Bitcoinの基本と体験

@ DG Lab - Anditto Heristyo

Agenda

1. 目標
2. BC2のBitcoin
3. Bitcoin-cliの体験
4. Bitcoinの概要

このセッションの目標

目標

1. Bitcoin-cliの体験
2. Bitcoin全般の話

全般的な話ですので、今は全部わからなくても大丈夫です。

事前チェック

- ソースコードはダウンロードしましたか？
- ちゃんとコンパイル出来ましたか？
- まだ問題がある方は、はやめに声をかけてください。

完了しなかったら続けないよ。。。

本当に困っている人はスタッフに声をかけてください。

BC2のBitcoinバージョン

ワークショップで使うBitcoin

https://www.github.com/dgarage/bc2

- Bitcoin Coreのフォークです。
- 中身はほぼ同じですが。。。

BC²バージョン

BitcoinのP2Pネットワーク → プライベートネットワーク

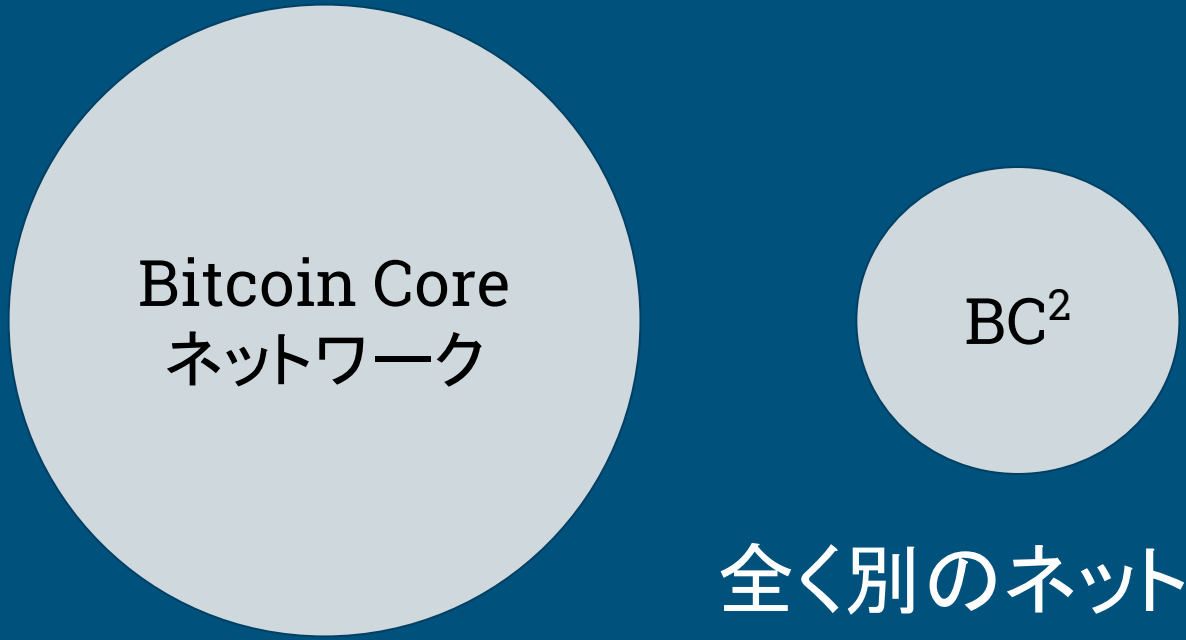
マジックナンバーが異なる

マイニングは難しい → 簡単に設定した

（10分ごとに生成する） → 1分にする

アドレスの頭は1または3 → BまたはCになる

BC^2とBitcoin Core



全く別のネットワーク

注意点

操作は本物のbitcoindおよび
bitcoin-cliと全く同じです。

両方持っている人は間違わないように、
ご注意をお願いします。

早速始めましょう！

まず、ターミナルを開いて、bitcoind を実行します。

```
$ cd ~/bc2/src

$ ./bitcoind -printtoconsole
```

あるいは:

```
$ ./bitcoind -daemon
```

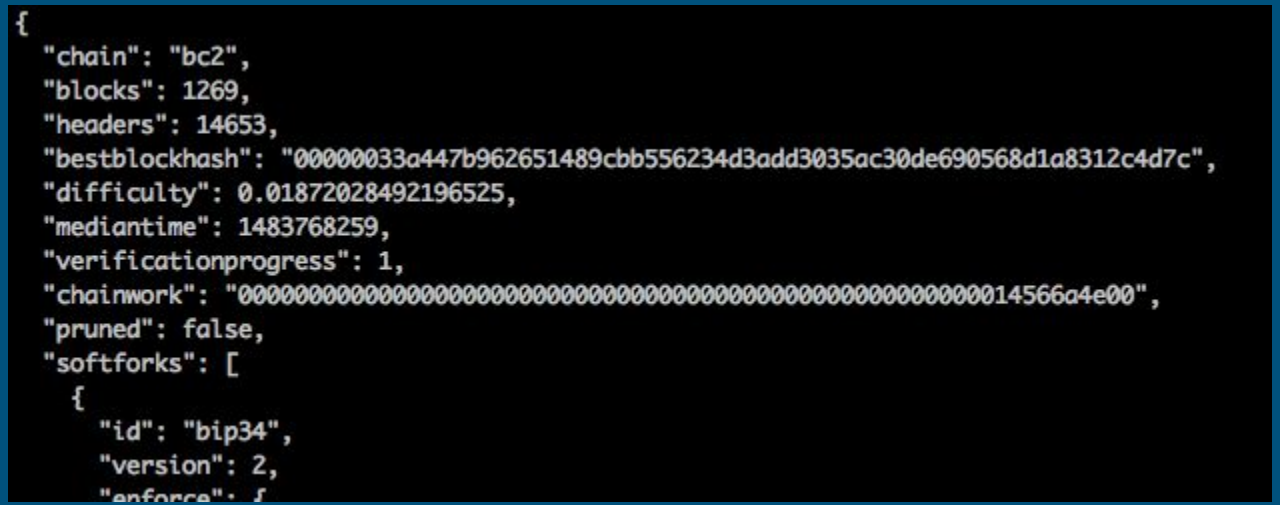
最初の出力

- ウォレット（秘密鍵）の作成
- ネットワークに繋ぐ
- ブロック情報の更新

確認

別のウインドウで:

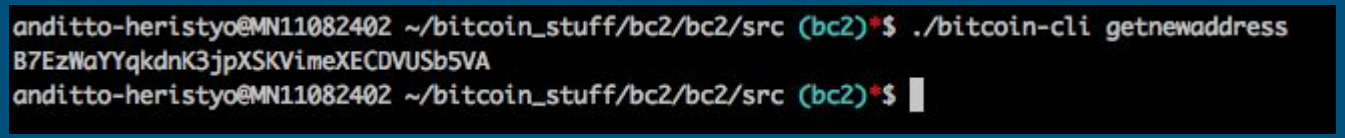
```
$ ./bitcoin-cli getblockchaininfo
```



(BC^2) Bitcoinをゲットしましょう！

アドレスを作成します:

```
$ ./bitcoin-cli getnewaddress
```



Bitcoinをゲットしましょう！

出力されたアドレスはここに入力してください：

http://172.16.120.201:8888/faucet.html

出力された TxID をどこかに保存してください。

残高のチェック：

```
$ ./bitcoin-cli getbalance
```

→ 何が出ましたか？

Mempool

トランザクション（Tx）が有効であれば、Mempool に入れます。
そして、ノードに送信します。

```
$ ./bitcoin-cli getrawmempool
```

マイニングします

```
$ ./bitcoin-cli generate 1
```

（マイニングは後ほど説明します）

```
$ ./bitcoin-cli generate 1 50000000
```

もう一度やってみると:

```
$ ./bitcoin-cli getbalance
```

BTCを送ってみよう

もう一度アドレスを作成して、グループ内の人にアドレスを聞いて、送ってみてください。

一番シンプルなやり方：

```
$ ./bitcoin-cli sendtoaddress <アドレス> <金額>
```

ブロックチェーン・エクスプローラ

BC²は: http://explorer.bc-2.jp

Bitcoinは: 検索すれば色々と見つかります。

データの信用性

結局自分のデータしか信用できません。

でも、他の人に頼らなくて、自分のデータがあるからこそ 分散台帳技術（Distributed Ledger Technology ）には価値があります。

bitcoin-cliのコマンド

```
$ ./bitcoin-cli help

$ ./bitcoin-cli help <コマンド>
```

https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list

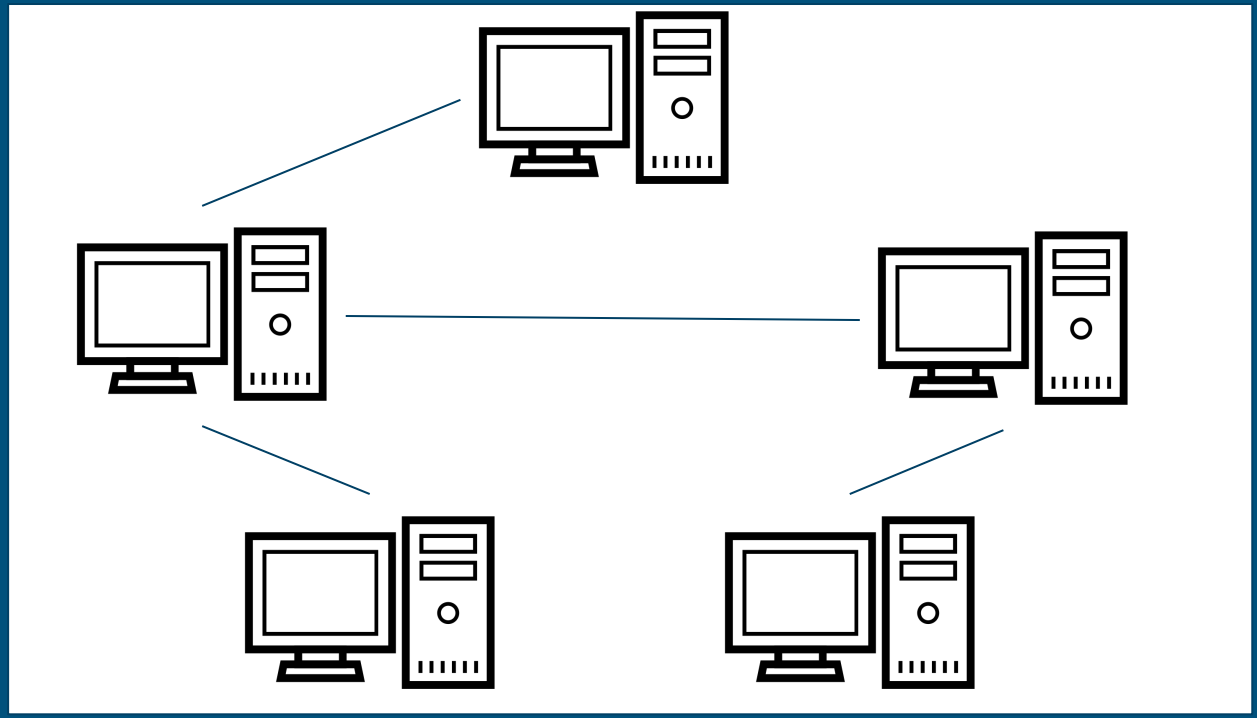
注意点：

ウォレットの”account”系のコマンドはバグっています。

Bitcoinの概要

3つのネットワーク

Bitcoinのネットワーク



Peer-to-Peer (P2P)

Icon by Creative Stall
From Noun Project

3つのネットワーク

1. Mainnet - メインネットワーク

```
$ bitcoin-cli <コマンド>
```

注意点

Mainnetはリアルなお金なので、気をつけてください。

3つのネットワーク

2. Testnet - テストネットワーク

```
$ bitcoin-cli -testnet <コマンド>
```

- 機能などは Mainnet とほぼ同じ。
- マイニングの難しさは低く設定されている。
- BTC は価値が無い。（だから価値がある）
- よく壊れている。

3つのネットワーク

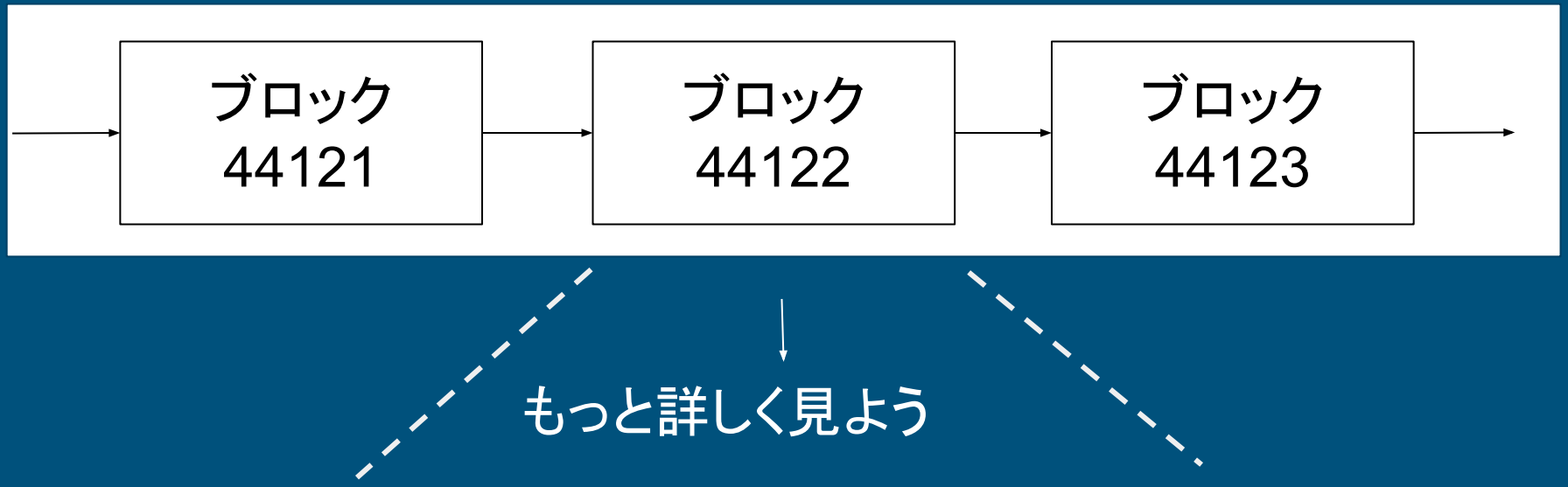
3. Regtest - Regression Test用のネットワーク

```
$ bitcoin-cli -regtest <コマンド>
```

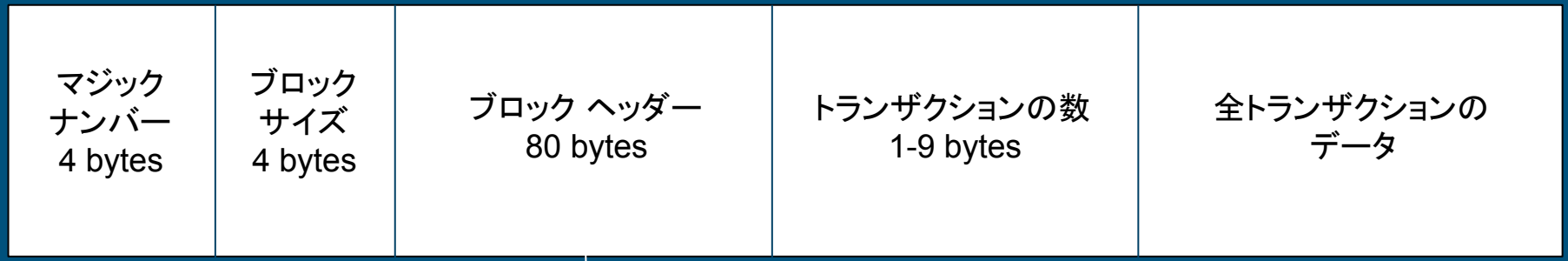
- 自分のローカル環境のみ。
- 自分でマイニングしないといけない。

ブロックチェーンの中身

ブロックチェーン



ブロックの中身



もっと詳しく見よう

```
$ ./bitcoin-cli getbestblockhash

$ ./bitcoin-cli getblock
```

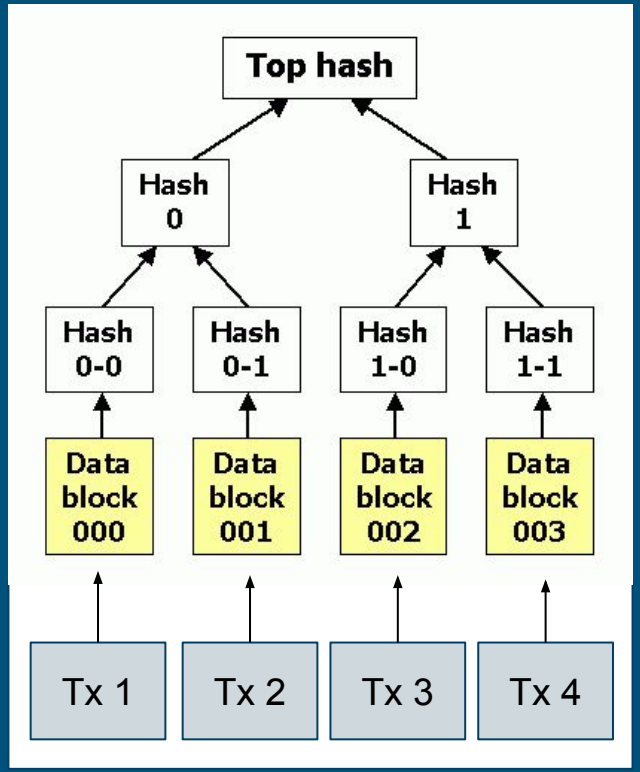
ブロックのヘッダー

バージョン 4 bytes	**hashPrevBlock** 前ブロックヘッダーの 256-bit ハッシュ 32 bytes	**hashMerkleRoot** トランザクション データのMerkleRoot 32 bytes	タイム スタンプ 4 bytes	**Bits** ターゲット （難しさ） 4 bytes	**Nonce** 32-bitの 数値 4 bytes

```
$ ./bitcoin-cli getbestblockhash

$ ./bitcoin-cli getblockheader
```

Merkle Root



https://ja.wikipedia.org/wiki/ハッシュ木

マイニング

マイニング

Bitcoin Coreの場合、自分の Regtest ネットワークであれば、先ほどと同じく:

```
$ ./bitcoin-cli -regtest generate 1
```

ASIC マイニング

Application Specific Integrated Circuit

Bitcoin（あるいは他のコイン）のためだけのIC。

例えば、Antminer S9は 14TH/s。

一方、現在私たちのネットワークは：

```
$ ./bitcoin-cli getmininginfo
```

Antminer S9
https://www.bitmain.com/

マイニング（実際は）

一番大きなマイナー Antpool、2015年2月の記事
http://qntra.net/2015/02/inside-the-bitcoin-mine-of-antpool-bw-com/

マイニングのインセンティブ

リソースを大量に使ってまで何故マイニングするのでしょう？

1. Block Reward（報酬）
    a. 現在は12.5 BTC
    b. CoinbaseのTxになる（TxInが無い）
2. Transaction Fee (手数料)
    a. ブロック全Txのデータから集める
    b. Block Reward は4年毎に半減し2140年頃ゼロになる

マイニングプール

基本的に:

参加者のプロセッシングパワーを合わせて、目指すハッシュ値が出る確率を高め、マイニングで獲得した報酬や手数料を分配します。

仕組みはマイニングプール毎に様々なので、ここでは説明しません。

Proof of Work

Proof of Work（PoW）

ブロックのハッシュ　＜　ターゲット

例えば:

0x00000000000000029d053e0ebe6a40db0fb47c5b24e30dbd751f0d767cfcda

<

0x00000000FFFF0000000000000000000000000000000000000000000000000000

もっと説明すると

1. 出力したハッシュの値は完全ランダム
2. 1番目のビットが “0” である確率は1/2
3. 2番目のビットも “0” である確率は更に1/2（つまり1/4）
4. ...

考え方：コインを投げ続けて、何回も勝ち続けている。

0x00000000000000000029d053e0...

Proof of Work（PoW）

```
$ ./bitcoin-cli getdifficulty
$ ./bitcoin-cli getblockchaininfo
```

difficulty = 最大のターゲット / 現在のターゲット
https://en.bitcoin.it/wiki/Difficulty

あるいは:

```
$ ./bitcoin-cli getblocktemplate
$ ./bitcoin-cli convertcompact <bits> #BC2のみ
```

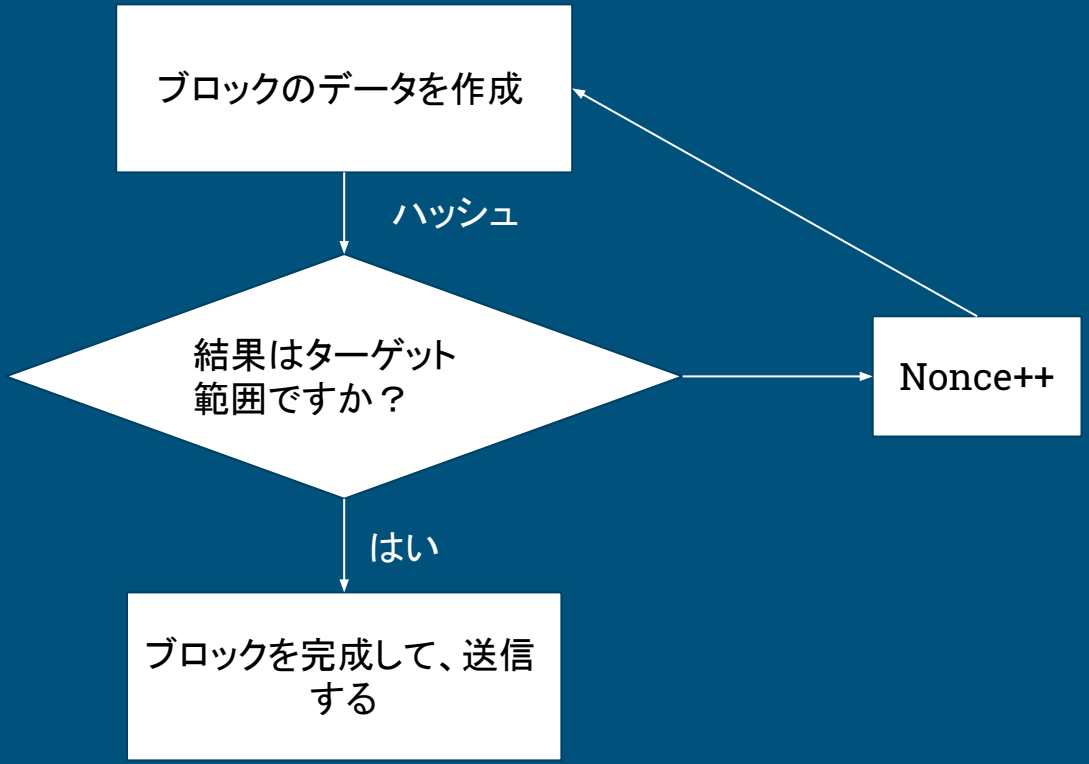
Proof of Work（PoW）

- Hashcash (Adam Back, 1997)

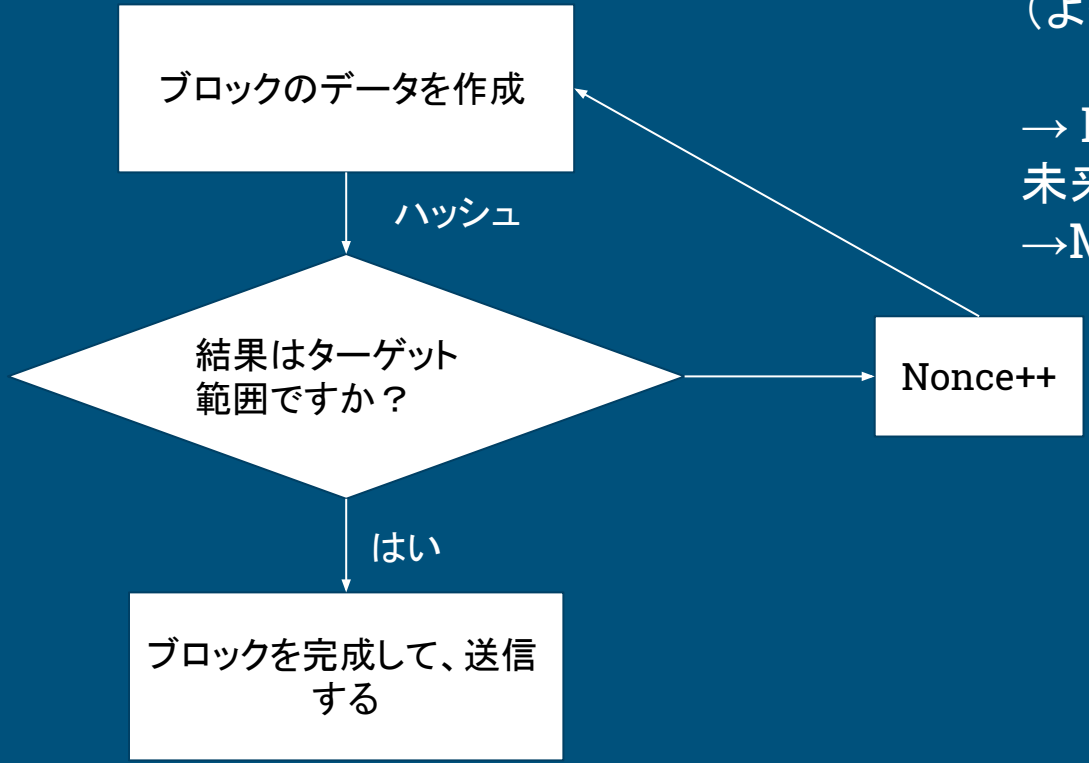
  （https://en.wikipedia.org/wiki/Hashcash）

- BTC Mainnet ：1ブロック → 10分ぐらい
  - 2016ブロックごとに再計算（2週間分）
- BC^2 ：1ブロック → 1分ぐらい
  - 60ブロックごとに再計算（1時間分）

Proof of Work（PoW）



Proof of Work（PoW）

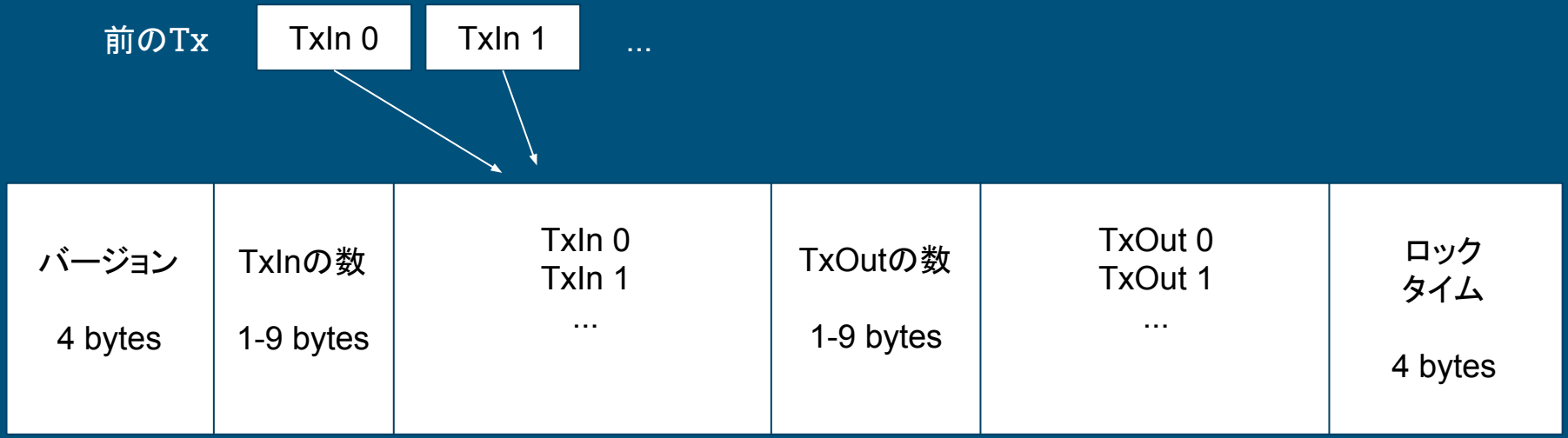


32-bit Nonceが限界になったら？
（よくあること）

→ ExtraNonce（時間++）
未来時間は2時間まで許されている
→MerkleRootの一番左のTxを変更
（Coinbase Tx）

トランザクション（Tx）

トランザクションの中身



午後のセッションで詳しく見ましょう。

TxInの中身

前のTxのハッシュ 32 bytes	そのTxのTxOutのインデックス 4 bytes	TxIn-スクリプトの長さ 1-9 bytes	**scriptSig** TxIn-スクリプト	シーケンス番号 (0xFFFFFFFF) 4 bytes

scriptSig → Unlocking Script

TxOutの中身

バリュー	TxOut-スクリプトの長さ	**scriptPubKey**
8 bytes	1-9 bytes	ロックのスクリプト

scriptPubKey → Locking Script

UTXO

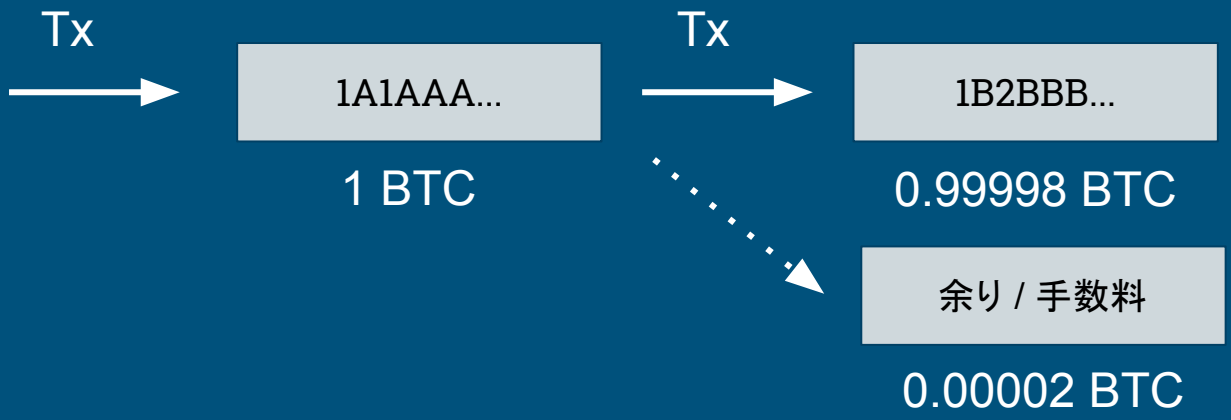
Unspent Transaction Output

まだ使われてない / 他のTxInとして使えるTxOut

ネットワーク全体のUTXO情報:

```
$ ./bitcoin-cli gettxoutsetinfo
```

手数料(Transaction Fee)



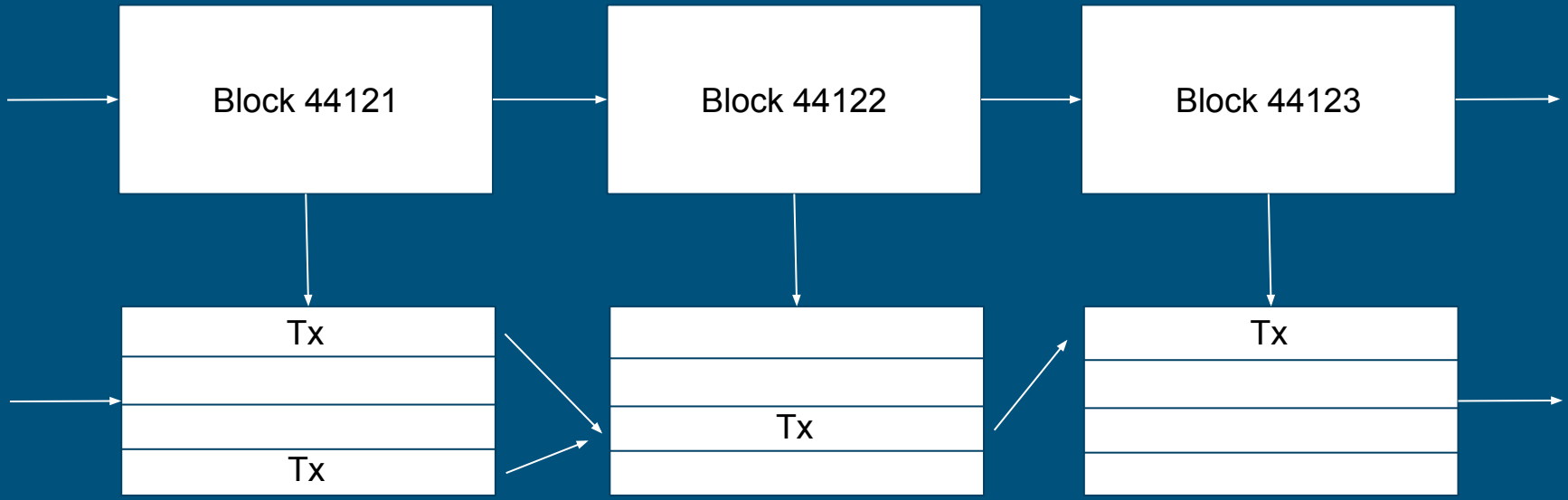
任意の値ですが、手数料を高く設定する理由は:

1. Txのサイズが大きい
2. 早めにブロックに入れて欲しい

トランザクションの流れ

1. Txの作成
2. Verify
3. ノードに送信する
4. 他のノードもVerifyする
5. Mempoolに入る
6. ノードの中にマイナーがいたらブロック作成時に組み入れる
7. ブロック作成が成功したら、ブロックチェーンに入る（confirmation=1）
8. その後ブロックが積み重なっていくとconfirmationの回数も上がる

ブロックチェーン



ブロックとTxの連鎖であり、ブロックとブロックの連鎖である。

コンセンサス

コンセンサス

複数の意味があります:

- プロトコル？
- デベロッパー？

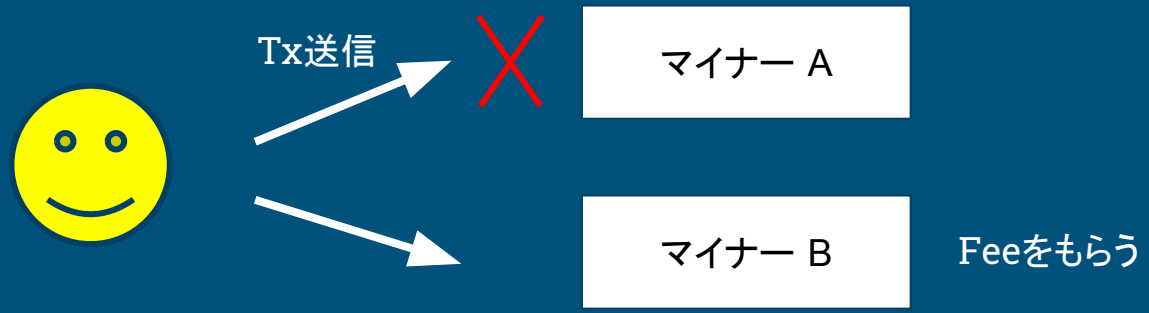
基本的に:皆が持っているブロックを同じにするためのルール。

考え方

望ましい（正しい）行動のほうがインセンティブが高い。

例：

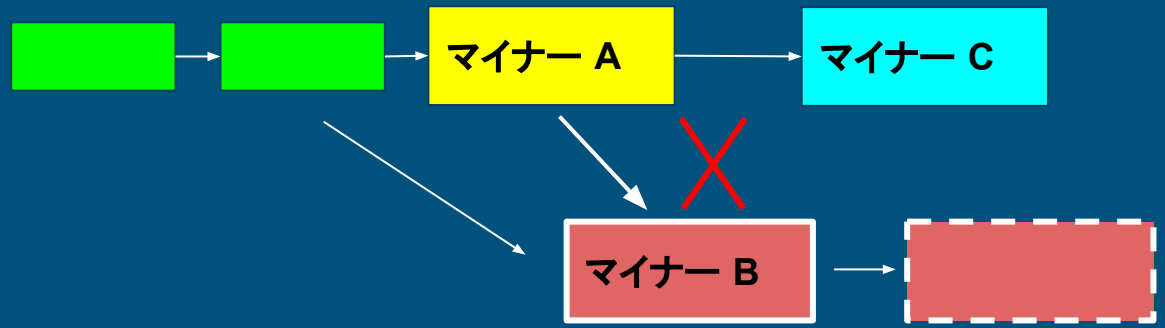
Tx をわざと却下するより、OKを出したほうが儲かる。



考え方

例:

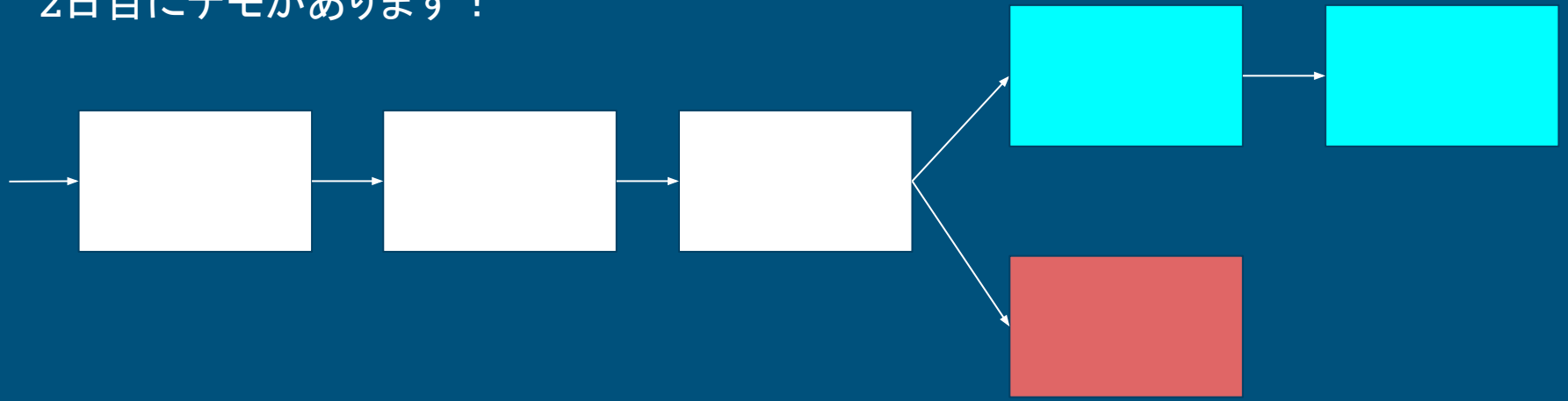
マイナーはブロックを却下するより、一番長い連鎖を作った方が儲かる。



コンセンサス

- どのチェーンが一番正しい？
- どういうブロックが有効（あるいは無効）？
- どのトランザクションが有効（あるいは無効）？

2日目にデモがあります！



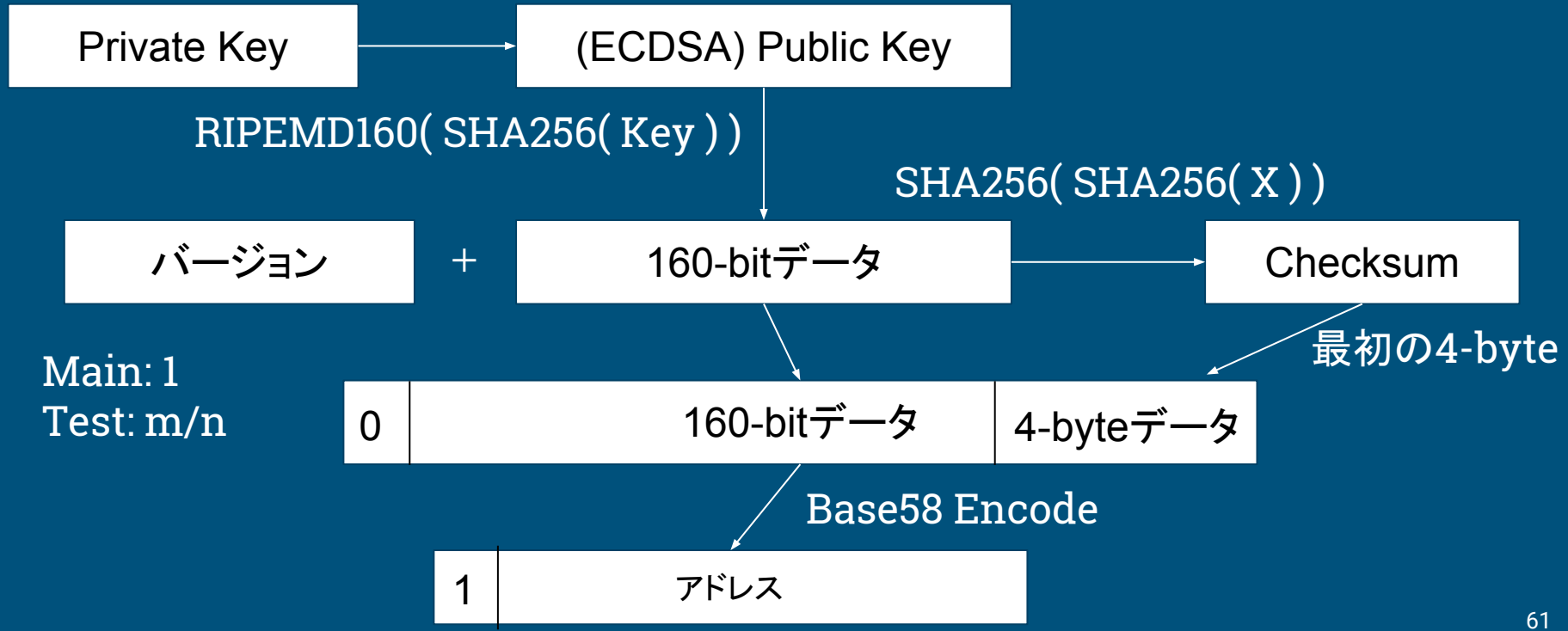
アドレス

Bitcoin アドレス

Mainnet の場合は現在 :

- P2PKH (Pay to Public Key Hash)
- P2SH (Pay to Script Hash)

Bitcoin アドレスの基本的な作り方



注意点

アドレスを使うのは1回のみ！（Address Re-use）

先ほども毎回 `getnewaddress` をやるべき。

問題：匿名性、セキュリティ、…

Address Re-use 問題

例：

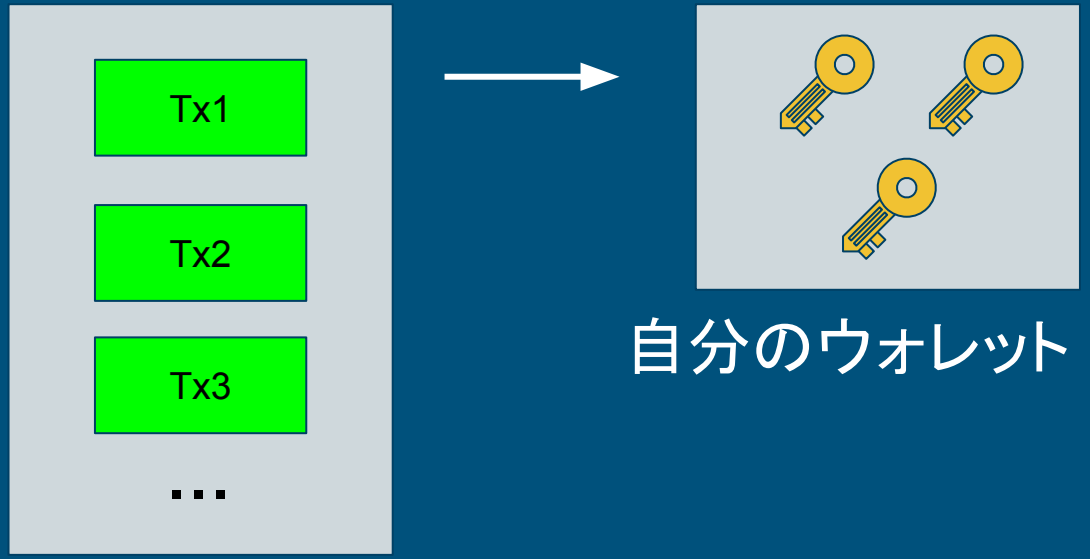
1. あなたのウォレットの残高はバレています
2. あなたの全てのTxがバレています
3. 自分が今はOKでも、次のTxを持ってる人は違うかも

別のセッションで、もっと詳しく説明します。

よくある間違い

- アドレスは“口座”ではありません
- アドレスにある金額は残高ではありません

UTXOモデル



ネットワークの UTXO Set

なぜUTXOモデル？

1. アカウンティング
    a. 通知
    b. インプットとアウトプットの管理
2. セキュリティ
    a. ダブルスペンド攻撃の防止
    b. プライバシー
3. スケーラビリティなど

アカウントモデルとは別のパラダイム

ウォレット

ウォレット

Bitcoinはオープンなシステムなので、鍵の管理システムは一番重要なセキュリティのポイント

注意

自分で鍵を管理していないことは自分のお金では無いのと同じです。

よくあるパターン

ユーザがエクスチェンジ（交換所）で円からBTCに交換した後、秘密鍵をそのまま同じサービスに放置する。

1つの解決方法：交換後に自分のウォレットに移動する。

今までの大きな事件：MtGox、Bitfinexなど

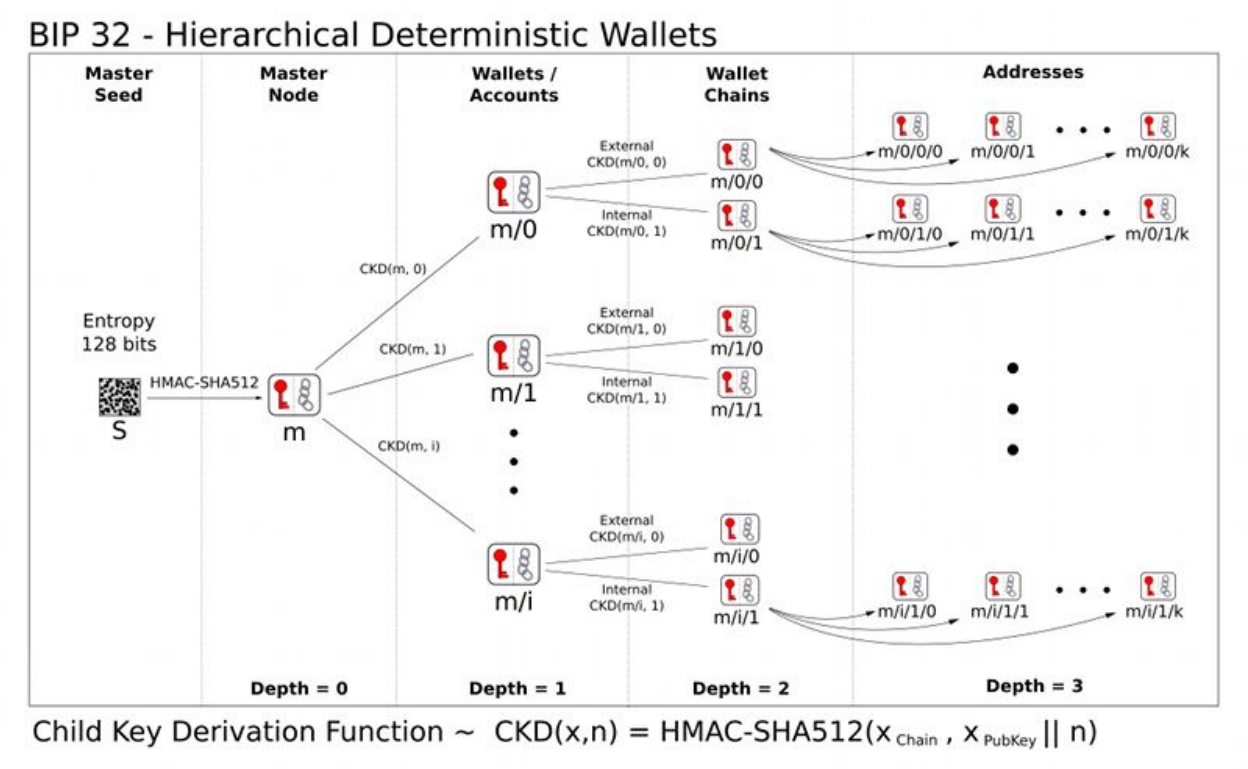
自分のBitcoindのウォレットの場合

- 環境にウイルス、Malware、キーロガーなどが無いことを確認する。
- ハードディスクを暗号化する。
- ウォレットの暗号化：
  - `$ ./bitcoin-cli encryptwallet “PASSPHRASE”`
  - `$ ./bitcoin-cli walletpassphrase “PASSPHRASE” TIMEOUT`

カリキュラムの都合上、**BC2ではウォレットの暗号化は行わない**でください。

HD (Hierarchical Deterministic) ウォレット

BIP 32



ハードウェア・ウォレット

https://www.ledgerwallet.com/

https://trezor.io/

ハードウェア・ウォレット

メリット:

1. 秘密鍵をハードウェア外に持ち出せない。
2. HDウォレットなので、ハードウェアを無くしてもウォレットを再現することが出来る。
3. Plausible Deniability / 犠牲ウォレット (Ledgerのみ)。

anditto@dglab.com
github.com/anditto