ナノ：手数料不要で分散された仮想通貨ネットワーク

Colin LeMahieu
clemahieu@nano.co

要旨—近年、需要の増加と限られたスケーラビリティにより、人気のある仮想通貨の平均トランザクション時間と手数料は増加し、満足のいかない結果をもたらしている。本稿は、各アカウントが独自のブロックチェーンを所持することにより、ほとんど即座のトランザクション速度と無制限のスケーラビリティを可能とする新たなブロックラティス アーキテクチャを採用した仮想通貨、ナノ（Nano）を紹介する。各ユーザーが独自のブロックチェーンを持ち、ネットワークと非同期で更新することを許可することにより、結果として最小限の諸費用で高速なトランザクションを可能としている。トランザクションはその送金額ではなくアカウントの残高を追跡し、セキュリティを損なうことなく積極的なデータプルーニングを可能とする。現在まで、ナノネットワークは 420 万トランザクションをわずか 1.7GB の台帳サイズで処理しており、手数料が不要で、瞬時の送金は、コンシュマートランザクションにおける第一の仮想通貨となる。

索引用語—仮想通貨、ブロックチェーン、ナノ、分散台帳、デジタル、トランザクション

I. はじめに

ビットコインが 2009 年に実装されて以来、伝統的な政府によって保証された通貨や金融システムから、トラストレスで安全な方法により資金を保管、送金する能力を有する暗号学を基にした近代的な支払いシステムへの移行が増加している [1]。効率よく機能させるため、通貨は簡単に送金可能で非可逆であり、手数料が不要または限られているべきである。トランザクション時間の増加と高額な手数料、不確かなスケーラビリティは日々の決済手段としてのビットコインの実用性に対して疑問を抱かせる。

本稿は、革新的なブロックラティスデータ構造を基に構築した、無制限のスケーラビリティとトランザクション手数料不要を可能とするローレイテンシの仮想通貨、ナノを紹介する。ナノは高パフォーマンスな仮想通貨であることを唯一の目的としたシンプルなプロトコルで設計されており、このプロトコルは低パワーのハードウェア上で実行が可能である。従って、誰しもが使うことのできる実用的で非中央集権な仮想通貨であることを可能としている。

本稿で記載している仮想通貨の統計情報は公開日現在のものである。

II. 仮想通貨の背景

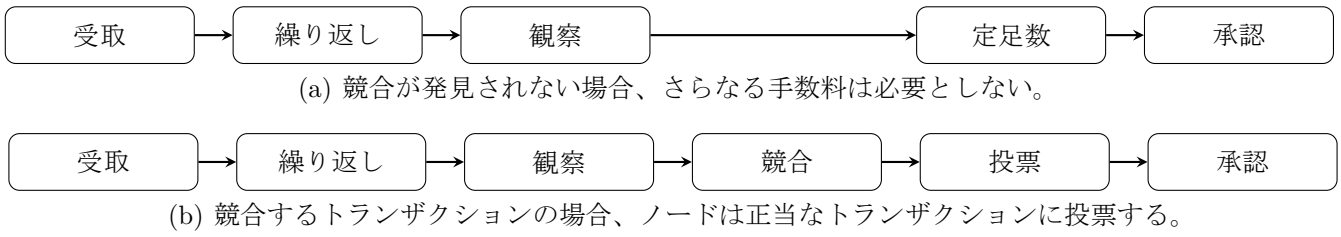
2008 年、ナカモトサトシという仮名で匿名の人物が、世界で初となる非中央集権仮想通貨のビットコインという概要を述べたホワイトペーパーを公開した [1]。ビットコインによりもたらされた重要な革新技術のブロックチェーンは、公に公開されており不変で、通貨のトランザクションの台帳として使用される分散データ構造である。残念ながら、ビットコインが成熟していくに連れ、プロトコルのいくつかの問題がビットコインの適応性を妨げている。

1) 低いスケーラビリティ：ブロックチェーンの各ブロックは限られたデータ量を保存することができる。つまり、このシステムは 1 秒間に限られたトランザクションしか処理することができず、ブロックを貴重な商品にしてしまった。現在、平均のトランザクション手数料は 10.38 ドルとなっている [2]。
2) ハイレイテンシ：平均のトランザクション承認時間は 164 分である [3]。
3) 非効率的な消費電力：ビットコインネットワークは概算で年間 27.28TWh の電力を消費し、トランザクション毎に 260kWh 使用している [4]。

ビットコインとその他の仮想通貨は、悪意のある攻撃者を防ぎつつ正当なトランザクションを検証するために、各自のグローバル台帳のコンセンサスを得ることで機能している。ビットコインはプルーフ・オブ・ワーク（以下 PoW とする）と呼ばれる経済対策によりコンセンサスを得る。PoW システムでの参加者は、ブロック全体のハッシュを目標範囲内に入れるようなノンスと呼ばれる解を演算するために競争を行う。この有効範囲は、正当なノンスを見つけるための一貫した平均時間を継続するため、ビットコインネットワーク全体の累積した演算能力に反比例する。この正当なノンスを発見した参加者はブロックチェーンへブロックを追加することを許可されるため、ノンスを計算するためにさらなる演算リソースを消費する参加者は、ブロックチェーンの状態において、より大きな役割を成す。PoW は 1 人のユーザーが非中央集権システムの中でさらなるパワーを得るため、まるで複数いるかのように振る舞うシビルアタックに対する抵抗力を備え、さらにグローバルなデータ構造に本来アクセスする際に本質的に存在する競合状態を大幅に軽減する。

対して代替案となるコンセンサスプロトコルのプルーフ・オブ・ステーク（以下 PoS とする）は 2012 年にピアコインによって導入された [5]。PoS システムでは、参加者がネットワーク内の仮想通貨で所有している財産に比例した重みを持ち投票を行う。この合意において、巨額な投資を行った参加者はより多くの決定権を与えられ、本質的にシステムの誠実さを維持し、さもなければその投資資産を失う。PoS は非経済的な演算能力の競争を廃止し、低パワーハードウェアでの軽いソフトウェアの実行のみを必要とする。

オリジナルのナノホワイトペーパーと β 版の実装は



(a) 競合が発見されない場合、さらなる手数料は必要としない。

(b) 競合するトランザクションの場合、ノードは正当なトランザクションに投票する。

図 1. ナノは典型的なトランザクションに追加の手数料を必要とせず、競合するトランザクションの場合、ノードはトランザクションを保有するため投票を行わなければならない。

2014 年 12 月に公開され、最初に非有効巡回グラフ（以下 DAG とする）を基にした仮想通貨の一つである [6]。その後間もなく、その他の DAG 仮想通貨の開発が開始され、中でも注目すべきは DAG コイン/バイトボールと IOTA [7], [8], である。これらの DAG を基にした仮想通貨は、ブロックチェーンの基準を打ち破り、システムパフォーマンスとセキュリティを向上した。IOTA が積み重ねられたトランザクションの累積的な PoW でコンセンサスを得るのに対し、バイトボールは誠実で評判が良く、ユーザーに信頼された“証人”から構成される“メインチェーン”に依存してコンセンサスを得る。

ナノは競合するトランザクションにおいてはバランス加重投票を介してコンセンサスを得る。このコンセンサスシステムは、強力な非中央集権システムは維持しつつ、より迅速で決定論的なトランザクションを提供する。ナノはこの開発を続け、最もパフォーマンスの高い仮想通貨の一つとして自身を位置づけている。

III. ナノの構成要素

ナノアーキテクチャ全体を記述する前に、システムを構成する個々の構成要素を定義する。

A. アカウント

アカウントは公開鍵と秘密鍵から成るデジタル署名の公開鍵部分である。公開鍵はアドレスとも呼ばれ、秘密鍵が公開されないのに対してそのほかのネットワーク参加者に共有される。電子署名されたデータパケットはそのコンテンツが秘密鍵の所有者によって承認されたことを保証する。一人のユーザーが多くのアカウントを所有することは可能であるが、アカウント毎に 1 つのパブリックアドレスしか存在できない。

B. ブロック/トランザクション

“ブロック”と“トランザクション”という用語はブロックが単一のトランザクションを含むため、よく同義語として使用される。具体的にはトランザクションはその行動を表すのに対し、ブロックはトランザクションのデジタルエンコードを表す。トランザクションは送信を行うアカウントが所有している秘密鍵によって署名される。

C. 台帳

台帳は各アカウントが独自のトランザクションチェーンを持つグローバルなアカウントセットである（図 2）。これは実行時の同意から、設計時の同意への置き換えに該当する重要な設計要素で、アカウント所有者のみが自身の独自チェーンを変更できる署名参照を介して、参加者全員が同意する。これは一見データ共有構造のように見える分散台帳を、非共有のセットへ転換する。

D. ノード

ノードはナノのプロトコルに従い、ネットワーク参加者のコンピュータ上で実行されるソフトウェアの一部である。ソフトウェアはもし台帳と任意の所有しているアカウントがあれば管理する。ノードは台帳全体か、各アカウントのブロックチェーンの最後の数ブロックを含む、プルーニングされた一部の履歴を保存する。新しいノードを建てる際は全体の履歴を検証し、ローカルでプルーニングすることが推奨される。

IV. システムのオーバービュー

その他のブロックチェーンを使用した多くの仮想通貨と違い、ナノはブロックラティス構造を使用する。各アカウントは自身のトランザクション/バランス履歴（図 2）と同等の独自のブロックチェーン（アカウントチェーン）を保持する。各アカウントチェーンはその所有者によってのみアップデートが可能で、これは各アカウントチェーンが即座にその他のブロックラティスと非同期でアップデートを行うことができ、迅速なトランザクションを実現している。ナノのプロトコルは極めて動作が軽いため、各トランザクションはインターネットを通じて

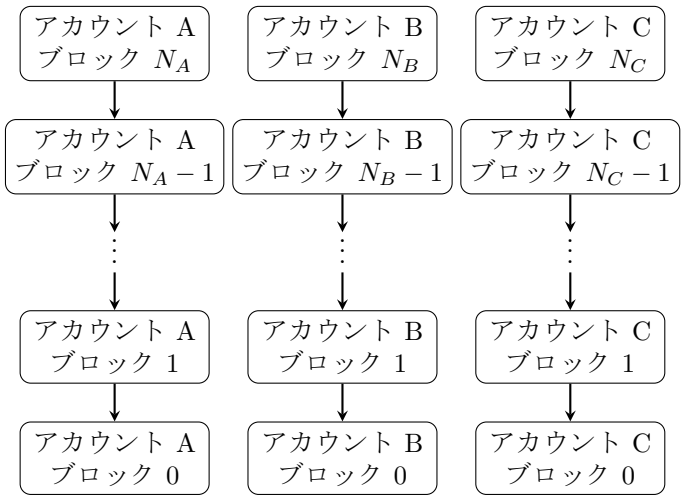


図 2. 各アカウントは自身のバランス履歴を含んだ独自のブロックチェーンを保有する。ブロック 0 は必ずオープントランザクションである必要がある（IV-B項）。

送信されるために必要とされる最小の UDP パケットサイズ内に収まる。また、ノードは大半のトランザクションでブロックを記録し、再度ブロードキャストする必要があるだけであるため、ハードウェア要求も同様に最小となる（図 1）。

システムはジェネシスバランスを含むジェネシスアカウントから開始される。このジェネシスバランスは固定された値であり、決して増やすことはできず、ジェネシスアカウントチェーン上に登録されている送信トランザクションを介してジェネシスバランスは分割され、その他のアカウントに送金される。全てのアカウントバランスの合計は、システムに上限値を与えている初期ジェネシスバランスを超えることはなく、この上限を増やす機能も存在しない。

本項では、いかにして異なるタイプのトランザクションがネットワークを介して構築され、伝達されるのかを段階的に説明を行う。

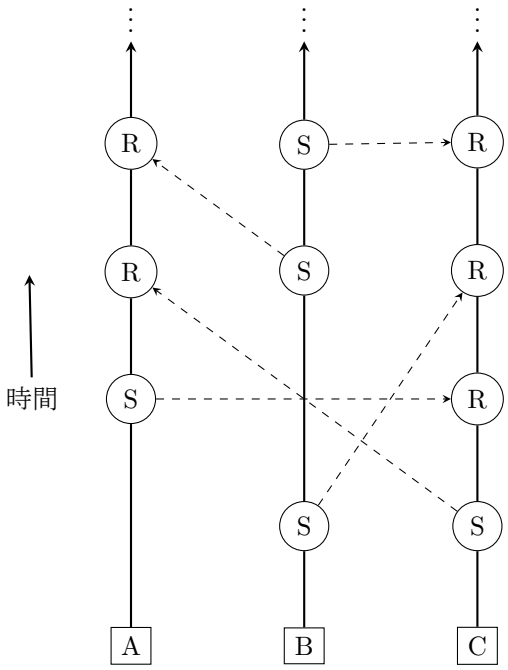


図 3. ブロックラティスの可視化。全ての資金移動は送金ブロック（S）と受信ブロック（R）、各アカウントチェーンの所有者（A, B, C）の署名を必要とする。

A. トランザクション

あるアカウントから他のアカウントへの資金送金は 2 つのトランザクションを必要とする。送信者は自身のバランスから送金額を差し引かれ、受信者はその送金額をアカウントのバランスへ追加する（図 3）。

送金金額を送金者と受信者のアカウントに別々のトランザクションとして分けるのには、下記のような重要な目的がある。

1) 次に送金される本質的に非同期なトランザクションを順序付けする。
2) トランザクションを UDP パケットに収まるように小さく保つ。
3) データの形跡を最小化することで台帳のプルーニングを容易にする。
4) 決済されたトランザクションと未決済分を隔離する。

1 つ以上の複数のアカウントからの同一のアカウントへの送金は非同期動作となり、ネットワークのレイテンシと送金アカウントは必ずしもお互いに通信する必要はない。従って、どのトランザクションが最初に生成されたかを知る普遍的で最適な方法は存在しない。追加は結合的であるため、オーダーのインプット順序は重要ではない。従って、単純にグローバルな同意が必要となる。これは実行時の同意を設計時の同意へ置き換える重要な設計要素となっている。受信者のアカウントはどの送金が最初に着金したかを管理でき、次のブロックの署名されたオーダーによって伝達される。

もしあるアカウントが多くの少額送金を一度に受け取る大きな送金を行いたい場合、UDP パケット内に収まる方法を紹介する。受信アカウントがインプット送金を配列する際、いつでも固定されたトランザクションサイズで任意の金額を送金する機能を有するため、そのアカウントの連続するバランスの合計を保持する。これはビットコインやその他の仮想通貨が使用しているインプット/アウトプットのトランザクションモデルとは異なる。

一部のノードは、あるアカウントの全てのトランザクション履歴を保存するリソースを拡大することに無関心で、各アカウントの現在のバランスにのみ関心を示す。アカウントがトランザクションを生成するとき、その蓄積されたバランスをエンコードし、ノードは最新のブロックの追跡を行えばよいため、正確性を維持しつつ履歴データを破棄することができる。たとえ設計時の合意に焦点を当てていても、ネットワークの問題を識別して対処を行うため、トランザクションの検証時に遅延機会が存在する。ナノの合意は約数ミリ秒から数秒で即座に形成されるため、ユーザーによく知られている次の 2 つのトランザクションの分類“決済済みと未決済”を示すことができる。決済済みトランザクションは、アカウントが受信されたブロックを生成したトランザクションであり、未決済トランザクションは受信者の累計バランスにまだ反映されていないことを表す。これは複雑性の置き換えで、そのほかの仮想通貨ではあまり見慣れない承認基準となっている。

B. アカウントを作成する

アカウントを作成するには、まずオープントランザクションを発行する必要がある（図 4）。オープントランザクションは常に全てのアカウントチェーンの最初のトランザクションであり、資金を最初に受け取った際に作成することができる。アカウントフィールドは署名に使用される秘密鍵から得られる公開鍵（アドレス）を保有し、ソースフィールドでは資金を送金するトランザクションのハッシュを含む。アカウントの作成において、ユーザーに代わって投票する代理人が選ばれる必要があり、これは後ほど変更することが可能である（IV-F項）。アカウントは自身を代理人として宣言することができる。

C. アカウントバランス

アカウントバランスは台帳自身の内部に記録される。トランザクションの送金額を記録するのではなく、検証（IV-I項）では送金ブロックと前のブロックのバランスの

```
open {
    account: DC04354B1...AE8FA2661B2,
    source: DC1E2B3F7C...182A0E26B4A,
    representative: xrb_1anr...posrs,
    work: 0000000000000000,
    type: open,
    signature: 83B0...006433265C7B204
}
```

図 4. オープントランザクションの詳細

確認を必要とする。受信アカウントは、受信ブロックにより与えられ計算された最終バランスに基づいて、以前のバランスを増加させることができる。これは高容量のブロックをダウンロードする際、処理速度を向上させるための措置である。また、アカウント履歴を要求する際は、送金額は既に入力されている。

D. アカウントから送金する

アドレスから送金を行うには、既に存在するオープンブロックを保持している必要があるため、バランスが必要となる（図 5)。以前のフィールドはアカウントチェーンの以前のブロックのハッシュを含んでおり、宛先フィールドはアカウントが送金する資金を含んでいる。送金ブロックは一度承認された後は不変であり、ネットワークへブロードキャストしたら、送金者のアカウントからそのバランスが差し引かれ、そのブロックが送金額を承認するために受取人が署名するまで未確認として待機する。送金アカウントと送金者はトランザクションを取り消すことができないため、未確認の送金は承認待ちとして考慮すべきではない。

```
send {
    previous: 1967EA355...F2F3E5BF801,
    balance: 010a8044a0...1d49289d88c,
    destination: xrb_3w...m37goeuufdp,
    work: 0000000000000000,
    type: send,
    signature: 83B0...006433265C7B204
}
```

図 5. 送金トランザクションの詳細

E. トランザクションの受取

トランザクションを完了するには、送金された資金の受取人が受信ブロックを自身のアカウントチェーンに生成する必要がある（図 6)。ソースフィールドは関連する送金トランザクションのハッシュを参照する。一度このブロックが生成されブロードキャストされると、そのアカウントバランスは更新され、資金は正式にアカウントへと送金されたことになる。

F. 代理人の割当

アカウント所有者は、自身に代わって投票するための代理人を選択する機能を有し、これは PoW または PoS

```
receive {
    previous: DC04354B1...AE8FA2661B2,
    source: DC1E2B3F7C6...182A0E26B4A,
    work: 0000000000000000,
    type: receive,
    signature: 83B0...006433265C7B204
}
```

図 6. 受信トランザクションの詳細

プロトコルに類を見ない強力な非中央集権ツールである。従来の PoS システムでは、アカウントの所有者は投票を行うためにノードを実行している必要があるが、ナノでは代理人に投票権を委託することでこの要求を緩和している。アカウント所有者はいつでも任意のアカウントにコンセンサスを再割当する機能を有する。変更トランザクションは古い代理人から投票加重を差し引き、新しい代理人へ加重されることによりアカウントの代理人を変更する（図 7)。このトランザクションでは資金移動はなく、代理人はアカウントの資金を使用する権限を持たない。

```
change {
    previous: DC04354B1...AE8FA2661B2,
    representative: xrb_1anrz...posrs,
    work: 0000000000000000,
    type: change,
    signature: 83B0...006433265C7B204
}
```

図 7. 変更トランザクションの詳細

G. フォークと投票

フォークは j がブロック $b_1, b_2, \ldots, b_j$ に署名した際に行なわれ、元チェーンと同じブロックを獲得する（図 8)。これらのブロックはアカウントの状態に相反する見解を引き起こすため、必ず解決しなければならない。唯一アカウントの所有者がアカウントチェーンのブロックに署名できるため、フォークは粗末なプログラミングかアカウント所有者による悪意のあるもの（二重支払い）に違いない。

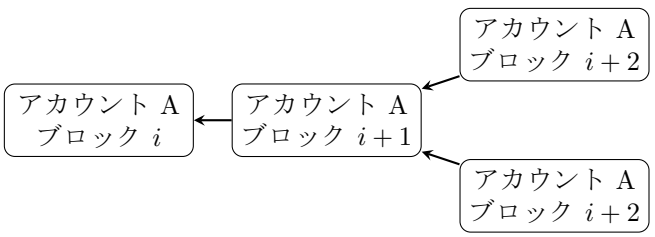


図 8. フォークは 2 つ（またはそれ以上）の署名されたブロックが同一の以前のブロックを参照した場合に起こる。古いブロックは左側で、新しいブロックは右側である。

フォークを検知して即座に代理人は台帳のブロック bi を参照して投票を開始し、ネットワークへブロードキャ

ストする。ノードの投票加重 wi は、その代理人としての指定された全てのアカウントバランスの合計となる。ノードはその他のオンラインである代理人 M から行なわれる投票を監視し、合計 1 分の 4 つの投票期間の累計を保有し、勝利したブロックを承認する（方程式 1)。

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

最も支持を得たブロック b* は投票の大多数を獲得した後、ノードの台帳へ保持される（方程式 2)。投票を得られなかったブロックは破棄され、もし代理人が台帳のブロックを置き換えた場合、より高い配列番号の新しい投票が作成され、ネットワークへブロードキャストされる。これは代理人が投票を行う唯一のシナリオとなる。

情況次第では、短時間のネットワークの接続問題によって送信されたブロックが全てのピアによって受け入れられない可能性がある。このアカウントのどの後続のブロックも、最初の送信を確認していないピアによって無効なブロックとして無視される。このブロックの再ブロードキャストは残りのピアによって受け入れられ、後続のブロックは自動的に読み込まれる。たとえフォークが発生する、またはブロックが行方不明になっても、トランザクションに参照されているアカウントのみが影響を受け、残りのネットワークはその他全てのアカウントのために処理中のトランザクションを処理する。

H. プルーフ・オブ・ワーク

4 つのトランザクションタイプは全て正しく占有されているワークフィールドを持つ。ワークフィールドでは、受信/送金/変更トランザクションの前のフィールドと連結している、またはオープントランザクション内のアカウントフィールドが一定の閾値以下となるようなノンスのハッシュをトランザクションの作成者が演算することができる。ビットコインとは異なり、ナノ上の PoW は単純に Hashcash と同様のスパム対策ツールとして使用され、約数秒で計算することが可能である [9]。一度トランザクションが送信されると、以前のブロックフィールドは既知であるため、後続のブロックのための PoW は事前に演算することが可能である。従って、PoW を演算するのに必要な時間よりトランザクション間の時間が長い限り、エンドユーザーに対して即座にトランザクションを表示することができる。

I. トランザクションの検証

有効とみなされるブロックは下記の特性を必ず持つ。

1) そのブロックはまだ台帳に書き込まれていない (二重トランザクション)。
2) 必ずアカウントの所有者によって署名されている。
3) 以前のブロックはアカウントチェーンの先頭のブロックで、もし先頭のブロックではないブロックが存在する場合はフォークとなる。
4) アカウントは必ずオープンブロックを持っている。
5) 演算されたハッシュは PoW の要求閾値を満たしている。

もしこれが受信ブロックの場合、ソースブロックハッシュが未確認の場合には確認を行う、つまり既に償還されているわけではないということを意味する。もし送金ブロックの場合、バランスは以前のバランスよりも少ない必要がある。

V. 攻撃ベクトル

ナノは全ての非中央集権の仮想通貨と同様に、経済的利益のため、またはシステムの破壊を試みる悪意のある勢力に攻撃される可能性がある。本項では何点かの起こりうる攻撃シナリオとその攻撃の結果、およびナノプロトコルが行う防止策を記述する。

A. ブロック相違同期

IV-G項では、ブロックが適切にネットワークに送信されず、ネットワークが以後のブロックを無視する可能性があるというシナリオを記述した。もしノードが以前のブロックを参照していないブロックを観測した場合、下記 2 つの選択肢がある。

1) 悪意のある攻撃者のブロックの可能性があるので無視する。
2) そのほかのノードに再同期を要求する。

再同期の場合、必要なトラフィック数の増加を容易にするため、ブートストラップノードと TCP 接続を確立する必要がある。しかしながら、もしブロックが実際に攻撃者のブロックであった場合、同期は不要であり、不必要にネットワークのトラフィックが増加する。これはネットワークアンプリフィケーション攻撃であり、サービス拒否を結果的に引き起こすことになる。

不必要な再同期を避けるには、ノードが同期を行うためにブートストラップノードとの接続開始の前に、悪意のある可能性があるブロックに対する投票の一定の閾値が観測されるまで待機する。もしブロックが十分な投票数を受け取らなかった場合、そのブロックは無意味なデータと見なすことができる。

B. トランザクションの洪水

悪意のある攻撃者は、所有するアカウント間で無駄ではあるが有効である大量のトランザクションを送信することにより、ネットワークを飽和状態にすることが可能となる。手数料が無料の場合、この攻撃を無制限に行うことができるが、各トランザクションに必要な PoW が、悪意のある攻撃者が大いに演算リソースへ投資をすることなく生成するトランザクションレートを制限している。たとえこのような攻撃で台帳を膨張させようとしても、全てのトランザクション履歴を持たないノードは、自身のチェーンから古いトランザクションを切り離すことができるため、このようなタイプの攻撃からほぼ全てのユーザーに対しストレージの使用を抑制する。

C. シビルアタック

ユーザーは単一のデバイスで、何百ものナノノードを生成することができる。しかし、投票システムはアカウントバランスを基に加重されるため、さらなるノードをネットワークに追加したとしても攻撃者は新しい投票権を得ることはできない。故に、シビルアタックによるアドバンテージを得ることはできない。

D. ペニー消費攻撃

ペニー消費攻撃は、ノードのストレージリソースを消費させるため、攻撃者が多数のアカウントに対して無限小量を消費する攻撃である。ブロックの発行は PoW によって制限されている、故にアカウントの作成とトランザクションは幾分か制限されている。全てのトランザクション履歴を持たないノードは、有効でない確率の高い統計的基準以下のアカウントをプルーニングすることができる。最後に、ナノは最小限の永続ストレージスペースを使用するように調整されているため、追加のアカウントを保存するのに必要なスペースは、オープンブロック＋インデックス化＝ 96B ＋ 32B ＝ 128B のサイズに比例する。これは 1GB に 800 万個のペニー消費アカウントを保存できることになる。もしノードが積極的にプルーニングを望む場合、アクセス頻度を基にして分布を計算し、使用頻度の低いアカウントに低速なストレージを付与することができる。

E. プリコンパイルされた PoW 攻撃

アカウントの所有者は、アカウントチェーンにブロックを追加する唯一のユーザーになるため、連続的なブロックをネットワークへブロードキャストする前に自身の PoW と並行して演算することができる。ここで攻撃者は無数の連続した最小の値を持つブロックを長期間に渡って生成する。ある時点で、攻撃者は多数の有効なトランザクションにより DoS 攻撃を行い、その他のノードは可能な限り早く処理し反映する。これはV-B項で記述したトランザクションの洪水の上級版で、このような攻撃は一時的な効果しか持たないが、>50% 攻撃（V-F項）のようなその他の攻撃の効率を高めるために同時に使用される可能性がある。この攻撃を緩和するため、トランザクションのレート制限とその他の手法が調査中である。

F. >50% 攻撃

ナノのコンセンサス基準は、バランス加重投票システムである。もし攻撃者が 50% を超える投票力を得ることが可能となった場合、コンセンサスを揺さぶり、システムを不安定な状態にすることができる。攻撃者は DoS 攻撃を行うことにより正常なノードが投票することを防ぐことで、失わなければならない攻撃コストを減少させることができる。ナノはこのような攻撃を防止するため下記のような対策を講じている。

1) このようなタイプの攻撃に対する第一の防衛は、システムに対する投資に結び付けられた投票加重である。アカウント所有者は本質的に自身の投資した資産を守るため、システムを正常に維持するインセンティブを持つ。台帳を改ざんする試みは、システム全体にとって破壊的な状態となり、投資した資産を破壊することになる。
2) 攻撃コストはナノの時価総額に比例する。PoW システムでは、金銭的な投資に比べ不相応な制御をもたらす技術が創案されることがあり、もし攻撃が成功するとその技術が再利用される可能性がある。ナノではシステム規模の攻撃コストはシステム自体と比例するため、もし攻撃が成功してもその攻撃に対して使用されたコストを取り戻すことはできない。
3) 最低限の投票の定足数を維持するため、次の防衛ラインは代理人投票である。接続の問題などで投票を確実に行うことができないアカウントの所有者は、自身のバランスの加重で投票ができる代理人を指名することができる。代理人数と多様性を最大化することでネットワークの弾性を強化している。
4) ナノにおけるフォークは決して偶発的なものではないため、ノードはフォークしたブロックの取扱いについて方策決定を行うことができる。攻撃者ではないアカウントがフォークしたブロックに対して脆弱性を持つのは、攻撃者のアカウントからバランスを受け取ったときのみである。フォークブロックから安全であることを望むアカウントには、フォークブロックを生成したアカウントからバランスを受け取る前に少し待つか、長時間待つまたは一切受け取らないという選択肢がある。このバランスを受け取ったアカウントは、その他のアカウントから隔離するために使用する別のアカウントを、疑わしいアカウントから資金を受け取った際に生成することもできる。
5) まだ実装されていないが、最終防衛ラインはブロックの結合である。ナノは投票を行うことで、フォークブロックを解決するため手を尽くしている。ノードは一定期間後にロールバックを行うことを防止するための結合ブロックを形成することができるようになる。曖昧なフォークを防ぐための迅速な修正時間に着目することで、ネットワークは十分に安全が確保されている。

より精巧な >50% 攻撃の詳細を図 9に示す。“オフライン”は、指定されてはいるが、投票するためにオンラインになっていない代理人の割合を示す。“ステーク”は攻撃者が投票を行うために投資した金額を示し、“アクティブ”はプロトコルに従って投票を行うオンラインになっている代理人を示す。攻撃者は DoS 攻撃を行い、その他の投票者をオフラインにすることで、失うはずだったステークを差し引きすることができる。もしこの攻撃が持続可能な場合、攻撃されている代理人は非同期となり、“非同期”と示される。最後に、攻撃者は代理人が古い台帳と再同期している間に新しい代理人へ DoS 攻撃を切り替えることによって、相対的な投票力を一気に得ることができる。これは“攻撃”と示される。

オフライン	非同期	攻撃	アクティブ	ステーク

図 9. >50% 攻撃要件を下げる可能性のある潜在的な投票配列。

もし攻撃者がこのような情況を組み合わせ、ステーク、ステークの費用を犠牲にして台帳の投票を覆すことができる。このような攻撃コストは、そのシステムの時価総額を調べることで見積もることができる。もし 33% の代理人がオフラインである、または DoS 攻撃を受けていると仮定した場合、攻撃者は投票でシステムを攻撃するため時価総額の 33% を購入する必要がある。

G. ブートストラップ被毒

攻撃者が長期的に古い秘密鍵とバランスを保持することが可能であるほど、バランスまたは代理人が新しいア

カウントへ移転しているため、そのとき存在していたバランスには参加している代理人を持たない可能性が高い。これはもしノードがその時点で代理人と比較して、定足数の投票ステークを攻撃者が得ているネットワークの古い状態へとブートストラップされた場合、そのノードの投票の決断を変動させることが可能ということを意味する。もしこの新たなユーザーが、攻撃ノードに加え誰かと協力したいとしても、全てのトランザクションは先頭のブロックが異なるため拒否される。最終結果として、ノードは悪い情報をネットワークの新しいノードへ与え、時間を無駄にさせることができる。これを防止するため、ノードはアカウントの初期データベースと既知の正当な先頭ブロックと組み合わせることが可能である。これはデータベースをジェネシスブロックまで遡ってダウンロードを行うことの代替手段となる。ダウンロードが最新に近いほど、この攻撃に対して正確に防衛する確率が高まる。結局のところ、攻撃者は最新のデータベースを持つノードとは取引を行えないため、この攻撃はおそらくブートストラップ中のノードへのジャンクデータ送信より悪くはない程度となる。

VI. 実装

現在の実装リファレンスは C++ で実装されており、2014 年移行のリリースは Github で行なわれている [10]。

A. 設計の特徴

ナノの実装はこのホワイトペーパーで記述しているアーキテクチャ基準に準拠している。追加の仕様を下記にて述べる。

1) 署名アルゴリズム: ナノは改良版 ED25519 楕円曲線アルゴリズムを使用し、全ての電子署名を Blake2b でハッシュ化している [11]。迅速な署名と検証、高いセキュリティのために ED25519 が選択された。

2) ハッシュ化アルゴリズム: ハッシュ化アルゴリズムはネットワークスパムを防ぐためだけに使用されているため、アルゴリズムの選択はマイニングを基にした仮想通貨と比較して重要性は低い。ナノの実装では Blake2b をブロックコンテンツに対するダイジェストアルゴリズムとして使用している [12]。

3) 鍵導出関数: リファレンスウォレットでは、ASIC によって解析を試みることを防ぐため、キーはパスワードで暗号化され、鍵導出関数により与えられる。現在のところ、Argon2 [13] は弾性のある鍵導出関数を生成することを目的とした公の競技会で優勝した唯一の鍵導出関数である。

4) ブロックインターバル: 各アカウントが独自のブロックチェーンを所有しているため、ネットワークの状態と非同期で更新することができる。従って、ブロックインターバルは存在せず、トランザクションは即座に発行される。

5) UDP メッセージプロトコル: ナノのシステムは、可能な限り最小限の演算リソースを使用して無期限に運用できるように設計されている。全てのシステムのメッセージは、ステートレスで単一の UDP パケット内に収まるように設計されている。これはまた、断続的な接続を行うライトピアが、短期的な TCP 接続を確立することなく、ネットワークへ接続することを容易にする。TCP は新しいピアがブロックチェーンを大量にブートストラップしたいときのみ使用される。

ノードは自身にエコーバックされた何点かのコピーが確認されるように、そのほかのノードからトランザクションのブロードキャストトラフィックを監視することで、ネットワークからトランザクションを受け取ったことを確認できる。

B. Ipv6 とマルチキャスト

コネクションレス型通信の UDP 上に構築することで、伝統的なトランザクションの殺到と投票のブロードキャストの代替えとして、Ipv6 マルチキャストを将来的に実装することが可能である。これによりネットワーク帯域の消費を減少し、今後ノードの方策に柔軟性を与える。

C. パフォーマンス

このホワイトペーパーを記述している現在は、420 万トランザクションがナノネットワークで処理されており、ブロックチェーンサイズは 1.7GB となっている。トランザクションの処理時間は約数秒と測定されている。現在のリファレンス実装は SSD 上では主に I/O バウンドで、秒間 10,000 トランザクションを処理できる。

VII. リソース使用

ここではナノノードが使用するリソースの概要を記述し、さらに特定のユースケースでリソース使用を減少させるアイデアを提案する。リソース使用を減少させたノードは通常、ライトノード、プルーニングノード、簡易支払い検証 (SPV) ノードと呼ばれる。

A. ネットワーク

ネットワーク活動の量は、いかネットワークが健全性に貢献しているかにより異なる。

1) 代理ノード: 代理ノードは、その他の代理人の投票トラフィックを監視し、自身の投票を発行している際、ネットワークの最大のリソースを要求する。

2) トラストレスノード: トラストレスノードは代理人ノードと類似しているが、監視者に過ぎない。また、代理人の秘密鍵を所有しておらず、さらに自身の投票を発行しない。

3) 信用ノード: 信用ノードは単一の代理人からの投票トラフィックを監視し、正しくコンセンサスを実行していることを信頼する。これは、このノードに送られる代理人からの投票トラフィックの量を減少させる。

4) ライトノード: ライトノードは最小限のネットワーク使用を可能にするアカウントのトラフィックのみを監視する信用ノードである。

5) ブートストラップノード: ブートストラップノードは、オンラインになっているノードの全ての台帳、または一部を提供する。これは高度なフロー制御を要求する大容量のデータを含むため、UDP ではなく TCP 接続で行われる。

B. ディスク容量

ユーザーの需要に応じて、ノード構成の違いによりストレージ要求が異なる。

1) 履歴ノード: 全てのトランザクションの完全な履歴を保有したいノードは最大のストレージ容量を必要とする。

2) 現在のノード: 蓄積したバランスとブロックを保持する設計により、ノードはコンセンサスに携わるために、各アカウントの最新または先頭のブロックのみを保有しておけばよい。もしノードが全ての履歴を保持したくない場合、先頭のブロックのみを保有するという選択が可能である。

3) ライトノード: ライトノードはローカルの台帳を保持せず、関連したアカウントの動向を監視するためだけにネットワークに参加するか、任意に所有する秘密鍵で新しくトランザクションを生成する。

C. CPU

1) トランザクション生成: 新しいトランザクションの生成を行いたいノードは、ナノのスロットルメカニズムをパスするために PoW のノンスを生成しなければならない。個々別々のハードウェアでの計算は付録 Aで評価している。

2) 代理人: 代理人はブロックの署名と投票を検証する必要があり、さらにコンセンサスに参加するため独自の署名を生成する。代理ノードのための CPU リソース量はトランザクションの生成よりも大幅に少なく、現代のどのコンピュータの単一の CPU でも動作するはずである。

3) 監視者: 監視ノードは自身の投票は行わない。署名生成の費用は最小限であるため、CPU 要求は代理ノードの実行とほとんど同一となる。

VIII. 結論

本稿では、新しいブロックラティス構造と委任した PoS 投票を使用した、トラストレスで手数料が不要かつ、ローレイテンシな仮想通貨のためのフレームワークを記述した。本ネットワークのリソース要求は最低限で、高出力なマイニングハードウェアを必要とせず、高いトランザクションスループットを処理できる。これは各アカウントが個別のブロックチェーンを保持し、接続問題とグローバルなデータ構造の非効率性を排除したことによって達成されている。また可能性のあるシステム上の攻撃ベクトルを確認し、ナノがいかにしてこのような攻撃に耐性を持つかを提示した。

付録 A
PoW ハードウェアベンチマーク

前述のように、ナノの PoW はネットワークスパムを減少させ、ノード実装は OpenCL に適合する GPU を活用できるアクセラレーションを提供する。表 Iは様々なハードウェアのベンチマーク比較を表している。現在、PoW の閾値は固定されているが、平均的な演算力の向上に応じて適応性の閾値が実装される可能性もある。

表 I
PoW ハードウェアベンチマーク

デバイス	秒間のトランザクション数
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

謝辞

ホワイトペーパーの作成と資料収集を行った Brain Pugh 氏に感謝の意を表する。

参考文献

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] “Bitcoin median transaction fee historical chart.” [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html
[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.
[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.
[8] S. Popov, “The tangle,” 2016.
[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf