な銀行口座のようなイメージ)
コントラクトアカウント: Ethereum上のコントラクトが持つアカウント(それぞれのコントラクトにも人間と同様にアカウントが与えられる)

人が管理するアカウントも、コントラクト(Ethereum上のプログラム)が持つアカウントもEthereum上では同等に扱われる

3.2.5

アカウントの構成

・Address　　:アカウントの識別子

・Nonce　　:発行したトランザクション数

・Balance　　:保持する通貨の量

・CodeHash　　: コントラクトコードのハッシュ値

・StorageRoot　: 上記以外のパラメータをまとめたハッシュ値

アカウントの構成要素

・Address: 識別子
・Nonce: このアカウントがこれまでに発行したトランザクションの数(トランザクションや実行するプログラムに順序付けをすることができる)
・Balance: 保有する通貨の量

3.2.6

Ethereumにおける状態

Account State
アカウントが保持する
パラメータを考慮した状態

World State
全てのAccount Stateが含まれた
Ethereum全体の状態

・トランザクションにより”Account State”が遷移する
・”Account State”の遷移に伴い”World State”が遷移する

Etheruemについて学ぶ際には、「状態」や「state」といった言葉がよく出てくる
Ethereumにおける状態とは、Ethereumをコンピュータとして考えた時の、変数の値であると考えるとわかりやすい。

Ethereumにおける状態には2種類ある

・Account State: それぞれのアカウントが持つ変数をまとめたもの(木構造を利用)
・World State: Ethereum内の全てのアカウントの状態をまとめたもの(木構造を利用)

3.3.1

Ethereumの処理の流れ

1. トランザクションが発行される
2. マイナーがブロックを作る
3. ブロックチェーンが形成される

Ethereumの処理の流れ

大きな流れは、
・トランザクションが発行される
・マイナーがブロックを作る
・ブロックチェーン が形成される

Ethereumはパブリックブロックチェーンの一つであり、基本的な処理の流れは既に説明したブロックチェーンの処理の流れと一致する。

3.3.1

Ethereumの処理の流れ

ブロックチェーンに
コントラクトを載せる

Ethereumにコントラクトを載せる

Ethereum上でコントラクトを実行するためには、事前に作成したコントラクトを
Etheruem上に載せておく必要がある

3.3.1

コントラクトを作成する

コントラクトコードを記述し、
コンパイルする

Ethereumにコントラクトを載せる

・コードを作成する(言語はSolidityという専用の言語。5章で扱う)
・コンパイルする

コードの作成とコンパイルに関しては、5章の演習時にツールの紹介を行う。

3.3.1

トランザクションを発行する

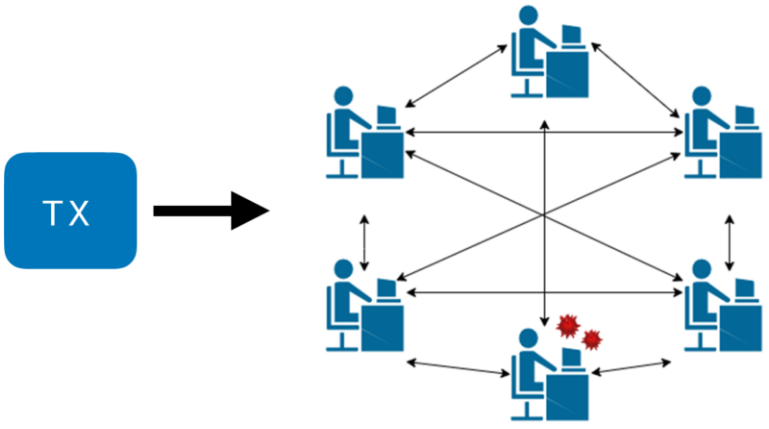
Init: ＥＶＭバイトコード

Ethereumにコントラクトを載せる

トランザクション内の「Init」という項目に、EVMバイトコード(コンパイルしたコード)を記述し、トランザクションを発行する。

3.3.1

ネットワークに伝達する



Ethereumにコントラクトを載せる

トランザクションをEthereumネットワークにブロードキャストする

3.3.1

マイナーがTxを処理する

・コントラクトアカウントを作成する
・WorldStateを遷移させたブロックを作る

Ethereumにコントラクトを載せる

トランザクションを受けとったマイナーが、トランザクション内のEVMバイトコードを元に、コントラクト用のアカウントであるコントラクトアカウントを作成する

アカウントが新規に作成されると(この時、今回発行されたトランザクションだけでなく、世界中で発行されたトンラククションが同時並行で処理される)、Ethereum内の全てのアカウントの状態をまとめたWorldStateが変化する。その変化を踏まえてマイナーがブロックを作る

3.3.1

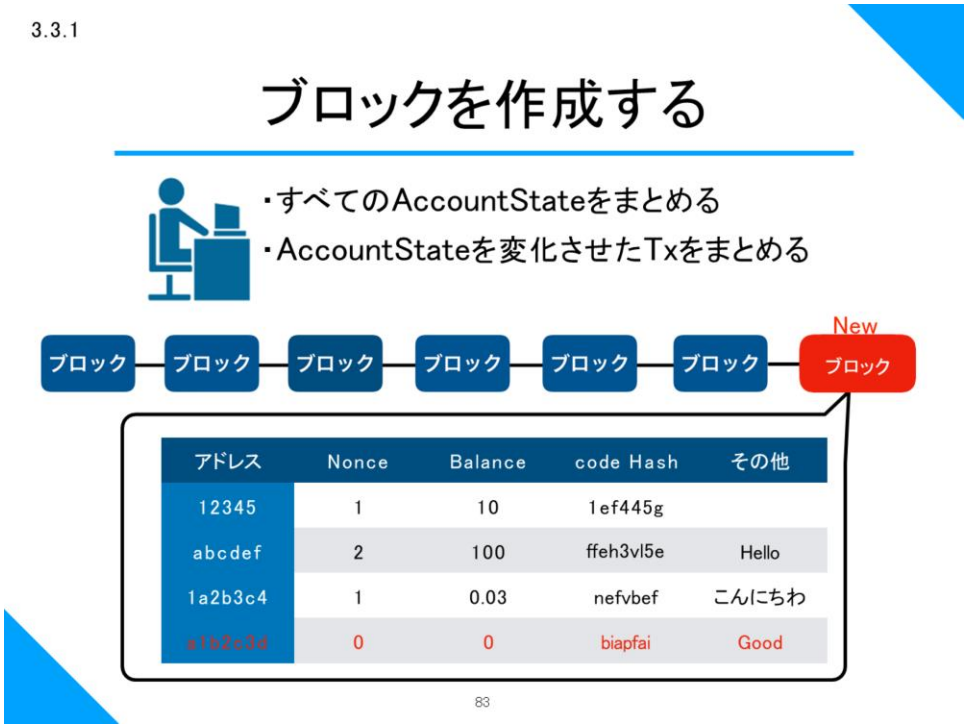
マイナーがTxを処理する

アドレス	Nonce	Balance	code Hash	その他
12345	1	10	1ef445g	
abcdef	2	100	ffeh3vl5e	Hello
1a2b3c4	1	0.03	nefvbef	こんにちわ
a1b2c3d	0	0	biapfai	Good

Ethereumにコントラクトを載せる

新たにアカウントが作成されると、その情報を手元のアカウントリストに反映する
（赤字で示されているのが、新たに作成されたアカウント）

3.3.1



Ethereumにコントラクトを載せる

アカウントの変更を踏まえて、新たなブロックを作成する

発行したトランザクションが新たなブロックに取り込まれた段階で、このトランザクションに対する処理は終了し、作成したコントラクトは実行可能な状態となる

3.3.2

Ethereumの処理の流れ

ブロックチェーン上の
コントラクトを実行する

コントラクトを実行する

既にブロックチェーンに登録されたコントラクトを実行する

3.3.2

トランザクションを発行する

・To　　: 実行するコントラクトのアドレス

・Data　: コントラクトに渡す情報

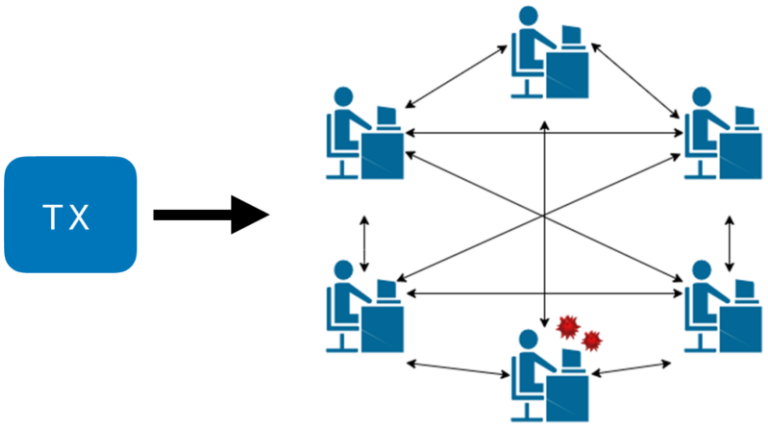
コントラクトを実行する

トランザクションを発行する。トランザクションの以下の項目を記入する

・To: どのコントラクトを実行したいか、コントラクトアカウントのアドレスを指定する
・Data: コントラクト内のどの関数を実行するか? その関数にどのような変数を渡すか?などといった情報を記入する

3.3.2

ネットワークに伝達する



コントラクトを実行する

トランザクションをEthereumネットワークにブロードキャストする

3.3.2

マイナーがTxを処理する

・実行するコードブロックチェーン上から見つける
・コードを実行し、AccountStateを遷移させる
・WorldStateを遷移させたブロックを作る

コントラクトを実行する

マイナーが
・トランザクションに記述されたコントラクトアカウントのアドレスから、該当のコントラクトを見つける(次のスライドに図示)

・コントラクト内のコードを実行し、コントラクトアカウントが持つパラメータを遷移させる(2枚先のスライドに図示)
・それを踏まえてブロックを作る(3枚先のスライドに図示)

3.3.2

マイナーがTxを処理する

コードを探す

発見！

アドレス	Nonce	Balance	code Hash	その他
12345	1	10	1ef445g	
abcdef	2	100	ffeh3vl5e	Hello
1a2b3c4	1	0.03	nefvbef	こんにちわ
a1b2c3d	5	0	1ef445g	

88

コントラクトを実行する

・トランザクションに記述されたコントラクトアカウントのアドレスから、該当のコントラクトを見つける

3.3.2

マイナーがTxを処理する

コードを実行する

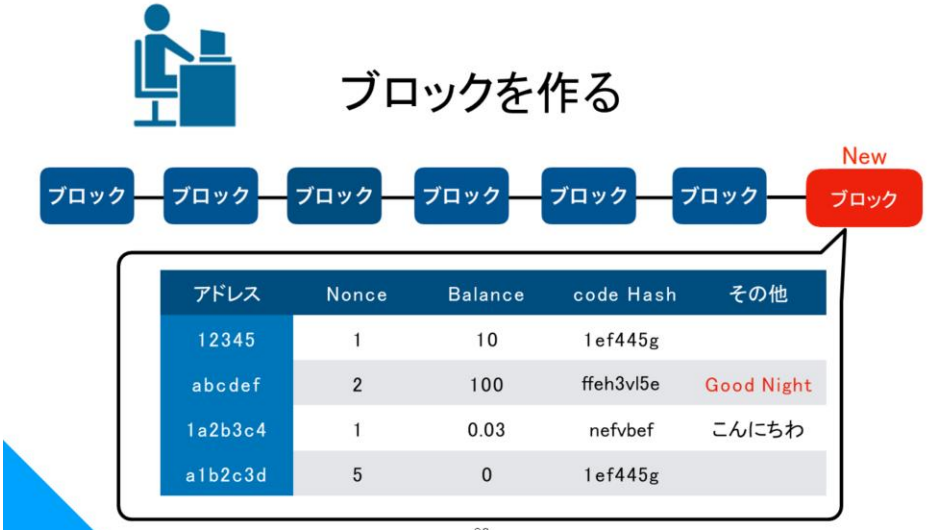
アドレス	Nonce	Balance	code Hash	その他
12345	1	10	1ef445g	
abcdef	2	100	ffeh3vl5e	Good Night
1a2b3c4	1	0.03	nefvbef	こんにちわ
a1b2c3d	5	0	1ef445g	

コントラクトを実行する

・コントラクト内のコードを実行し、コントラクトアカウントが持つパラメータを遷移させる

3.3.2

ブロックを作成する



90

コントラクトを実行する

・アカウントの状態を踏まえてブロックを作る

4.DApps(分散型アプリケーション)

4.1.1

DAppsとは？

4.1.1

Decentralized Applications

（分散型アプリケーション）

ブロックチェーンを使った
アプリケーション

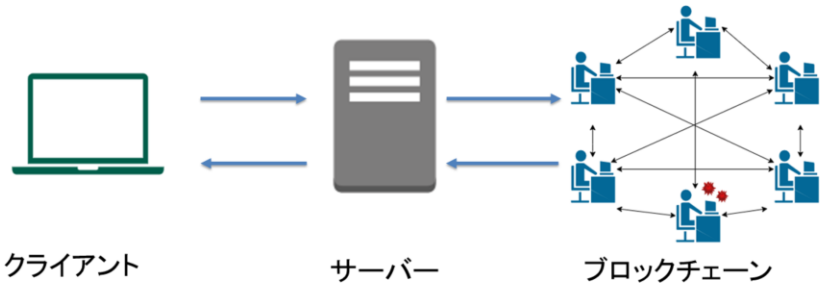
・DAppsはDecentralized Applicationsの略(Decentralizedは非中央集権の意味)
・日本語では、分散型アプリケーションと略されることが多い

大きくまとめるとDAppsとはブロックチェーンを利用したアプリケーションのこと

4.1.2

DAppsの構成

従来のアプリケーションのサーバーサイドを部分的に
ブロックチェーンに置き換える



従来のシステムのサーバーサイドで行われる処理の「一部」をブロックチェーンに置き換える

4.1.2

DAppsの構成

処理の全てをブロックチェーンで置き換えることはできない

・データ量の制約
・手数料の存在

従来のシステムのサーバーサイドで行われる処理の「一部」をブロックチェーンに置き換える
→処理の全てをブロックチェーンで置き換えることはできない

・データ容量の問題
トランザクションに書き込むことのできるデータ容量には制約がある。画像や動画などのデータは基本的にはブロックチェーンに保存することはできない

・手数料の問題
大量のデータの書き込み、複雑な処理の要求をすると求められる手数料が高額となり、利用者の負担となる

4.2.1

DAppsの利点

決済の仕組みを簡単に作ることができる

仮想通貨と密接な関係を持つ
ブロックチェーンを利用することで、
簡単に決済の仕組みをシステムに組み込むことができる

決済の仕組みを簡単に作ることができる

ブロックチェーンは内部に仮想通貨を持つものが多く、ブロックチェーンを利用することで決済の仕組みを簡単に実装することができる

4.2.1

DAppsの利点

情報の公共化

パブリックブロックチェーンを用いることで、
特定の企業のみが情報を管理するのではなく、
公共のデータとして情報を管理することができる

情報の公共化

これまでのサービスの大半は、情報を特定の企業のみが管理してきた
→その企業がなくなると、蓄積してきた情報は失われてしまう

パブリックブロックチェーンにデータを記録することで、公共の場でデータを管理することができる
→特定の企業の存亡にデータが依存しない

4.2.2

DAppsの課題点

中央集権的

ブロックチェーンだけを見ると非中央集権的だが、
多くのDAppsではこの特徴を活かし切れていない

大半のDAppsではブロックチェーン以外にサーバーを必要とすることが大半であるため

中央集権

これまでのサービスの大半は、情報を特定の企業のみが管理してきた
→その企業がなくなると、蓄積してきた情報は失われてしまう

パブリックブロックチェーンにデータを記録することで、公共の場でデータを管理することができる
→特定の企業の存亡にデータが依存しない

4.2.2

DAppsの課題点

可用性

大半のDAppsではブロックチェーン以外に
サーバーを必要とすることが大半
→この点が単一障害点となる

可用性

大半のDAppsではサーバーを必要とする→Webサーバーなど

システム全体で一つでも管理者の存在する部分があれば、その点が単一障害点となる

ブロックチェーンの可用性を活かし切れていない

4.3.1

DAppsの事例

ゲーム

CryptoKitties

- 仮想的な猫を育成するゲーム
- 育成した猫を仮想通貨で売買することができる

クリプ豚

- 仮想的な豚を育成するゲーム
- 日本で作られた初めてのブロックチェーンゲーム

ゲーム

・ブロックチェーンを用いたゲームが多く存在する。
・それらのゲームの大半は、育成ゲームのようなものが多く、育成したキャラクターを仮想通貨で売買することができる

4.3.2

DAppsの事例

分散型取引所

仮想通貨取引所

仮想通貨と法定通貨、異なる種類の仮想通貨同士を
交換するためのシステム

取引所の多くは、企業により運営されている

分散型取引所(Decentralized Exchange)

仮想通貨と法定通貨、仮想通貨動詞の交換をするためのサービス(一般に仮想通貨を購入するためには利用することが多い)

仮想通貨取引所の大半は企業により運営されている

難点
・ユーザーの秘密鍵が一か所にまとまるため、攻撃の対象になりやすい
・これまでに数々の流出事件があった

4.3.2

DAppsの事例

ブロックチェーン上で管理者の存在しない取引所

→分散型取引所

Ether Delta

- Ethereum上に構築される分散型取引所
- Ethereum上の通貨のみ取引することができる

分散型取引所(Decentralized Exchange)

・ブロックチェーン上で管理者不在の仮想通貨取引所

Ether Delta
・Ethereum上に構築される分散型取引所
・Ethereum上の通貨のみ取引することができる(EthereumではEthereum上の独自の通貨を発行することができる)

5. コントラクトの作成

5.1.1

コントラクトの開発言語

Solidity

Ethereum上で
コントラクトを作成するための専用の言語

Solidity

・Ehtereum上でコントラクトを作成するために作られたオブジェクト指向言語
・C++やJavascriptの影響を受けて作成されている

5.1.1

コントラクトの作成環境

Remix

Ethereum上で
コントラクトを作成するための統合開発環境

Remix

Ethereum上でコントラクトを作成するための統合開発環境
・コードの作成
・コンパイル
・ブロックチェーンへのデプロイ

手元に環境を構築しなくとも、ブラウザから利用することができる(今回の講義ではRemixのブラウザ版を利用する)

5.1.1

Remixでコントラクトを作成する

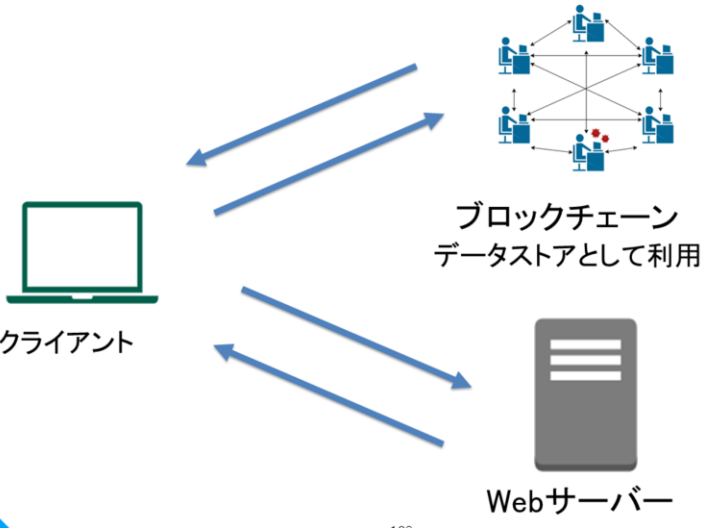
6. 演習①

6.1.1

ブロックチェーンを利用した Webアプリケーションを作る

6.1.1

作成するアプリケーションの構成



Webサーバー: HTML, CSS, Javascriptファイルを保持する
ブロックチェーン(Ethereum): データストアとして機能させる(DBの代替)

6.1.1

Go Ethereum(Geth)

・ Ethereumのクライアントソフト
・ 参加者として必要な振る舞いを全てすることができる

ローカル環境でEthereumと同じ動作環境を
作ることができる

Go Ethereum

Etheruemのクライアントソフト(インターネット上で配布されている)

・Ethereumの参加者として必要な要件は全て満たしている
・オープンソース

既存のEthereumネットワークに参加するだけでなく、新たにネットワークを作ることもできる。

コントラクトのデプロイの前にGethで動作を確認することで、本番環境と同じ状態でテストをすることができる

6.1.1

プライベートネットワークの作成

手元に自身のマシン1台だけの
Ethereumのプライベートネットワークを作成
する

Go Ethereumでは実際のEthereumのネットワークに接続するだけでなく、自身で新たにEthereumのネットワークを作成することができる

6.4.1

MetaMask

Google Chromeのプラグインとして利用できる
Ethereumウォレット

ウォレット
→仮想通貨を管理、送金なども行うアプリ

Metamask

MetamaskはGoogle Chromeのプラグインとして利用できるEthereumウォレット

ウォレットとは仮想通貨を管理、送金なども行うアプリ(仮想通貨を扱うためには必須)

Metamaskは単純なウォレット機能だけでなく、ブラウザ上のJavaScriptにブロックチェーンにアクセスするための機能も提供してくれる

7. 演習②

7.1.1

フレームワークを利用してDAppsを作る

作成するシステムについては6章と同様

DApps作成のフレームワーク

Truffle

Ethereum上でのスマートコントラクトを
開発するためのフレームワーク

Truffle

・Ethereumを利用したスマートコントラクトを作るためのWebアプリケーションのフレームワーク
・コントラクトの作成から、フロントエンドの作成、Ethereumネットワークへの接続の設定までおこなうことができる

7.3.1

Ethereumのネットワークの種類

メインネットワーク
価値をもった通貨が取引されるネットワーク

テストネットワーク
価値を持たない通貨が取引されるテスト用ネットワーク

プライベートネットワーク
個人が構築するネットワーク

Ethereumには大きく分けて3種類のネットワークが存在する

メインネットワーク
価値を持った通貨が取引されるネットワーク(本当のEthereumネットワーク、本番環境)

テストネットワーク
・メインネットワークと同様に世界中に広がるネットワーク
・テストネットワーク上の通貨は他の通貨と交換できない(ネットワーク外では価値を持たない)

テストネットワークの種類
・Ropsten: 最も最初に作成されたテストネットワーク。メインネットワークのEhtereumと振る舞いが同じ
・Kovan: コンセンサスアルゴリズムを本番環境とは変え、よりテストに向けたネットワーク

・プライベートネットワーク
Etherumのクライアントソフトを利用して、誰しもが作ることのできるネットワーク

演習1. オリジナルの仮想通貨を発行する

7.3.1

ERCトークン

ERC (Ethereum Request for Comment)

ERC20 Token Standard

Ethereum上で発行するトークンの規格

ERC (Ethereum Request for Comment)

Ethereumのアプリケーションレベルの標準化や企画が定義されている(コントラクトを作る際に模範となるコードがGithubに公開されている)
これに準拠することで安全なコントラクトを作成することができる

ERC20 : Ethereum上にオリジナルのトークン(仮想通貨)を発行するためのコードがまとめられている。また、統一されたインターフェイスを持つため、仮想通貨取引所で扱ってもらいやすい

スマートコントラクト開発実践

令和2年度「専修学校による地域産業中核的人材養成事業」
スマートコントラクトを使用したシステム開発人材の育成

スマートコントラクト開発入門指導マニュアル

令和3年2月

学校法人 麻生塾　麻生情報ビジネス専門学校
〒812-0016　福岡県福岡市博多区博多駅南２丁目１２-３２