The tangle

Serguei Popov[*] for Jinn Labs

August 11, 2017. Version 1.1

概要

本論文で，IOTA（IoT 業界の為の暗号通貨）のバックボーンとして利用されているテクノロジーを分析する．その主な特徴は，ブロックチェーンに変わって，トランザクション保存のために，*tangle*(すなわち有向非巡回グラフ) を使うことである．このテクノロジーはブロックチェーン技術の自然な後継であり，進化の次のステップである．グローバルな規模で行われるマイクロ決済に必要な機能とともに公開される．

特に，この論文の貢献の一つは，新たに到着するトランザクションがアタッチすべきタングルのサイトを選択する MCMC アルゴリズムの族である．

1　システムの紹介と説明

この 6 年間のビットコインの勃興と成功はブロックチェーン技術の価値を証明するものであった．だが，このテクノロジーにはいくつも欠点があり，ただ一つの各暗号通貨のグローバルプラットフォームとして利用される事を妨げている．それらの一つ，特記すべきはマイクロペイメント（少額決済）が不可能であることである．マイクロペイメントは急速に発展しつつあるモノのインターネット（IoT）業界において重要度が増している．特に，現行のシステムではトランザクション（取引）を実行する際，フィー（手数料）を支払わなければならない．ゆえに，ビットコインやブロックチェーンは，ほんの少額の送金を行うにあたって合理性に欠ける．なぜなら，送金額の何倍ものフィーを払うことになるからである．その一方，フィーを撤廃するのは簡単でなく，フィーこそがブロックの生成者にとってのインセンティブ（誘因）であるからである．また，現行の暗号通貨は役割（トランザクションの発行者やトランザクションの承認者）が明確に分離されたヘテロジニアス（異種混在）システムであることも観察される．そのようなシステムは一部の要素に対し，不可避的な差別を作り出し，それがコンフリクト（衝突）を生み，すべての要素がコンフリクトの解決に

[*] a.k.a. `mthcl`; author's contact information: `e.monetki@gmail.com`

リソースを費やさなければならなくなる．これら全ては，ビットコインや他の多くの暗号通貨が基盤とするブロックチェーン技術とは違うソリューションの探求を正当化するものである．

本論文では，「ブロックチェーンレス」アプローチについて検討される．これは現在 *iota*[1] において実装されているものである．IOTA は最新の IoT 業界のための暗号通貨として設計された．ただし，IOTA の実装の具体的な詳細よりも，ブロックチェーンを取り除こうとしたときに発生する一般的な機能や問題点や難点に焦点を当てる．

一般的に，タングルベースの暗号通貨は次のような仕組みである．グローバルなブロックチェーンの代わりに，*tangle* と呼ばれる DAG（= directed acyclic graph，有向非巡回グラフあるいは閉路を含まない有向グラフ）がある．ノードによって発行されたトランザクションはタングルのサイト集合（siteset）を構成する（すなわちタングルグラフはトランザクションを保管する台帳である）．そのエッジ集合（edgeset：枝の集合）は次のように取得される：新たなトランザクションが到着すると，前にある 2 つ[*1]のトランザクションを承認しなければならない．これらの承認は図 1 等にある通り，有向エッジ（directed edge：矢印）によって表わされる（図において時間は常に左から右へである）．もしトランザクション A とトランザクション B の間に有向エッジがない場合でも，少なくとも「2」の長さの有向パス（$A \to X \to B$ となる道順）がある場合，これを「A は B を 間接的に承認する」と呼ぶ．また「ジェネシス」トランザクションがあり，これは他の全てのトランザクションによって（直接的または間接的に）承認される．図 2 を参照のこと．創始フェーズ（genesis）は次のように記述される．まず初めに全てのトークンを残高として持つアドレスがあるとする．次にジェネシストランザクションがその全てのトークンを他のいくつかの「ファウンダー」アドレスに送信する．全てのトークンが創始フェーズにおいて生成され，その後，トークンの生成はなく，また，マイニングは「マイナーが金銭的報酬を受け取る」という意味では存在しないことを強調したい．

用語について一言：サイト (*sites*) とは，タングルグラフに表されるトランザクションである．ネットワークは ノードから構成されている．ノードがトランザクションを発行する主体である．

さて，主なアイデアを次に示す：トランザクションを発行するには，ユーザーは他のトランザクションを承認するための仕事をする必要があり，それがネットワークのセキュリティに貢献する．各ノードは承認されたトランザクションがコンフリクトしていないことを確認し，コンフリクトするトランザクションを（直接間接的を問わず）承認しないことを前提とする[*2]．トランザクションがもっと多くの（直接間接的の）承認を得るにつれ，システムによってより受け入れられるようになる．別の言い方をすると，二重支払いのトランザクションがシステムによって受け入れられるのは

[*1] これは最もシンプルなアプローチである．トランザクションが一般的な $k \geq 2$ となる k 個の別のトランザクションを承認しなければならない類似のシステムやもっと複雑なルールも研究されたい

[*2] コンフリクトするトランザクションを承認する新たなトランザクションをノードが発行する場合，他によって承認されずよって忘れ去られるリスクを冒すことになる

より難しく（または実際的に不可能と）なる．承認されるトランザクションについていかなるルールも課していないことを見て取るのは重要である．むしろ，他の多数のノードがある種の「参照」ルールに従った場合（ファームウェアが事前にインストールされている特殊なチップを搭載したノードである IoT のコンテキストにおいて妥当な仮定に思われる)，そのとき，すべての固定ノードにとって同種のルールに常に従うほうが良いと私達は主張する[*3]．

もっと具体的には，トランザクションを発行するにはノードは以下を行う：

- 最初に，どれかのアルゴリズムに従って別の承認する 2 つのトランザクション（一般に，これらの 2 つのトランザクションが一致することがある）を選ぶ．
- この 2 つのトランザクションがコンフリクトしていないかチェックし，コンフリクトしているトランザクションは承認しない．
- トランザクションが有効であるためには，ノードはビットコインマイニングと似た暗号学的パズル（ヘビーな計算を要するもの）を解く必要がある（例：あるノンスのハッシュと承認されたトランザクションからのデータを合わせたものが特定の形（例えば一定数のゼロが先頭に来る形）を取るようなノンスを見付けなければならない)．

一般に，非同期ネットワークなので，各ノードは必ずしも同じトランザクションの集合を見ていない事を観察するのは重要である．またタングルがコンフリクトしているトランザクションを含む場合があることも言及に値する．各ノードはどの有効なトランザクション[*4]が台帳に載せられる権利があるかについてコンセンサスを達成する必要はない（全て台帳に載せることが可能だ）が，コンフリクトしているトランザクションがある場合，どちらのトランザクションが「孤児」となるべきか決める必要がある（後のトランザクションによって承認されなくなるという意味である)．2 つのコンフリクトしているトランザクション間の決定に使用する主要ルールは次の通り：ノードがチップ選択（tip selection）アルゴリズム[*5]（4.1 章を参照のこと）を何度も実行し，2 つのうちどちらのトランザクションが選択されたチップによって（間接的に）承認される可能性が高いかを見る．例えば，チップ選択アルゴリズムを 100 回実行した後，あるトランザクションが 97 回選択された場合，「97% の信頼度で確認された」と言う．

また，次の質問についてコメントしたい ([4] を参照のこと)：「各ノードがトランザクションを増殖させる動機付けはどこにあるのか？」実際，私達の設定では各ノードにはトランザクションを増殖させる動機付けはない．全てのノードがいくつかの統計情報を計算しており，そのうちの一つは

[*3] これについては 4.1 章の終わりで更にコメントする

[*4] 規約に従って発行されたトランザクションのこと

[*5] 上記で言及されたように，他のノードがチップ選択のために（ほぼ）同じアルゴリズムに従うと仮定する十分な理由がある

隣人のノードから受け取った新しいトランザクションの数である．ある特定のノードが「怠けすぎ」の場合，隣人のノードによって drop，外されるので，ノードがトランザクションを発行しなくても (従って，自身を承認する新しいトランザクションを共有する直接のインセンティブがない場合でも)，「勤勉に働く」インセンティブは依然として存在する．

この先の各章では，2 章でいくつかの表記法を導入した後，どの 2 つのトランザクションを承認するかを選択するアルゴリズム，トランザクション承認全般を測定する規則 (3 章，特に 3.1 章)，そして起こりうる攻撃シナリオ (4 章) について議論する．また，数式に恐れを抱く（考えにくいが) 場合，読み手は数式などを読み飛ばして各章の末尾の「結論」に直行することも出来る．

暗号通貨のコンテキストにおける DAG の利用というアイデアがしばらく前からあったことは付言されるべきである ([3, 6, 7, 9, 12] を参照のこと)．具体的には，[7] の取り組みにおいて，GHOST プロトコルが導入された．これはビットコインプロトコルの修正版であり，主台帳をブロックチェーンの代わりにツリーにしたものである．そのような修正は確認の回数を減らし，ネットワーク全体のセキュリティを改善することが示された．論文 [9] の著者達は，DAG ベースの暗号通貨モデルを検討している．ただし私達のモデルとは異なり，DAG のサイト (sites) は（個別のトランザクションでなく）ブロックで，マイナー達がトランザクションのフィーを求めて競い，（ビットコインと同じく）新たなトークンが生成されうるものである．また観察すべきは，[6] の取り組みで，私達のと類似性を持つソリューションが提案されている．ただし特定のチップ承認戦略について議論されてはいない．この IOTA のホワイトペーパーの最初のヴァージョン 0.6 が公開された後，いつかの他の研究（[8] など）が登場している．また P2P 決済チャネルの確立によってビットコインでのマイクロペイメントを可能にする [2, 10] のアプローチにも言及しておく．

2　加重 等

ここで，トランザクションの加重そして関連の概念を定義する．トランザクションの加重は，トランザクションを発行するノードがそのトランザクションに「投資」した仕事量に比例する．実際には加重は 3^n の各値を取ることが出来る（現在の IOTA の実装形式である)．n は正の整数で，許容可能な値の非空なインターバルに属する[*6]．実際には，殆どの場合，実際にどのように加重が得られたかは無関係である．すべてのトランザクションに正の数（= 加重）がついていることだけが重要である．一般的には，加重が大きいほど，トランザクションがタングルに対してより「重要」な意味を持つ，というアイデアである．通常，スパムやさまざまな種類の攻撃を避けるため，短期間で “ 許容可能な ” 加重を持つあまり多数のトランザクションを生成することはできない．

[*6] このインターバルは有限でもあるべきである—4 章の「大きな加重攻撃」を参照のこと

私達にとって必要な概念の一つは，トランザクションの累積加重である．それは，このトランザクション自身の加重に加え，直接間接的に私達のトランザクションを承認する全てのトランザクションの自身の加重の合計として定義される．この累積加重の計算アルゴリズムは図 1 に示されている．各ボックスはトランザクションを表す．ボックス内右下隅の数字はトランザクション自身の加重を示す．一方，大きめで太字の数字は累積加重である．例えば，トランザクション F は，トランザクション A，B，C，E によって直接間接的に承認される．トランザクション F の累積加重は $9 = 3 + 1 + 3 + 1 + 1$（F の加重と A，B，C，E の加重の合計）となる．

図 1 の上半分において，承認されていないトランザクション（tips,「チップ」) は A と C だけである．新たなトランザクション X が現れ A と C を承認すると，X が唯一のチップとなる．そして他の全てのトランザクションの累積加重は 3（X の加重）だけ増加する．

承認アルゴリズムの議論については，他の変数を導入する必要がある．まず，タングル側（= トランザクション）に導入する：

- 高さ (height)，ジェネシスへ方向付けられた最長のパスの長さ
- 深さ (depth)，どれかのチップへ（つまりジェネシスとは逆の方向に）方向付けれられた最長のパスの長さ

例えば，図 2 で，G は 1 の高さで 3 の深さを持つ一方，(逆パス F，B，A のため）D は 3 の高さで 2 の深さである．また，スコアと言う概念を導入したい．トランザクションのスコアは，このトランザクションによって承認された全てのトランザクションの加重と自身のトランザクションの加重の合計として定義される．図 2 を参照のこと．なお，唯一のチップは A と C のみである．トランザクション A は（直接間接的に）トランザクション B，D，F，G を承認する．よって A のスコアは $1 + 3 + 1 + 3 + 1 = 9$ となる．同様に C のスコアは $1 + 1 + 1 + 3 + 1 = 7$ となる．

実際，このホワイトペーパーの主張を理解するために，すべての重みが 1 に等しいと仮定しても差し支えない．なので，以降，この仮定のみ考慮する．特に，累積加重は私たちのトランザクションを直接間接的に承認したトランザクションの個数に 1 を足したものである．この時スコアは私たちのトランザクションが直接間接的に承認したトランザクションの個数である．

また，上記のメトリクス（加重，累積加重，高さ，深さ，スコア）の内，累積加重が私達にとってもっとも重要であることを見て取っていただきたい（ただし高さ，深さ，スコアも一部の議論で扱われる)．

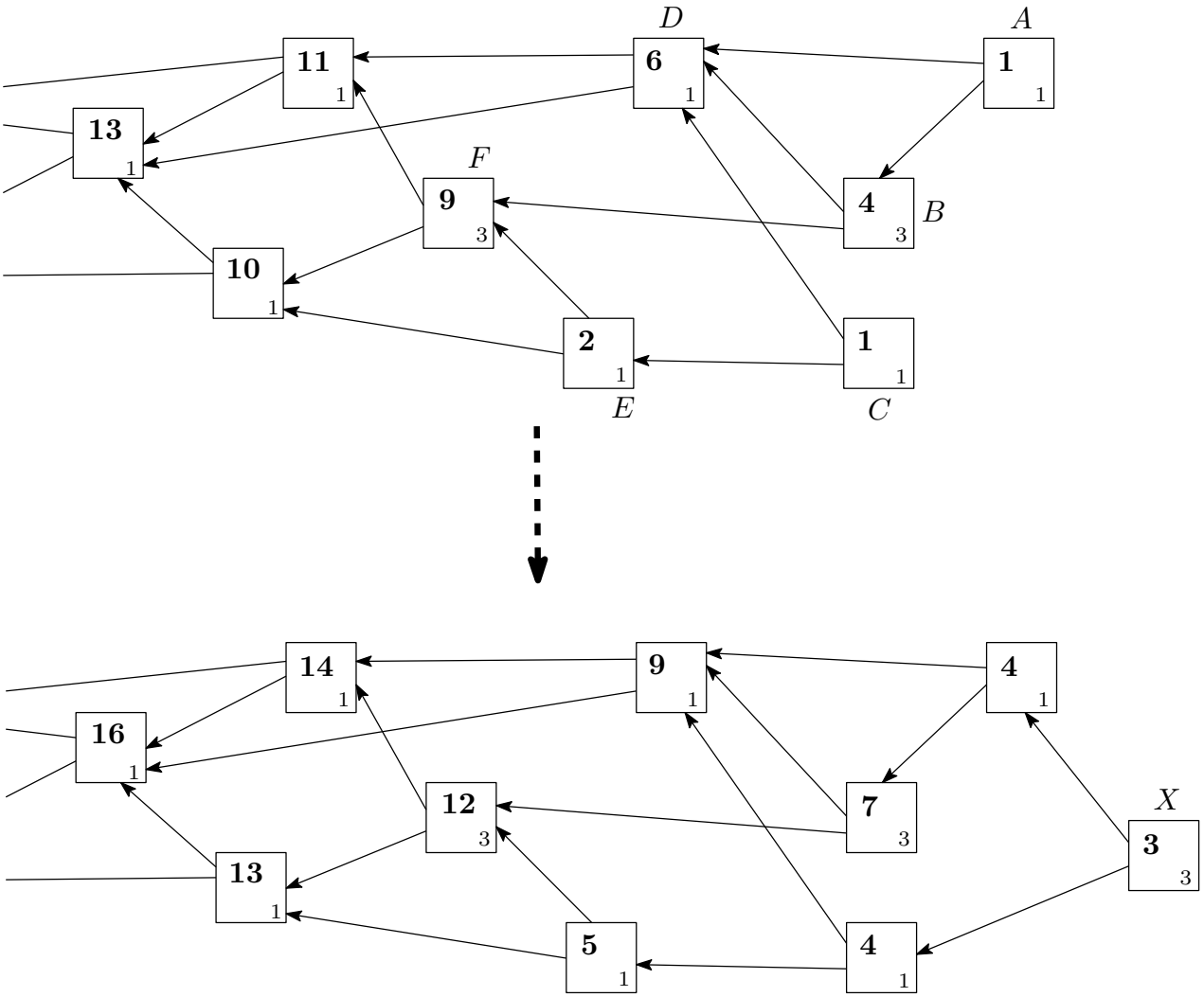

図 1　加重の (再) 計算について

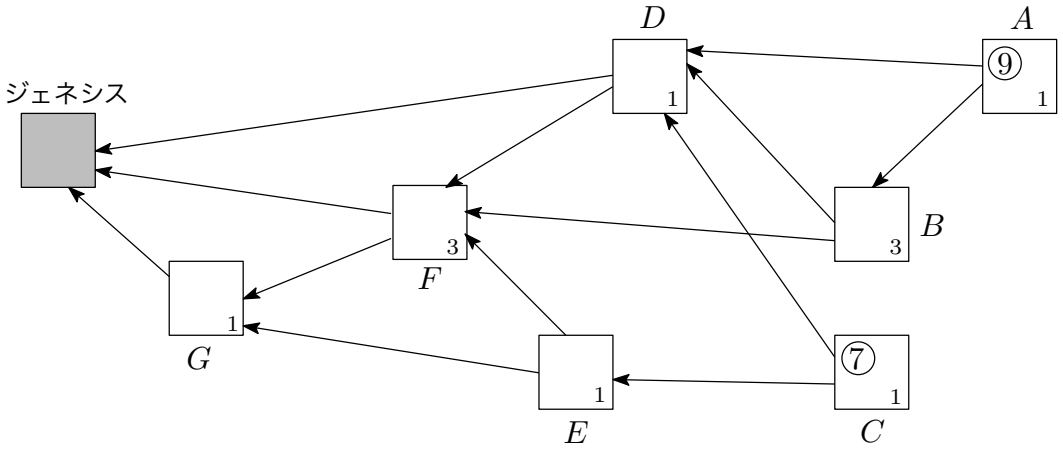

図 2　トランザクション A と C のスコアの計算について

3　システムの安定性とカット集合

仮にシステムの t の時点において，$L(t)$ をチップの合計数（承認されていないトランザクションの数）とする．当然，確率過程 $L(t)$ は安定にとどまる[*7]と期待される．（より正確には，正再帰．正式な定義については [11] の 4.4 章と 6.5 章を参照のこと．特に，正再帰は $\mathbb{P}[L(t) = k]$ as $t \to \infty$ の極限が存在すべきで，全ての $k \geq 1$ について正であるべきことを暗に意味する）．直観的に，$L(t)$ が定数のあたりで変動し，無限にエスケープしない（その結果沢山の承認されていないトランザクションを残さない）ことを私達は期待する．

$L(t)$ の安定性特性 (stabilityproperties) を分析するために，いくつか仮定する必要がある．1 つはトランザクションが多数のだいたい独立な実在物によって発行されることであり，そのため入力されるトランザクションのプロセスはポアソン点過程によってモデル化できる (例えば [11] の 5.3 章参照のこと)．仮に λ をこのモデル化したポアソン過程における単位時間あたりのトランザクションの期待発生回数とする．単純化のため，これは定数であるとする．全てのデバイスがほぼ同等の演算処理能力を持つこと，L のチップがあり，トランザクションの合計数が N である状況においてデバイスがトランザクションを発行するために必要な計算を行うのに必要な平均時間を $h(L, N)$ とする．そして，すべてのノードは，次のように振る舞うことを仮定しよう：まず，ノードがトランザクション発行の際，単に 2 つのチップをランダムに選択し承認する．一般に「正直なノード」がこの戦略を適用するのは良いアイデアではない．これはいくつかの実用上の欠点があり，特に，「怠惰」なノードや悪意のあるノードに対して十分な防御にならないからである[*8](下の 4.1 章を見よ)．

であるが，分析が簡単であるためこの戦略を熟考する．そうすればより複雑なチップ選択戦略のためのシステムの挙動に関する洞察を得ることができよう．この戦略では，異なるチップを承認するポアソンフローは独立であり，$2\lambda/L$ の速度を持つと仮定できる（これは [11] の命題 5.3 に従うものである．いくつかのありえるサブタイプのリストに従って，ポアソン過程の各事象を独立に分類すると，各サブタイプの事象の過程は独立したポアソン過程となる）．よって，

$$\mathbb{P}[\text{誰も与えられたチップを時間 } h(L,N) \text{ の間承認しない}] = \exp\Big(-\frac{2\lambda h(L,N)}{L}\Big). \tag{1}$$

これは私たちのデバイスがトランザクションを発行した時に予想されるチップの総数の増加が次式に等しいことを意味する．

$$1 - 2\exp\Big(-\frac{2\lambda h(L,N)}{L}\Big). \tag{2}$$

[*7] プロセスが時間的に均一（斉時的）であるという追加の仮定に基づく

[*8] 攻撃者に都合の良いチップ選択をさせてしまが，私達が特定のチップ選択戦略を押し付けようとしていないことにご注意いただきたい

上の数式において，「1」はトランザクションによって新たに生成されたチップに対応し，第二項は「消された」チップの予想される数である．実際，式 (1) により（さらにノードは少なくとも 2 つの承認可能なチップがある場合，常に 2 つの異なるチップを選択することも前提とする），承認のために始めに選択された各チップはトランザクションが確率 $\exp\left(-\frac{2\lambda h(L,N)}{L}\right)$ で発行された時点では依然としてチップのままである．

今，$L(t)$ は実際，最も近い隣のトランザクションとの $\mathbb{N} = \{1, 2, 3, \ldots\}$ 上の連続時間ランダムウォークである．確かに，2 つの選択されたトランザクションが既に他によって承認された場合，プロセスは 1 つ右にジャンプし，選択されたトランザクションが両方とも承認されなかった場合，プロセスは 1 つ左にジャンプし，最後の可能性としては，同じ所に留まる．

今ここで，このプロセスの典型的な振る舞いを理解するために，式 (2) における傾向（すなわちチップ数の増加）が小さな $L(t)$ については正であり，大きな L については負である（少なくとも $h(L, N) = o(L)$ as $L \to \infty$ の場合，あるいは計算処理/伝播時間への主な貢献がチップを扱うことに由来していないと仮定して）事を観察しよう．言い換えると，$L(t)$ が小さい時増加傾向にあり，大きい時減少に向かう．故に，$L(t)$ の「典型的な」値 L_0 は式 (2) が 0 になり（そのためチップ数は増加または減少する明確な傾向を持たない），次式となる．

$$L_0 \approx \frac{2\lambda h(L_0, N)}{\ln 2} \approx 2.89 \cdot \lambda h(L_0, N). \tag{3}$$

明らかに，上に定義された L_0 はチップの集合の典型的なサイズでもある．また，トランザクションの初めての承認に予想される時間はおおよそ $L_0/2\lambda$ である．

（少なくともノードがチップを承認しようと試みる場合）任意の固定時間 t において，$s \in [t, t + h(L_0, N)]$ のときにチップであったトランザクションの集合が一般的にカット集合を構成することを観察されたい（時間 t より後に発行されたトランザクションからジェネシスへの任意のパスがこの集合を通過しなければならないという意味においてである）．カット集合のサイズが時々小さくなることは重要である；この小さなカット集合を DAG 剪定（せんてい）やその他のタスクのためのチェック・ポイントして利用することが出来る．

これは観察するのに重要なことであるが，上記の “純粋にランダムな” 承認戦略は実際の場面においてあまり良いものとはいえない．なぜなら，チップの承認を奨励しないからである．「怠けたい」ユーザーはかなり古い一対のトランザクションを常に承認することができ（よって最近のトランザクションの承認に貢献しない），そのような行動によって罰せられることもない[*9]．また，悪意のあるものは，新たなトランザクションが非常に高い確率で悪意のあるチップを選択し，「正直なノード」

[*9] 攻撃者に都合の良いチップ選択をさせてしまが，私達が特定のチップ選択戦略を押し付けようとしていないことにご注意いただきたい

に属するチップを放棄するように，(例えば，固定された一対のトランザクションを承認する多くのトランザクションを発行することによって）チップの数を人工的に膨らませることができる．この種の問題を避けるためには，「より良い」チップに対して偏った戦略を採用する必要がある．そのような戦略の 1 つの例は，以下の 4.1 章に示さる[*10]．

トランザクションが初めて承認されるのにかかると予想される時間について議論を始める前に，原則的に 2 つの型を区別できる事を観察されたい（図 3 を参照のこと）．

- 低負荷：正常なチップの数が小さく，頻繁に 1 になりさえする．これはトランザクションのフローが十分に小さく，複数の異なるトランザクションが同じチップを承認する可能性が低い時に起こりうる．また，ネットワークの通信遅延がとても短く，各デバイスの演算が高速な場合，トランザクションのフローがかなり大きい状況でも多数のチップが現れる可能性は低い．また，人工的にチップの数を膨らませようとする攻撃者がいないことを前提としなければならない．
- 高負荷：正常なチップの数が大きい．トランザクションのフローが十分に大きく，ネットワークの通信遅延とともに演算遅延が発生し，複数の異なるトランザクションが同じチップを承認する可能性が高い時に起こりうる．

もちろん，この分類は略式的なものであり，2 つの型の間に明確な境界線はない．それでもこれらの原則的に異なる 2 つの型を考えることは示唆に富むと思われる．

低負荷型において，状況は比較的簡単である．最初（または最初の一つ）の入ってきたトランザクションは直ぐに私達のトランザクションを承認するだろうから，最初の承認は位数 λ^{-1} の平均時間で起こる．

ここで，高負荷な状況，すなわち L_0 が大きい状況を考えてみる．上述したように，異なるチップへの承認のポアソンフローは独立しており，およそ $2\lambda/L_0$ の速度を有すると仮定してもよい．したがって，トランザクションが最初に承認される期待時間は，約 $L_0/(2\lambda) \approx 1.45h$（式 (3) を思い出そう）である．しかし，より精妙な（「より良い」品質のチップを支持する）承認戦略のために，トランザクションが他から承認されるまで多くの時間を受動的に待つのは良い考えでないことは注目に値する．これは「より良い」チップは現れ続け，承認において優先されるからである．むしろ，トランザクションが承認を待っている時間が長すぎる（すなわち L_0/λ よりもはるかに長い）場合は，空のトランザクション[*11]を追加してトランザクションを促進することが良い戦略である．つまり，私

[*10] 実際，著者の思いはチップ承認戦略こそがタングルベースの暗号通貨構築のための最も重要なパーツであるということである．そこにいくつも攻撃ベクトルが隠れている．また，特定のチップ承認戦略を強制する方法がないので，各ノードが，他ノードの多数が従うことを知った上で自発的にそうするようなものでなければならない．

[*11] 空のトランザクションはトークンの転送を含まないトランザクションである．それでも，空のトランザクションは 2

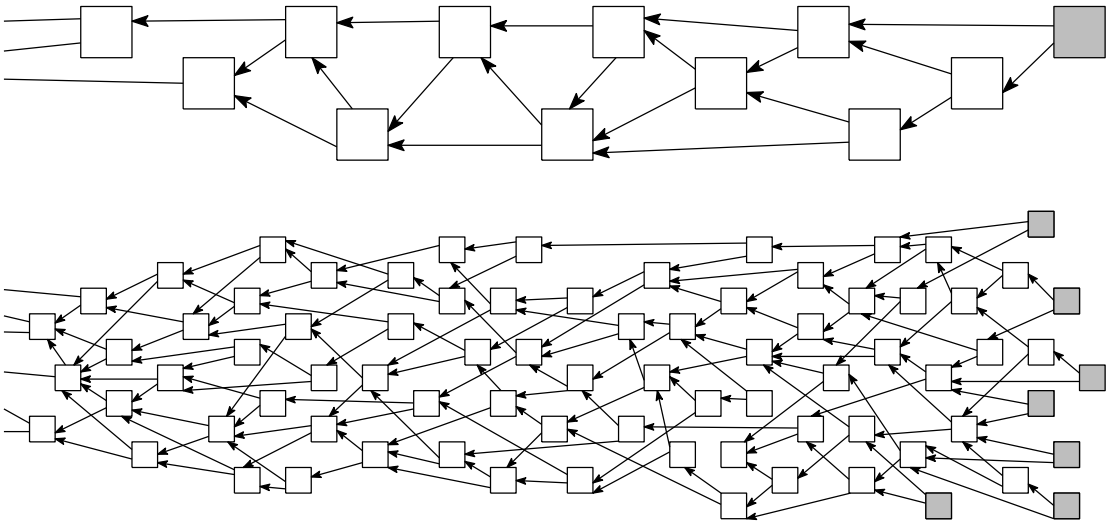

図 3　低負荷と高負荷の型においてのタングルとその典型的なチップ集合（灰色）．高負荷において，一部のトランザクションは初めて承認されるまでにかなり待つ場合があることを観察されたい．

たちの古いトランザクションを「より良い」チップの 1 つと共に承認する空のトランザクションを発行する戦略である（この空のトランザクションはさらに承認される可能性が高い）．

今ここで，高さとスコアに基づく承認戦略は，特定のタイプの攻撃に対して脆弱であるかも知れないことが明らかになっている．4.1 章を参照のこと．私達はそのような攻撃に対し防御するより精巧な戦略[*12]についてそこで議論していく．にも関わらず，最も単純なチップ選択戦略（「ランダムに 2 つのチップを選択」）は依然として検討の価値がある．最も分析が容易で，従ってシステムの定性的および定量的な振る舞いについて一定の知見を与えうるからである．

■結論：

1. 私達は図 3 に描かれたように，2 つの型，低負荷と高負荷を区別する．
2. 低負荷型において，通常多くのチップはなく（1 つか，2 つ），チップは $\Theta(\lambda^{-1})$ で，初めて承認される．λ は入ってくるトランザクションのフローの速度である．
3. 高負荷型において，典型的なチップの数は承認戦略（どのように新たなトランザクションが承認のために別の 2 つのトランザクションを選択するか）による．

つの他のトランザクションを承認しなければならず，従ってネットワークセキュリティに貢献する．

[*12] 実際，著者の思いはチップ承認戦略こそがタングルベースの暗号通貨構築のための最も重要な要素であるということである．そこにいくつも攻撃ベクトルが隠れている．また，特定のチップ承認戦略を強制する方法がないので，各ノードが，他ノードの多数が従うことを知った上で自発的にそうするようなものでなければならない．

4.「ランダムに 2 つのチップを選択」戦略において，典型的なチップの数は式 (3) によって与えられる．この戦略が典型的なチップの数に対して最適であることが示され得る．ただし，この戦略の採用は実用性に欠ける．なぜならチップの承認を奨励しないからである．
5. ただし，より精巧な戦略が必要とされる．そのような一連の戦略が 4.1 章で説明される．
6. 高負荷型において，チップが承認されるまでの典型的な時間は $\Theta(h)$ である（h はノードの平均計算/伝播時間）．ただし，上記の時間内に最初の承認が起こらなかった場合，追加の空のトランザクションでトランザクションを促進するのは（発行者/受領者にとって）良いアイデアである．

3.1 通常どれくらい早く累積加重が成長するか?

明らかに，低負荷型において私達のトランザクションが何度か承認された後，その累積加重は λ のスピードで成長する．それは原則的に全ての新しいトランザクションが間接的に私達のトランザクションを参照するようになるためである．[*13]

同様の理由で高負荷型については，トランザクションが十分に古く大きな累積加重を持つ場合，累積加重は速度 λ で成長する．また，始めにおいて承認されるまでしばらくの間トランザクションは待たなければならないことがあるとわかっているが，その時点でトランザクションの累積加重がランダムな振る舞いをすることは明らかである．トランザクションがいくつかの承認を得た後でどのような振る舞いをするのかを見るため，(単純化のため，私達のトランザクションが作成された瞬間から時間をカウントする)$H(t)$ によって時間 t における累積加重の期待値を示し，$K(t)$ によって時間 t における私達のトランザクションを承認するチップの期待数（または単に「私達のチップ」）を示す．

$h := h(L_0, N)$ という短縮表記も用いる．また，時間内に，全体のチップの数がほぼ定数（L_0 に等しい）という単純化された仮定も行う．ここでは「2 つのチップをランダムに承認する」という戦略を用いて議論する．定性的な動作は他の妥当な戦略とほぼ同じ結果になると期待される．時間 t の時点でシステムに到来するトランザクションが，システムの状態に基づき時間 $t-h$ の時点で，通常 2 つの承認するトランザクションを選択することを観察されたい（なぜならノードは実際にトランザクションを発行する前に何らかの計算や確認をせねばならないからである）．少なくとも「私達の」チップの一つを承認する確率が $1-\left(1-\frac{K(t-h)}{L_0}\right)^2 = \frac{K(t-h)}{L_0}\left(2-\frac{K(t-h)}{L_0}\right)$ であることを得るのは難しくない（左辺の式は承認された 2 つのチップが私達のものでない確率を 1 から引いたもの）．

[*13] 全てのトランザクションの加重が 1 に等しいと仮定したことを思い出していただきたい．したがって，累積加重はちょうど私達のトランザクションを（直接若しくは間接的に）参照した全てのトランザクションの数に 1 を加えた数である．

例えば [11] の例 6.4 と相似して，次のように書くことが出来る．

$$H(t+\delta) = H(t) + \lambda\delta\frac{K(t-h)}{L_0}\Big(2 - \frac{K(t-h)}{L_0}\Big) + o(\delta),$$

従って次の微分方程式を演繹できる．

$$\frac{dH(t)}{dt} = \lambda\frac{K(t-h)}{L_0}\Big(2 - \frac{K(t-h)}{L_0}\Big). \tag{4}$$

式 (4) を利用できるためには，まず $K(t)$ を計算する必要がある．$t-h$ の時点でのチップが t の時点でのチップでないかも知れなので，どのように行うかすぐには明確でなく，新たに到来したトランザクションがそのようなチップを承認した場合，元のトランザクションを承認するチップの全体の数は 1，増加する．今ここで，欠かすことの出来ない洞察は，$t-h$ の時点のチップが t の時点でチップであり続ける確率は約 $\frac{1}{2}$ である（実際に式 (3) を式 (1) に代入すればよい）．つまり，t の時点で事実上 $K(t-h)$ の「前の」チップの半分がチップであり続け，残りの半分が最低 1 度は既に承認されている．仮に p_1 を新たに到来したトランザクションが B から少なくとも 1 つのトランザクションを承認し，A からは全くトランザクションを承認しない確率とする．また，仮に p_2 を両方の承認されたトランザクションが A に属する確率とする．明らかに，p_1 と p_2 はそれぞれ「私達の」チップの現在の数が新たなトランザクションが到来した際に 1 増加する，または 1 減少する確率である．私達が持っているのは

$$\begin{aligned} p_1 &= \Big(\frac{K(t-h)}{2L_0}\Big)^2 + 2\times\frac{K(t-h)}{2L_0}\Big(1 - \frac{K(t-h)}{L_0}\Big), \\ p_2 &= \Big(\frac{K(t-h)}{2L_0}\Big)^2 \end{aligned}$$

（最初の式を得るには，p_1 が両方の承認されたチップが B に属する確率と最初のチップが B に属し 2 つ目のチップが $A\cup B$ に属さない確率の 2 倍に等しいことを観察する）．式 (4) と類似して，$K(t)$ の微分方程式は次のように表すことができる．

$$\frac{dK(t)}{dt} = (p_1 - p_2)\lambda = \lambda\frac{K(t-h)}{L_0}\Big(1 - \frac{K(t-h)}{L_0}\Big). \tag{5}$$

式 (5) を厳密に解くのは難しいので，更に単純化された仮定を行う．まず，私達は $K(t)$ が固定された，$\varepsilon > 0$ について εL_0 のレベルに到達した時，ほぼ $(1-\varepsilon)L_0$ までかなり早く成長すると気付く (実際，式 (5) の右辺は，0 から離れることになる)．

さて， $K(t)$ が L_0 に対して小さい時，私達は式 (5) の右側の最後の係数を「落とす」ことができる（少なくとも 1 に近い定数になるので右辺は次のようになる $\lambda\frac{K(t-h)}{L_0}$）．

従って，以下の式 (5) を簡略化したヴァージョンを得る（$\frac{\lambda h}{L_0} = \frac{1}{2}\ln 2$ であることを思い出すこと）．

$$\frac{dK(t)}{dt} \approx \frac{\ln 2}{2h} K(t-h), \tag{6}$$

ここで境界条件は $K(0) = 1$ である．$K(t) = \exp(c\frac{t}{h})$ の形の解を考えるため，式 (6) に $K(t) = \exp(c\frac{t}{h})$ を代入して，次式が得られ，

$$\frac{c}{h}\exp\left(c\frac{t}{h}\right) \approx \frac{\ln 2}{2h}\exp\left(c\frac{t}{h} - c\right),$$

よって，次式が近似解となる．

$$K(t) = \exp\left(W(\frac{1}{2}\ln 2)\frac{t}{h}\right) \approx \exp\left(0.27\frac{t}{h}\right) \tag{7}$$

ここで $W(\cdot)$ はランベルト W 関数[*14]と呼ばれるものである．

よって，式 (7) において対数をとると，$K(t)$ が εL_0 に達する時間はおおよそ次式であることが分かる．

$$t_0 \approx \frac{h}{W(\frac{1}{2}\ln 2)} \times (\ln L_0 - \ln \varepsilon^2) \lesssim 3.76 \cdot h \ln L_0. \tag{8}$$

式 (4) に戻り（先ほどと同様右側の最後の係数を「落とす」），私達は「適応期間」の間（t_0 が式 (8) の場合 $t \le t_0$），次が成立することを得る．

$$\begin{aligned}\frac{dH(t)}{dt} &\approx \frac{2\lambda}{L_0}K(t-h)\\ &\approx \frac{\ln 2}{h\exp\left(W(\frac{1}{2}\ln 2)\right)}\exp\left(W(\frac{1}{2}\ln 2)\frac{t}{h}\right)\\ &= \frac{W(\frac{1}{2}\ln 2)}{h}\exp\left(W(\frac{1}{2}\ln 2)\frac{t}{h}\right)\end{aligned}$$

従って，

$$H(t) \approx \left(W(\tfrac{1}{2}\ln 2)\frac{t}{h}\right) \approx \exp\left(0.27\frac{t}{h}\right). \tag{9}$$

読み手には，上で議論されたとおり，適応期間の後，累積加重 $H(t)$ は λ のスピードで原則的に線形成長することを思い出していただきたい．私達は「指数関数的成長」(式 (9) にあるような) は，適応期間の間に累積加重が「とても早く」成長することを意味しないことを強調したい．むしろ，振る舞いは図 4 に示されるとおりである．

[*14] オメガ関数もしくは対数積として知られている．$x \in [0, +\infty)$ の時，$x = W(x)\exp(W(x))$ の特性を持つ

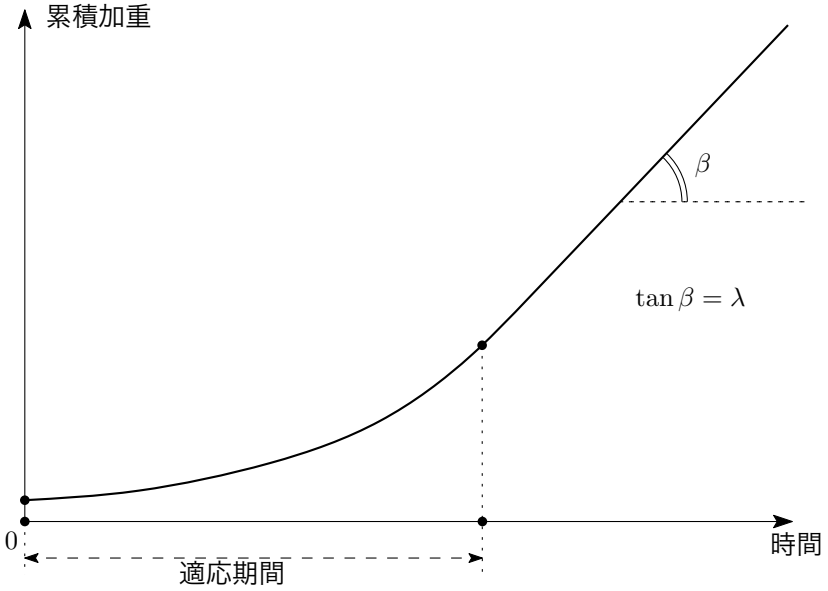

図 4　累積加重の成長

■結論：

1. 低負荷型において，私達のトランザクションが数回承認された後，その累積加重は λw のスピードで成長していく．ここで w は一般的なトランザクションの加重平均である．
2. 高負荷型において，再び，私達のトランザクションが数回承認された後，まずその累積加重 $H(t)$ が式 (9) による，いわゆる適応期間の間，増加するスピードで成長し，適応期間が終わった後，λw のスピードで成長する．図 4 を参照のこと．事実，全ての合理的な戦略では，適応期間の終了後，累積加重はこのスピードで成長していく．なぜなら原則的に全ての新たに到来したトランザクションは間接的に私達のトランザクションを承認するからである．
3. 適応期間は，現在のチップのほとんどが（間接的に）私達のトランザクションを承認するまでの時間と考えることが出来る．適応時間の典型的な長さは式 (8) によって与えられる．

4　起こりうる攻撃シナリオ

私たちは明らかな攻撃シナリオについて議論することから始める．攻撃者はタングルネットワークを追い越そうとする．

1. 攻撃者は商人に支払い，商人がトランザクションが既に十分に大きい累積加重を得たと見な

した後，商品を受け取る．

2. 攻撃者は二重支払いのトランザクションを発行する．
3. 攻撃者は多数の小さなトランザクション（持てる演算処理能力を結集して非常にアグレッシブにトランザクション）を発行する．これは直接的にも間接的にも元のトランザクションを承認しないが，二重支払いのトランザクションを承認する．
4. 攻撃者が多数のシビルアイデンティティを持つ（Sybil Attack）かもしれないこと，また必ずしもチップを承認する必要がないことも観察しよう．
5. 3 とは異なり，攻撃者は持てる演算処理能力を結集して，正当なトランザクション（商人に支払うために使用される）の前に 2 つのトランザクションを承認する「大きな」二重支払いトランザクション（非常に大きい自分自身の加重を持つ[*15]）を発行するかもしれない．
6. 攻撃者の望みは彼の「sub-DAG」が main-DAG を上回ること，つまり，DAG が二重支払いのトランザクションから成長し続け，そして，正当なブランチが破棄されることである（図 5 を参照のこと）．

実際，一つの大きな二重支払いトランザクションが攻撃者にチャンスを増加させることを示している．それだけでなく，この数理モデルの「理想的」なシチュエーションにおいて，この攻撃は常に成功する．

実際，$W^{(n)}$ を二重支払いのトランザクションへ少なくとも 3^n の加重を与えるノンスを得るのに必要な時間と仮定してみよう．$W^{(n)}$ を $\mu 3^{-n}$ のパラメーターを持つ指数関数的に分布する確率変数（$\mu^{-1}3^n$ の期待値を持つ）と仮定できよう．μ は攻撃者の演算処理能力の大きさを表す．

商人が，累積加重が少なくとも w_0 になる時に正当なトランザクションを受け入れる（これはオリジナルトランザクションから t_0 時間単位で起こる）と仮定する．そうすると累積加重が線形速度 λw で成長すると期待するのは妥当である．λ は（正直なユーザー達によって発行された）トランザクションのシステムへの全体の到着率であり，w は一般的なトランザクションの加重を意味する．$w_1 = \lambda w t_0$ とし，その時点における正当なブランチの典型的な合計加重を示す．

$\lceil x \rceil$ は x 以上の最小の整数であり， $n_0 = \left\lceil \frac{\ln w_1}{\ln 3} \right\rceil$ と定義される．つまり， $3^{n_0} \geq w_1$ である（実際，w_1 が大きい時 $3^{n_0} \approx w_1$）．もし，長さ t_0 t のインターバル時間の間に，攻撃者が少なくとも 3^{n_0} の加重を与えるノンスを得ることが出来た場合，そのとき攻撃は成功する．そのようなイベントの確率は，

$$\mathbb{P}[W^{(n_0)} < t_0] = 1 - \exp(-t_0 \mu 3^{-n_0}) \approx 1 - \exp(-t_0 \mu w_1^{-1}) \approx \frac{t_0 \mu}{w_1}$$

[*15] ここでトランザクションの自身の加重が逸脱する可能性があることを再び仮定する；実のところ，下記の議論から加重が逸脱することは（少なくとももものすごく加重が逸脱することは）良い考えでないことが明らかになるだろう

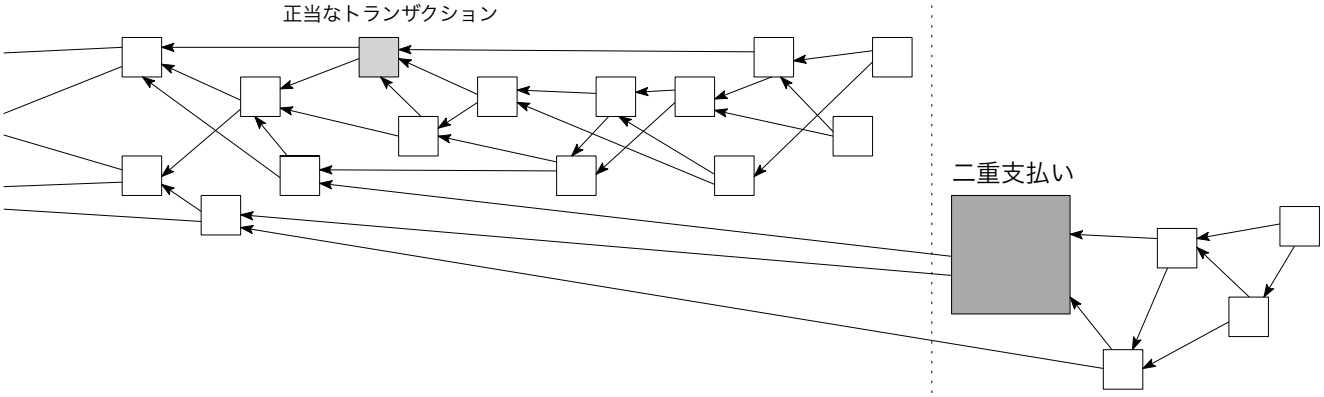

図 5　「大きい加重」攻撃

(少なくとも $\frac{t_0\mu}{w_1}$ が小さい場合，妥当な仮定である)．そのとき，もし，この「即座の」攻撃が成功しなかった場合，攻撃者は継続して $n > n_0$ から 3^n の加重を与えるノンスを探し続け，見つかった場合は正当なブランチの合計加重が 3^n より小さい事を期待するかもしれない．今，この確率は，

$$\mathbb{P}[\lambda w W^{(n)} < 3^n] = 1 - \exp\big(-\mu 3^{-n_0} \times (3^{n_0}/\lambda w)\big) = 1 - \exp(-\mu/\lambda w) \approx \frac{\mu}{\lambda w}.$$

つまり $\frac{\mu}{\lambda w}$ は通常小さいはずであるにも関わらず，各「レベル」n で，攻撃者は一定の確率で成功する．従って，最終的には成功を収める．事実，通常成功までの時間はおおよそ $3^{\frac{\lambda w}{\mu}}$ である．この数量は非常に大きいかも知れないが，「最初の」(t_0 の時間の間）攻撃が成功する確率は依然として小さくない．よって私達は次の結論にたどり着き，対策をしなければならない．例えば，自身の加重に上限を設けたりあるいは定数に設定したりする（3 章で言及したように後者はベストな解決策でないかも知れない．なぜなら，スパムからの十分な保護を提供しないためである.)．

今ここで，最大加重が例えば 1（一般的なケースは単に再スケーリングすることで扱うことができる）に上限設定されるシチュエーションについて討論し，攻撃者が成功する確率を見積もっていこう．

与えられたトランザクションが発行された瞬間から t_0 単位時間で累積重量 w_0 が累積されたと仮定する．また，そのトランザクションの適応期間は終了しており，その累積重量は速度 λ に比例して増加すると仮定する．今，攻撃者はこの取引を二重支払（ダブルスペンド）したいと思っている．そのために，最初の取引が発行された時点[*16]で，二重支払を秘密裏に準備し，二重支払を承認する他のトランザクションの生成を開始する．ある時点（商人が正当な取引を受け入れることを決定した後）に，攻撃者のサブタングルが正当なサブタングルを上回ると，二重支払攻撃が成功するでしょ

[*16] または以前に；これについては後ほど考察する

う．もしそれが起こらなければ，二重支払の取引は他人に承認されず（なぜなら正当な取引はより多くの累積加重を獲得し，本質的にすべての新しいチップが間接的にそれを承認するだろうから）孤立するだろう．

前と同じように，μ を攻撃者のコンピューティングパワー（例，トランザクションの生成スピード）とする．私達はまた，トランザクションが即座に伝搬すると平易に仮定する．$G_1, G_2, G_3, \ldots$ はパラメータ μ を持つ（つまり期待値 $1/\mu$ を持つ）独立同分布（i.i.d.）な指数確率変数とし，$V_k = \mu G_k$，$k \geq 1$ としよう．明らかに，$V_1, V_2, V_3, \ldots$ はパラメータ 1 を持つ独立同分布（i.i.d.）な指数確率変数である．

t_0 の時点で商人がトランザクションを受け入れると決めたと仮定してみよう（その時点でそのトランザクションの累積加重が w_0 であった事を思い出していただきたい）．攻撃者が二重支払いに成功する確率を見積もってみる．$M(\theta) = (1-\theta)^{-1}$ をパラメータ 1 を持つ指数分布の積率母関数（[14] の 7.7 章を参照のこと）とする．これは $\alpha \in (0,1)$ において以下が成り立つことが知られている[*17]．

$$\mathbb{P}\Big[\sum_{k=1}^{n} V_k \leq \alpha n\Big] \approx \exp\big(-n\varphi(\alpha)\big), \tag{10}$$

ここで $\varphi(\alpha) = -\ln\alpha + \alpha - 1$ は $\ln M(-\theta)$ のルジャンドル変換である．一般的な事実として，$\alpha \in (0,1)$ において $\varphi(\alpha) > 0$ が成り立つことを観察されたい（パラメータが 1 の指数分布に従う確率変数の期待値が 1 に等しいことも思い出していただきたい）．

また $\frac{\mu t_0}{w_0} < 1$ も仮定する（でないと，攻撃者のサブタングルが最終的に正当なタングルに勝る確率が 1 に近くなる）．今ここで，t_0 の時点で w_0 を上回るには，攻撃者は最低でも t_0 の間に最大加重が m の w_0（単純化のために整数部分を省く）を発行することが可能でなければならない．従って，式 (10) を用いて，二重支払いのトランザクションが t_0 の時点でより大きい累積加重を持つ確率はおおよそ以下であることを得る．

$$\begin{aligned}
\mathbb{P}\Big[\sum_{k=1}^{w_0/m} G_k < t_0\Big] &= \mathbb{P}\Big[\sum_{k=1}^{w_0} V_k < \mu t_0\Big] \\
&= \mathbb{P}\Big[\sum_{k=1}^{w_0} V_k < w_0 \times \frac{\mu t_0}{w_0}\Big] \\
&\approx \exp\big(-w_0\varphi\big(\tfrac{\mu t_0}{w_0}\big)\big).
\end{aligned} \tag{11}$$

つまり，上の確率が小さくなるためには，通常 $\frac{w_0}{m}$ が大きく，$\varphi\big(\frac{\mu t_0}{w_0}\big)$ がそれほど小さくない必要がある．

[*17] これはいわゆる大偏差原理の結果である．上界の簡単でためになる導出には一般書 [13] に加え，[14] の 8.5 章の命題 1.9 を，下界の（それほど簡単でない）導出には [5] の 1.9 章を参照のこと

$t \geq t_0$ の時，正当なトランザクションの累積加重がおおよそ $w_0 + \lambda(t - t_0)$ であることに注意されたい（適応期間が終わると累積加重が λ のスピードで成長すると仮定したことを思い出していただきたい）．式 (11) と類似して，二重支払いのトランザクションが時間 $t \geq t_0$ の時，もっと大きい累積加重を持つ確率についておおよそ次が得られる．

$$\exp\big(-(w_0 + \lambda(t - t_0))\varphi\big(\tfrac{\mu t}{w_0 + \lambda(t-t_0)}\big)\big). \tag{12}$$

その時，$\frac{\mu t_0}{w_0} \geq \frac{\mu}{\lambda}$ であることは真実でなくてはならない（適応期間中，累積加重は λ 未満の速度で成長するから）．ともあれ，二重支払いが成功する確率は次とみなせる．

$$\exp\big(-w_0\varphi\big(\max(\tfrac{\mu t_0}{w_0}, \tfrac{\mu}{\lambda})\big)\big). \tag{13}$$

例えば，$\mu = 2$，$\lambda = 3$ の時（攻撃者の力は残りのネットワークの力よりわずかに少なくなる），トランザクションは時間 12 までに累積加重 32 を得ると仮定する．この時，$\max(\frac{\mu t_0}{w_0}, \frac{\mu}{\lambda}) = \frac{3}{4}$ であり，$\varphi\big(\frac{3}{4}\big) \approx 0.03768$ となり，そして式 (13) は約 0.29 の上界を与える．しかし，もし $\mu = 1$（その他のパラメータはそのまま）と仮定すると，$\max(\frac{\mu t_0}{w_0}, \frac{\mu}{\lambda}) = \frac{3}{8}$ であり，$\varphi\big(\frac{3}{8}\big) \approx 0.3558$ となり，式 (13) は約 0.00001135 の上界を与える．これは実にかなりの変化である．

上の議論から，システムがセキュアであるために，$\lambda > \mu$ が真であるべき事を見て取ることは重要である（さもなくば式 (13) の概算は無用のものとなる）．すなわち，「正直な」トランザクションのインプットフローは攻撃者の演算処理能力と比べて十分に大きくあるべきである．これはタングルベースシステムの初期におけて，追加のセキュリティ措置（チェックポイント）の必要性を指摘するものである．

また，コンフリクトするトランザクションのどちらが有効かを決定する戦略に関して，累積加重だけに頼る際，注意を払わなければならない．なぜなら，4.1 章 で記述されたもの（攻撃者はかなり前から二重支払いのトランザクションを準備し，秘密裏にそれを参照するサブチエーン/サブタングルを構築し，商人が正当なトランザクションを受け入れた後，そのサブタングルをばらまく）と似た攻撃の対象となりうるからである．むしろ，2 つのコンフリクトするトランザクションから決めるもっと良い方法は，次の章で記述されるようなものかも知れない：チップ選択アルゴリズムを実行し，2 つの内どちらのトランザクションがある選択されたチップにより（間接的に）承認されるかを見ることである．

4.1 パラサイト・チェーン攻撃と新しいチップ選択アルゴリズム

次の攻撃を考えてみよう (図 6)：攻撃者が秘密裏にチェーン/サブタングルを構築し，時折メインのタングルを参照しスコアを得る（良いチップのスコアがおおよそメインのタングルにある全ての自身の加重の合計である一方，攻撃者のチップのスコアもおおよそパラサイト・チェーンにある全て

の自身の加重の合計であることに注意)．また，十分に強力なコンピューターを持つ，独力でチェーンを構築する者にとってネットワークの通信遅延は問題とならない[*18]ので，攻撃者はパラサイトのチップにもっと高さ (height) を与えることが出来るかも知れない．最後に，正直なノードが単純な提供されているチップから選ぶいくつかの選択戦略を用いる場合，攻撃者は，攻撃の際，チップの数を人工的に増やすことも可能である（同じ攻撃者のトランザクションを承認する多数のトランザクションを発行することによって．図 6 参照のこと)．

この攻撃を防御するには，私達はメインのタングルがより多くの（アクティブな）ハッシュパワーを持つはずだと言う事実を用いて，攻撃者よりも多くのトランザクションに累積加重を与えるようにするつもりである．MCMC (マルコフ連鎖モンテカルロ) アルゴリズムを用いて参照する 2 つのチップを選ぶというアイデアである．

$\mathcal{H}_x$ をあるサイト（= タングルグラフ上に表されるトランザクション）の現在の累積加重であるとする．全ての自身の加重が 1 に等しい事を思い出していただきたい．ゆえに，チップの累積加重は常に 1 で，他のサイトの累積加重は少なくとも 2 である．

いくらかの粒子（ランダム・ウォーカーとして知られる）をタングルのサイトに配置し，チップに向かってランダム[*19]に歩かせるというのがアイデアである．ウォークによって「選ばれた」チップは，承認の候補となる．アルゴリズムは次のように記述される．

1. W と（例えば）$2W$ の間の累積加重を持つ全てのトランザクションを考える（選ばれる W [*20]は妥当な大きさである)．
2. N 個の粒子を独立的にそこに配置する（N はそれほど大きくない，10 程度[*21])．
3. これらの粒子は「チップに向かって」独立した離散時間ランダムウォークを行う（x から y への移行は y が x を承認した時のみ可能)．
4. 初めにチップ集合に到達する 2 つのランダムウォークは承認される二つのチップを指し示す．(しかしながら，このルールを次のように変更することが賢明かもしれない．最初にあまりにも速くチップに到達したランダムウォーカーを放棄する．ランダムウォーカーが「怠惰な

[*18] これは攻撃者が常に自身のトランザクションをネットワークの情報に頼ることなく承認できるからである

[*19] ランダム性の「標準的な」ソースはない；各ノードは，ランダムウォークのシミュレーションにおいて擬似乱数ジェネレーターを利用する

[*20] 粒子をタングルの「深部」に配置する（直ぐにチップに行かないように）が，「深過ぎない」所へ（妥当な時間内にチップを見つけるように）という考え方である．また，$[W, 2W]$ は恣意的なもので，代わりに例えば $[W, 3W]$ や，他に何でも選ぶことが出来る．他のウォーカーの出発点の選び方も考えられる．例えば，ノードは t_0 から $2t_0$ までの単位時間の間に受け取られたランダムなトランザクションを単に取ることができる．t_0 はある一定の時間値である．

[*21] 重ねて，この選択は主として恣意的なものである；追加のセキュリティ措置のために 2 つではなくいくつかの粒子を私達は用いる．粒子が偶然に（長いと思われる）攻撃者のチェーンにジャンプし，そこに永くとどまり，他のチップが先に選択されるという考えである

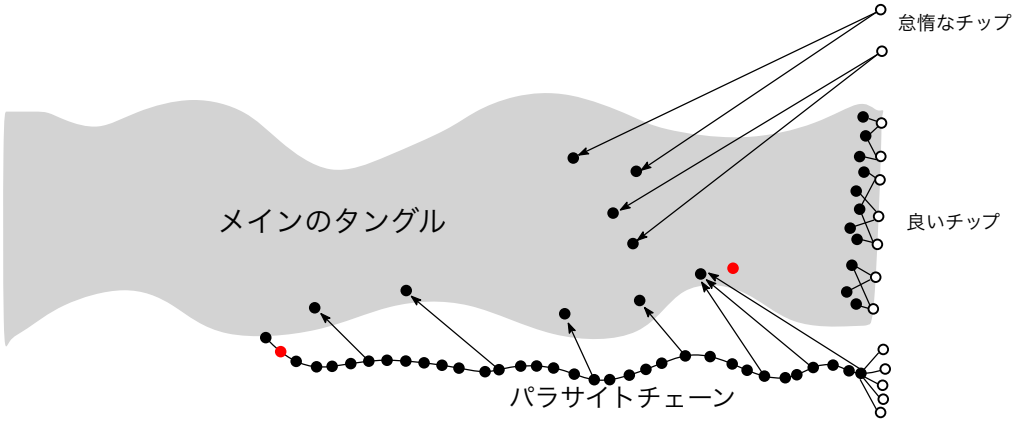

図 6　チップ選択アルゴリズム．二つの赤い円が二重支払いの試みを示す．

チップ」の 1 つに行き着いたかもしれないから．）

5. ウォークの移行確率は次のように定義される：もし，y が x を承認する（これを $y \rightsquigarrow x$ と表示する）場合，移行確率 P_{xy} は $\exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)$ に比例する．

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1}, \tag{14}$$

ここで，$\alpha > 0$ は選択されるパラメーターである（例えば，$\alpha = 1$ から始めることが出来る）

このアルゴリズムが「ローカル」であり，計算を行うためにジェネシスまで戻る必要はないことに注意いただきたい．特に，タングル全体の累積加重を計算する必要がないことに気づく．せいぜいタングル全体の累積加重を計算することは（間接的に）ウォークの開始点を承認するサイトのために必要である．

このアルゴリズムが意図された通りに機能するのを見るには，まず「怠惰なチップ」（認証を行うことを回避するため意図的に古いトランザクションを承認するチップ）を考える．図 6 参照のこと．粒子がそのようなチップによって承認されているサイトにあったとしても，累積加重の差がとても大きくなるため，怠惰なチップが選択される可能性がないことをご覧頂きたい．式 (14) を確認のこと．

次に，以下のような攻撃を検討してみよう：攻撃者が秘密裏に，彼/彼女のアカウントを空にし，自身のコントロール下にある別のアカウント（図 6 の最も左の赤い円で示される）に移すトランザクションを含むチェーン（「パラサイトチェーン」）を構築する．ある時点で攻撃者はメインのタングルでトランザクションを発行し（もう一つの赤い円で示されている），商人がそれを受け入れるまで待つ．パラサイトチェーンは時折メインのタングルを参照するので (「パラサイト」と呼ばれるゆえん)，チエーンにおける累積加重がそれほど大きくないにも関わらず，そのサイトは良い高さ/スコ

ア（メインのタングル以上）を持つ．商人のトランザクション後は，メインのタングルを参照できないことに注目．また，攻撃者は図式にあるとおり，人工的に彼/彼女のチップの数を膨らまそうと試みるかも知れない．ノードにパラサイトチェーンを参照させ，よって「良い」タングルが孤児化されるというのが攻撃者のアイデアである．

今ここで，MCMC（マルコフ連鎖モンテカルロ）選択アルゴリズムが，高い確率で攻撃者のチップの一つを選択しない理由は理解に難くないであろう．基本的に，なぜアルゴリズムが怠惰なチップを選択しないのかの理由と同じである：パラサイトチェーンのサイトの方がそれらが参照するメインのタングルのサイトよりはるかに小さい累積加重を持つであろうからである．従って，ランダムウォークがパラサイトチェーンにまで飛び移る可能性はない（そこで始まったのでない限りだが，その可能性はあまりなく，なぜならメインのタングルの方がより多くのサイトを持つからである）．

次に，なぜノードがこのアルゴリズムに（少なくとも近似的に）従うのかについてコメントしたい．1 章で見た通り，少なくともネットワークのノードの「かなりの」部分が参照アルゴリズムに従うと仮定することが妥当である事を思い出していただきたい．また，演算処理とネットワークの遅延のため，チップ選択アルゴリズムは，タングルの過去のスナップショットにおいてより機能する（トランザクションが発行された瞬間に関して）．さて，参照アルゴリズムにおいて意図的にこのスナップショットをもっと過去に移す[*22]のは良い考えかも知れない．その理由については続編で説明したい．攻撃者のトランザクションがすぐに承認される確率の最大化を望む「利己的な」ノードを想像していただきたい．本章の（他のノードの多数によって採用された）MCMC アルゴリズムは，チップ集合における確率分布を定義する．明らかに，そのノードにとって自然な最初の選択は，その分布のうち可能な限り多くを収めるチップを選ぶことである．しかし，多くの他のノードも利己的に振る舞い，同じ戦略を用いた場合（それも妥当な仮定に思われる），全員が負けることになる．多くの新たなトランザクションが同じ 2 つのチップを（だいたい）同じタイミングで選び，以後の承認において競争は過多となる（各ノードは過去のスナップショットを利用するため，それらは多数ノードによる 2 つのチップから来る累積加重の増加をまだ「感じない」）．ゆえに，利己的なノードであっても，いずれかのランダムなチップ承認アルゴリズム[*23]を用いる必要があり，選択されたチップの確率分布は，ある意味でデフォルトの確率分布（参照チップ選択アルゴリズムによって生成されたもの）から「当たらずといえども遠からず」である．私達はこの「塊状となった」（利己的

[*22] まずランダムウォークがそのスナップショットに関して（過去の）チップを見つけ，続けて現在のタングルの「実際の」チップに向けウォークするということである

[*23] MCMC を何度も実行する以外により良いチップを簡単に見つける（つまり「正直な」ノードによって選択される可能性が高い）方法はないように思われる．一方では，MCRW を何度も実行するには時間と他のリソースが要求される．それに幾らかの時間を費やすとタングルの状態は既に変化している可能性があり，よって新たに開始する必要がある．少なくともかなりの割合の他のノードがデフォルトのチップ選択戦略に従うと仮定すれば，これはノードが MCMC チップ選択戦略を放棄して他の戦略を支持する理由がないことを説明する．

なノードの存在による）確率分布がデフォルトの確率分布と同等であるとは主張しないが，上の議論から，近いはずであることが分かると思う．それは「悪い」チップを選ぶ確率が低いものにとどまる事を意味する．どうであれ（ビットコインと異なり），ノードが利己的に振る舞うインセンティブはない．なぜなら，確認時間の僅かな減少しか得られないからである．このノードが MCMC チップ選択を放棄する理由はない．

また，必ずしも移行確率が式 (14) で定義された通りでなければならないわけではない．指数の代わりに，例えば $f(s) = s^{-3}$ など，何か他の急速に減少する関数も利用できる．W と N についても選択の幅はかなりある．実際，著者としてはこの章の主要な貢献はチップ選択における MCMC の利用というアイデアそのものであると感じている．正確にどのようにこれらのパラメーターが選択されるべきかに関する「理論的な」議論があるかは不明である．

4.2 分裂攻撃

上述の MCMC アルゴリズムに対抗したこの攻撃スキームは Aviv Zohar が提示したものである．高負荷型において，攻撃者はタングルを 2 つに分裂させ，両方の間でバランスを取り，両方とも成長を続けるとする．正直なノードが 2 つのブランチを同時に参照する（よって結合させる）のを避けるため，分裂攻撃の初めにおいて，攻撃者は少なくとも 2 つのコンフリクトするトランザクションを配置しなければならない．次に彼/彼女はネットワークの約半数がそれぞれのブランチに貢献することによって，彼/彼女が比較的少ない演算処理能力でランダムな変動を「補償」できることを期待する．そうすると，攻撃者は同じ資金を二つのブランチで消費する事ができる．

そのような攻撃に対し防御するには，2 つのブランチの間でバランスを保つことを非常に困難にする「シャープな閾値」のルールを用いる必要がある（ビットコインにおける「最長のチェーンを選択」と同じく）．例を挙げると，一つのブランチが 537 の合計加重（あるいは他のメトリクスを用いても良い）を持ち，別のブランチの合計加重もかなり近い，例えば 528 であると仮定してみよう．もし，そのよう状況においてほぼ 1/2 の確率で正直なノードが最初のノードを選択したとしたら，おそらく攻撃者は 2 つのブランチの間でバランスを取ることが出来るであろう．しかし，正直なノードが最初のブランチを 1/2 よりかなり大きい確率で選好する場合，攻撃者はおそらくバランスを維持できず，なぜなら不可避なランダムな変動の後，ネットワークはブランチの一つを選び，もう一つの方を放棄するであろう．明らかに，MCMC アルゴリズムをそのように振る舞わせるためには，非常に早く減衰する関数 f を選ばなければならず，ランダムウォークを（比較的）大きな深さを持つノード（それにより分裂が作られる前に始まる可能性が高い）から始めなければならない．この場合，ブランチ間の合計加重の差が小さかったとしても，ランダムウォークは「より重い」ブランチを選ぶ．

ネットワーク同期化の各問題のためもあり，攻撃者の仕事は非常に困難なものであることは指摘しておく価値があると思われる．彼/彼女は，最近発行されたトランザクションの多くについて知らないかも知れない[*24]．そのような攻撃を受け流すもう一つの効果的な方法には，十分に強力な主体による多数のトランザクションの同時発行である（1 つのブランチで）．それにより急激にパワーバランスを変え，攻撃者による対応を困難にする．また，攻撃者が分裂を維持できた場合，最近のトランザクショ ンは 50% の確認の確実性しか持たず (1 章参照)，成長もしないことも見て取っていただい．ゆえに「正直な」各ノードは，分裂前のトランザクションだけを承認し始めると決めるかも知れない（そして特に，2 つのコンフリクトするトランザクションのどちらも承認しない）．

チップ選択アルゴリズムの他の修正を検討することも出来る．例えば，あるノードが 2 つの大きなサブタングルに直面した場合，まず自身の加重の合計が大きい方を選び，次に上述した MCMC アルゴリズムを用いてそこでチップ選択を行う．

また，次のアイデアも検討の余地がある：式 (14) における移行確率は $\mathcal{H}_x - \mathcal{H}_y$ だけでなく，ウォーカーが，（弱い方のブランチに入るのを避けるため）タングルの深部にいるが，チップに近い時より広がる（それにより選択する 2 つのトランザクションの選択において十分なランダム性がある）場合，マルコフ連鎖の次のステップがほぼ決定的な形で $\mathcal{H}_x$ にも依存する．

■結論：

1. 私達は，攻撃者がシステムを追い越こすことによって，二重支払いを試みた場合のいくつかの攻撃戦略を検討した．
2.「大きな加重」攻撃は，二重支払いを行うために，攻撃者は二重支払いのトランザクションに非常に大きな加重を与えようと試み，結果，正当なサブタングルをそれだけで加重において上回る事を意味する．許される自身の加重に制限がない場合，確かにこの戦略はネットワークにとって脅威である．ソリューションとして，トランザクションの自身の加重を制限するか，定数に設定することさえもありうる．
3. トランザクションの最大の自身の加重が m である状況において，攻撃者にとって最良の戦略は，常に最大の自身の加重を持つ二重支払いのトランザクションを参照するトランザクションを生成することである．攻撃者の演算処理能力に比べて「正直な」トランザクションのインプットフローが十分に大きい時，二重支払いのトランザクションがより多くの累積加重を持つ確率は式 (13) を用いて見積もることが出来る（式 (13) の下の各例も参照のこと）．
4.「パラサイトチェーン」の構築に基づく攻撃は，高さやスコアに基づく承認戦略を無用である．なぜなら，攻撃者は正当なタングルより高い値をそれらについて得るであろうからであ

[*24] なので「本当の」累積加重は，彼/彼女が思うそれより大きく異なるかも知れない

る．一方，4.1 章で記述された MCMC チップ選択アルゴリズムはこの種の攻撃に対しよく保護を提供するように見受けられる．

5. ボーナスとして，「怠惰なノード」（タングル認証のために必要な計算を避けるため，古いトランザクションの承認しかしないノード）からの保護も提供する．

5　量子計算への耐性

（現時点において仮説的なものだが）十分に大型の量子コンピュータは，繰り返し答えを推測して確認することが唯一の解決方法であるような問題において，非常に効率的たりうることが知られている．ビットコインブロックを生成するためノンスを見つけるプロセスは，そのような問題の良い例である．現在，ブロック生成できるための適切なハッシュを見つけるために平均で約 2^{68} ノンスを確認しなければならない．量子コンピュータは従来のコンピューターでは $\Theta(N)$ 回の演算を要する上のような問題を解くのに $\Theta(\sqrt{N})$ 回の演算を要することが知られている ([15] を見よ)．従って，ビットコインの採掘（マイニング）において，量子コンピュータは従来のコンピュータに比べ，約 $\sqrt{2^{68}} = 2^{34} \approx 170$ 億倍も効率が良いことになる．また，ブロックチェーンがハッシュパワーの増加に対応して難易度を増加させなかった場合，孤立したブロック率の増加につながることも言及に値する．

同様の理由で，上に記述された「大きな加重」攻撃は量子コンピュータによって遥かに効率的であろう事を指摘する．しかし，（4 章で提案されたように）加重の上限を設定することで，量子コンピュータによる攻撃も受け流すことが出来る．理由は次の通りである．IOTA では，トランザクションの発行のため適切なハッシュを見つけるのにチェックしなければならないノンスの数は巨大なものではなく，約 3^8 にすぎない．「理想的な」量子コンピュータにとって，得られる効率はよって $3^4 = 81$ のオーダーとなり，受け入れることが出来るものである．（また，$\Theta(\sqrt{N})$ が $10\sqrt{N}$ かその周辺に，容易に収束できることを思い出していただきたい．）また，このアルゴリズムにおいて，ノンスが見つかる時間はトランザクション発行に要する他の各タスクに必要な時間に比べさほど大きくないようになっており，IOTA は量子コンピュータに対して一層の耐性を持っている．

従って，上の議論はタングルが，（ビットコイン）ブロックチェーンと比べて，量子コンピュータを用いる敵対者に対しより強力な保護を提供することを示すものである．

謝辞

Cyril Grünspan と風間 徹（前の草案においていくつかの間違いを指摘してくれた）に感謝の意を表する.

参考文献

[1] Iota: a cryptocurrency for Internet-of-Things. See `http://www.iotatoken.com/`, and `https://bitcointalk.org/index.php?topic=1216479.0`

[2] bitcoinj. Working with micropayment channels. `https://bitcoinj.github.io/working-with-micropayments`

[3] people on nxtforum.org (2014) DAG, a generalized blockchain. `https://nxtforum.org/proof-of-stake-algorithm/dag-a-generalized-blockchain/` (registration at `nxtforum.org` required)

[4] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, Aviv Zohar (2012) On Bitcoin and red balloons. *Proc. 13th ACM Conf. Electronic Commerce*, 56–73.

[5] Richard Durrett (2004) Probability – Theory and Examples. *Duxbury advanced series.*

[6] Sergio Demian Lerner (2015) DagCoin: a cryptocurrency without blocks. `https://bitslog.wordpress.com/2015/09/11/dagcoin/`

[7] Yonatan Sompolinsky, Aviv Zohar (2013) Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. `https://eprint.iacr.org/2013/881.pdf`

[8] Yonatan Sompolinsky, Yoad Lewenberg, Aviv Zohar (2016) SPECTRE: Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections. `https://eprint.iacr.org/2016/1159.pdf`

[9] Yoad Lewenberg, Yonatan Sompolinsky, Aviv Zohar (2015) Inclusive Block Chain Protocols. `http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive_btc.pdf`

[10] Joseph Poon, Thaddeus Dryja (2016) The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. `https://lightning.network/lightning-network-paper.pdf`

[11] Sheldon M. Ross (2012) *Introduction to Probability Models.* 10th ed.

[12] David Vorick (2015) Getting rid of blocks. `slides.com/davidvorick/braids`

[13] Amir Dembo, Ofer Zeitouni (2010) *Large Deviations Techniques and Applications.* Springer.

[14] Sheldon M. Ross (2009) *A First Course in Probability.* 8th ed.

[15] Gilles Brassard, Peter H Ø yer, Alain Tapp (1998) Quantum cryptanalysis of hash and claw-free functions. *Lecture Notes in Computer Science* **1380**, 163–169.