ビットコイン： P2P 電子通貨システム

中本 哲史
satoshi@gmx.com
www.bitcoin.co.jp

概要　真の P2P 電子通貨が実現すると、金融機関の介在無しに、利用者から利用者へと直接オンラインで支払いできるようになるだろう。電子署名によって、その機能の一部は実装可能である。だが従来の方法では、多重使用を禁ずるために第三者機関を設置する必要があり、電子通貨の利点を生かせなかった。本論文で提案するのは、多重使用問題を P2P ネットワークで解決する方法である。このネットワークは、ハッシュ関数による演算量証明を利用する。その証跡をチェーンでつなぎ続けることにより、いつ、どのような取引が行われたかを証明可能にする。チェーン内の取引履歴を改ざんしようとしたら、時間をかけて演算量証明をやり直さなければならない。過去の出来事を時系列的に確認する場合には、ネットワーク上で最長のチェーンを調べれば良い。さらに、最長チェーンは、CPU 能力を最も費やした計算結果でもある。CPU 能力を持つ者の大半が、ネットワークへの攻撃者を無視していれば、その善良なノード群が作るチェーンは、攻撃者のそれを長さで上回り続ける。このネットワークに必要な規則は、極めて簡素である。メッセージはベストエフォートで拡散すれば良いし、各ノードはいつ離脱・再接続しても構わない。再接続時に最長チェーンを受け取ることによって、離脱していた間に何が起きたかを把握できるからである。

1　はじめに

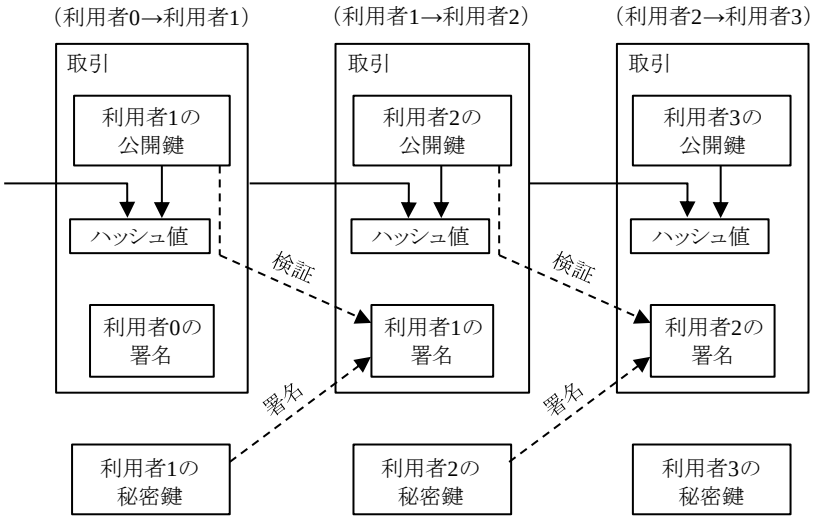
インターネットを介した既存の取引は、支払いを電子的に実行するために、信頼できる第三者機関を必要とする。現在は、もっぱらそれを金融機関が請け負っている。大半の取引は問題なく行われているが、信頼に基づくモデルにつきものの脆弱性問題に苦しみ続けている。また金融機関が間に入ると、利用者間のいざこざを仲裁する必要性が生じるため、完全に非可逆的な取引を行えない。仲裁には費用がかかるため、取引コストが増大したり、取引額の下限を設けて少額取引を制限せざるを得なくなったりする。これにより、非可逆的な支払いで非可逆的なサービスを受けるという枠組みを実現できず、多大なコストが生じてしまう。また、可逆的な取引には、互いに信頼できる相手とでなければ成り立たないという問題もある。そのため、販売側は顧客のことをより深く知ろうとし、本来必要ではないような情報まで要求して、顧客を苛立たせる。そのような対策を立てても、ある割合で詐欺が発生することは回避できない。こういったコストや、支払いが確実に行われるかわからないという問題は、物質的な通貨を利用することで解

決する。だが、電子取引では、信頼機関を設けること以外の解決策が見つからなかった。

必要なのは、第三者機関が無くとも二者が取引を行えることである。そのためには、信頼ではなく暗号技術に基づいた支払いシステムがあれば良い。計算理論的に非可逆的な計算を拠り所として取引を行えば、売人を詐欺者から守り、預託機構と連携すれば顧客も守る。本論文で提案する方法により、取引履歴に残された時刻を計算理論で検証すれば、通貨を多重には使用されなくなる。そのために、P2P 分散タイムスタンプサーバを利用する。善良なノード群が、攻撃者のノード群よりも CPU 能力で上回っていれば、このシステムはセキュリティ的に安全である。

2　電子通貨のやり取り

本システムでは、コインを、デジタル署名をつなげたチェーンの形で表現する。コインを支払う側は、前回までのコインの取引内容をハッシュ化した値と、受取人の公開鍵をハッシュ化した値とを合わせて、電子的に署名する。それをコインの最後に付け加え、受取人に送信する。受取人は電子署名を検証することにより、そのコインを誰が所有していたかという履歴を辿れる。



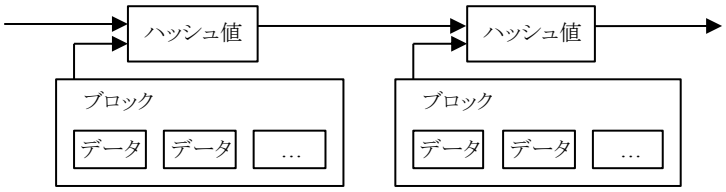
ここで問題となるのは、受け取ったコインが、過去に多重使用されたかもしれないことである。これまでに提案された解決法は、信頼できる「造幣局」がすべての取引を監視し、多重使用がないことを保証するというものだった。取引が成立するたびにコインは造幣局に返され、新たなコインが発行される。造幣局が直接発行したコインであることが、多重使用されていないことの証明となる。これは、すべての取引を造幣局経由で行うのと同じで、金融システムの命運が造幣局を運営する組織に左右されるという、銀行と同様の問題を含んでいる。

ともかく、過去の所有者の誰もがコインを他に使用していないことを、受取人が検証できれば良い。そこで、コインが多重に使用された場合は最初の取引だけを有効とし、それ以降の取引は無効であるとする。このように取り決めておけば、すべての取引を調

査し、自分の取引が 2 番目以降でないことを確認すれば良いだけである。造幣局ベースのモデルでは、造幣局が取引をすべて監視しているから、どの取引が先に行われたかが明確に分かる。これを、信頼できる機関無しで実現するには、取引が公開され[1]、それを元にした同一の取引履歴を、利用者全員が共有しているようなシステムが必要である。受取人は取引のたびに、コインが他人に送られていないことについて、ノードの大多数が認めるか否かを確認することになる。

3　タイムスタンプサーバ

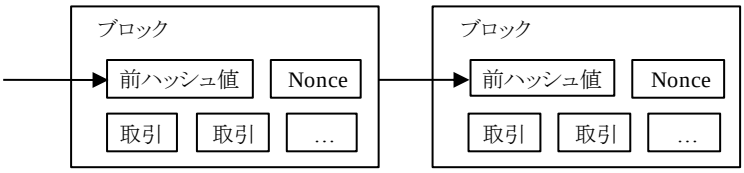
提案する解決法は、タイムスタンプサーバを必要とする。タイムスタンプサーバとは、ブロックを受け取ってハッシュ値を計算し、一般的には新聞やネットニュースのような仕組みで配信するサーバである[2-5]。データをハッシュ化しタイムスタンプに組み込むことにより、そのデータがその時点で存在したことを、明確に証明する。また、タイムスタンプは、一つ前の段階のタイムスタンプをハッシュ化したものを組み込み、チェーンを作る。そのため、タイムスタンプが後方に連なるに従って、信頼性が増して行くことになる。



4　演算量証明

P2P ベースで分散タイムスタンプサーバを実装するためには、新聞やネットニュースなどよりも、むしろアダム・バックのハッシュキャッシュ[6]のようなシステムが必要である。これは、演算に費やした時間を他人に頼らずに証明する手法で、例えば「SHA-256 のようなハッシュ関数を通すと最初の n ビットがすべて 0 となるような値」を総当たりで発見することによって実現できる。そのような演算にかかる平均時間は、n に対して指数関数的に増大する。それに対して、検証する側はハッシュ関数を 1 回計算するだけで良い。

提案するタイムスタンプネットワークでは、ブロックデータと Nonce と呼ばれる数字を組み合わせる。Nonce を適当な値から 1 ずつ増やし続けて、ハッシュ値の最初の n ビットがすべて 0 になる場合を探索する。このように CPU を使って演算量を証明しておけば、ブロックの内容を改ざんして辻褄を合わせるためには、同じ計算をもう一度行わなくてはならなくなる。さらにその後ろにブロックがチェーンでつながっていたら、それらすべてに対する演算をやり直す必要がある。



演算量を証明することにより、集団における意思決定をどう行うべきかという問題も解決する。多数決方式を採用し各 IP アドレスに 1 票ずつを割り当てるような方法では、誰かが IP アドレスを大量に確保した時点で制度が崩壊してしまう。

それに対して、演算量証明に基づく本システムは、各 CPU に 1 票を与えるようなものである。実際には、最も CPU 能力を費やして作られた情報、すなわち最も長いチェーンを、集団の意思決定の結果であると取り決める。CPU 能力の大多数が善良なノードによって構成されていれば、不正がないチェーンが最速で成長し、ねつ造した情報を含むどのチェーンよりも長くなるはずである。もし、攻撃者が過去のブロックを改ざんしようとしたら、それ以降のブロックをすべて計算し直し、善良なノード群が計算するチェーンの長さに追いつき追い越す必要がある。後に示すが、計算能力的に劣る攻撃者が正しいチェーンに追いつける確率は、追いつくべきブロック数に対して指数関数的に小さくなる。

年月が経つと、ハードウェアの速度が向上したり、世間の注目度合いによって参加ノード数が変動したりするだろう。生成されるブロック数がそのような影響を受けないように、演算量証明の難易度(difficulty)が、過去一定期間の生成ブロック数をもとに設定される。ブロックの生成間隔が短すぎる時には、難易度が高くなる。

5　ネットワーク

本ネットワークは、次に示すステップで動作する。

1) 取引を行うと、その情報がすべてのノードに広められる。
2) 各ノードは、取引情報を集めてブロックを生成する。
3) 各ノードは、ブロックに対する演算量証明を開始する。
4) 演算量証明に成功した最初のノードは、そのブロックを全ノードに広める。
5) 各ノードは、ブロックを、多重使用が無く正しい取引だけを含むことを確認して、受け付ける。
6) 各ノードは、受け付けたブロックのハッシュ値を埋め込んで、次のブロックを生成し始める。これが、ブロックを受け付けたことの表明ともなる。

各ノードは、最も長いチェーンのみを正当なものと見なし、その最終ブロックから演算量証明を開始して、さらにチェーンを伸ばそうとする。2 つのノードが、内容の異なるブロックを同時に広めている場合は、受信順序の食い違いにより、演算量証明の対象

チェーンが集団で統一されていないかもしれない。その場合、各ノードは演算しない側のブロックも保存しておく。この引き分けの状況は、次に誰かが演算を終えチェーンを伸ばしたときに解消する。演算量証明に負けた側のチェーンを計算していたノードは、長いチェーンに乗り換える。

取引情報は、必ずしも全てのノードに行き渡る必要は無い。一定数のノードに情報が届き、ある程度の時間が経てば、ブロックに取り込まれるからである。ブロック化された後も、多少の通信障害には耐えて拡散する。ノードがブロックを受信できなかった場合は、次のブロックが届いたときにそのことを知り得るので、欠けたブロックを他のノードに要求できる。

6　ネットワーク参加者への報酬

ブロックの最初には、特殊な取引が記述される。ここに書かれるのは、コインが新たに生まれ、それをブロックの生成者が所有したという情報である。ブロックの生成者にコインを与えることは、各ノードがネットワークを維持する動機となり、また中央機関が権力を使ってコインを配分する代わりの枠組みともなる。新たなコインが一定量ずつ生まれることは、金鉱採掘に例えれば、資金を投入して金を採掘し流通に乗せることに該当する。資金に当たるのは、本システムでは CPU 時間や電力である。

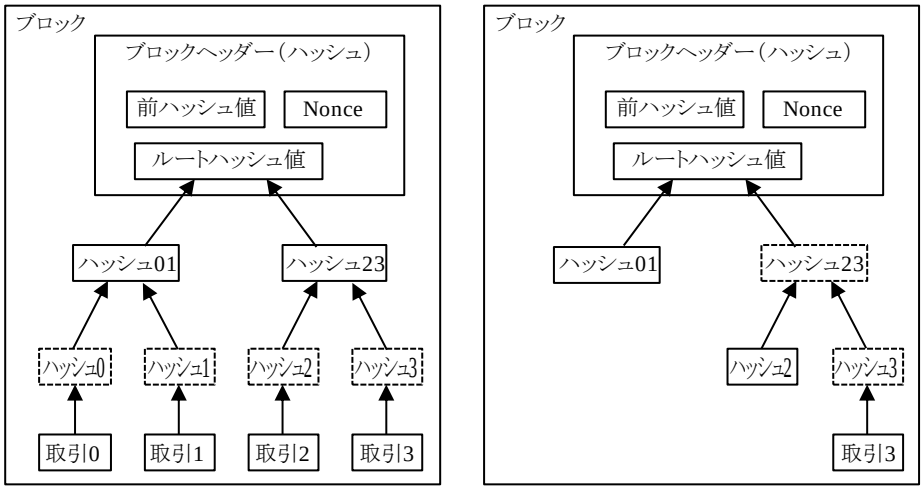
コインの生成者は、報酬として取引手数料も受け取れる。これは、取引上の受取額を支払額よりも少なく設定できれば良い。その取引を含むブロックを生成した者が、差額を手数料として得る。将来、決められた全枚数のコインが流通に乗った暁には、インフレから解放された状況が出来上がるが、それ以降は取引手数料がブロックを生成する動機となる。

報酬には、各ノードが不正を働かないように動機付ける役割もある。もし、欲張りな攻撃者が CPU 能力を集め、善良なノード群のそれを上回ったとする。攻撃者はその能力を、自身が支払った額を奪い戻して横領するか、真っ当な方法で新たなコインを生成するかのどちらかしか選択できない。この場合、ルールを破り通貨システムと自身が持つ価値を崩壊させるよりも、ルールに従って、新たに生まれるコインの過半数を得る方が賢明だと、判断するだろう。

7　使用ディスクスペースの節約

コインの取引履歴が十分な数のブロックに埋め込まれたら、使用ディスクスペースを節約するために、古い履歴を消去しても構わない。取引情報のデータ構造にはハッシュ木[7][2][5]を採用しているので、データを消去しても、ブロックのハッシュ値は変わらない。ハッシュ木の性質により、そのルートハッシュ値は取引データ全体のハッシュ値になる。この値はブロックのヘッダーに格納されている。古くなったブロックでは、ハ

ッシュ木の枝を刈り取ってサイズを縮小できるが、この際、途中の不要なハッシュ値も削除可能である。

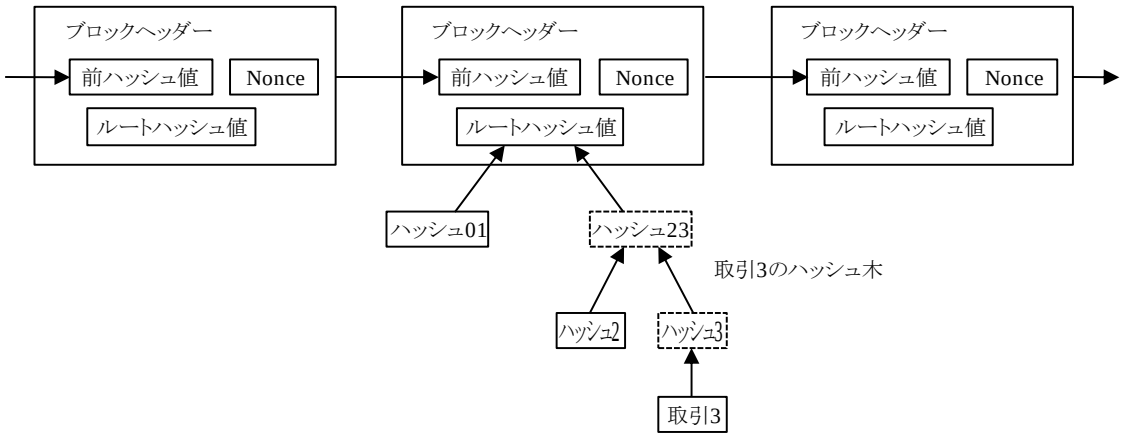


ハッシュ木に格納された取引情報　　取引0～取引2の木を刈り取ったブロック

ブロックのヘッダー部分の大きさは約 80byte である。従って、ブロックを 10 分間に 1 個ずつ生成する場合、1 年間に 80byte×6×24×365=4.2MB が必要になる。2008 年現在、一般的な PC は 2GB の RAM を搭載して販売されており、ムーアの法則によればこれが 1 年間に 1.2GB の勢いで成長している。そのため、ヘッダー部分だけならば、すべてのブロックをメモリに格納できるだろう。

8　取引の簡易検証法

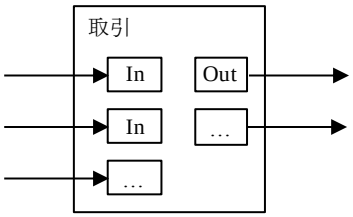
取引を検証するだけで良いなら、ノードの役割をすべて果たす必要はない。その代わりに普段は、最長チェーン内の各ブロックヘッダーを、それを確実に持っているノードに要求して手に入れておけば良い。そして検証時には、その取引を含むブロックのハッシュ木を要求する。ユーザ単独では取引を検証できないが、そのブロックを調べれば、ネットワーク上のいずれかのノードが取引を受け入れたと確認できる。また、その後にブロックが連なっていれば、ネットワーク全体が取引を承認したこともわかる。



善良なノード群がネットワークを制御できていれば、この検証法は信頼できる。しかし攻撃者の力が強くなると、信頼性は減少する。各ノードは自身で取引を検証できるので問題ないが、この簡易検証法しか使わないユーザは、攻撃者がネットワーク内で一定の演算能力を持つ場合に、ねつ造された取引を見破れない。これに対抗するには、不正ブロックを発見した善良なノードが、そのことをユーザに知らせるような仕組みが必要かもしれない。ユーザの画面に対して、ブロック全体をダウンロードして問題がある取引の整合性を確認するように、警告を伝えられれば良い。頻繁に支払いを受けるようなビジネス利用者は、セキュリティを確保するために素早い検証が必要なので、他者に頼らず自身でノードを設置すべきだろう。

9　コインの融合と分離

ビットコインでは、コインを一枚一枚別々に処理するので、そのままではすべての送金を最小単位に分割せざるを得ず、不便である。そこで、1 度の取引に複数のインプット（前回の取引への参照）と複数のアウトプット（支払先のビットコインアドレスと支払額）を設け、コイン額を融合・分離できるように作られている。あらかたの場合、インプットは、過去のより大きな取引への参照か、複数の小さな取引への参照を寄せ集めたものになるだろう。対してアウトプットは、受取人への支払いと、もしあればお釣りの返却とで成り立つ。



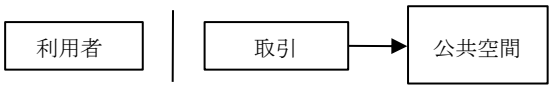
このように設計すると、一つの取引が複数の取引に依存し、それらの取引もさらに多くの取引に依存することになるが、問題ではない。コインの取引履歴を、1 本に連なったリストとして取り出す必要性は無いからである。

１０　プライバシー

既存の銀行モデルでは、取引情報へのアクセスを関係者や第三者信頼機関のみに限定することによって、ある程度のプライバシーを維持している。本システムでは、全ての取引を公開するのでこの手法は使えないが、情報の流れを別の方法でせき止め、プライバシーを確保する。それは、公開鍵を匿名にすることである。ある額を誰かが誰かに送金したことは全参加者に知られるが、他に情報がなければ、どういった人物が取引に関わったかは見抜かれない。株式市場で言えば、取引の時刻や額を示すティッカーテープが公になっても、誰が売買したかはわからないのと同じことである。

既存のプライバシーモデル

本システムのプライバシーモデル



さらなる安全策として、取引のたびに鍵のペアを生成し、参加者が取引とリンクされないようにしても良い。ただし、複数の出金が融合して一つになった取引では、出金者が同一である場合はそのことが知られてしまう。何らかの原因で、取引に使われた鍵と出金者とがリンクされた時には、その他の取引も知られてしまうというリスクはある。

１１　数学的根拠

正しいチェーンよりも速く、攻撃者が自身のチェーンを伸ばすには、どれ程の計算能力を持てば良いだろうか。ただしこれが成功しても、正当な量以上のコインを生成したり、扱ったことがないコインを他人から盗み出せるようになったりするわけではない。善良なノードは、不正な取引を受け付けず、また不正な取引が埋まったブロックも受け付けないからだ。攻撃者がなし得る詐欺行為は、他人に支払った過去の取引を無効にし、コインを奪い戻すことのみである。それも、取引から時間が経つほど困難になる。

善良なチェーンと攻撃者のチェーンとの闘いは、±1 歩ずつのランダムウォークと見なせる。善良なチェーンが 1 ブロック伸びた場合は 1 歩進み、攻撃者のチェーンが 1 ブロック伸びた場合は 1 歩下がる。

攻撃者のチェーンが、より長い善良なチェーンに追いつく確率は、「ギャンブラー破産問題」の理論を使って計算できる。ギャンブラーは一定額の赤字から開始するが、無限の資金を持ち何回でも賭け続けられるとする。そして、一度でも損益無しの状態になれば勝ちである。その確率は、攻撃者が善良なチェーンに追いつく確率と同じで、次のような式になる[8]。

p=善良なノードが先にブロックを生成する確率

q=攻撃者が先にブロックを生成する確率$(p + q = 1)$

q_z=zブロック遅れている攻撃者のチェーンが、一度でも善良なチェーンに追いつく確率

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

$p > q$という本システムの前提下では、攻撃者が追いつく確率は、追いつくべきブロック数が増えるほど指数関数的に小さくなる。従って、もし最初に運良く相手との差を詰められなければ、差が急激に開き、追いつく可能性はほとんど無くなるだろう。

支払人による取引改ざんを事実上不可能にするために、受取人はどの程度待てば良いだろうか。攻撃者は、受取人に対してしばらくの間正しく支払われたと思い込ませ、その後に払った額を奪い戻す。受取人は、いずれ不正な取引であるとの警告を受けるが、支払人はその時刻をできるだけ遅らせたい。

受取人は、取引のたびに鍵のペアを生成し、相手に公開鍵を送って間を開けずに支払うよう促す。さもないと、支払人が、チェーンを偶然何ブロックか伸ばせた瞬間を狙って、取引を実行するかもしれないからである。攻撃者である支払人は、取引を実行してから、それと異なる情報をブロックに含めて演算量証明を開始し、正規のものとは別にチェーンを伸ばし始める。

受取人が、自分の取引がブロックに埋め込まれ、その後にチェーンが z ブロック伸びたことを確認したとする。その間に攻撃者が影でどれだけチェーンを伸ばしたかは、分からない。しかし、善良なチェーンが設定通りの間隔でブロックを生成したと仮定したら、攻撃者が伸ばせるチェーンの長さはポアソン分布に従い、そのパラメータ λ は次のようになる。

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

攻撃者が追いつく確率は、開始時に攻撃者が k ブロックを計算し終えている確率（ポアソン分布の確率関数）に、$z-k$ ブロック差から追いつく確率を掛け、それをすべての k について足し合わせれば良い。

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

無限の項を足し合わせなくても良いように、式を変形する。

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

これを C 言語で記述すると、次のようになる。

```
#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;
}
```

具体的な数値を計算すると、攻撃者が追いつく確率は、z に対して指数関数的に減少することがわかる。

```
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
```

受取人はどの程度の時間を待つべきだろうか。攻撃者が改ざんできる確率を 0.1%未満に抑えたい場合は、次に示す個数のブロックを善良なノードが生成するまでである。

```
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

信頼関係を必要としない電子取引システムを提案した。まず、電子署名だけに頼った通貨を考えたが、所有権を強力に制御できるものの、多重使用を防げないことが判明した。この問題を解決するため、各利用者が取引を公開し、演算量証明を利用してその履歴を記録する P2P ネットワークを考案した。もし、善良なノードが CPU 能力の大半を占めていれば、攻撃者による偽造は計算論的にほとんど不可能である。本ネットワークの形は決まっておらず、素朴な仕組みなので、堅牢性に優れている。さらに、各ノードは協調せずばらばらに動いて良い。そしてすべての情報について、必ず届かなければな

らないという場所を設ける必要は無く、またベストエフォートで広がれば良い。つまり、ノードはすべて平等に扱える。各ノードはいつネットワークから離れ、再接続しても構わない。その間の演算量証明は他のノードから得られる。ネットワークは CPU 能力をもとに投票を実施し、現行のチェーンに対して、正しいブロックを追加し不正なブロックを拒否する。この同意メカニズムによりルールが守られ、ネットワークの維持に貢献した各利用者が報酬を得る。

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 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, no2, 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," 1957.9