

From the Editors

It is a great pleasure to deliver to you the 7th issue of “Journal of Informatics and Regional Studies.” This Journal intends to provide researchers and practitioners with the forum of discussion and sharing findings and ideas about Informatics and Regional Studies. We welcome you to join us to share your idea on this Journal.

This volume is a Special Issue to cover the topics of Bit Coin. There are wide varieties of research and papers from commercial based to academic. However, because of the nature of the system which is basically written in English and focused more on economic implications, it is difficult to find written documents which cover both technical and social implications in Japanese.

Professor Shigeichiro Yamasaki, in collaborations with me (Okada), has been providing such a work. He held dedicated workshops about Bit Coin, many times. This Special Edition of Journal of Informatics and Regional Studies presents the readership the fruit of such a study.

This volume's first part is a reproduction of Satoshi Nakamoto's paper adopted from Bitcoin Wiki under Creative Commons Attribution 3.0. Then, Professor Yamasaki's presentations from the workshops hosted by him are reproduced.

This edition of Journal is consisted by the collections of up-to-date detailed research topics on Bitcoins. The editor would like to express sincere thank to the contributor of paper and the presenter and discussants in the Workshops who make this wonderful omnibus of journal come to existence.

HITOSHI OKADA, PhD, *Editor-in-Chief*

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo 101-8430 JAPAN

SHIRO UESUGI, PhD, *Executive Editor*

Matsuyama University
4-2 Bunkyo, Matsuyama City,
Ehime 790-8578 JAPAN

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshi@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

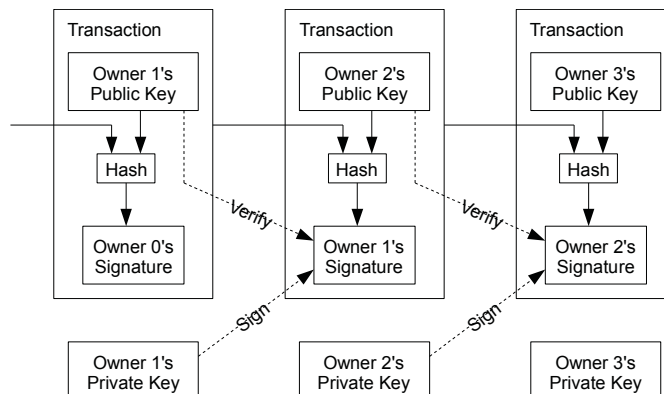
Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

* This paper is adopted under Creative Commons Attribution 3.0 from
https://en.bitcoin.it/wiki/Bitcoin_white_paper

2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

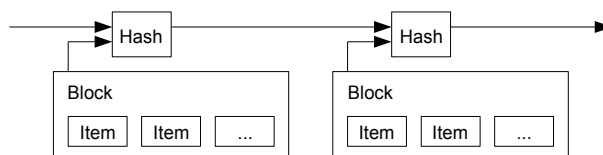


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

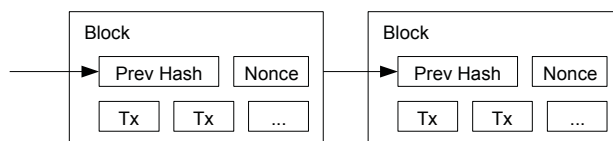
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

5. Network

The steps to run the network are as follows:

- 1) New transactions are broadcast to all nodes.
- 2) Each node collects new transactions into a block.
- 3) Each node works on finding a difficult proof-of-work for its block.
- 4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
- 5) Nodes accept the block only if all transactions in it are valid and not already spent.
- 6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

6. Incentive

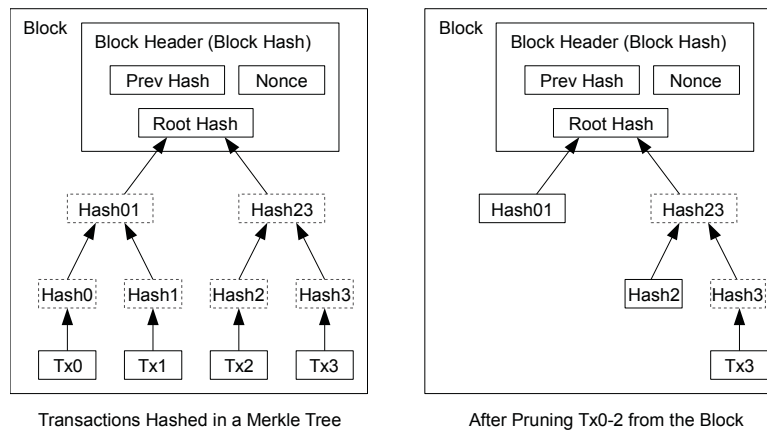
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

7. Reclaiming Disk Space

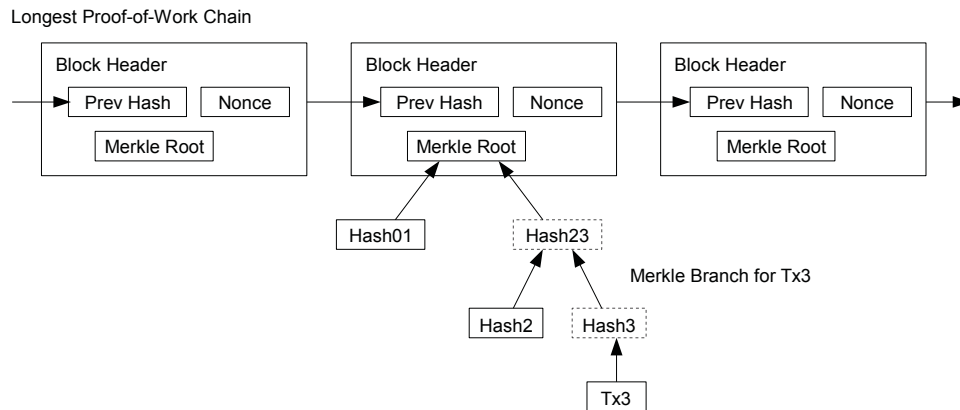
Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, $80 \text{ bytes} * 6 * 24 * 365 = 4.2\text{MB}$ per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

8. Simplified Payment Verification

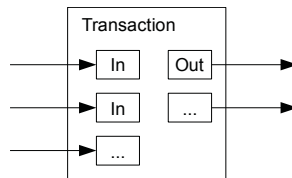
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

9. Combining and Splitting Value

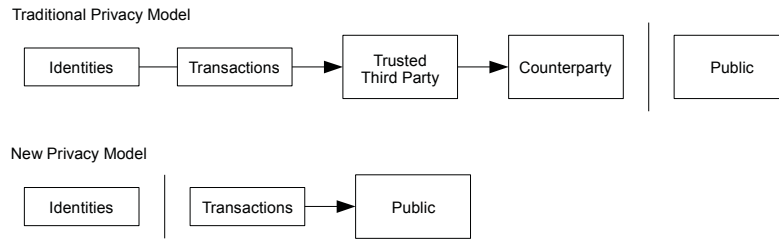
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

10. Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

11. Calculations

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:

p = probability an honest node finds the next block
 q = probability the attacker finds the next block
 q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z \frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \begin{cases} (q/p)^{(z-k)} & \text{if } k \leq z \\ 1 & \text{if } k > z \end{cases}$$

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^z \frac{\lambda^k e^{-\lambda}}{k!} (1 - (q/p)^{(z-k)})$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```


Running some results, we can see the probability drop off exponentially with z .

```
q=0.1
z=0    P=1.0000000
z=1    P=0.2045873
z=2    P=0.0509779
z=3    P=0.0131722
z=4    P=0.0034552
z=5    P=0.0009137
z=6    P=0.0002428
z=7    P=0.0000647
z=8    P=0.0000173
z=9    P=0.0000046
z=10   P=0.0000012
```

```
q=0.3
z=0    P=1.0000000
z=5    P=0.1773523
z=10   P=0.0416605
z=15   P=0.0101008
z=20   P=0.0024804
z=25   P=0.0006132
z=30   P=0.0001522
z=35   P=0.0000379
z=40   P=0.0000095
z=45   P=0.0000024
z=50   P=0.0000006
```

Solving for P less than 0.1%...

```
P < 0.001
q=0.10  z=5
q=0.15  z=8
q=0.20  z=11
q=0.25  z=15
q=0.30  z=24
q=0.35  z=41
q=0.40  z=89
q=0.45  z=340
```

12. Conclusion

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

References

- [1] W. Dai, "b-money," <http://www.weidai.com/bmoney.txt>, 1998.
- [2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.
- [3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.
- [4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.
- [5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.
- [6] A. Back, "Hashcash - a denial of service counter-measure," <http://www.hashcash.org/papers/hashcash.pdf>, 2002.
- [7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.
- [8] W. Feller, "An introduction to probability theory and its applications," 1957.

bitcoin

勉強会 1

CACANet Fukuoka / 近畿大学
山崎重一郎

今日の目的は

bitcoinの仕組みを理解したい

bitcoin

□ Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto

(山崎訳をつくりました)

- 完全なP2P型電子コイン
- 「信頼できる第三者機関」を一切利用しない

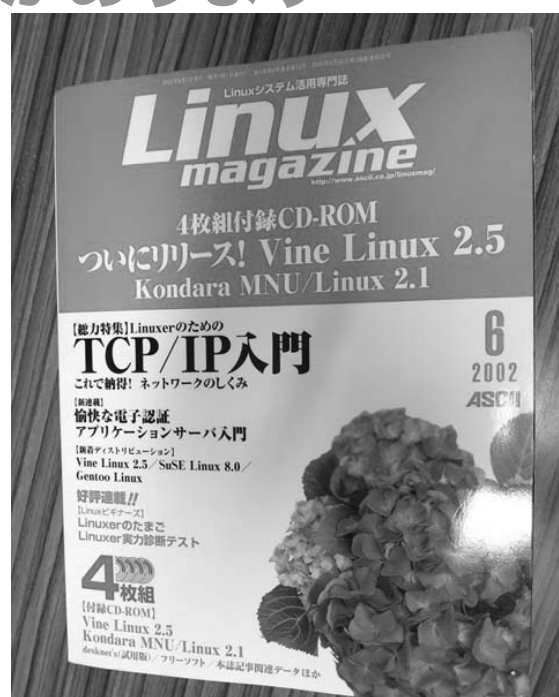
実はP2P型の電子地域通貨を 実装したことがあります

Travecoup

(トラベリング・クーポン)

Linux Magazine で連載
2002年6月号、8月号

OpenSSL とRubyで実装
pkcs#7電子署名



Travecoupの ヤップ島の石貨モデル



コインを偽造するのに
すごい労力がかかる

歴代所有者の名前を彫る

最新の所有者がわかるよう
にする

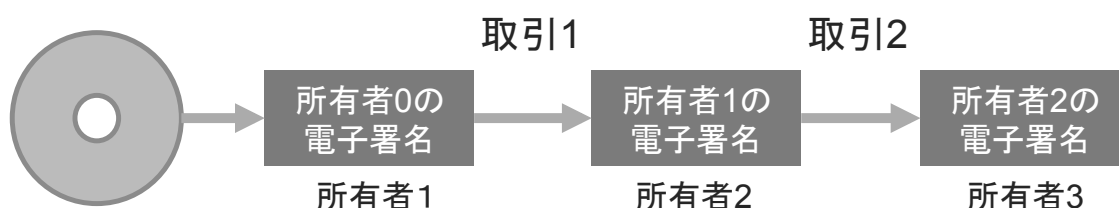
電子コイン＝電子署名のチェーン

□ Travecoupの電子コイン

秘密鍵と公開鍵のペアを持っている (PKI)

□ bitcoinの電子コインも同じ

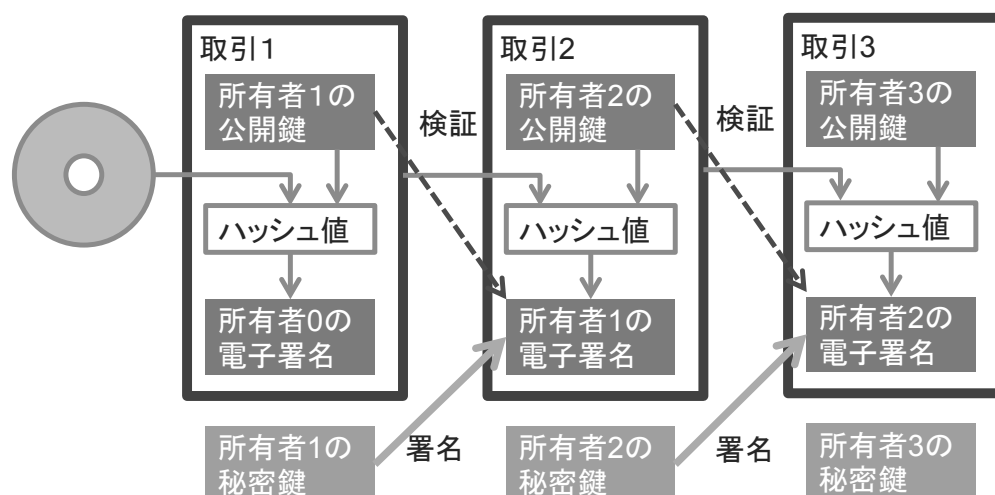
取引のたびに、前の所有者が次の所有者の公開鍵に
電子署名する



bitcoinの取引と署名検証

□ 取引（受取人の公開鍵への署名）

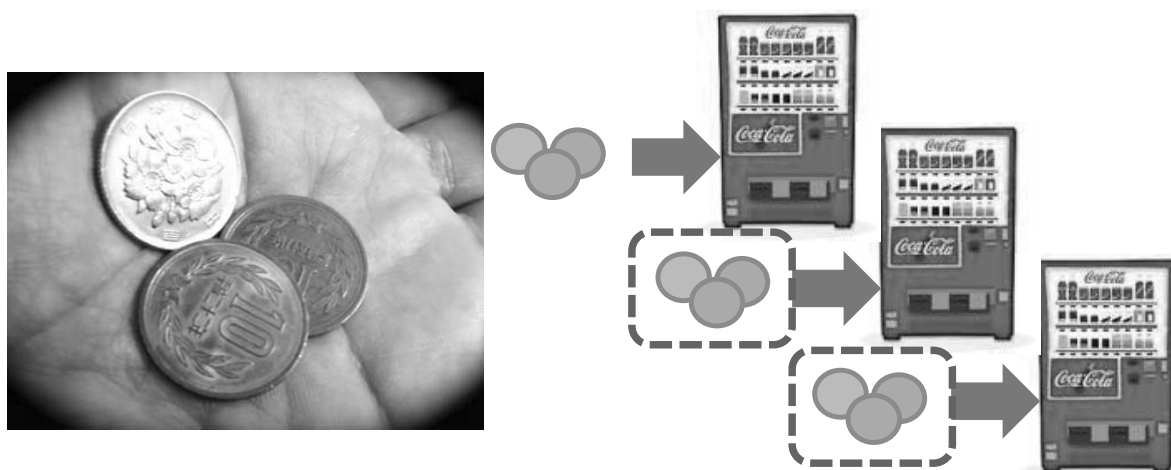
□ bitcoin=取引の電子署名のチェーン



電子コインの多重使用問題

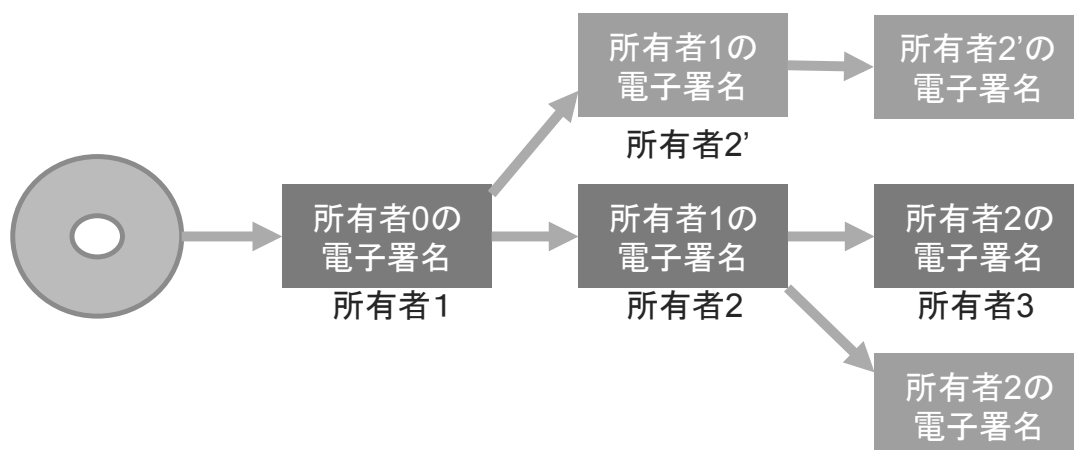
□ 現実のコインは、使えば手元からなくなる

□ 電子コインは、使っても、まだ手元にある！



電子コイン＝電子署名のチェーン

□ 多重使用＝チェーンが分岐すること



P2P型で多重使用を防止

□ 取引という出来事を「公的」に保証

タイムスタンプサービス

□ P2Pで「公的」な証拠を構成する

プルーフオブワーク

□ 「公的」とは「みんなの意見」のこと

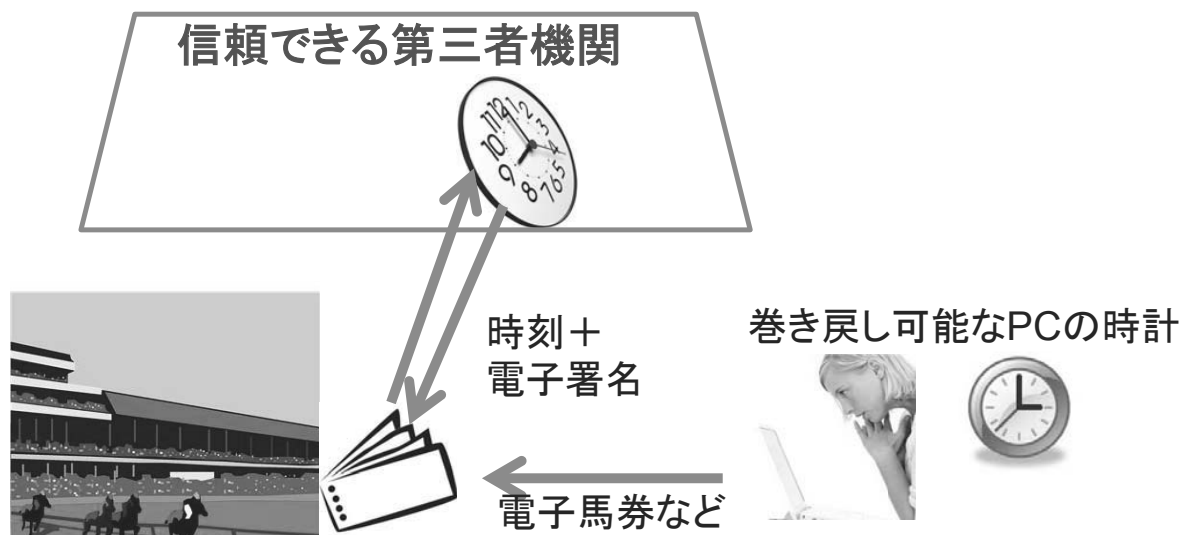
★ 「信頼できる第三者機関」じゃない！

P2Pの多数決で「みんなの意見」を判断する！

普通のタイムスタンプサービス

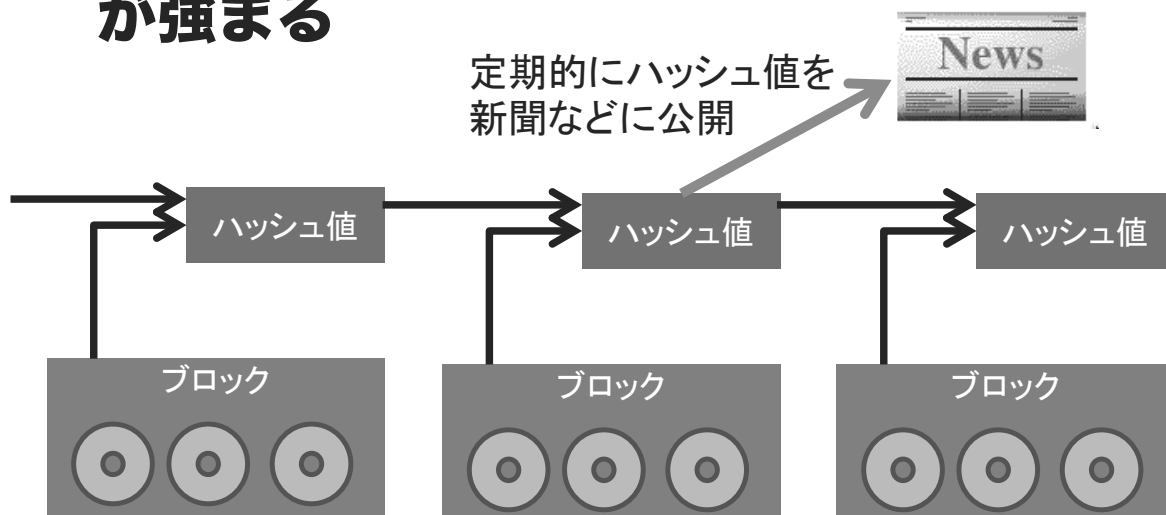
□出来事の実在を「公的」に保証する

取引、契約、賭博、発明特許、届け出



タイムスタンプ

□タイムスタンプが増えるほど、実在性が強まる



P2P型タイムスタンプサービス

□ 証明したいこと

電子コインによる取引の实在

取引の順序

□ 「みんなの意見」

偽装をしようとする、膨大なコストがかかる！

★プルーフ・オブ・ワーク

プルーフ・オブ・ワーク

□ がんばった証拠（P2Pで確認可能な）

例. もしパスワードのタイプを間違ったら …

（ブルートフォース対策として）

10秒待てば、再入力できる → ×

腕立て100回やったら、… → ×

検証可能なCPUを酷使する計算、… → ○

プルーフ・オブ・ワーク

□ ハッシュ計算を使う例 (Hashcash)

データ + nonce (いろいろ変えてみる)

↓ SHA256

0000000001000111010001...1



「ハッシュ値がゼロ8個で始まるnonceを求めよ」

プルーフ・オブ・ワーク

□ (長期的に) 難しい計算？

計算機の性能向上にあわせて、ゼロの数を増やす

データ + nonce (いろいろ変えてみる)

↓ SHA256

0000000000000111010001...1



「ハッシュ値がゼロ9個で始まるnonceを求めよ」

bitcoinのタイムスタンプ

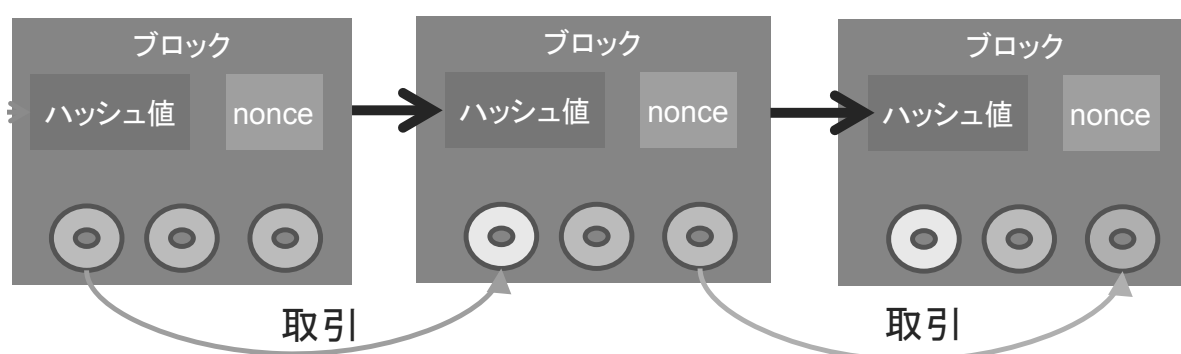
□ プルーフ・オブ・ワークを利用

取引はP2Pネットワーク全体に放送される

ノードは取引をブロックに取り込む

ノードは、プルーフ・オブ・ワークを探す

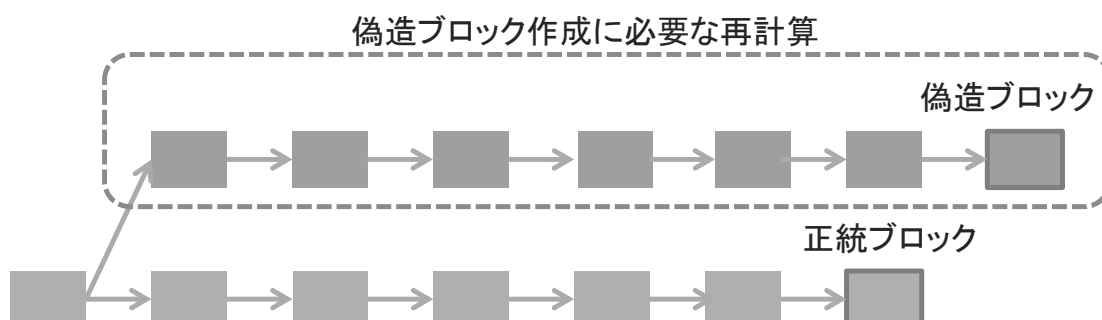
プルーフ・オブ・ワークを見つけたノードは、それを放送



CPUパワーによる正統性証明

□ 最長のチェーンが正統なタイムスタンプ

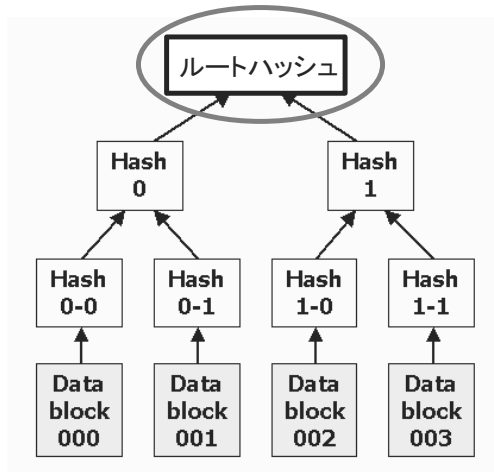
□ 偽造ブロックの作成には、チェーン全体の再計算が必要



マークル・ツリー構造

□ 分散ファイルのハッシュ値を検証可能にする

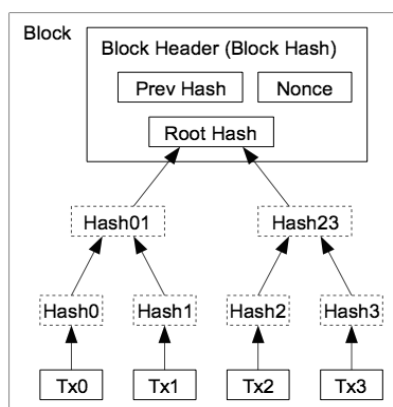
ルートハッシュさえ安全に入手できれば、それ以下のハッシュ値やデータ本体は、P2Pネットワークのどこから入手してもよい



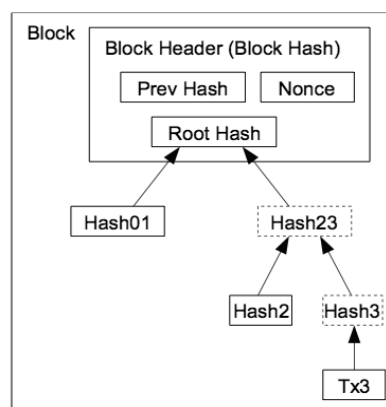
ブロック情報のコンパクト化

□ マークル・ツリーのルートハッシュだけをブロックヘッダに入れる

□ 実際の取引情報は、必要に応じて入手する



Transactions Hashed in a Merkle Tree



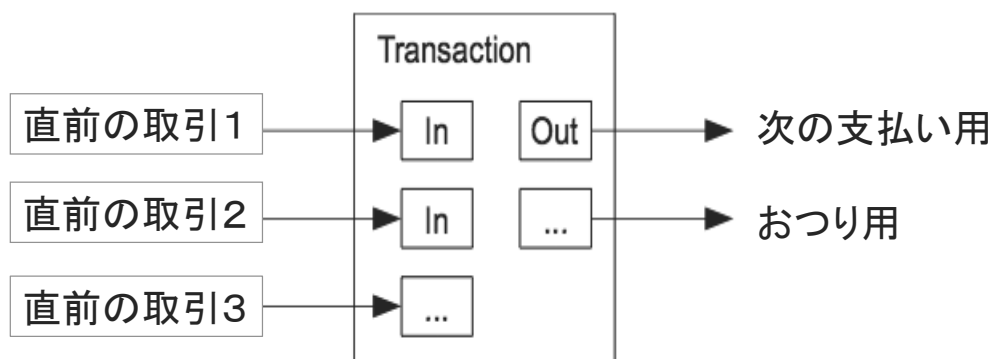
After Pruning Tx0-2 from the Block

価値の分割と統合

□取引には

インプット：1～複数個

アウトプット：1 or 2



bitcoinのプライバシー

□すべての取引は「公に」なっている

□公開鍵と本人のリンク可能性は消せる

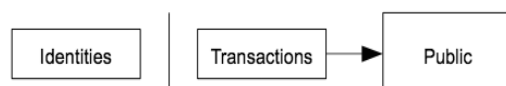
取引のたびに鍵ペアを新規に生成すればより強力

複数のインプットがある取引では、同一性が維持される
のでリンク可能になる

Traditional Privacy Model



New Privacy Model



ギャンブラーの破滅問題

- カジノとギャンブラーの勝率がそれぞれ50%
- ゲームを続けるとどうなるか？

→ **カジノの圧勝**

- 所持金が多い方が有利だから

攻撃者がチェーンを偽造する

- 追いつき、追い越す確率
- ギャンブラーの破滅問題になる

bitcoin

勉強会2

近畿大学
山崎重一郎

今日の目的

bitcoinの仕組

サトシ・ナカモトの論文にそって

bitcoinへの疑問

ビジネス利用に関連して

bitcoinでビジネス（峰松さん）

bitcoinクライアントのインストール
運用方法や注意点などあれこれ

bitcoin

□ 原典の論文

Bitcoin: A Peer-to-Peer Electronic Cash System **Satoshi Nakamoto**

完全な**P2P**型電子コイン

「信頼できる第三者機関」を一切利用しない

完全なP2P型電子貨幣の利点

□ 金融機関を利用するコストがない

現金に近いコストで電子決済ができる

□ 可逆取引の範囲が広がる

(返品などのときの追加負担が小さい)



完全なP2P型電子貨幣の利点

□ 現金と同程度の決済のプライバシー

当事者だけで決済できる

□ 当事者の信用確認が不要

現金なら、幼児でもジュースが買えるのと同じ



bitcoin = 電子署名のチェーン

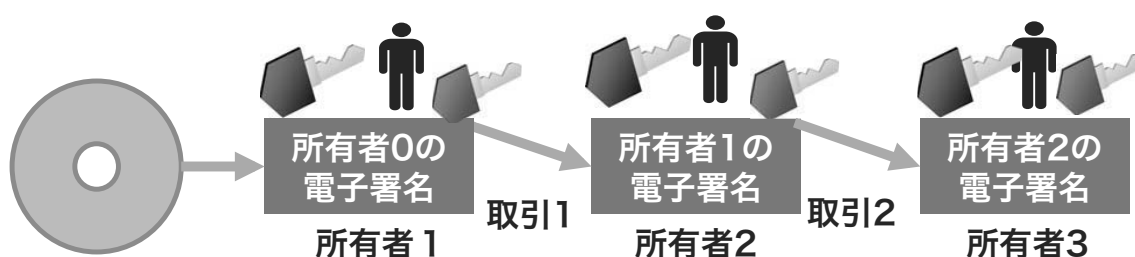
□ 公開鍵暗号

全員が「秘密鍵」と「公開鍵」の鍵ペアを持っている

□ 取引

取引 = 「所有権を変更しました」 証明書（電子署名）

bitcoin = 取引のチェーン



秘密鍵と公開鍵

□鍵の生成 (=Bitcoin財布をつくる)

取引のたびに鍵生成する方が安全



#秘密鍵 (極秘！これで電子署名する)

4e52fe643051db48dedb2fcfe95cbcce6b2d87d78a83
609f1cabcd7b1d994716



#公開鍵 (公開：これで電子署名を検証できる)

0489ec619ab6fda6982e25739971ca046488c0b506bf
8a61ccc3d08f6d11cb01cc54dd1c026cc0407a94365c
3404f1232494950a71338d297447b85f298205cc22

bitcoinアドレス

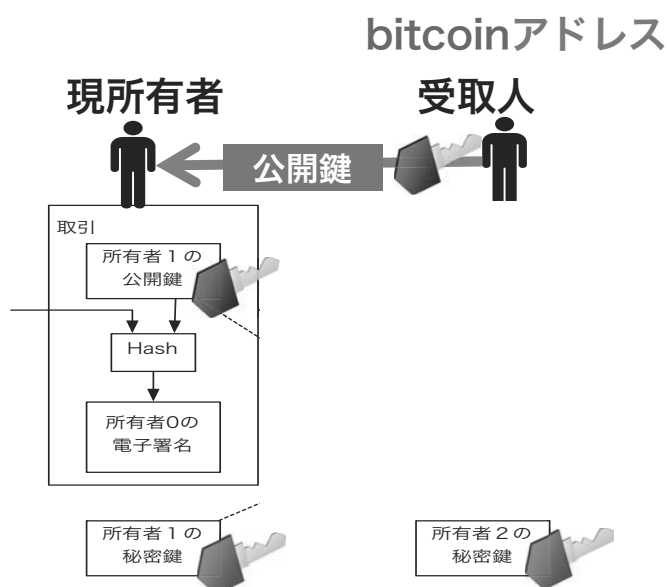
□公開鍵のハッシュ値

base58(RIPEMD-160(SHA-256(公開鍵)))

1NDdooBLUd6nBuLpRgtjER2pPDdsA9kKPC

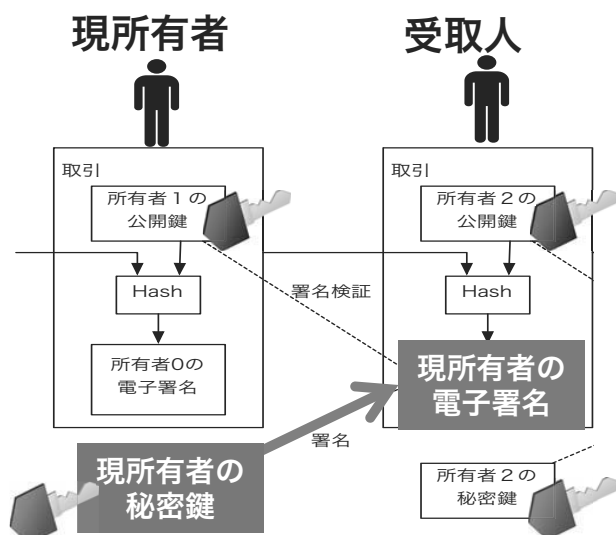
bitcoinの取引

(1) 現所有者が、受取人の公開鍵を受け取る



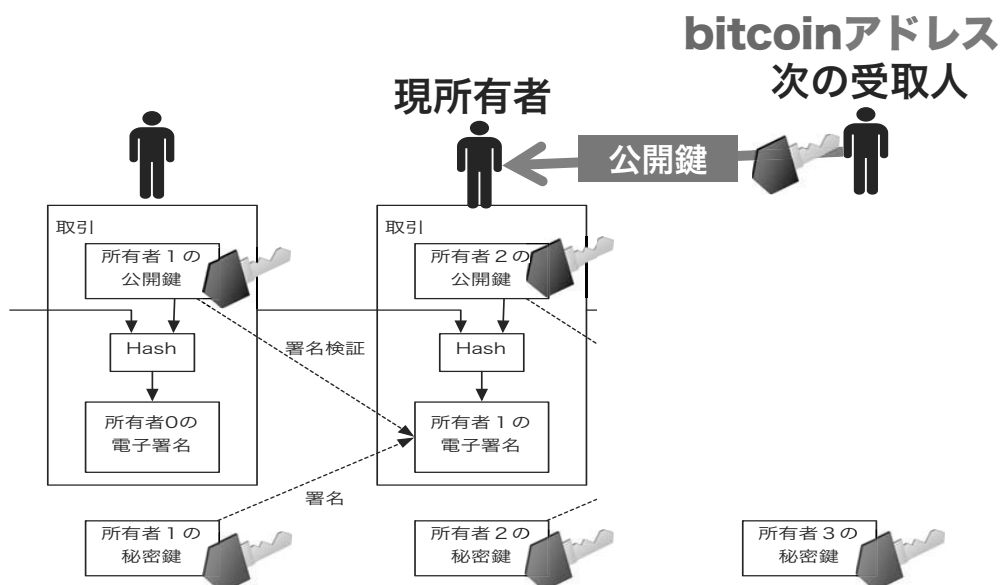
bitcoinの取引

(2) 「取引履歴」と「受取人の公開鍵」の結合に、 現所有者の秘密鍵で電子署名



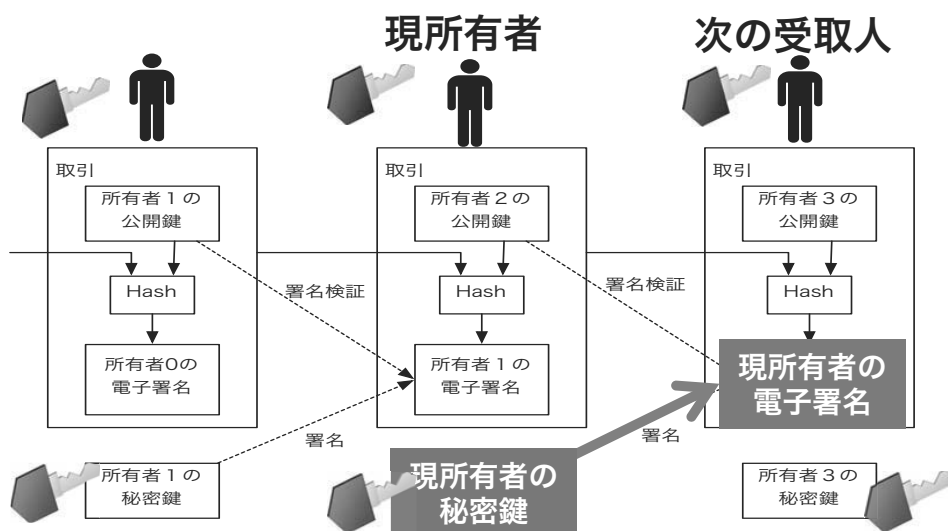
bitcoinの取引

(3) 現所有者が、次の受取人の公開鍵を受け取る



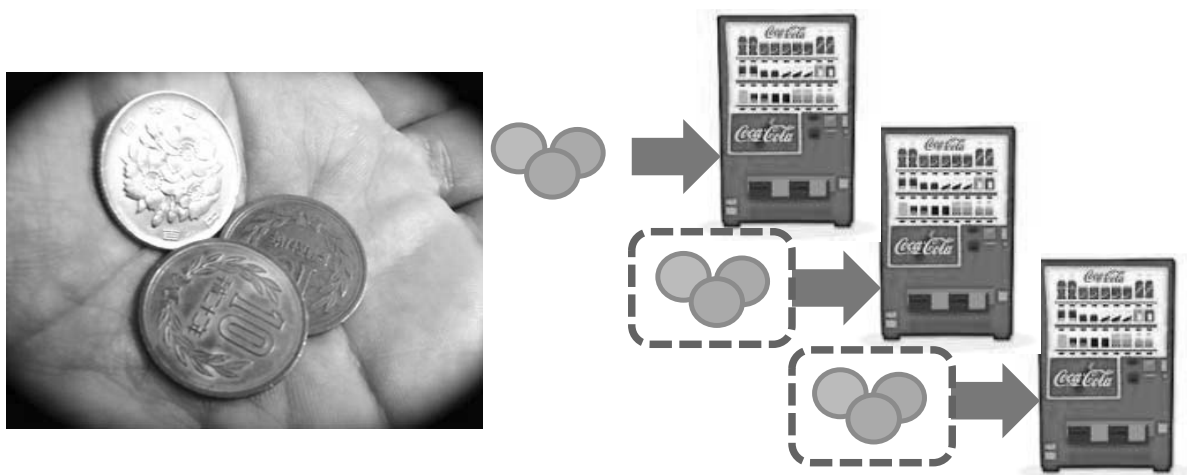
bitcoinの取引

(4) 「取引履歴」と「次の受取人の公開鍵」の結合に、 現所有者の秘密鍵で電子署名



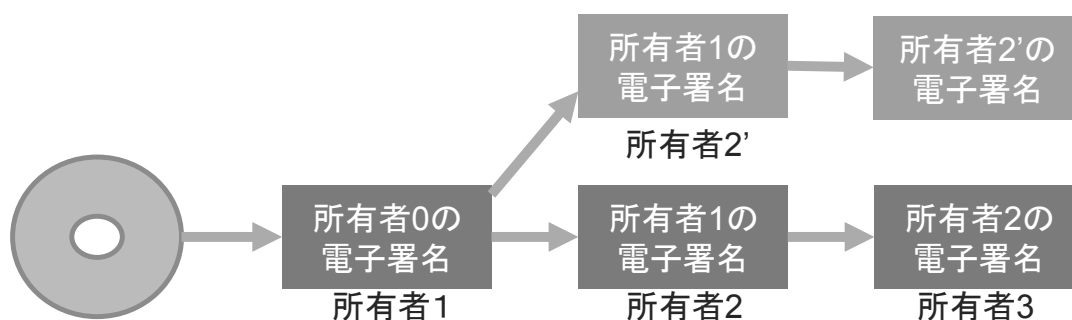
電子コインの多重使用問題

- 現実のコインは、使えば手元からなくなる
- 電子コインは、使っても、まだ手元にある！



電子コインの多重使用問題

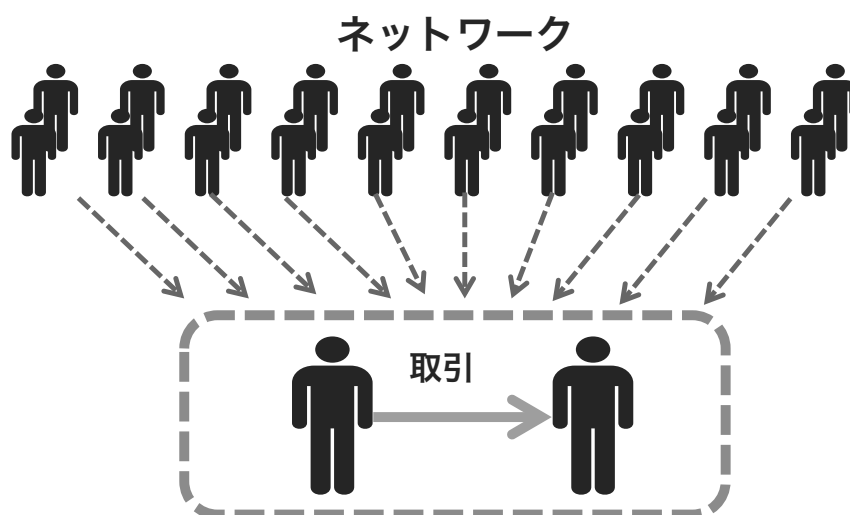
- 多重使用＝署名チェーンが分岐すること



多重使用が「無かったこと」の証明

□すべての取引を監視しないといけない！

ネットワークの大多数の人が取引を見ている



P2P型タイムスタンプサービス

□bitcoinの中核アイデア！！

暗号技術とP2P技術の絶妙の組み合わせ

個々の技術は「枯れている」ものを利用

□git などのP2Pシステムにも似ている

取引のタイムスタンプを「分散ファイル共有」

P2P型タイムスタンプサービス

□ 偽造防止方法は暗号技術に頼らない

□ 「大多数の力」を利用

多数決で正しさを証明

投票にCPUパワーを消費する計算が必要

=プルーフ・オブ・ワーク

プルーフ・オブ・ワーク

□ ハッシュ計算を使う例 (Hashcash)

データ + nonce (いろいろ変えてみる)

↓ SHA256

000000001000111010001...1

「ハッシュ値がゼロ8個で始まるnonceを求めよ」

プルーフ・オブ・ワーク

□将来すごい計算機が出てきたら？

計算機の性能向上にあわせて、ゼロの数を増やす

データ + nonce (いろいろ変えてみる)

↓ SHA256

000000000000000111010001...1

「ハッシュ値がゼロ9個で始まるnonceを求めよ」

プルーフ・オブ・ワーク

□10分あたりのブロック算出数を一定に保つように調整

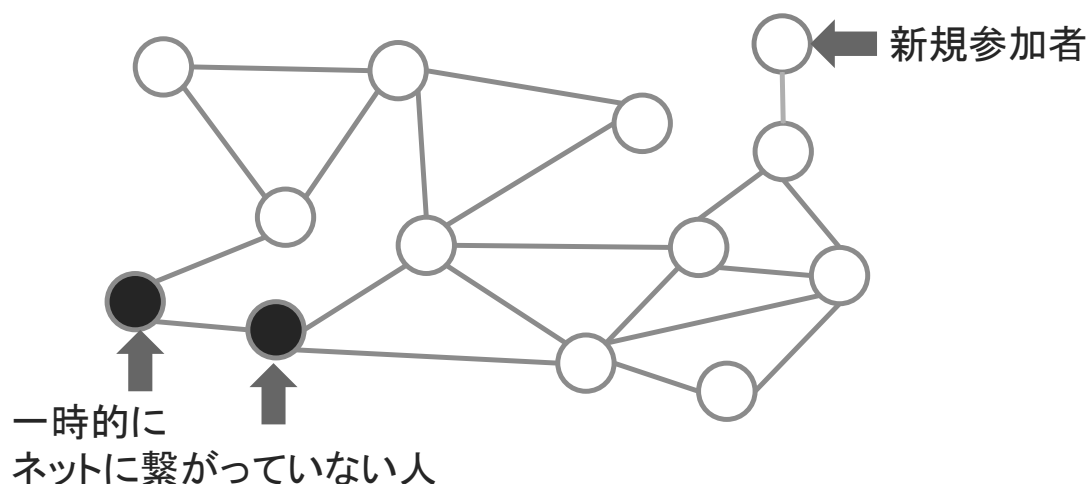
最初の4年で10,500,000 BTC

約4年ごとに半減させて追加

ブロック算出速度が速くなりすぎると、
プルーフオブワークスの問題の困難さを増加させま

bitcoinのP2P型ネットワーク

□ ノードは、自由に離脱／再接続が可能



P2Pネットワークへの接続方法

□ DNS seed

以下のようなDNで（wellknownな）ノードのIPアドレスがわかる

seed.bitcoin.sipa.be

dnsseed.bitcoin.dashjr.org

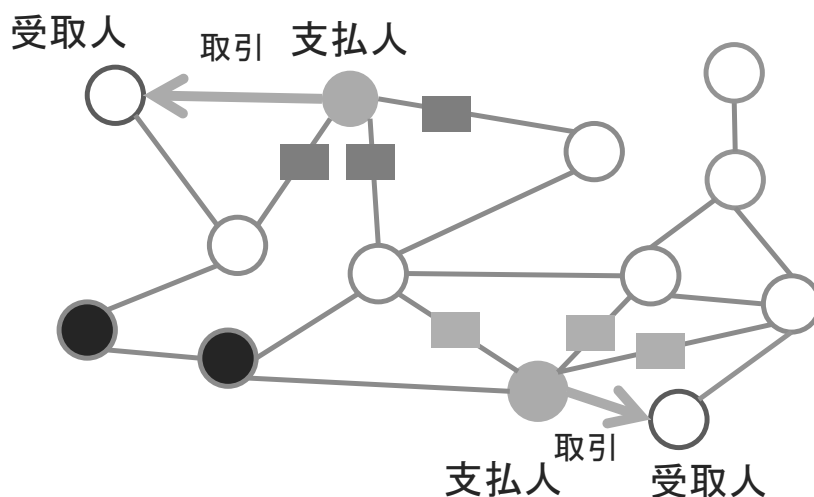
bitseed.xf2.org

```
$ host bitseed.xf2.org
bitseed.xf2.org has address 91.121.164.210
bitseed.xf2.org has address 46.19.139.72
bitseed.xf2.org has address 184.174.153.96
bitseed.xf2.org has address 187.139.68.233
bitseed.xf2.org has address 212.7.21.46
bitseed.xf2.org has address 69.164.206.88
bitseed.xf2.org has address 62.75.216.13
bitseed.xf2.org has address 207.164.207.54
bitseed.xf2.org has address 78.129.167.5
bitseed.xf2.org has address 177.71.192.164
bitseed.xf2.org has address 69.64.34.118
bitseed.xf2.org has address 62.75.253.91
bitseed.xf2.org has address 94.23.6.26
bitseed.xf2.org has address 176.9.142.163
```

bitcoinの取引発生

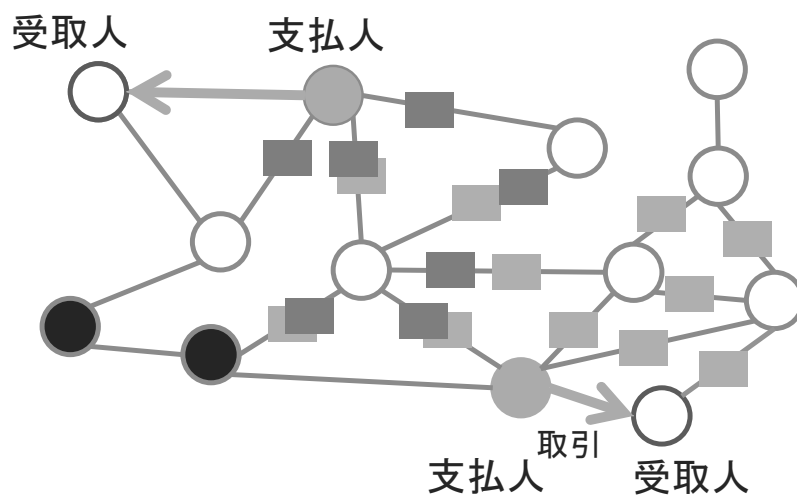
□ ある時間で新しく取引が発生

支払人がそれぞれ「取引」を全ノードに放送



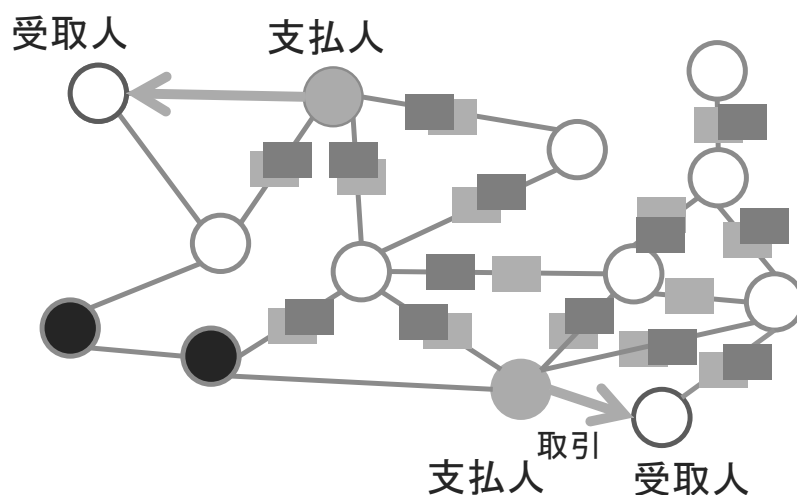
P2Pネットワークの放送

□ 各ノードは、受信データを他ノードにパス



P2Pネットワークの放送

□ 各ノードは、受信データを他ノードにパス

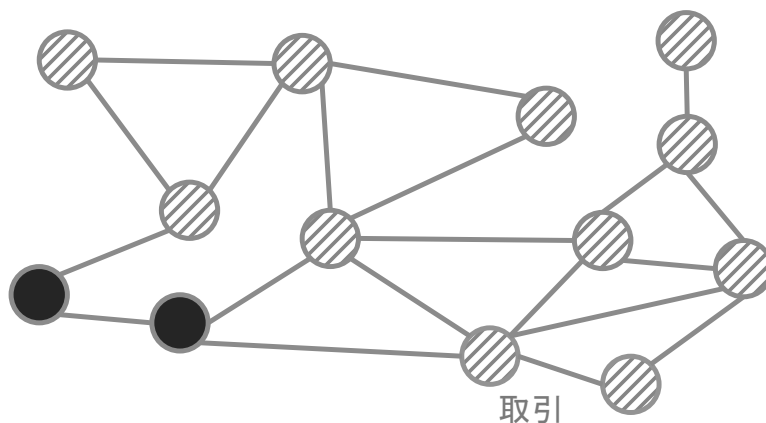


「取引」情報を受け取ったノードは

□ プルーフ・オブ・ワーク発見処理を開始！

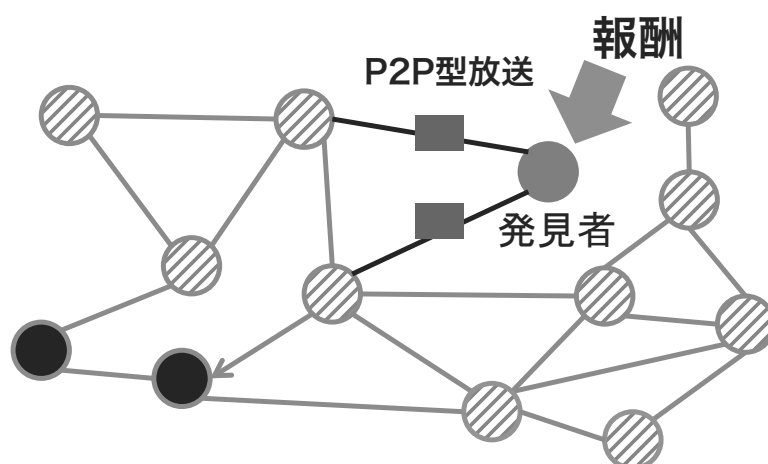
実際に計算に参加するのは、マイニングノードだけ

★10分程度で見つかるように難易度が調整される



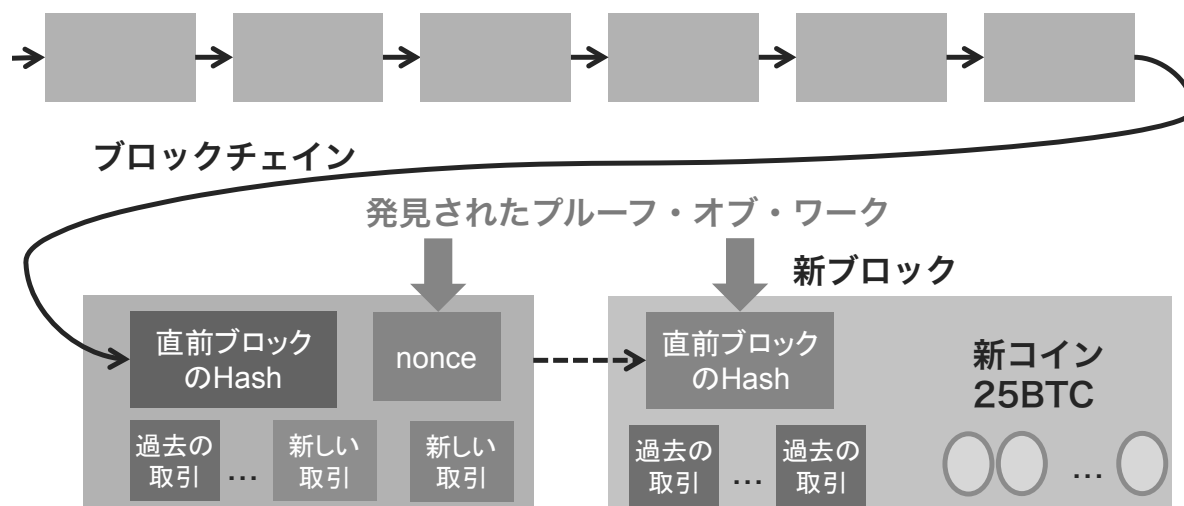
プルーフ・オブ・ワークを発見

- 「発見者」は、結果を全ノードにむけて「放送」
- 「発見者」には報酬がある



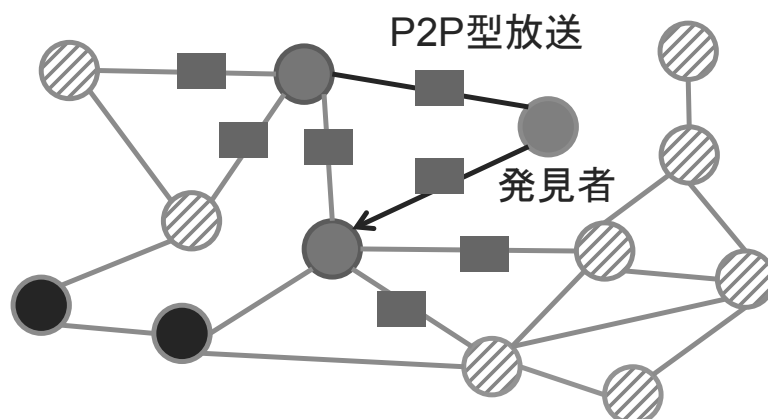
発見の報酬

- 新ブロックを生成し、新しいbitcoinを追加
- 追加したbitcoinは、発見者のものになる



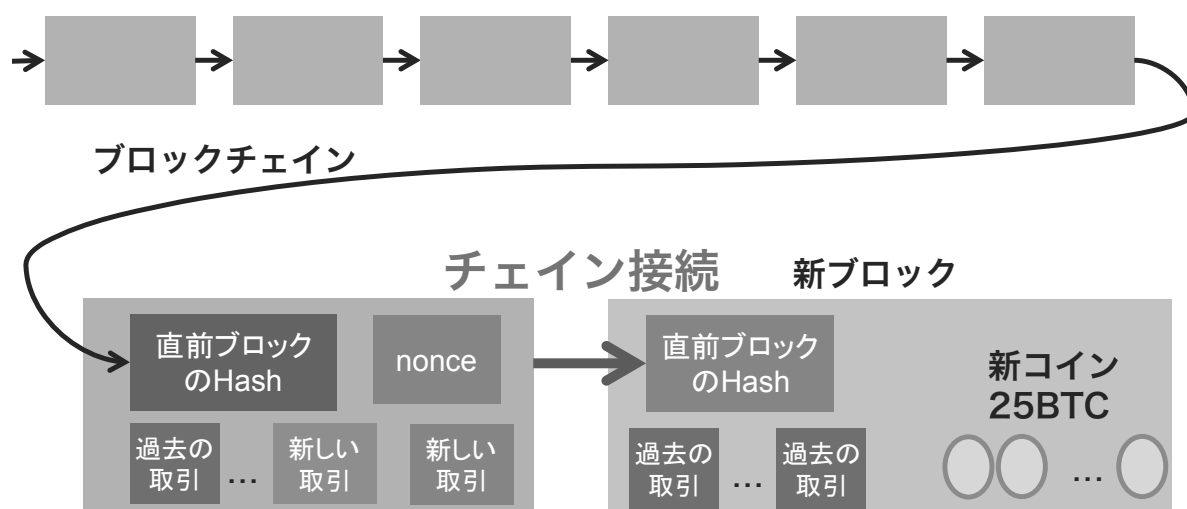
計算結果を受信したノードは

- (1) 取引ブロックの内容を検証
- (2) 検証結果が正しいとき、承認
- (3) 他のノードにパス



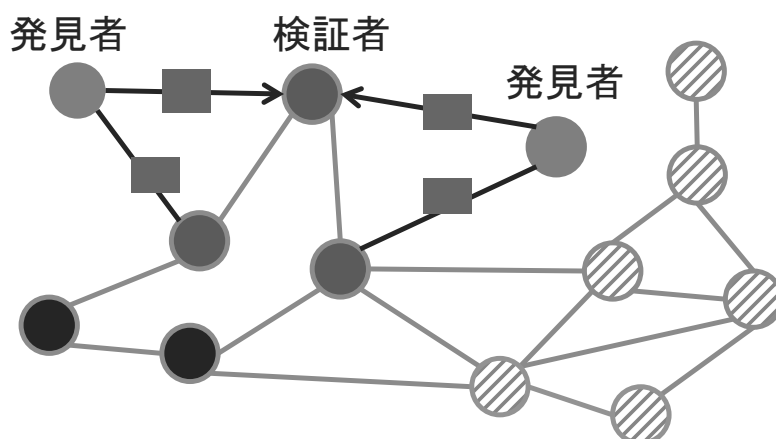
取引の承認とは

□ 新ブロックを、ブロックチェーンにつなぐこと



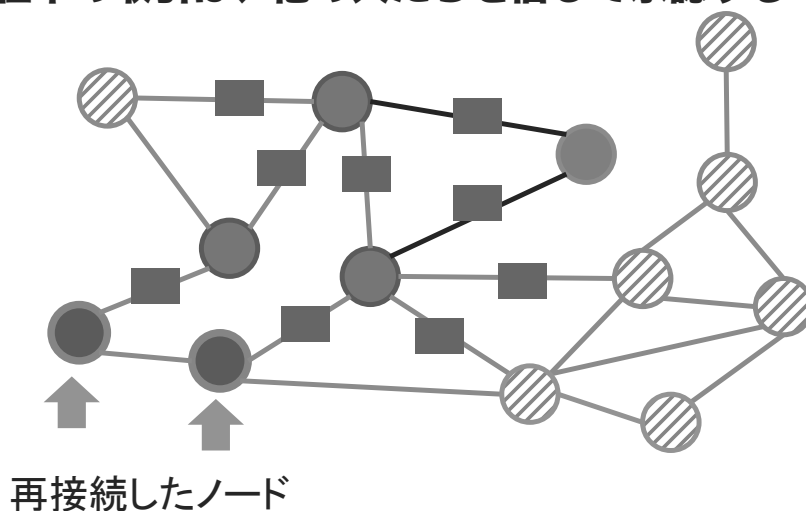
複数のノードが同時に発見した場合

- 検証者は、一方をブロックチェーンに接続するが、他方も保存
- 次のブロックが伸びたときに、長い方を選択



bitcoinのタイムスタンプ

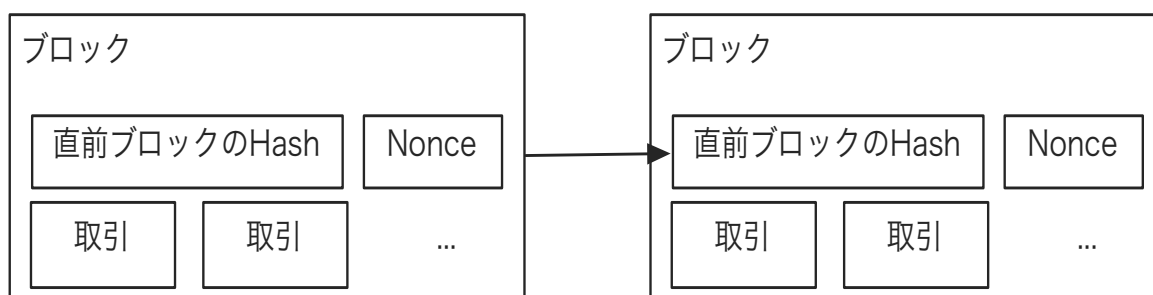
- ネットワークに繋がっていなかった人は
後で繋がったときに、最新のブロックチェーンを受け取る
不在中の取引は、他の人たちの信じて承認する



取引ブロックの内容

□ 論理的には

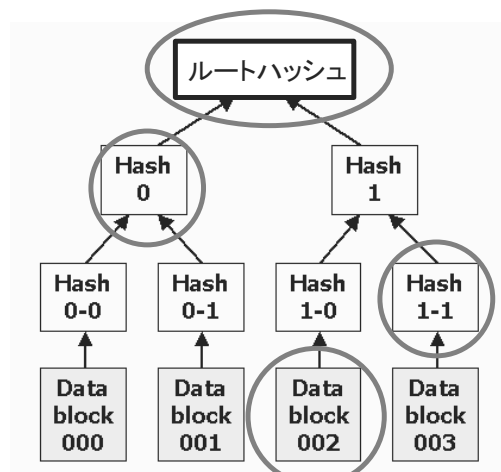
取引ブロックにはすべての取引履歴が入っています
直前の取引ブロックのHashを連鎖的につなぐことで、
含まれる取引の信頼度を上げていきます



マール・ツリー構造

□ ハッシュの連鎖の2分木構造

ルートハッシュさえあれば、部分的な情報で検証可能
多くのP2Pファイル共有システムで利用されている方法



データ002を検証したい場合

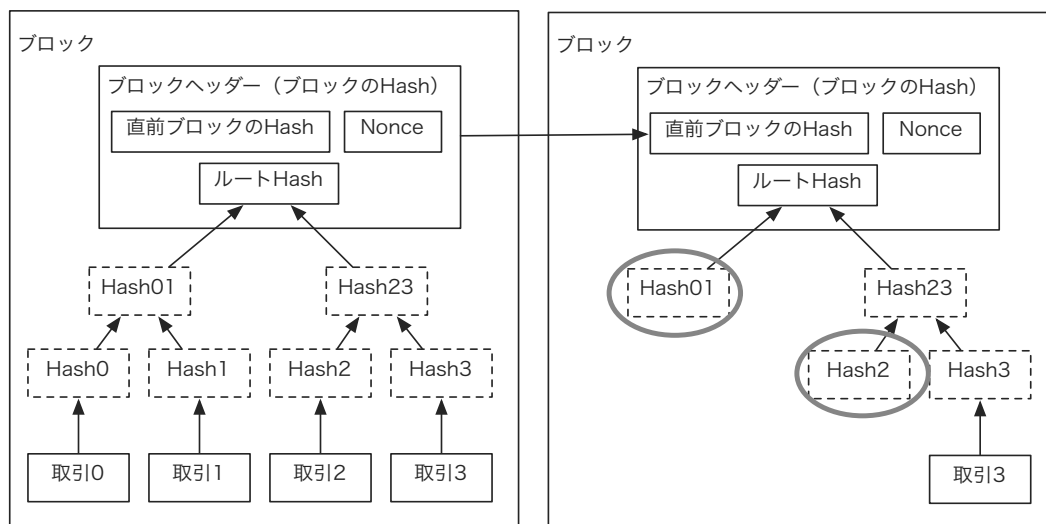
ルートハッシュ
Hash0
Hash1-1

があれば検証できる

ブロックのコンパクト化

□ ブロックヘッダーにはルートHashだけ

検証に必要なハッシュはP2Pネットワークから入手



CPU消費量による正統性証明

□ 最長のチェーンが正統なもの

実際は、長さではなく合計計算量

(攻撃者とのチェーン延長競争になる)

□ 偽造ブロックを作成するには

チェーン延長競争に勝つ必要がある

CPU消費量による正統性証明

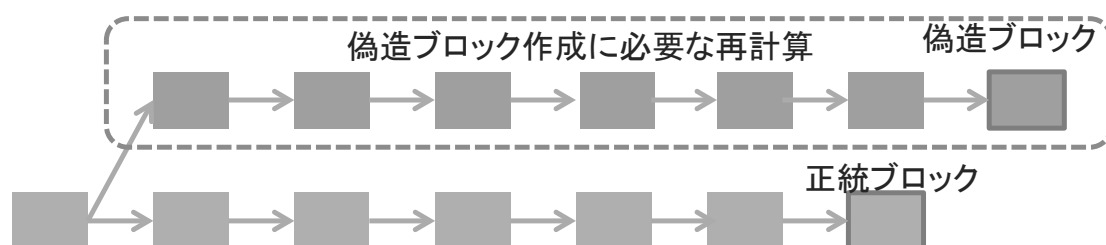
□ 支払人が「すごい計算能力」を持っている場合

受取人が、前回と同じ公開鍵を使った場合

支払人が「自分に送金」のような偽の取引を作成し

支払い直後にブロックのチェーンを伸ばす計算ができる

□ 「ネット全体」 VS 「すごい計算能力」の競争



取引のインプットとアウトプット

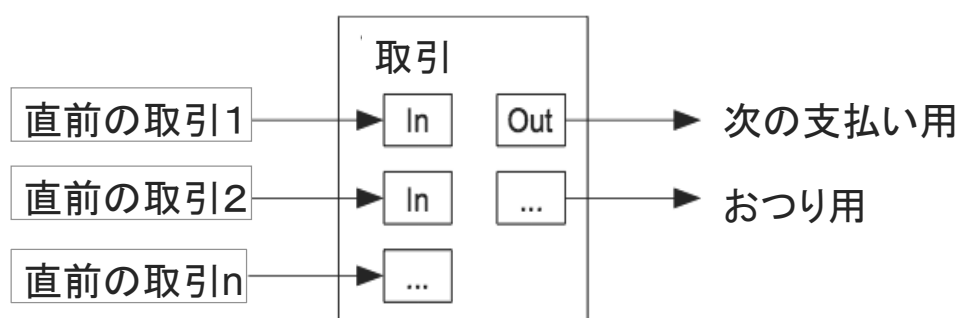
□ 取引による支払いは、コインの組み合わせ

インプット：1～n

アウトプット：1 or 2

□ 取引手数料

インプット金額 - アウトプット金額



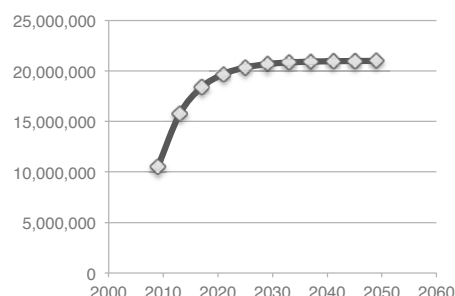
bitcoinのインセンティブ

□ ビットコインの通貨総量

最初の4年で10,500,000 BTC

約4年ごとに半減させて追加

10分あたりの生成数を増減し調整



□ 新ブロック発見時のコイン生成

最初の4年 $10500000\text{BTC} / 210000\text{BLK} = 50\text{BTC}$

次の約4年 $5250000\text{BTC} / 210000\text{BLK} = 25\text{BTC}$

□ 取引手数料報酬

新ブロック発見者は、それに組込む取引から手数料を得ることができる

手数料は任意だが、少ないと次ブロックに組み込まれないかもしれない

流通量が一定量を超えると（約4年）手数料を受け取ることができる

bitcoinのプライバシー

□ すべての取引は「公に」なっている

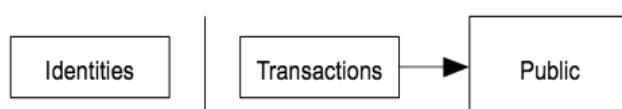
□ 公開鍵と本人のリンク可能性は消せる

取引のたびに鍵ペアを新規に生成すればより強力

Traditional Privacy Model



New Privacy Model



bitcoinへの疑問

□ bitcoinに関連する法的枠組み

資金決済法、出資法、銀行法などなどとの関係

「決済手段」「決済方法」

P2Pネットワークのノード全員で決済行為をしている～

□ bitcoinで安定的にビジネスは可能か？

法制度が未整備な現状と将来

欧米や世界の動向、事例

bitcoin

勉強会3

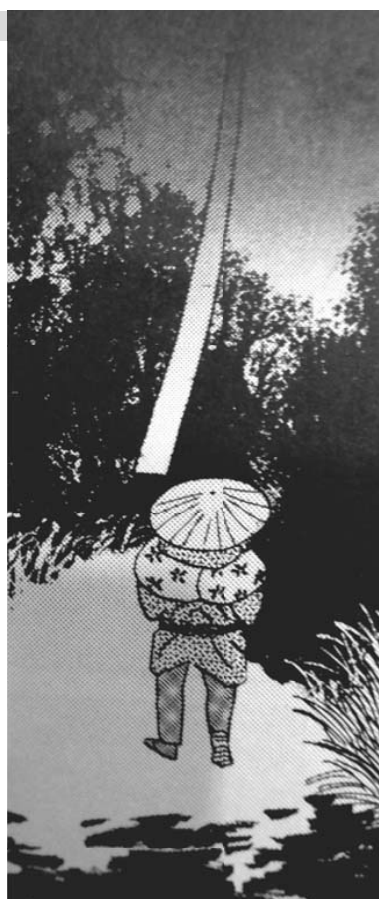
近畿大学
山崎重一郎

ビットコイン 一反もめん説

国立情報学研究所
岡田仁志先生より

注) この絵に描かれて
いる物語では一反
もめんではなく貉の
しわざの一つです

杉浦日向子著
百物語より抜粋

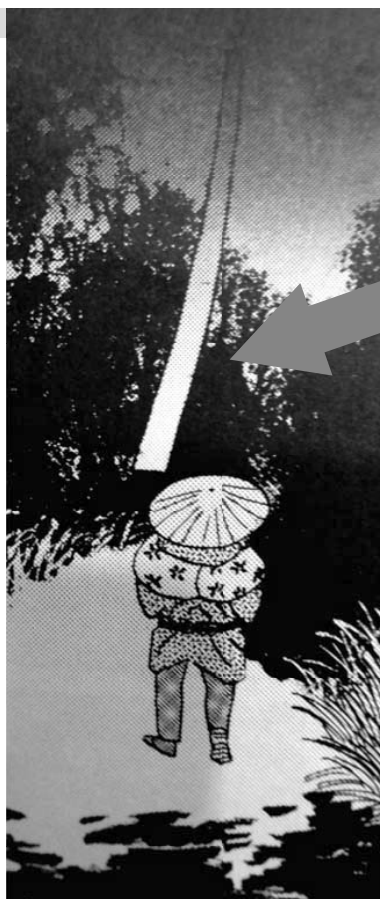


大福帳を管理する
銀行みたいなものは
ない

見える人だけに見える
大福帳

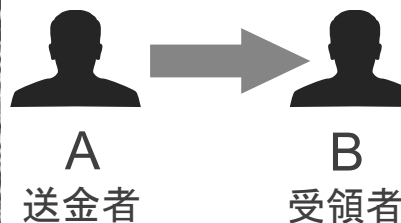


杉浦日向子著
百物語より抜粋



取引の記録

AがBに50BTC
支払った



杉浦日向子著
百物語より抜粋



放送は実際
はリレーで実行

送金者が
取引の記録
を放送

杉浦日向子著
百物語より抜粋

A
送金者

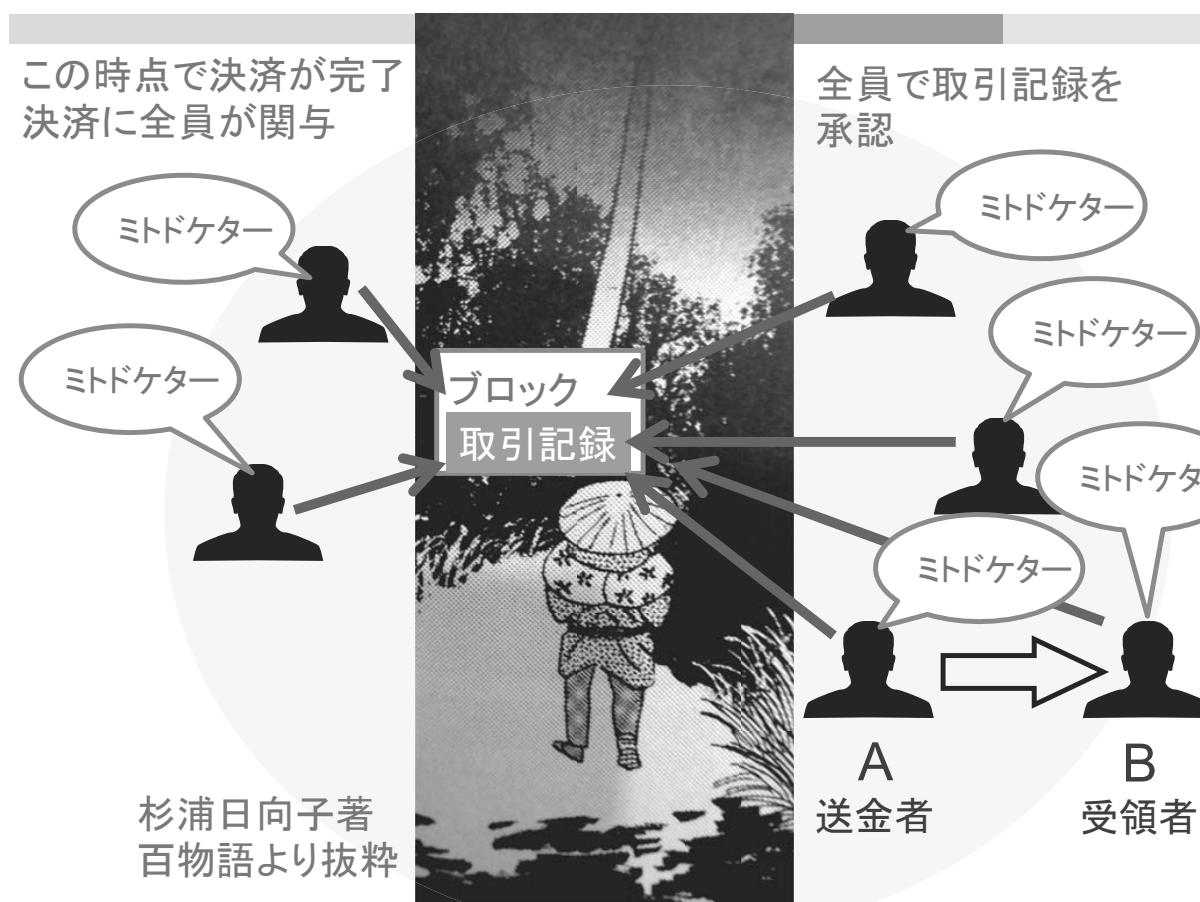
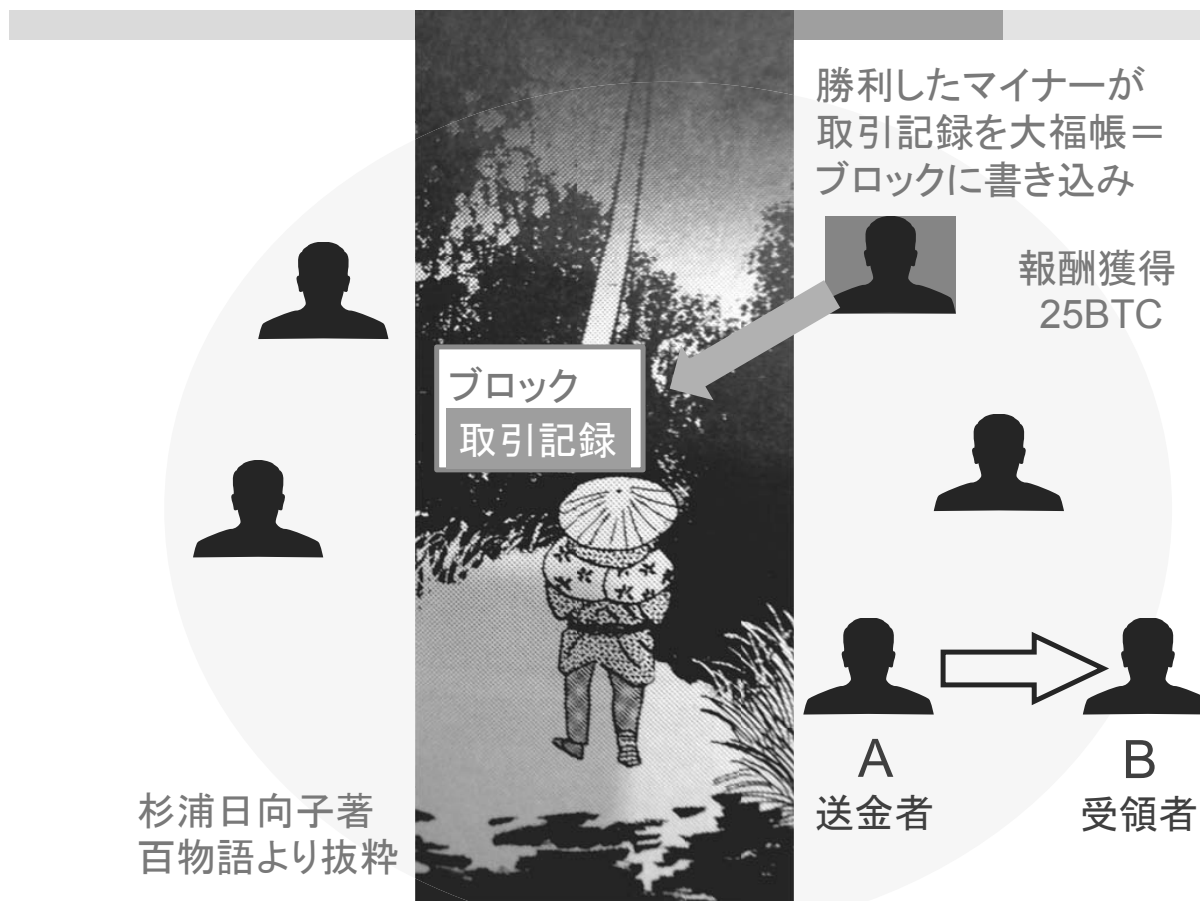
B
受領者

杉浦日向子著
百物語より抜粋

A
送金者

B
受領者

マイナーが
計算競争



一反もめん＝大福帳
はP2P型分散ファイル
システム

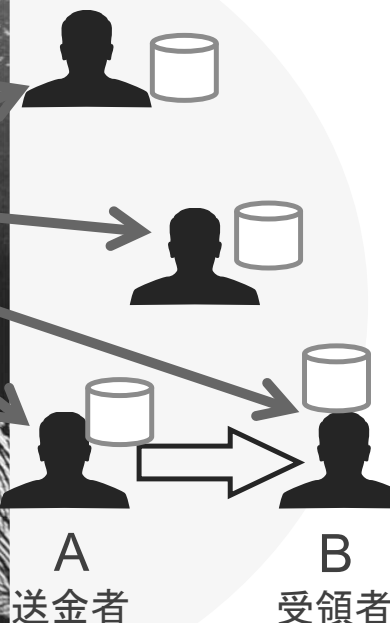
銀行はない



杉浦日向子著
百物語より抜粋



全員で取引記録を
保存



ビザンティン将軍問題

□裏切り者がいるネットワークで合意する

多数決をとりたいが、失敗させられてしまう



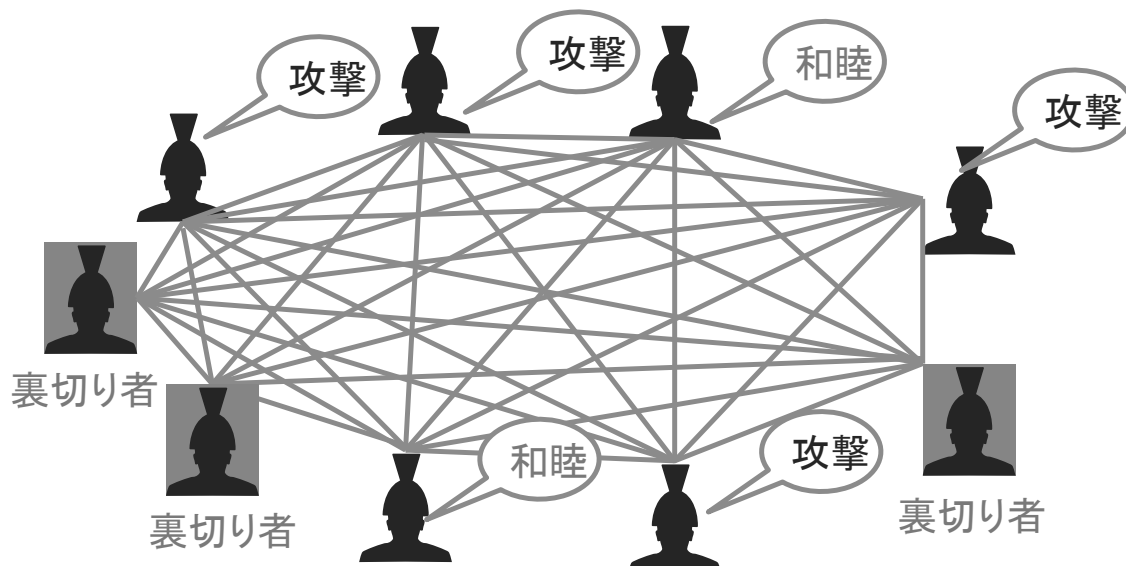
古典的解：情報交換を繰り返し、認知レベルを上げる

第1段階、相互の意見の交換

第2段階、自分が知った他の将軍たちの意見（意見ベクトル）の交換

第3段階、...（意見ベクトルのベクトル）の交換

1/3以上裏切り者がいると解が無い

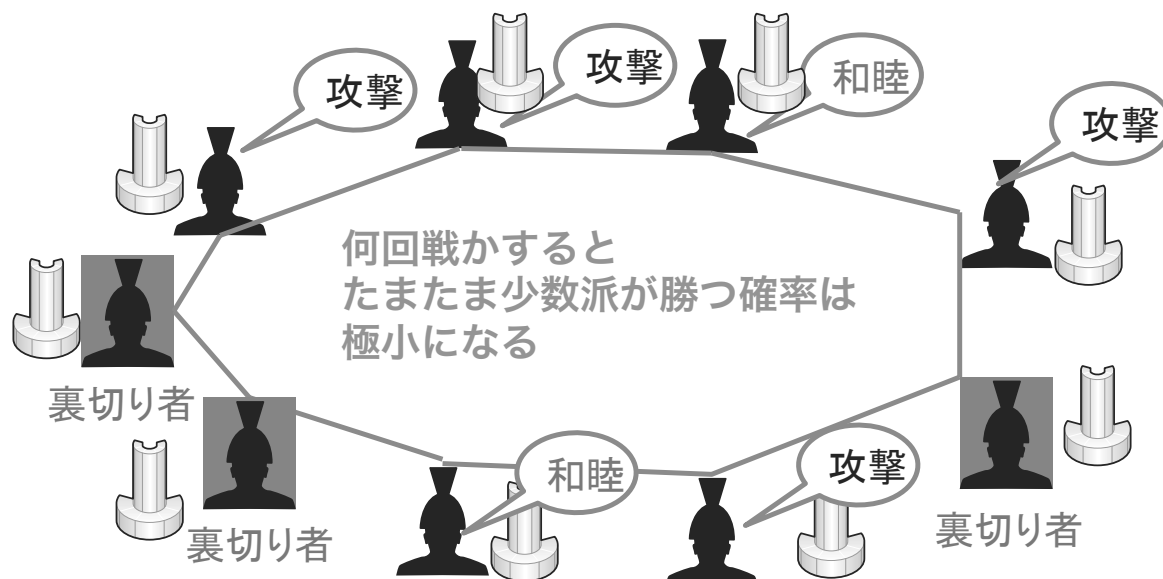


サトシ・ナカモトの解

ネット上の計算資源の総量が有限

「攻撃」チーム VS 「和睦」チーム

計算競争をして勝った方の意見を採用！ → 人間の欲望を利用



プルーフ・オブ・ワークと 計算難易度の調整

□ 難しい計算をした証拠

ハッシュキャッシュという暗号学的に無意味な計算

□ 2016ブロックの計算ごとに難易度を調整

$2016 \times 10 \text{分} = 2 \text{週間}$

計算完了が2週間より早ければ→難しくする

計算完了が2週間より遅ければ→簡単にする

難易度の変遷

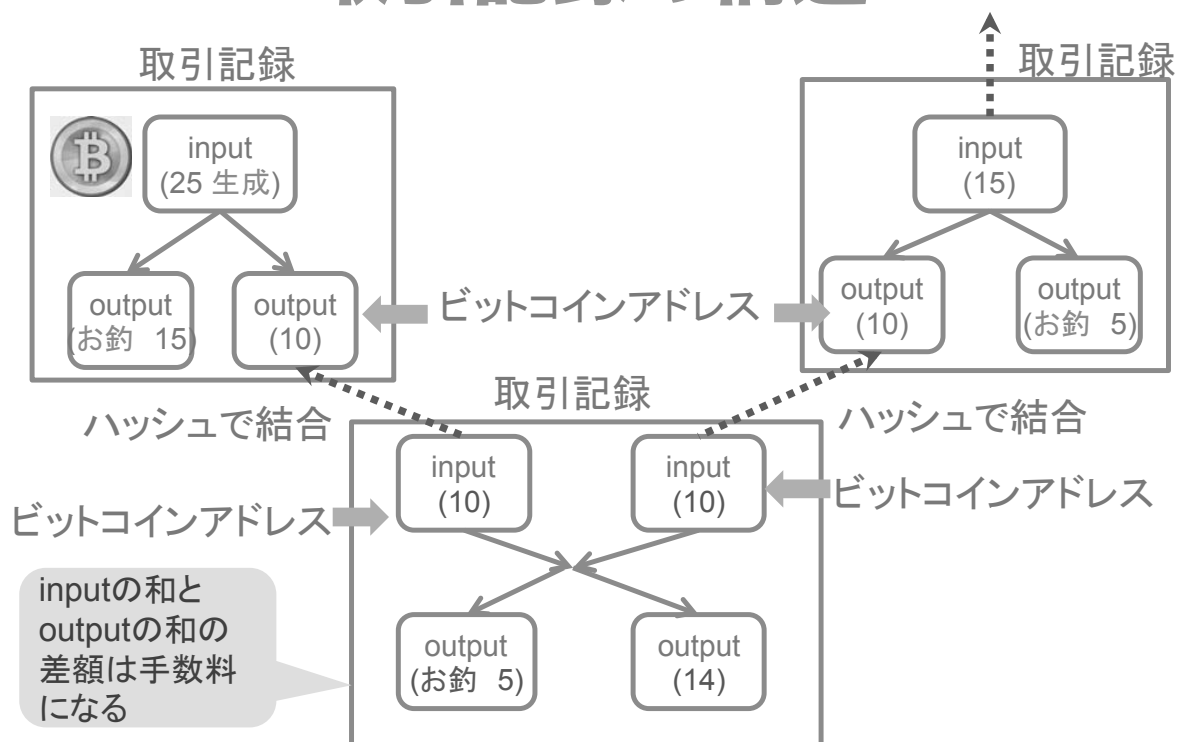


ビットコイン



ライトコイン

取引記録の構造



取引記録のデータ構造

Input: インプット

Previous tx:

f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6

Index: 0

scriptSig: 電子署名と公開鍵

304502206e21798a42fae0e854281abd38bacd1aead3ee3738d9e1446618c4571d10

90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output: アウトプット

Value: 5000000000 支払い金額 (単位はsatoshi = 1億分の1BTC)

scriptPubKey: OP_DUP OP_HASH160

404371705fa9bd789a2fcd52d2c580b65d35549d

OP_EQUALVERIFY OP_CHECKSIG ← 検証スクリプト

取引記録のデータ構造

□ インプット

直前の取引のハッシュ値：直前の取引とのリンクをつくる

インデックス： 直前の取引のアウトプットの識別

□ 電子署名と公開鍵

公開鍵：この取引による送金者の公開鍵

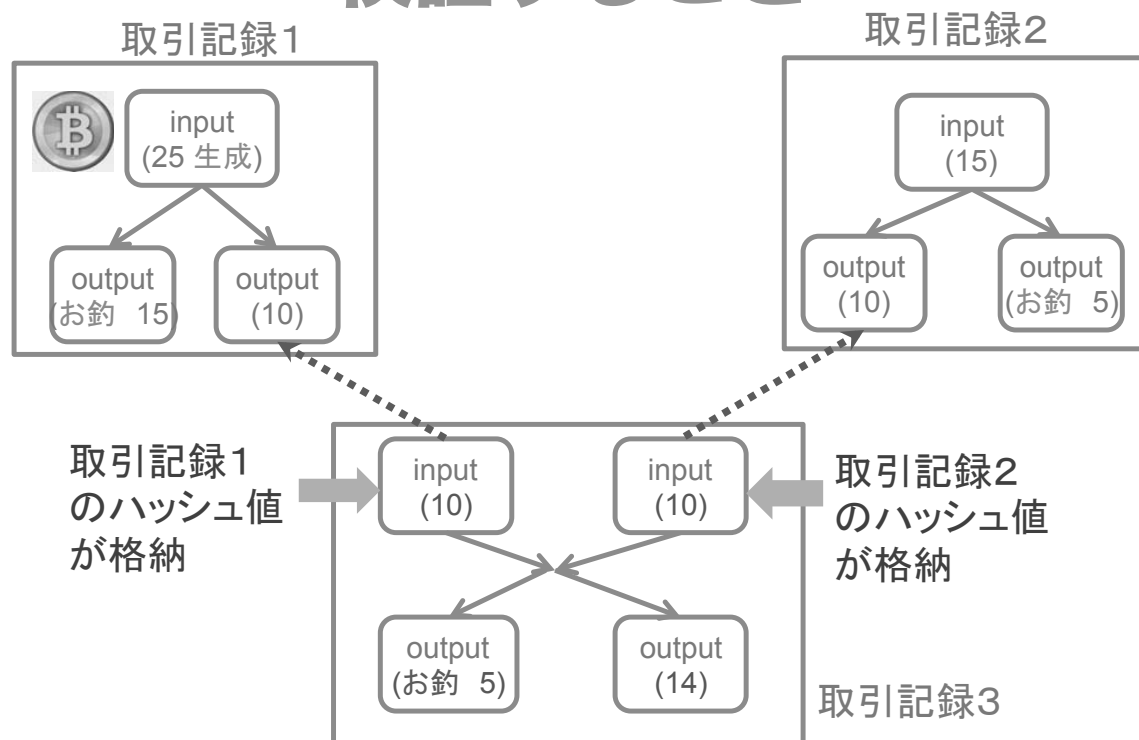
電子署名：この取引記録を単純化形式へのハッシュへの署名

□ アウトプット

バリュー：支払い金額

scriptPubKey:検証スクリプトの後半部分

検証すること



検証とスクリプト

□ 検証すること

各インプットがそれぞれアウトプットの対応
ハッシュの一致、電子署名の正しさ

- BitcoinはForth言語的スクリプト言語が利用可能
- インプットを直前のアウトプットの連続性の検証
- 各インプットが参照している直前のアウトプットを検証する

bitcoinへの攻撃方法

□ Transaction malleability (Dos攻撃)

署名可塑性、スクリプト可塑性 Mt.Goxへの攻撃？

□ 51%攻撃（コインの多重使用）

マイニングプールの50%以上を支配できる場合の攻撃

□ Race攻撃（コインの多重使用）

店は10分待たないという前提。送金者が、同一のコインで、代金支払と自分自身への送金の二つを同時に放送する。

□ Finny攻撃（コインの多重使用）

店は10分待たないという前提。マイナー自身が支払者で、こっそり自分への支払い取引をブロックに入れる

□ ベクター76攻撃（コインの多重使用）

Race攻撃とFinny攻撃の組み合わせ

電子署名やハッシュ計算の大前提

□ ハッシュ関数

乱数を発生する関数

入力が同一なら、対応する出力も同一

しかし、入力が1ビットでも違うと、出力は全く違う

□ 要件

署名対象は、通信過程で1ビットも変更されてはならない

経由するマシン、OS、プロトコルなどに依存しない

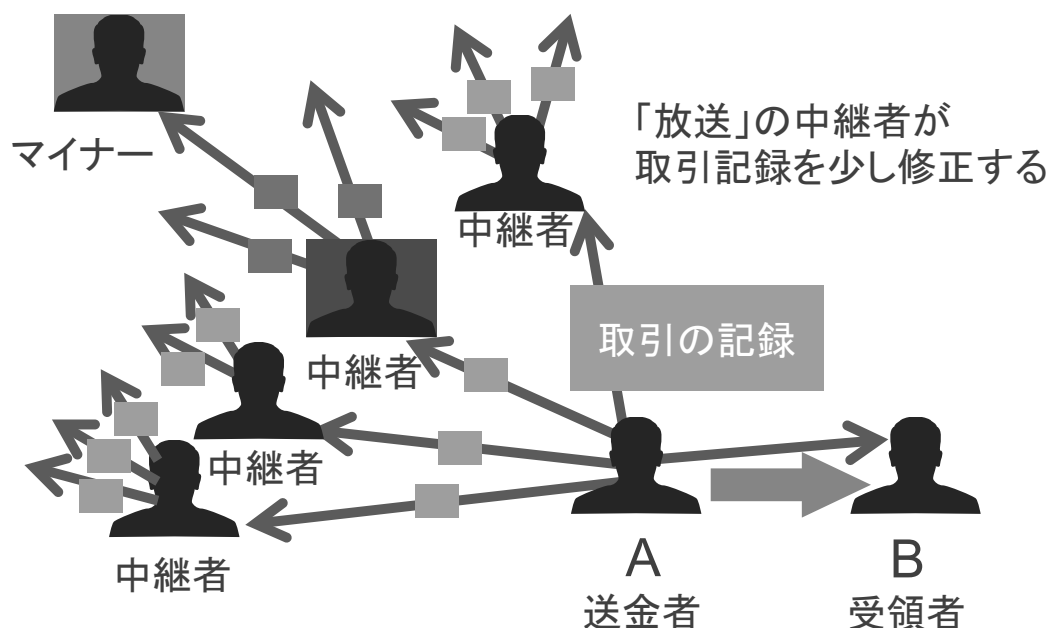
□ 実現技術

ASN.1：抽象構文

DER: エンコード方法 (型、長さ、値)

Transaction malleability

署名展性、スクリプト展性



署名展性

- 電子署名検証は成功状態のまま取引データを改ざんする方法
ecdsaの署名 = (r, s) という整数の組
しかし、 $(r, -s \pmod n)$ でも署名検証は成功する
- その結果
取引のハッシュが違うので、ブロックに組み込まれ、承認されると
正統な取引の連鎖が途絶え、そのコインがそれ以降使えなくなる

スクリプト展性

- 取引には Forth言語に似たスクリプト言語が記述されている
- スクリプト部分は署名対象の外部
- スクリプトを変更できてしまう

典型的対処方法

- 悪意のリレーノードを見つけるのは容易
($r, -s \pmod n$) を出してきたノード見つけばいい
- 接続する相手を選ぶ
交換所のような有名な送金者ノードは、中継ノードをなくして、直接信頼できる有力なマイナーと接続すればよい

opensslのECDSA署名を試す

ビットコインの電子署名は、ECDSA Secp256k1 を利用しています。

- openssl version 1.0.1e以上を使う
 - プライベート鍵の生成(ランダムに生成されます)
openssl ecparam -genkey -name secp256k1 -out privkey.pem
 - プライベート鍵から公開鍵の生成
openssl ec -in privkey.pem -pubout -out pubkey.pem
- Ruby opensslを利用 (require 'openssl')
 - ECDSA署名

```
skey = OpenSSL::PKey::EC.new(File.read('./privkey.pem'))
hash = OpenSSL::Digest::SHA256.digest('abc12345')
sign = skey.dsa_sign_asn1(hash)
```
 - ECDSAの署名検証

```
pkey = OpenSSL::PKey::EC.new(File.read('./pubkey.pem'))
pkey.dsa_verify_asn1(hash, sign)
```


Bitcoinの匿名性について

□ anonymityとpseudonymity

匿名（意図的）と仮名（ID切り離し）で明確に区別される

□ ビットコインアドレスは仮名

全ての取引記録が、ビットコインアドレスで追跡可能

例：Mt.Goxのビットコインアドレスに匿名性はない

□ ビットコインを匿名化するサービスも存在する

ミクシングサービスを経由させると追跡が難しくなる

bitcoin

勉強会4

近畿大学
山崎重一郎

概要

- 今後のBitcoinの制度設計に重要な課題のひとつ
Bitcoinのアイデンティティとプライバシー
- Bitcoinにおける匿名化技術
Mixing service技術の解説
- Bitcoin取引履歴の追跡方法
ブロック・チェーンの中身を調査する
- Bitcoinの技術の視点からの規制のあり方
取引所、ミクシングサービス、採掘業者への課題

電子貨幣の「ギュゲスの指輪」

□ プラトンの著書「国家」

透明人間による、知られることのない悪事の物語

秩序ある活発な経済と個人の尊厳

□ ネットの登場以来繰り返された議論

匿名発言、通信履歴、位置情報、人間関係

□ 仮想通貨のアイデンティティとプライバシー

本人確認

国家、超国家企業（仮想通貨のGoogle, facebookが登場？）

仮名の世界での秩序形成

Bitcoinのアイデンティティとプライバシー

□ ブロックチェーンに全取引が存在し全員が確認可能

□ 取引＝「誰が誰にいくら払った」という記録

□ 誰＝Bitcoinアドレス

仮名（匿名ではない）アイデンティティの隠蔽を目的としない

□ しかし取引とBitcoinアドレスを匿名化技術は存在

mixing service

□ 人権としてのプライバシー

「ほうっておいてくれ」という権利

「私の情報にアクセスするのなら、私の承諾を得てくれ」という権利

「私が主観的に嫌だを感じる機微情報に触れないで」 などなど...

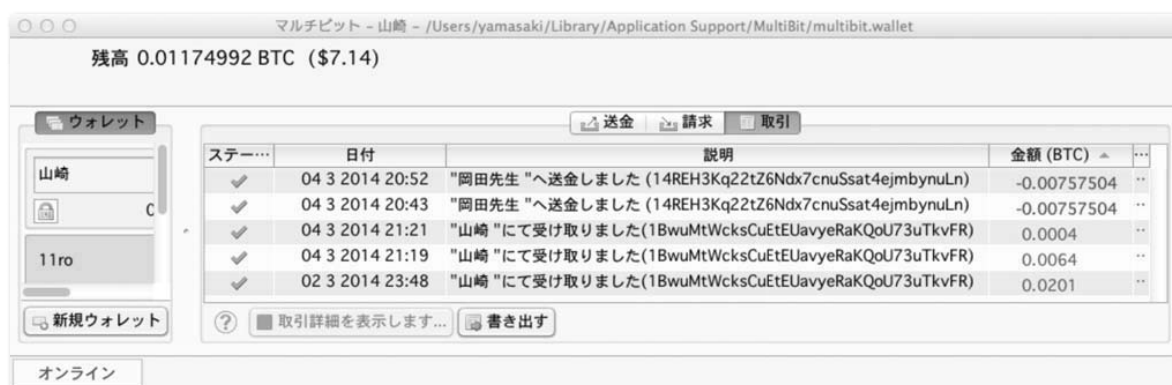
Bitcoinアドレス

□ 支払い先を識別するアドレス

公開鍵のハッシュ値をbase58エンコードしたもの

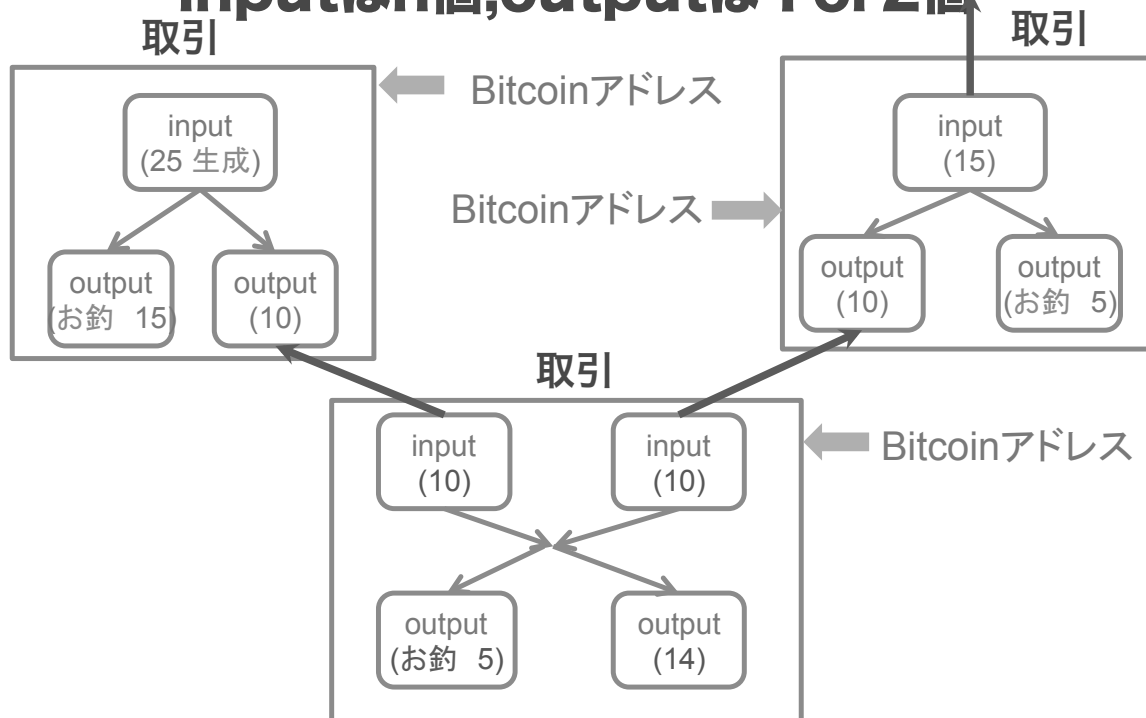
□ 取引=バランスシートの履歴を集約する鍵

□ Bitcoinウォレット

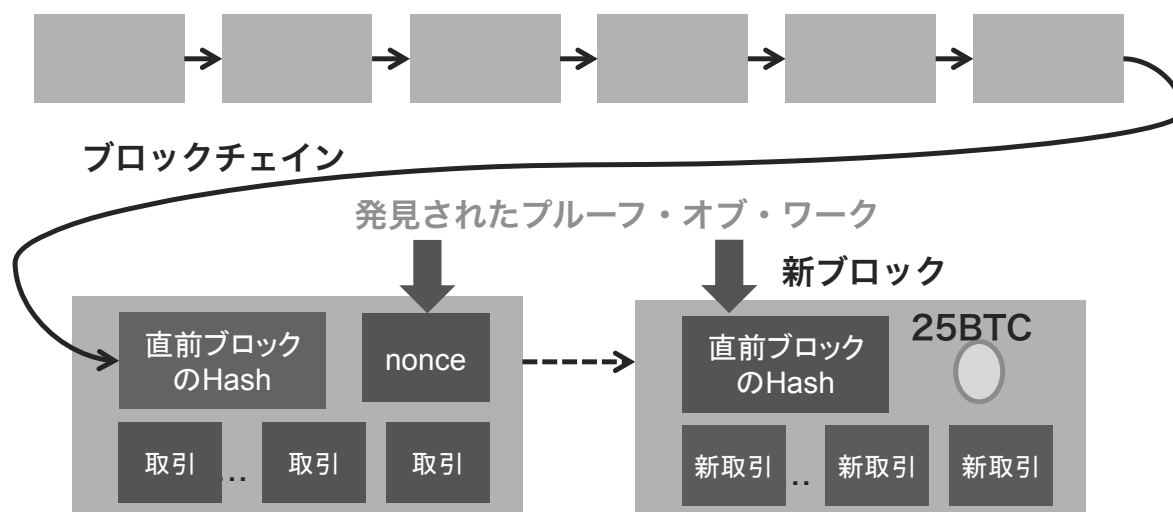


取引のネットワーク

inputはn個,outputは1 or 2個



Bitcoinのブロック・チェーンには 過去の全ての取引の記録が入っている



取引グラフの例

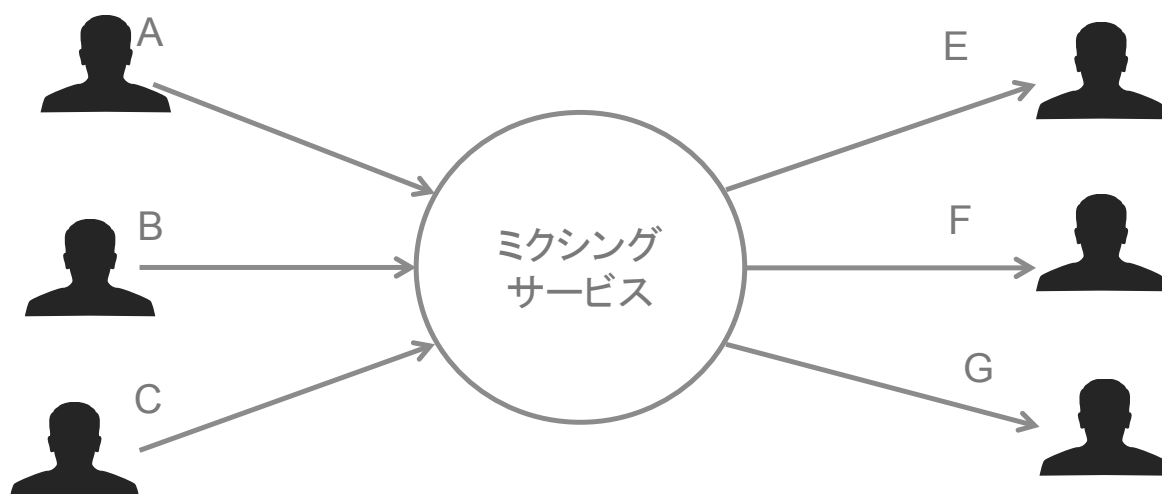
● は取引（バランスシート）
それぞれのinputとoutputが連鎖接続



mixing service

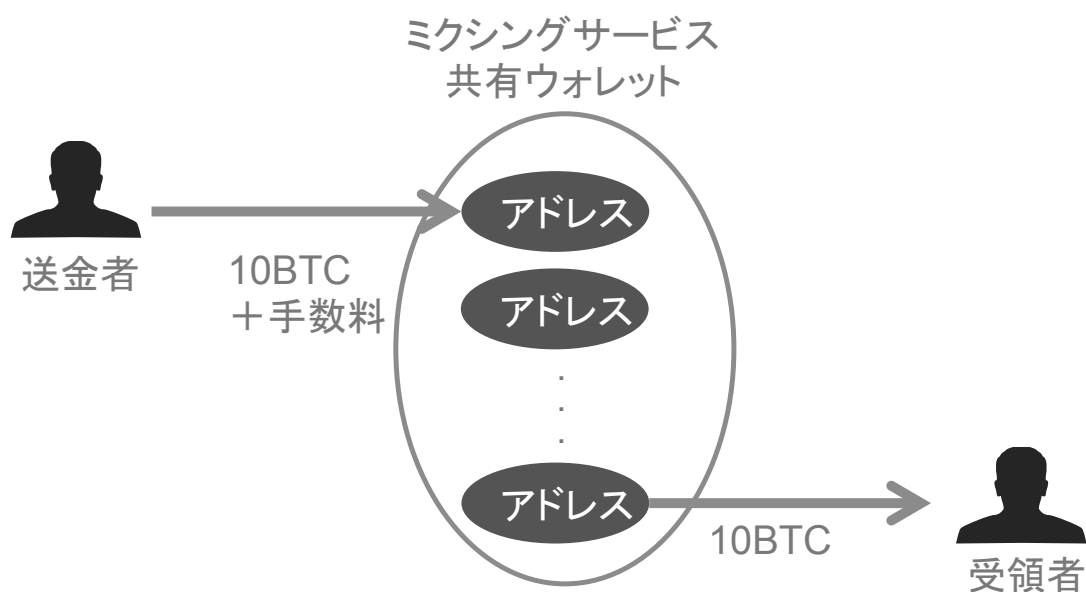
□ビットコインの取引の流れを匿名化する

仮名から仮名への取引の流れを攪乱する



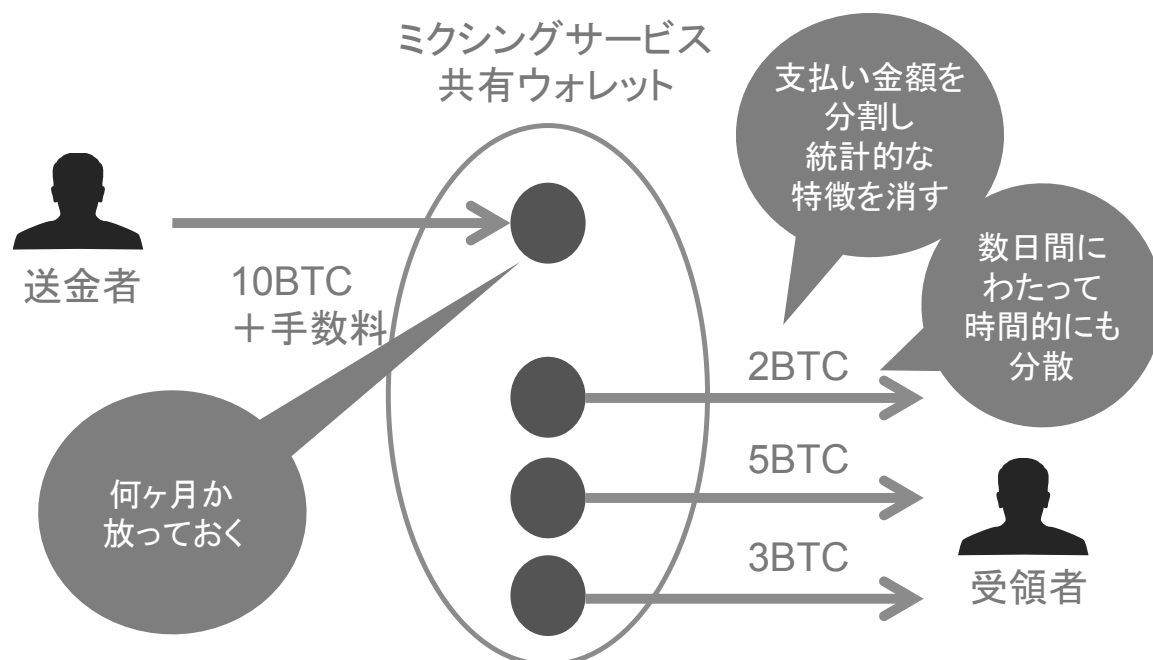
mixing serviceの原理

取引の連鎖を追跡しにくくする



mixing serviceの原理2

プロファイリングされにくくする



mixing serviceの例

Malte Möser氏の論文
Anonymity of Bitcoin Transactions より

	アドレス数	ウォレット	手数料	所要時間	Tor	エスクロー
OnionBC		有り	3%		使用	有り
Bitcoin Fog	5	有り	1~3%	6~96時間	使用	
Bit Laundry	1	無し	2.49%+ α	1~10日		
Blockchain.info	-	有り	0.5%	なし		

mixing serviceの弱点

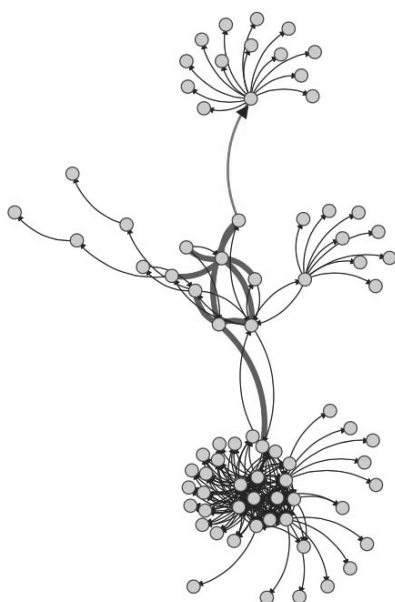
- **mixing serviceの運営主体が信用できない**
- **mixing serviceには取引情報が残る**
- **mixing service自身は扱い額の2倍以上の大量のコイン保有が必要**
- **mixing serviceのアドレスを経由したコインは「汚染されている」とみなされ、市場価値が下がる**

ブロックチェーンの追跡方法

- **ブロックチェーンは簡単に検索可能**
<https://blockchain.info> のAPIを利用
- **ノード情報の取得**
<http://blockchain.info/rawaddr/Bitcoinアドレス>

flexcoin社のコイン盗難経路

2014年3月2日にこのアドレスに盗まれました



flexcoin
the bitcoin bank
FAQ
Bitcoin Bank
Register
Activat
Login

Flexcoin is shutting down. (March 3 2014)

On March 2nd 2014 Flexcoin was attacked and robbed of all coins in the hot wallet. The attacker made off with 896 BTC, dividing them into these two addresses:

```
1NDuavq4SHYFEmqCDBS7DLMNvgdu
1QFvCSH9epKqT0d9QH6eGn56dCHgy6
```

As Flexcoin does not have the resources, assets, or otherwise to come back from this loss, we are closing our doors immediately.

Users who put their coins into cold storage will be contacted by Flexcoin and asked to verify their identity. Once identified, cold storage coins will be transferred out free of charge. Cold storage coins were held offline and not within reach of the attacker. All other users will be directed to Flexcoin's "Terms of service" located at "Flexcoin.com/terms" a document which was agreed on, upon signing up with Flexcoin.

Flexcoin will attempt to work with law enforcement to trace the source of the hack.

Updates will be posted on [twitter](#) as soon as they become available.

Update (March 4 2014)

During the investigation into stolen funds we have determined that the extent of the theft was enabled by a flaw within the front-end.

The attacker logged into the flexcoin front end from IP address 207.12.85.117 under a newly created username and deposited to address 1D8D9Bv52wG2ZAw29q27M97Ape3Ly

The coins were then left to sit until they had reached 6 confirmations.

The attacker then successfully exploited a flaw in the code which allows transfer between flexcoin users. By sending thousands of simultaneous requests, the attacker was able to "move" coins from one user account to another until the sending account was overdrawn, before balances were updated.

This was then repeated through multiple accounts, snowballing the amount, until the attacker withdrew the coins. ([Here and Here](#))

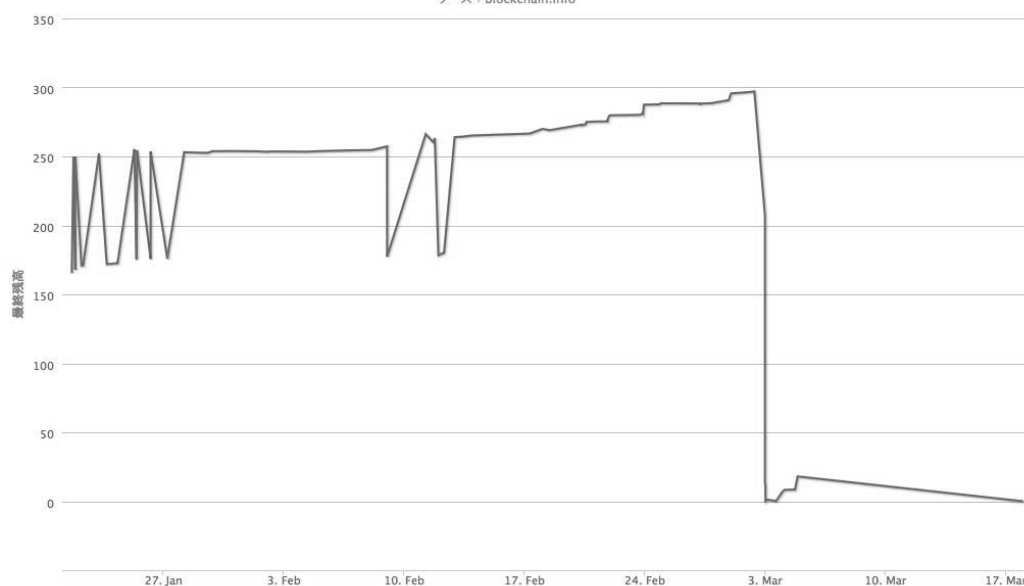
Flexcoin has made every attempt to keep our servers as secure as possible, including regular testing. In our ~3 years of existence we have successfully repelled thousands of attacks. But in the end, this was simply not enough.

Having this be the demise of our small company, after the endless hours of work we've put in, was never our intent. We've failed our customers, our business, and ultimately the Bitcoin community.

Please direct any and all questions to admin@flexcoin.com and we will reply to you as soon as possible.

カナダflexcoin社の残高

ビットコインアドレスの残高: 16Ln8mBGUNXo1mnK4hiZhEhsEaWkbWdcPd
ソース: blockchain.info



カナダflexcoin社の盗難取引

サマリー		Transactions	
Address	1GEhfbjbUaCVYBa6Av6zwRqnwkrq9vNozE	取引件数	2
Hash 160	a7213d77b6ae0ef1aed3cd6427176e50a49810a0d	Total Received	540.17952145 BTC
Tools	Taint Analysis - Related Tags - Unspent Outputs	最終残高	0 BTC

支払いのリクエスト Donation Button



Transactions (Oldest First) Filter

a1b887233c06490fbdeb2c8779fd47e1f93a68d16928766d45879dcfc39571e2 (手数料: 0.0001 BTC - Size: 226 bytes) 2014-03-02 23:34:32

1GEhfbjbUaCVYBa6Av6zwRqnwkrq9vNozE (540.17952145 BTC - Output) →

1AXxRoMHRbbk43doiTmuTiouprHtRTATi - (使用済み)	40.17942145 BTC
1NDkevapt4SWYFEmquCDBSf7DLMTNVggdu - (使用済み)	500 BTC

-540.17952145 BTC

Mt.Goxのコイン盗難事件?

□ 850K BTCくらい盗まれたらしい

でも、Bitcoinアドレスを公表していないので
本当かどうか確認する方法がありません。

2014/3/20に、昔の200K BTCが見つかった?

□ 峰松さんのご協力を得て調査してみた

峰松さんがMt.Goxにアクセスしたアドレスから6ホップ

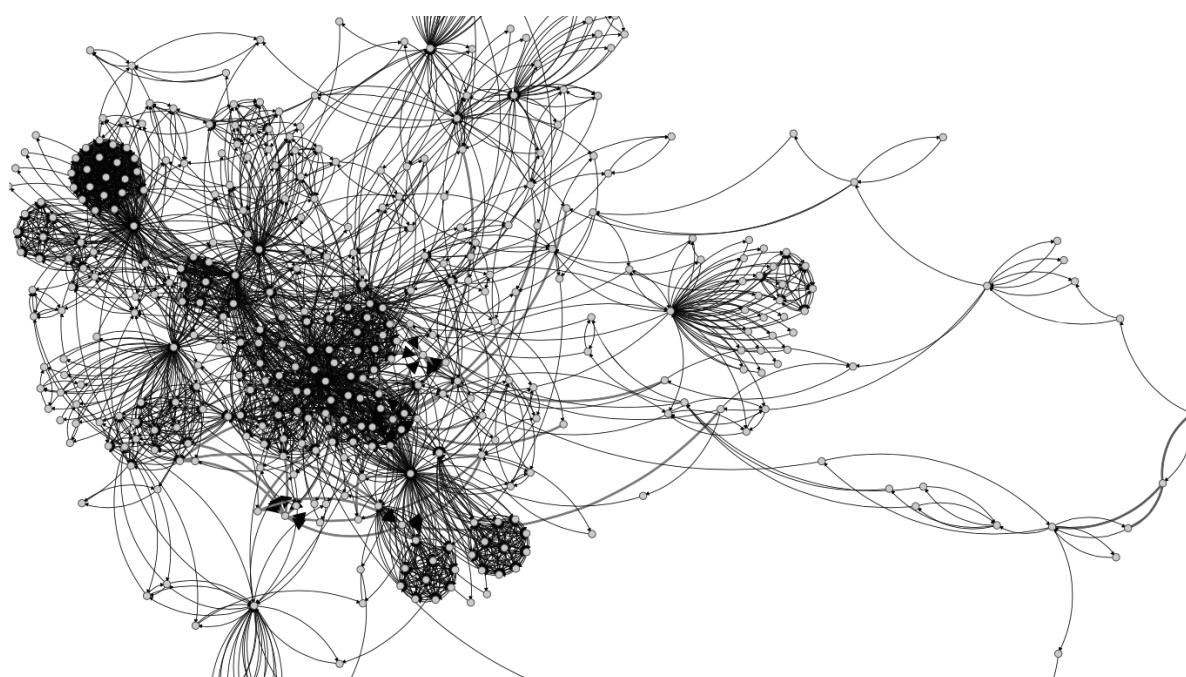
2013年7月28日以降

10BTC以上の流れのみを追跡

Mt.Gox -> 6ホップの トランザクショングラフ (6282ノード)



注目ノードからMt.Goxへの逆経路のみに枝刈り

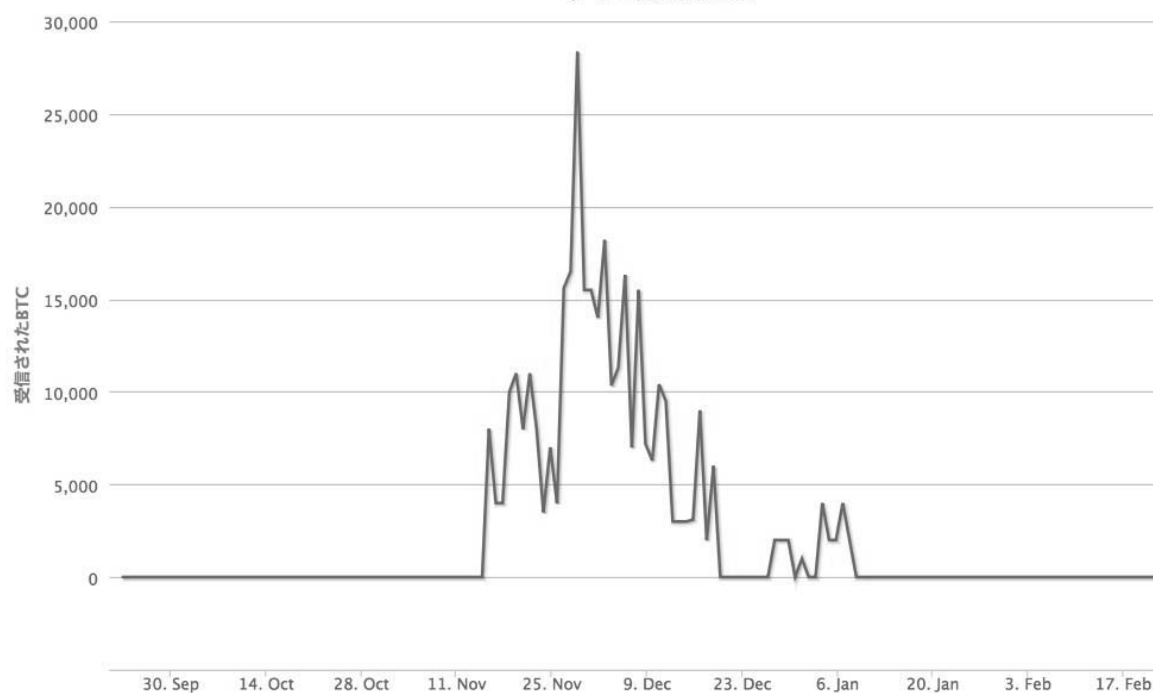


アドレス1の入金状況 合計27万7千BTC

ソース：blockchain.info



アドレス2の入金状況 合計34万6千BTC



詳細な追跡結果

経路 1

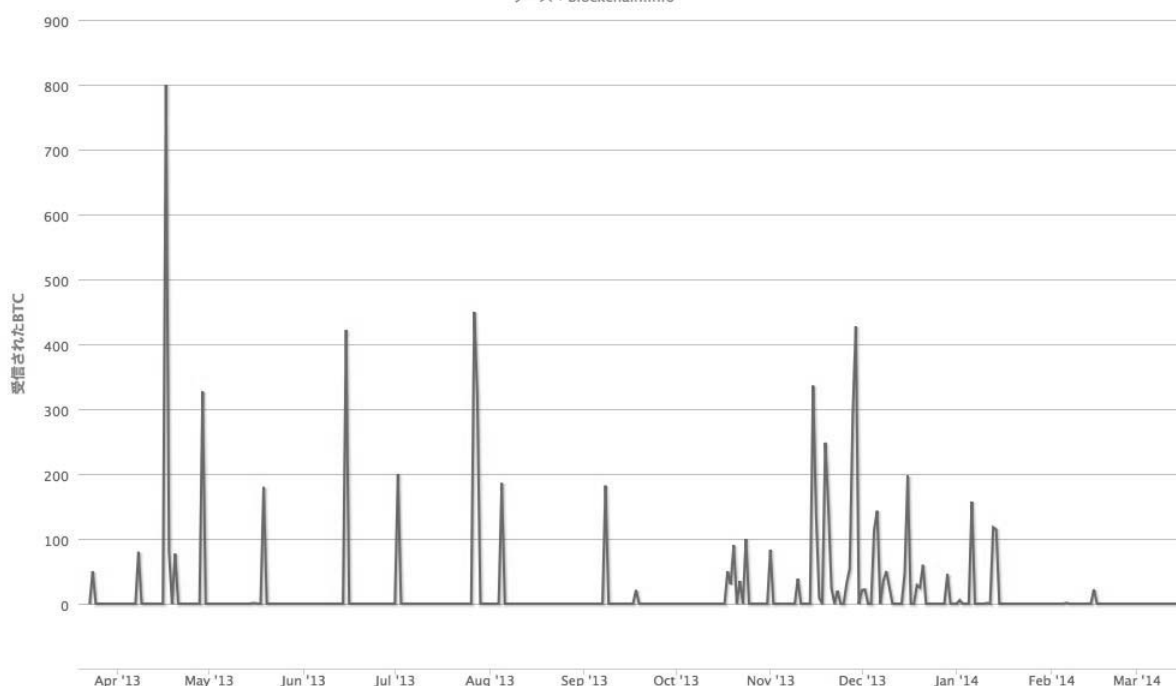
1F--8QnikfPUoo8WVFnyai3e1Hcov9y8T<- 277K BTC 2014-02-11,13 頻繁な入出金
 1E--xLjaY5mXr9V2oW4nDRpH4A7V1kSs4H<- 7K BTC 2014-02-17, 2014-02-15
 1E--A3nezmNSF3ffYFmXqTYB1VaW4kWGk4<- 20K BTC 2013-12-18
 1A--NMM1twUKBtNhGGmDidh1JHja9DRzmk<- 205 BTC 2014-03-03
 1P--VYNndik7r4wzVxoMghdwsLvtzTEEt1<- 38 BTC 2013-08-14
 16--RxtrdGhb6hmJXsjeJWkwWPYg6FNBUU<- 58 BTC 2013 08-14
 1H--a9mUXG128suKf6vNkNEgEAeaozCkuB<- Mt.Gox

経路 2

1N--642sbgCMbNtXFajz9XDACDFnFzdXzV<- 117K BTC 2014 3-10 頻繁な入出金
 1M--n87DdbkY3Grg3G5WBzvWCQcPDTToTR8<- 1K BTC 2014 3-20
 1M--a5ePQrpG6YM1dLoUjadgEvsGyH5r6p<- 32K BTC 頻繁な入出金
 1A--NMM1twUKBtNhGGmDidh1JHja9DRzmk<- 205 BTC 2014-3-3
 1P--VYNndik7r4wzVxoMghdwsLvtzTEEt1<- 38 BTC 2013-8-14
 16--RxtrdGhb6hmJXsjeJWkwWPYg6FNBUU<- 58 BTC 2013-8-14
 1H--a9mUXG128suKf6vNkNEgEAeaozCkuB<- Mt.Gox

直前のノードの履歴1

ソース: blockchain.info



直前のノードの履歴2



ブロックチェーン追跡プログラム

```
require 'net/http'
require 'uri'
require 'json'
VL=1000000000 #探索する最低金額 10BTC
TL=1375000000 #起点時刻 2013-07-28 17:26:40 +0900
tmfilter=->time{->txs{txs.select{|tx|tx["time"]>time unless tx["time"]==nil}}}
outputs=->txs{txs.map{|tx|tx["out"]}.flatten}
vfilter=->out{out.select{|o|o["value"]>VL}}
outaddrs=->txs{txs.map{|tx|tx["addr"]}.uniq}
genlinks=->links{links.map{|w|w[1].map{|a|w[0],a,w[2]}}.reduce([]){|s,x|s+=x;s}.uniq.reject{|l|
l[0]==l[1]}}
outlinks=->nodes{->tl{->from{genlinks[from.map{|me|node=JSON.parse(Net::HTTP.get
URI.parse("http://blockchain.info/rawaddr/#{me}"))}
nodes<<node
[me,outaddrs[vfilter[outputs[tmfilter[tl][node["txs"]]]],node["total_received"]]]}}}
from=["1HcUa9mUXG128suKf6vNkNEgEAeaozCkuB"] #起点アドレスのリスト
links=[]
nodes=[]
srcs=from
begin
links+=outlinks[nodes][TL][from]
from=links.map{|x|x[1]}.uniq - srcs
srcs=(from+srcs).uniq
p links.size
p found=links.any?{|x|x[2]>100000000000000} #発見したい取引量のノード
end until found
links #結果が入っている
```

ブロックチェーン逆探索プログラム

□ 発見したノードから起点への逆経路のみを選択

```
ayasi=["1N2--42sbgCMbNtXFajz9XDACDFnFzdXzV",...]
reverse=->addrs{->links{
  if addrs==[] then []
  else
    rev=links.select{|x|addrs.member?(x[1])}
    (rev+reverse[rev.map{|ly|ly[0]}.uniq][links-rev]).uniq
  end
}}
revs=reverse[ayasi][links]
```

Transaction malleabilityの実際

インプットスクリプトの最初のところ

4d4c00304402201e17d13357bf..

4d = OP_PUSHDATA2: 続く2バイトをスタックにpushしろ

4c = OP_PUSHDATA : 続く1バイトをスタックにpushしろ

普通の実装なら、次のようになっている

4c304402201e17d13357bf..

ブロックチェーンに入ると最初の OP_PUSHDATA は見えない

Input Scripts

```
304402201e17d13357bf3cb47b64aeeb3203ee500affe909a59
02b945a7d017d28cf0fdccc73b76a43a964dcca22d0c2a5255
```

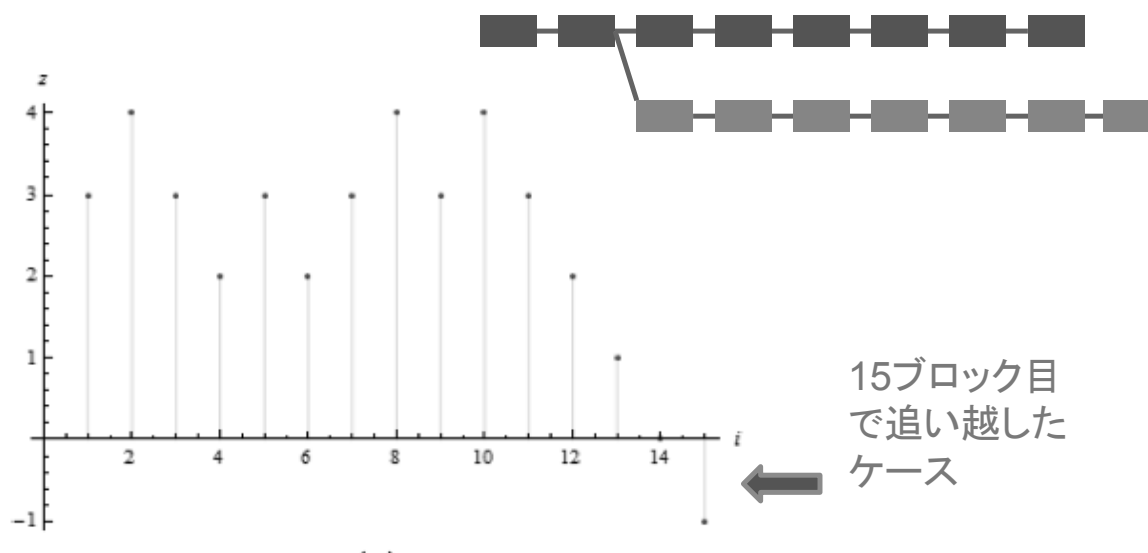
Bitcoinのspendable状態とは

- 取引やコインがコンファームされた安全な状態になったこと
- プルーフ・オブ・ワークは、攻撃者がたまたま勝つ場合もある
- コンファーム数が増えるにつれて確信できるようになる
- 標準的なコンファーム数：送金6、新規生成コイン100

多重使用攻撃

Meni Rosenfeld: Analysis of hashrate-based double-spending, 2012

z は、正統なブロックチェーンと偽造したブロックチェーンの差



取引所の課題

□ 資金洗浄防止

ガイドラインに沿った厳密な本人確認

□ 適正なセキュリティ・レベルの確保

フロント・オフィス・システム攻撃（カナダFlexcoin社）

バーチャル・ウォレットへの攻撃⇒ウォレットの暗号化

コールド・ストレージ管理の安全性（人的脅威、内部不正）

□ 最新仕様／プロトコルの遵守

コンファーム数：送金6、新規生成コイン100

Mt.Gox社は「未熟」なコインを送金し対応のため自動再送

採掘業者への規制

□ 取引所と採掘業者の連携

新規作成コインの質保証（コンファメーション数など）

取引展性のような仲介者攻撃が抑制できる

レーシング攻撃などの採掘業者による多重使用攻撃の抑止

ミクシングサービスへの規制

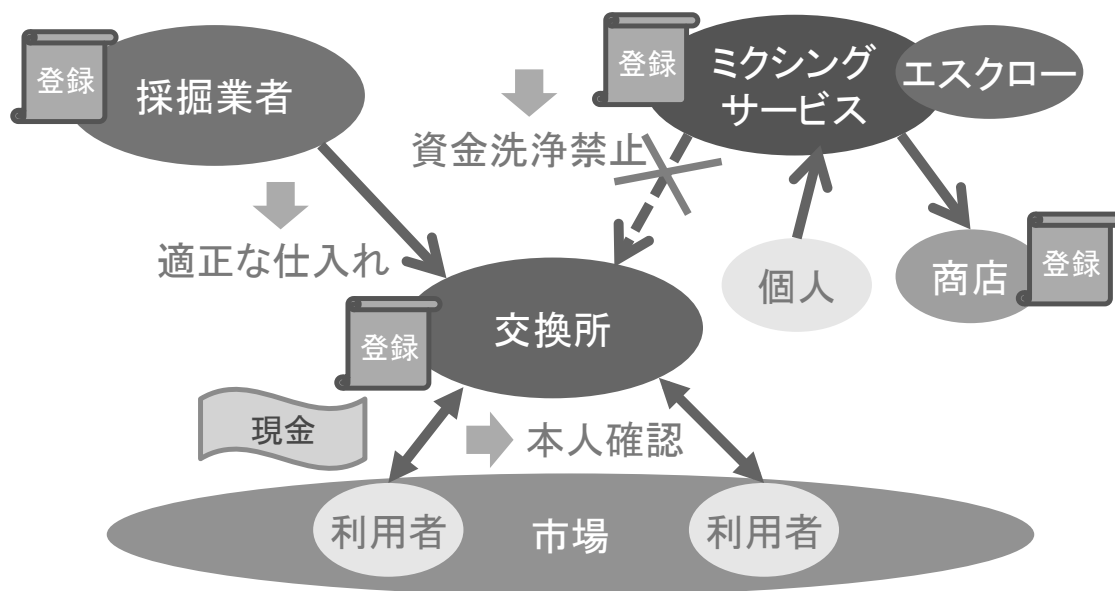
- 資金洗浄に利用される
- 盗難されたコインの追跡が不可能になる
- 犯罪取引に利用される
- 利用者のプライバシーは完全には保護されない
サービス側には利用者の支払い履歴が後にも残る
- プライバシ保護（人権） 目的の匿名化は必要

登録制？

- 主要なノードを登録制にする
交換所、採掘業者、ミクシングサービス(利用する商店)
- 登録制のコスト？
人や設備にコストがかかると金融機関と同じになる？
- 日本だけで規制しても意味がない？
ビットコインは、素性、履歴によって価値が変わる
日本の業者の信頼度を上げることは国際的な位置づけとして
意味がある

Bitcoinエコシステムの秩序

犯罪予防、利用者保護のための政府の関与
制度、規制



bitcoinのしくみと 設計思想

近畿大学
山崎重一郎

オリジナル論文

Satoshi Nakamoto (仮名)

**Bitcoin: A Peer-to-Peer Electronic Cash System,
2008**

論文の冒頭での主張

- 銀行などの「信頼できる第三者」が不要
- 当事者だけでオンライン決済が可能な
- 電子コインの多重使用問題に対する
- P2Pネットワーク型の解決方法

ソフトウェアだけで実現されている

□ 主なbitcoinウォレット

Desktop wallets

Desktop wallets are installed on your computer. They give you complete control over your wallet. You are responsible for protecting your money and doing backups.



Bitcoin
Core



MultiBit



Hive



Armory



Electrum

Mobile wallets

Mobile wallets allow you to bring Bitcoin with you in your pocket. You can exchange bitcoins easily and pay in physical stores by scanning a QR code or using NFC "tap to pay".



Bitcoin
Wallet



iPhone
bitWallet
送金機能なし！

Web wallets

Web wallets allow you to use Bitcoin on any browser or mobile and often offer additional services. However, you must choose your web wallet with care as they host your bitcoins.



bitcoinウォレットの使い方

□ 新しいウォレットを作成

(1) 秘密鍵を生成 (ECDSA Secp256k1)

(2) 秘密鍵から公開鍵を生成

(3) 公開鍵からビットコインアドレスを生成

base58(RIPEMD160(SHA-256(公開鍵)))

197bCWJMXpiYENn4ksAYgz9vwdF5EUpn4c



bitcoinウォレットの使い方

□ bitcoinを受け取る

自分のbitcoinアドレスを相手に教える



bitcoinウォレットの使い方

□ bitcoinを送金する

相手のbitcoinアドレスと金額を入れて送金



bitcoinウォレットの使い方

□ 取引履歴の確認

日時、アドレス、金額、ステータス（確認済みか？）

ステータスが確認済みなら、次の支払いに使用できる



ブロックチェーン

□Bitcoinの中核は取引記録の帳簿

ヤップ島の巨大石貨説



全員が
誰のものが
知っている

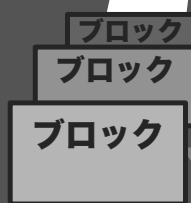
一反もめん説（岡田先生）



参加者全員
に見える
大福帳

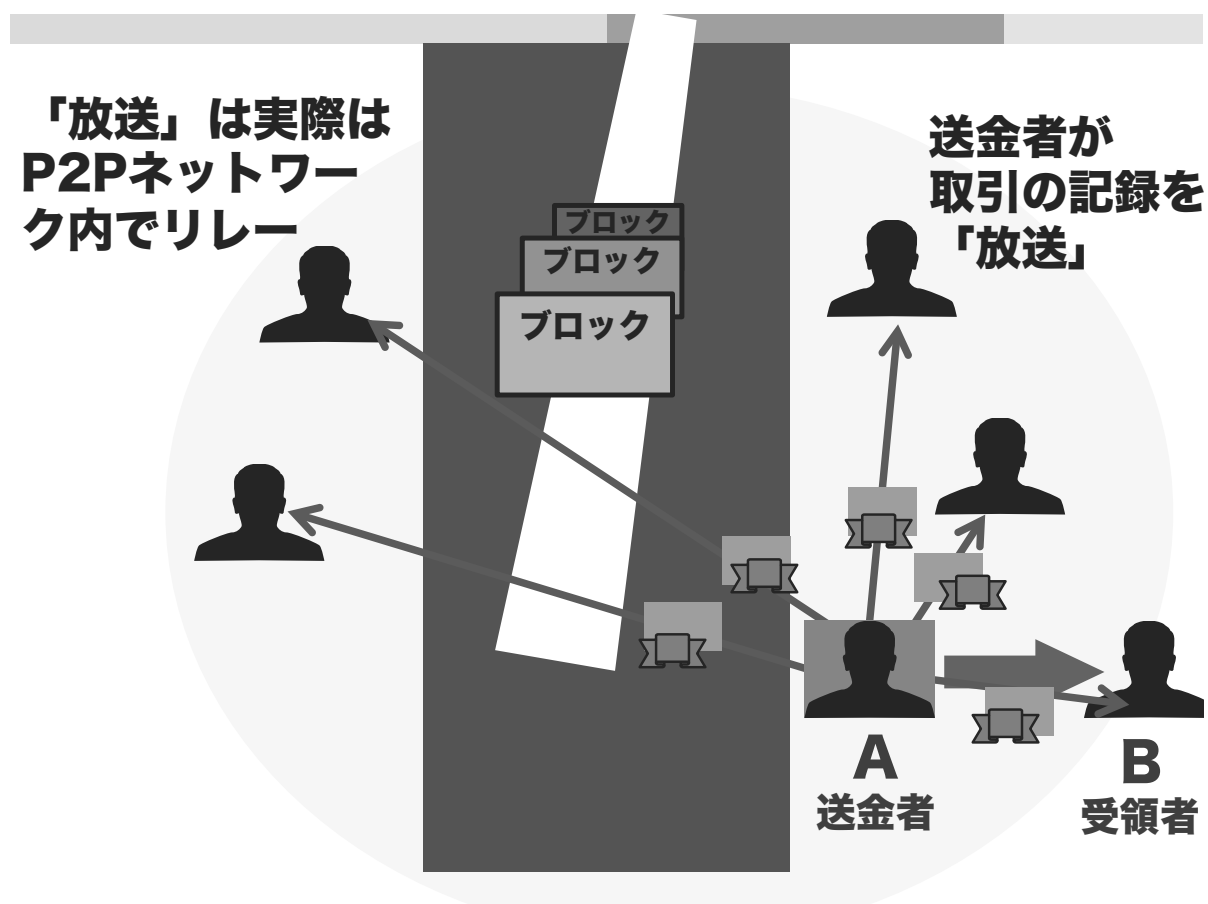
ブロックチェーン
は、抽象的な情報

P2Pネットワーク
上に冗長に存在

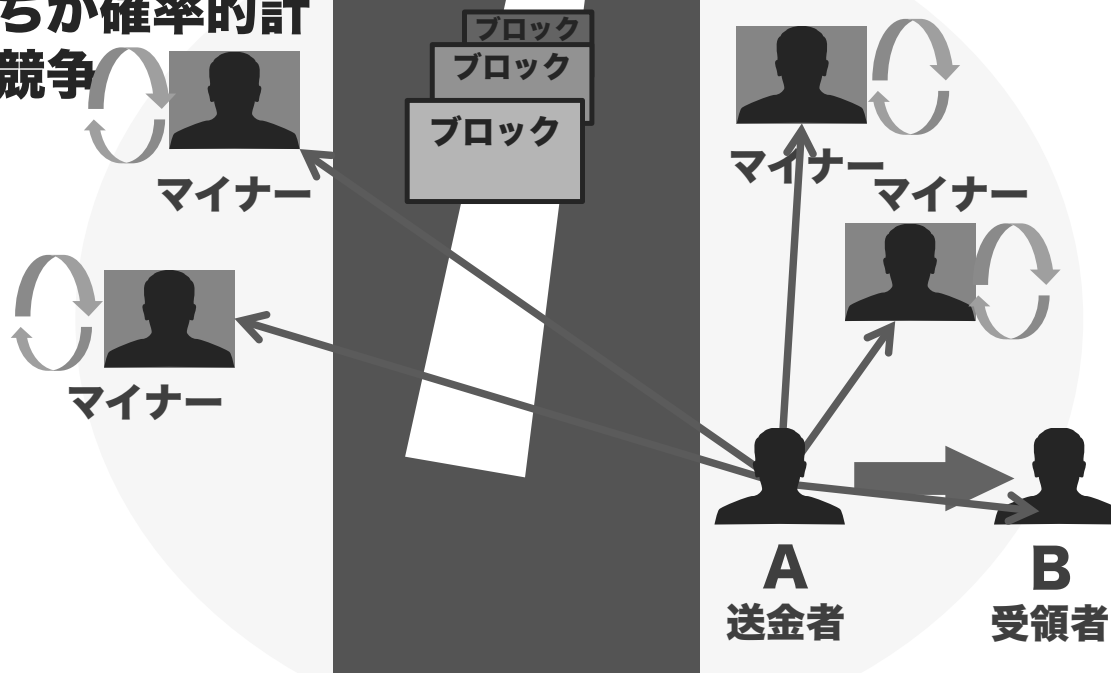


見える人にだけ
見える大福帳

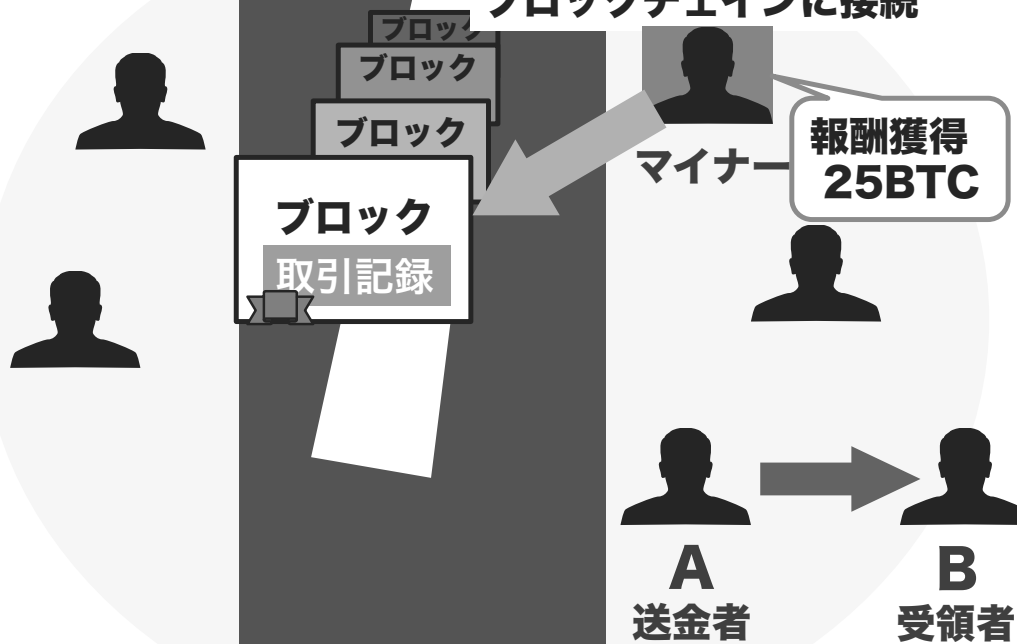
ブロックチェーン

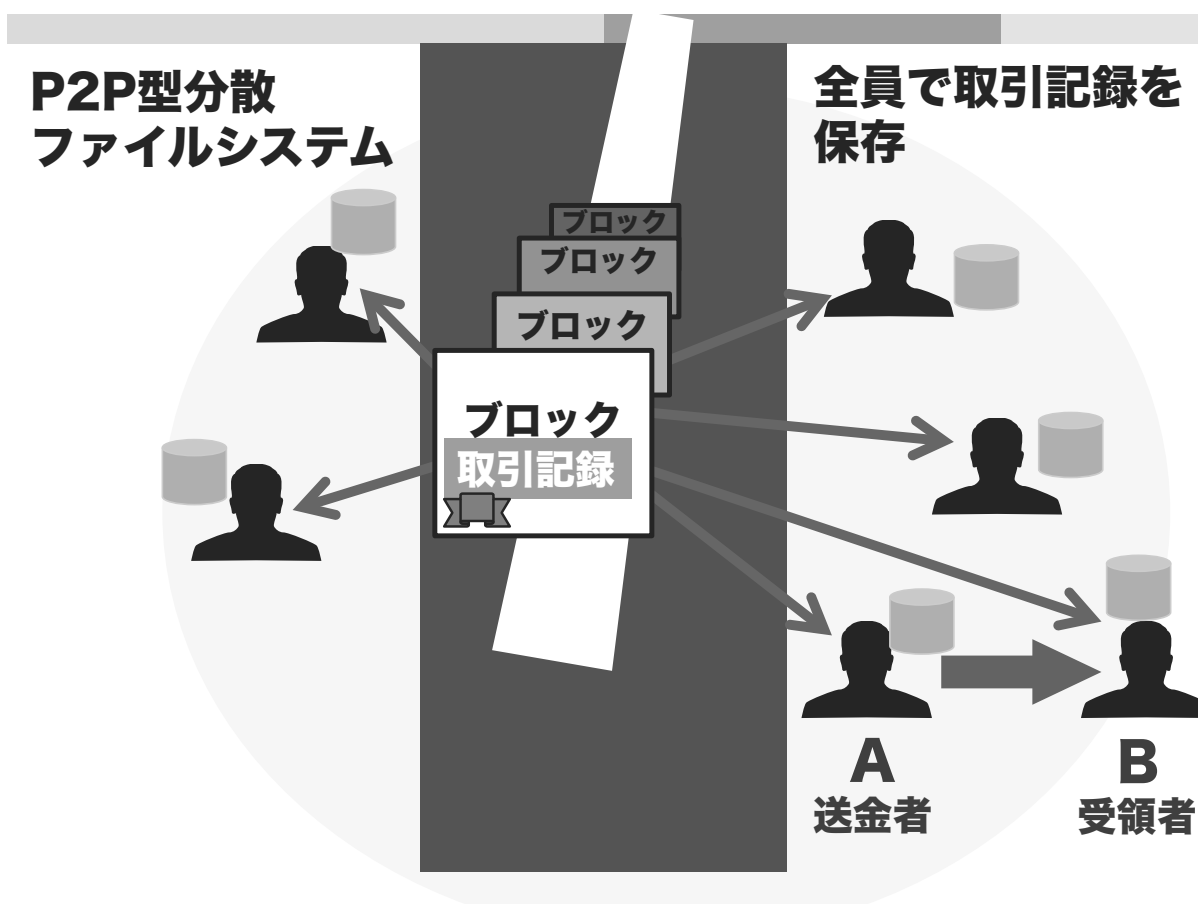
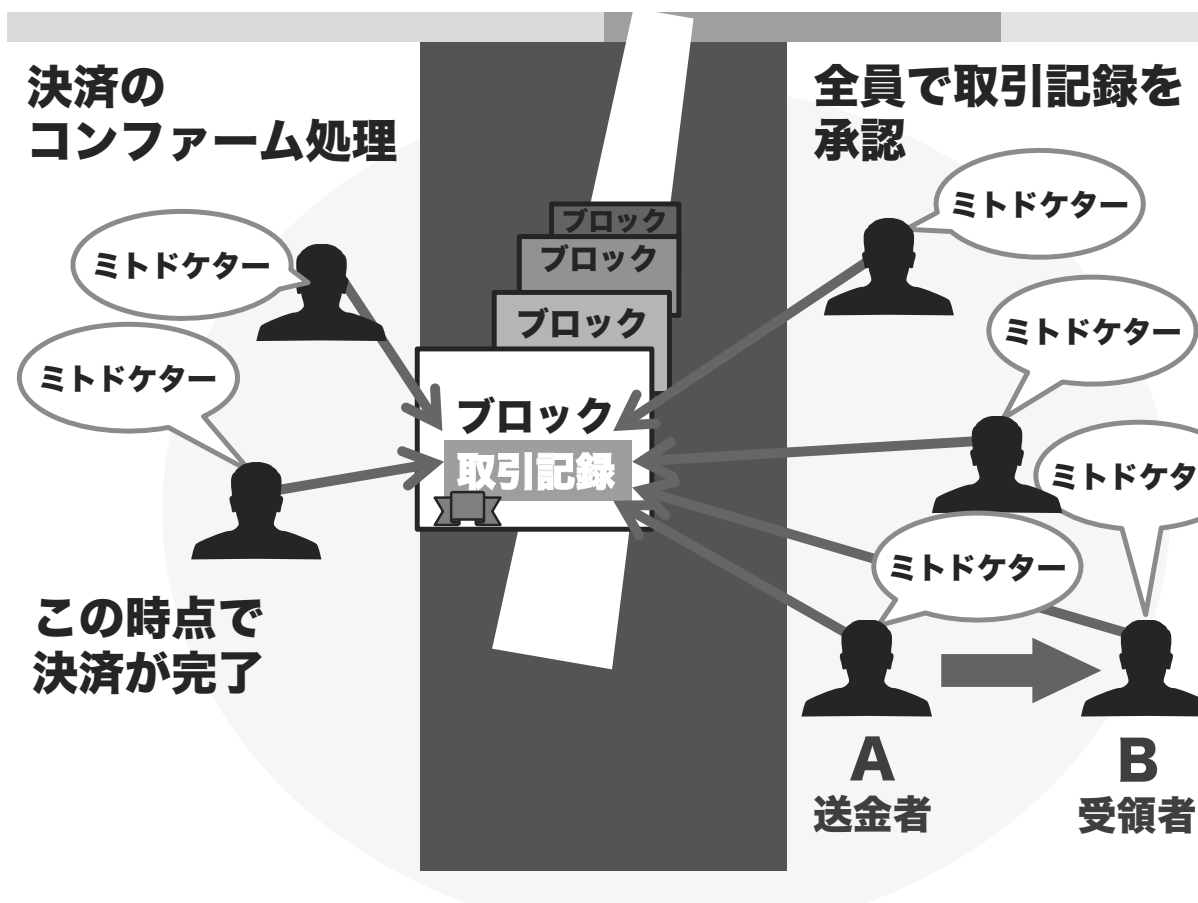


取引記録を受け
取ったマイナー
たちが確率的計
算競争



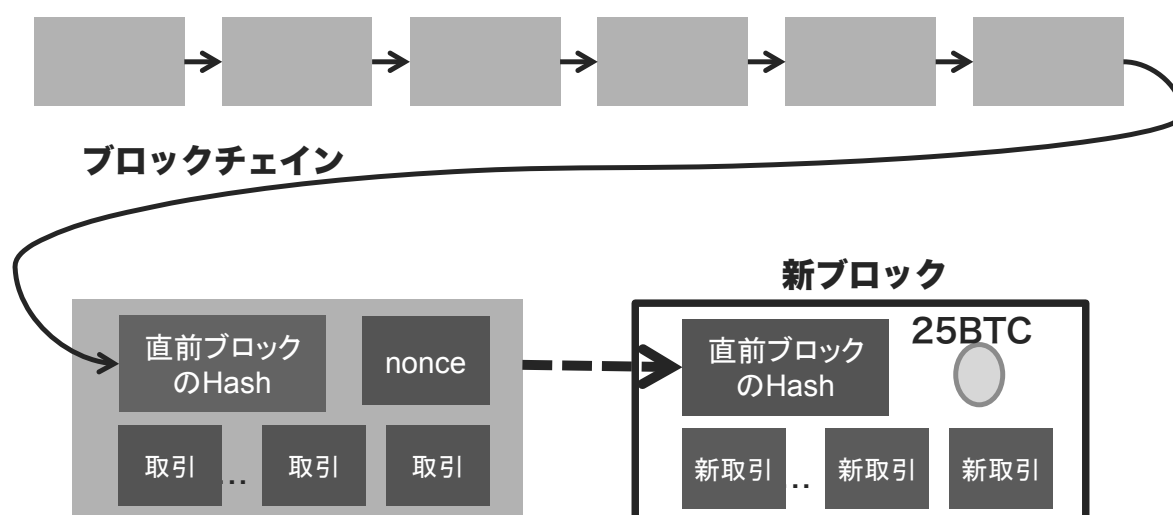
勝利したマイナーは
新規ブロックを生成
取引記録をブロックに登録し
ブロックチェーンに接続





ブロック・チェーンには

□ bitcoin経済開闢以来の
全ての取引の記録が入っている



技術的跳躍としてのBitcoin

□ 電子貨幣の理想像

Okamoto、Ohta の電子貨幣の6条件 (1991年)

- (1) 独立性: 物理媒体に依存せず、ネットワークで送受信可能
- (2) 安全性: 複製や偽造ができない
- (3) プライバシ: 使用者や使用履歴が特定されない
- (4) オフライン性: 支払時にオンライン検査をとらなわない
- (5) 転々流通性: 当事者間で直接譲渡可能
- (6) 分割可能性: 額面を分割して使用可能

Bitcoinは5条件をほぼ完全に実現

- (1) 独立性：ソフトだけ、インターネット経由で決済
- (2) 安全性：複製も偽造もできない
- (3) プライバシ：実用上十分なレベル
- (4) オフライン性：△ P2P型コンファームが必要
- (5) 転々流通性：実現されている
- (6) 分割可能性：実現されている

電子貨幣の 技術的跳躍

決済コストの（見かけ上の）低減

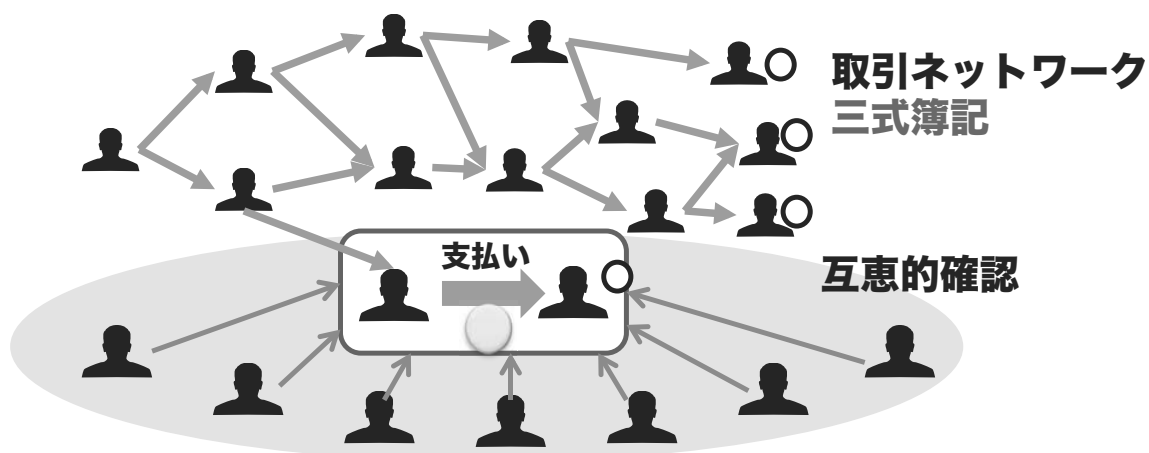
- 「信頼できる第三者」が決済方法を提供
決済コストが大きく見える



決済コストの（見かけ上の）低減

- bitcoinにおける決済の正統性の確認

- (1) 参加者全員が決済の確認コストを互恵的に分担
- (2) 三式簿記を利用したネットワーク型の取引正統性の証明



インターネットの 通信資源の 互恵的利用に似ている？

Bitcoinの安全性の拠り所

- コミュニティの互恵的良心？ → ×
- 法制度？ → ×
- 金融機関や大企業の信用？ → ×



人々の欲望！

**(コミュニティ、国家、金融機関が滅びても)
人々の欲望が続く限り機能しつづけるシステム**

金融機関の破綻 国民通貨の危機への耐性

地下経済や違法取引での利用

Bitcoinを実現する技術

□暗号技術とP2P技術の組合せ

暗号技術：標準化された堅実な技術のみ利用

P2P技術：普通のP2Pファイル共有システム

技術的跳躍はどこに？

- 電子貨幣の多重使用問題
- ブロックチェーンの偽造攻撃対策

技術的跳躍はどこに？

- ブロックチェーンの記録が正統なものであるという合意の形成
- 「ビザンティン将軍問題」とみなす
(分散システムの合意問題)

ビザンティン将軍問題

□裏切り者たちがいる合意問題

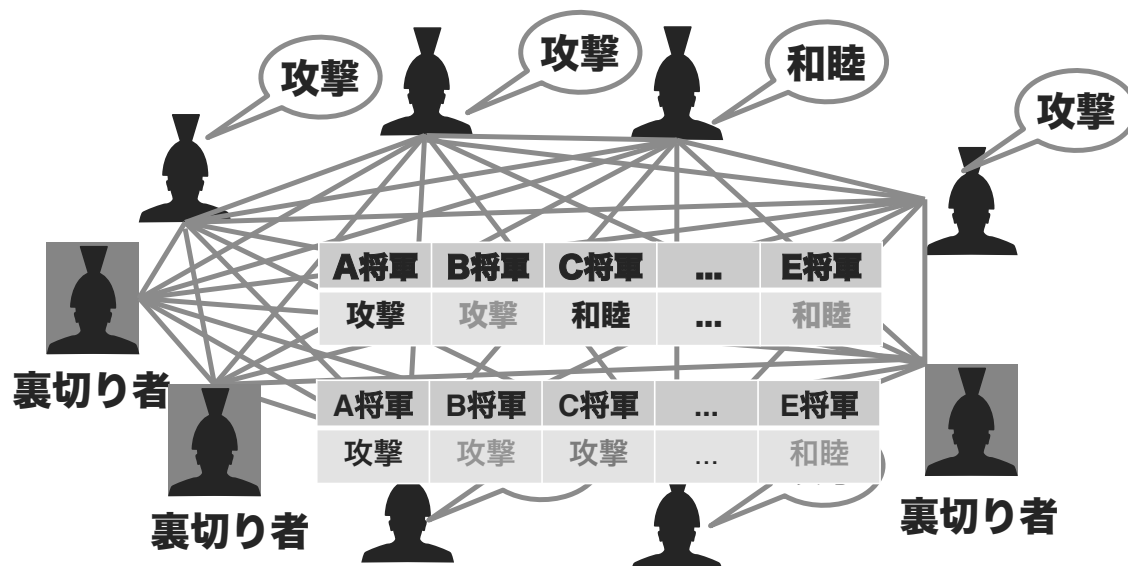
裏切り者たちが合意を失敗させようとする



ビザンティン将軍問題の古典的解

裏切り者が全体の1/3以上になると解が存在しない

解が存在しても、裏切り者の巧妙な結託は難問



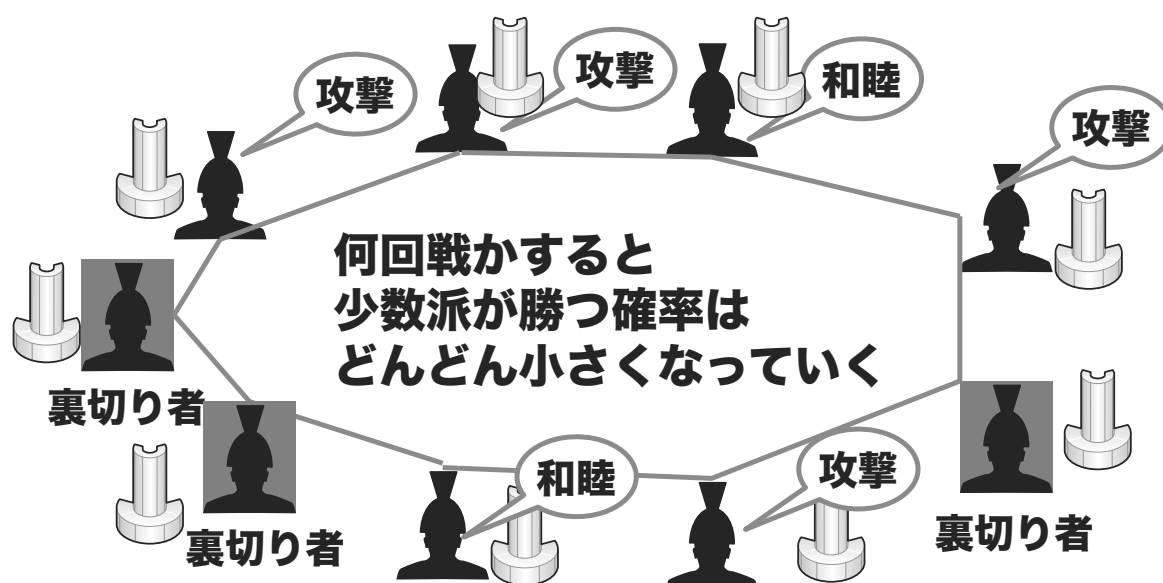
サトシ・ナカモトの方法

前提：ネット上の計算資源の総量が一定



サトシ・ナカモトの方法

少数派（50%未満）が勝利することもあるが



電子貨幣の多重使用の防止方法



最大の計算量プールによる記録を
唯一の正統なものとする

ブロックチェーンの分岐と 計算競争

- 複数の可能性を残した状態をつくる



しばらくして先に伸びた方が正統とみなし
他方は棄却される



プルーフ・オブ・ワーク

- 時間がかかる計算を頑張った証拠
- ハッシュ関数を使う例

データ + nonce (いろいろ変えてみる)



SHA-256

0000000001000111010001...1

ハッシュ値がゼロ**8**個で始まるnonceが見つければ
勝利！

プルーフ・オブ・ワーク

- 難易度を上げる方法

ネットワーク上の計算資源の総量が増えたとき

データ + nonce (いろいろ変えてみる)



SHA-256

000000000000111010001...1

ハッシュ値がゼロ**9**個で始まるnonceが見つければ
勝利！

プルーフ・オブ・ワークの 計算難易度の調整

- 2016ブロックの計算ごとに難易度を調整
- 平均10分で答えが出るようにする

2016×10分=2週間

計算完了が2週間より早ければ→難しくする

計算完了が2週間より遅ければ→簡単にする

プルーフ・オブ・ワーク の難易度の変遷



ビットコイン



ライトコイン

Bitcoin技術がもたらしたもの

- 情報がインターネット経由で「もの」のように個人から個人に転々流通する

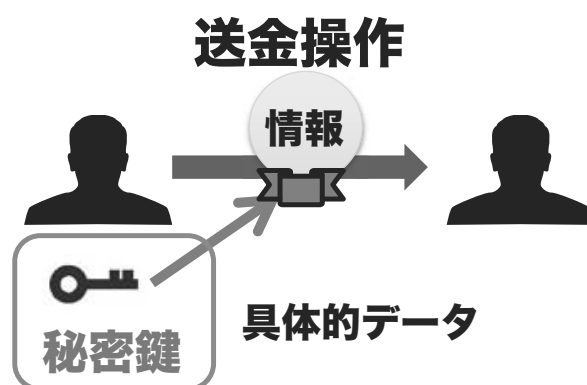


- ★ 流通する情報の所在は、ネットワーク全体である！
特定の媒体やサーバなどではない

抽象的な「情報」を
「もの」のように
扱う必要が発生

例：Bitcoinの盗難

「情報」＝対象 秘密鍵＝操作手段

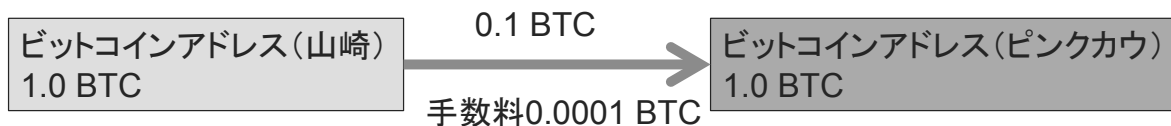


Bitcoinを盗む2つの方法

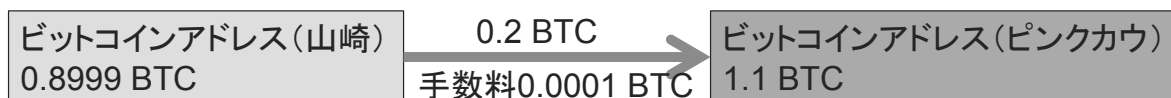
- (1) 抽象的な「情報」を盗む
- (2) 秘密鍵（データ）を盗む

Bitcoin取引の例（シナリオ）

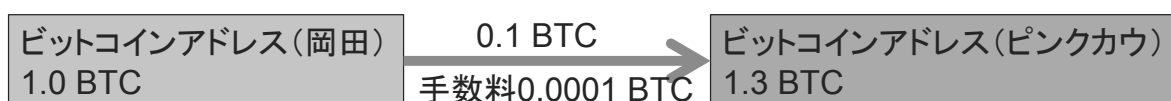
取引1



取引2



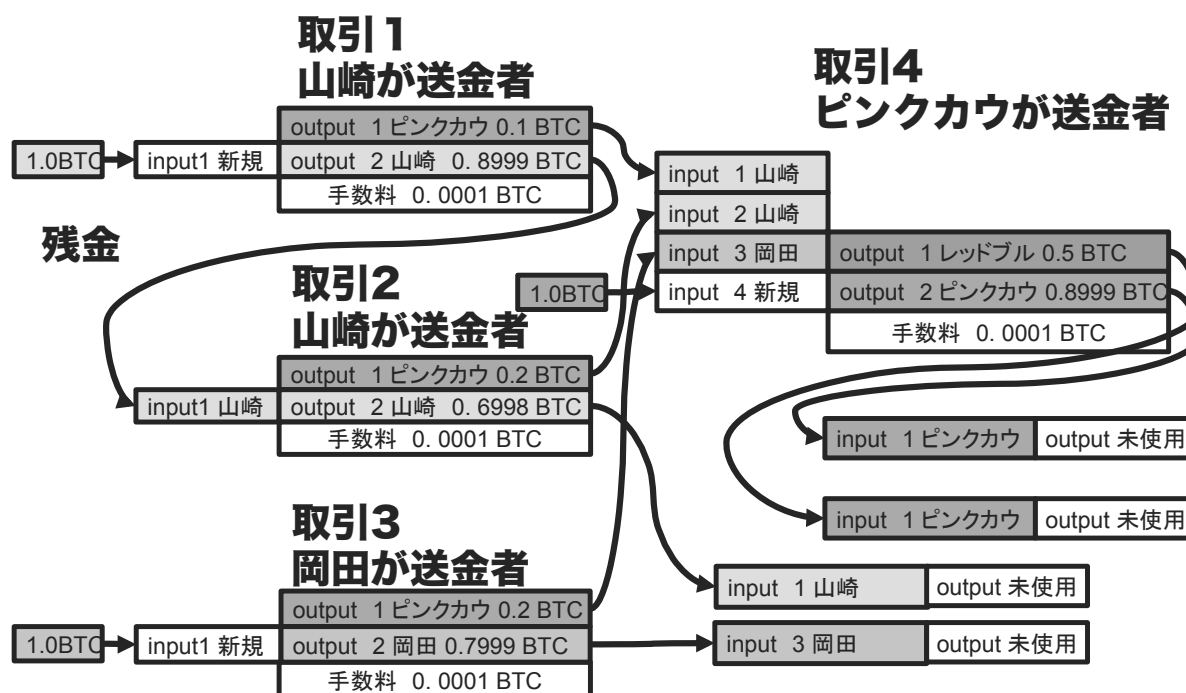
取引3



取引4

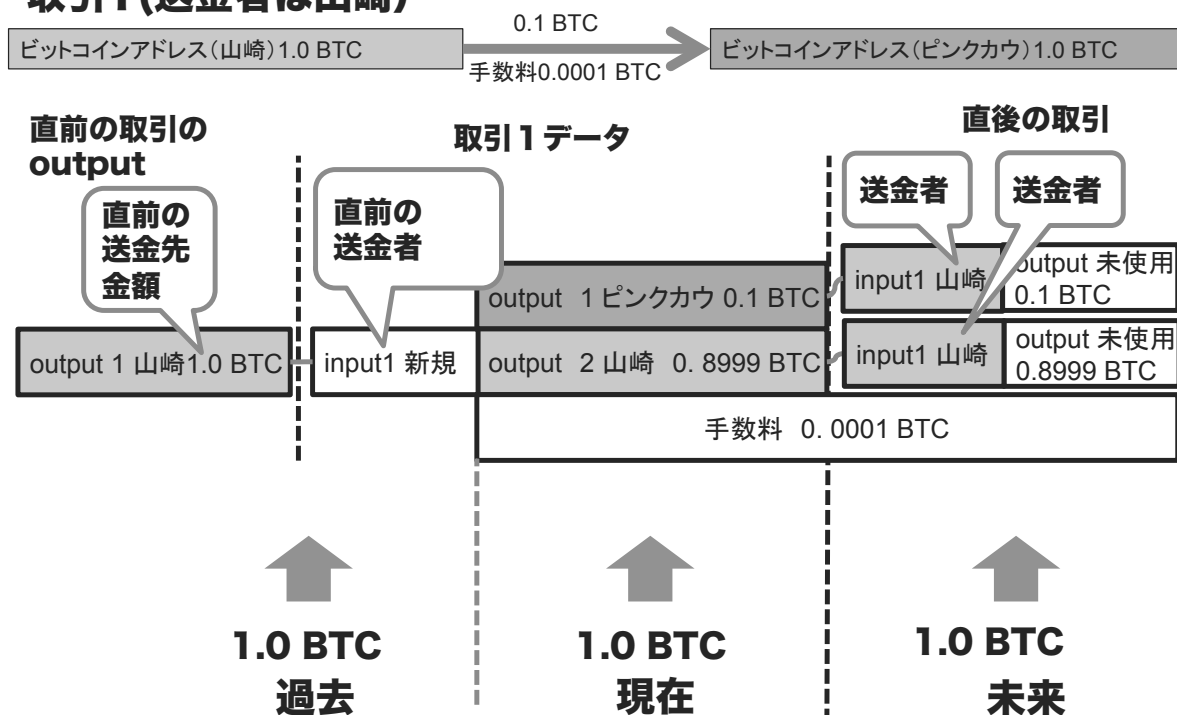


Bitcoinのトランザクショングラフ



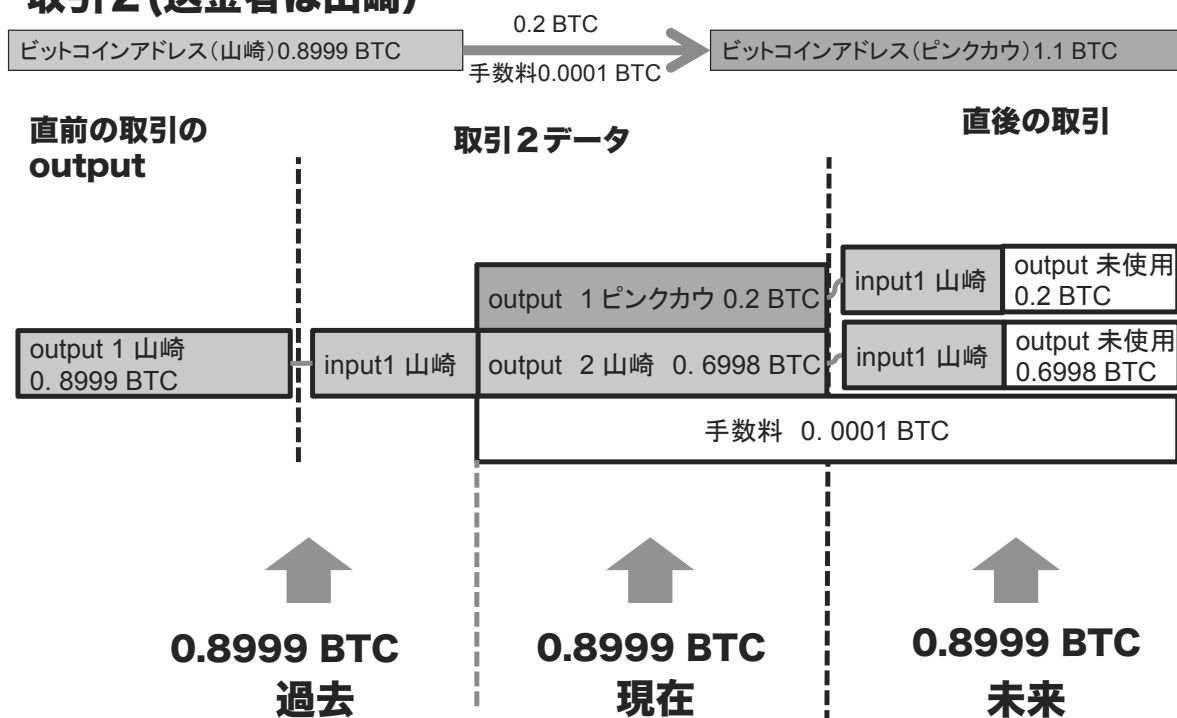
Bitcoinの取引データ

取引1 (送金者は山崎)



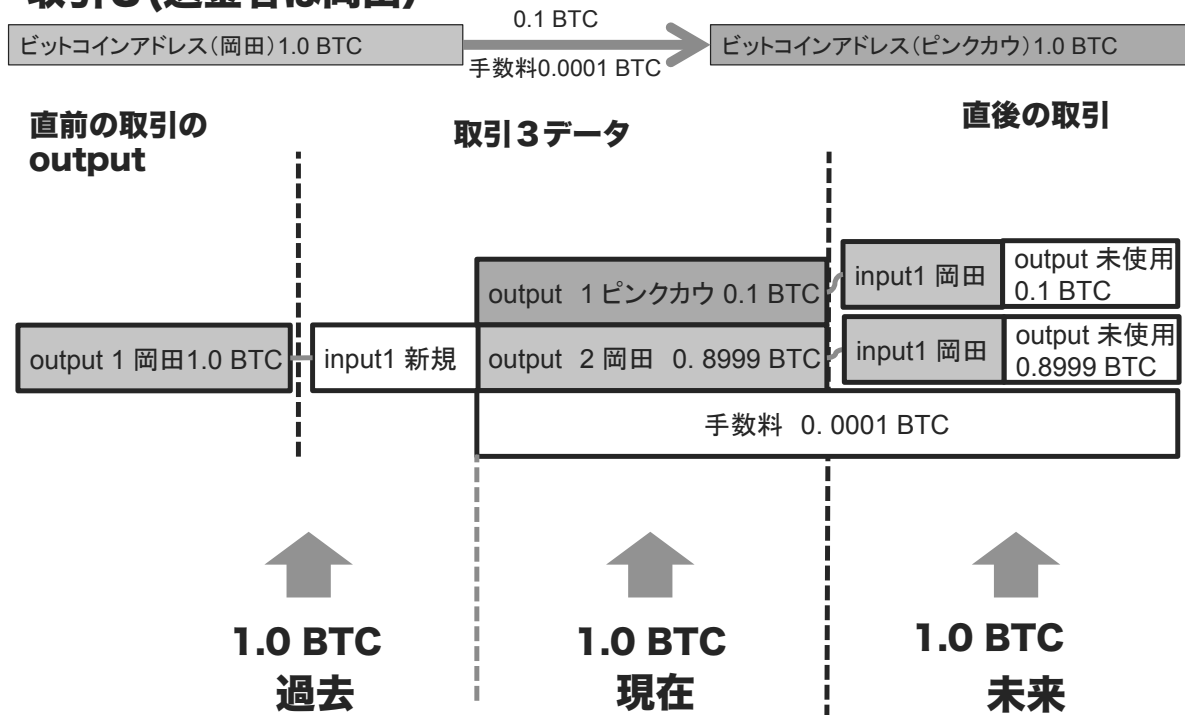
Bitcoinの取引データ

取引2 (送金者は山崎)



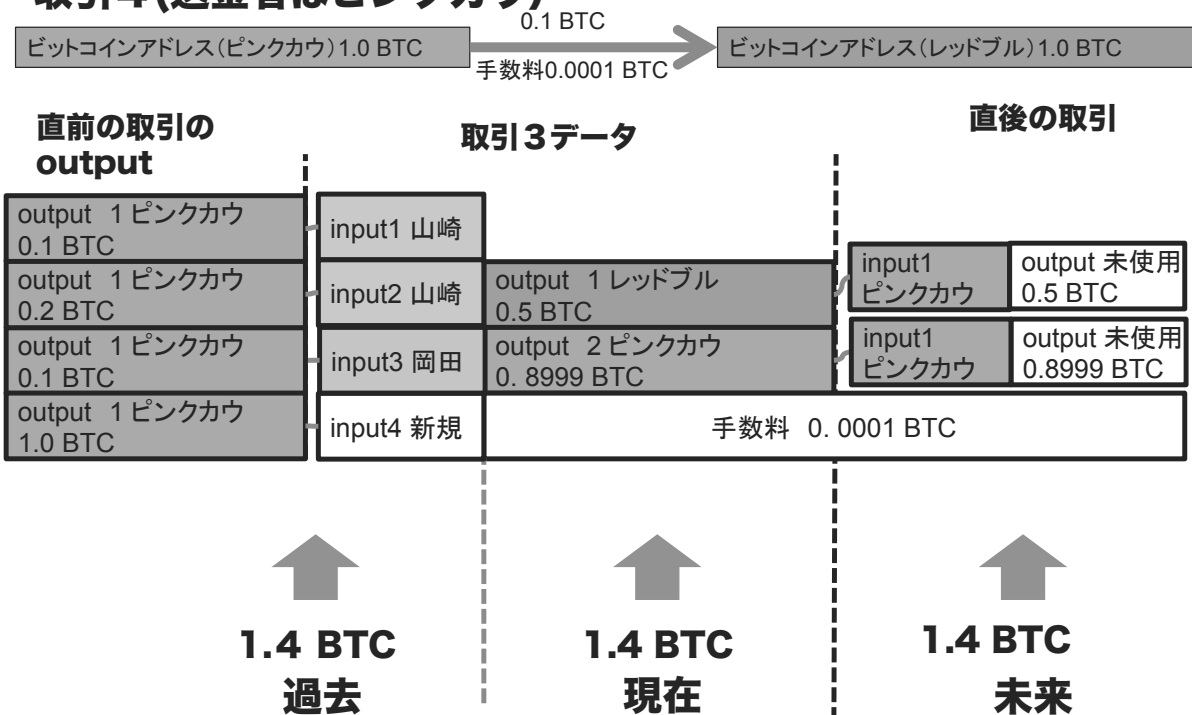
Bitcoinの取引データ

取引3 (送金者は岡田)

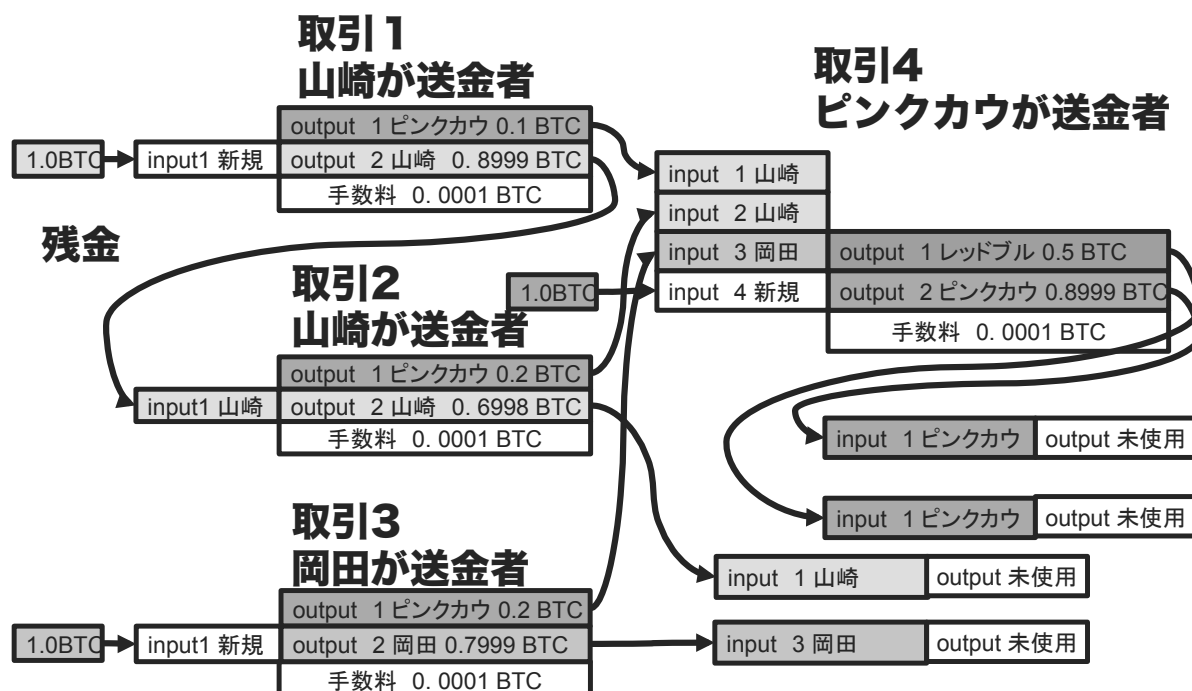


Bitcoinの取引データ

取引4 (送金者はピンクカウ)



Bitcoinのトランザクショングラフ



トランザクション・グラフの成長



□ 10分間に成長した部分

□ 成長した部分のみ新規ブロックに組み込まれる

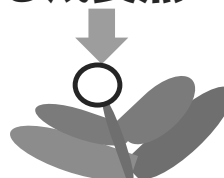
新規ブロック

新規取引

...

新規取引

□ 各取引の先端は、次に成長を始める成長点



Double Entry Accounting

□複式簿記（資本主義、株式会社の発達）

株式会社が株主に資産と利益を報告

対象は1企業の閉じたシステム

会計期：1年



Triple Entry Accounting

□三式簿記（bitcoin）

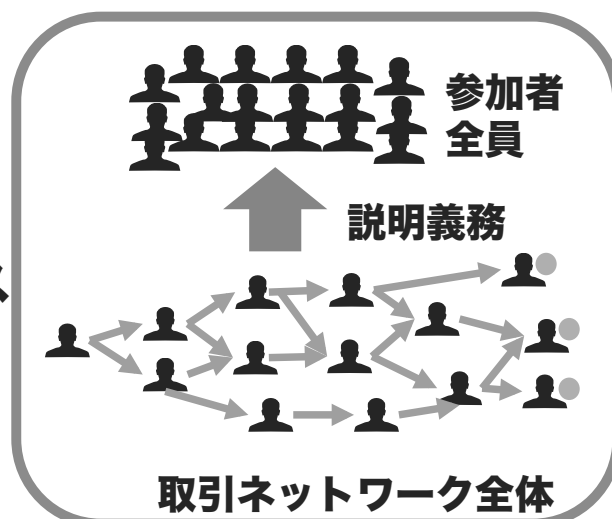
取引ネットワーク全体の正統性を検査

参加者全員に報告

会計期：約10分間

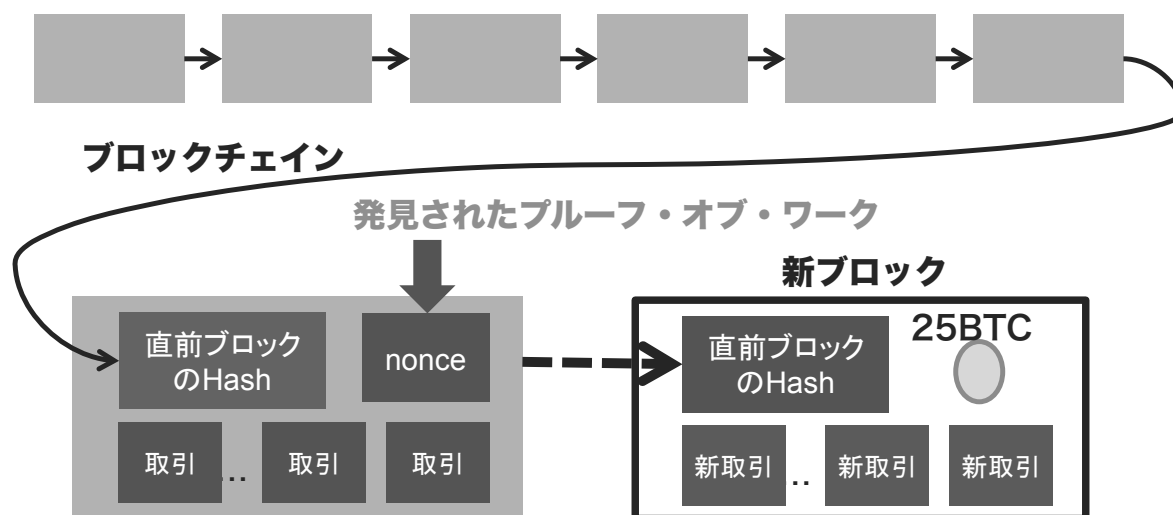
過去、現在、未来

タイムスタンプサービス



ブロック・チェーンには

- bitcoin経済開闢以来の
全ての取引の記録が入っている



Bitcoinの アイデンティティとプライバシー

- ブロックチェーン

bitcoin開闢以来の全取引履歴を誰でも確認可能

- Bitcoinアドレスは仮名

仮名：「本人」とリンクしない

匿名性（アイデンティティの積極的隠蔽）はない

しかしBitcoinの匿名化技術は存在

□mixing service

意図的に取引履歴を匿名化のためのサービス

□匿名化の目的

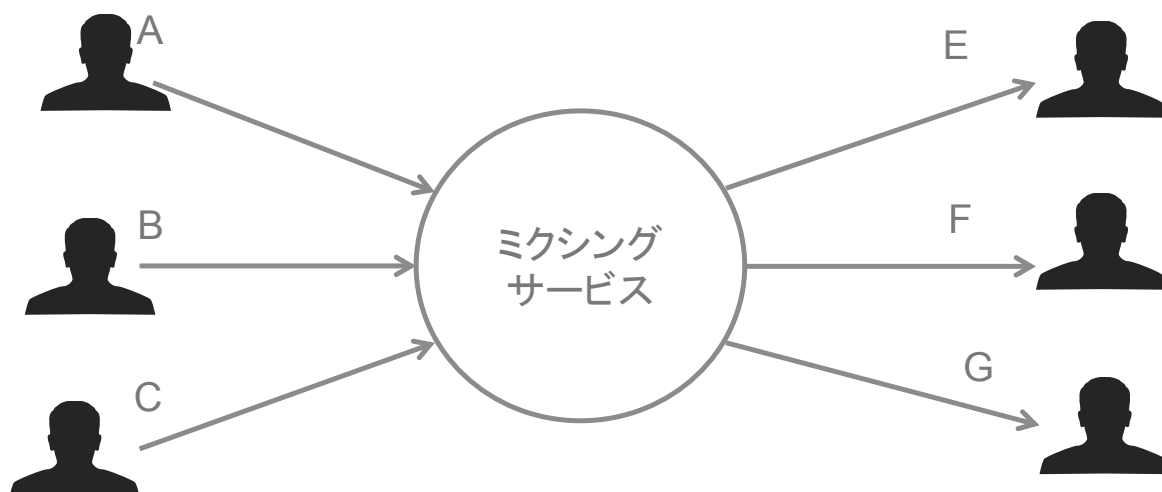
非合法取引、マネーロンダリング

人権としての決済プライバシー

mixing service

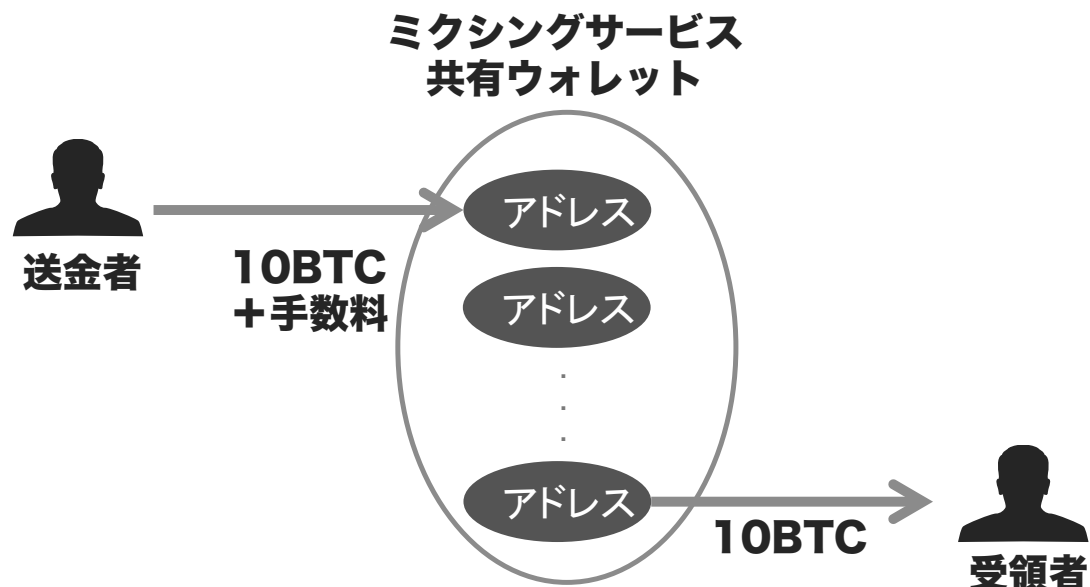
□ビットコインの取引の流れを匿名化する

仮名から仮名への取引の流れを攪乱する



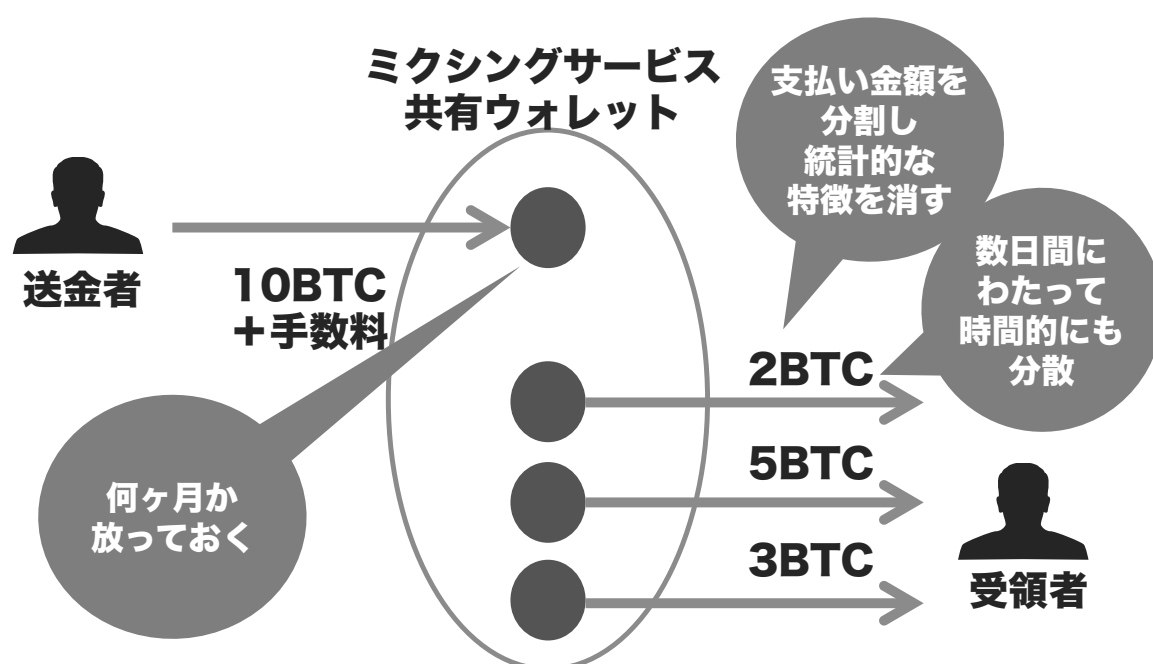
mixing serviceの原理

□ 共有ウォレット



mixing serviceの原理 2

□ プロファイリング防止技術の例



mixing serviceの弱点

- **運営主体が信用できない**
- **mixing serviceに取引履歴が残る**
- **mixing service自身は扱い額の2倍以上の
大量のコイン保有が必要**

**プライバシー保護技術は
bitcoinの今後の
技術的課題の一つ**

マイニングのコスト

- 実際はブロックチェーンを伸ばす作業
- 計算競争に勝利すると新規コイン
- 専用ハード無しで勝利できる確率は僅少
- 現状は非常に電力を消費する

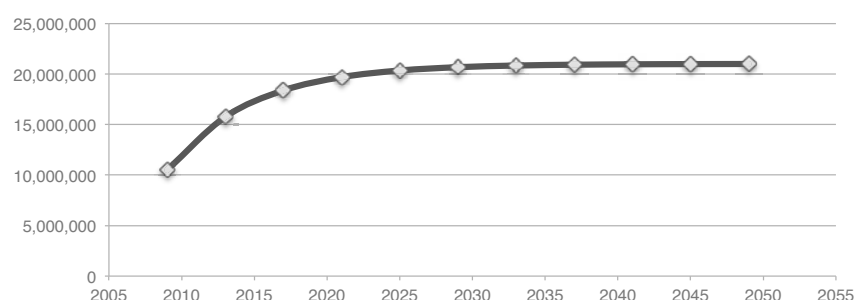
マイニングのインセンティブ


- 新ブロック生成時の報酬

最初の4年 50BTC、次の約4年 25BTC

- 取引手数料報酬

現在の実装では手数料の支払いは自発的行為で、無料でもOK





設計思想としては
消費電力量＝採掘コインの価値
ではないが...
人間の欲望の加熱速度
今後の研究課題も多い