

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7536796号
(P7536796)

(45)発行日 令和6年8月20日(2024.8.20)

(24)登録日 令和6年8月9日(2024.8.9)

(51)国際特許分類

F I

H O 4 L 9/32 (2006.01) H O 4 L 9/32 2 0 0 Z

請求項の数 26 (全69頁)

(21)出願番号	特願2021-569490(P2021-569490)	(73)特許権者	318001991
(86)(22)出願日	令和2年4月22日(2020.4.22)		エヌチェーン ライセンシング アーゲー
(65)公表番号	特表2022-533777(P2022-533777 A)		スイス・6 3 0 0・ツーク・グラーフエ ナウヴェーク・6
(43)公表日	令和4年7月25日(2022.7.25)	(74)代理人	100107766
(86)国際出願番号	PCT/IB2020/053800		弁理士 伊東 忠重
(87)国際公開番号	WO2020/240293	(74)代理人	100070150
(87)国際公開日	令和2年12月3日(2020.12.3)		弁理士 伊東 忠彦
審査請求日	令和5年3月24日(2023.3.24)	(74)代理人	100135079
(31)優先権主張番号	1907392.3		弁理士 宮崎 修
(32)優先日	令和1年5月24日(2019.5.24)	(72)発明者	ワハブ, ジャド
(33)優先権主張国・地域又は機関	英国(GB)		イギリス ダブリュー1ダブリュー 8エ ービー ロンドン マーケット プレイス 3 0 エヌチェーン ホールディングズ リミテッド 内
		最終頁に続く	

(54)【発明の名称】 プルーフ・オブ・ワーク

(57)【特許請求の範囲】

【請求項1】

ブロックチェーンネットワークのノード検証において、コンピュータに実装される方法であって、

実行可能コードを含む第1トランザクションを取得するステップと、
情報を含む第2トランザクションを受信するステップであり、前記情報は、第1 E C D S A 署名の r 部および s 部の少なくとも提出されたインスタンスと、ノンスと、をさらに含む、ステップと、

第1トランザクションからのコードを実行するステップであり、前記コードは、H P o W (f (r , d)) が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されている、ステップと、を含み、

r は前記 r 部の提出されたインスタンスであり、d はノンスであり、H P o W はハッシュ関数であり、f は r および d を組み合わせた関数である、

方法。

【請求項2】

f は、連結 $r || d$ である、
請求項1に記載の方法。

【請求項3】

前記事前決定された条件は、 $H p u z (r || d)$ が、事前決定されたターゲット値未満

であること、または、少なくとも事前決定された先頭ゼロの最小数を有することである、請求項 1 または 2 に記載の方法。

【請求項 4】

前記方法は、さらに、

公開鍵を取得するステップであり、前記第 1 E C D S A 署名は、前記公開鍵に対応する秘密鍵に基づいてメッセージに署名し、前記メッセージは、前記第 2 トランザクションの一部である、ステップと、

E C D S A 検証機能を適用するステップであり、前記公開鍵と前記メッセージに基づいて、前記第 2 トランザクションで受信した前記第 1 E C D S A 署名を検証し、

前記コードは、前記第 1 E C D S A 署名の前記検証のさらなる条件について真の結果を返すように構成されている、ステップと、を含む、

請求項 1 乃至 3 いずれか一項に記載の方法。

【請求項 5】

前記コードは、さらに、前記第 1 E C D S A 署名の前記 r 部に対応する参照値を含み、前記参照値は、前記 r 部の参照インスタンスまたはその変換であり、

前記コードは、前記参照値が前記第 2 トランザクションで受信した r 部の前記参照インスタンスに対応することを検査し、かつ、さらなる条件で真の結果を返すように構成されている、

請求項 4 に記載の方法。

【請求項 6】

前記参照値は、前記 E C D S A 署名の前記 r 部の参照インスタンスである、

請求項 5 に記載の方法。

【請求項 7】

前記コードは、

前記 $H P o W (f(r, d))$ が、前記事前決定された条件を満たしていることを検査し、かつ、

前記 E C D S A の検証機能が、

$$R' = H s i g (m) s - 1 \cdot G + r' s - 1 \cdot P \text{ を計算し、}$$

$[R']_x = r'$ であることを検査する、

ように構成されており、

前記 r' は前記第 1 E C D S A 署名の r 部の前記参照インスタンスであり、 s は前記第 1 E C D S A 署名の前記 s 部であり、 P は第 1 公開鍵であり、 m は前記第 1 E C D S A 署名によって署名された前記第 2 トランザクションの部分であり、 $H s i g$ は前記第 1 E C D S A 署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' の X 座標を表し、「 $\cdot$ 」は楕円曲線スカラ乗算を表しており、

前記コードは、前記検査の両方が真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている、

請求項 6 に記載の方法。

【請求項 8】

前記コードは、

前記 $H P o W (f(r, d))$ が、前記事前決定された条件を満たしていること、および、 $r' = r$ であることを検査し、かつ、

前記 E C D S A の検証機能が、

$$R' = H s i g (m) s - 1 \cdot G + r s - 1 \cdot P \text{ を計算し、}$$

$[R']_x = r$ であることを検査する、

ように構成されており、

前記 r は前記第 1 E C D S A 署名の r 部の前記提出されたインスタンスであり、前記 r' は前記第 1 E C D S A 署名の r 部の前記参照インスタンスであり、 s は前記第 1 E C D S A 署名の前記 s 部であり、 P は第 1 公開鍵であり、 m は前記第 1 E C D S A 署名によって署名

された前記第2トランザクションの部分であり、 $Hsig$ は前記第1ECDSA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' の x 座標を表し、「 $\cdot$ 」は楕円曲線スカラ乗算を表しており、

前記コードは、前記検査の3つ全てが真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている、

請求項6に記載の方法。

【請求項9】

前記参照値は、前記第1ECDSA署名の r 部の参照インスタンスの変換であり、

前記コードは、前記提出されたインスタンスにおいて同じ変換を実行し、かつ、前記参照値と比較することによって、前記提出されたインスタンスが前記参照値に対応することを検査するように構成されている、

請求項5に記載の方法。

【請求項10】

前記参照値は、ハッシュ値であって、前記第1ECDSA署名の r 部の前記参照インスタンスのハッシュである、

請求項9に記載の方法。

【請求項11】

前記コードは、

前記 $HPoW(f(r, d))$ が、前記事前決定された条件を満たしていること、および、 $h = H_{puz}(r)$ であることを検査し、かつ、

前記ECDSAの検証機能が、

$R' = Hsig(m)s^{-1} \cdot G + rs^{-1} \cdot P$ を計算し、

$[R']_x = r$ であることを検査する、

ように構成されており、

ここで、 r' は前記第1ECDSA署名の r 部の前記参照インスタンスであり、 r は前記第1ECDSA署名の r 部の前記提出されたインスタンスであり、 s は前記第1ECDSA署名の前記 s 部であり、 h はハッシュ値であり、 H_{puz} は h を生成するように r' をハッシュするために使用されたハッシュ関数であり、 P は第1公開鍵であり、 m は前記第1ECDSA署名によって署名された前記第2トランザクションの部分であり、 $Hsig$ は前記第1ECDSA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' の x 座標を表し、「 $\cdot$ 」は楕円曲線スカラ乗算を表しており、

前記コードは、前記検査の3つ全てが真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている、

請求項10に記載の方法。

【請求項12】

第1公開鍵を取得することは、前記第2トランザクションにおける情報の一部として前記第1公開鍵を受信すること、を含む、

請求項4に記載の方法。

【請求項13】

第1ECDSAの r 部および s 部の提出されたインスタンスは、第1当事者によって第2当事者に与えられた一時的鍵、またはその逆も同様、および、前記第2当事者の秘密鍵である第1秘密鍵を使用して、前記第2当事者によって生成されたものであり、かつ、

前記ノンスは、また、前記第2当事者のコンピュータ装置においてプルーフ・オブ・ワークを前記第2当事者が実行すること生成されたものである、

請求項1乃至12いずれか一項に記載の方法。

【請求項14】

$P = V \cdot G$ 、

$k \in [1, n-1]$ 、

$R = k \cdot G$ 、

10

20

30

40

50

$r = [R] x$ 、かつ、

$s = k - 1 (Hsig(m) + rV) \bmod n$ 、であり、

ここで、 P は第1公開鍵であり、 V は前記第1秘密鍵であり、 k は一時的鍵であり、 n は素数係数であり、 G は楕円生成点であり、 m は前記第1ECDSA署名によって署名された前記第2トランザクションの部分であり、 $Hsig$ は前記第1ECDSA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 $[R]x$ は R の x 座標を表し、かつ、「 $\cdot$ 」は楕円曲線スカラー乗算を表している、

請求項13に記載の方法。

【請求項15】

前記第2トランザクションを受信することは、前記第2当事者から前記第2トランザクションを受信することを含む、

請求項13または14に記載の方法。

【請求項16】

前記コードによって返される結果が真であることを条件として、前記第1当事者のためのサービスをトリガすること、を含む、

請求項13乃至15いずれか一項に記載の方法。

【請求項17】

前記第2トランザクションで受信した前記情報は、前記第2当事者のさらなる秘密鍵を使用して前記第2トランザクションの一部に署名する前記第2当事者のさらなる暗号署名を含み、

前記さらなる秘密鍵は、さらなる公開鍵に対応している、

請求項13乃至16いずれか一項に記載の方法。

【請求項18】

前記さらなる公開鍵に基づいて、前記第1当事者及び／又は第三者が、前記第2当事者の識別情報を探索することを可能にするマッピングが利用可能である、

請求項17に記載の方法。

【請求項19】

前記コードは、前記さらなる公開鍵を使用して、さらなる暗号署名を検証し、かつ、前記さらなる暗号署名が検証されることを条件として、真の結果を返すように構成されている、

請求項17または18に記載の方法。

【請求項20】

前記第2トランザクションで受信した前記情報は、さらに、前記第1当事者の秘密鍵を使用して前記第2トランザクションの一部に署名する、前記第1当事者の暗号署名、を含む、

請求項17乃至19いずれか一項に記載の方法。

【請求項21】

前記第2トランザクションで受信した情報は、前記第1ECDSA署名とは異なる r 部の値を有しているが、前記第1ECDSA署名と同じ第1秘密鍵を使用している、追加的ECDSA署名を含み、

前記コードは、前記第1公開鍵を使用して使って前記追加的ECDSA署名を検証し、かつ、前記追加的ECDSA署名が検証されることを条件に、真の結果を返すように構成されている、

請求項13乃至20いずれか一項に記載の方法。

【請求項22】

前記トランザクションそれぞれは、1つ以上の入力および1つ以上の出力を含むデータ構造を備え、各出力はロッキングスクリプトを含み、かつ、各入力は、アンロッキングスクリプトおよび別のトランザクションの出力に対するポインタを含み、

前記コードは、前記第1トランザクションのロッキングスクリプトによって構成され、前記第2トランザクションで受信した前記情報は、前記第2トランザクションの入力にお

10

20

30

40

50

ける前記アンロッキングスクリプトによって構成され、かつ、前記第2トランザクションの前記入力における前記ポインタは、前記第1トランザクションの前記出力を指し示し、前記方法は、

前記コードが真の結果を返すことを少なくとも条件として、前記トランザクションを検証するステップを含み、かつ、

前記検証に応答して、

検証ノードによる1つ以上のブロックへのマイニングのためのトランザクションのプールに前記第2トランザクションを含むステップ、及び／又は、

前記第2トランザクションをブロックチェーンネットワークの少なくとも1つの他のノードに転送ステップ、

のうち少なくとも1つを含む：

請求項1乃至21いずれか一項に記載の方法。

【請求項23】

前記トランザクションは、アカウントベースのモデルに従って構成されており、

前記コードは、前記第1トランザクションに含まれるスマートコントラクトによって構成されている、

請求項1乃至21いずれか一項に記載の方法。

【請求項24】

コンピュータ読取り可能記憶装置において具体化され、かつ、ネットワークのノード上で実行されると、請求項1乃至23いずれか一項に記載の方法を実行するように構成されている、コンピュータプログラム。

【請求項25】

ネットワークのノードであって、

1つ以上のメモリユニットを含むメモリと、

1つ以上の処理ユニットを含む処理装置と、を備え、

前記メモリは、前記処理装置上で実行するように構成されたコードを保管しており、

前記コードは、前記処理装置において請求項1乃至23いずれか一項に記載の方法を実施するように構成されている、

ノード。

【請求項26】

第2当事者のコンピュータ装置において、コンピュータに実装される方法であって、

実行可能なコードを含む第1トランザクションを監視するステップであり、

前記コードは、 $HPoW(f(r, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、

ここで、 r はEDSCA署名の r 部であり、 d はノンスであり、 $HPoW$ はハッシュ関数であり、 f は r および d を組み合わせた関数であり、前記 r 部は第1当事者によって指定される、

ステップと、

一時的鍵に基づいて、前記 r 部を生成するステップと、

$HPoW(f(r, d))$ が前記事前決定された条件を満たすように、前記ノンス d の値を探索するステップと、

前記第1トランザクションにリンクされた第2トランザクションを策定するステップであり、

前記第2トランザクションは、第1EDSCA署名の少なくとも r 部および s 部と、さらに、前記ノンス d とを含む、情報を含む、

ステップと、

ブロックチェーンに記録するために、ブロックチェーンネットワークを介して伝搬される、前記第2トランザクションを送信するステップと、

を含む、方法。

10

20

30

40

50

【発明の詳細な説明】

【技術分野】

【0001】

本開示は、ブロックチェーンのコンテキストにおけるプルーフ・オブ・ワーク(proof-of-work、PoW)の概念に関する。

【背景技術】

【0002】

ブロックチェーンとは、ピアツーピア(peer-to-peer、P2P)ネットワーク内の複数のノードそれぞれにおいて、ブロックチェーンの重複コピーが維持される、分散データ構造の形態を指す。ブロックチェーンは、データのブロックのチェーンを含み、各ブロックは、1つ以上のトランザクションを含む。各トランザクションは、シーケンス内の先行トランザクションを指し示すことができる。新しいブロックに含めるために、トランザクションをネットワークにサブミットできる。新しいブロックは、「マイニング(“mining”）」として知られているプロセスによって生成され、これは、「プルーフ・オブ・ワーク(“proof-of-work”）」を実行するために競合する複数のマイニングノードそれぞれを含み、すなわち、ブロックに含まれるのを待つ保留中のトランザクションのプールに基づいて暗号パズルを解決する。

【0003】

従来、ブロックチェーン内のトランザクションは、デジタル資産、すなわち、価値のストアとして機能するデータを伝達するために使用される。しかしながら、ブロックチェーンの上に追加の機能をレイヤ(layer)するために、ブロックチェーンを利用することもできる。例えば、ブロックチェーンプロトコルは、トランザクションの出力に追加のユーザデータを保管することを可能にする。現代のブロックチェーンは、単一トランザクション内に保管できる最大データ容量を増加させ、より複雑なデータを組み込むことを可能にしている。例えば、これは、ブロックチェーン内に電子文書を保管するために使用されてよく、または、オーディオまたはビデオデータでさえも保管する。

【0004】

ネットワーク内の各ノードは、フォワーディング(forwarding)、マイニング(mining)、およびストレージ(storage)の3つの役割のいずれか1つ、2つ、または全てを持つことができる。転送ノードは、ネットワークのノード全体にトランザクションを伝播させる。ノードをマイニングすると、トランザクションをブロックにマイニングする。ストレージノードは、それぞれ、ブロックチェーンのマイニングされたブロックのコピーを保管する。トランザクションをブロックチェーンに記録させるために、当事者は、伝搬されるネットワークのノードの1つにトランザクションを送信する。トランザクションを受信するマイニングノードは、新しいブロックにトランザクションをマイニングするために競合する可能性がある。各ノードは、同じノードプロトコルを尊重するように設定されている。このプロトコルには、トランザクションが有効である1つ以上の条件が含まれる。無効なトランザクションは、ブロックに伝搬されたり、マイニングされたりしない。トランザクションが検証され、それによってブロックチェーンに受け入れられたと仮定すると、トランザクション(任意のユーザデータを含む)は、P2Pネットワークの各ノードに不変のパブリックレコードとして保管されたままである。

【0005】

プルーフ・オブ・ワークパズルをうまく解いて最新のブロックを作ったマイナは、典型的には、デジタル資産の新たな量を生み出す「生成トランザクション(“generation transaction”）」と呼ばれる新しいトランザクションで報われる。プルーフ・オブ・ワークは、ブロックのマイニングに大量のコンピュータリソースを必要とし、二重支払い(double-spending)を含むブロックは他のノードには受け入れられない可能性が高いため、ブロックに二重支払いトランザクションを含めることによって、労働者がシステムを詐称(cheat)しないようにインセンティブを与える。

【0006】

「出力ベース（“output-based”）」モデル(UTXOベースモデルとも呼ばれる)において、トランザクションのデータ構造は、1つ以上の入力と1つ以上の出力から構成されている。使用可能な出力は、ときどきUTXO(「未使用トランザクション出力（“unspent transaction output”）」)と呼ばれるデジタル資産の量を指定する要素(element)で構成される。出力は、さらに、出力を償還(redeeming)するための条件を指定するロッキングスクリプトを含んでもよい。各入力は、先行トランザクションにおけるそのような出力へのポインタを含み、さらに、指し示された出力のロッキングスクリプトをアンロックするためのアンロッキングスクリプトを含んでもよい。そこで、2つのトランザクションを1回目と2回目のトランザクションと呼ぶ。第1トランザクションは、デジタル資産の量を指定する少なくとも1つの出力を含み、出力をアンロックする1つ以上の条件を定義するロッキングスクリプトを含む。第2トランザクションは、第1トランザクションの出力へのポインタと、第1トランザクションの出力をアンロックするためのアンロッキングスクリプトとを含む、少なくとも1つの入力を含む。

【0007】

このようなモデルでは、第2トランザクションがブロックチェーンで伝搬され記録されるP2Pネットワークに送られるとき、各ノードで適用される有効性の条件の1つは、アンロッキングスクリプト(unlocking script)が第1トランザクションのロッキングスクリプト(locking script)で定義された1つ以上の条件の全てを満たすことである。もう1つは、第1トランザクションの出力が、以前に有効であった別のトランザクションによってまだ償還されていないことである。これらの条件のいずれかに基づいて第2トランザクションが無効であることを発見したノードは、それを伝搬せず、ブロックチェーンに記録されるブロックにマイニングするように、それを含まない。

【0008】

代替的なタイプのトランザクションモデルは、アカウントベースのモデルである。この場合、各トランザクションは、過去のトランザクションのシーケンスで先行トランザクションのUTXOに戻すことにより、むしろ、絶対的な残高(account balance)を参照することによって、移転される金額を定義しない。全てのアカウントの現在の状態は、マイナによって、ブロックチェーンに分離されて保管され、そして、絶えず更新される。また、アカウントベースモデルのトランザクションは、トランザクションの検証と同時に各ノードで実行されるスマートコントラクト(smart contract)を含むことができる。

【0009】

どちらのモデルのトランザクションにも知識証明を含めることができる。「知識証明(“knowledge proof”または“proof of knowledge”）」とは、当事者が何らかのデータを知っているというテスト(例えば、それを d と呼ぶ)を指す技術用語である。出力ベースのトランザクションモデルの場合の例として、1つのトランザクション Tx_1 の出力におけるロッキングスクリプトは、ハッシュパズルを含むことができる。第2トランザクション Tx_2 の入力が Tx_1 のこの出力を指している場合、 Tx_2 の入力のアンロッキングスクリプトは、ハッシュパズルを解決しなければ、 Tx_1 の出力を償還することができない。ハッシュパズルは、 d のハッシュである、ハッシュ値 h を含む。すなわち、 $h=H_{puz}(d)$ である。このパズルは、また、ノードで Tx_2 のアンロッキングスクリプトと一緒に実行すると、 Tx_2 のアンロッキングスクリプトから d であると称するデータ値 d' をとり、それをハッシュ関数 H_{puz} と共にハッシュ化し、そして、 Tx_1 のアンロッキングスクリプトに含まれるハッシュ値 h と比較するスクリプトを含む。すなわち、比較の結果がイエス(または、当該技術分野の用語において「真(“true”）」)である場合にのみ、 $h \neq H_{puz}(d')$ であるか否かをチェックして、 Tx_1 の出力のロックをアンロックだけする。従って、 Tx_2 の受益者(beneficiary)は、 d の知識証明のためにするため Tx_2 のアンロッキングスクリプトに d が含まれている場合、 Tx_1 の出力のみをアンロックすることができる。

【0010】

従来のハッシュパズルを単独で使用する問題のことは、悪徳なマイナまたは他のノードが、 Tx_2 のアンロッキングスクリプト内の d を観察することができ、次いで、 Tx_2 の自分自

10

20

30

40

50

身のバージョン Tx_2^* を作成して、マイニング（またはパブリッシュ）し、 Tx_2 における意図された受信者（例えば、ボブ）の代わりに Tx_2^* の出力において自分自身で支払うことである。これを避けるための既存の方法は、 Tx_1 のロッキングスクリプト内に「pay-to-public key hash」（P2PKH）の要件を追加することである。これは、dのための知識証明に加えて、意図された支払先の暗号署名が、 Tx_2 のアンロッキングスクリプトに含まれることを必要とする。

【0011】

ハッシュパズルとP2PKHは、また、出力ベースモデルのロッキングスクリプトおよびアンロッキングスクリプトではなく、アカウントベースモデルのスマートコントラクトを使用して実装することもできる。

【0012】

当業者によく知られているように、暗号署名（cryptographic signature）は、秘密鍵 V に基づいて生成され、そして、秘密鍵と公開鍵のペアの対応する公開鍵 P に基づいて検証され得る。秘密鍵 V をメッセージ m に適用することによって生成された署名が与えられた場合、他の当事者は、 P を使用して、その当事者が V を知ることなく、その署名が V を使用して生成されたことを検証することが可能である（従って、署名自体を検証することは、それ自身の権利において、知識証明の別の形態である）。

【0013】

このためのアルゴリズムの一つの形態は、楕円曲線暗号（elliptic curve cryptography, ECC）に基づいて動作する楕円曲線デジタル署名アルゴリズム（ECDSA）である。この場合、 P および V は、以下によって関連付けられる。

$$P=V \cdot G$$

ここで、 P は2つの要素ベクトル（ P_x, P_y ）であり、 V はスカラーであり、および、 G は二次元楕円曲線上の既定の点（「生成点（“generator point”）」）を表す2つの要素ベクトル（ G_x, G_y ）である。演算“ $\cdot$ ”は、スカラー楕円曲線乗算であり、一楕円曲線上のある点から別の点へ変換される既知の演算形式である。

【0014】

ECDSA署名は、当該技術において、それぞれ r 部(r)および s 部(s)として一般的に知られている2つの要素からなるタプレット（tuple）（ r, s ）である。署名(r, s)は、秘密鍵 V をメッセージ m に適用することによって生成される。ブロックチェーン上の記録トランザクションの場合、 m はトランザクションの一部となり、そして、トランザクションの検証を可能にするために、トランザクションの一部に加えて、署名がトランザクション上に明確にタグ付けされる。例えば、出力ベースのモデルにおいて、署名は、 Tx_1 の出力のアンロッキングするために、 Tx_2 の一部に署名し、そして、 Tx_2 のロッキングスクリプトに含まれる。署名された部分は、通常、トランザクションの出力を含むため、署名、従ってトランザクションを無効にすることなく、これらを変更することはできない。

【0015】

どのトランザクションモデルが使用されても、署名(r, s)は、次のように計算される。

$$r=[R]_x、ここで、R=k \cdot G$$

$$s=k^{-1}(H_{sig}(m)+rV) \bmod n$$

ここで、 $[R]_x$ は2つの要素ベクトル $R=(R_x, R_y)$ の x 座標を示す。 K は一時（ephemeral）キーとして知られており、それは、典型的にはランダムに、集合 $[1, n-1]$ から選択される。ここで、 n はプライムモジュラス（prime modulus）であり、そして、 $[1, n-1]$ は1から $n-1$ の包括的な範囲の実数の集合である。 H_{sig} はハッシュ関数であり、ハッシュパズルで使用するハッシュ関数 H_{puz} と比較して、同一または異なる形式のハッシュ関数であり得る。

【0016】

署名(r, s)、メッセージ m 、および公開鍵 P の知識を与えられれば、秘密鍵 V を知らない当事者は誰でも、その署名がメッセージ m に対する秘密鍵 V を使って生成されたことを検証することが可能である。これは、以下の計算によるものである。

$$R'=H_{sig}(m)S^{-1} \cdot G+rs^{-1} \cdot P$$

10

20

30

40

50

そして、 $[R']_x=r$ であることを検証する。署名は、これが正しい場合にのみ有効であるが、そうでない場合には有効ではない。これは、公開鍵Pに関連した当事者が実際にメッセージの署名者であったことの検証と見なすことができる。

【発明の概要】

【0017】

ブロックチェーンネットワークでは、トランザクションをブロックにマイニングするために必要なプルーフ・オブ・ワークは、ネットワーク内の各マイニングノードによって適用されるノードプロトコルの固有の特徴である。しかしながら、ここにおいては、追加の「レイヤ2 (“layer2”)」のプルーフ・オブ・ワークを追加することが望ましいことが認められている(すなわち、基本ノードプロトコルの上に層化されている)。特に、本開示によれば、これは、ECDSA署名のr-部に基づいて、ここにおいては「rパズル (“r-puzzle”)」と称されるタイプのパズルの変形でTx₁におけるチャレンジを設定することによって達成することができる。

【0018】

ここにおいて開示される一つの態様に従って、ブロックチェーンネットワークのノード検証において、コンピュータに実装される方法が提供される。本方法は、実行可能コードを含む第1トランザクションを取得するステップと、情報を含む第2トランザクションを受信するステップであり、前記情報は、第1ECDSA署名のr部およびs部の少なくとも提出されたインスタンスと、ノンスと、をさらに含む、ステップと、第1トランザクションからのコードを実行するステップ、を含む。前記コードは、 $H_{pow}(f(r, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、rは前記r部の提出されたインスタンスであり、dはノンスであり、H_{pow}はハッシュ関数であり、fはrおよびdを組み合わせた関数である。

【0019】

任意の実施形態では、第1トランザクションのコードに設定されたrパズルは、第2トランザクションにr部の特定の値を含めることを要求するチャレンジをさらに含んでよい。従って、プルーフ・オブ・ワークと同様に、rパズルは、また、署名のr部に基づく知識証明を実装するために使用することもできる。

【0020】

バックグラウンドで議論された脆弱性を避けるためにP2PKHと組み合わせられた場合、知識証明としてハッシュパズルを使用することに問題が存在する。それにより、ノードは、そうでなければ、dを見て、証明者の代わりにノードオペレータに支払う第2トランザクションの新しいバージョンへと交換 (swap) することができる。つまり、P2PKHは、dを最初に知っていた当事者にのみ支払いが行われることを保証するが、それは、また、Tx₁の出力が、特定の事前に決定された受取人または受取人のセットに結び付けられていることも意味する(代替の受取人を含む可能性のある代替条件を指定することは可能であるが、それらの受取人は、未だ、事前に特定されていなければならない)。

【0021】

ここにおいて認識されているように、特定の秘密価値の知識を証明することができるが、その価値を明らかにすることを回避する方法で、不特定の当事者によって償還可能なトランザクションを認めることが望ましい。例えば、アリスは秘密鍵を渡す誰もがアンロッキングされ得る第1トランザクションを設定したいが、相手を事前に指定したくないとする。例えば、第1トランザクションは、アリスに代わって手紙や小包のような何らかの物品の引渡しを受け取るために誰かに支払うか、かつ／あるいは、物品を引渡すために配達会社に支払うことができる。第1トランザクションは、秘密の知識を証明する第2トランザクションによって償還することができる。配達を注文する時点で、アリスは第1トランザクションを設定し、配達会社に提供するか、ブロックチェーンに公表することができる(または、配達会社が第1トランザクションを作成できるように必要な詳細を提供するだけでよい)。従って、配達会社は、配達が完了すれば支払いが行われると確信している。しか

しながら、アリスは、この段階で、誰が自分のために配達を受け取るかを決定することを望んでいない。その代わりに、彼女は、後になってから、1人以上の信頼されている相手(例えば、彼女の同居人であるチャーリーやボブは、配達日に彼らが居ることを確認した)に秘密の価値を提供する。これは、アリスの秘密の価値を証明する第2トランザクションを提供することによって、誰もが彼女に代わって署名することを可能にする。

【0022】

これは、例示的な一つの例に過ぎないことが理解されるであろう。別の例として、トランザクションは、同意事項に対する同意を示すことに使用され得る。アリスは、現在、合意を設定したいと思うかもしれないが、その場合、その事実が、1つ以上の信頼される当事者のサブセットに、署名権限または委任状を彼女に代わって署名するように与えることを決定した後だけである。例えば、合意を設定した時点でアリスは自分に署名するつもりだったかもしれないが、その後で、彼女の精神的能力が失われているか、または、何らかの理由で署名することができなくなることがわかったときには、他の誰かに委任状を割り当てる必要がある(この場合、ボブとチャーリーは彼女の家族または仕事仲間であり得る)。

【0023】

より一般的に、主なPoWのアイデアには必須ではないが、従来のハッシュパズルに代わるものを提供することも望ましいだろう。特に、パズルの代替的な形態は、ノードにその価値を明らかにし、または、ブロックチェーン上でそれを公開することなく、秘密価値の知識の証明を可能にし、それは、特定の身元(identity)に結び付されない。

【0024】

これに対処するために、ここにおいて開示される任意の実施形態に従って、本方法は、公開鍵を取得するステップであり、前記第1ECDSA署名は、前記公開鍵に対応する秘密鍵に基づいてメッセージに署名し、前記メッセージは、前記第2トランザクションの一部である、ステップと、ECDSA検証機能を適用するステップであり、前記公開鍵と前記メッセージに基づいて、前記第2トランザクションで受信した前記第1ECDSA署名を検証し、前記コードは、前記第1ECDSA署名の前記検証のさらなる条件について真の結果を返すように構成されている、ステップと、を含む。前記コードは、さらに、前記第1ECDSA署名の前記r部に対応する参照値を含み、前記参照値は、前記r部の参照インスタンスまたはその変換である。この場合、前記コードは、前記参照値が前記第2トランザクションで受信したr部の前記参照インスタンスに対応することを検査し、かつ、さらなる条件で真の結果を返すように構成されている。

【0025】

第1トランザクションのコードに含まれるr部の指示は、r部の参照インスタンスであってよく、または、その変換、例えば、r部を構成するコンポーネントのハッシュであってもよい(ハッシュ化されたコンポーネントは、単にr部自体と等しいか、または、例えば、別のデータ要素dと連結することができる)。いずれにせよ、コードは、チャレンジャー(challenger)(例えば、ボブ)が一時的鍵kを知っていたに違いないことを証明する「rパズル(“r-puzzle”)」を設定する(ソリューションがkの知識なしで提供されていたことは不可能である)。本パズルは、この知識証明の基礎としてr部を有利に使用し、チャレンジャーにノードに対して一時的鍵kを提出することや、第2トランザクション(例えば、 T_{x_2})でそれを開示することを要求しない。これは、例えば、第1当事者(例えば、アリス)が、ボブやチャーリーなどの一時的な秘密鍵または秘密鍵として、署名権限を1つ以上の第2当事者に割り当てるために使用できることを意味する。rパズルは、kを明らかにすることなくkの知識を証明するので、悪意のあるマイナや他のノードは、意図された受益者の代わりに、自分自身に支払うであろう有効な第2トランザクション $T_{x_2}^*$ を形成するために、自分自身の署名を生成することができない。さらに、署名自体のr部は、システム内の身元にリンクされていないので、これは、kを知る誰もが証明を満たすことができることを意味する。第2トランザクションの有効性の条件は、第1トランザクションによって特定の身元と結び付けられる必要はない。例えば、アリスは、第1トランザクションをロックする相手を事前に決定しなくてよい。例えば、彼女は、第1トランザクションを設定するための

詳細を配達会社に提供し、その後に、誰が彼女に代わって荷物を受け取る権限を与えるかを決定することができる。

【0026】

より一般的に、本開示の範囲は、任意の「レイヤ2 (“layer2”)」のプルーフ・オブ・ワークにまで拡張することができ、それにより、トランザクションにおけるコードは、ブロックチェーンにおいてマイナによって既に実行されているプルーフ・オブ・ワークの上に追加的なプルーフ・オブ・ワークパズルを重ねる (layer) ために使用される。

【0027】

従って、ここにおいて開示される別の態様に従って、ブロックチェーンネットワークのノード検証において、コンピュータに実装される方法が提供される。本方法は、実行可能コードを含む第1トランザクションを取得するステップと、少なくとも第1部とノンスを有する情報を含む第2トランザクションを受信するステップと、第1トランザクションからのコードを実行するステップと、を含む。前記コードは、 $H_{Pow}(f(q, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、 q は第1部分であり、 d はノンスであり、 H_{Pow} はハッシュ関数であり、 f は r および d を組み合わせた関数である。

【図面の簡単な説明】

【0028】

本開示の実施形態の理解を助け、そのような実施形態がどのように実施され得るかを示すために、単なる実施例として、添付の図面が参照される。

【図1】図1は、ブロックチェーンを実装するためのシステムの概略ブロック図である。

【図2】図2は、ブロックチェーンに記録されるトランザクションのいくつかの例を概略的に示している。

【図3】図3は、ブロックチェーンを実装するためのシステムの別の概略ブロック図である。

【図4】図4は、出力ベースのモデルのノードプロトコルに従ったトランザクションを処理するためのノードソフトウェアの部分の概略ブロック図である。

【図5】図5は、トランザクションの例示的なセットを概略的に示したものである。

【図6A】図6Aは、楕円曲線デジタル署名アルゴリズム(ECDSA)の背後にある原理のいくつかを概略的に示している。

【図6B】図6Bは、楕円曲線デジタル署名アルゴリズム(ECDSA)の背後にある原理のいくつかを概略的に示している。

【図6C】図6Cは、楕円曲線デジタル署名アルゴリズム(ECDSA)の背後にある原理のいくつかを概略的に示している。

【図6D】図6Dは、楕円曲線デジタル署名アルゴリズム(ECDSA)の背後にある原理のいくつかを概略的に示している。

【図7】図7は、ここにおいて r パズル(または同義的に r チャレンジ)と称される一種の知識証明の可能な一つの実施例の概略図である。

【図8】図8は、 r パズルの別の可能な実装の概略図である。

【図9】図9は、 r パズルの別の可能な実装の概略図である。

【図10】図10は、 r パズルのさらに別の可能な実装の概略図である。

【図11】図11は、アカウントベースモデルのノードプロトコルに従って、トランザクションを処理するためのノードソフトウェアの概略ブロック図である。

【図12】図12は、ECDSA署名のための例示的なフォーマットを概略的に示している。

【図13】図13は、 r パズルの1つの形式に対するロック・アンド・ロッキングスクリプトの実装例のステップバイステップ・スクリプト分析である。

【図14A】図14Aは、基礎となるブロックチェーンプロトコルの一部として、マイナによって既に行われた固有のプルーフ・オブ・ワークの上に追加のプルーフ・オブ・ワークを課す r パズルの形態のいくつかの例を概略的に示す。

【図14B】図14Bは、基礎となるブロックチェーンプロトコルの一部として、マイナに

よって既に行われた固有のプルーフ・オブ・ワークの上に追加のプルーフ・オブ・ワークを課すrパズルの形態のいくつかの例を概略的に示す。

【図14C】図14Cは、基礎となるブロックチェーンプロトコルの一部として、マイナによって既に行われた固有のプルーフ・オブ・ワークの上に追加のプルーフ・オブ・ワークを課すrパズルの形態のいくつかの例を概略的に示す。

【図14D】図14Dは、基礎となるブロックチェーンプロトコルの一部として、マイナによって既に行われた固有のプルーフ・オブ・ワークの上に追加のプルーフ・オブ・ワークを課すrパズルの形態のいくつかの例を概略的に示す。

【発明を実施するための形態】

【0029】

ある暗号スキームにおいて、検証者 (verifier) は、ある人 (証明者 (prover) またはチャレンジャー (challengee) と呼ばれる) が知識証明 (knowledge proof) と呼ばれるものに何らかの情報を持っていることを確信させる必要があり得る。事実上、これは、情報の一部を直接的に検証者に提供することによって行うことができる。代替的に、情報に依存する計算を実行するために、証明者が要求されることもある。好ましくは、検証者自身がチャレンジ (challenge) を設定するために情報を知る必要がなく、また、証明者が情報を知っていることを検証するために情報を検証者に開示する必要もない計算である。計算方法については、入力データに対して検証計算を行わなければならない。秘密値の知識を証明する簡単な方法は、プリイメージ (preimage) と衝突耐性 (collision resistance) の機能のために暗号ハッシュ関数を使用することである。このハッシュ法は、ハッシュ関数がプライベート鍵公開鍵暗号システムの基本的な部分を形成するので、多くのブロックチェーンアプリケーションで容易に統合できる。この種の知識証明は、典型的にハッシュパズル (hash puzzle) と呼ばれるブロックチェーン応用において非常に多岐にわたるものである。

【0030】

UTXOベースのブロックチェーンでは、ハッシュパズル (ハッシュ値のプリイメージ) に対するソリューションを支払い条件として設定することができ、トランザクション検証の一部として検証がマイナによって実行される。しかしながら、このアプローチにおいて、トランザクションは、また、特定の秘密鍵を使用する署名を要求する必要がある。そうでなければ、マイナは、ブロック内にトランザクションを含める前に、ハッシュパズルのソリューションを受け取るからである。これは、悪意のあるマイナに、マイナに属する住所に向けられた出力を伴う支払いトランザクション (spending transaction) を作り出す機会を与える。

【0031】

本開示においては、この問題を回避する一方で、マイナ (または、転送ノード (forwarding node)) が検証を行うことを可能にする知識証明が提供される。これを行うために、知識プルーフが、楕円曲線デジタル署名アルゴリズム (ECDSA) 署名に対応する一時的鍵に接続されている。このアルゴリズムで使用される暗号プリミティブは、多くのブロックチェーンに固有であるため、現在のインフラストラクチャに容易に統合することができる。

【0032】

システム概要の例

図1は、ブロックチェーン150を実装するための例示的なシステム100を示す。システム100は、典型的にはインターネットのような広域インターネットワークであるパケット交換ネットワーク101を含む。パケット交換ネットワーク101は、パケット交換ネットワーク101内にピアツーピア (P2P) オーバーレイネットワーク106を形成するように配置された複数のノード104を含む。各ノード104は、異なるピアに属する異なるノード104を有するピアのコンピュータ装置を含む。各ノード104は、1つ以上のプロセッサ、例えば、1つ以上の中央処理装置 (CPU)、アクセラレータプロセッサ、特定用途向けプロセッサ、及び／又はフィールドプログラマブルゲートアレイ (FPGA) を含む処理装置、を含んでいる。各ノードは、また、メモリ、すなわち、非一時的コンピュータ読取可能媒体、

または媒体の形態でのコンピュータ読取可能記憶装置を備える。メモリは、1つ以上のメモリ媒体、例えば、ハードディスクなどの磁気媒体、ソリッドステートドライブ（SSD）、フラッシュメモリ、またはEEPROMなどの電子媒体、及び／又は、光ディスクドライブなどの光媒体を使用する1つ以上のメモリユニットを含んでよい。

【0033】

ブロックチェーン150は、データ151のブロックのチェーンを含み、ブロックチェーン150それぞれのコピーは、P2Pネットワーク160内の複数のノードそれぞれに維持される。チェーン内の各ブロック151は、1つ以上のトランザクション152を含み、ここで、このコンテキストにおけるトランザクションは、一種のデータ構造（data structure）を参照する。データ構造の性質は、トランザクションモデルまたはスキームの一部として使用されるトランザクションプロトコルのタイプに依存している。所与のブロックチェーンは、典型的に、全体を通して、1つの特定のトランザクションプロトコルを使用する。1つの一般的なタイプのトランザクションプロトコルにおいて、各トランザクション152のデータ構造は、少なくとも1つの入力および少なくとも1つの出力を含む。各出力は、ユーザ103に属するデジタル資産の量（quantity）を表す量（amount）を指定し、ユーザに対して出力は暗号的にロックされている（アンロックするためにはユーザの署名を必要とし、それによって、償還または支払いされる）。各入力は、先行トランザクション152の出力を指し示し、それによって、トランザクションをリンクする。

【0034】

ノード104の少なくとも一部は、トランザクション152を転送し、かつ、それによって伝搬する、転送ノード104Fの役割を担う。ノード104の少なくともいくつかは、マインブロック（mine block）151をマイニングするマイナ104Mの役割を担う。ノード104の少なくとも一部は、ストレージノード104Sの役割を担い、各ノードは、それぞれのメモリに同じブロックチェーン150のそれぞれのコピーを保管する。各マイナノード104Mは、また、ブロック151へのマイニングを待つトランザクション152のプール154を維持する。所与のノード104は、転送ノード104、マイナ104M、ストレージノード104S、または、これらの2つもしくは全ての任意の組み合わせであり得る。

【0035】

所与の現在トランザクション（present transaction）152jにおいて、入力（または、各入力）は、トランザクションのシーケンスにおいて先行トランザクション152iの出力を参照するポインタを含み、この出力が現在トランザクション152jにおいて償還されるか、または「支払われる（“spent”）」ことを指定する。一般的に、先行トランザクションは、プール154または任意のブロック151内の任意のトランザクションとすることができる。先行トランザクション152iは、必ずしも現在トランザクション152jが作成される時、または、ネットワーク106に送信される時に存在する必要はないが、先行トランザクション152iは、現在トランザクションが有効であるために存在し、かつ、検証される必要がある。従って、ここにおいては、「先行（“preceding”）」とは、ポインタによってリンクされた論理的な順序での先行を指し、必ずしも時間的な順序で作成または送信する時刻を指すのではなく、従って、トランザクション152i、152jが順序通りに作成または送信されることを必ずしも排除しない（孤立トランザクション（orphan transaction）について下記を参照のこと）。先行トランザクション152iは、同様に、先行（antecedent）トランザクションまたは先行（predecessor）トランザクションと呼ばれ得る。

【0036】

現在トランザクション152jの入力は、また、先行トランザクション152iの出力がロックされているユーザ103aの署名も含む。次に、現在トランザクション152jの出力は、新しいユーザ103bに対して暗号的にロックすることができる。従って、現在トランザクション152jは、先行トランザクション152iの入力で定義された量を、現在トランザクション152jの出力で定義された新しいユーザ103bに移転（transfer）することができる。ある場合に、トランザクション152は、複数のユーザ間で入力量を分割するために複数の出力を有してもよい（そのうちの1つは、変更を与えるために、元のユーザ103aであってよ

い)。場合によっては、トランザクションが、複数の入力を持ち、1つ以上の先行トランザクションの複数の出力から金額を一緒に集め、そして、現在トランザクションの1つ以上の出力に再配布することもできる。

【0037】

上記は、「出力ベース（“output-based”）」トランザクションプロトコルと呼ばれることがあり、ときどき、未使用トランザクション出力(UTXO)タイプのプロトコルとも呼ばれる(ここで、出力はUTXOと呼ばれている)。ユーザの総合収支（total balance）は、ブロックチェーン151に保管されたどの1つの番号にも定義されず、その代わりに、ユーザは、ブロックチェーン151内の多くの異なるトランザクション152に散在している、そのユーザの全てのUTXOの値を照合（collate）するために、特別な「ウォレット（“wallet”）」アプリケーション105を必要とする。

10

【0038】

代替的なタイプのトランザクションプロトコルは、アカウントベースのトランザクションモデルの一部として、「アカウントベース（“account-based”）」プロトコルと呼ばれ得る。アカウントベースの場合、各トランザクションは、過去のトランザクションのシーケンスで先行トランザクションのUTXOに戻すことによって、むしろ、絶対的なアカウント収支を参照することによって、移転される量（amount）を定義しない。全てのアカウントの現在の状態は、ブロックチェーンに分離されたマイナによって保管され、そして、絶えず更新される。このようなシステムにおいて、トランザクションは、アカウントの実行中トランザクション・タリー（tally）（「ポジション（“position”）」とも呼ばれる）を使用してオーダーされる。この値は、暗号署名の一部として送信者によって署名され、トランザクション参照計算の一部としてハッシュされる。さらに、オプションのデータフィールドも、また、署名されたトランザクションであり得る。このデータフィールドは、例えば、先行トランザクション（previous transaction）IDがデータフィールドに含まれている場合、先行トランザクションを示してよい。

20

【0039】

いずれのタイプのトランザクションプロトコルでも、ユーザ103が新しいトランザクション152jを制定（enact）することを望む場合、ユーザは、自分のコンピュータ端末102からP2Pネットワーク106のノード104の1つに新しいトランザクションを送信する(今日では、これは通常、サーバまたはデータセンタであるが、原則として他のユーザ端末でもよい)。このノード104は、各ノード104で適用されるノードプロトコルに従ってトランザクションが有効であるか否かをチェックする。ノードプロトコルの詳細は、全体的なトランザクションモデルを形成するとともに、問題のブロックチェーン150で使用されているトランザクションプロトコルのタイプに対応している。ノードプロトコルは、典型的に、新規トランザクション152jにおける暗号署名が、トランザクション152の順序付けされたシーケンスにおける先行トランザクション152iに依存する、期待される署名と一致することをチェックするために、ノード104を必要とする。出力ベースの場合に、これは、新しいトランザクション152jの入力に含まれるユーザの暗号署名が、新しいトランザクションが支払う先行トランザクション152iの出力に定義された条件と一致することをチェックすることを含み得る。ここで、この条件は、典型的に、新しいトランザクション152jの入力における暗号署名が、新しいトランザクションの入力が指し示す（point）先行トランザクション152iの出力をアンロックすることを、少なくともチェックすることを含む。いくつかのトランザクションプロトコルにおいて、条件は、少なくとも部分的に、入力及び／又は出力に含まれるカスタムスクリプトによって定義されてよい。代替的に、単にノードプロトコルだけで固定（fix）することもでき、代替的に、これらの組み合わせによることもある。いずれにせよ、新しいトランザクション152jが有効である場合、現在ノードは、P2Pネットワーク106内のノード104の1つ以上の他のノードにそれを転送（forward）する。これらのノード104の少なくとも一部は、同じノードプロトコルに従って同じテストを適用して、新しいトランザクション152jを1つ以上のさらなるノード104に転送ノード104Fとしても機能する。このようにして、新しいトランザクションは、ノード104のネ

30

40

50

ットワーク全体に伝搬される。

【0040】

出力ベースのモデルでは、所与の出力(例えば、UTXO)が支払われるか否かの定義は、ノードプロトコルに従った別の、前方 (onward) トランザクション152jの入力によって未だ有効に償還されているか否かである。トランザクションが有効であるための別の条件は、それが支払または償還を試みる先行トランザクション152iの出力が、別の有効なトランザクションによってまだ使用され/支払されていないことである。再度、有効でない場合、トランザクション152jは、ブロックチェーンに伝搬または記録されない。これは、同じトランザクションの出力を複数回支払おうとする二重支払いを防ぐ。一方、アカウントベースのモデルは、アカウントバランスを維持することによって、二重支払いを防ぐ。この場合も、トランザクションの順序が定義されているため、残高は、一度に単一の定義済み状態になる。

【0041】

検証に加えて、ノード104Mの少なくとも一部は、また、「プルーフ・オブ・ワーク」によって支えられている、マイニングとして知られるプロセスにおけるトランザクションのブロックを最初に作成するために競い合っている。マイニングノード104Mでは、まだブロックに現れていない有効なトランザクションのプールに、新しいトランザクションが追加される。次いで、マイナは、暗号パズルを解決しようとすることによって、トランザクションのプール154からトランザクション152の新しい有効なブロック151を組み立てようと競争する。典型的に、これは、「ノンス (nonce)」がトランザクションのプール154と連結され、ハッシュ化されると、ハッシュの出力が所定の条件を満たすように、ノンス値を探索 (search) することを含む。例えば、所定の条件は、ハッシュの出力が、既定の数の先頭ゼロを有することであってよい。ハッシュ関数の特性は、入力に関して予測不可能な出力を持つことである。従って、この探索は、ブルートフォースによってのみ実行することができ、従って、パズルを解決しようとしている各ノード104Mにおいて、相当量の処理リソースを消費する。

【0042】

パズルを解くための第1マイナノード104Mは、これをネットワーク106に通知し、そのソリューションを証明として提供し、これは、ネットワーク内の他のノード104によって容易にチェックすることができる(ソリューションがハッシュに与えられると、それがハッシュの出力を条件に合致させることをチェックすることは簡単である)。勝者 (winner) がパズルを解決したトランザクションプール154は、次に、そのような各ノードにおいて勝者が発表したソリューションをチェックしたことに基づいて、ストレージノード104Sとして作用するノード104の少なくとも一部によって、ブロックチェーン150内の新しいブロック151として記録される。ブロックポインタ155は、また、チェーン内の先に生成されたブロック151n-1に戻る新しいブロック151nにも割り当てられる。プルーフ・オブ・ワークは、新しいブロック151を作成するために多大な労力を要し、二重支払いを含むブロックは他のノード104によって拒否される可能性が高いので、二重支払いがそれらのブロックに含まれないようにするために、二重支払いのリスクを低減するのに役立つ。一旦生成されると、ブロック151は、同じプロトコルに従ってP2Pネットワーク106内のストレージノード104Sそれぞれで認識され、維持されるので、修正することができない。ブロックポインタ155は、また、ブロック151に逐次的な順序を課す。トランザクション152は、P2Pネットワーク106内の各ストレージノード104Sにおいて順序付けられたブロックに記録されるので、これは、トランザクションの不変の公開台帳 (public ledger) を提供する。

【0043】

パズルを解決するためにいつでも異なる104Mのマイナが、いつソリューションの探索を始めたかに応じて、いつでも、未マイニング (unmined) トランザクションプール154の異なるスナップショットに基づいて、そうすることができることに留意されたい。誰がそれぞれのパズルを解くかは、最初に、どのトランザクション152が次の新しいブロック

10

20

30

40

50

151nに含まれるかを定義し、現在の未マイニングトランザクションのプール154が更新される。次いで、マイナ104Mは、新たに定義された未決済（outstanding）プール154からブロックを生成するために競争を続ける。また、生じ得る「フォーク（“fork”）」を解決するためのプロトコルも存在する。これは、2人のマイナ104Mが互いの非常に短い時間内にパズルを解決し、ブロックチェーンの競合するビュー（conflicting view）が伝播するようにするものである。要するに、フォークの先（prong）のどちらが伸びても、最も長い方が最終的なブロックチェーン150となる。

【0044】

ほとんどのブロックチェーンにおいて、勝ったマイナ104Mは、（あるユーザから別のユーザにデジタル資産の量を移転する通常のトランザクションとは対照的に）どこにもない新しい量のデジタル資産を創出する特別な種類の新しいトランザクションによって自動的に報酬を受け取る。従って、勝ったノードは、ある量のデジタル資産を「マイニングした（“minded”）」したことになる。この特殊なタイプのトランザクションは、ときどき、「ジェネレーション（“generation”）」トランザクションと呼ばれる。それは、自動的に新しいブロック151nの一部を形成する。この報酬は、マイナ104Mがブルーフ・オブ・ワーク競争に参加するインセンティブを与える。通常の（非ジェネレーション）トランザクション152は、その出力の1つに追加のトランザクション料を指定し、そのトランザクションが含まれたブロック151nを生成した勝ったマイナ104Mに、しばしば、さらに報酬を与える。

【0045】

マイニングに含まれる計算リソースのため、典型的に、マイナノード104Mの少なくともそれぞれは、1つ以上の物理的サーバーユニット、またはデータセンタ全体を含むサーバの形態をとる。各転送ノード104M及び／又はストレージノード104Sは、サーバ又はデータセンタの形態をとることもできる。しかしながら、原則として、任意の与えられたノード104は、ユーザ端末または互いにネットワーク接続されたユーザ端末のグループの形態をとることができる。

【0046】

各ノード104のメモリは、それぞれの役割または複数の役割を実行し、ノードプロトコルに従ってトランザクション152を処理するために、ノード104の処理装置上で動作するように構成されたソフトウェア400を保管する。ここにおいてノード104に帰属されたいずれのアクションも、それぞれのコンピュータ装置の処理装置上で実行されるソフトウェア400によって実行され得ることが理解されよう。ノードソフトウェア400は、アプリケーション層、オペレーティングシステム層、プロトコル層などの下位層、または、これらの任意の組み合わせにおいて、1つ以上のアプリケーションで実装することができる。また、ここにおいて使用される用語「ブロックチェーン（“blockchain”）」は、一般的な技術の種類を指し、特定の専有ブロックチェーン、プロトコルまたはサービスに限定されない一般的な用語である。

【0047】

また、ネットワーク101には、ユーザを消費する役割の複数の当事者103それぞれのコンピュータ装置102も接続されている。これらは、トランザクションにおける支払人（payer）および受取人（payee）の役割を果たすが、他の当事者のためのマイニングまたは伝播トランザクションには必ずしも参加しない。それらは必ずしもマイニングプロトコルを実行するわけではない。2人の当事者103およびそれぞれの機器102は、説明のために示されており、第1当事者103aおよびそのそれぞれのコンピュータ装置102a、並びに、第2当事者103bおよびそのそれぞれのコンピュータ装置102bである。より多くのこのような当事者103およびそれらのそれぞれのコンピュータ装置102がシステムに存在し、参加することができるが、便宜上、それらは図示されていないことが理解されよう。各当事者103は、個人または組織であってもよい。純粋に例示として、第1当事者103aは、ここにおいてアリス（アリス）と称され、第2当事者103bは、ボブ（ボブ）と称されるが、これは限定的なものではなく、ここにおいてアリスまたはボブという言及は、それぞれ「第1当事者」および「第2当事者」と置き換えることができることが理解されるであろう。

【0048】

各当事者103のコンピュータ装置102は、1つ以上のプロセッサ、例えば、1つ以上のCPU、GPU、他のアクセラレータプロセッサ、特定用途向けプロセッサ、及び／又はFPGAを備えるそれぞれの処理装置を備える。各当事者103のコンピュータ装置102は、さらに、メモリ、すなわち、非一時的コンピュータ読取可能媒体または媒体の形態のコンピュータ読取可能記憶装置を備える。このメモリは、1つ以上のメモリ媒体、例えば、ハードディスクのような磁気媒体、SSD、フラッシュメモリまたはEEPROMのような電子媒体、及び／又は、光学ディスクドライブのような光学媒体を使用する1つ以上のメモリユニットを含むことができる。各当事者103のコンピュータ装置102上のメモリは、処理装置上で動作するように配置された少なくとも1つのクライアントアプリケーション105のそれぞれのインスタンスを含むソフトウェアを保管する。ここにおいて与えられた当事者103に帰属されたいずれのアクション（action）も、それぞれのコンピュータ装置102の処理装置上で実行されるソフトウェアを使用して実行され得ることが理解されよう。各当事者103のコンピュータ装置102は、少なくとも1つのユーザ端末、例えば、デスクトップまたはラップトップコンピュータ、タブレット、スマートフォン、またはスマートウォッチのようなウェアラブルデバイスを備えている。所与の当事者103のコンピュータ装置102は、ユーザ端末を介してアクセスされるクラウドコンピューティングリソースのような、1つ以上の他のネットワーク化されたリソースを含んでもよい。

10

【0049】

クライアントアプリケーション105は、最初に、サーバからダウンロードされた適切なコンピュータ読取可能な記憶媒体又は媒体上の任意の所与の当事者103のコンピュータ装置102に提供されてもよく、もしくは、リムーバブルSSD、フラッシュメモリー、リムーバブルEEPROM、リムーバブル磁気ディスクドライブ、磁気フロッピーディスク又はテープ、CD又はDVD ROMのような光ディスク、又はリムーバブル光学ドライブのようなリムーバブル記憶装置上に提供されてもよい。

20

【0050】

クライアントアプリケーション105は、少なくとも「ウォレット（“wallet”）」機能を備える。これには主に2つの機能がある。これらのうちの1つは、それぞれのユーザ当事者103が、ノード104のネットワーク全体にわたって伝搬され、それによってブロックチェーン150に含まれるトランザクション152を作成し、署名し、送信することを可能にすることである。もう1つは、現在所有しているデジタル資産の量（amount）をそれぞれの当事者に報告することである。出力ベースのシステムでは、この第2機能は、当該当事者に属するブロックチェーン150全体に散在する様々な152トランザクションの出力において定義されている量を照合することを含む。

30

【0051】

注：様々なクライアント機能は、与えられたクライアントアプリケーション105に統合されていると説明することができるが、これは必ずしも限定的なものではなく、代わりに、ここにおいて記載されている任意のクライアント機能は、例えば、APIを介したインターフェイス、または、一方が他方へのプラグインであるような、2つ以上の別個のアプリケーションのスイート（suite）で実装することができる。より一般的に、クライアント機能は、アプリケーション層またはオペレーティングシステムなどの下位層、または、これらの任意の組み合わせで実装することができる。以下は、クライアントアプリケーション105に関して説明されるが、これは限定的ではないことが理解されるであろう。

40

【0052】

各コンピュータ装置102上のクライアントアプリケーションまたはソフトウェア105のインスタンスは、P2Pネットワーク106の転送ノード104Fの少なくとも1つに動作可能に結合されている。これにより、クライアント105のウォレット機能は、トランザクション152をネットワーク106に送信することができる。クライアント105は、また、ストレージノード104のうちの1つ、一部、または全部にコンタクトして、それぞれの当事者103が受領者（recipient）である任意のトランザクションについてブロックチェーン150に

50

問い合わせる (query) ことができる (または、ブロックチェーン150の実施形態では、ブロックチェーン150は、部分的にその公衆の目 (public visibility) を通じてトランザクションの信頼を提供する公共施設 (public facility) であるため、実際には、ブロックチェーン150内の他の当事者のトランザクションを検査する)。各コンピュータ装置102上のウォレット機能は、トランザクションプロトコルに従ってトランザクション152を形成し、送信するように構成されている。各ノード104は、ノードプロトコルに従ってトランザクション152を検証するように構成されたソフトウェア400を実行し、転送ノード104Fの場合は、ネットワーク106全体にトランザクション152を伝播させるためにトランザクション152を転送する。トランザクションプロトコルとノードプロトコルは互に対応しており、所与のトランザクションプロトコルは所与のノードプロトコルと共に所与のトランザクションモデルを実装する。同じトランザクションプロトコルが、ブロックチェーン150内の全てのトランザクション152に使用されている (ただし、トランザクションプロトコルは、トランザクションの異なるサブタイプを許可してもよい)。同じノードプロトコルは、ネットワーク106内の全てのノード104によって使用される (ただし、多くのノードは、そのサブタイプに対して定義されたルールに従って異なるトランザクションのサブタイプを異なるように処理し、また、異なるノードは異なる役割を引き受け、従って、プロトコルの異なる対応する側面を実装することができる)。

【0053】

上述のように、ブロックチェーン150はブロック151のチェーンを含み、各ブロック151は、上述のように、プルーフ・オブ・ワークプロセスによって作成された1つ以上のトランザクション152jのセットを含む。各ブロック151は、また、ブロック151への逐次的順序を規定するように、チェーン内で先に生成されたブロック151に戻るブロックポイント155を含む。ブロックチェーン150は、また、プルーフ・オブ・ワークプロセスによって新しいブロックに含まれることを待つ有効なトランザクション154のプールを含む。各トランザクション152は、トランザクションの順序を定義するために、先行トランザクションへのポイントを含む (トランザクションの順序152は、分岐 (branch) が許されることに注意する)。ブロック151のチェーンは、チェーンの第1ブロックであった生成ブロック (genesis block、Gb) 153に戻る。チェーン150の初期に1つ以上のオリジナルトランザクション152は、先行トランザクションではなく生成ブロック153を指し示していた。

【0054】

所与の当事者103、例えばアリスが、ブロックチェーン150に含まれるように新しいトランザクション152jを送信することを望む場合、彼女は、関連するトランザクションプロトコルに従って (彼女のクライアントアプリケーション105においてウォレット機能を使用して) 新しいトランザクションを策定 (formulate) する。次に、クライアントアプリケーション105からトランザクション152を、彼女が接続されている1つ以上の転送ノード104Fの1つに送信する。例えば、これは、アリスのコンピュータ102に最も近いか、または最善に接続されている転送ノード104Fであってもよい。任意の与えられたノード104が新しいトランザクション152jを受信すると、ノードプロトコルおよびそのそれぞれの役割に従って、それを処理する。これは、最初に、新たに受信されたトランザクション152jが「有効 (“valid”)」であるための特定の条件を満たすか否かをチェックすることを含み、その例については、すぐに詳細に説明され。いくつかのトランザクションプロトコルにおいて、検証のための条件は、トランザクション152に含まれるスクリプトによってトランザクションごとに設定可能である。代替的に、条件は、単にノードプロトコルのビルトイン (built-in) 機能であってもよく、または、スクリプトとノードプロトコルの組み合わせによって定義されてもよい。

【0055】

新たに受信されたトランザクション152jが、有効であるとみなされるテストをパスするという条件で (すなわち、「有効である (“validated”)」という条件で)、トランザクション152jを受信する任意のストレージノード104Sは、そのノード104Sに維持されているブ

10

20

30

40

50

ックチェーン150のコピー内のプール154に、新たに有効とされたトランザクション152を追加する。さらに、トランザクション152jを受信する任意の転送ノード104Fは、検証されたトランザクション152をP2Pネットワーク106内の1つ以上の他のノード104に伝搬する。各転送ノード104Fは同じプロトコルを適用するので、トランザクション152jが有効であると仮定すると、これは、P2Pネットワーク106全体に間もなく伝搬されることを意味する。

【0056】

一旦、1つ以上のストレージノード104で維持されるブロックチェーン150のコピー内のプール154に入ると、マイナノード104Mは、新しいトランザクション152を含むプール154の最新バージョンのプルーフ・オブ・ワークパズルを解決するために競争（competing）を開始する（他のマイナ104Mは、依然として、プール154の古い見解に基づいてパズルを解決しようとしているが、そこに到達した者は誰でも、最初に、次の新しいブロック151が終了し、新しいプール154が開始する場所を定義し、最終的には、誰かが、アリスのトランザクション152jを含むプール154の一部のパズルを解決する）。一旦、新しいトランザクション152jを含むプール154についてプルーフ・オブ・ワークが行われると、それはブロックチェーン150内のブロック151のうちの1つの一部となる。各トランザクション152は、以前のトランザクションへのポインタから構成されるので、トランザクションの順序もまた、不変的に記録される。

【0057】

異なるノード104は、最初に、所与のトランザクションの異なるインスタンスを受信することができ、従って、1つのインスタンスがブロック150にマイニングされる前に、どのインスタンスが「有効」であるかの競合するビューを有し、その時点で、全てのノード104は、マイニングされたインスタンスが唯一の有効なインスタンスであることに同意する。ノード104が1つのインスタンスを有効として受け入れ、次いで、2つのインスタンスがブロックチェーン150に記録されていることを発見した場合、そのノード104は、これを受け入れなければならない、最初に受け入れた未マイニングインスタンスを破棄する（すなわち、無効（invalid）として扱う）。

【0058】

UTXOベースのモデル

図2は、トランザクションプロトコルの例を示している。これは、UTXOベースのプロトコルの一つの例である。トランザクション152（「Tx」と略す）は、ブロックチェーン150（各ブロック151は1つ以上のトランザクション152を含む）の基本的なデータ構造である。以下は、出力ベースのプロトコルまたは「UTXO」ベースのプロトコルを参照して説明されている。しかしながら、これは、全ての可能な実施形態に限定されるものではない。

【0059】

UTXOベースのモデルにおいて、各トランザクション（「Tx」）152は、1つ以上の入力202および1つ以上の出力203を含むデータ構造を備えている。各出力203は、未使用トランザクション出力（unspent transaction output、UTXO）を含んでもよく、これは、別の新しいトランザクションの入力202のソースとして使用することができる（UTXOがまだ償還されていない場合）。UTXOは、デジタル資産の量（価値のストア）を指定する。また、他の情報の中でも、それが来たトランザクションのトランザクションIDを含んでよい。トランザクションデータ構造は、また、入力フィールド202および出力フィールド203のサイズのインジケータを含み得る、ヘッダ201を含んでもよい。ヘッダ201は、また、トランザクションのIDを含んでもよい。実施形態において、トランザクションIDは、トランザクションデータのハッシュ（トランザクションID自体を除く）であり、マイナ104Mに提出された生（raw）トランザクション152のヘッダ201に保管される。

【0060】

アリス103aが、問題のデジタル資産の量をボブ103bに移転するトランザクション152jを作成したいと考えているとする。図2では、アリスの新しいトランザクション152jが「Tx₁」とラベル付けされている。これは、先行トランザクション152iの出力203において

アリスにロックされているデジタル資産の量を順番に取り、その少なくとも一部をボブに移転する。先行トランザクション152iは、図2では「Tx₀」とラベル付けされており、Tx₀およびTx₁は、単なる任意のラベルである。これらは、必ずしも、Tx₀がブロックチェーン151の第1トランザクションであること、または、Tx₁がプール154の直近の次のトランザクションであることを意味しない。Tx₁は、アリスにロックされた未使用（unspent）出力203を依然として有する、先行（すなわち、先立つ（antecedent））トランザクションのいずれかを指し示すことができる。

【0061】

先行トランザクションTx₀は、アリスがその新しいトランザクションTx₁を作成する時点、または、少なくとも彼女がそれをネットワーク106に送信する時点で、既に検証され、ブロックチェーン150に含まれていてよい。それは、その時点で既にブロック151のうちの1つに含まれていてもよく、代替的に、プール154内でまだ待機していてもよく、その場合、新しいブロック151にすぐに含まれることになる。代替的に、ノードプロトコルが「孤立（“orphan”）」トランザクションのバッファリングを許可する場合、Tx₀およびTx₁を作成して一緒にネットワーク102に送信することも、Tx₀をTx₁の後に送信することも可能である。ここにおいてトランザクションシーケンスのコンテキストで使用される「先行（“preceding”）」および「後続（“subsequent”）」という用語は、トランザクションで指定されたトランザクションポイントによって定義されるシーケンスにおけるトランザクションの順序を指す（トランザクションポイントが、他のどのトランザクションを指すか、等）。それらは、「前任者（“predecessor”）」と「後任者（“successor”）」、「前任者（“antecedent”）」と「後任者（“descendant”）」、「親（“parent”）」と「子供（“child”）」等と置き換えることもできる。これは、必ずしもそれらが生成され、ネットワーク106に送られ、または、任意の与えられたノード104に到着する順序を意味しない。それにもかかわらず、先行トランザクション（前任トランザクションまたは「親」）を指し示す後続トランザクション（後任トランザクションまたは「子」）は、親トランザクションが検証されるまで、かつ、親トランザクションが検証されない限り、検証されない。親がノード104に到着する前にノード104に到着した子は孤児とみなされる。ノードプロトコル及び／又はマイナの行動に応じて、それは、破棄され、または、親が待機するのを待つために一定時間バッファリングされることがある。

【0062】

先行トランザクションTx₀の1つ以上の出力203のうちの1つは、ここにおいてUTXO₀とラベル付けされた特定のUTXOを備える。各UTXOは、UTXOによって表されるデジタル資産の量を指定する値と、後続のトランザクションが検証されるため、従ってUTXOが正常に償還されるために、後続のトランザクションの入力202においてアンロッキングスクリプトによって満たされなければならない条件を定義するロッキングスクリプトを含む。典型的に、ロッキングスクリプトは、特定の当事者（それが含まれているトランザクションの受益者（beneficiary））に対する金額をロックする。すなわち、ロッキングスクリプトは、アンロッキング条件を定義し、典型的には、後続のトランザクションの入力におけるアンロッキングスクリプトが、先行トランザクションがロックされる相手の暗号署名を含む、という条件を含む。

【0063】

ロッキングスクリプト（別名scriptPubKey）は、ノードプロトコルによって認識されるドメイン固有の言語で書かれたコードの一部である。そのような言語の特定の例は、「スクリプト（“Script”）」（大文字のS）と呼ばれる。ロッキングスクリプトは、トランザクション出力203を支払うために必要な情報、例えば、アリスの署名の必要条件を指定する。トランザクションの出力には、アンロッキングスクリプトが表示される。アンロッキングスクリプト（別名scriptSig）は、ロッキングスクリプトの基準を満たすために必要な情報を提供するドメイン固有の言語で書かれたコードの一部である。例えば、ボブの署名を含んでもよい。アンロッキングスクリプトは、トランザクションの入力202に現れる。

【0064】

10

20

30

40

50

図示の例では、Tx₀の出力203におけるUTXO₀は、ロッキングスクリプト[Checksig P_A]を含み、これは、UTXO₀が償還されるために(厳密には、UTXO₀を償還しようとする後続のトランザクションが有効であるために)、アリスの署名Sig P_Aを必要とする。[Checksig P_A]は、アリスの公開鍵と秘密鍵のペアからの公開鍵P_Aを含む。Tx₁の入力202は、Tx₁を指し示すポインタを含む(例えば、そのトランザクションID、TxID₀、実施形態ではトランザクションTx₀全体のハッシュである)。Tx₁の入力202は、Tx₀の任意の他の可能な出力の中でそれを識別するために、Tx₀内のUTXO₀を識別するインデックスを含む。Tx₁の入力202は、さらに、鍵ペアからのアリスの秘密鍵をデータの所定の部分に適用することによって作成された、アリスの暗号署名を含むアンロッキングスクリプト〈Sig P_A〉を含む(ときどき、暗号では「メッセージ (“message”)」と呼ばれる)。有効な署名を提供するためにアリスが署名する必要があるデータ(または「メッセージ」)は、ロッキングスクリプト、ノードプロトコル、または、これらの組み合わせによって定義される。

【0065】

新しいトランザクションTx₁がノード104に到着すると、ノードは、ノードプロトコルを適用する。これは、ロッキングスクリプトとアンロッキングスクリプトと一緒に実行して、アンロッキングスクリプトがロッキングスクリプトで定義されている条件を満たしているか否かをチェックすることを含む(この条件は1つ以上の基準を含み得る)。実施形態において、これは、2つのスクリプトを連結することを含む。

〈Sig P_A〉 〈P_A〉 | | [Checksig P_A]

ここで、「| |」は連結を表し、「〈...〉」はデータをスタック上に置くことを意味し、「[...]」アンロッキングスクリプト(この例ではスタックベースの言語)で構成される関数である。同様に、スクリプトは、スクリプトを連結するのではなく、共通のスタックで互いに実行することができる。いずれにせよ、一緒に実行する場合、スクリプトは、Tx₀の出力のロッキングスクリプトに含まれるように、アリスの公開鍵P_Aを使用して、Tx₁の入力のロッキングスクリプトが、データの期待される部分に署名するアリスの署名を含むことを認証する。この認証を実行するためには、データ自体の期待される部分(「メッセージ」)もTx₀順序に含める必要がある。実施形態において、署名されたデータは、Tx₀の全体を含む(従って、別個の要素は、データの署名された部分が既に本質的に存在するので、クリアに指定することを含める必要がある)。

【0066】

公開-秘密 (public-private) 暗号による認証の詳細は、当業者にとって周知であろう。基本的に、アリスが秘密鍵を用いてメッセージを暗号化することによってメッセージに署名した場合、アリスの公開鍵とクリアのメッセージ(暗号化されていないメッセージ)が与えられると、ノード104のような別のエンティティは、そのメッセージの暗号化されたバージョンがアリスによって署名されていないことを認証することができる。署名は、典型的には、メッセージをハッシュし、ハッシュに署名し、署名としてメッセージのクリアなバージョンにこれをタグ付けすることによって、公開鍵の所有者が署名を認証することを可能にする。従って、ここで、データの特定の部分またはトランザクションの一部などに署名することへの言及は、実施形態において、そのデータの部分またはトランザクションの一部のハッシュに署名することを意味することができることに留意されたい。

【0067】

Tx₁のアンロッキングスクリプトが、Tx₀のロッキングスクリプトで指定された1つ以上の条件を満たす場合(示される例では、アリスの署名がTx₁で提供され、認証されている場合)、ノード104は、Tx₁が有効であるとみなす。それがストレージノード104Sである場合、これは、ブルーフ・オブ・ワークを待つトランザクションのプール154にそれを追加することを意味する。転送ノード104Fである場合、それは、トランザクションTx₁をネットワーク106内の1つ以上の他のノード104に転送し、その結果、それがネットワーク全体に伝搬されることになる。一旦、Tx₁が検証され、ブロックチェーン150に含まれると、これは、Tx₀からのUTXO₀を支払われたものとして定義する。Tx₁は、未使用トラン

10

20

30

40

50

ザクション出力203を使用する場合にのみ有効であることに留意されたい。別のトランザクション152によって既に支払われた出力を支払おうとする場合、 Tx_1 は、たとえ他の全ての条件が満たされていても無効となる。従って、ノード104は、先行トランザクション Tx_0 において参照されたUTXOが既に使用されているか否か(既に別の有効なトランザクションへの有効な入力を形成しているか否か)もチェックする必要がある。このことは、ブロックチェーン150がトランザクション152に定義された順序を課することが重要である理由の1つである。実際には、所与のノード104は、トランザクション152が支払われたUTXO 203の別個のデータベースのマーキングを維持することができるが、最終的には、UTXOが支払われたか否かを定義するのは、ブロックチェーン150内の別の有効なトランザクションへの有効な入力を既に形成しているか否かである。

10

【0068】

与えられたトランザクション152の全ての出力203で指定された総量が、その全ての入力202によって指摘される総量よりも大きい場合、これは、ほとんどのトランザクションモデルにおける無効性の別の根拠である。従って、このようなトランザクションは、ブロック151に伝搬されたり、マイニングされたりすることはない。

【0069】

UTXOベースのトランザクションモデルでは、所定のUTXOを全体として使用する必要があることに注意する。UTXOで定義されている量の一部を、別の量が支払われている一方で、支払われたものとして「残しておく(“leave behind”)」ことはできない。しかしながら、UTXOからの量は、次のトランザクションの複数の出力に分割できる。例えば、 Tx_0 においてUTXO₀で定義された量は、 Tx_1 の複数のUTXOに分割できる。従って、アリスがボブにUTXO₀で定義された量の全てを与えることを望まない場合、彼女は、残りの量を使って、 Tx_1 の第2出力に自分自身を変更し、または、別の当事者に支払うことができる。

20

【0070】

実際には、また、アリスは、たいてい、勝ったマイナの報酬(fee)を含める必要がある。なぜなら、今日では、ジェネレーショントランザクションの報酬だけでは、典型的に、マイニングを動機付けるのに十分ではないからである。アリスがマイナのための報酬を含まない場合、 Tx_0 は、マイナのノード104Mによって拒否される可能性が高く、従って、技術的には妥当であるが、それはまだ伝播されず、ブロックチェーン150に含まれる(マイナのプロトコルは、マイナ104Mが望まない場合には、トランザクション152を受け入れるよう強制しない)。一部のプロトコルでは、マイニング報酬は、独自の別個の出力203を必要としない(すなわち、別個のUTXOを必要としない)。その代わりに、入力202によって示される総量と所与のトランザクション152の出力203で指定される総量との間の差は、勝ったマイナ104に自動的に与えられる。例えば、UTXO₀へのポイントが Tx_1 への唯一の入力であり、かつ、 Tx_1 は1つの出力UTXO₁しか持っていないとする。UTXO₀で指定されたデジタル資産の量がUTXO₁で指定された量より多い場合、その差は、勝ちの(winning)マイナ104Mへ自動的に行く。しかしながら、代替的または追加的に、必ずしも、トランザクション152のUTXO 203のうち独自のものにおいて、マイナ報酬を明示的に指定できることは除外されない。

30

40

【0071】

アリスとボブのデジタル資産は、ブロックチェーン150内の任意のトランザクション152において、彼らに対してロックされている未使用UTXから成る。従って、典型的に、所与の当事者103の資産は、ブロックチェーン150を通して、様々なトランザクション152のUTXO全体に分散される。ブロックチェーン150内のどこにも、所与の当事者103の総残高を定義する1つの番号が保管されていない。各当事者にロックされ、別の前方トランザクションにまだ費やされていない全ての様々なUTXOの値をまとめることは、クライアントアプリケーション105におけるウォレット機能の役割である。これは、ストレージノード104Sのいずれかに保管されたブロックチェーン150のコピー、例えば、各当事者のコンピュータ装置102に最も近接しているか、または最も良好に接続されているストレージ

50

ノード104Sに問い合わせることによって行うことができる。

【0072】

スクリプトコードは、しばしば、概略的に表現される(すなわち、正確な言語ではない)ことに注意されたい。例えば、[Checksig P_A]=OP_DUP OP_HASH160 <H(P_A)> OP_EQUALVERIFY OP_CHECKSIGを意味するように[Checksig P_A]を書くかもしれない。「OP_...」は、スクリプト言語の特定のオペコードを指す。OP_CHECKSIG(「Checksig」とも呼ばれる)は、2つの入力(署名および公開鍵)を取り込み、楕円曲線デジタル署名アルゴリズム(ECDSA)を使用して署名の妥当性を検証するスクリプトオペコードである。ランタイムで、署名(「sig」)の発生は、全てスクリプトから削除されるが、ハッシュパズルなどの追加要件は、「sig」入力によって検証されるトランザクションに残る。別の例として、OP_RETURNは、トランザクション内にメタデータを保管することができ、それによって、メタデータをブロックチェーン150に不変に記録することができるトランザクションの不確定出力を生成するためのスクリプト言語のオペコードである。例えば、メタデータは、ブロックチェーンに保管することが望ましい文書を含むことができる。

【0073】

署名P_Aは、デジタル署名である。実施形態において、これは楕円曲線secp256k1を使用するECDSAに基づいている。デジタル署名は、特定のデータに署名する。実施形態では、所与のトランザクションについて、署名は、トランザクション入力の一部、および、トランザクション出力の全部または一部に署名する。署名する出力の特定の部分は、SIGHASHフラグに依存する。SIGHASHフラグは、署名の最後に含まれる4バイトのコードであり、どの出力が署名されるかを選択する(従って、署名の時点で固定される)。

【0074】

ロッキングスクリプトは、それぞれのトランザクションがロックされている相手の公開鍵を含んでいるという事実を参照して、ときどき、「scriptPubKey」と呼ばれる。アンロッキングスクリプトは、対応する署名を提供するという事実を参照して、ときどき、「scriptSig」と呼ばれる。しかしながら、より一般的には、ブロックチェーン150の全てのアプリケーションにおいて、償還されるUTXOに対する条件が、署名の認証を含むことは必須ではない。より一般的に、スクリプト言語は、任意の1つ以上の条件を定義するために使用され得る。従って、より一般的な用語である「ロッキングスクリプト」および「アンロッキングスクリプト」が望ましい。

【0075】

オプションのサイドチャネル

図3は、ブロックチェーン150を実装するためのさらなるシステム100を示す。システム100は、追加の通信機能が含まれることを除いて、図1に関連して説明したものと実質的に同じである。アリスおよびボブのコンピュータ装置102a、102bそれぞれの上のクライアントアプリケーションは、追加的な通信機能を備えている。すなわち、アリス103aは、(いずれかの当事者または第三者の勧誘で)ボブ103bと別個のサイドチャネル301を確立することを可能にする。サイドチャネル301は、P2Pネットワークとは別にデータの交換を可能にする。このような伝達は、ときどき、「オフチェーン(“off-chain”）」と呼ばれる。例えば、これは、当事者の一方がネットワーク106にそれをブロードキャストすることを選択するまで、ネットワークP2P 106上に公表されること(さえも)なく、または、チェーン150上でそのように進むことなく、アリスとボブとの間でトランザクション152を交換するために使用され得る。代替的または追加的に、サイドチャネル301は、キー、交渉された金額又は条件、データ内容等の他のトランザクション関連データを交換するために使用することができる。

【0076】

サイドチャネル301は、P2Pオーバーレイネットワーク106と同じパケット交換ネットワーク101を介して確立されてよい。代替的または追加的に、サイドチャネル301は、移動セルラネットワークのような異なるネットワーク、又はローカル無線ネットワークのようなローカルエリアネットワーク、又はアリスとボブの装置102a、102bとの間の直接的

な有線又は無線リンクを介して確立することができる。一般的に、ここにおける任意の箇所で言及されるサイドチャネル301は、データ「オフチェーン（“off-chain”）」、すなわち、P2Pオーバーレイネットワーク106とは別に交換するための1つ以上のネットワーク技術または通信媒体を介する任意の1つ以上のリンクを含んでもよい。2つ以上のリンクが使用される場合、全体としてのオフチェーンリンクのバンドルまたはコレクションは、サイドチャネル301と称されてよい。従って、アリスとボブがサイドチャネル301を介して特定の情報またはデータ、もしくは、そのようなものを交換すると言われる場合、これは必ずしもこれらのデータの全てが正確に同じリンクまたは同じタイプのネットワークを介して送信されなければならないことを意味しないことに注意されたい。

【0077】

ノードソフトウェア

図4は、UTXOまたは出力ベースのモデルの例において、P2Pネットワーク106の各ノード104上で実行されるノードソフトウェア400の例を示す。ノードソフトウェア400は、プロトコルエンジン401、スクリプトエンジン402、スタック403、アプリケーションレベル決定エンジン404、および、1つ以上のブロックチェーン関連機能モジュール405のセットを備える。任意の所与のノード104において、これらは、マイニングモジュール405M、転送モジュール405F、および、ストレージモジュール405Sのいずれか1つ、2つ、または全ての3つを(ノードの役割または役割に応じて)含んでもよい。プロトコルエンジン401は、トランザクション152の異なるフィールドを認識し、ノードプロトコルに従ってそれら进行处理するように構成されている。別の先行トランザクション152m-1 (Tx_{m-1}) の出力を指し示す入力を有するトランザクション152m (Tx_m) (例えば、UTXO) が受信されると、プロトコルエンジン401は、 Tx_m 内のアンロッキングスクリプトを識別し、それをスクリプトエンジン402に渡す。また、プロトコルエンジン401は、 Tx_m の入力内のポイントに基づいて Tx_{m-1} の識別および探索を行う。それは、 Tx_{m-1} が未だブロックチェーン150上に無い場合、ペンディングトランザクションの各ノードの自身のプール154から、または、 Tx_{m-1} が既にブロックチェーン150上にある場合には、各ノードに保管されたブロックチェーン150内のブロック151のコピーから、 Tx_{m-1} を探索することができる。いずれにせよ、スクリプトエンジン401は、 Tx_{m-1} の指し示された出力内のロッキングスクリプトを識別して、これをスクリプトエンジン402に渡す。

【0078】

従って、スクリプトエンジン402は、 Tx_{m-1} のロッキングスクリプト、および、対応する Tx_m の入力からのアンロッキングスクリプトを有する。例えば、 Tx_1 および Tx_2 が図4に示されているが、同様のことは、例えば、 Tx_0 および Tx_1 、等のような、任意のペア (pair) トランザクションにも当てはまり得る。スクリプトエンジン402は、前述のように、2つのスクリプトを一緒に実行する。それは、使用されるスタックベース (stack-based) のスクリプト言語(例えば、Script)に従って、データをスタック403に配置して、スタックからデータを取り出すことを含む。

【0079】

スクリプトを一緒に実行することによって、スクリプトエンジン402は、アンロッキングスクリプトがロッキングスクリプトに定義された1つ以上の基準を満たすか否かを判断する。すなわち、アンロッキングスクリプトが含まれている出力を「アンロック (“unlock”)」するか、である。スクリプトエンジン402は、この決定の結果をプロトコルエンジン401に戻す。スクリプトエンジン402は、アンロッキングスクリプトが対応するロッキングスクリプトに指定された1つ以上の基準を満たすと判断した場合、結果の「真 (“true”)」を返す。それ以外の場合は、結果の「偽 (“false”)」を返す。

【0080】

出力ベースのモデルでは、スクリプトエンジン402からの「真」の結果は、トランザクションの有効性の条件の1つである。典型的には、プロトコルエンジン401によって評価された1つ以上のさらなるプロトコルレベルの条件も同様に満たされなければならない。例えば、 Tx_m の出力に指定されたデジタル資産の総量が入力によって指し示された総量を超え

10

20

30

40

50

ないこと、および、 $T_{x_{m-1}}$ の指し示された出力が別の有効なトランザクションによって未だ費やされていないこと、といったものである。プロトコルエンジン401は、スクリプトエンジン402からの結果を、1つ以上のプロトコルレベル条件と共に評価し、それらが全て真である場合にのみ、トランザクション T_{x_m} を検証する。プロトコルエンジン401は、トランザクションがアプリケーションレベル決定エンジン404に対して有効であるか否かの表示を出力する。 T_{x_m} が実際に検証されているという条件のみで、決定エンジン404は、 T_{x_m} に関するそれぞれのブロックチェーン関連機能を実行するために、マイニングモジュール405Mおよび転送モジュール405Fの一方または両方を制御するように選択することができる。これは、ブロック151へのマイニングのためにノードのそれぞれのプール154に対して T_{x_m} を追加するマイニングモジュール405M、及び／又は、P2Pネットワーク106内の別のノード104に対して T_{x_m} を転送する転送モジュール405Fを含むことができる。しかしながら、実施形態において、決定エンジン404は、無効なトランザクションを転送またはマイニングすることを選択しないが、これは、逆に、それが単に有効であるという理由で、有効なトランザクションのマイニングまたは転送をトリガすることを必ずしも意味しないことに留意されたい。任意的に、実施形態において、決定エンジン404は、これらの機能のいずれか又は両方をトリガする前に、1つ以上の追加条件を適用することができる。例えば、ノードがマイニングノード104Mである場合、決定エンジンは、トランザクションが有効であり、かつ、十分なマイニング料を残すことを条件としてのみ、トランザクションをマイニングするように選択し得る。

【0081】

また、ここにおいて「真」および「偽」という用語は、必ずしも単一の2進数(ビット)の形式のみで表される結果を返すことに限定されないが、これは、確かに1つの可能な実装であることにも注意されたい。より一般的に、「真」は、成功または肯定的な結果を示す任意の状態を指し、「偽」は、失敗または否定的な結果を示す任意の状態を指す。例えば、アカウントベースのモデル(図4には示されていない)において、「真」の結果は、ノード104による署名の暗黙の、プロトコルレベルの検証、および、スマートコントラクトの追加の肯定的な出力の組み合わせによって示され得る(両方の個々の結果が真であれば、全体の結果は真であると見なされる)。

【0082】

トランザクションセットの例

図5は、ここにおいて開示される実施形態に従って使用するためのトランザクションのセット152を図示する。このセットは、第0トランザクション T_{x_0} 、第1トランザクション T_{x_1} 、および第2トランザクション T_{x_2} を含む。「第0」、「第1」、「第2」は、単に便利なラベルであることに注意する。これらのトランザクションは、必ずしもブロック151またはブロックチェーン150において次々に直ちに配置されること、または、第0トランザクションがブロック151またはブロックチェーン150における最初のトランザクションであることを意味しない。また、これらのラベルは、必然的に、これらのトランザクションがネットワーク106に送信される順序について何も意味しない。これらは、1つのトランザクションの出力が次のトランザクションの入力によって指し示される論理的手順のみを参照する。システムによっては、親を、その子供の後でネットワーク106に送信することが可能であることを覚えておくこと(その場合、「孤児」である子供は、親が到着するのを待っている間に、1つ以上のノード104で一定期間バッファリングされる)。

【0083】

第0トランザクション T_{x_0} は、また、アリス103aにロックされるデジタル資産の量のソースとして作用するという点で、本目的のためのソーストランザクションとも呼ばれる。第1トランザクション T_{x_1} は、本目的のためにチャレンジトランザクションまたはパズルトランザクションとも呼ばれ得る。これは、ソーストランザクション T_{x_0} からのデジタル資産の量を、第2トランザクション T_{x_2} に依存して条件付きで移転し、 r パズルに対するソリューションを提供する仲介者として機能する。第2トランザクション T_{x_2} は、第1トランザクション T_{x_1} で設定された r パズルに対するソリューションを提供し、その結果得られ

た支払いを証明者 (prover) (または、証明者が代行している受益者になる可能性がある者)にロックするトランザクションであるため、証明トランザクション、または、支払いトランザクションとも呼ばれる。実施形態は、証明者(第2当事者)がたまたまボブであるという例として説明することができるが、後の議論に基づいて理解されるように、rパズルは、rパズルを解決する有効な署名を提供する限り、アイデンティティにかかわらず、実際に任意の第2当事者が証明者であることを可能にする。

【0084】

図5に示すように、ソーストランザクションTx₀は、デジタル資産の量を指定する少なくとも1つの出力203₀(例えば、Tx₀の出力0)を含み、この出力をアリス103aにロックするロッキングスクリプトをさらに含む。これは、ソーストランザクションTx₀のロッキングスクリプトは、少なくとも1つの条件を満たす必要があることを意味する。それは、出力のアンロック(従って、デジタル資産の量の償還)を試みるトランザクションの入力は、そのアンロッキングスクリプトにアリスの暗号署名(すなわち、アリスの公開鍵を使用)を含まなければならないことである。この意味で、Tx₀の出力で定義される量は、アリスが所有していると言える。出力は、UTXOと呼ばれ得る。Tx₀の入力が(Tx₀の総出力をカバーするのに十分である限り)先行トランザクションの出力が戻って指し示す現在の目的について、特に重要ではない。

【0085】

この場合に、ソーストランザクションTx₀の出力をアンロックするトランザクションは、第1トランザクションTx₁(チャレンジトランザクション)である。従って、Tx₁は、少なくとも1つの入力202₁(例えば、Tx₁の入力0)を有し、入力1はTx₀の関連する出力(図示の例では、Tx₀の出力0)へのポインタを含み、かつ、さらに、少なくともアリスの署名を必要とする、その出力のロッキングスクリプトで定義される条件に従って、Tx₀の指し示された出力をアンロックするように構成されたアンロッキングスクリプトを含む。Tx₀のロッキングスクリプトによってアリスに要求される署名は、Tx₁の一部に署名するために必要である。プロトコルによっては、署名が必要なTx₁の部分が、Tx₁のアンロッキングスクリプトで定義された設定になる場合がある。例えば、署名に付加される1バイトであるSIGHASHフラグによってこれが設定され得るので、データに関して、アンロッキングスクリプトは、〈Sig P_A〉〈sighashflag〉〈P_A〉のように見える。代替的に、署名される必要のある部分は、単にTx₁の固定部分またはデフォルト部分であり得る。いずれにせよ、署名される部分は、典型的に、アンロッキングスクリプト自体を除外し、Tx₁の入力の一部または全てを除外し得る。しかしながら、Tx₁の署名された部分は、少なくともrパズルを含む出力203₁を含んでいる(この例におけるTx₁の出力0を、以下で参照のこと)。

【0086】

第1トランザクションTx₁は、少なくとも1つの出力203₁(例えば、Tx₁の出力0、これもまた、UTXOと称され得る)を有する。第1トランザクションTx₁の出力は、いずれか一方にロックされない。Tx₀と同様に、少なくとも1つの出力(例えば、Tx₁の出力0)を有し、これは、移転されるべきデジタル資産の量を指定し、さらに、その出力のアンロックし、従って、この量を償還するために必要なものを定義するロッキングスクリプトを含む。しかしながら、このロッキングスクリプトにより、rパズルに対してソリューションを提供する任意の当事者が、その出力をアンロックすることができる。

【0087】

第2トランザクション(支払いトランザクション)Tx₂は、少なくとも1つの入力202₂(例えば、Tx₂の入力0)を有しており、これは、Tx₁の上述の出力(図示の例では、Tx₁の出力0)へのポインタを含み、また、Tx₁のアンロッキングスクリプトで定義されたアンロック条件の1つ以上の要件を満たすことに基づいて、Tx₁の前記出力をアンロックするように構成されたアンロッキングスクリプトを含む。ここにおいて開示される実施形態によれば、アンロック条件は、対応するアンロッキングスクリプトがrパズルに対するソリューションを含むという少なくとも1つの要件を含む。rパズルは、楕円曲線暗号化(ECC)署名のr部(r-part)に基づいてTx₁のロッキングスクリプトで定義されたチャレンジ(challen

10

20

30

40

50

ge)を含み、これは、Tx₂のアンロックングスクリプトにおいてそれらの署名(または、少なくともそのs部(s-part))を含む、任意の当事者(この場合は偶然にボブである)によって満たされ得る。Tx₀のロックングスクリプトとは異なり、それがrチャレンジ(すなわち、rパズル)を満たす有効な署名である限り、どの当事者の署名もTx₁のロック状態をアンロックするために使用できることに注意する。この例について、より詳しく簡潔に説明する。ボブは、ここでは、単に、証明者または第2当事者の例として選ばれているが、実際には、rパズルは、第2当事者が証明者となることを可能にする。例えば、チャーリー、ドラ、エゼキエル、などである。いくつかの実施形態において、Tx₁におけるアンロック条件は、例えば、Tx₂のアンロックングスクリプトにもアリスの署名を含めることを必要とする、1つ以上のさらなる条件付きとすることもできる。

10

【0088】

第2トランザクションTx₂は、ボブに転送するデジタル資産の量を指定する少なくとも1つの出力2022(例えば、Tx₂の出力0)と、これをボブにロックするロックングスクリプトとを有する(すなわち、アンロックングスクリプト内のボブの署名を含む、さらに前方トランザクションを必要とする)。この意味で、ターゲットトランザクションTx₂の出力は、ボブによって所有されていると言える。この出力は、再びUTXOと呼ばれる。

【0089】

証明者の署名によって署名されたTx₂の部分(例えば、ボブの場合にはSig P_B)は、少なくともこの出力2032、すなわち、証明者への支払いをロックする出力(この例ではTx₂の出力0)を含むことになる。

20

【0090】

実施形態において、Tx₁の出力2031におけるロックングスクリプトは、出力をアンロックするための複数の代替条件、例えば複数の代替rパズルを定義することが可能である。この場合、Tx₂の入力2022のアンロックングスクリプトは、代替のアンロック条件のいずれかを満たす場合に、Tx₁の出力をアンロックする。

【0091】

第0(すなわち、ソース)トランザクションTx₀は、アリス、証明者r(例えば、ボブ)、または第三者によって生成されてよい。これは、典型的には、アリスが、Tx₀の入力で定義された量を取得した先行当事者の署名を必要とする。これは、アリス、ボブ、前述の当事者、または他の第三者によってネットワーク106に送信されてよい。

30

【0092】

第1トランザクション(すなわち、チャレンジトランザクション)Tx₁は、アリス、証明者(例えば、ボブ)または第三者によって生成されてよい。実施形態において、それはアリスの署名を必要とするので、それはアリスによって生成され得る。代替的に、ボブまたは第三者によってテンプレートとして生成され、サイドチャネル301を介して送信されるなど、署名するためにアリスに送信されてよい。次いで、アリスは、署名されたトランザクションをネットワーク106に自身で送るか、ボブまたは第三者に送ってそれらをネットワーク106に転送するか、または、ボブまたは第三者が署名されたTx₁へと組み立ててネットワーク106に転送するために彼女の署名を送信するだけでよい。ネットワーク106にTx₁を送信する前の任意のオフチェーン交換は、サイドチャネル301を介して行うことができる。

40

【0093】

第2トランザクション(すなわち、トランザクションの証明または支払い)Tx₂は、アリス、証明者(例えば、ボブ)、または第三者によって生成されてよい。最初のバージョンは、証明者の署名及び／又はデータを必要とするため、ボブによって生成され得る。代替的に、アリスまたは第三者によってテンプレートとして生成され、サイドチャネル301を介してボブに送信されるなど、ボブに署名するために送信される。次いで、ボブは、署名されたトランザクションを自身でネットワーク106に送信するか、または、それらをネットワーク106に転送するためにそれをアリスまたは第三者に送るか、または、単に彼の署名を送信して、アリスまたは第三者が署名されたTx₂へと組み立ててネットワークに転送す

50

ることができる。

【0094】

トランザクションの異なるエレメントを生成し、組み立てることができる様々な場所が存在し、P2Pネットワーク106の最終的な宛先に直接的または代理的に送信するための様々な方法があることが理解されよう。開示された技術の実施範囲は、これらの点のいずれにも限定されない。

【0095】

また、ここにおいて、「アリスによる(“by Alice”）」、「ボブによる(“by Bob”）」、および「第三者による(“by a third party”）」などのフレーズは、それぞれ「アリス103aのコンピュータ装置102aによる」、「ボブ103bのコンピュータ装置102bによる」、および「第三者のコンピュータ装置による」の略称(short-hand)として使用され得ることが理解されよう。また、特定の当事者の装置は、その当事者で使用される1つ以上のユーザ装置、または、その当事者で使用されるクラウドリソースなどのサーバリソース、これらの組み合わせから構成され得ることに再度注意すること。これは、必ずしも単一のユーザーデバイス上で実行されるアクションに限定されるわけではない。

【0096】

楕円曲線デジタル署名アルゴリズム(ECDSA)

公開鍵暗号は、多くの異なるブロックチェーンアーキテクチャにおけるトランザクションを安全にするための基礎として使用されている。公開鍵暗号の利用には、公開鍵暗号化とデジタル署名方式が含まれる。公開鍵暗号は、特定の関数は計算が容易であるが、特別な知識なしではリバースすることが困難であるという原則に基づいている。このような関数はトラップドア(trapdoor)関数と呼ばれ、リバースするために必要な特別な知識はその関数のトラップドアと呼ばれる。計算が容易であるということは、与えられた入力(または、入力のセット)について合理的な時間枠でトラップドア関数を計算することが計算上実行可能であることを意味し、トラップドアを知らずに結果からその入力(または、複数の入力)を推測することが計算上実行不可能であるということを反転するには困難である。

【0097】

公開鍵暗号のコンテキストにおいて、鍵ペアとは、公開鍵(誰でも自由に利用できるようにすることができる)、および、対応する秘密鍵(特定のエンティティまたはグループのみに知られているという意味で秘密であると仮定される)を意味する。公開鍵は、トラップドア機能を定義し、対応する秘密鍵は、その機能をリバースするために必要なトラップドアである。

【0098】

公開鍵暗号のコンテキストにおいて、暗号化は、トラップドア機能に基づいている(すなわち、暗号化は「順方向(“forward direction”）」で実行される)。一方、復号(decryption)は、トラップドア機能の逆転(reversal)に基づいている(すなわち、復号は「逆方向(“reverse direction”）」で実行される)。これは、トラップドアが既知である場合にのみ実行可能である。

【0099】

デジタル署名のコンテキストにおいて、署名検証は、公開鍵を使用して順方向で実行され、そして、署名生成は、逆方向に実行され、かつ、秘密鍵を使用してのみ実行可能である。

【0100】

ブロックチェーンのコンテキストでは、公開鍵暗号に基づくデジタル署名が、暗号技術的に、トランザクションに署名し、かつ、トランザクション署名を検証するための基礎として使用される。

【0101】

ECCは、楕円曲線の数学的特性を利用する公開鍵暗号の一形態であり、そして、DSA(Digital Secure Algorithm)のような他の暗号方式よりも様々な利点を有している。

【0102】

10

20

30

40

50

「楕円曲線デジタル署名アルゴリズム」(ECDSA)は、デジタル署名生成および検証のための基礎としてECCを使用するデジタル署名方式のクラスを指す。ECDSAのいくつかの原則を以下に概説する。

【0103】

数学的用語ではECCは、有限領域 (finite field) 上の素数秩序の楕円曲線の代数構造を利用する。有限領域とは、要素の有限集合と、集合内の要素に適用されたときに算術演算の通常の規則(連想性、可換性など)を満たす乗算、加算、減算、および除算の関連演算の集合を意味する。つまり、「通常 (“normal”）」の意味での加算や乗算などではなく、本質的には同じように動作する演算である。

【0104】

楕円曲線演算:

ECCのコンテキストにおいて、加算、減算、および乗算演算は、それぞれ、ここにおいて「+」と示される楕円曲線点加算、ここにおいて「-」と示される楕円曲線点減算、および、ここにおいて「・」と示される楕円曲線スカラー乗算である。加法及び減法演算は、それぞれ楕円曲線上の2つの点に適用され、楕円曲線上の3つ目の点を返す。しかしながら、乗法演算は、楕円曲線上のスカラー及び単一点に適用され、楕円曲線上の2つ目の点を返す。対照的に、分割はスカラーで定義される。

【0105】

説明のために、図6Aは、 R^2 における楕円曲線 ε を示しており、 R^2 は、全ての実数値の二次元座標の集合であり、そして、 $(x, y) \in R^2$ は、 R^2 の要素を示している。楕円曲線 ε は、以下の式を満たす点の集合である。

$$\varepsilon : y^2 = x^3 + ax + b$$

【0106】

加算 (addition): 楕円曲線 ε 上の任意の2点A、Bが与えられたとき、AとBが交差した線が ε を再交差し、1つの追加点Cのみが示される。AとBの楕円曲線の加算、すなわち $A + B$ は、Cの「反射 (“reflection”）」として定義される。Cに交差する水平線を取り、Cの反射は、その線と交差する楕円曲線上の他の点である。この定義は $A = B$ のケースにも当てはまる。修正は、Cが、Aで ε に対するタンジェントが再び ε と交差する点であることである。この定義は、 ∞ と示される、無限大での点を定義することにより交差する2点が垂直であるケースにも当てはまる。楕円曲線上の点であり、かつ、そこで任意の垂直線が楕円曲線と交差するからである(例えば、DおよびEとラベル付けされた点は垂直に水平に整列されているため、 $D + E = \infty$ である)。

【0107】

減算 (subtraction) / 加算の逆数 (additive inverse): 上記の反射の定義は、任意の点に適用され、楕円曲線の点減算 (point subtraction) の定義を提供する。すなわち、 $A - B$ は、Aと、Bの反射との合計である。Bの反射は、より正式にはBの「加法的逆数 (“additive inverse”）」と呼ばれ、 $-B$ 逆数で示される。この表記法を使用して、楕円曲線の減算は次のように数学的表記法で定義できる。

$$A - B = A + (-B)$$

【0108】

従って、図6Bでは、 $C = -(A + B)$ and $(A + B) = -C$ である。また、この定義の下では、 $D = -E$ であり、代数構造の一般的な法則、すなわち、楕円曲線上の任意の点とその加算的な逆数の楕円点の加算が、無限大の点であることにも注意する。すなわち、以下のとおりである。

$$A + (-A) = \infty \quad \forall A \in \varepsilon$$

してください。

【0109】

無限大 ∞ の点は、より形式的には「単位元 (“identity element”）」と呼ばれる(正規演算との並列 (parallel) および正規演算からの偏差 (deviation) の両方に注意する。正規演算では、任意の数 a と、その加法的な逆数 $-a$ との和は0であり、0は正規演算の

恒等要素である)。単位元 ∞ のもう一つの特性は、通常の算術演算を反映して、 ∞ それ自身を含む ε における任意の点Aについて、 $A + \infty = A$ であることである(任意の実数aについて $a + 0 = 0$ のステートメントに類似している)。

【0110】

乗算 (multiplication): 楕円曲線点加算の定義から、楕円曲線スカラ乗算の定義は次のようになる。楕円曲線点Aと整数vの乗算は以下のように定義される。

$$v \cdot A = A + \dots + A \quad (v \text{ 回の加算})$$

【0111】

つまり、vとして、楕円曲線の点は、Aそれ自体が加算されることになる。

【0112】

注: 楕円曲線スカラ乗算は、当技術分野では楕円曲線点乗算とも呼ばれる。これら2つの用語は、本開示において同じ意味を有する。

【0113】

除算 (Division) / 乗算の逆数 (multiplicative inverse): 除算の演算は、スカラーに関して定義される。スカラーvが与えられた場合、その「乗算の逆数」は、スカラー v^{-1} において以下のように定義される。

$$v v^{-1} = 1$$

【0114】

図6Aは、上述の演算の直感的な可視化を提供し、ここでは、全ての实数Rを含む無限領域の上で ε が定義される。

【0115】

図6Bは、ECCのコンテキストにおいて上記の演算が実際にどのように適用されるかをより詳細に示しており、式によって定義される ε_n を示す。

$$\varepsilon_n : y^2 = x^3 + ax + b \mod p$$

【0116】

ここで、pは素数(素数係数)であり、modはモジュロ演算を示す。上記の式を満足する点の集合は有限であり、これらの点の1つを除く全ての点が白色の円として図6Bに示され、残りの点は単位元 ∞ である。素数pは、楕円曲線の定義の一部を形成し、自由に選択することができる。楕円曲線が良好な暗号特性を持つためには、pは十分に大きくあるべきである。例えば、256ビットのpが、所定のブロックチェーンモデルで指定される。

【0117】

対照的に、下付き文字「n」は、ここにおいて、上記で定義される点加算の下で楕円曲線点によって形成されるグループの順序と呼ばれる(略して、これは楕円曲線 ε_n の順序と呼ぶことができる)ー以下を参照のこと。

【0118】

言い換えれば、nはグループの順序であり、そして、pは領域の順序である。全体としてn個の楕円曲線点が存在する。楕円曲線上の各点は、2つの数字/座標(x, y)で表され、ここで、xおよびyは、全て、 $-(p-1), \dots, 0, \dots, (p-1)$ の範囲内である。

【0119】

図6Bにおける ε_n は、図6Aの水平対称性に類似した水平対称性が示すことが分かる。これは、プライムファイル(prime files)に対する楕円曲線の一般的特性であり、従って、 ε_n 上の点の加算の逆数の定義は、依然として保持されることが分かる。いくつかの点は、水平に整列したカウンターポイント(例えば、(0, 0))を持たず、そのような点は、それら自身の加算の逆数である。

【0120】

ε_n 上で2つの点AおよびBと交差する「線(“line”)」 $l_{A,B}$ は、点の無限大セットになり、より小さな黒色の円で表され、類似の幾何学的要件を満たしており、かつ、楕円曲線スカラ乗算の定義が依然として成立する。図6Aと同様に、図6Bは、点 $A + B = -C$ を示しており、それは、線 $l_{A,B}$ が ε_n を再び交差する点 $C = -(A + B)$ の加算の逆数

10

20

30

40

50

である。

【0121】

ε_n 上の任意の2点の楕円曲線加算 $A + B = -C$ は、次の式で代数的に定義できる。

$$A = (x_A, y_A),$$

$$B = (x_B, y_B),$$

$$C = (x_C, y_C) = -(A + B),$$

$$x_C = (\lambda^2 - x_A - x_B) \bmod p,$$

$$y_C = ((\lambda(x_C - x_A) + y_A) \bmod p,$$

$$= ((\lambda(x_C - x_B) + y_B) \bmod p,$$

ここで、 $A \neq B$ の場合、

$$\lambda = (y_A - y_B)(x_A - x_B)^{-1} \bmod p$$

かつ、 $A = B$ の場合、

$$\lambda = (2y_A)^{-1}(3x_A^2 + a) \bmod p$$

【0122】

上記の目的のために、整数 v の乗算の逆数 v^{-1} の定義は、次のように修正される。

$$v v^{-1} \equiv 1 \pmod{p}$$

【0123】

つまり、整数 v の乗算の逆数は、 $v \bmod p$ のモジュラー逆数である。

【0124】

$B = -A$ である場合は特別なものであり、前述のように、単位元 ∞ の導入によって解決される。 $-$ 示されるように、この場合、 $A + B = A + (-A) = \infty$ である。 $B = \infty$ である場合も特別な場合であり、上記のように、 $A + \infty = A$ として解決される。

【0125】

楕円曲線スカラー乗算の定義は、楕円曲線加算のこの定義を採用し、それ以外は同じままである。

【0126】

他のコンテキストにおいて、スカラー v の乗算の逆数 v^{-1} の定義は次のようになる。

$$v v^{-1} \equiv 1 \pmod{n}$$

【0127】

コンテキストにおいて、 $\bmod n$ または $\bmod p$ いずれの乗法の逆数が定義されているのか明らかであろう。

【0128】

実際には、数を $\bmod n$ または $\bmod p$ いずれとして扱うべきかを識別するために、以下のチェックが適用される。

1. その数はECポイントの座標を表しているか？

a. はいの場合、 $\bmod p$ である。

2. その数はECポイントの乗算に使用されるか？

a. はいの場合、 $\bmod n$ である。

【0129】

ただし、両方のチェックが肯定的な回答を与える場合があり、その場合、数は、 $\bmod n$ および $\bmod p$ でなければならない。

【0130】

楕円曲線暗号(ECC)

楕円曲線演算は、秘密の値を不明瞭にするユニークな能力を提供し、多くの現代の暗号システムの基礎を形成している。特に、有限領域上の楕円曲線点の逆スカラー乗算は難しい問題である(計算的には実行不可能である)。

【0131】

秘密鍵 V は整数の形式をとり、対応する公開鍵 P は「生成点 (“generating point”)」 G から導出された楕円曲線 ε_n 上の点 P である。それは、また、以下のように、楕円曲線 ε_n 上の点である。

10

20

30

40

50

$$P = V \cdot G = G + \dots + G \quad (V \text{ 回の加算})$$

【0132】

ここで、「 $\cdot$ 」は、 a 、 b 、および n (楕円曲線パラメータ)によって定義される楕円曲線 E_n におけるスカラー乗算を示す。

【0133】

十分に大きい V について、実際に V の楕円曲線加算を行って P を導出することは困難であり、すなわち、計算的に実行不可能である。しかしながら、 V が既知であれば、楕円曲線演算の代数的性質を利用することによって、より効率的に P を計算することができる。 P の計算に使用できる効率的なアルゴリズムの例は、「double and add」アルゴリズムであり、—これは、既知の場合にのみ実装可能である。

10

【0134】

逆に、 V が知られていない場合は、たとえ G と P の両方が知られていても、計算的に可能な導出方法(すなわち、スカラー乗算を逆転すること)存在しない(これが、いわゆる「離散対数問題 (“discrete-logarithm problem”)」である)。攻撃者 (attacker) は、 G から開始して、 P まで楕円曲線点加算を繰り返し実行することによって、「ブルートフォース (“brute force”)」 P を試みることができる。その時点で彼は、 V が、彼が実行しなければならなかった楕円曲線点加算の数であることを知るだろう。しかし、それは計算上不可能であることが分かる。従って、 V は、上記の意味でトラップドアの要件を満たしている。

【0135】

20

ECCにおいて、公開鍵 P 、生成鍵 G 、および楕円曲線 E_n は公開されており、既知であると仮定されるが、一方で、秘密鍵 V は秘密である。

【0136】

楕円曲線デジタル署名検証アルゴリズム (ECDSA)

ブロックチェーンシステムにおいて、ユーザまたは他のエンティティは、典型的には、それらのIDを証明するために使用される秘密鍵 V を保持しており、対応する公開鍵 P は、次のように計算される。

$$P = V \cdot G$$

【0137】

秘密鍵 V は、ECDSAを使用して1つのデータ (a piece of data) m (「メッセージ」)に署名することができる。

30

【0138】

ECDSAのさらなる詳細は、例えば、以下に記載されており、ここにおいて、その全体が、参照により、組み込まれている。“RFC 6979-Deterministic Usage of the Digital Signature Algorithm(DSA) and Elliptic Curve Digital Signature Algorithm(ECDSA)” Tools.ietf.orgである。

【0139】

図6Cは、署名生成機能の概略機能ブロック図を示している(署名生成器_600は、公開鍵／秘密鍵ペア (V , P) についてECDSA署名 (r , s) を生成する)。EDSA署名は値のペアであり、ここにおいては、それぞれ r 部 (r) および s 部 (s) と呼ばれている。

40

【0140】

署名生成は、公開鍵 P を導出するために使用される、同じ楕円曲線 E_n および発生点 G に基づいており、従って、楕円曲線パラメータ a 、 b 、および n 、並びに、発生点 G は、署名生成器600への入力として示されている。

【0141】

署名生成器600の一時的鍵生成器602は、「一時的 (“ephemeral”)」鍵 $k \in [1, n-1]$ を生成する。すなわち、1から $n-1$ までの包括的な範囲である。

【0142】

r 部生成器604は、以下のようにして対応する公開一時的鍵 k を計算する。

$$R = k \cdot G$$

50

次いで、計算した点のx座標をとる($[]_x$ は、楕円曲線点のx座標をとるプロセスを示している)。

$$r = [R]$$

これは、署名のr部である。

【0143】

s部生成器606は、 $k \bmod n$ のモジュラー逆数 k^{-1} （すなわち、結果として $k^{-1}k \equiv 1 \pmod{n}$ ）であり、—上記を参照のこと）、および、 $H(m)$ と示される(必要に応じて切り捨てられる)、メッセージmのハッシュを使用して、署名のs部を以下のように計算する。

$$s = k^{-1} (H(m) + rV) \bmod n$$

10

【0144】

本例において、メッセージmは、トランザクション608に含まれるべきデータを含む(本例では、1つ以上のトランザクション出力)。これは、メッセージmに署名するプロセスと呼ばれてよく、そして、メッセージmは、トランザクションの署名部分と呼ばれてよい。

【0145】

メッセージmおよび署名(r, s)は、順番に、トランザクション608の一部を形成する。本例において、署名(r, s)は、アンロックングスクリプトの一部としてトランザクション608の入力に含まれる。

【0146】

図6Dは、トランザクション608を検証するための署名検証機能(署名検証器)620の概略機能ブロック図を示す。署名検証器620によって実行される計算は、上記のように、公開されている、同じ楕円曲線 E_n および生成点Gに基づいている。

20

【0147】

署名は、入力として秘密鍵Vを必要とする。すなわち、有効な署名を生成するために、署名の知識を必要とするが、署名のペア(r, s)、メッセージm、および公開鍵Pのみが、署名(r, s)を検証するために必要とされる。署名を検証するために、署名検証器620は、トランザクションmの署名部分をハッシュする(署名(r, s)を生成するために使用されるのと同じハッシュ関数Hを適用する)。次に、以下の計算を使用して検証プロセスを実行する。

$$R' = H(m) s^{-1} \cdot G + r s^{-1} \cdot P$$

30

【0148】

署名は、 $[R']_x = r$ が場合にのみ、有効であり(すなわち、署名検証が成功する)、それ以外の場合は、無効である(すなわち署名検証が失敗する)。本例において、rは、トランザクション608に含まれる署名のr部を示す。

【0149】

署名検証プロセスで使用される公開鍵Pは、例えば、先行トランザクションのロックングスクリプトで指定することができる。この場合、署名検証は、先行トランザクションのロックングスクリプトに指定された公開鍵、および、(後の)トランザクション608の署名(r, s)を使用して実行される。—そして、署名(r, s)が、先行トランザクションで指定された公開鍵に対応する秘密鍵Pおよび後のトランザクション608の署名部mに基づいて生成されていない場合には、失敗する。従って、秘密鍵Vを保持する者のみが、先行トランザクションの出力を主張することができ(典型的には、後のトランザクション608の出力に自身の公開鍵を含めることによる)、後のトランザクション608の署名部mは、署名(r, s)を無効にすることなく変更することはできない。

40

【0150】

Rパズル(R-PUZZLE)

以下は、ECDSAに基づく知識証明の新しい形態について説明する。例示として、チャレンジャーは、第1トランザクションTx₁においてrパズルを設定する第1当事者のアリスである。P2Pブロックチェーンネットワーク106を自ら作成し、公開することによるか、または、それらをTx₁へと組み立てるために第三者に必要な詳細を提供することによるか、

50

いずれかである。検証器(実際にプルーフを実行する当事者)は、ネットワークのノード104のオペレータ、例えばマイナである。rパズルに対するソリューションは、ネットワーク106にTx₂を公開することによって提供される。rパズルは本質的にアイデンティティと結びついていないので、証明者は任意の第2当事者であり得るが、例として、下記は、証明者が偶然ボブであるシナリオに関して記述され得る。証明者は、Tx₂を自ら作成し、公表するか、または、第三者がTx₂へと組み立て、公表するために必要な詳細を提供することができる。

【0151】

暗号ハッシュ関数は、入力の小さな変化が出力の予測不可能な変化を導く場合に、入力を実定的に不明瞭にする手段を提供する。従来のハッシュ関数には、MD5、RIPEMD-160、SHA-1、およびSHA-256[5]があり、それぞれが衝突耐性 (collision resistance) (同じ出力を生成する2つの入力を見つける可能性が非常に小さい)とプリメージ抵抗を提供する(ハッシュ値 $h = H(d)$ が与えられると、入力 d をを見つけるのは非常に難しい)。

【0152】

従来のハッシュパズルは、以下のように設定することができる。本アイデアは、第1トランザクションTx₁を設定し、第2トランザクションTx₂がその入力に特定の1つのデータを含むことを条件に、その出力を第2トランザクションTx₂によって償還することを可能にすることである。

【0153】

ブロックチェーントランザクションにおいて、第1当事者(アリス)は、ロッキングスクリプト内のハッシュ値 h を使用して、非標準トランザクションTx₂を、次のように、単純に作成できる。

OP_HASH160 <h> OP_EQUALVERIFY

【0154】

ここで、 $h = H_{puz}(d)$ であり、かつ、 H_{puz} はパズルで使用されるハッシュ関数である(上記の例では、ロッキングスクリプトに従って、このハッシュ関数はHASH160であることを要するが、他の実装では、別の形式のハッシュ関数を使用することができる)。このロッキングスクリプトが含まれているUTXOを償還するには、後続のトランザクションのアンロッキングスクリプトにおけるハッシュパズルソリューションが必要になる。従って、アドレスAddr_Bobを有する第2当事者のための支払いトランザクションTx₂は、 d だけを含む必要のあるアンロッキングスクリプトを用いて構築されるだろう。

【表1】

$TxID_2$	
入力	出力
0. TxID1 Unlocking script: <d>	0. Address: Addr_Bob Amount: {VALUE}

【0155】

ここで、TxID_iは、Tx_iのトランザクションIDである。ロッキングスクリプトは次のように記述する。データ値 d をTx₂の入力におけるアンロッキングスクリプトから取り出し、それをハッシュし、そして、Tx₁の出力におけるロッキングスクリプトに含まれるハッシュ値 h と等しいか否かを確認する。従って、Tx₂のアンロッキングスクリプトに d を提供することによって、出力はアンロックされる。

【0156】

この単純な例では、Tx₂におけるハッシュパズルソリューションを有するユーザのトランザクションを見た後で、このトランザクションを最初に受け取ったマイナは、悪意を持ってトランザクションを拒否し、ハッシュパズルに対する同じソリューションを用いて新しいマリート (malleated) バージョンのTx₂^{*}を作成し、出力を自分のアドレスAddr_Minerに変更することができる。その後、悪意のあるマイナは自分自身でブロック151へTx₂^{*}をマイニングしようとすることができ、そして、もしTx₂がマイニングされる前に成功すれば、そのマイナはボブの代わりに支払いを受けることになる。

【表 2】

TxID ₂ [*]	
入力	出力
1. TxID1 Unlocking script: <d>	1. Address: Addr_Miner Amount: {VALUE}

【0157】

デジタル署名は、所有権を証明し、そして、未使用トランザクション出力(UTXO)を償還するためにブロックチェーントランザクションにおいて一般的に使用されている。これにより、特定の当事者にロックされるTx₁のようなトランザクションの出力が可能になる。最も一般的な例は、pay-to-public-key-hash(P2PKH)トランザクションであり、ここで、トランザクションの出力は、公開鍵の特定のハッシュに対してロックされる(その当事者のアドレスとしても、また、機能する)。公開鍵Pのロックングスクリプトは、次のとおりである。

OP_DUP OP_HASH160 <h_p> OP_EQUALVERIFY OP_CHECKSIG

【0158】

ここで、h_p=H_{sig}(P)であり、かつ、H_{sig}は、署名で使用されるハッシュ関数である(上記の例では、ロックングスクリプトに従って、このハッシュ関数はHASH160であることを要するが、他の実装では、別の形式のハッシュ関数を使用することができる)。別のトランザクションへの入力として、このUTXOを使用できるようにするには、Pをする有効なECDSA署名を有するアンロックングスクリプトを提供する必要がある。

<sig> <P>

【0159】

ストリング全体(アンロックング+ロックングスクリプト)は、マイナによって評価され、それは、正しい公開鍵が提供されていること、および、署名が有効であり、かつ、Pに対応することを確認する。ロックングスクリプトは、基本的に、Tx₂の入力におけるアンロックングスクリプトから公開鍵Pを取り出し、それをハッシュし、そして、Tx₁の出力におけるロックングスクリプトに含まれるハッシュ値h_pと同じか否かをチェックする。そして、また、Tx₂の署名部分の知識が与えられて、ECDSA検証機能に基づいて、Tx₂のアンロックングスクリプトから公開鍵Pを使用して署名sigを検証する。ECDSA検証機能は、OP_CHECKSIGオペコードによって呼び出される。

【0160】

従って、出力は、Pに対応する秘密鍵Vに基づいて署名された有効な署名sigをアンロックングスクリプトにおいて、提供することによってのみ、アンロックすることができる。

【0161】

これをハッシュパズルと一緒にすると、上述の脆弱性 (vulnerability) は、ハッシュパズルソリューションと共に、意図された受信者からのデジタル署名を要求することによ

って修正することができる。ロッキングスクリプトは以下のように構成される。

```
OP_HASH160 <h> OP_EQUALVERIFY OP_DUP OP_HASH160 <hp> OP_EQUALVERIFY OP_CHECKSIG
```

そして、対応するアンロッキングスクリプトは、以下であることを要するだろう。

```
<sig> <P> <d>
```

【0162】

しかしながら、これは、誰がそれを償還できるかを、公開鍵Pの所有者に制限する。ここにおいて、いくつかのアプリケーションでは、これが望ましくないことが認められている。例えば、アリスがパズルをセットアップした後にのみ、署名権限を指定する能力を保持することを望む場合である。

【0163】

ここにおいて、ハッシュパズル機能は、一時的なランダム値であり得る、ECDSA署名におけるr部を利用することによってエミュレートすることができることが認識される。ECDSA署名は、2つの主要部分、rおよびs、から構成されている。上記のように、 $r = [k \cdot G]_x$ である。従来のハッシュパズル $h = H(d)$ の代わりに、楕円曲線加算の反転の難処理性 (intractability) は、ここにおいてrパズル (r-puzzle) と呼ばれる類似のパズルを形成することができる。このパズルを解くためには、ソリューション値kを得る必要がある。ここでkは、rに対応している一時的鍵であるが。

【0164】

従来のハッシュパズルにおいて、リスクは、パズルを解くときにブロックチェーン上でdを明らかにすることである。しかしながら、rパズルで、kは決して明らかにならない。その代わりに、rが明らかにされ、そして、署名と共にrから、kの知識を証明することができる。

【0165】

ハッシュパズル機能をエミュレートするために、rパズルの作成者は、まず、値kを得るために、いくつかの他のプリイメージデータをハッシュ化することができる。なぜなら、kは固定サイズでなければならない、一方で、ハッシュパズルのプリイメージデータは、任意の長さであってよいからである(そして、ハッシュ関数の1つの特性は、入力データの長さに関係なく、固定長の値を出力することである)。例えば、256ビット長の秘密／一時的鍵を使用する場合、kを取得するためにrパズルに対するプリイメージデータがハッシュされるべきである。代替的に、しかしながら、kのいくつかの適切な長さの値を選択し、そして、直接的にそれ自身の権利における秘密値として使用することができる(すなわち、いくつかの他の、先行するプリイメージから導出する必要はない)。

【0166】

この方法は、支払いにECDSA署名を使用する任意のブロックチェーンシステムで使うことができる。例示として、以下にUTXOベースのモデルにおける例示的な実装を説明する。スクリプト言語において、OP_CHECKSIGオペコードは、スタック上の署名と公開鍵を必要とする(スタックの上部に公開鍵があり、そして、その直ぐ下に署名がある)。rパズルの場合、スクリプトは、提供された署名内の値がrパズルチャレンジに使用されたr値と同じであることをチェックするように構成されている。言い換えると、スクリプトは、署名が公開鍵において有効であることを(OP_CHECKSIGを通して)チェックするだけでなく、あらかじめブロックチェーン上で公表されるrパズルのr値を使用して署名が生成されることを確認する。

【0167】

これから、図7～図10を参照して、rパズルのいくつかの例示的な実装を説明する。それぞれの場合において、証明者、例えばボブは、Tx₂の署名の一部に署名することによって署名(r,s)を作成した。この書式の署名は、ときどき、“sig”と呼ばれる。暗号署名のコンテキストにおいて、署名された部分は、また、“message”(m)とも呼ばれる。署名された部分(メッセージ)mは、少なくともTx₂の出力203₂を含み、その出力は、結果としてボブへの支払いをロックする。1つ以上の出力が存在する場合、mは、出力の一部または全て

10

20

30

40

50

を構成することができる。mは、また、ロックタイムといった他の部分も含めることができる。しかしながら、典型的には、アンロックングスクリプト自体を除外する(もちろん、少なくとも署名自体を除外しなければならない)。メッセージmとして署名されるTx₂の部分は、Sighashによって設定されるか、またはデフォルトであるか、またはプロトコルの固定機能であり得る。

【0168】

おそらく、rパズルの最も簡単な実装が図7に示されている。Tx₁におけるロックングスクリプトは、ここにおいてr'とラベル付けされる、参照インスタンス(reference instance)またはr部を含んでいる。この方法では、Tx₂におけるロックングスクリプトは、ボブの署名の少なくともs部を含むことだけを必要とする。また、ボブがm署名するために使用した秘密鍵Vに対応する公開鍵Pを含む見得る。Tx₁のロックングスクリプトは、ノード104でスクリプトエンジン402が実行する場合、Tx₂のアンロックングスクリプトからsおよびPを取り出し、そして、以下のオペレーションを実行するように構成されている。

I) $R' = H_{sig}(m) s^{-1} \cdot G + r' s^{-1} \cdot P$ 、かつ、

II) $check[R']_x = r'$

【0169】

ここで、r'はTx₁のロックングスクリプトから取られ、そして、sおよびmはTx₂のアンロックングスクリプトから取られる。ボブの公開鍵Pは、Tx₂のアンロックングスクリプトから取ることもできるし、または、他の手段によって知ることでもある。H_{sig}はハッシュ関数であり、最初のECDSA署名を生成する際にハッシュmに使用される。それは任意の形式のハッシュ関数であり得る。どのような形式であれ、このハッシュ関数の形式(タイプ)は、両端で事前に決められ(predetermined)、かつ、既知であると仮定され得る。Gは、固定値であり、公知のベクトル値である。

【0170】

ロックングスクリプトは、上記のチェックが真であるという条件で「真」の結果を返すが、そうでなければ、「偽」の結果を返すように設定されている。UTX0の場合、アンロックングスクリプトと一緒にロックングを実行すると真の(すなわち成功した)結果は、トランザクションの有効性に対する要件である。従って、Tx₂の妥当性は、rパズルの結果についてプロキシ(proxy)として使用することができる。または別の言い方では、Tx₂の妥当性はrパズルにソリューションを与えることを条件とする。すなわち、ボブがrパズルをパスしない場合、彼のトランザクションTx₂はネットワーク106を介して伝搬されず、ブロックチェーン150内に記録されない(そして、Tx₁の出力で定義された任意の支払いは、償還されない)。

【0171】

図7の例は、数学的な意味では最も単純であるが、このことは、任意の所与のノードプロトコルまたはスクリプト言語と統合することが最も単純であることを必ずしも意味しない。もし支払者(spender)が、〈r,s〉および〈P〉とは対照的に、アンロックングスクリプトにおける〈s〉と〈P〉だけを提供する場合、スクリプトは、これを説明(account)する必要がある。オペレーションI)~II)は、標準Checksigtタイプのオペコードのオペレーションではない。OP_CHECKSIGのオペコード(op-code)は署名がDER形式であることを期待しているので、〈s〉値のみがアンロックングスクリプトで提供されている場合、DER形式で有効な署名を生成するためにはロックングスクリプトに追加のオペコードが必要となる(OP_CATを連結する、など)。図8は、簡潔に記載されており、数学的には余分のステップを含んでいるが、実際にはスクリプトのようなスクリプト言語とより簡単に統合する代替的な例を示しており、Tx₂の入力から取り出されるrおよびsの両方に基づいてECDSA署名検証を呼び出すための専用のオペコードを既に有している。

【0172】

注記:全ての可能な実施形態においてTx₂にPを含めることは必須ではない。実際、メッセージmおよび(r,s)の知識から、または、この場合は(r',s)、公開鍵2つの可能な値Pおよび-Pを計算することが可能である(ただし、どちらがどちらであるかは分からない)。

2つの検証が、次いで、どちらが正しいものであるかを識別するために使用され、または、代替的に、1ビットのフラグが、2つの可能なソリューションのどちらを使用するかを信号化するためにTx₂で使用され得る。この後者のアプローチは、現在、いくつかのアカウントベースのプロトコルで使用されている。しかしながら、(r,s)からPと-P、および、mを掲載するための演算のオペコードをスクリプト言語(例えば、Script)が持たない現在のUTXOベースのプロトコルでは使用されない傾向がある。それにもかかわらず、1つが導入され得ること、または、オペレーションがロッキングスクリプトに単に明示的にコード化され得る可能性を除外すべきではない。別の可能性は、アリスが、Pへのアクセスを既に知っているか、または有しているか、もしくは、サイドチャネル301を介して受け受信していることである。しかしながら、これは、PをTx₂にマップするための別のルックアップを必要とするだろう。

10

【0173】

別の例示的な実装を図8に示す。ここで、rパズルは、Tx₂のアンロッキングスクリプトが明示的にr-部の提出(submit)されたインスタンスrを含むことを要求する。Tx₁のロッキングスクリプトは、r部に対するテストを含み、そのテストは、提出されたインスタンスrに対して比較されるr部の参照インスタンスr'を含む。この方法において、Tx₂のアンロッキングスクリプトは、少なくともボブの署名のr部(r)およびs部(s)を含む必要がある。また、ボブがm署名のために使用した秘密鍵Vに対応する公開鍵Pも含み得る。Tx₁のロッキングスクリプトは、ノード104でスクリプトエンジン402が実行する場合、Tx₂のアンロッキングスクリプトからr、s、およびPを取り出し、そして、以下のオペレーションを実行するように構成されている。

20

I) check $r' = r$

II) compute $R' = H_{\text{sig}}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ 、かつ、

III) check $[R']_x = r$

【0174】

ここで、r'はTx₁のロッキングスクリプトから取られ、そして、s、r、およびmはTx₂のアンロッキングスクリプトから取られる。ボブの公開鍵Pは、Tx₂のアンロッキングスクリプトから取ることもできるし、または、他の手段によって知ることできる。前述したように、(r,s)およびm、または、(r,s)およびmから派生することによる、といったものである。

30

【0175】

ロッキングスクリプトは、ステップI)とステップIII)の両方のチェックが真であるという条件で「真」の結果を返すが、そうでなければ、「偽」の結果を返すように設定されている。再び、UTXOベースのケースでは、これにより、rパズルの知識ブルーフの結果に依存して、トランザクションの妥当性を決定することができる。数字I~IIIは必ずしも順序を意味しないことに注意されたい。I)~III)の後にチェックI)を実行することができるが、III)は、II)の後に実行されることを要する。

【0176】

図8の方法において、ステップI)およびIII)のみが、ECDSA検証機能によって実行される従来のオペレーションである。従って、ほとんどのプロトコルでは、スクリプト内の既存のChecksigtオペコード(OP_CHECKSIG)のような専用のオペコードによって呼び出すことができる。ステップI)は、汎用のオペコードを使用してロッキングスクリプトへと別々にコード化することができる(例が簡単に示される)。また、Checksigtのような専用のオペコードを使用する代わりに、ステップII)およびIII)が原則として汎用オペコードを使用して明示的に符号化(encoded)できることも除外されない。

40

【0177】

トランザクションプロトコルの一つの例において、トランザクションECDSA署名は、図12に示すように、ASN.1(抽象構文記法1 (Abstract Syntax Notation One)) DER(識別符号化規則 (Distinguished Encoding Rules))符号化フォーマットを使用している。最初のバイトフィールドは、ASN.1シーケンス番号を示すフラグ0x30を含む。次のバ

50

イトフィールドは、16進数のシーケンスの長さを含む。3番目のバイトフィールドは、ASN.1整数を示すフラグ0x02を含む。その後、ECDSA署名のr値は、次の32バイトまたは33バイトに含まれる。フィールドは32バイトであるが、rの最初のバイトが0x7fより大きい場合(最初のビットが1)、r値の前に0の加算バイトが追加され、33バイト長となる。これは、整数の最初のビットを符号として解釈するDERフォーマットエンコーディングの結果として行われる。ゼロの余分のバイトは、負の値として解釈されないように値の先頭に追加されている。ECDSA署名のs値についても同じことが行われる。最後に、1バイトフィールド、ハッシュタイプ(ht)が、トランザクションのビットコイン署名のタイプ(SIGHASH_ALL、SIGHASH_NONE、など)に対応するDERエンコーディングに追加される。

【0178】

アリス(A)が、パズルに対するソリューションを得た人なら誰でも支払できるrパズルトランザクションを作成したい場合を考えてみる。これを達成するために、彼女は、以下のような新しいトランザクションTx₁を作成する。入力セクションには、支払いされる以前トランザクションTx₀のアンロッキングスクリプトが含まれている。簡単にするために、アリスの署名と公開鍵を使って支払われる標準的なP2PKHであると仮定する。出力セクションには、ロッキングスクリプト(スクリプトパブキー (script pub key))、または、別の言葉で言えば、rパズルチャレンジが含まれる。図12に示されているように、署名はいくつかのプロトコルでDER符号化形式を使うかもしれないので、スクリプトは符号化された署名からrの値を抽出し、次いで、それが〈r〉と等しいことをチェックする必要がある。その後、スクリプトは、署名が公開鍵上で有効であることをチェックする必要がある。スクリプトがどのように働くか詳細が図5に示されている。太字のオペコードは、基本的には、署名からrを抽出する方法である。

【表3】

TxID1	
入力	出力
任意の 支払い入力	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP 〈r〉 OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

【0179】

対応するアンロッキングスクリプトを以下に示す。ここで、署名sig_rがrを使用し、かつ、支払者 (spender) ボブ(B)が任意の秘密鍵／公開鍵ペアを使用して署名を計算できる。sig_rは、(r,s)であることに留意する。

〈P_B〉 〈sig_r〉

【0180】

図13は、ステップ毎 (step-by-step) のスクリプト分析を示している。

【0181】

一時的鍵kは、アリスによって生成され、そして、ボブ(および、任意的に、1人以上の他の潜在的な証明者)に与えられてよい。代替的に、kは、ボブによって生成され、そして、アリスに与えられて、ボブだけ(または、ボブがkを共有することを選択した任意の者)が解決できるrパズルを設定することができる。どちらの場合でも、証明者ボブは、アリスがrパズルに対するソリューション (k)を知っているのを、送信者アリスが自分自身でトランザクションを支払わない (not to spend) ことを信頼しなければならない。これを防ぐために、証明者ボブはパズルを作成し、次いで、Rパズルトランザクションを作成するときに使用するために、r値をアリスに送信し得る。その後、ボブは、値kを保持する限り、秘密鍵／公開鍵ペアを使用して、後日、出力を償還することができる。これは、rパズルに対するソリューションであり、そして、鍵の一形態と見なすことができる。一方で、いくつかのケースでは、アリスがkを知っているという事実が有利な特徴であり得る。例えば、これは、秘密鍵パズルを作成し、それを通して、一般化されたアトミックスワッ

プ(atomic swap)を作成するために使用することができる。

【0182】

図9は、別の例示的なrパズルを示しており、pay to public hash (P2PKH)に類似して、ここにおいては“pay to r-puzzle hash”(P2RPH)と呼ぶことができる。セキュリティとプライバシーを向上させるために、r値は、Tx₁(ネットワーク106のノード104を通して伝搬され、ブロックチェーン150において配置されるもの)内に置かれる前にハッシュされ得る。公開鍵自体の代わりに公開鍵のハッシュのみがブロックチェーン上にある、P2PKHと同様に、Rパズルでも同じことができる。

【0183】

ここにおいて、rパズルは、再び、Tx₂のアンロックングスクリプトがr部の提出されたインスタンスrを含むことを要求する。再度、Tx₁のロックングスクリプトは、r部のテストを含むが、今回は、r'のハッシュの形式におけるr部の圧縮インスタンスの形式である。すなわち、 $h = H(r')$ 。これは、提出されたインスタンスrと、再び、比較される。この方法では、Tx₂におけるアンロックングスクリプトは、また、少なくともボブの署名のr部(r)およびs部(s)を含む必要がある。また、ボブがm署名するために使用した秘密鍵Vに対応する公開鍵Pも含み得る。Tx₁のロックングスクリプトは、ノード104でスクリプトエンジン402が実行する場合、Tx₂のアンロックングスクリプトからr、s、およびPを取り出し、そして、以下のオペレーションを実行するように構成されている。

I) check $h = H_{puz}(r)$ 、かつ、

II) compute $R' = H_{sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ 、かつ、

III) check $[R']_x = r$

【0184】

ここで、hはTx₁のロックングスクリプトから取られ、そして、s、r、およびmはTx₂のアンロックングスクリプトから取られる。ハッシュ値 $h = H_{puz}(r)$ であり、ここで、 H_{puz} は、rパズルのハッシュで使用されるハッシュ関数である。それは、ハッシュ関数の任意の形式である可能性がある。これは、 H_{sig} と同じか、または、異なる形式であり得る。どのような形態をとるにせよ、 H_{puz} の形態は、両端で事前に決められ、かつ、既知であると仮定され得る。ボブの公開鍵Pは、また、アンロックングスクリプトTx₂から取ることもできるし、または、他の手段によって知ることできる。前述したように、(r,s)およびm、または、(r,s)およびmから派生することによる、といったものである。

【0185】

ロックングスクリプトは、ステップI)とステップIII)の両方のチェックが真であるという条件で「真」の結果を返すが、そうでなければ、「偽」の結果を返すように設定されている。II)~III)の前または後にチェックI)を実行することができるが、III)は、II)の後に実行されることを要する。

【0186】

また、図8の場合と同様に、ステップII)およびIII)のみが、ECDSA検証機能によって実行される従来のオペレーションである。従って、ほとんどのプロトコルでは、スクリプト内の既存のChecksigsオペコード(OP_CHECKSIG)のような専用のオペコードによって呼び出すことができる。ステップI)は、汎用のオペコードを使用してロックングスクリプトへと別々にコード化することができる。

【0187】

トランザクションチャレンジTx₁におけるロックングスクリプトの例が以下に示されている。

10

20

30

40

50

【表 4】

TxID1	
入力	出力
任意の 支払い入力	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT OP_DROP OP_HASH160 $\langle h \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

【0188】

任意のタイプのハッシュ関数が使用され、両方の当時者である、送信者と受信者との間で一貫したものであり得る。しかしながら、P2PKH標準に一致したままで、我々は、OP_HASH160、SHA-256のダブルハッシュ、そして、次いでRIPEMD-160を使用する。

【0189】

対応するアンロッキングスクリプトを以下に示す(以前のセクションと同じである)。ここでは、署名 sig_r は r を使用し、そして、支払者ボブ(B)は任意の秘密鍵／公開鍵ペアを使用して署名を計算できる。

$\langle P_B \rangle \langle \text{sig}_r \rangle$

【0190】

図9の例は、従って、非変換インスタンス r の代わりに、 r チャレンジの基礎として r 部のハッシュを使用する点を除けば、図8とちょうど同じある。

【0191】

これらの場合のいずれにおいても、 Tx_1 のアンロッキングスクリプトが「真の」結果について追加の基準を課し得ることが除外されないことに注意されたい。例えば、ロックタイムまたは追加署名の要求は一つの例であろう。

【0192】

上記の技術のいずれかの使用例は、一般知識チャレンジとしてのものである。あるソリューション k を有する任意のチャレンジ、または、 k にハッシュされ得る、あるソリューションを検討する。アリスは、次いで、パズルに結合されたRパズルを作成することができる。すなわち、彼女は $r = [k \cdot G]_x$ を定義することができる。

【0193】

一つの例として、アリスは数学の教授である。彼女は r パズルトランザクション Tx_1 を構築することができる。ここで、根底にある(underlying) k 値は、学生が解くように動機付けされる数学的な質問に対するソリューションである。ソリューションを作成する人は誰でも、そのソリューションを使って署名(r, s)を作成することができる。ここで、 r はアンロッキングスクリプトにおける値と一致するので、報酬を要求する。署名は、真正性を提供するだけでなく、他の誰にもソリューションを明らかにすることなく、ソリューションの知識証明となる。従って、Rパズルは、あるソリューションまたは情報一般の知識を、それを露出するリスクなしに証明する安全なメカニズムを提供する。それは、スクリプトのアンロッキングにおいて必要な署名を上品に(elegantly)再利用し、そして、ソリューションを見つけた人は誰でも、任意の公開鍵 P が使用され得るので、プライバシーと共に報酬を請求することができる。

【0194】

このスキームは、また、トークンまたはデジタルチケットの形式としても使用できる。例えば、イベントの主催者は、出席者に対して、デジタルチケットとして異なる価値 k を発行することができる。出席者がイベントに参加したい場合、 r パズルの使用を通じて秘密トークン(secret token)の知識を証明することができる。

【0195】

別の例示的なユースケースとして、 r パズルは署名者認可スキーム(signatory authorisation scheme)として使用することができ、ここで、一方の当事者は、他方の当事者に署名する権利を委任することができる。アンロッキングスクリプトに一致する r 値を伴う署

10

20

30

40

50

名が提供されている場合にのみアンロックされ得る r パズル トランザクション Tx_1 を検討する。これは、1 人だけが値 k を知っており、 $r = [k \cdot G]_x$ が、そうした署名を生成できることを意味する。しかしながら、その人が k の知識を誰かに伝えれば、これは、事実上、彼または彼女に代わって他の人に署名する権限を与えることになる。

【0196】

例えば、アリスが配達 (delivery) を受けたいと仮定する。彼女は、配達を受け入れるためにそこにいないかもしれないことを心配している。彼女は、ボブとチャーリー (チャーリー) の両方に k のコピーを渡し、そして、彼らが彼女に代わって配達を受けることができる。デイブ (デイブ) が小包を配達する場合、彼女は、ボブに小包をリリースするために期待される r 値を伴う署名を得る必要がある。

【0197】

このようなシナリオにおいて、 k は一時的な秘密鍵として、そして、 r は一時的な公開鍵として機能すると考えることができる。 k および r が特定のアイデンティティにリンクされていないことを除いて、 V および P にそれぞれ類似している。

【0198】

結合値 (joint value) R パズル

図9のハッシュ R パズル ($P2RPH$) の一つの拡張として、ハッシュ化前に r と連結された追加値 d を含めることが可能である ($h = H_{puz}(r || d)$ を取得するため)。その場合、証明者 (例えば、ボブ) は、 r パズルを解くだけでなく、 r も知っていなければならない。この例示的な実装を図10に示す。

【0199】

Tx_1 のロッキングスクリプトは、ノード104でスクリプトエンジン402が実行する場合、 Tx_2 のアンロックスクリプトから r 、 s 、 P 、および d を取り出し、そして、以下のオペレーションを実行するように構成されている。

- I) $check\ h_{joint} = H_{puz}(r || d)$ 、かつ、
- II) $compute\ R' = H_{sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ 、かつ、
- III) $check\ [R']_x = r$

【0200】

ここで、 $r || d$ は、いずれかの順序 (r が最初または d が最初) での r および d の連結を表している。チャレンジ トランザクション Tx_1 におけるロッキングスクリプトの例を以下に示す。

【表5】

$TxID$	
入力	出力
任意の 支払い入力	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT
	OP_DROP_2 OP_ROLL OP_CAT OP_HASH160 $\langle h_{joint} \rangle$
	OP_EQUALVERIFY
	OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG

【0201】

対応するアンロックスクリプトを以下に示す (d を含む場合を除き、以前のセクションと同じである)。署名 $sig_r P_B$ は r を使用し、そして、証明者ボブ (B) は任意の秘密鍵 / 公開鍵ペアを使用して署名を計算することができる。

$\langle sig' \rangle \langle P_B \rangle \langle d \rangle \langle sig_r \rangle$

【0202】

余分の署名 sig' は、セキュリティのために追加された機能である (後述するオプションの

10

20

30

40

50

セキュリティ機能についてのセクションを参照のこと)。しかしながら、このことは、全ての可能な実施形態において要求される必要はない。

【0203】

例示的なユースケースは、CLTVリンクRパズルである。この場合、データ値 d はCLTV(Check Lock Time Verify)トランザクション出力にリンクされている時間値 t であり得る。この背後にある動機は、P2RPHハッシュ内の前に出力が支払えない時間 t を隠し、それをRパズルにリンクすることである。その場合、証明者(例えばボブ)は、 r パズルを解くだけでなく、時間 t を知り、それを支払う特定の時間まで待たなければならない。トランザクションにおけるロッキングスクリプトの例を以下に示す。

【表6】

$TxID1$	
入力	出力
任意の 支払い入力	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT OP_DROP OP_2 OP_ROLL OP_CHECKLOCKTIMEVERIFY OP_CAT OP_HASH160 $\langle h_{joint} \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_CHECKSIGSIG

【0204】

対応するアンロッキングスクリプトを以下に示す。ここで、署名 sig_{rP_B} は r を使用し、そして、支払者ボブ(B)は任意の秘密鍵／公開鍵ペアを使用して署名を計算することができる。

$\langle \text{sig}' \rangle \langle P_B \rangle \langle t \rangle \langle \text{sig}_r \rangle$

【0205】

余分の署名 sig' は、セキュリティのために追加された機能である(後述するオプションのセキュリティ機能についてのセクションを参照のこと)。しかしながら、このことは、全ての可能な実施形態において要求される必要はない。

【0206】

上記は、連結に関して説明されてきた。しかしながら、これをある関数 $f(r, d)$ に一般化することも可能である。例えば、 $\langle r \rangle \langle d \rangle \text{OP_ADD}$ として実装されるように、 f は、 r および d の加法であり得る。

【0207】

Rパズルのプルーフ・オブ・ワーク

上記の結合値パズルの変形として、証明トランザクション Tx_2 で受け取った値 d は、PoW(proof-of-work)パズルの形式でノンス値として使用され得る。支払者は、 r パズルを解くだけでなく、所定の困難性であるターゲット(target)のハッシュを結果として生じるノンス値を見つけるための作業も行う必要がある。言い換えれば、以下の方程式についてスクリプトにおけるチェック、または、ターゲットより多くの先頭ゼロを伴う値である。

$$H_{puz}(r || d) < target$$

【0208】

注記:一般的に、所与のハッシュ関数は、単一の基本ハッシュ関数か、または、2つ以上の構成要素(constituent)ハッシュ関数の合成であってよい。例えば、実施形態において、 $H_{puz} = H[H(\dots)]$ である。一般的なブロックチェーンマイニングプロトコルにおいて、トランザクションヘッダは、プルーフ・オブ・ワークにおいて2回ハッシュされる。従って、これを模倣するために、ここにおいて開示された r パズルPoWの実施形態において、 $r || d$ は、また、2回ハッシュされ得る。しかしながら、これは任意的であり、そして、他の実施形態において、 H_{puz} は、ただの単一ハッシュであってよい。

【0209】

好ましくは、このチェックは、プルーフ・オブ・ワークで使用された r 値が署名に対応す

ることを検証するために、Tx₂において受信された署名タプル(r,s)に対して実行される少なくとも規則的なECDSA署名検証と組み合わせられる。ロックングスクリプトおよびアンロックングスクリプトは、一緒に実行されると、両方のチェックが満たされた場合にのみ「真」の結果を出力する。つまり、rと一緒にノンスがターゲットを満たし、そして、署名が検証される。

【0210】

この場合、Tx₁のロックングスクリプトは、ノード104でスクリプトエンジン402が実行する場合、Tx₂のアンロックングスクリプトからr、s、P、およびナンスを取り出し、そして、以下のオペレーションを実行するように構成されている。

check $H_{Pow}(r || d) \leq target$ 、かつ、
compute $R' = H_{sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ 、かつ、
check $[R']_x = r$

10

【0211】

ここで、 $r || d$ は、いずれかの順序(rが最初またはdが最初)でのrおよびdの連結を表しており、そして、dはナンスである。これは、図14Aに示されている。一般的に、 H_{Pow} はハッシュ関数と同じか、または、 H_{sig} 及び／又は H_{Pow} に対する異なる形式のハッシュ関数であってよい。

【0212】

トランザクションTx₁における対応するロックングスクリプトを以下に示す。

【表7】

20

TxID	
入力	出力
任意の 支払い入力	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT
	OP_DROP OP_2 OP_ROLL OP_CAT OP_SHA256 OP_SHA256 <target>
	OP_LESSTHAN OP_VERIFY OP_CHECKSIGVERIFY OP_CHECKSIGSIG

30

【0213】

対応するアンロックングスクリプトを以下に示す(ノンスとデータ値dを除き、以前のセクションと同じである)。ここで、署名 $sig_r P_B$ はrを使用し、そして、支払者ボブ(B)は任意の秘密鍵／公開鍵ペアを使用して署名を計算することができる。

<sig'> <P_B> <nonce> <sig_r>

【0214】

余分の署名sig'は、セキュリティのために追加された機能である(後述するオプションのセキュリティ機能についてのセクションを参照のこと)。しかしながら、このことは、全ての可能な実施形態において要求される必要はない。

【0215】

40

ここでのノンスは、全てのノード104により従われる基本ノードプロトコルに従ってトランザクションをブロック151にマイニングするために、トランザクションのプール154上のマイニングノードによって実行される固有の、根底にあるブルーフ・オブ・ワークに使用されるノンスとは異なることに留意されたい。従って、開示される技術は、基本ブロックチェーンプロトコルの上に、証明者(例えば、ボブ)によって実行される追加のブルーフ・オブ・ワークを重ねる。ボブが出力を償還するためには、kの値を知るだけでなく、同様に、有効なノンス値を得るために、あるブルーフ・オブ・ワークを行う必要がある。

【0216】

例示的なユースケースは、チャレンジャーである、アリスが、(ブルーフ・オブ・ワークに時間がかかるので)所定の時間が経過した後で、証明者(ボブ)だけがTx₁の出力で定義さ

50

れたデジタル資産の量を受け取ることができるようにすることを望んでいる場合であろう。これは、ロックタイムの代替として使用され得る。

【0217】

別の例は、計算能力の証明である。アリスは、利用可能な最も計算能力を誰が持っているか特定したいだろう。彼女はPoW rパズルを作成する。それを解く人は誰でも、最初に出力を支払い、そして、彼らが最大のハッシュパワーを持っていることを示す。このコンテキストでは、公開鍵Pまたは別のP₂を使用して、それらのアイデンティティをソリューションにリンクすることができる場合にも有用であろう(次のセクションを参照のこと)。アリスは、次いで、この計算能力を必要とする勝者のいくつかの更なるサービスを使用することを選択し得る。

10

【0218】

一般的に、ソリューションのr部を生成するために必要とされるkの値は、アリスまたは第三者によって証明者に与えられるか、または、証明者(または潜在的な証明のうち1つ)によってアリスに与えられ得る。例えば、計算能力の証明の例において、アリスまたは第三者は、kを潜在的な証明者に配布して、彼らがチャレンジに参加できるようにすることができる。

【0219】

上記のロッキングスクリプトの例において、ロッキングスクリプトは、何のr値が使用されているかのチェックが含まれていないことに注意する。これは、プルーフ・オブ・ワーク自体を検証するためには必須ではない。

20

【0220】

しかしながら、任意的に、実施形態において、アンロッキングスクリプトは、どのr値が使用されるかをチェックするために、いくつかの余分のコード(図示せず)を含む。これは、例えば、上述のPoW rパズルを、図7~9に関連して記載された任意のチェックと組み合わせることによって行われ得る。図14Bは、図7の検証とrパズル プルーフ・オブ・ワークの組み合わせを示している。図14Cは、図8の検証とrパズル プルーフ・オブ・ワークの組み合わせを示している。図14Cは、図9の検証とrパズル プルーフ・オブ・ワークの組み合わせを示している。これらの場合、ロッキングスクリプトおよびアンロッキングスクリプトは、一緒に実行されると、全てのチェックが満たされた場合にのみ「真」の結果を出力する。ここで、数字I)、II)、およびIV)は、彼がチェックする特定の順序が必須であることを意味するものではなく、そして、一般的に、チェックは互いに対して任意の順序で実行され得ることに注意されたい。

30

【0221】

他の実施形態では、ロッキングスクリプト内のr部について参照値を持たない利点が存在し得る。これにより、誰もが、他人がそれを盗むことなく、ソリューションを提供することができる。本アイデアは、証明者がk値を選んで、rを計算することである。次いで、その証明者は、dの可能な全ての値を実行する。どれもうまくいかなければ、その証明者は異なるkを試し得る。rとdのペアを見つけた後で、証明者は、r部としてrを用いて署名を構築することができる。

【0222】

一方、ロッキングスクリプトにおける固定r'(またはr'のハッシュ)は、証明者が、まずrパズルを解くか、または、この場合は望ましくないが、k値を与えられることを必要とする。

40

【0223】

上記は、連結に関して説明されてきた。しかしながら、rパズル プルーフ・オブ・ワークの任意のバージョンにおいて、これをある関数f(r,d)に一般化することも可能である。例えば、〈r〉 〈d〉 OP_ADDとして実装されるように、fは、rおよびdの加法であり得る。また、f(r,d)によって満たされるべき条件は、ハッシュがターゲット値未満であるか、または、少なくとも事前決定された数の先頭のゼロを有することに限定されない。一般的に、例えば、ハッシュがターゲットより大きい、または、例えば、所定の範囲内である

50

ような任意の条件が、プルーフ・オブ・ワークに使用され得る。さらに、同様のコメントは、上述のように、証明者の公開鍵Pについても適用される。すなわち、Pは、Tx₂で受け取り、または、r部およびs部から派生し、もしくは、いくつかの他の手段を介して受け取ることができる。

【0224】

任意的なセキュリティ機能#1

kに基づく署名が公表される場合、その値を知っている人なら誰でも、署名を作成するために使われた秘密鍵Vの値を引き出すことができる。これは、以下の署名方程式においてVについて解くことによって行うことができる。

$$s = k^{-1} (H(m) + rV) \bmod n$$

Vについて解いて、以下を得る。

$$V = r^{-1} (sk - H(m) \bmod n)$$

【0225】

多くの場合、トランザクションの受取人がkを知っている唯一の人であるため、このことは重大なリスクをもたらさない。他の場合に、支払者は、Rパズルへのソリューションに署名するために使用された秘密鍵Vを再使用しないように注意しなければならない。優れたセキュリティの実践では、利用者が公開鍵／秘密鍵のペア(P, V)を再利用することは決してせず、新しい資金を受け取る時には必ず新しい公開鍵／秘密鍵ペアを使用することが望ましいと定められている。

【0226】

原則として、公開鍵と秘密鍵のペア(P, V)は「永続的(“permanent”)」である。つまり、何度も使うことができる。ランダムな一時的鍵kの使用は、これを保証すべきである。しかしながら、乱数ジェネレータの実装が不十分なインシデントが存在してきた。同じ一時的鍵kおよび同じ秘密鍵を使用して2つの異なるメッセージに署名する場合、2つの署名から秘密鍵Vを引き出すことができる。すなわち、(r, s)およびkが与えられると、Vを計算することができる。ここで、 $r = [k \cdot G]_x$ であり、そして、Vは署名で使われた公開鍵Pに対する秘密鍵である。乱数発生器が署名プロセスの最中に故障した場合、前回と同じ乱数を生成し、その結果、秘密鍵を公に漏らすかもしれない。この問題に対処するために、人々は乱数発生器を修理するのではなく、公開鍵の再利用を避けるようになる。

【0227】

本ケースでは、アリスがkを知っているとしても、彼女は、V、ボブの公開鍵に対する秘密鍵を知らない。アリスがボブにkを引き継ぐと、ボブは、彼の秘密鍵を使用して(r, s)を提供することによって、rパズルを解くことができるだろう。アリスが署名をみると、彼女はkを知っているので、Vを導き出すことができるだろう。これは、ボブにとって望ましくない。従って、ボブは、好ましくは(P, V)の再使用を避けるべきである。

【0228】

しかしながら、これに伴う問題は、ボブの公開鍵Pが、ボブを識別する永続的な手段として使用できないことである。

【0229】

これに対処するために、ここにおいて開示される実施形態に従って、ボブは、対応する公開鍵P₂を有する別個の秘密鍵V₂を使用して、Tx₂にボブの追加の署名sig₂を含むことができる。彼は、また、追加の署名と共にP₂を含む。このように、2つのタイプの公開鍵と秘密鍵のペアが存在している。第1タイプは、一回限りの使用(one-time use)のためにその場(on the fly)で生成されるものである。他方のタイプは、いくつかの追加プロトコル、例えばHDウォレット、に従って生成されるものである。ボブは、rパズル署名のために第1タイプの鍵ペアを使用することができ、そして、第2署名のために第2タイプを使用することができる。

【0230】

次いで、アリスは、ボブの身元(identity)、例えば、固有の名前、ユーザ名、または

ネットワークアドレスを、公開鍵と身元との間のマッピングに基づいて調べる (look up) ために、このさらなる公開鍵を使用することができる。マッピングは、例えば、公開鍵を身元にマッピングする公開データベースにおいて利用可能とすることができ、または、マッピングは、単にアリスとボブとの間で事前に合意されたもの (例えば、アリスのコンピュータ装置 102a 上に私的に保存されている) であってよい。

【0231】

再び署名権者のユースケースを検討する。例えば、アリスは配達を受け取ることを望んでいるが、自分自身が配達を受け取れないかもしれない。彼女はボブとチャーリーの両方に k のコピーを渡し、そうして、彼らが彼女に代わって配達を受け入れられる。デイブは荷物を運んでいる。彼は、期待される r 値で署名しなければならない。ここで、デイブの記録または規制遵守について、デイブは受信者の身元も検証する必要があると想像してみる。

10

【0232】

ボブがそこで配達を受け入れると仮定する。ボブが、彼の公開鍵および k に基づく署名を作成する場合には、アリスとチャーリーの両方がボブの秘密鍵 V を作成することができる。公開鍵が一回限りの使用のために設計されているのであれば、このことは問題ではない。しかしながら、ボブが、彼の身元を将来に証明するためにこの公開鍵を必要とする場合には、理想的でない。

【0233】

この問題に対処するために、実施形態は、ボブを識別するために使用することができるボブからの r パズルとは独立した 1 つ以上の署名を Tx_2 に含むことができる。例えば、余分の (extra) 署名および対応する公開鍵 P_2 が、デイブが受け入れるのと同じトランザクション内の OP_RETURN 出力 (支払不能 (unspendable) 出力) に追加できる。代替案は、 r パズルトランザクションのロッキングスクリプトに余分の $OP_CHECKSIG$ を含めることである。余分の署名に使用されたトランザクションおよび公開鍵を閲覧することによって、アリスは、彼女のために誰が署名したかを知ることができる。

20

【0234】

その他のケースにおいては、使用前に値 k が漏れかもしれないという懸念が存在し得る。これに対処するために、アリスは、 r パズルトランザクションに $P2PKH$ を追加して、より安全にすることができる。アリスが、彼女の署名権 (signing right) をボブに委任したいと仮定する。アリスは、ボブから一回限りの公開鍵 k を取得し、そして、 r 値を指定するだけでなく、余分の公開鍵 P_2 も指定する r パズルトランザクションを作成する。

30

【0235】

アリス自身が同様に署名できるようにするために、任意的にアリスは 2 つの MultiSig のうち 1 つを作成できる。ロッキングスクリプトの例を以下に示す。

【表 8】

$TxID$	
入力	出力
任意の 支払い入力	$OP_DUP\ OP_3\ OP_SPLIT\ OP_NIP\ OP_1\ OP_SPLIT\ OP_SWAP\ OP_SPLIT$ $OP_DROP\ \langle r \rangle\ OP_EQUALVERIFY$ $OP_OVER\ OP_CHECKSIGVERIFY\ OP_CHECKSIGVERIFY$ $OP_1\ \langle Alice's\ PK \rangle \langle Bob's\ PK \rangle\ OP_2\ OP_CHECKMULTISIG$

40

【0236】

r パズルは、いつ r パズルのソリューションをボブに渡すかをアリスが選択できるので、より多くのフレキシビリティを提供することに注意されたい。彼女は、トランザクションがマイニングされた後でさえ、渡す (pass on) か、渡さないかを決定することができる。

【0237】

50

kが漏れた場合に、人々は、漏洩したkと共に署名に署名するために使用される秘密鍵を発見することができる。しかしながら、別の秘密鍵 V_2 、すなわち、ボブを識別するために使用できる公開鍵にリンクされた秘密鍵が、存在する。出力が危うくされるには、攻撃者は、2つの独立した秘密を取得する必要がある、それは、それらのうち1つだけを危うくする (compromising) よりも、はるかにありそうにない。

【0238】

上記の例において、 Tx_2 のロッキングスクリプトは、従来のP2PKH(rパズルで使用されたものではなく、余分の署名によってアンロックされるもの)によって、ボブの余分の公開鍵 P_2 にロックされていることに注意する。rパズル技術は、ユーザに対して追加の選択を可能にする。いくつかのアプリケーションでは、証明者が身元に関係なくチャレンジを満たすように、rパズルを使用することが望ましい場合がある。一方、他のアプリケーションでは、ハッシュパズルとP2PKHの組み合わせが依然として望ましい場合があり、rパズルは、それと共に任意的に使用することができる。これについて、後で詳しく説明する。

10

【0239】

しかしながら、身元ルックアップ及び／又はセキュリティのために対応する余分の署名 P_2 が必要とされ、P2PKHにおけるように、事前に特定の証明者の身元に結び付けられている Tx_1 のロッキングスクリプトがない場合、上記のロッキングスクリプトは、それに応じて適合させることができる。

【0240】

つまり、余分の署名に単にChecksigを含めることはできるが、対応する公開鍵 P_2 にOP_EQUALVERIFYを含めることはできない。

20

【0241】

任意的なセキュリティ機能#2

上記の方法におけるもう一つの潜在的なセキュリティ脆弱性は、署名の偽造可能性 (forgeability) である。これは、資金を要求しようとするマイナによって (ハッシュパズルと同じように) 悪用されるかもしれない。(支払者から) トランザクションを受け取るマイナは、元のトランザクションで支払者が使用したのと同じ署名を使って、資金を彼自身に送るためにトランザクションを変更することができる。これは、以下のように行われる。

【0242】

$P = V \cdot G$ を元のトランザクションに署名するために使用される公開鍵／秘密鍵のペアとする。署名 (r, s) を得るために m によって、以下のように示されるものである。

$$r = [k \cdot G]_x, \text{ かつ、} \\ s = k^{-1} (H(m)) + rV \bmod n$$

30

【0243】

このトランザクションを使用するには、支払者は、以下のアンロッキングスクリプトを使用する。

$\langle P \rangle \langle r, s \rangle$

【0244】

このトランザクションを受け取ったマイナは、トランザクションを新たなものへと変更することができる。 m' によって示されており、以下の新しいアンロッキングスクリプトを使用して資金 (fund) を自分自身に送る新しいトランザクションに変更することができる。

40

$\langle P' \rangle \langle r, s \rangle$

ここで、 $P' = V' \cdot G$ は、以下のような公開鍵／秘密鍵のペアである。

$$V' = V + r^{-1} [H(m) - H(m')] \text{、かつ、} \\ P' = P + r^{-1} [H(m) - H(m')] \cdot G$$

【0245】

マイナは、 V' を知る必要がないことに注意する (彼らは V を知らないからである)。検証プロセスは、以下の計算を使用して行われる。

$$R' = H(m) s^{-1} \cdot G + r s^{-1} \cdot P$$

50

【0246】

署名は、 $(R')_x = r$ である場合にのみ有効であり、そうでなければ、無効である。

【0247】

新しいトランザクション m' と新しいアンロッキングスクリプトを用いて、検証プロセスは、以下のとおりである。

$$\begin{aligned} R' &= H(m') s^{-1} \cdot G + r s^{-1} \cdot P' \\ &= H(m') s^{-1} \cdot G + r s^{-1} \cdot \{P + r^{-1} [H(m) - H(m')] \cdot G\} \\ &= r s^{-1} \cdot P + H(m) s^{-1} \cdot G \\ &= r \end{aligned}$$

(プライム表記 (primed notation) は、ここにおいては、以前とは異なる意味を有することに注意する。このコンテキストにおけるプライム表記は、参照インスタンスを指すものではない)。

【0248】

この潜在的な脆弱性に対処するために、実施形態は、秘密鍵 V を知っていなければマイナが提供することができない別のメッセージ $m_{\text{signature}}$ におけるアンロッキングスクリプトに別の余分の署名 sig' を含むことができる。その場合、アンロッキングスクリプトは、以下のようになる。

$\langle \text{sig}' \rangle \langle P \rangle \langle \text{sig}_r \rangle$

【0249】

sig' は、異なるメッセージ $m_{\text{signature}}$ における署名であってもよいので、メッセージを変更するには元のものとは異なる signature フラグを使用するだけである(例えば、デフォルトのフラグである SIGHASH_ALL の代わりに SIGHASH_NONE)。また、 sig' は、秘密鍵を漏らさないように、 r の異なる値を使用しなければならない(なぜなら、秘密鍵は、同じ一時的な鍵を使用する2つの署名から派生され得るからである)。最終的に、トランザクションは、以下に示すように、最後に別の OP_CHECKSIG を含める必要がある。

【表9】

$TxID$	
入力	出力
任意の 支払い入力	$\text{OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP}$ $\text{OP_SPLIT OP_DROP } \langle r \rangle \text{ OP_EQUALVERIFY}$ $\text{OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG}$

【0250】

これは、公開鍵 P に対する秘密鍵 V を知っている人だけが別の署名を作成することができるように、 r パズルと同じ公開鍵 P を使わなければならない。そして、上記の攻撃は不可能である。

【0251】

攻撃者は、公開鍵を、攻撃者が秘密鍵の知識を有していない別の公開鍵に置き換えようとしている。この攻撃を防ぐために、チャレンジは、また、秘密鍵の知識も要求する。この場合、1つの署名では不十分である。従って、2つの署名が必要である。両方の署名は、同じ秘密鍵の知識の証明 (proof) とみなされる。これは、チャレンジが異なる一時的な鍵を持つことを主張するので、安全である。

【0252】

P2PKH+P2PRPH

R パズルが知識証明 (knowledge proof) として使用されるのと同様に、 $P2PKH$ 出力は、また、 $P2PKH$ 出力における公開鍵に対応する秘密鍵の知識の知識証明でもある。これは、本質的にパズル k を、 $P2PKH$ 出力における公開鍵 $P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G$ にマップする秘密鍵 V_{puzzle} で置き換えることによって行われる。 r パズルの実施形

態では、知識証明は、証明者が選択することができ、かつ、身元にリンクするために使用することができる公開鍵を伴う。P2PKHでは、通常の支払い署名および公開鍵(秘密のパズルの知識を証明するもの)は、特定の身元にリンクするための別の署名および公開鍵を伴わなければならない。簡単に言えば、以下に示すように、証明者は、P2PKHアンロックに対して別の署名および公開鍵を追加することができ、それは、ロックキングスクリプトに対する別のOP_CHECKSIGに対応している。

【表 1 0】

TxID	
入力	出力
任意の 支払い入力	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_CHECKSIG

10

【 0 2 5 3 】

対応するアンロックングスクリプトを以下に示す。

$\langle \text{sigP}_{\text{ID}} \rangle$ $\langle P_{\text{ID}} \rangle$ $\langle \text{sigP}_{\text{puzzle}} \rangle$ $\langle P_{\text{puzzle}} \rangle$

【 0 2 5 4 】

$\langle \text{sigP}_{\text{ID}} \rangle$ $\langle P_{\text{ID}} \rangle$ と、パズルまたはトランザクションとの間に、暗号技術的なリンクが存在しないことに注意されたい。マイナおよび誰もが、それらを別の公開鍵において別の署名と置き換えることができ、これは、マイナがオープンなハッシュパズルをインターセプト (intercept) できる方法と少し似ている。マイナおよびインターセプタは(オープンなハッシュパズルの場合のように)資金を自分自身に送るためにメッセージを変更することができない。しかしながら、自分自身の身元とリンクされて見えるようにするために、 $\langle \text{sigP}_{\text{ID}} \rangle$ $\langle P_{\text{ID}} \rangle$ を自分自身の身元と置き換えることを止めるものは何もない。

20

【 0 2 5 5 】

他方、ここにおいて開示された技術に基づいて、P2PKHとP2RPHの両方を一緒に同じスクリプト内で使用することが可能であり、上記の場合と同様、それをインターセプトし、置き換えることができないように、第2署名における暗号リンクを強制することができる。

30

【表 1 1】

TxID	
入力	出力
任意の 支払い入力	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY
	OP_CHECKSIGVERIFY
	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT
	OP_DROP OP_HASH160 $\langle H(r_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_SWAP
	OP_CHECKSIG

40

【 0 2 5 6 】

ロックキングスクリプトにおいては、 $P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G$ であり、一方、 $r = [R]_x = [V_{\text{puzzle}} \cdot G]_x$ であり、そうして、それらは基本的に等価である。公開鍵は、圧縮されても、圧縮されなくてもよく、そして、どちらの方法でもプレフィックスが付加されるので、それらは実際には等しくない。以下に示すように、 $\langle H(P_{\text{puz}}$

50

z l e) >=<H (r p u z z l e) >を作成するために署名からr値を抽出するとき、明示的にプレフィックスをスクリプトに追加することも可能である。しかしながら、これは、実際にはあまり有益ではない。

【表 1 2】

TxID	
入力	出力
任意の 支払い入力	OP_DUP OP_HASH160 <H(P _{puzzle})> OP_EQUALVERIFY
	OP_CHECKSIGVERIFY
	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAT OP_SPLIT
	OP_DROP OP_PUSHTA1 <r - prefix> OP_SWAP OP_CAT
	OP_HASH160 <H(r _{puzzle})> OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

10

(ここで、H (. . .) は、ロッキングスクリプトの模式的表現における三角括弧<>を参照し、これは、ハッシュ関数ではなくハッシュの値を参照するように実際には意図されていることに注意する。三角括弧<>は、この値をスタックに置くことを意味する)。

20

【 0 2 5 7】

両方のトランザクションについて対応するアンロッキングスクリプトは、以下のとおりである。

<PID> <r_{puzzle},s> <sigP_{puzzle}> <P_{puzzle}>

【 0 2 5 8】

アカウントベースのモデルにおける代替的な実装

上記は、主に出力ベースのモデル(例えば、UTXOベースのモデル)における実装の観点から説明されてきた。しかし、これは限定的ではないことが理解されるであろう。図11は、アカウントベース (account-based) モデルを用いた可能な代替的な実装を示している。

30

【 0 2 5 9】

要するに、アカウントベースモデルにおいて、rパズル機能は、ユーザによって呼び出される (called) スマートコントラクト機能に含めることができる。一方の当事者は、スマートコントラクトにおいてrパズル値(または、ハッシュrパズル値)を設定することができ、そして、次いで、他方の当事者が、その後で、スマートコントラクトに署名を提供するだろう。

【 0 2 6 0】

UTXOブロックチェーンアーキテクチャにおいて、第1トランザクションのアンロッキングスクリプトに具体化された要件は、第2トランザクションが有効として受け入れられ、かつ、ブロックチェーン内に記録されるために、第2トランザクションのアンロッキングスクリプトによって満たされなければならない。現在の状況において、これは、トランザクションの検証プロセスの一部として既にマイナによって行われている作業を活用するので、有益である。現在のコンテキストにおける具体的な例として、トランザクションがブロックチェーンに追加されたという事実は、それがブロックチェーンネットワーク全体のノードによって検証されたことを意味し、それは、次に、そのロッキングスクリプトがいくつかの特定の有用な要件を満たすことを意味する。利害関係者 (interested party) は、これらの要件が満たされているか否かを自らチェックする必要はない。一すなわち、彼らは、トランザクションがブロックチェーンで正常に記録されたという事実によってそれらの要件が満たされると単純に仮定することができる。このことは、トランザクションが有効になるには、完了時にスクリプトが「真」の結果を返さなければならないこと(ト

40

50

ランザクションが有効になるための他の要件が存在し得る)、そして、スクリプトが「偽」の結果を返す場合(ここにおいて使用される用語によれば、スクリプトが失敗した場合を含む、例えば、OP_VERIFYオペコードがスクリプトを終了するので)トランザクションは無効であるという事実による。

【0261】

しかしながら、他のブロックチェーンモデル(例えば、所定のアカウントベースのアーキテクチャ)において、トランザクションの妥当性と実行中のトランザクションコードの結果との間のこの相互依存性は、必ずしも反映されない。例えば、所定のスマートコントラクトブロックチェーンでは、ブロックチェーンプロトコルによって課される「基本的な(“basic”)」有効性要件のセットを満たすならば、トランザクションは有効であり、従って、ブロックチェーン上の記録に受け入れられ得る。従って、第2トランザクションは、第1トランザクションのコードに具体化されたいくつかの要件を満たさなくても、依然として有効として受け入れられ、そして、ブロックチェーンに記録され得る。第1トランザクションのコードは、例えば、スマートコントラクトコードであってよい。

【0262】

第2トランザクションが、第1トランザクションによって作成されたスマートコントラクトアカウント宛てのものであると仮定すると、そのトランザクションにどのように応答するかを決定するのはスマートコントラクトコードである。すなわち、例えば、いくつかの要件が満たされない場合には、それを無視することができる(または、そうでなければ、誤った結果を返す)。一方で、その要件が正しければ、スマートコントラクトアカウントの残高(balance)から差し引かれたデジタル資産の量を証明者に報奨(reward)することができる(または、そうでなければ、真の結果を返す)。ある意味で、これは、スマートコントラクト(エージェント)による「エージェントレベル(“agent-level”)」処理、すなわち、スマートコントラクトコードに明示的にコード化されたものを、ノードによって「暗黙的に(“implicitly”)」実行される「プロトコルレベル(“protocol-level”)」処理から抽象化する。すなわち、トランザクション上で実行される処理は、それがブロックチェーンネットワークが動作するブロックチェーンプロトコルによって課される有効性の要件を満たすか否かを決定する。従って、そのようなブロックチェーンアーキテクチャにおいて、トランザクションそれぞれのプロトコルレベルでのノードによる「有効/無効」決定は、スマートコントラクトによってエージェントレベルでのそのトランザクションに関して返される「真/偽」結果から切り離されてよく、その場合、トランザクションはプロトコルレベルで有効であると決定されるが、それにもかかわらず、エージェントレベルでは偽の結果を返す。

【0263】

これは、トランザクションが有効であるために「真」の結果を返すスクリプトが必要とされる、UTXOアーキテクチャとは対照的である。トランザクションは、スクリプトがスタック上で真以外のものを残して終了または完了した場合、無効となる(これらの結果のいずれも、その用語がここにおいて使用されるように「偽」の結果を構成する)。

【0264】

トランザクションの有効性に関する基本的な要件の1つは、トランザクションが有効な署名を含むことである。従って、上記のUTXOの例では、チャレンジトランザクション自体のコードによって署名が検証される一方で(例えば、署名を検証し、署名検証について真/偽を返すOP_CHECKSIGオペコード、または、署名を同じようにチェックし、追加的に、結果が真であることを検証するOP_CHECKSIGVERIFYオペコードを使用し、スクリプトは、そうでない場合に終了する)、代替的なブロックチェーンアーキテクチャにおいては、署名は、上記の意味で処理ノードによって暗黙的に検証され得る。これは、トランザクションコード自体において署名チェックをコード化する必要性を回避し得る。

【0265】

現在のコンテキストにおける具体的な例として、トランザクションは、例えば、有効な署名を含むので、プロトコルレベルでは有効であると見なされ得るが、例えば、いくつか

10

20

30

40

50

の他の要件が満たされないので、未だにアプリケーションレベルでは偽の結果を返し得る。
【0266】

図11は、アカウントベースモデルに従ってトランザクションを処理するためのノードソフトウェア400の代替案を示しており、ノードソフトウェアは、ここでは400accとラベル付けされている。このノードソフトウェア400accのインスタンスは、ネットワーク106のアカウントベース・バージョンのノード104それぞれで実装され得る。アカウントベースのノードソフトウェア400accは、アカウントベースのプロトコルエンジン401acc、コントラクトエンジン402acc(いくらかスクリプトエンジン402に類似している)、アプリケーションレベルの決定エンジン404、および、1つ以上のブロックチェーン関連機能モジュール405のセット、を備える。任意の所与のノード104において、これらは、マイニングモジュール405M、転送モジュール405F、および、ストレージモジュール405Sのいずれか1つ、2つ、または全ての3つを(ノードの役割または役割に応じて)含んでよい。プロトコルエンジン401accは、トランザクションの異なるフィールドを認識し、ノードプロトコルに従ってそれら进行处理するように構成されている。ノードソフトウェア400accは、また、各ノード104のメモリ内に複数のアカウントそれぞれのアカウント状態406を維持する。これらは、例えば、アリス、証明者(例えばボブ)、及び／又は、アリスと証明者との間で制定されるコントラクトに応じて借方記入(debited)または貸方記入(credit)される別の当事者のアカウントを含む。コントラクトエンジン402accは、トランザクションで受け取ったスマートコントラクトの結果に応じて、アカウント状態を変更するように構成されている。スマートコントラクトは、また、「エージェント(“agent”）」とも呼ばれる。

【0267】

図11は、また、トランザクションTx₁^{acc}およびTx₂^{acc}のペアを示しており、図7～図10に関連して上述したのと同様または類似のrパズル機能を実装することができる。それぞれは、ソースアカウントアドレス1102(ソースアドレスフィールド)、および、宛先アカウントアドレス1103(宛先アドレスフィールド)を含む。第1トランザクションTx₁^{acc}は、ソースアカウントアドレス1102aおよび宛先アカウントアドレス1103aを含む。第2トランザクションTx₂^{acc}は、ソースアカウントアドレス1102bおよび宛先アカウントアドレス1103bを含む。第1トランザクションTx₁^{acc}は、また、スマートコントラクト1101も含む。スマートコントラクト1101は、アリスによるチャレンジ(パズル)を含み得る。それは、アリスまたはアリスに代わる第三者によって、アリスから提供された詳細情報を使用して作成することができる。第2トランザクションTx₂^{acc}は、任意的に、ユーザ指定ペイロードデータを搬送するための1つ以上のフリーデータフィールド1104を含み得る。これは、証明者、例えばボブ、によって提供されたパズルに対するソリューションの少なくとも一部を含んでよい。トランザクションTx₁^{acc}およびTx₂^{acc}は、アリスとその証明者がそれぞれ署名する。各トランザクションは、また、それぞれの当事者の署名1105a、1105bも含む。

【0268】

トランザクションは、ネットワーク106を介してブロードキャストされる。プロトコルエンジン401accは、各トランザクションを受信すると、署名1105が有効であるか否かを暗黙的に検証する。すなわち、これは、プロトコルエンジン401accの固有の機能であり、そして、スマートコントラクト1101において指定する必要はない。従って、プロトコルエンジン401accは、少なくともそれぞれの署名が有効であるという条件の下で、転送及び／又はマイニングのために各トランザクションを検証する。また、妥当性を満たすためには、1つ以上の追加条件を必要とし得る。有効であれば、アプリケーションレベルの決定エンジン404は、マイニングモジュール405M及び／又は転送モジュール405Fを制御して、それぞれにトランザクションをマイニング及び／又は転送するか否かを選択することができる。

【0269】

そうしたアカウントベースのモデルにおいて、アリス、ボブ、およびスマートコントラ

クト自体には、異なるアカウントアドレスを有する、別個のアカウントが割り当てられる。トランザクションは、その宛先アドレスフィールドにおけるアドレスへ（“to”）、ソースアドレスフィールドのアドレスから（“from”）送信するように言われる。スマートコントラクトのアカウントを作成するために、スマートコントラクトのバイトコードを含むトランザクションが、トランザクション内のブロックチェーンにアップロードされる。そうしたアカウント生成トランザクションについて、宛先フィールド内の宛先アドレス1103は、ブロックチェーン内でこれまでに使用されたことのないアドレスであるべきであり、そして、一旦トランザクションが受け入れられると、そのアドレスは、新しく作成されたスマートコントラクトアカウントのアドレスとなる。その後、スマートコントラクトを「呼び出す（“call”）」するため、すなわち、さらなるトランザクションに依存してスマートコントラクトのバイトコードを実行させるために、さらなるトランザクションがそのアドレスに送信され得る。「宛先（“destination”）」アドレス1103は、コントラクトを成立させるための仲介アドレスとして作用する。—すなわち、アリスは、1つ以上の要件を指定するスマートコントラクトを作成するために、 Tx_1^{acc} をそのアドレスに送信し、ボブは、スマートコントラクトを呼び出すために、同じアドレスに Tx_2^{acc} を送信する。それは、次いで、スマートコントラクトに、 Tx_2^{acc} が指定された要件を満たすか否かを検証させる。「ソース（“source”）」アドレス1102は、コントラクトの当事者であるユーザのアカウントを指定する。—すなわち、 Tx_1^{acc} が指定された要件を満たしていると、スマートコントラクト判断する場合、スマートコントラクトは、それ自身のアカウントの残高からデジタル資産の量を差し引く（deduct）ように構成され得る。そして、 Tx_2^{acc} におけるソースアドレス1102bを有するアカウント（つまり、ボブのアカウント）の残高をその量まで貸方記入（credited）させる（直感的に、 Tx_2^{acc} を送信することによって、ボブは、スマートコントラクト（宛先アドレスフィールドで識別される）に対して、そのアカウント（ソースアドレスフィールドで識別される）の貸方記入を効果的に要求する）。

【0270】

プロトコルエンジン401accが Tx_2^{acc} を受信すると、それが有効であるという条件で、 Tx_2^{acc} における宛先アドレス1103bに一致するアカウントを探す。 Tx_1^{acc} が処理され、かつ、有効であると仮定すると、そのアカウントは、 Tx_1^{acc} のおかげで存在し、そして、 Tx_1 において提供されるスマートコントラクトコードに関連付けられる。これにตอบสนองして、プロトコルエンジン401accは、コントラクトエンジン402accを制御して Tx_1^{acc} からのスマートコントラクト1101を実行し、コントラクトで定義される基準に応じて、スマートコントラクトの1つ以上のフィールドからのデータをオペランドデータとして取り込む。オペランドデータは、例えば、フリーデータフィールド1104の1つ以上からのデータ、及び／又は、署名フィールド1105bからの署名を含んでよい。 Tx_1^{acc} のスマートコントラクト1101において定義された1つ以上の基準を Tx_2^{acc} からのオペランドデータが満たすことを条件として、コントラクトエンジン402accは、スマートコントラクト1101で定義された変更に従って、1つ以上の当事者（アリス、証明者、及び／又は、1つ以上の第三者）のアカウント状態406を変更する。そうでなければ、アカウント状態406に対するこの修正は行われない。しかしながら、いくつかのアカウントベースのシステムにおいて、スマートコントラクトの結果は、トランザクションの有効性の条件ではないことに注意する。従って、 Tx_1^{acc} のスマートコントラクト1101に設定された基準を Tx_2^{acc} が満たさない場合でも、 Tx_2^{acc} は、なおも失敗したトランザクションのレコードとして伝搬され、そして、ブロックの中へマイニングされる。また、それは、依然として、マイニング料を課すことができる（従って、プロトコルエンジン401は、一方の当事者および勝ったマイナのアカウント状態406を依然として変更することができる）。

【0271】

rパズルを実装するために、rパズル機能の少なくとも一部は、 Tx_1^{acc} のスマートコントラクト1101の中にコード化することができ、そして、そのソリューションは、 Tx_2^{acc} のデータフィールド1104のうちの1つ以上に提示され得る。例えば、これは、図7の変形を実装するために使用され得る。任意的に、プロトコルエンジン401accの暗黙の署名検証

10

20

30

40

50

機能のいくつかを、例えば、図8～10の変形のうち1つを実装するために、利用し得る。図8～10の場合、ステップⅠⅠ)およびⅠⅠⅠ)は、 Tx_2^{acc} の署名を検証するときに、プロトコルエンジン401accの暗黙の機能であってもよい(署名検証それ自体が、プロトコルエンジン401accによって実装されるノードプロトコルの固有の機能であることを思い出すこと)。従って、ステップⅠ)を、 Tx_1^{acc} のスマートコントラクト1101においてこれの上部に層化することだけが要求される。スマートコントラクトは、Ⅰ)の結果が真であるか否か、および、プロトコルエンジン401accが Tx_2^{acc} は有効であることを示すか否かをチェックする。両方が「はい(yes)」の場合、検証について「真」という全体的な結果を宣言する。つまり、ボブはrパズルによって設定されたチャレンジを成功裡に満足した。図8～10の実装の場合、図9および図10の場合のデータ値dのみが、フリーデータフィールド1104に含まれる必要があることに留意されたい。署名情報は、署名フィールド1105bに含まれる。

10

【0272】

スマートコントラクトアカウントは、また、アカウントに関連付けられた(論理的な)データストレージ素子である「データレジスタ(“data register”)」(図示せず)にインデックス付けしている。上記に概説したUTXOモデルにおいては、値がロッキングスクリプト自体に埋め込まれており、そして、スマートコントラクトコード1101の特定部分にも同じことが成立し得る。しかしながら、スマートコントラクトのスマートコントラクトバイトコードは、そのアカウントレジスタの1つ以上に保管されたデータ上で、代替的にまたは追加的に実行され得る。さらに、スマートコントラクトアカウントが作成された後に、スマートコントラクトアカウントレジスタに値を保管することが一般的に可能である。それで、例えば、スマートコントラクトアカウントは、スマートコントラクトバイトコードを含む Tx_1, α^{acc} のチャレンジトランザクションによって作成され得る。次に、別個の「中間(“intermediate”)」トランザクション Tx_1, β^{acc} が(現在存在する)スマートコントラクトアカウントに送信される。それは、スマートコントラクトアカウントのレジスタ\$Rに特定の値vを保管する効果がある。スマートコントラクトは、(例えば)指定されたソースアカウントアドレスからのみ、そのようなデータを受け入れるように設定されてよい。例えば、最初にスマートコントラクトを作成したのと同じ当事者(アリス)。 Tx_2^{acc} が受信されると、コントラクトエンジン402accによって実行される操作(例えば、「アクセスレジスタ\$Rにアクセスして、値を Tx_2^{acc} のデータフィールド\$Dの値と比較する」)

20

30

【0273】

従って、いくつかの実装形態では、チャレンジトランザクション Tx_1, α^{acc} は、rパズルの動作を定義し得るが(例えば、証明トランザクション Tx_2^{acc} の署名のr部をレジスタ\$Rにおける値と一致するかを見るために比較する)、\$Rにおける値は、証明トランザクション Tx_2^{acc} のr部と比較されるが、その中の値は、中間トランザクション Tx_1, β^{acc} によって設定されてよい。

40

【0274】

注記:いくつかのアカウントベースのモデルは、署名1105に公開鍵Pを含める必要がない。その代わりに、単に1ビットのフラグを含む。既に述べたように、(r, s)から2つの可能な鍵Pおよび-Pと、メッセージを導出することができる。フラグflgは、これらの2つの可能なソリューションのうち、実際には、証明者が Tx_2^{acc} におけるメッセージに署名するために使用した秘密鍵Vに対応する公開鍵であることを信号化するために使用される。プロトコルエンジン401accは、これ(r, s)およびflgを使用し、 Tx_2^{acc} で明示的に受信する代わりに、証明者の公開鍵Pを導出する。この実装方法は、また、出力ペー

50

スのモデルでも可能であり、アカウントベースのモデルに特有のものではないが、多くの現在の出力ベースのモデルで使用されているスクリプト言語では、 r および s から P を導出するための専用のオペコードが存在しないため、スタックベースの言語の既存の汎用オペコードを使用して、この機能をアンロックングスクリプトに明示的にコード化することは複雑である。さらに、特定のアカウントベースのモデルは、トランザクションのソースアドレスを、そのトランザクションに署名するために使用された公開鍵から導出することに留意されたい。従って、ソースアドレスはトランザクションで別々に符号化される必要はなく、公開鍵が署名から導出される場合、これはソースアドレスも間接的に署名から導出され得ることを意味する。

【0275】

10

結論

上記の実施形態は、単なる例示として説明したものであることが理解されるであろう。

【0276】

より一般的に、ここにおいて開示される教示の第1インスタンス化に従って、ブロックチェーンネットワークのノード検証において、コンピュータに実装される方法が提供される。本方法は、実行可能コードを含む第1トランザクションを取得するステップと、情報を含む第2トランザクションを受信するステップであり、前記情報は、第1ECDSA署名の r 部および s 部の少なくとも提出されたインスタンスと、ノンスと、をさらに含む、ステップと、第1トランザクションからのコードを実行するステップ、を含む。前記コードは、 $H_{pow}(f(r, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、 r は前記 r 部の提出されたインスタンスであり、 d はノンスであり、 H_{pow} はハッシュ関数であり、 f は r および d を組み合わせた関数である。

20

【0277】

実施形態において、第2の、現在開示されている教示の任意的なインスタンス化により、第1インスタンス化に従った方法が提供されてもよく、ここで、 $r || d$ は連結(concatenation)である。

【0278】

本開示の第3の、任意的なインスタンス化により、第2インスタンス化に従った方法が提供され得る。ここで、前記事前決定された条件は、 $H_{puz}(r || d)$ が、事前決定されたターゲット値未満であること、または、少なくとも事前決定された先頭ゼロの最小数を有することである。

30

【0279】

本開示の第4の、任意的なインスタンス化により、第1、第2、および第3インスタンス化に従った方法が提供され得る。本方法は、公開鍵を取得するステップであり、前記第1ECDSA署名は、前記公開鍵に対応する秘密鍵に基づいてメッセージに署名し、前記メッセージは、前記第2トランザクションの一部である、ステップと、ECDSA検証機能を適用するステップであり、前記公開鍵と前記メッセージに基づいて、前記第2トランザクションで受信した前記第1ECDSA署名を検証する、ステップと、を含む。ここで、前記コードは、前記第1ECDSA署名の前記検証のさらなる条件について真の結果を返すように構成されている。

40

【0280】

出力ベースのモデル(例えば、UTXOベースのモデル)では、ECDSA検証機能は、第1トランザクションの出力(例えば、UTXO)のロックングスクリプト内のオペコードによって呼び出され得る。オペコードは、検証ノードにあらかじめ保管されたECDSA検証機能のインスタンスを呼び出すことができる。代替的に、アカウントベースのモデルのように、ECDSA検証機能は、ノードの暗黙の機能であってもよく、ノードプロトコルの一部として自動的に実行され、第1トランザクション(アカウントベースの場合はスマートコントラクト)のコードによって明示的に呼び出される必要はない。別の代替案として、ECDSA検証がコードに明示的にコード化され得ることは除外されない。

50

【0281】

第5の、任意的なインスタンス化により、第4インスタンス化に従った方法が提供される。ここで、前記コードは、さらに、前記第1ECD SA署名の前記r部に対応する参照値を含み、前記参照値は、前記r部の参照インスタンスまたはその変換であり、かつ、前記コードは、前記参照値が前記第2トランザクションで受信したr部の前記参照インスタンスに対応することを検査し、かつ、さらなる条件で真の結果を返すように構成されている。

【0282】

第6の、任意的なインスタンス化により、第5インスタンス化に従った方法が提供される。ここで、前記参照値は、前記ECD SA署名の前記r部の参照インスタンスである。

10

【0283】

第7の、任意的なインスタンス化により、第6インスタンス化に従った方法が提供される。前記コードは、

前記 $H_{Pow}(f(r, d))$ が、前記事前決定された条件を満たしていることを検査し、かつ、

前記ECD SAの検証機能が、 $R' = H_{Sig}(m) s^{-1} \cdot G + r' s^{-1} \cdot P$ を計算し、

$[R']_x = r'$ であることを検査する、ように構成されている。

ここで、前記 r' は前記第1ECD SA署名のr部の前記参照インスタンスであり、 s は前記第1ECD SA署名の前記s部であり、 P は第1公開鍵であり、 m は前記第1ECD SA署名によって署名された前記第2トランザクションの部分であり、 H_{Sig} は前記第1ECD SA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' のX座標を表し、「 $\cdot$ 」は楕円曲線スカラー乗算を表している。この場合、前記コードは、前記検査の両方が真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている。

20

【0284】

H_{Pow} はハッシュパズルで使用するハッシュ関数である。 H_{Sig} は、ECD SA署名と検証で使用するハッシュ関数である。 H_{Pow} と H_{Sig} は、ハッシュ関数の同じ形式であってもなくてもよい。

【0285】

30

第8の、任意的なインスタンス化により、第6インスタンス化に従った方法が提供される。前記コードは、

前記 $H_{Pow}(f(r, d))$ が、前記事前決定された条件を満たしていること、および、 $r' = r$ であることを検査し、かつ、

前記ECD SAの検証機能が、

$R' = H_{Sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ を計算し、

$[R']_x = r$ であることを検査する、ように構成されている。

ここで、前記 r は前記第1ECD SA署名のr部の前記提出されたインスタンスであり、前記 r' は前記第1ECD SA署名のr部の前記参照インスタンスであり、 s は前記第1ECD SA署名の前記s部であり、 P は第1公開鍵であり、 m は前記第1ECD SA署名によって署名された前記第2トランザクションの部分であり、 H_{Sig} は前記第1ECD SA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' のX座標を表し、「 $\cdot$ 」は楕円曲線スカラー乗算を表している。この場合、前記コードは、前記検査の3つ全てが真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている。

40

【0286】

第9の、任意的なインスタンス化により、第5インスタンス化に従った方法が提供される。ここで、前記参照値は、前記第1ECD SA署名のr部の参照インスタンスの変換であり、前記コードは、前記提出されたインスタンスにおいて同じ変換を実行し、かつ、前記参照値と比較することによって、前記提出されたインスタンスが前記参照値に対応する

50

ことを検査するように構成されている。

【0287】

第10の、任意的なインスタンス化により、第9インスタンス化に従った方法が提供され得る。ここで、前記参照値は、ハッシュ値であって、前記第1ECD SA署名のr部の前記参照インスタンスのハッシュである。

【0288】

第11の、任意的なインスタンス化により、第10インスタンス化に従った方法が提供され得る。ここで、前記コードは、

前記 $H_{pow}(f(r, d))$ が、前記事前決定された条件を満たしていること、および、 $h = H_{puz}(r)$ であることを検査し、かつ、

前記ECD SAの検証機能が、

$R' = H_{sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ を計算し、

$[R']_x = r$ であることを検査する、ように構成されている。

ここで、 r' は前記第1ECD SA署名のr部の前記参照インスタンスであり、 r は前記第1ECD SA署名のr部の前記提出されたインスタンスであり、 s は前記第1ECD SA署名の前記s部であり、 h はハッシュ値であり、 H_{puz} は h を生成するように r' をハッシュするために使用されたハッシュ関数であり、 P は第1公開鍵であり、 m は前記第1ECD SA署名によって署名された前記第2トランザクションの部分であり、 H_{sig} は前記第1ECD SA署名を生成する際に m をハッシュするために使用されたハッシュ関数であり、 G は楕円生成点であり、 $[R']_x$ は R' の x 座標を表し、「 $\cdot$ 」は楕円曲線スカラー乗算を表している。この場合、前記コードは、前記検査の3つ全てが真であるという条件で真の結果を返すように構成されているが、そうでなければ、偽の結果を返すように構成されている。

【0289】

H_{puz} はハッシュパズルで使用されるハッシュ関数である。 H_{puz} は、 H_{sig} 及び／又は H_{pow} と同じハッシュ関数の形式であってもなくてもよい。

【0290】

第12の、任意的なインスタンス化により、第1乃至第11いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記第1公開鍵を取得することは、前記第2トランザクションにおける情報の一部として前記第1公開鍵を受信すること、を含む。

【0291】

代替的に、第2トランザクションの情報は、第1ECD SA署名のr部のs部とサブミットされたインスタンスの両方を含み、前記取得することは、第1ECD SA署名のr部のs部分サブミットされたインスタンスの組み合わせから第1公開鍵を導出することを含む。

【0292】

別の代替案として、取得することが、例えば、第2トランザクションに関連してサイドチャンネルを介して第1公開鍵を受信すること、または、第2トランザクションまたはサイドチャンネルを介して、第1公開鍵のインデックスまたは第1公開鍵と秘密鍵の所有者のIDを受信し、これを使用して第1公開鍵を探索すること、を含み得ることは除外されない。

【0293】

第13の、任意的なインスタンス化により、第1乃至第12いずれか1つのインスタンス化に従った方法が提供され得る。ここで、第1ECD Aのr部およびs部の提出されたインスタンスは、第1当事者によって第2当事者に与えられた一時的鍵、またはその逆も同様、および、前記第2当事者の秘密鍵である第1秘密鍵を使用して、前記第2当事者によって生成されたものであり、かつ、前記ノンスは、また、前記第2当事者のコンピュータ装置においてブルーフ・オブ・ワークを前記第2当事者が実行すること生成されたものである。

【0294】

第14の、任意的なインスタンス化により、第13のインスタンス化に従った方法が提供され得る。ここでは、 $P = V \cdot G$ 、 $k \in [1, n-1]$ 、 $R = k \cdot G$ 、 $r = [R]_x$ 、

かつ、 $s = k^{-1} (H_{sig}(m) + rV) \bmod n$ 、である。

ここで、Pは第1公開鍵であり、Vは前記第1秘密鍵であり、kは前記一時的鍵であり、nは素数係数であり、Gは楕円生成点であり、mは前記第1ECDSA署名によって署名された前記第2トランザクションの部分であり、 H_{sig} は前記第1ECDSA署名を生成する際にmをハッシュするために使用されたハッシュ関数であり、 $[R]_x$ はRのx座標を表し、かつ、「 $\cdot$ 」は楕円曲線スカラ乗算を表している。

【0295】

第15の、任意的なインスタンス化により、第13または第14のインスタンス化に従った方法が提供され得る。ここで、前記第2トランザクションを受信することは、前記第2当事者から前記第2トランザクションを受信することを含む。

【0296】

第16の、任意的なインスタンス化により、第13乃至第15いずれか1つのインスタンス化に従った方法が提供され得る。本方法は、前記コードによって返される結果が真であることを条件として、前記第1当事者のためのサービスをトリガすること、を含む、

【0297】

例えば、サービスは、第1当事者によって委託されているか、第1当事者に代わって履行されるか、または、第1当事者の利益のために履行され得る。サービスは、コンピュータ化されたサービスであってよく、前記トリガは、サービスを自動的にトリガすることを含み得る。

【0298】

一時的鍵を第2当事者(「ボブ」)に与えることによって、これは、第1当事者(「アリス」)が、ボブに彼女に代わってサービスの署名をさせることを可能にするが、ボブが一時的鍵kをノードに開示したり、ブロックチェーン上で公開したりする必要はない。さらに、プロセスは特定の秘密鍵(または対応する公開鍵)と結び付いていないため、これは、アリスが一時的鍵のコピーを第三者(「チャーリー」)に渡すことができることを意味し、ボブとチャーリーのいずれかがサービスに成功裏に署名することができる。これは、r部がチャレンジの基礎として使用され、かつ、r部が特定の身元(identity)にマップされていないので可能である。

【0299】

第17の、任意的なインスタンス化により、第13乃至第16いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記第2トランザクションで受信した前記情報は、前記第2当事者のさらなる秘密鍵を使用して前記第2トランザクションの一部に署名する前記第2当事者のさらなる暗号署名を含み、前記さらなる秘密鍵は、さらなる公開鍵に対応している。

【0300】

さらなる署名は、ECC署名、または他のタイプ、例えば、RSA署名であり得る。

【0301】

第18の、任意的なインスタンス化により、第17のインスタンス化に従った方法が提供され得る。ここでは、前記さらなる公開鍵に基づいて、前記第1当事者及び／又は第三者が、前記第2当事者の識別情報を探索することを可能にするマッピングが利用可能である。

【0302】

例えば、身元は、第2当事者の個人名、会社名、ユーザ名、またはネットワークアドレスであってよい。第三者は、例えば、前述のサービスの提供者であってよい。

【0303】

第19の、任意的なインスタンス化により、第17または第18のインスタンス化に従った方法が提供され得る。ここで、前記コードは、前記さらなる公開鍵を使用して、さらなる暗号署名を検証し、かつ、前記さらなる暗号署名が検証されることを条件として、真の結果を返すように構成されている。

【0304】

10

20

30

40

50

第20の、任意的なインスタンス化により、第17乃至第19いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記第2トランザクションで受信した前記情報は、さらに、前記第1当事者の秘密鍵を使用して前記第2トランザクションの一部に署名する、前記第1当事者の暗号署名、を含む。

【0305】

実施形態において、本方法は、第1当事者の秘密鍵に対応する公開鍵を取得することを含み得る。ここで、前記コードは、第2当事者の暗号署名を検証し、第1当事者の暗号署名がさらなる条件で真の結果を返すように構成されている。

【0306】

実施形態において、マッピングは、第2当事者及び／又は第三者が、第1当事者の公開鍵に基づいて第1当事者の身元を探索することを可能にするために利用可能であり得る。例えば、第1当事者の身元は、第1当事者の個人名、会社名、ユーザ名、または、ネットワークアドレスであり得る。

【0307】

第21の、任意的なインスタンス化により、第13乃至第20いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記第2トランザクションで受信した情報は、前記第1ECDSA署名とは異なるr部の値を有しているが、前記第1ECDSA署名と同じ第1秘密鍵を使用している、追加的ECDSA署名を含み、

前記コードは、前記第1公開鍵を使用して使って前記追加的ECDSA署名を検証し、かつ、前記追加的ECDSA署名が検証されることを条件に、真の結果を返すように構成されている。

【0308】

実施形態において、追加的ECDSA署名は、第1ECDSA署名とは異なる第2トランザクションの部分に署名し得る。

【0309】

第22の、任意的なインスタンス化により、第1乃至第21いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記トランザクションそれぞれは、1つ以上の入力および1つ以上の出力を含むデータ構造を備え、各出力はロッキングスクリプトを含み、かつ、各入力は、アンロッキングスクリプトおよび別のトランザクションの出力に対するポインタを含み、前記コードは、前記第1トランザクションのロッキングスクリプトによって構成され、前記第2トランザクションで受信した前記情報は、前記第2トランザクションの入力における前記アンロッキングスクリプトによって構成され、かつ、前記第2トランザクションの前記入力における前記ポインタは、前記第1トランザクションの前記出力を指し示す。前記方法は、前記コードが真の結果を返すことを少なくとも条件として、前記トランザクションを検証するステップを含み、かつ、前記検証に応答して、検証ノードによる1つ以上のブロックへのマイニングのためのトランザクションのプールに前記第2トランザクションを含むステップ、及び／又は、前記第2トランザクションをブロックチェーンネットワークの少なくとも1つの他のノードに転送ステップ、のうち少なくとも1つを含む。

【0310】

第1および第2トランザクションを含む複数のトランザクションそれぞれに対して、ネットワークのノードの少なくともいくつかは、トランザクションが有効である条件で各トランザクションを伝播するように構成されており、かつ、ネットワークのノードの少なくともいくつかは、トランザクションが有効である条件で、各トランザクションをブロックチェーンの少なくとも一部のコピーに記録するように構成されている。第2トランザクションの有効性は、少なくとも真の結果を返すコードを条件とする。

【0311】

第23、任意的なインスタンス化により、第1乃至第22いずれか1つのインスタンス化に従った方法が提供され得る。ここで、前記トランザクションは、アカウントベースのモデルに従って構成されており、前記コードは、前記第1トランザクションに含まれるス

10

20

30

40

50

マートコントラクトによって構成されている。

【0312】

実施形態において、いずれのモデルにおいても、検証ノードは、マイニングノード、転送ノード、及び／又は、ブロックチェーンの少なくとも一部を保管するストレージノード(例えば、ブロックチェーンの全コピーを保管する全コピーストレージノード)であってよい。実施形態において、前記第1トランザクションの前記取得は、第1当事者、例えば前記第1当事者、から前記第1トランザクションの少なくとも一部を受信することを含んでよい。実施形態において、前記第1トランザクションの取得は、第1当事者から第1トランザクションを受信することを含んでよい。代替的に、実施形態において、前記第1トランザクションの取得は、検証ノードで第1トランザクションを策定することを含んでもよい。実施形態において、前記第1トランザクションの前記取得は、前記第1当事者からr部の参照インスタンスを少なくとも受信し、前記ノードのうちの1つで前記第1トランザクションに策定することを含んでよい。実施形態において、前記第1トランザクションの取得は、検証ノードにおいてr部を生成することを含む第1トランザクションを策定することを含んでよい。

10

【0313】

実施形態において、前記第2トランザクションの前記受信は、第2当事者、例えば前記第2当事者、から第2トランザクションを受信することを含んでもよい。実施形態において、第2トランザクションは、少なくとも部分的に第2当事者によって生成されたものである。実施形態において、第2トランザクションは、第2当事者によって生成されたものである。実施形態において、前記第2トランザクションの受信は、直接的に、もしくは、第1当事者または第三者を介して、第2トランザクションを第2当事者から受領することを含んでよい。実施形態において、第2トランザクションは、第2当事者によって第三者に提供されている、少なくとも第1ECDSA署名のs部(および、実施形態では、第1ECDSA署名及び／又はデータ要素のr部の提出されたインスタンス)に基づいて、第三者によって生成されたものである。

20

【0314】

ここにおいて開示される教示の第24インスタンス化に従って、コンピュータ読取り可能記憶装置において具体化され、かつ、ネットワークのノード上で実行されると、第1乃至第22いずれか1つのインスタンス化に従った方法を実行するように構成されている、コンピュータプログラムが提供され得る。

30

【0315】

ここにおいて開示される教示の第25インスタンス化に従って、ネットワークのノードが提供される。本ノードは、1つ以上のメモリユニットを含むメモリと、1つ以上の処理ユニットを含む処理装置と、を備える。ここで、前記メモリは、前記処理装置上で実行するように構成されたコードを保管しており、前記コードは、前記処理装置において第1乃至第23いずれか1つのインスタンス化に従った方法を実施するように構成されている、

【0316】

ここにおいて開示される教示の第26インスタンス化に従って、第2当事者のコンピュータ装置において、コンピュータに実装される方法が提供される。本方法は、実行可能なコードを含む第1トランザクションを監視するステップであり、前記コードは、 $H_{pow}(f(r, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、 r はECDSA署名の r 部であり、 d はノンスであり、 H_{pow} はハッシュ関数であり、 f は r および d を組み合わせた関数であり、前記 r 部は第1当事者によって指定される、ステップと、一時的鍵に基づいて、前記 r 部を生成するステップと、 $H_{pow}(f(r, d))$ が前記事前決定された条件を満たすように、前記ノンス d の値を探索するステップと、前記第1トランザクションにリンクされた第2トランザクションを策定するステップであり、前記第2トランザクションは、第1ECDSA署名の少なくとも r 部および s 部と、さらに、前記ノンス d とを含む、情報を含む、ステップと、ブロックチェーンに記録するために、ブロックチ

40

50

チェーンネットワークを介して伝搬される、前記第2トランザクションを送信するステップと、を含む。

【0317】

実施形態において、第2当事者の装置で実行される方法は、さらに、ここにおいて開示されたインスタンス化または他の特徴のいずれかに対応すること含んでよい。

【0318】

ここにおいて開示される教示の第27インスタンス化に従って、ブロックチェーンに記録するためのトランザクションのセットが提供される。前記セットは、コンピュータで読取り可能な媒体またはメディア上で具体化され、実行可能なコードを含む第1トランザクションと、第1ECDSA署名のr部およびs部の少なくとも提出されたインスタンスと、さらに、ノンスとを含む、

第2トランザクションと、を含む。前記コードは、 $H_{PoW}(f(r, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、rは前記r部の提出されたインスタンスであり、dはノンスであり、 H_{PoW} はハッシュ関数であり、fはrおよびdを組み合わせた関数である。

【0319】

実施形態において、第1トランザクションおよび第2トランザクションは、さらに、ここにおいて開示される任意のインスタンス化または他の特徴に従って構成されてよい。

【0320】

ここにおいて開示される教示の第27インスタンス化に従って、ブロックチェーンネットワークのノード検証において、コンピュータに実装される方法が提供される。本方法は、実行可能コードを含む第1トランザクションを取得するステップと、少なくとも第1部およびノンスを有する情報を含む第2トランザクションを受信するステップと、第1トランザクションからのコードを実行するステップと、を含む。前記コードは、 $H_{PoW}(f(q, d))$ が前記コードで定義されている事前決定された条件を満たしていることを検証し、かつ、前記条件で真の結果を返すように構成されており、ここで、qは第1部分であり、dはノンスであり、 H_{PoW} はハッシュ関数であり、fはqおよびdを組み合わせた関数である、

【0321】

rパズルバースのプルーフ・オブ・ワークに関連してここにおいて開示される任意の特徴は、任意のそうした「第2層（“layer-s”）」に一般化することができる。

【0322】

第1部分qは、データの任意の所定の部分であり得る。それは、ノンスを除く第2トランザクションのいずれかの部分であり得る。

【0323】

実施形態において、rパズルのPoWパズルに関連して開示される任意の特徴は、このより一般的なPoWパズルに関連して適用され得る。例えば、実施形態におけるインスタンスについて、fは連結 $q || d$ である。実施形態において、上記の事前設定された条件は、 $H_{puz}(q || d)$ が事前設定された目標値未満であること、または、少なくとも事前設定された最小数の先行するゼロを有することであってよい。

【0324】

実施形態において、第2トランザクションで受信した情報は、さらに、暗号署名を含んでよい。本方法は、暗号署名が公開鍵に対応する秘密鍵に基づいてメッセージに署名する公開鍵を取得することを含んでよく、ここで、メッセージは第2トランザクションの一部である。本方法は、第2トランザクションで受信した暗号署名を、公開鍵とメッセージに基づいて検証するために検証機能を適用することを含んでよく、コードは、暗号署名の前記検証のさらなる条件に基づいて、真の結果を返すように構成されている。実施形態において、公開鍵は、証明者の識別子にマッピングされてよい。このことは、第1当事者が証明者の身元を調べるために使用することができる。

10

20

30

40

50

【 0 3 2 5 】

さらなるインスタンス化に従って、第 2 8 のインスタンス化に対応して、第 2 当事者の装置で実行される、プログラム、方法、または、トランザクションのセットが提供され得る。

【 0 3 2 6 】

開示された技術の他の変形例またはユースケースは、一旦ここにおいて開示されると、当業者に対して明らかになり得る。本開示の範囲は、記載された実施形態によって限定されるものではなく、添付の請求項によってのみ限定される。

10

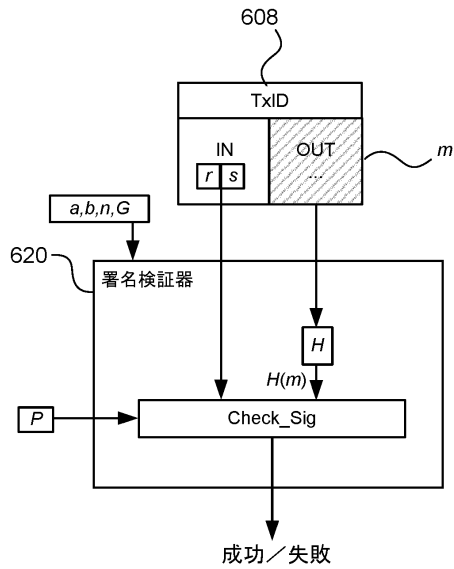
20

30

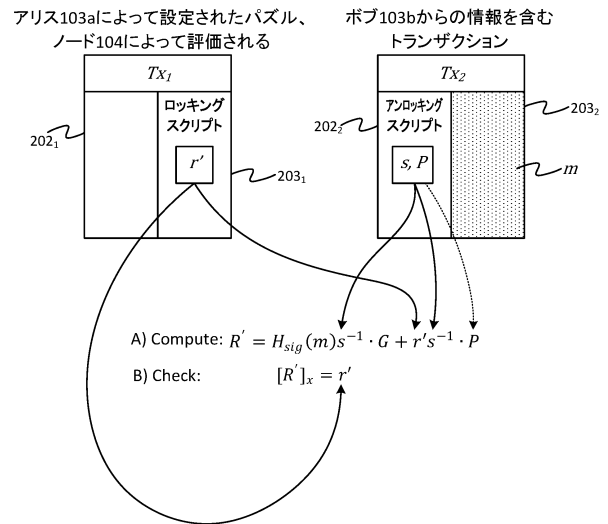
40

50

【図 6 D】

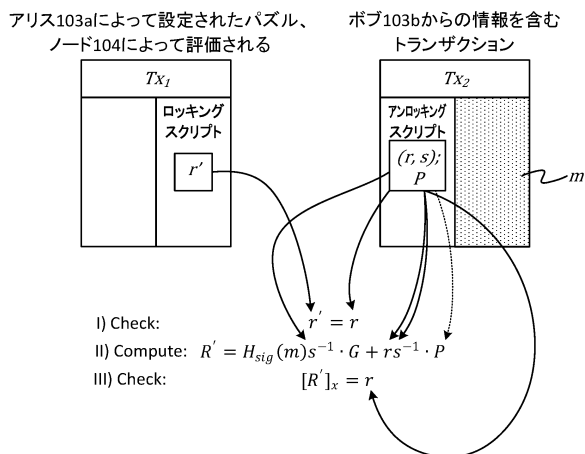


【図 7】

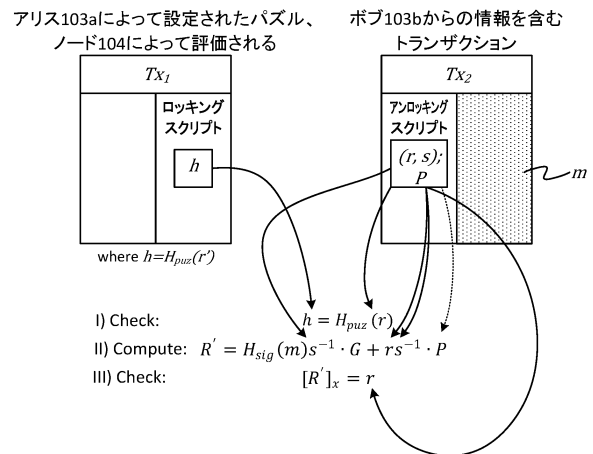


10

【図 8】



【図 9】



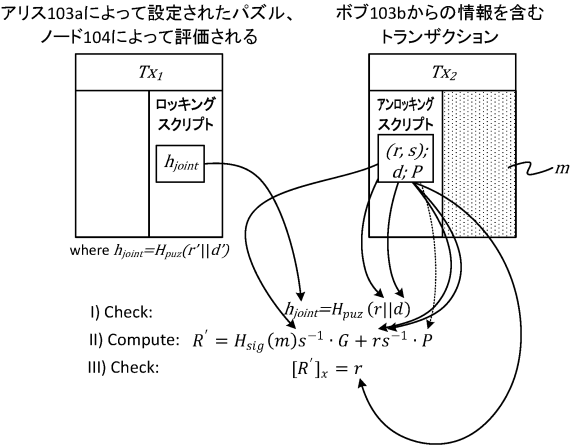
20

30

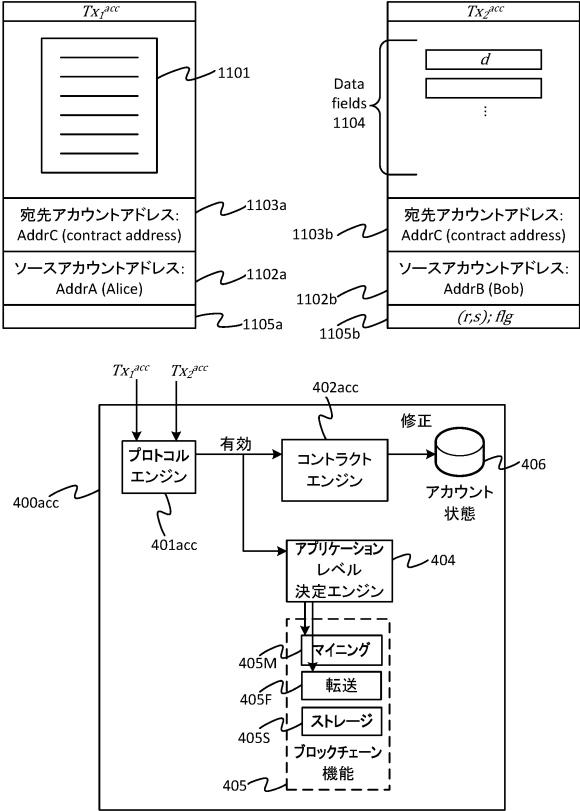
40

50

【図 10】



【図 11】



10

20

【図 12】



【図 13】

スタック	スクリプト	説明
empty	$\langle r_b \rangle (sig_b) OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_CHECKSIG OP_SPILT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	scriptSig and scriptPubkey are combined
$\langle r_b \rangle (sig_b)$	$OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Unlocking script constants pushed to stack
$\langle r_b \rangle (sig_b) (sig_b) OP_3$	$OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Tos (sig) duplicated and constant (3) pushed to stack
$\langle r_b \rangle (sig_b) (sig_b) [3] (sig_b) [3; 1]$	$OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Tos (sig) split at the third byte, right before len(r)
$\langle r_b \rangle (sig_b) (sig_b) [3; 1] OP_1$	$OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Second to Tos item removed and constant (1) pushed to stack
$\langle r_b \rangle (sig_b) (sig_b) [4; 3] (sig_b) [3; 4]$	$OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Tos split 1 byte in (at byte 4 of sig) and the top two items on the stack are swapped
$\langle r_b \rangle (sig_b) (sig_b) [4; 36/37] (sig_b) [36/37; 1]$	$OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	The second to tos (sig starting at index 4) is split at len(r) = (sig_b [3; 4]) bytes in which will be either 36 or 37 since r will either be 32 or 33 bytes
$\langle r_b \rangle (sig_b) (sig_b) [4; 36/37] (r)$	$OP_EQUALVERIFY OP_SWAP OP_CHECKSIG$	Drop tos (rest of sig) and push (r) to stack
$(sig_b) (r_b)$	$OP_CHECKSIG$	Check if $(sig_b [4; 36/37]) = (r)$
True	empty	Signature is checked for top two stack items

30

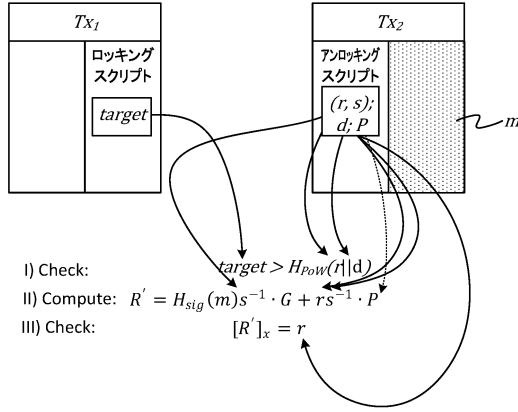
40

50

【図 1 4 A】

アリス103aによって設定されたパズル、
ノード104によって評価される

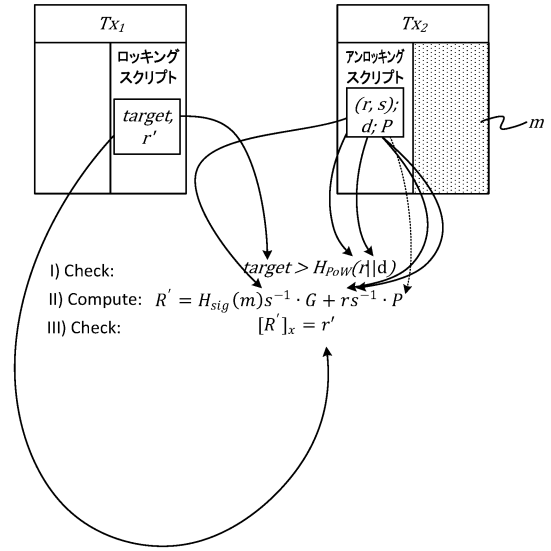
ボブ103bからの情報を含む
トランザクション



【図 1 4 B】

アリス103aによって設定されたパズル、
ノード104によって評価される

ボブ103bからの情報を含む
トランザクション



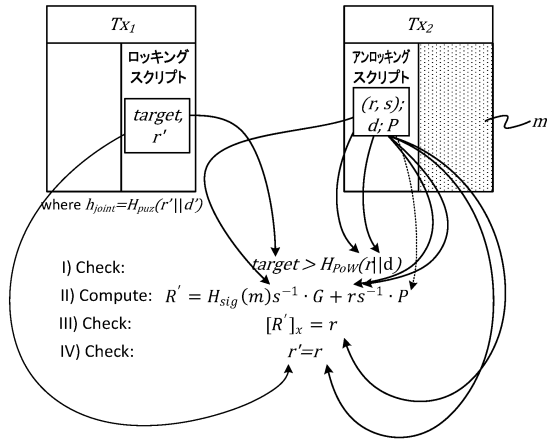
10

20

【図 1 4 C】

アリス103aによって設定されたパズル、
ノード104によって評価される

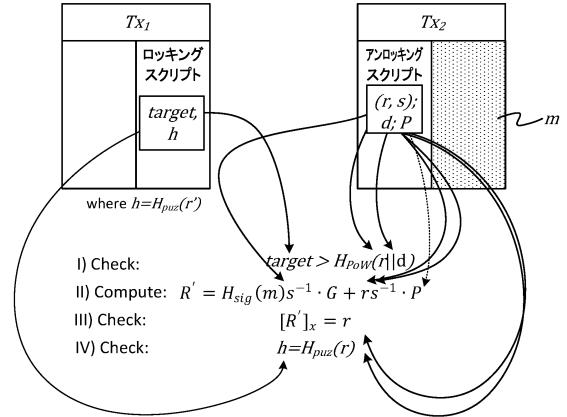
ボブ103bからの情報を含む
トランザクション



【図 1 4 D】

アリス103aによって設定されたパズル、
ノード104によって評価される

ボブ103bからの情報を含む
トランザクション



30

40

50

フロントページの続き

- (72)発明者 ジャーン, ウエイ
 イギリス ダブリュー1ダブリュー 8エーピー ロンドン マーケット プレイス 30 エヌチェー
 ン ホールディングズ リミテッド 内
- (72)発明者 ドワロン, ブロック
 イギリス ダブリュー1ダブリュー 8エーピー ロンドン マーケット プレイス 30 エヌチェー
 ン ホールディングズ リミテッド 内
- (72)発明者 ライト, クレイグ
 イギリス ダブリュー1ダブリュー 8エーピー ロンドン マーケット プレイス 30 エヌチェー
 ン ホールディングズ リミテッド 内
- 審査官 中里 裕正
- (56)参考文献 米国特許出願公開第2016/0028552 (US, A1)
 国際公開第2018/215876 (WO, A1)
 BALDIMTSI, F. et al., Indistinguishable Proofs of Work or Knowledge, Lecture Notes in Co
 mputer Science, Vol.10032, 2016年, pp.902-933
- (58)調査した分野 (Int.Cl., DB名)
 H04L 9/32
 JSTPlus/JMEDPlus/JST7580 (JDreamIII)
 IEEE Xplore