概要

私たちは、既存のProof of Workをオーバーレイする Proof of Stake ベースのファイナリティシステムであるCasperを紹介します。

CasperはProof of Stakeアルゴリズムの研究と、ビザンチンフォールトトレラントコンセンサス理論を組み合わせた、部分的コンセンサスメカニズムです。

私たちのシステムを紹介し、望ましい機能をいくつか証明し、Long Range RevisionsとCastastrophic Crashesに対する防御を示します。

Casperのオーバーレイは、ほとんどのProof of Work チェーンのブロック逆転に対する保護を提供します。

1 導入

ここ数年、Proof of Stake (PoS) ベースのブロックチェーンコンセンサスアルゴリズムに関する多くの研究が行われてきました。

PoS システムでは、ブロックチェーンはシステム内でコインを保持する人が参加できるプロセスを通じて、新しいブロックを追加し合意します。エージェントが持つ影響は、コインの数 (つまり「ステーク」) に比例します。

これは、Proof of Work (PoW) の“マイニング” に代わる非常に効率的な代替手段であり、マイニングの高いハードウェアおよび電気コストなしでブロックチェーンを稼働させることができます。

PoSの設計には2つの主要な考え方があります。

1つ目はチェーンベースのProof of Stake[1, 2]で、これはProof of Workの仕組みを模倣したブロックの連鎖を特徴とし、ステークホルダーに新しいブロックを作成する権利を疑似ランダムに割り当てることによってマイニングをシミュレートします。

これには Peercoin[3]やBlackcoin[4]、そしてIddo Bentov’sの研究が含まれます。

2つ目が、ビザンチンフォールトトレラント (BFT) ベースのProof of Stakeで、これはPBFTなどのBFTコンセンサスアルゴリズムに関する30年前の研究に基づいています。

[6] BFTアルゴリズムは、一般的な数学的特徴が証明されています。

例えば、プロトコル参加者の2/3以上が正直にプロトコルに従っている限り、ネットワークの待ち時間に関わらず、アルゴリズムは競合するブロックをファイナライズすることができないことを数学的に証明することができます。

Proof of StakeのためのBFTアルゴリズムの再利用は、Tendarmint[7] によって初めて導入され、[8]のような近代的なインスピレーションを受けています。

CasperはこのBFTの伝統に従いますが、いくつかの変更があります。

1.1 私たちの取り組み

Casper the Friendly Finality Gadget は、提案メカニズムの上にオーバーレイとして存在し、ブロックを提案するメカニズムです。

Casperはこれらのブロックをファイナライズする責任があり、正規トランザクション有する、本質的な台帳を表す、固有のチェーン(メインチェーン)を選択します。

Casperは安全性を提供しますが、選択された提案メカニズムに依存し活用されます。それはつまり、攻撃

者が提案メカニズムを完全に制御する場合、Casperは2つの矛盾するチェックポイントをファイナライズすることを防ぎます。

しかし、攻撃者はCasperが将来のチェックポイントを確定するのを妨げる可能性があります。

CasperはBFTアルゴリズムが必ずしもサポートしていない、いくつかの新機能を導入しています。

1. Accountability.
   バリデーターがルールに違反した場合、違反を検出して、どのバリデーターがルールに違反しているかを知ることができます。
   Accountability は、不正なバリデーターにペナルティを課し、チェーンベースのPoSで悩まされている”Nothing at Stake” 問題を解決します。
   ルールに違反した場合のペナルティは、バリデータ達のデポジットです。
   この最大のペナルティは、プロトコル違反に対する防御です。
   なぜなら、Proof of Stakeのセキュリティは、マイニング報酬の利益を大幅に上回るように設定できるペナルティのサイズに基づいているため、Proof of StakeはProof of Workより厳密で強力なセキュリティインセンティブ提供します。
2. Dynamic Validators.
   Validator setが時間の経過と共に変化する安全な方法を紹介します (第3章)。
3. Defebses.
   Long Range Revision攻撃と非常に弱いトレードオフの同期性の仮定を犠牲にしてバリデーターの1/3以上がオフラインになった攻撃に対する対策を紹介します(第4章)
4. Modular overlay.
   オーバーレイとしてのCasperの設計は、既存のProof of Workチェーンへのアップグレードとして実装することを容易にします。

私たちは段階的にCasperを説明していきます。最初にシンプルなバージョン (第2章)、次にバリデータセットの変更 (第3章)。

そして最後に攻撃に対する防御を (第4章) 説明します。

2 Casper プロトコル

Ethereum内では、提案メカニズムは最初に既存のProof of Work チェーンとなり、Casperの最初のバージョンをPoW/PoSのハイブリットシステムにしました。

将来のバージョンでは、PoWによる提案の仕組みはより効率的なものに置き換えられます。

例えば、ブロック提案をある種のPoSラウンドロビンブロック署名方式に変換すると想像できます。

このシンプルなバージョンのCasperでは、固定されたバリデーターセットと提案メカニズム（例えば、おなじみのProof of Work提案メカニズム）があると仮定して、既存のブロックの子ブロックを生成し、常に成長するブロックツリーを形成します。

[9]から、ツリーの根は一般に「ジェネシスブロック」と呼ばれています。

通常の状況下では、提案メカニズムは、リンクされたリストにおいて順番にブロックを次々に提案することを期待します (すなわち、各「親」ブロックがちょうど1つの「子」ブロックを有する)。

しかし、ネットワークの待ち時間や意図的な攻撃があった場合、提案メカニズムは必然的に同じ親の複数の

子を必然的に生成することになります。

Casperの仕事は、各親から1人の子供を選択し、ブロックツリーから1つの正規のチェーンを選択することです。

フルブロックツリーを扱うのではなく、効率を上げるために、Casperはチェックポイントツリーを構成するチェックポイントのサブツリーのみを考慮します (図 1a) 。

ジェネシスブロックはチェックポイントであり、ブロックツリー (またはブロック番号) の高さが100の倍数である全てのブロックもチェックポイントです。

ブロックの高さ100*k ブロックの「チェックポイントの高さ」は単純にkです。

等価的に、チェックポイントcの高さh(c)は、親リンクに沿ってルートからルート(c)まで伸びるチェックポイントチェーンの要素の数です (図 1b) 。

各バリデーターはデポジットを持っています。これはバリデーターとして参加する時の、コインの数です。

参加後、各バリデータのデポジットは報酬と罰則に伴って昇降します。Proof of Stake の安全性は、バリデーターの数ではなく、デポジットのサイズに由来します。

したがって、本書の残りの部分では、「バリデーター2/3」と言う場合、これはデポジットの加重割合を指しています。

つまり、合計のデポジットサイズがバリデーターセット全体のデポジットサイズの2/3に等しいバリデーターセットだということです。

バリデーターは4つの情報 (表1) である：2つのチェックポイント s および t と、高さ h(s) および h(t) を含む投票メッセージをブロードキャストすることができます。

チェックポイントツリーでは s が t の祖先であることが必要です。

そうでなければ、投票は無効と見なされます。

バリデーターvの公開鍵がバリデーターセットにない場合、投票は無効と見なされます。

併せて、バリデーターの署名と共に、これらの投票をv, s, t, h(s), h(t)の形で書く。

我々は以下の用語を定義する。

1. Supermajority link は、少なくとも 2/3 のバリデーター（デポジットによる）がソースaとターゲットbとで投票を公開するように、a→bと書かれたチェックポイントの順序付きペア（a, b）です。Supermajority links はチェックポイントをスキップすることができます。つまり、h(b)¿ h(a)+1です。図1cは、赤の3つの異なるSupermajority links を示しています：r→b1, b1→b2, そして b2→b3。
2. 2つのチェックポイント a と bは、それらが別々のブランチ内のノードである場合、すなわち祖先または他方の子孫のいずれでもない場合にのみ、矛盾すると呼ばれる。
3. チェックポイント c は、(1) ルートである場合にjustifiedされ、(2) c’ がjustifiedされるsupermajority link c’ →c が存在する場合、図1cは、4つの justified ブロックのチェーンを示しています。
4. チェックポイント c は、finalizedされ、supermajority link c→c’ があり、c’ が c の直接の子である場合、finalizedされます。同様に、チェックポイントcは、チェックポイント c がjustifiedされ、supermajority link c→c’ が存在し、チェックポイント c と c’ が衝突していない場合にのみ終了し、h(c’) = h(c)+1である。

これらのルールのいずれかに違反したバリデーターは、デポジットを払い落とし (slashed)ます。

Casperの最も注目すべき特性は、バリデーターの1/3が2つのCasper Commandments / slashing conditions の内1つに違反し、2つの矛盾するチェックポイントをfinalizeすることが不可能であることです。

(図2)

バリデーターが slashing conditionに違反した場合、違反の証拠をトランザクションとしてブロックチェーンに組み込むことができます。その時点で、証拠トランザクションの提出者に与えられた小さな”ファインダー報酬”が付いています。現在のEthereumでは、slashing conditionを強制停止するには、EthereumのProof of Workブロック提案者に51%の攻撃を成功させる必要があります。

2.1 安全性とその可能性の証明

私たちは、Casperの2つの基本的な特性、すなわち、accountable safety と d plausible liveness を証明しています。

Accountable safetyとは1/3が Slashing condition に違反しない限り(つまり、デポジット総額の少なくとも3分の1が失われていることを意味します)、2つの競合するチェックポイントをfinalize することはできません。

Plausible livenessとは、以前の出来事 (例えば、スラッシングイベント、遅延ブロック、検閲攻撃など) に関係なく、バリデータの2/3以上がプロトコルに従うならば、バリデータを違反することなく常に新しいチェックポイントをfinalizeすることができることを意味する、Slashing conditionのことです。

バリデーターの2/3が軽度のSlashing conditionに違反していないという前提の下で、我々は以下の特性を有します:

1. If s1 → t1 and s2 → t2 are distinct supermajority links, then h(t1) 6= h(t2).
2. s1→t1とs2→t2が異なる超大多数のリンクである場合、不等式h(s1) ¡h(s2)¡h(t2) ¡h(t1)は成り立たない。

   これらの2つの特性から、任意の高さnに対して、
3. h(t) = nで最大でも1つの超大多数のリンクs→tが存在する。
4. 高さnの正当化されたチェックポイントが最大でも1つ存在する。

この4つの特性を手にして、Main theoremsに移ります。

1. Theorem 1 (Accountable Safety). 2つの競合するチェックポイントamとbnの両方を確定することはできません。
2. Theorem 2 (Plausible Liveness). 完成したチェーンを拡張する子が存在する場合は、新しい大域チェックポイントを生成するために、Supermajority linksを常に追加することができます。

2.2 Casperのフォーク選択ルール

Casperは標準のPoW設計よりも複雑です。そのため、フォークの選択を調整する必要があります。変更されたフォーク選択ルールの後には、すべてのユーザー、バリデーター、さらには基礎となるブロックの提案メカニズムが必要です。

ユーザ、バリデーター、またはブロック提案者の代わりに、 ”常に最長のチェーン上に構築する”という標準のPoWフォーク選択ルールに従うと、Casperが“stuck” し、最長チェーンの上に構築されたブロックを finalize できない (もしくは justifiedされていても) 一部のバリデータが利他的にそのデポジットを犠牲にす

ることはありません。

これを避けるために、私たちは小説 Correct by Construction のフォーク選択ルールを紹介します：FOLLOW THE CHAIN CONTAINING THE JUSTIFIED CHECKPOINT OF THE GREATEST HEIGHT。

このフォーク選択ルールは、Correct by Constructionです。なぜなら、Plausible Livenessの証明(Theorem 2)から導かれるからです。

最大の高さを持つ justifiedなチェックポイントの上に新しいチェックポイントを常に finalize することが常に可能であると正確に述べています。

このフォーク選択ルールは、セクション3と4で調整されます。

3 Enabling Dynamic Validator Sets

Validators setは変更できる必要があります。

新しいバリデーターは参加できなければならず、既存のバリデーターは残しておく必要があります。

これを達成するために、私たちはブロックのDynastyを定義します。

ブロックbのDynastyは、ルートからブロックbの親までの連鎖内のfinalizeされたチェックポイントの数です。

バリデーターのdeposit messageがdのブロックに含まれている場合、バリデーターvは最初のブロックでd+2のvalidator setに参加します。このバリデーターのStart dynastyをd+2 と呼びます。DS(ν)

Validator setから脱ける場合には、バリデータが ”withdraw” message を送信する必要があります。

バリデーターνのwithdraw messageが Dynasty dのブロックに含まれている場合も同様に、最初のブロックに設定されたバリデーターはd+2のDynastyになります。

バリデータのEnd dynasty DE(ν) を d+2 と呼ぶ。

withdraw messageがまだ含まれていない場合は、DE(ν)=∞となります。

バリデーターνがValidator setsを離れると、バリデーターの公開キーはValidator setsに再び参加することを永久に禁止されます。

これにより、単一の識別子に対して複数のStart/Endのを処理する必要がなくなります。

End dynastyが始まる時、デポジットが回収される前に、バリデーターのデポジットは、withdrawal delayと呼ばれる長期間ロックされます（「4ヶ月分のブロック」と考える）。

withdrawal delayの間、バリデーターがいかなる命令にも違反した場合、デポジットはslashされます。

私たちは与えられたDynasty dに対して、the forward validator set, the rear validator set の2つのValidator Setsを生成する2つの関数を定義します。それらは、

Vf(d) ≡ ν : DS(ν) d ¡ DE(ν) Vr(d) ≡ ν : DS(ν) ¡ d DE(ν) .

Dynasty dが The forward validator set, d+1がThe rear validator setという意味であることに注意してください。

チェーンが自身の現在のDynastyを”知る” ことができるようにするためには“finalization”の定義をわずかに制限する必要があります。

チェックポイントCは、それがJustifiedされ、Cからチェックポイントツリー内の直接の子へのSupermajority linkが存在する場合、Finalizeされます。

ここで、Finalizeにはさらに1つの追加条件があります。cは、Supermajority link c → c’の投票と、cを再

帰的にJustified するSupermajority linkの全てがc’の子の前のブロックチェーンに含まれている場合、ブロック番号h(c’) * 100に前に置く。

Dynamic Validator Setsをサポートするために、Supermajority linkと finalizationを次のように再定義します。

1. チェックポイントの順序づけられたペア(s, t)があり、t がDynsaty dに含まれる場合、Dynasty dのThe forward validator setの2/3以上がs→t へのリンクに対して投票を発行し、同様に、Dynasty dのThe rear validator setの2/3以上のThe rear validator setもs→tへのリンクに対して投票を行なった場合、s→tのリンクはSupermajority linkを持つ
2. チェックポイントcは、cがJustify され、c→c’ (c’がcの子である) のSupermajority linkがある場合、cはFinalizeされます。c→c’のSupermajority linkへの投票が、c をJustify するSupermajority linkの投票がc’のブロックチェーンに含まれていて、c’の子の前にブロック番号h(c’) * 100

The forward validator setとThe rear validator setは大きく異なります。

2つのvalidator setsが実質的に異なる場合、“stitching” メカニズムは、その場合の安全の失敗を防止する。

証拠が1つのチェーンに含まれているが、他のチェーンには含まれていないため、 finalizeなチェックポイントの2人の孫が異なるDynastyを持つ。

この例については、図4を参照してください。

図4: dynamic validator setsからの攻撃. バリデータセット stitchingメカニズムがなければ、2つの競合するチェックポイントcとc’はバリデーターをslashしなくても両方をfinalizeすることができます。この場合、cとc’は同じ高さであり、したがってIの命題に違反しますが、バリデータセットcとc’が互いに素であるため、誰もslashしません。

4 攻撃の停止

Proof of Stakeシステムに対してよく知られた2つの攻撃があります。Long Range RevisionsとCastastrophic Crashesです。私たちはそれぞれを順番に話し合いました。

4.1 Long Range Revisions

withdrawal delay後のバリデーターは、バリデータークライアントとの間に同期生の仮定を導入します。

バリデーターの連合がデポジットを取り下げた場合、その連合が過去にデポジットの2/3以上を持っていた場合、歴史的な大多数を使って矛盾するチェックポイントを確定することができます。(彼らは既に彼らのお金を回収しているからです。)

これは、Long Range Revisions攻撃と呼ばれます。図5を参照してください。

簡単に言えば、finalizeされたブロックを元に戻さないフォーク選択ルールによって。

ロングレンジ攻撃が防止され、各クライアントが「ログオン」し、ある一定の頻度(例えば、１２ヶ月に1回)でチェーンの完全な最新のレビューを得られることを期待します。

それより古いブロックをfinalizeする「ロングレンジリビジョン」フォークは単に無視されますが、全てのクライアントが既にその高さでfinalizeされたブロックを見ていて、元に戻すことを拒否するため、無視され

ます。

これは、ロングレンジリビジョン攻撃と呼ばれます。図5を参照してください。

メカニズムの非公式な証明は次のように行います。仮定:

1. 2つのクライアント間に最大の通信遅延δがあるので、あるクライアントが時刻tで何らかのメッセージを聞くと、h五家のすべてのクライアントは時刻t+δまでにそれを聞いたことが保証される。これは、ブロックがネットワークによって受信された間の「時間ウィンドウ」[tmin, tmax] について話すことができ、幅tmax-tminが最大でδであることを意味する。
2. すべてのクライアントがローカルクロックを完全に同期させていると仮定します(相違点は通信遅延δの一部として扱うことができます)。
3. ブロックにはtimestampsが必要です。クライアントが現地時間TLを有する場合、タイムスタンプTBがTB ¿ TL（すなわち、将来）を満たすブロックを拒絶し、最終的に（ただしチェーンの一部として受け入れることができる）ブロックを受け入れることを拒絶するTB ¡ TL - δ（過去にあまりにも遠すぎる）
4. バリデーターが時刻tにスラッシュ違反を見た場合（それは2回の投票の後に聞こえる時です)、スラッシングのない証拠をまだ含んでいないチェーンの一部であるtimestamps ¿ t +2δのブロックを拒否します。

仮に、大量のslashing違反により、2つの矛盾したfinalizeされたチェックポイントc1とc2が発生するとします。

2つの時間ウィンドウが交差しない場合、すべてのバリデーターが最初にチェックポイントに合意し、誰もがルールに従ってfinalizeされたチェックポイントを元に戻さない場合は問題はありません。

図5: ロングレンジ攻撃。一定の間隔でクライアントがjustifyされたチェーンの完全な知識を得る限り、ロングレンジ攻撃の影響を受けません。

2つの時間ウィンドウが交差する場合、次のようにケースを処理できます。c1の時間窓を[0、 δ]、c2の時間窓を[δ - ε、2δ - ε]とする。

両方のタイムスタンプは少なくとも0です。時間2δまでに、すべてのクライアントがスラッシュ違反を見たことが保証されているので、証拠取引をまだ含まないチェーンを持つタイムスタンプ¿4δのブロックを拒否します。

したがって、ω¿4δであれば、悪意のあるバリデータは、クライアントが受け入れるすべてのチェーンにおいて預金を失うことが保証されます。

ここで、ωは「引き出し遅延」(図5)、エンドエポックと検証者間の遅延彼らの預金を受け取る。

ネットワーク遅延のために、特定のチェーンに ”断りのない”証拠が提出されたのか、それとも遅すぎたのかをクライアントが合意しない可能性があります。

しかし、これは安全性の失敗ではなく、ただの生き残りの失敗であり、この可能性は、（証拠の混入を防ぐために必要となる）破損した提案メカニズムが最終性を妨げる可能性があることが既に分かっている。

私たちはまた、攻撃が短期間になると非公式に主張することで、包括的タイムアウトの問題を回避することができます。

なぜなら、バリデーターは証拠をスラッシュして攻撃とみなし、攻撃の一部ではない真正な小数のバリデーターによってサポートされている別のブランチに切り替えることなく(長時間実行しているチェーンを知るため)攻撃を止めて攻撃者を壊すことなく、長期的な連鎖を認識するからです。

4.2 Castastrophic Crashes

バリデーターの1/3以上が同時にクラッシュフェイルしたとします。つまり、ネットワークパーティション、コンピュータの障害、またはバリデータ自体が悪意のあるためにネットワークに接続されなくなったとします。

この時点から直感的に言えば、Supermajority linksは作成できないため、将来のチェックポイントをFinalizeすることはできません。

私たちは、チェックポイントに投票しないバリデーターのデポジットをゆっくりと流出させる”inactivity leak”を導入することで、立て直すことができます。

結局のところ、デポジットの規模は、投票しているバリデーターが超大多数であるほど、十分に低くなるまで減少します。

最も単純な式は次のようなものです。”すべての時代のデポジットサイズDのバリデーターは投票に失敗し、D*p(fore 0 ¡ p ¡ 1)を失います”

Catastrophic Crashesをより早く解決するには、Finalizeされていないブロックの長いストリークが発生した場合のリーク率を高める式が最適です。

この流出したetherは、 ω日後に焼却か、バリデーターに戻すことができます。

流れ出た資産を焼却するか戻すか、そして”inactivity leak”の正確な公式は、ビザンチンフォールトトレランスではなく、経済的インセンティブの問題であるため、このペーパーの範囲外です。

Inactivity leakは、2つのチェックポイントのうちの1つのみでバリデーターを失うだけで、バリデーターをスラッシングせずに、2つの競合するチェックポイントがFinalizeされる可能性をもたらします(図6)。

バリデーターが2つのサブセットに分割され、サブセットVAがチェーンAに投票し、サブセットVBがチェーンBに投票しているものとします。

チェーンA上では、VBのデポジットはリークし、その逆もあり、各サブセットはそれぞれのチェーン上に大多数を持ち、2つの矛盾するチェックポイントを明示的にスラッシュせずにFinalizeすることができます(ただし、各サブセットはリークによる2つのチェーンの内いずれか1つでそれらの大部分のデポジットを失うことになります)。

このような状況が発生した場合、各バリデーターは、最初に確認した最終チェックポイントを単純に優先する必要があります。

これらの様々な攻撃から回復するための正確なアルゴリズムは、未解決の問題として残っています。今のところ、バリデーターは明らかに不正な行為（証拠を含まないなど）を検出し、手動で“minority soft fork”を作成できると仮定しています。

このマイノリティフォークは、市場の大多数のチェーンと競合する独自のブロックチェーンと見ることができます。

多数のチェーンが本当に悪意のある攻撃者と結びついて運営されている場合、市場はマイノリティ・フォークを支持すると見なすことができます。

図6：Inactivity leak。左のチェックポイントはすぐにFinalizeすることができます。

オフラインバリデーターのデポジットが十分に消耗したら、しばらくして右のチェックポイントを確定することができます。

5 結論

私たちはビザンチンフォールトトレランスに関する文献から導き出された、Proof of StakeシステムであるCasperを紹介しました。

Casperには、2つのslashing conditions、[11]によってインスピレーションを受けた、correct-by-constructionのフォーク選択ルール、そして、dynamic validator setsが含まれます。

最後に2つの一般的な攻撃から防御する、Casperへの拡張機能 (not reverting finalized checkpointsとthe inactivity leak) を紹介しました。

Casperは未完成です。例えば、完全に妥協したブロック提案の仕組みは、Casperが新しいブロックをFinalizeするのを防ぐでしょう。

CasperはほぼすべてのPoWチェーンに対するPoSベースの厳重なセキュリティー強化です。

Casperが完全には解決しない問題、特に51%の攻撃に関連する問題は、user-activated soft forksを用いて修正できます。

将来の開発では間違いなくCasperのセキュリティを向上させ、user-activated soft forksの必要性を減らします。

Future Work. 現在のCasperシステムは、Proof of Workのブロック提案メカニズムをベースにしています。

私たちはブロック提案メカニズムをProof of Stakeに変換したいと考えています。

私たちはValidator setの重みが報酬と罰則に変わった場合でも、Accountable safetyとPlausible livenessを証明したいと考えています。

今後の課題は、Proof of Stakeの一般的な攻撃を考慮したフォーク選択ルールの正式な仕様です。

今後のワークペーパーでは、Casperの金銭的インセンティブとその結果を説明し、分析します。

攻撃者をブロックするためのこのような自動化された戦略に関連する特定の経済的問題は、異なるクライアント間の不一致度と攻撃者が負担するコストの間の比率の上限を証明しています。

Acknowledgements. 私たちは頻繁に議論してくれたJon Choi、Karl Floersch、Ozymandias Haynes、Vlad Zamfirに感謝します。

6 参考文献

[1] Pass, R. & Shi, E. Fruitchains: A fair blockchain. In Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, 315–324

[2] Introducing dfinity crypto techniques (2017). URL https://dfinity.org/pdf-viewer/pdfs/viewer.html?file=../library/intro-dfinity-crypto.pdf.

[3] King, S. & Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake 19 (2012). URL https: //decred.org/research/king2012.pdf.

[4] Vasin, P. Blackcoin’s proof-of-stake protocol v2 (2014). URL http://blackcoin.co/ blackcoin-pos-protocol-v2-whitepaper.pdf.

[5] Bentov, I., Gabizon, A. & Mizrahi, A. Cryptocurrencies without proof of work. In Sion, R. (ed.) International Conference on Financial Cryptography and Data Security, 142–157 (Springer, 2016).

[6] Castro, M., Liskov, B. & et. al. Practical byzantine fault tolerance. In Leach, P. J. & Seltzer, M. (eds.) Proceedings of the Third Symposium on Operating Systems Design and Implementation, vol. 99, 173–186 (1999).

[7] Kwon, J. Tendermint: Consensus without mining (2014). URL https://tendermint.com/static/docs/tendermint.pdf.

[8] Chen, J. & Micali, S. ALGORAND: the efficient and democratic ledger. CoRR abs/1607.01341 (2016). URL http://arxiv.org/abs/1607.01341.

[9] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash systems (2008). URL https://bitcoin.org/bitcoin.pdf.

[10] Buterin, V. Minimal slashing conditions (2017). URL https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c.

[11] Sompolinsky, Y. & Zohar, A. Accelerating bitcoin' s transaction processing. fast money grows on trees, not chains. 2013 (2013).

Japanese Transalation / 日本語訳

Technical Transalators / 技術翻訳

Abstract / Kazuki Yamaguchi
1. Introduction / Kazuki Yamaguchi
2. The Casper Protocol / Kazuki Yamaguchi
3. Enabling Dynamic Validator Sets / Kazuki Yamaguchi
4. Stopping Attacks / Kazuki Yamaguchi
5. Conclusions / Kazuki Yamaguchi
References / Kazuki Yamaguchi

Contributors / 修正協力