ビットコイン：ピアツーピアの電子マネーシステム

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

概要純粋なピアツーピア版の電子マネーがあれば、金融機関を通さずに、ある当事者から別の当事者へ直接オンライン決済を行うことができるようになる。デジタル署名はソリューションの一部を提供するが、二重消費を防ぐために信頼できる第三者が依然として必要である場合、主な利点は失われる。我々は、ピアツーピアネットワークを用いた二重支出の問題に対する解決策を提案する。ネットワークはハッシュベースのプルーフ・オブ・ワークの継続的なチェーンに取引をハッシュ化することでタイムスタンプを付与し、プルーフ・オブ・ワークをやり直さない限り変更できない記録を形成している。最も長いチェーンは、一連の出来事を証明するだけでなく、それが最大のCPUパワーのプールから生まれたことを証明するものである。CPUパワーの過半数が、ネットワークを攻撃するために協力していないノードによって制御されている限り、彼らは最長のチェーンを生成し、攻撃者を出し抜くことができる。ネットワーク自体は最小限の構造しか必要としない。メッセージはベストエフォート方式で送信され、ノードは自由にネットワークから離脱したり再参加したりすることができ、離脱中に起こったことの証明として最長のプルーフ・オブ・ワークチェーンを受け入れる。

1. はじめに

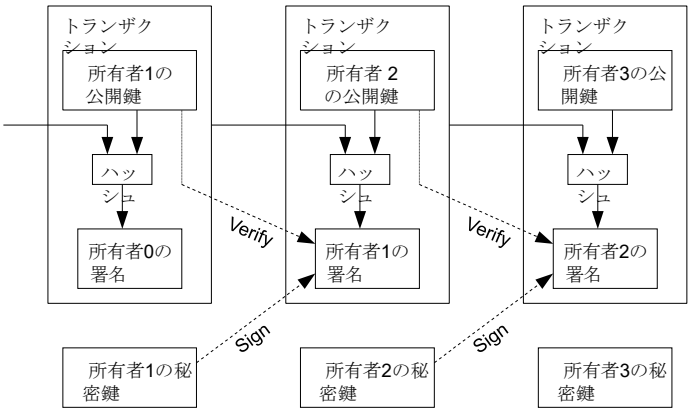
インターネット上での商取引は、電子決済を処理するために、信頼できる第三者機関としての金融機関にほぼ全面的に依存するようになりました。このシステムはほとんどの取引で十分に機能するものの、信頼に基づくモデル特有の弱点に悩まされている。金融機関は紛争を調停することを避けられないため、完全に非可逆的な取引は実際には不可能である。仲介のコストは取引コストを高め、実用的な最小取引サイズを制限し、少額の気軽な取引の可能性を断ちます。また、非可逆的なサービスに対して非可逆的な支払いを行うことができなくなるという、より広範なコストが発生するのです。可逆性があることで、信頼の必要性が広がります。販売者は、顧客に対して警戒心を持ち、必要以上の情報を求めていかなければならない。一定の割合の不正は避けられないものとして受け入れられている。これらのコストと支払いの不確実性は、直接的には現物の通貨を使うことで回避できますが、信頼できる当事者なしに通信チャネル上で支払いを行う仕組みは存在しません。

必要なのは、信頼の代わりに暗号的な証明に基づく電子決済システムであり、信頼できる第三者を介することなく、意思のある二者が直接取引できるようにすることである。計算上逆引きが不可能な取引は売り手を詐欺から守り、日常的なエスクローの仕組み

は買い手を守るために容易に実装できるだろう。本論文では、P2P分散タイムスタンプサーバを用いて、取引の時系列的順序の計算機的証明を生成することで、二重支出問題の解決策を提案する。このシステムは、誠実なノードが集団で攻撃者ノードのいかなる協力グループよりも多くのCPUパワーを制御する限り、安全である。

2. トランザクション

我々は電子コインをデジタル署名の連鎖と定義している。各所有者は、直前の取引のハッシュと次の所有者の公開鍵に電子署名を行い、これをコインの末尾に追加することでコインを次の所有者に譲渡する。受取人は署名を検証することで、所有権の連鎖を確認することができる。

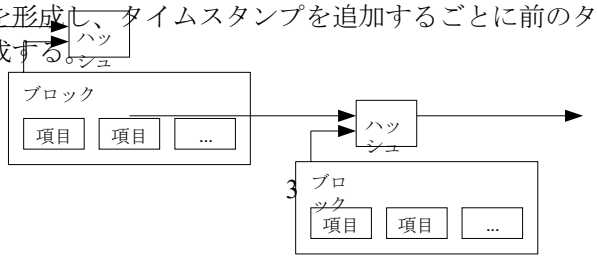


もちろん問題は、所有者の一人がコインを二重に使っていないことを受取人が確認できないことである。一般的な解決策は、信頼できる中央機関（造幣局）を導入し、すべての取引で二重使用がないかどうかをチェックすることである。各取引の後、コインは新しいコインを発行するために造幣局に返却されなければならず、造幣局から直接発行されたコインだけが二重使用されていないことが信用されるのである。この解決法の問題点は、貨幣システム全体の運命が造幣局を運営する会社に依存し、すべての取引が銀行のように造幣局を経由しなければならないことである。

前の所有者がそれ以前の取引に署名していないことを受取人が知るための方法が必要である。我々の目的では、最も古い取引が重要であるため、後に行われた二重支払いの試みは気にしない。取引がないことを確認する唯一の方法は、すべての取引を認識することである。造幣局ベースのモデルでは、造幣局はすべての取引を把握しており、どの取引が先に到着したかを判断していた。信頼できる当事者なしでこれを実現するには、取引は公的に発表され
[1]、参加者が受け取った順番の単一の履歴に合意するシステムが必要である。受取人は、各取引の時点で、大多数のノードが最初に受け取ったものであることに合意していることを証明する必要がある。

3. タイムスタンプサーバー

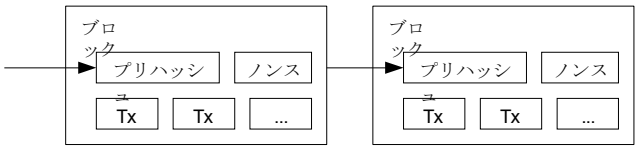
我々が提案する解決策は、まずタイムスタンプサーバーから始まる。タイムスタンプサーバーは、タイムスタンプされる項目のブロックのハッシュを取り、新聞やUsenetの投稿のように、そのハッシュを広く公開することで機能します[2-
5]。タイムスタンプは、そのデータがハッシュに入るために、明らかにその時点で存在していたはずであることを証明するものである。各タイムスタンプは前のタイムスタンプを含んでハッシュを形成し、タイムスタンプを追加するごとに前のタイムスタンプが強化され、連鎖を形成する。



4. プルーフ・オブ・ワーク

ピアツーピアの分散タイムスタンプサーバーを実現するには、新聞やUsenetの投稿ではなく、Adam BackのHashcash [6]のようなプルーフオブワークシステムを使用する必要があります。プルーフ・オブ・ワークは、SHA-256のようにハッシュ化したときに、ハッシュがゼロビット数で始まるような値をスキャンすることである。平均的に必要な作業は、必要なゼロビットの数の指数関数であり、1つのハッシュを実行することによって検証することができる。

我々のタイムスタンプネットワークでは、ブロックのハッシュに必要なゼロビットを与える値が見つかるまでブロック内のnonceをインクリメントすることでプルーフ・オブ・ワークを実装している。一度、プルーフ・オブ・ワークを満たすためにCPUの労力が費やされると、その作業をやり直さない限り、ブロックは変更できない。このブロックの後に後続のブロックが連鎖しているため、このブロックを変更するためには、そのブロックの後にあるすべてのブロックをやり直す必要があるのです。



また、プルーフ・オブ・ワークは、多数決における代表権の決定という問題を解決する。もし多数決が1IPアドレス1票に基づくものであれば、多くのIPを割り当てることができる誰によっても覆される可能性があります。プルーフ・オブ・ワークは、基本的に1CPU-1票である。多数決は最も長いチェーンで表現され、そのチェーンには最大のプルーフ・オブ・ワークの労力が投入されている。CPUパワーの過半数が誠実なノードによって制御されている場合、誠実なチェーンは最も速く成長し、競合するチェーンを凌駕することになる。過去のブロックを修正するためには、攻撃者はそのブロックとそれ以降のすべてのブロックのプルーフ・オブ・ワークをやり直し、正直なノードの作業に追いつき、追い越す必要がある。後ほど、遅い攻撃者が追いつく確率は、後続のブロックが追加されるにつれて指数関数的に減少することを示す。

ハードウェアの速度の向上と、時間と共に変化するノード稼働への関心を補うため、プルーフ・オブ・ワークの難易度は、1時間当たりの平均ブロック数をターゲットとする移動平均によって決定される。ブロックの生成速度が速すぎると、難易度は高くなる。

5. ネットワーク

ネットワークを実行する手順は以下の通りです。

1) 新しいトランザクションはすべてのノードにブロードキャストされる。
2) 各ノードは新しいトランザクションをブロックに集めます。
3) 各ノードは、自分のブロックに対して難しいプルーフ・オブ・ワークを見つけることに取り組む。
4) あるノードがプルーフ・オブ・ワークを見つけると、そのブロックを全ノードにブロードキャストする。
5) ノードは、そのブロックに含まれるすべての取引が有効であり、かつすでに費やされていない場合にのみ、そのブロックを受け入れる。

6) ノードは、受理されたブロックのハッシュを前のハッシュとして、チェーンの次のブロックの作成に取り組むことで、ブロックの受理を表明します。

ノードは常に最長の鎖を正しいものと考え、それを延長する作業を続ける。2つのノードが異なるバージョンの次のブロックを同時にブロードキャストした場合、一部のノードはどちらかを先に受信する可能性がある。その場合、先に受信した方を処理するが、長くなった場合に備えてもう一方の枝を保存しておく。次のプルーフ・オブ・ワークが見つかり、一方のブランチが長くなった時点で同点が解消され、もう一方のブランチで作業していたノードは長いほうのブランチに切り替える。

新規取引のブロードキャストは必ずしもすべてのノードに到達する必要はない。多くのノードに到達しさえすれば、すぐにブロックに入ることができる。ブロックブロードキャストはメッセージの欠落にも寛容である。ノードがブロックを受信しなかった場合、次のブロックを受信して見逃したことに気づけば、そのブロックを要求します。

6. インセンティブ

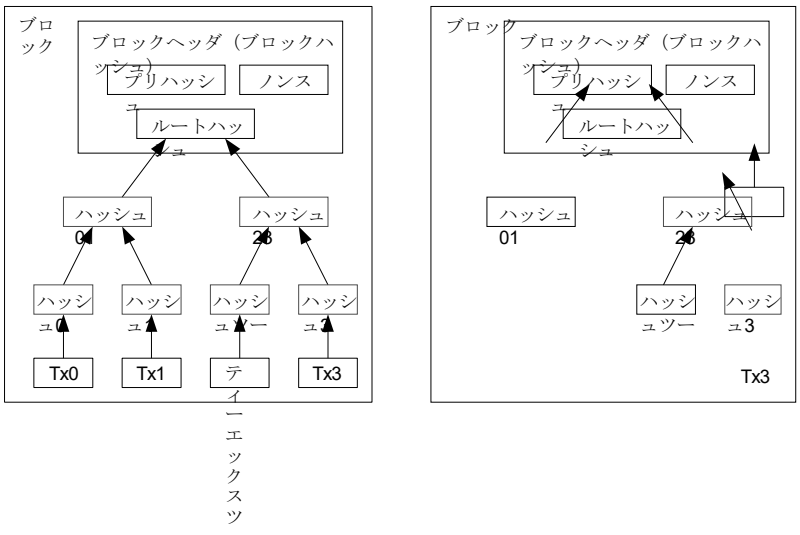
慣習として、ブロックの最初の取引は、そのブロックの作成者が所有する新しいコインを開始する特別な取引である。これはノードにネットワークをサポートするインセンティブを与え、コインを発行する中央機関が存在しないため、最初にコインを流通させる方法を提供するものである。一定量の新しいコインを着実に追加していくことは、金の採掘者が金の流通を増やすために資源を費やすことに似ている。この場合、消費されるのはCPU時間と電力である。

インセンティブは取引手数料で賄われることもある。取引の出力値が入力値より小さい場合、その差は取引手数料となり、その取引を含むブロックのインセンティブ値に加えられる。所定数のコインが流通すると、インセンティブは取引手数料に完全に移行し、完全にインフレフリーになることができる。

このインセンティブは、ノードが誠実であり続けることを促すのに役立つかもしれない。もし貪欲な攻撃者が誠実なノードよりも多くのCPUパワーを集めることができたら、そのパワーを使って支払いを詐取するか、新しいコインを生成するか、どちらかを選ばなければならない。攻撃者は、システムや自分の富の正当性を損なうよりも、自分に有利なルール、つまり他の誰よりも多くの新しいコインを獲得できるルールに従って行動する方が有益であると考えるはずである。

7. ディスクの空き容量を確保する

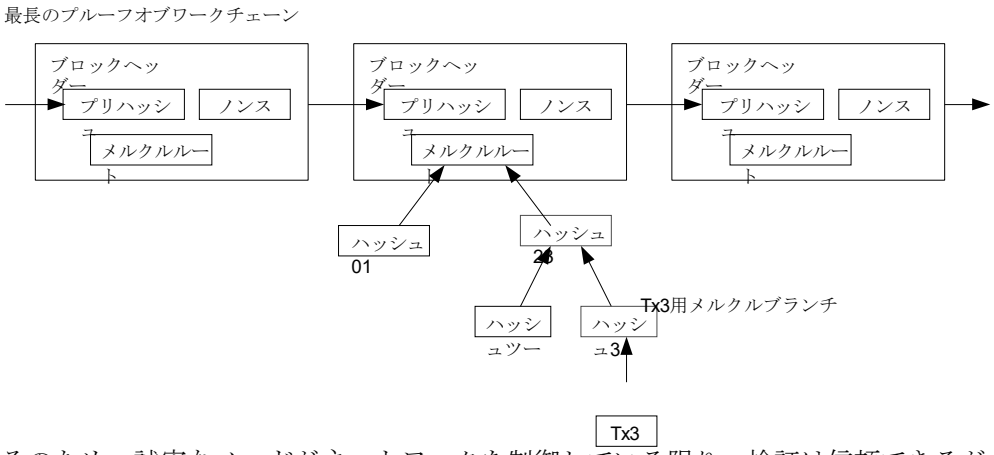
コインの最新の取引が十分な数のブロックの下に埋まると、それ以前の使用済みの取引はディスクスペースを節約するために破棄することができる。ブロックのハッシュを壊さずにこれを実現するために、トランザクションはブロックのハッシュにルートだけを含むメルクル木 [7][2][5]
でハッシュ化される。古いブロックは木の枝を切り離すことでコンパクトにすることができる。内部のハッシュは保存する必要がない。



トランザクションのないブロックヘッダは約80バイトとなる。ブロックが10分ごとに生成されると仮定すると、80バイト×6×24×365＝4.2MB/年になります。2008年現在、2GBのRAMを搭載したコンピュータが一般的に販売されており、ムーアの法則によれば、現在の成長率は
年間1.2GBであれば、ブロックヘッダをメモリ上に保持する必要があっても、ストレージに問題はないはずだ。

8. 入金確認の簡略化

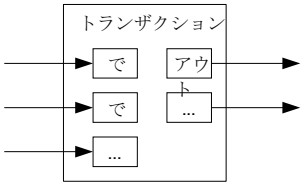
ネットワークノードをフル稼働させることなく、支払いを検証することが可能である。ユーザーは最長のプルーフ・オブ・ワークチェーンのブロックヘッダーのコピーを保持するだけでよく、最長のチェーンを持っていると確信するまでネットワークノードに問い合わせることで取得できる。その取引を自分で確認することはできないが、チェーン上のある場所にリンクさせることで、ネットワークノードがそれを受け入れたこと、そしてその後に追加されたブロックがネットワークに受け入れられたことを確認することは可能である。



そのため、誠実なノードがネットワークを制御している限り、検証は信頼できるが、ネットワークが攻撃者に制圧された場合はより脆弱になる。ネットワーク・ノードは自分自身で取引を検証することができるが、攻撃者がネットワークを圧倒し続ける限り、簡略化された方法は攻撃者が捏造した取引に騙される可能性がある。これを防ぐための1つの戦略は、無効なブロックを検出したときにネットワークノードからのアラートを受け付け、ユーザーのソフトウェアにブロック全体とアラートされたトランザクションをダウンロードさせ、矛盾を確認させることである。頻繁に支払いを受ける企業は、より独立したセキュリティと迅速な検証のために、独自のノードを実行したいと思うだろう。

9. 価値の結合と分割

コインを個別に扱うことは可能であろうが、転送の際に1セントごとに個別のトランザクションを行うのは扱いにくい。価値の分割と結合を可能にするために、トランザクションには複数の入力と出力が含まれる。通常、以前の大きな取引からの単一の入力か、より小さな金額を組み合わせた複数の入力があり、最大で2つの出力がある：1つは支払い、もう1つはお釣りがあれば送り手に返す。

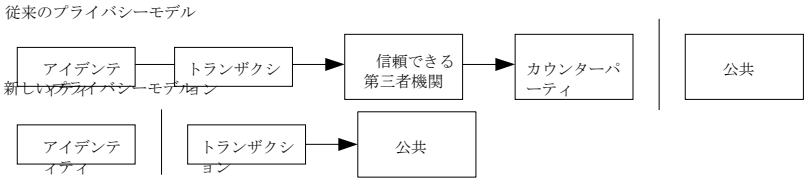


あるトランザクションがいくつかのトランザクションに依存し、それらのトランザク

ションがさらに多くのトランザクションに依存するようなファンアウトは、ここでは問題にならないことに注意すべきである。トランザクションの履歴の完全なスタンドアロンコピーを抽出する必要は決してない。

10. プライバシー

従来の銀行モデルは、情報へのアクセスを当事者と信頼できる第三者に限定することで、あるレベルのプライバシーを実現しています。すべての取引を公表する必要があるため、この方法は使えない。しかし、情報の流れを別の場所で断ち切ることで、プライバシーを維持することができる。つまり、公開鍵を匿名にするのだ。つまり、公開鍵を匿名にすることである。一般の人は、誰かが誰かに金額を送っていることを知ることができるが、その取引と誰かを結びつける情報はない。これは、証券取引所が公開する情報のレベルに似ている。個々の取引の時間と規模、つまり「テープ」は公開されるが、その当事者が誰であるかは分からない。



追加のファイアウォールとして、共通の所有者にリンクされないように、各トランザクションに新しいキーペアを使用する必要があります。多入力トランザクションの場合、必然的にそれらの入力が同じ所有者によって所有されていることが明らかになるため、やはり多少のリンクは避けられない。そのリスクは、鍵の所有者が明らかになった場合、リンクによって同じ所有者に属する他のトランザクションが明らかになる可能性があることである。

11. 算出方法

我々は、攻撃者が誠実なチェーンよりも速く別のチェーンを生成しようとするシナリオを考えている。たとえこれが達成されたとしても、無から価値を生み出したり、攻撃者のものでなかったお金を奪ったりといった、任意の変更に対してシステムを開放することはない。ノードは無効な取引を支払いとして受け入れることはなく、正直なノードはそれらを含むブロックを決して受け入れない。攻撃者ができるのは、最近使ったお金を取り返すために自分の取引の1つを変更することだけである。

正直者の鎖と攻撃者の鎖の間の競争は、二項ランダムウォークとして特徴づけることができる。成功事象は正直者の鎖が1ブロック伸びてリードが+1されることであり、失敗事象は攻撃者の鎖が1ブロック伸びて差が-1されることである。

ある赤字から攻撃者が追いつく確率は、ギャンブラーの破滅問題に類似している。無限の信用を持つギャンブラーが赤字からスタートし、損益分岐点を目指して無限の試行錯誤をするとする。彼が損益分岐点に到達する確率、あるいは攻撃者が正直な連鎖に追いつく確率は、以下のように計算できる[8]。

p = 誠実なノードが次のブロックを発見する確率
q = 攻撃者が次のブロックを発見する確率
q_z = 攻撃者がzブロックの後ろから追いつくことができる確率

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

*p>q*とすると、攻撃者が追いつかなければならないブロックの数が増えるにつれて、確率は指数関数的に減少する。このように不利な条件下では、早い段階で幸運な突進をしなければ、遅れをとるにつれてその確率はどんどん小さくなっていく。

次に、新しい取引の受信者が、送信者が取引を変更できないと十分に確信するまでに、どれくらいの時間待つ必要があるかを考える。送信者は攻撃者であり、受信者にしばらくの間自分が支払ったように思わせ、時間が経過した後に自分への支払いに切り替えたいと考えていると仮定する。そうなれば受信者は警告を受けるが、送信者はそれが手遅れになることを望んでいる。

受信者は新しい鍵ペアを生成し、署名の直前にその公開鍵を送信者に渡す。これにより、送信者は前もってブロックの連鎖を準備し、十分な幸運に恵まれるまで連鎖に取り組み、その瞬間に取引を実行することができなくなる。取引が送信されると、不正な送信者は自分の取引の別バージョンを含む並列チェーンで秘密裏に作業を開始する。

受信者はトランザクションがブロックに追加され、その後に*z個*のブロックがリンクされるまで待機する。彼は攻撃者の正確な進捗量を知らないが、誠実なブロックがブロックごとに平均的な期待時間を要したと仮定すると、攻撃者の潜在的な進捗は期待値を持つポアソン分布になる。

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

攻撃者が現在も追いつける確率を求めるには、攻撃者の進歩の度合いに応じたポアソン密度を、その時点から追いつける確率に掛け合わせればよい。

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

とす

分布の無限尾の合計を避けるために再整理すると・・・。

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

Cコードに変換する...

```
#インクルード <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));
    }
    を返します。
}
```

いくつかの結果を実行すると、確率はzとともに指数関数的に低下することがわかる。

```
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
```

Pが0.1%未満になるように解くと...

```
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. 結論

私たちは、信頼に頼らない電子商取引のシステムを提案しています。まず、電子署名で作られたコインという通常の枠組みを採用した。これは所有権の管理を強力に行うが、二重消費を防ぐ方法がなく不完全である。これを解決するために、我々はプルーフ・オブ・ワークを用いたピアツーピアネットワークを提案し、取引履歴を公開記録することにした。このネットワークは、構造化されていないシンプルさゆえに堅牢である。ノードはほとんど協調することなく、一度にすべての作業を行う。メッセージは特定の場所に送られることはなく、ベストエフォートで配送されればよいので、ノードを識別する必要はない。ノードは自由にネットワークから離脱し、再参加することができ、離脱中に起こったことの証明としてプルーフ・オブ・ワークのチェーンを受け入れる。ノードはCPUパワーで投票し、有効なブロックを受け入れればそれを拡張し、無効なブロックを拒否すればそのブロックを拒否することを表明する。必要なルールやインセンティブは、このコンセンサスメカニズムによって強制することができる。

参考文献

[1] W.Dai, "b-money", http://www.weidai.com/bmoney.txt, 1998.

[2] H.Massias, X.S. Avila, and J.-J. Quisquater, "Design of the 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 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," *Inc. 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.