

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7551658号
(P7551658)

(45)発行日 令和6年9月17日(2024.9.17)

(24)登録日 令和6年9月6日(2024.9.6)

(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 C

H O 4 L 9/32 2 0 0 Z

請求項の数 32 (全71頁)

(21)出願番号	特願2021-569310(P2021-569310)	(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2022-533751(P2022-533751 A)		スイス・6 3 0 0・ツーク・グラーフエ
(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/IB2020/053807		弁理士 伊東 忠重
(87)国際公開番号	WO2020/240295	(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23日(2023.3.23)	(74)代理人	100135079
(31)優先権主張番号	1907395.6		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, ジェド
(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】 知識証明

(57)【特許請求の範囲】

【請求項1】

楕円曲線デジタル署名アルゴリズム（ECDSA）検証関数に基づき知識証明を実行する、コンピュータにより実施される方法であって、前記方法は、ブロックチェーンネットワークの検証ノードにおいて、

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であって、前記コードは第1ECDSA署名のr部分の参照インスタンスを指定する要素を含む、ステップと、

前記第1ECDSA署名の少なくともs部分を含む情報を含む第2トランザクションを受信し、第1公開鍵を取得するステップであって、前記第1ECDSA署名は、前記第1公開鍵に対応する第1秘密鍵に基づき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、

前記第1トランザクションからの前記コードを実行するステップであって、前記コードは、どの秘密鍵が前記第1秘密鍵として使用されたかに関係なく、以下：

前記第2トランザクション内で受信されたメッセージ及び前記取得された第1公開鍵が与えられると、前記ECDSA検証関数が、前記第1ECDSA署名に適用されるとき、前記第2トランザクション内で受信された前記s部分が前記第1トランザクションにより指定された前記r部分の参照インスタンスに対して有効なs部分であることを検証する、

ことを条件として、真の結果を返すよう構成される、ステップと、を含む方法。

【請求項2】

前記要素は、前記第 1 ECDSA 署名の前記 r 部分の参照インスタンスである、請求項 1 に記載の方法。

【請求項 3】

前記 ECDSA 検証関数は、以下を実行し：

【数 3 7】

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

$[R']_x = r'$ であることのチェック、

10

ここで、 r' は前記第 1 ECDSA 署名の前記 r 部分の参照インスタンスであり、 s は前記第 1 ECDSA 署名の前記 s 部分であり、 P は前記第 1 公開鍵であり、 m は前記第 1 ECDSA 署名により署名された前記第 2 トランザクションの一部であり、 H_{sig} は前記第 1 ECDSA 署名を生成する際に m をハッシングするために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示し、

前記コードは、前記 ECDSA 検証関数により実行された前記チェックが真であることを条件として、真の結果を返し、その他の場合に偽の結果を返すよう構成される、請求項 2 に記載の方法。

【請求項 4】

前記第 2 トランザクション内で受信された前記情報は、前記第 1 ECDSA 署名の前記 r 部分の提出されたインスタンスを更に含み、前記コードは、前記 r 部分の前記提出されたインスタンスが前記参照インスタンスに等しいことをチェックするよう構成される、請求項 1 又は 2 に記載の方法。

20

【請求項 5】

前記 ECDSA 検証関数は、前記第 2 トランザクション内で受信された前記 s 部分が前記 r 部分の前記提出されたインスタンスに対して有効な s 部分であることをチェックし、両方のチェックを一緒に行うことにより、前記有効な s 部分であることを検証する、請求項 4 に記載の方法。

【請求項 6】

前記コードは、以下を実行し：

【数 3 8】

$r' = r$ であることをチェックする第 1 チェック、

30

前記 ECDSA 検証関数は、以下を実行し：

【数 3 9】

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

$[R']_x = r$ であることをチェックする第 2 チェック、

40

ここで、 r は前記第 1 ECDSA 署名の前記 r 部分の提出されたインスタンスであり、 r' は前記第 1 ECDSA 署名の前記 r 部分の前記参照インスタンスであり、 s は前記第 1 ECDSA 署名の前記 s 部分であり、 P は前記第 1 公開鍵であり、 m は前記第 1 ECDSA 署名により署名された前記第 2 トランザクションの一部であり、 H_{sig} は前記第 1 ECDSA 署名を生成する際に m をハッシングするために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示し、

前記コードは、前記第 1 チェック及び前記第 2 チェックの両方が真であることを条件として、真の結果を返し、その他の場合に偽の結果を返すよう構成される、請求項 2 に従属する請求項 5 に記載の方法。

50

【請求項 7】

前記要素は、前記第 1 ECDSA 署名の前記 r 部分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同じ変換を実行することにより、前記提出されたインスタンスが前記参照インスタンスに等しいことの前記チェックを実行するよう構成される、請求項 4 又は 5 に記載の方法。

【請求項 8】

前記要素は、ハッシュ値であり、前記第 1 ECDSA 署名の前記 r 部分の参照インスタンスを含むコンポーネントのハッシュである、請求項 7 に記載の方法。

【請求項 9】

前記コンポーネントは、前記第 1 ECDSA 署名の前記 r 部分の参照インスタンス r' であり、前記コードは、以下を実行し：

10

【数 4 0】

$h = H_{puz}(r)$ であることをチェックする第 1 チェック、

前記 ECDSA 検証関数は、以下を実行し：

【数 4 1】

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

$[R']_x = r$ であることをチェックする第 2 チェック、

20

ここで、 r は前記第 1 ECDSA 署名の前記 r 部分の前記提出されたインスタンスであり、 s は前記第 1 ECDSA 署名の前記 s 部分であり、 h はハッシュ値であり、 H_{puz} は r' をハッシュングして h を生成するために使用されたハッシュ関数であり、 P は前記第 1 公開鍵であり、 m は前記第 1 ECDSA 署名により署名された前記第 2 トランザクションの一部であり、 H_{sig} は前記第 1 ECDSA 署名を生成する際に m をハッシュングするために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示し、

前記コードは、前記第 1 チェック及び前記第 2 チェックの両方が真であることを条件として、真の結果を返し、その他の場合に偽の結果を返すよう構成される、請求項 5 に従属する請求項 8 に記載の方法。

30

【請求項 10】

前記コンポーネントは、前記第 1 ECDSA 署名の前記 r 部分の参照インスタンス r' とデータ値の参照インスタンス d' との組合せ $f(r', d')$ であり、前記第 2 トランザクション内で受信される前記情報は、前記第 1 ECDSA 署名の前記 r 部分の提出されたインスタンス r 及び前記データ値の提出されたインスタンス d を更に含み、前記コードは、以下を実行するよう構成され：

【数 4 2】

$h_{joint} = H_{puz}(f(r, d))$ であることをチェックする第 1 チェック、

40

前記 ECDSA 検証関数は、以下を実行し：

【数 4 3】

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

$[R']_x = r$ であることをチェックする第 2 チェック、

ここで、 s は前記第 1 ECDSA 署名の前記 s 部分であり、 h_{joint} はハッシュ値であり、 H_{puz}

50

は $f(r', d')$ をハッシングして h_{nt} を生成するために使用されたハッシュ関数であり、 P は前記第 1 公開鍵であり、 m は前記第 1 ECDSA 署名により署名された前記第 2 トランザクションの一部であり、 H_{sig} は前記第 1 ECDSA 署名を生成する際に m をハッシングするために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示し、

前記コードは、前記第 1 チェック及び前記第 2 チェックの両方が真であることを条件として、真の結果を返し、その他の場合に偽の結果を返すよう構成される、請求項 5 に従属する請求項 8 に記載の方法。

【請求項 11】

前記組合せ f は連結であり、

【数 44】

$f(r', d')$ は $r' || d'$ であり、 $H_{\text{puz}}(f(r, d))$ は $H_{\text{puz}}(r || d)$ である、

請求項 10 に記載の方法。

【請求項 12】

前記データ値は、目標時点を指定するロックタイム値を含み、前記コードは、前記目標時点に達することを更なる条件として、真の結果を返すよう構成される、請求項 10 又は 11 に記載の方法。

【請求項 13】

前記第 1 公開鍵を取得する前記ステップは、前記第 2 トランザクション内の前記情報の部分として前記第 1 公開鍵を受信するステップを含む、請求項 1 ～ 12 のいずれかに記載の方法。

【請求項 14】

前記コードは、第 2 ECDSA 署名の r 部分の参照インスタンスを更に指定し、

前記コードは、複数の代替条件のうちのいずれか 1 つが満たされる場合に真の結果を返すよう構成され、前記代替条件は、

i) 前記第 1 ECDSA 署名により署名された前記第 2 トランザクションの一部及び前記第 1 公開鍵が与えられると、前記第 1 ECDSA 署名の前記 s 部分が前記第 1 ECDSA 署名の前記 r 部分の参照インスタンスに対して有効な s 部分であることを検証することと、

ii) 前記第 2 ECDSA 署名により署名された前記第 2 トランザクションの一部及び前記第 2 ECDSA 署名を生成するために使用された第 2 秘密鍵に対応する第 2 公開鍵が与えられると、前記第 2 ECDSA 署名の前記 s 部分が前記第 2 ECDSA 署名の前記 r 部分の参照インスタンスに対して有効な s 部分であることを検証することと、

を含む、請求項 1 ～ 13 のいずれかに記載の方法。

【請求項 15】

前記第 1 ECDSA 署名の少なくとも前記 s 部分は、第 2 パーティにより、第 1 パーティにより前記第 2 パーティに与えられた若しくは逆も同様である一時鍵、及び前記第 2 パーティの秘密鍵である第 1 秘密鍵を用いて生成される、請求項 1 ～ 14 のいずれかに記載の方法。

【請求項 16】

【数 45】

10

20

30

40

50

$$P = V \cdot G,$$

$$k \in [1, n - 1],$$

$$R = k \cdot G,$$

$$r = [R]_x, \text{ and}$$

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

10

ここで、Pは前記第1公開鍵であり、Vは前記第1秘密鍵であり、kは前記一時鍵であり、nは素数モジュラスであり、Gは楕円生成元であり、mは前記第1ECDSA署名により署名された前記第2トランザクションの一部であり、H_{sig}は前記第1ECDSA署名を生成する際にmをハッシング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座標を示し、「・」は楕円曲線スカラー乗算を示す、請求項15に記載の方法。

【請求項17】

前記第1ECDSA署名の前記r部分の提出されたインスタンスrも前記第2パーティにより生成される、少なくとも請求項4に従属する請求項15又は16に記載の方法。

【請求項18】

前記データ値の提出されたインスタンスdも前記第2パーティにより生成される、少なくとも請求項10、11、又は12に従属する請求項17に記載の方法。

20

【請求項19】

前記第2トランザクションを受信する前記ステップは、前記第2パーティから前記第2トランザクションを受信するステップを含む、請求項15～18のいずれかに記載の方法。

【請求項20】

前記コードにより返された結果が真であることを条件として、前記第1パーティのためにサービスをトリガするステップ、を含む請求項15～19のいずれかに記載の方法。

【請求項21】

前記第2トランザクション内で受信された前記情報は、前記第2パーティの更なる秘密鍵を用いて前記第2トランザクションの部分に署名する、前記第2パーティの更なる暗号署名を含み、前記更なる秘密鍵は更なる公開鍵に対応する、請求項15～20のいずれかに記載の方法。

30

【請求項22】

前記第1パーティ及び／又は第三者が前記更なる公開鍵に基づき前記第2パーティのアイデンティティを検索できるようにするマッピングが利用可能である、請求項21に記載の方法。

【請求項23】

前記コードは、前記更なる公開鍵を用いて前記更なる暗号署名を検証し、前記更なる暗号鍵が検証されたことを更なる条件として真の値を返すよう構成される、請求項21又は22に記載の方法。

40

【請求項24】

前記第2トランザクション内で受信された前記情報は、前記第1パーティの秘密鍵を用いて前記第2トランザクションの部分に署名する、前記第1パーティの暗号署名を含む、請求項15～23のいずれかに記載の方法。

【請求項25】

前記第2トランザクション内で受信された前記情報は、前記第1ECDSA署名と同じ第1秘密鍵を用いるが、前記r部分の異なる値を有する追加ECDSA署名を含み、

前記コードは、前記第1公開鍵を用いて前記追加ECDSA署名を検証し、前記追加ECDSA署名が検証されたことを更なる条件として真の結果を返すよう構成される、請求項15～24のいずれかに記載の方法。

50

【請求項 26】

前記第2トランザクションは、デジタルアセットの量を指定し、前記量は、少なくとも前記コードが真の結果を返すことを条件としてアンロックされる、請求項1～25のいずれかに記載の方法。

【請求項 27】

トランザクションの各々は、1つ以上のインプットと1つ以上のアウトプットを含むデータ構造を有し、各アウトプットはロックスクリプトを含み、各インプットはアンロックスクリプトと別のトランザクションのアウトプットへのポインタを含み、

前記コードは、前記第1トランザクションの前記ロックスクリプトに含まれ、前記第2トランザクション内で受信される前記情報は、前記第2トランザクションのインプット内の前記アンロックスクリプトに含まれ、前記第2トランザクションの前記インプット内のポインタは、前記第1トランザクションの前記アウトプットを指し、

前記方法は、少なくとも前記コードが真の結果を返すことを条件として、前記トランザクションを検証するステップと、

前記検証に応答して、以下：

前記検証ノードにより1つ以上のブロックへとマイニングするために、前記第2トランザクションをトランザクションプールに含めるステップ、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ノードのうちの少なくとも1つの他のノードへ転送するステップ、

のうちの少なくとも1つを含む、ステップと、

を含む請求項1～26のいずれかに記載の方法。

【請求項 28】

前記第2トランザクションのアウトプットのうちの1つは、更新された第1ECDSA署名のr部分の参照インスタンスを指定するコードの更新されたバージョンを含み、前記方法は、

前記更新された第1ECDSA署名の少なくともs部分を含む情報を含むトランザクションのセットのうちの第3トランザクションを受信し、更新された第1公開鍵を取得するステップであって、前記更新された第1ECDSA署名は、前記更新された公開鍵に対応する更新された第1秘密鍵に基づき、前記第3トランザクションの部分に署名する、ステップと、

前記第3トランザクションから前記コードの更新されたバージョンを実行するステップであって、前記更新されたコードは、前記第2トランザクションの署名済み部分及び前記取得された第1公開鍵が与えられると、どの秘密鍵が前記更新された第1秘密鍵として使用されたかに関係なく、前記更新された第1ECDSA署名に適用された前記更新された第1ECDSAが、前記更新されたs部分が前記第2トランザクション内で指定された前記r部分の参照インスタンスに対する有効なs部分であると検証することを条件として、真の値を返すよう構成される、ステップと、

を含む請求項27に記載の方法。

【請求項 29】

トランザクションは、アカウントに基づくモデルに従い構成され、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含まれる、請求項1～26のいずれかに記載の方法。

【請求項 30】

ネットワークのノードにより実行されると、前記ノードに請求項1～29のいずれかに記載の方法を実行させる、コンピュータプログラム。

【請求項 31】

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機器と、

を含み、

前記メモリはコードを格納し、前記コードは、前記処理機器上で実行されると、前記処

10

20

30

40

50

理機器に請求項 1 ～ 29 のいずれかに記載の方法を実行させるよう構成される、ノード。

【請求項 32】

楕円曲線デジタル署名アルゴリズム (ECDSA) 検証関数に基づき知識証明を設定する、コンピュータにより実施される方法であって、前記方法は、第 1 パーティのコンピュータ機器を用いて、

第 2 パーティへ一時鍵を送信するか、又は前記第 2 パーティから前記一時鍵を受信するステップと、

前記一時鍵に基づき、第 1 ECDSA 署名の r 部分の参照インスタンスを決定するステップと、

実行可能コードを含む第 1 トランザクションを形成するステップであって、前記コードは前記第 1 ECDSA 署名の前記 r 部分の参照インスタンスを指定する要素を含む、ステップと、

を含み、

前記コードは、第 2 トランザクションからの前記第 1 ECDSA 署名の少なくとも s 部分、及び第 1 公開鍵、に作用するよう構成され、前記第 1 ECDSA 署名は、前記第 1 公開鍵に対応する第 1 秘密鍵に基づき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り、

前記コードは、どの秘密鍵が前記第 1 秘密鍵として使用されたかに関係なく、以下：

前記第 2 トランザクション内で受信されたメッセージ及び前記第 1 公開鍵が与えられると、前記 ECDSA 検証関数が、前記第 1 ECDSA 署名に適用されるとき、前記第 2 トランザクション内で受信された前記 s 部分が前記第 1 トランザクションにより指定された前記 r 部分の参照インスタンスに対して有効な s 部分であることを検証する、

ことを条件として、真の結果を返すよう構成される、方法。

【発明の詳細な説明】

【技術分野】

【0001】

本開示は、ブロックチェーンに記録するためのトランザクションのセットにより実装され得る知識証明の形式に関する。

【背景技術】

【0002】

ブロックチェーンとは、分散型データ構造の形式を指し、ブロックチェーンの複製のコピーは、ピアツーピア (peer-to-peer (P2P)) ネットワーク内の複数のノードのそれぞれにおいて維持される。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、シーケンスの中の先行するトランザクションを指してよい。トランザクションは、新しいブロックに含まれるようネットワークへ提出され得る。新しいブロックは、「マイニング」として知られる処理により生成される。「マイニング」は、複数のマイニングノードの各々が「proof-of-work」を実行するために競争する、つまりブロックに含まれることを待っている保留中のトランザクションのプールに基づき、暗号パズルを解くことを含む。

【0003】

従来、ブロックチェーン内のトランザクションは、デジタルアセット、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、ブロックチェーンの上に追加の機能を積み重ねるために、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アウトプットに追加のユーザーデータを格納することを可能にしてよい。現代のブロックチェーンは、単一トランザクション内に格納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ドキュメント (electronic document)、或いはオーディオ若しくはビデオデータを格納するために使用され得る。

【0004】

ネットワーク内の各ノードは、転送、マイニング、及び記憶のうちの任意の 1、2、又

10

20

30

40

50

は3つ全部の役割を有することができる。転送ノードは、ネットワークのノードを通じてトランザクションを伝播させる。マイニングノードは、トランザクションのマイニングを実行してブロックにする。記憶ノードは、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彼ら自身のコピーを格納する。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、パーティは、該トランザクションを、伝播させるようにネットワークのノードのうちの1つへ送信する。トランザクションを受信したマイニングノードは、トランザクションをマイニングして新しいブロックにするよう競争してよい。各ノードは、トランザクションが有効であるための1つ以上の条件を含む同じノードプロトコルに関するよう構成される。無効なトランザクションは、伝播されず、マイニングされてブロックにされることもない。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、(任意のユーザデータを含む) トランザクションは、従って、不変の公開レコードとしてP2Pネットワークの各ノードに格納されたままである。

【0005】

最新のブロックを生成するためにproof-of-workパズルを解くことに成功したマイナーは、標準的に、デジタルアセットの新しい額を生成する「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により報酬を受ける。proof-of-workは、ブロックをマイニングするために膨大な量の計算リソースを必要とするので、及び二重支払いの企てを含むブロックは他のノードにより受け入れられない可能性があるので、マイナーが、彼らの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りシステムを騙さないことを奨励する。

【0006】

「アウトプットに基づく」モデル (UTXOに基づくモデルと呼ばれることもある) では、所与のトランザクションのデータ構造は、1つ以上のインプット及び1つ以上のアウトプットを含む。任意の使用可能アウトプットは、時にUTXO (「unspent transaction output (未使用トランザクションアウトプット)」) と呼ばれる、デジタルアセットの額を指定する要素を含む。アウトプットは、アウトプットを償還 (redeem) するための条件を指定するロックスクリプトを更に含んでよい。各インプットは、先行するトランザクション内のそのようなアウトプットへのポインタを含み、ポイントされたアウトプットのロックスクリプトをアンロックするためのアンロックスクリプトを更に含んでよい。従って、トランザクションのペアを考えると、それらを、第1トランザクション及び第2トランザクションと呼ぶ。第1トランザクションは、デジタルアセットの額を指定する、及びアウトプットをアンロックする1つ以上の条件を定義するロックスクリプトを含む、少なくとも1つのアウトプットを含む。第2トランザクションは、第1トランザクションのアウトプットへのポインタと、第1トランザクションのアウトプットをアンロックするためのアンロックスクリプトと、を含む少なくとも1つのインプットを含む。

【0007】

このようなモデルでは、第2トランザクションが伝播されてブロックチェーンに記録されるようP2Pネットワークへ送信されると、各ノードにおいて適用される有効性のための条件のうちの1つは、アンロックスクリプトが第1トランザクションのロックスクリプト内で定義された1つ以上の条件のうちの全部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アウトプットが、別の前の有効なトランザクションによって未だ償還されていないことである。これらの条件のうちのいずれかに従い第2トランザクションが無効であると分かった任意のノードは、該トランザクションを伝播させず、ブロックチェーンに記録させるためにマイニングしてブロックに含めることもしない。

【0008】

トランザクションモデルの代替のタイプは、アカウントに基づくモデルである。この場合、各トランザクションは、過去の一連のトランザクションにおいて、先行するトランザクションのUTXOに戻って参照することによって移転される量を定義するのではなく、絶対的な口座 (アカウント) 残高を参照することによって移転される。全てのアカウントの現在の状態は、ブロックチェーンとは別個のマイナーによって格納され、絶えず更新され

10

20

30

40

50

る。アカウントに基づくモデルのトランザクションは、トランザクションを検証すると同時に各ノードで実行するスマートコントラクトを含むこともできる。

【0009】

いずれかのモデルのトランザクションは、知識証明を含むことができる。「知識証明 (Knowledge proof)」又は「知識の証明 (proof of knowledge)」は、パーティがデータの何らかのピース、例えばこれを d と呼ぶ、を知っていることをテストすることを表す従来の用語である。例として、アウトプットに基づくトランザクションモデルの場合には、1つのトランザクション Tx_1 のアウトプット内のロック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 のインプットが Tx_1 のこのアウトプットを指す場合、 Tx_2 の該インプット内のアンロックスクリプトは、 Tx_1 のアウトプットを償還することに成功するためには、ハッシュパズルを解かなければならない。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。つまり、 $h = H_{puz}(d)$ 。パズルは、スクリプトのピースを含むこともできる。該スクリプトのピースは、ノードにおいて Tx_2 のアンロックスクリプトと一緒に実行されると、 Tx_2 のアンロックスクリプトから d であるとされているデータ値 d' を取り入れ、それをハッシュ関数 H_{puz} によりハッシングし、 Tx_1 のロックスクリプトに含まれるハッシュ値 h と比較する。つまり、該スクリプトのピースは、 $h = H_{puz}(d')$ であるかどうかをチェックし、比較の結果がyes (肯定) である (又は従来の用語で「真」である) 場合にのみ、 Tx_1 のアウトプットをアンロックする。従って、 Tx_2 の受取人は、 d の知識を証明する Tx_2 のアンロックスクリプトに d が含まれる場合にのみ、 Tx_1 のアウトプットをアンロックできる。

【0010】

従来のハッシュパズルを単独で使用するに伴う問題は、不徳なマイナー又は他のノードが Tx_2 のアンロックスクリプト内に d を観察し、従って、彼自身のバージョン Tx_2^* を生成し及びマイニングする (又は発行する) ことができ、 Tx_2 内の意図された受信者 (例えばBob) の代わりに Tx_2^* のアウトプット内で彼自身に支払うことができることである。これを回避するための既存の方法は、 Tx_1 のロックスクリプトにP2PKH (pay-to-public key hash) 要件を追加で含めることである。 d についての知識証明に加えて、これは、 Tx_2 のロックスクリプトに含まれることが意図される受取人の暗号署名を要求する。

【0011】

ハッシュパズル及びP2PKHは、アウトプットに基づくモデルのロック及びアンロックスクリプト以外に、アカウントに基づくモデルではスマートコントラクトを用いて実装されることもできる。

【0012】

当業者に馴染みのあるように、暗号署名は、秘密鍵 V に基づき生成され、秘密-公開鍵ペアの対応する公開鍵 P に基づき検証できる。秘密鍵 V をメッセージ m に適用することにより生成される署名が与えられると、別のパーティが、 V を知らずに、 P を用いて、署名が V を用いて生成されたことを検証することが可能になる (従って、署名自体を検証することは、それ自体の能力において、知識証明の別の形式である)。

【0013】

このためのアルゴリズムの1つの形式は、楕円曲線暗号 (elliptic curve cryptography (ECC)) に基づき動作する楕円曲線デジタル署名 アルゴリズム (elliptic curve digital signature algorithm (ECDSA)) である。

【0014】

この場合、 P 及び V は以下により関連付けられる：

$$P = V \cdot G$$

ここで、 P は2要素ベクトル (P_x, P_y) であり、 V はスカラーである、 G は、2次元楕円曲線上の所定の点 (「生成元」 (generator point)) を表す2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、楕円曲線上の1つの点を別の点に変換する演算の知られている形式である。

【0015】

ECDSA署名は、それぞれr部分 (r-part、(r)) 及びs部分 (s-part、(s)) として従来一般に知られている2個の要素で構成されるタプル(r,s)である。署名(r,s)は、メッセージmに秘密鍵Vを適用することにより生成される。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場合、mはトランザクションの一部であり、署名は平文で該部分に加えてトランザクションにタグ付けされ、該トランザクションが検証されることを可能にする。例えば、アウトプットに基づくモデルでは、署名は、Tx₂の部分に署名し、Tx₁のアウトプットをアンロックするために、Tx₂のロックスクリプトに含まれる。署名済み部分は、標準的に、トランザクションのアウトプットを含み、従って、これらは、署名及び従ってトランザクションを無効にすることなく変更することができない。

【0016】

どんなトランザクションモデルが使用されても、署名(r,s)は以下のように計算される：

【数1】

$$r = [R]_x, \text{ where } R = k \cdot G$$

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

ここで、[R]_xは、2要素ベクトルR=(R_x,R_y)のx軸を示す。kは一時鍵として知られている。これは、集合[1,n-1]から、標準的にはランダムに選択され、ここで、nは素数モジュラスであり、[1,n-1]は、両端を含む整数1からn-1までの実数の整数の集合である。H_{sig}は、ハッシュパズルで使用するハッシュ関数H_{puz}と比べて同じ又は異なる形式のハッシュ関数であり得るハッシュ関数である。

【0017】

署名(r,s)、メッセージm、及び公開鍵Pの知識が与えられると、秘密鍵Vを知らない任意のパーティが、署名がメッセージmに対して秘密鍵を用いて生成されたことを検証することが可能になる。これは、以下を計算することにより行われる：

【数2】

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

及び、[R']_x=rを検証する。これが真の場合にのみ、署名が有効であり、その他の場合には有効ではない。これは、公開鍵Pに関連付けられたパーティが実際にメッセージの署名者であったことの検証として取り入れることができる。

【発明の概要】

【0018】

可能なマイナー攻撃を回避するために、P2PKHと組み合わせて、知識証明としてハッシュパズルを使用することに伴う問題は、P2PKHは、先ず支払いが、dを知っているパーティに対してのみ行われることを保証するが、それは、Tx₁のアウトプットが特定の所定の受信者又は受信者のセットに結びつけられていることも意味するということである（代替の受信者を含み得る代替の条件を指定することが可能である）。

【0019】

ここで、特定のシークレット値を開示することを回避する方法で、該値の知識を証明できる任意の不特定パーティにより該償還可能なトランザクションを可能にすることが望ましい場合があることが認識される。例えば、Aliceは、彼女がシークレット鍵を与える者は誰でもアンロックできる第1トランザクションを設定したいが、彼女はそれらのパーティが誰であるか予め指定したくないとする。例えば、第1トランザクションは、Aliceの代わりに手紙又は小包のような何らかの品物の配達を受け取るために誰かに支払う、及び／又は品物を配達するために運送会社に支払ってよい。第1トランザクションは、シークレットの知識を証明する第2トランザクションにより償還できる。配達の日付の時点で、Aliceは、第1トランザクションを設定し、それを運送会社に提供するか、又はそれをブロックチェーンに発行してよい（又は、例えば、単に運送会社が第1トランザクションを生成す

10

20

30

40

50

ることを可能にするために必要な詳細を提供する)。従って、運送会社は、配達が完了すると支払いが行われるという信頼を有する。しかしながら、Aliceは、この段階で、誰が彼女の代わりに配達を受け取るかを決定したくない。代わりに、彼女は、後に1つ以上の信頼パーティ(例えば、彼女の同居人Charlie及び／又はBobであり、彼らは配達の日在宅であると確認されている)にシークレット値を提供するだけである。これは、彼らのいずれかが、Aliceのシークレット値の証明を実証する第2トランザクションを提供することにより、彼女の代わりに署名することを可能にする。

【0020】

これはほんの1つの説明のための例であることが理解される。別の例として、トランザクションは、契約条件への合意を示すために使用され得る。Aliceは、1つ以上の信頼パーティのうちのサブセットに、彼女の代わりに署名する署名権限又は代理権を与えることを決定した後にのみ、合意したいと望み得る。例えば、合意する時点で、Aliceは、彼女自身が署名しようとしていたが、後になって、彼女は精神的能力を失っているか又は何らかの理由で署名することができないと分かり、従って、誰か他の者に代理権を割り当てる必要がある(例えば、この場合、Bob及びCharlieが彼女の家族又は仕事関係者であり得る)。

【0021】

より一般的には、従来のハッシュパズルに代替案を提供することが望ましい場合がある。ここで、パズルの代替形式は、シークレット値をノードに開示することなく又はそれをブロックチェーンに発行することなく、そしてそれは特定のアイデンティティに結び付けられず、該シークレット値の知識の証明を可能にする。

【0022】

本開示によると、本願明細書で「rパズル(r-puzzle)」又は同義的に「rチャレンジ(r-challenge)」と呼ばれる新しいタイプの知識証明が提供される。これは、チャレンジ(つまり、パズル)の基礎としてECDSA署名のr部分に対応する参照値に基づく。参照値は、第1トランザクションに(例えば、Tx₁のロックスクリプトに)、第2トランザクションが第1トランザクションを償還するために(例えば、Tx₂のアンロックスクリプト内に)指定されたr部分を含む署名を含むことを要求するチャレンジとして、含まれる。第2トランザクション内のrパズルに対する解を提供することにより、これは、証明者が対応する一時鍵kを知っているに違いないことを証明するが、第2トランザクション内でkを開示する必要がない。従って、kは一時秘密鍵として使用でき、rは対応する一時公開鍵と同様に動作する。

【0023】

本願明細書に開示される一態様によると、楕円曲線デジタル署名アルゴリズム(elliptic curve digital signature algorithm, ECDSA)検証関数に基づき知識証明を実行する、コンピュータにより実施される方法が提供される。方法は、ブロックチェーンネットワークの検証ノードにおいて、

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であって、該コードは第1ECDSA署名のr部分の参照インスタンスを指定する要素を含む、ステップと、

第1ECDSA署名の少なくともs部分を含む情報を含む第2トランザクションを受信するステップと、

第1公開鍵を取得するステップであって、第1ECDSA署名は第1公開鍵に対応する第1秘密鍵に基づきメッセージに署名し、メッセージは第2トランザクションの部分である、ステップと、を含む。方法は、次に、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。コードは、第1秘密鍵として誰の公開鍵が使用されたか(従って、誰の公開鍵が第2トランザクション内で使用されるか)に関係なく、ECDSA検証関数が、第1ECDSA署名に適用されるとき、第2トランザクション内で受信されたメッセージ及び取得された第1公開鍵が与えられると、第2トランザクション内で受信されたs部分が第1トランザクションにより指定されたr部分の参照インスタンスに対応することを検証することを条件として、真の結果を返すよう構成される。

【0024】

第1トランザクションのコードに含まれる要素は、r部分自体の参照インスタンスであってよく、又はその変換、例えば、r部分を含むコンポーネントのハッシュ（ここで、ハッシュされたコンポーネントは、r部分自体と等しくなるか、又は例えば別のデータ値dと連結され得る）、であってよい。従って、文脈中の「指定される」は、必ずしも、その明示的な値を含むこと（これは勿論1つの可能な実装である）を意味しないことに留意する。より一般的には、それは、s部分の提出されたインスタンスがECDSA検証アルゴリズムに従い参照インスタンスに有効に対応するかどうかをチェックすることを可能にする、r部分の参照インスタンスと等しい又はそれから導出される任意の要素（例えば、そのハッシュ）を表すことができる。

【0025】

それが指定される方法が何であっても、コードは、従って、被挑戦者（challengee）（例えば、Bob）が一時鍵kを知っているに違いないことを証明する「rパズル」を設定する（kを知らずに解が提供されることは実現不可能である）。パズルは、有利なことに、r部分を、この知識証明の基礎として使用し、被挑戦者が一時鍵kをノードに提出すること又はそれを第2トランザクション（例えば、Tx₂）内で開示することを要求しない。これは、例えば、kが、第1パーティ（例えば、Alice）により、彼女がBob及び／又はCharlieのような1つ以上の第2パーティに署名権限を割り当てるために使用できるある種の一時鍵又はシークレットとして使用できることを意味する。rパズルは、kを開示することなくkの知識を証明するので、悪意あるマイナー又は他のノードは、意図された受益者の代わりに彼／彼女自身に支払い得る有効な第2トランザクションTx₂*を形成するために彼／彼女自身の署名を生成できない。更に、署名のr部分はそれ自体がシステム内の任意のアイデンティティにリンクされないの、これは、kを知っている者は誰でも証明を満たすことができる。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り特定のアイデンティティに結び付けられる必要はない。従って、例えば、Aliceは、誰が第1トランザクションをロックするかを予め決定する必要がない。例えば、彼女は、運送会社に第1トランザクションを設定するための詳細を提供し、後に、彼女の代わりに小包を受け取るための署名権限を誰に与えるかを決定し得る。

【図面の簡単な説明】

【0026】

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、例としてのみ、以下の添付の図面を参照する。

【図1】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】ブロックチェーンに記録されるトランザクションの幾つかの例を概略的に示している。

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

【図4】アウトプットに基づくモデルのノードプロトコルに従う、トランザクションを処理するノードソフトウェアのピースの概略ブロック図である。

【図5】トランザクションの例示的なセットの概略図である。

【図6A】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幾つかを概略的に示す。

【図6B】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幾つかを概略的に示す。

【図6C】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幾つかを概略的に示す。

【図6D】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幾つかを概略的に示す。

【図7】本願明細書でrパズル（又は同義的にrチャレンジ）と呼ばれるタイプの知識証明の1つの可能な実装の概略図である。

【図8】rパズルの別の可能な実装の概略図である。

【図9】rパズルの別の可能な実装の概略図である。

【図 1 0】 rパズルの更に別の可能な実装の概略図である。

【図 1 1】 アカウントに基づくモデルのノードプロトコルに従う、トランザクションを処理するノードソフトウェアのピースの概略ブロック図である。

【図 1 2】 ECDSA署名の例示的なフォーマットを概略的に示す。

【図 1 3】 rパズルの 1つの形式のロック及びアンロックスクリプトの例示的な実装のステップ毎のスクリプト分析である。

【図 1 4 A】 一連のrパズル転送トランザクションを概略的に示す。

【図 1 4 B】 一連のrパズル転送トランザクションを概略的に示す。

【発明を実施するための形態】

【0027】

幾つかの暗号方式では、検証者は、ある人（証明者又は被挑戦者と呼ばれる）が知識証明と呼ばれる情報の何らかのピースを有することを納得させる必要がある場合がある。単純に、これは、検証者に情報のピースを直接提供することにより行うことができる。代替として、証明者は、情報のピースに依存する計算を実行することを要求され得る。望ましくは、関連する計算は、検証者自身がチャレンジを設定するために情報のピースを知る必要がない、又は証明者が情報のピースを知っていることを検証するために情報のピースが検証者に開示される必要がないようなものである。計算方法について、検証計算は入力データに対して実行されなければならない。シークレット値の知識を証明する直接的な方法は、暗号ハッシュ関数の原像（preimage、プリメージ）及び衝突耐性の特徴により、暗号ハッシュ関数の使用を通じるものである。このハッシュ方法は、ハッシュ関数が多いブロックチェーンアプリケーションの秘密鍵—公開鍵暗号システムの基本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に容易に統合できる。この種の知識証明は、標準的にハッシュパズルと呼ばれるブロックチェーンアプリケーションにおいて非常に多い。

【0028】

UTXOに基づくブロックチェーンでは、ハッシュパズルに対する解（ハッシュ値の原像）は、使用条件として設定できる。従って、検証は、トランザクション検証の部分としてマイナーにより実行される。しかしながら、このアプローチでは、トランザクションは、特定の秘密鍵を用いる署名も要求しなければならない。そうでなければ、マイナーは、ブロックにトランザクションを含める前に、ハッシュパズル解を受け取るからである。これは、悪意あるマイナーに、マイナーに属するアドレスに向けられたアウトプットを有する支払いトランザクションを生成する機会を与え得る。

【0029】

本開示では、知識証明は、検証がマイナー（又は転送ノード）により実行されることを可能にしたまま、この問題を回避する知識証明が提供される。これを行うために、知識証明は、楕円曲線デジタル署名アルゴリズム（elliptic curve digital signature algorithm（ECDSA））署名に対応する一時鍵に接続される。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のものであるので、現在のインフラストラクチャに直ちに統合できる。

【0030】

<例示的なシステムの概要>

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む。各ノードはまた、メモリ、すなわち、1つ以上の非一時的コン

10

20

30

40

50

コンピュータ読取可能媒体の形態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、固体ドライブ (solid-state drive (SSD))、フラッシュメモリ又は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学的媒体を使用する1つ以上のメモリユニットを含んでもよい。

【0031】

ブロックチェーン150は、データのブロック151のチェーンを指し、ブロックチェーン150のそれぞれのコピーは、ピアツーピア(peer-to-peer (P2P))ネットワーク160内の複数のノードのそれぞれにおいて維持される。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、この文脈ではトランザクションは、一種のデータ構造を参照する。データ構造の性質は、トランザクションモデル又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する。所与のブロックチェーンは、典型的には、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では、各トランザクション152のデータ構造は、少なくとも1つのインプット及び少なくとも1つのアウトプットを含む。各アウトプットは、そのアウトプットが暗号的にロックされている(アンロックされ、それによって償還又は使用されるために、そのユーザ103の署名を必要とする)ユーザに属するデジタルアセットの数量を表す額 (amount) を指定する。各インプットは、先行するトランザクション152のアウトプットを逆にポイントし、それによってトランザクションをリンクする。

【0032】

ノード104の少なくとも幾つかは、トランザクション152を転送し、それによって伝播する転送ノード104Fの役割を引き受ける。ノード104の少なくともいくつかは、ブロック151をマイニングするマイナー104Mの役割を担う。ノード104の少なくとも幾つかは、記憶ノード104S (「フルコピー (full-copy)」ノードとも呼ばれる) の役割を引き受け、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格納する。各マイナーノード104Mは、マイニングされてブロック151にされることを待ってるトランザクション152のプール154も維持する。所与のノード104は、転送ノード104、マイナー104M、記憶ノード104S、又はこれらの2つ若しくは全部の任意の組み合わせであり得る。

【0033】

所与の現在のトランザクション152jにおいて、インプット (又はそのそれぞれ) は、トランザクションのシーケンスの中の先行トランザクション152iのアウトプットを参照するポインタを含み、このアウトプットが現在のトランザクション152jにおいて償還されるか又は「消費される (spent)」ことを指定する。一般に、先行するトランザクションは、プール154又は任意のブロック151内の任意のトランザクションであり得る。先行するトランザクション152iは、必ずしも、現在のトランザクション152jが生成された又はネットワーク106へ送信されたときに存在する必要はないが、先行するトランザクション152iは、現在のトランザクションが有効であるために存在し検証されている必要がある。従って、本願明細書で「先行する」は、ポインタによりリンクされた論理的シーケンスの中で先行するものを表し、必ずしも時系列の中での生成又は送信の時間を表さない。従って、それは、必ずしも、トランザクション152i, 152jが順不同で生成され又は送信されることを排除しない (以下の親のない (orphan) トランザクションに関する議論を参照する)。先行するトランザクション152iは、等しく、祖先 (antecedent) 又は先行 (predecessor) トランザクションと呼ばれ得る。

【0034】

現在のトランザクション152jのインプットは、先行するトランザクション152iのアウトプットがロックされているユーザ103aの署名も含む。また、現在のトランザクション152jのアウトプットは、新しいユーザ103bに暗号的にロックできる。従って、現在のトランザクション152jは、先行するトランザクション152iのインプットで定義された量を、現在のトランザクション152jのアウトプットで定義された新しいユーザ

10

20

30

40

50

103bに移転することができる。幾つかの場合には、トランザクション152が複数のアウトプットを有し、複数のユーザ間でインプット量を分割してよい(変更を行うために、そのうちの1人がオリジナルユーザとなる)。場合によっては、トランザクションが複数のインプットを有し、1つ以上の先行するトランザクションの複数のアウトプットから量をまとめ、現在のトランザクションの1つ以上のアウトプットに再分配することもできる。

【0035】

上記は「アウトプットベースの」トランザクションプロトコルと呼ばれることがあり、時には「未使用のトランザクションアウトプット(unspent transaction output (UTXO))タイプのプロトコル」(アウトプットはUTXOと呼ばれる)とも呼ばれる。ユーザの合計残高は、ブロックチェーンに格納されている1つの数値で定義されるのではなく、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に分散されている該ユーザの全てのUTXOの値を照合するために、特別な「ウォレット」アプリケーション105を必要とする。

【0036】

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、別のタイプのトランザクションプロトコルを「アカウントベース」のプロトコルと呼ぶことがある。アカウントベースの場合、各トランザクションは、過去の一連のトランザクションにおいて、先行するトランザクションのUTXOに戻って参照することによって移転される量を定義するのではなく、絶対的な口座(アカウント)残高を参照することによって移転される。全てのアカウントの現在の状態は、ブロックチェーンとは別個のマイナーによって格納され、絶えず更新される。このようなシステムでは、トランザクションは、アカウントの連続したトランザクション記録(いわゆる「ポジション」)を用いて発注される。この値は、送信者により彼らの暗号署名の一部として署名され、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、任意的なデータフィールドもトランザクションに署名することができる。このデータフィールドは、例えば、前のトランザクションIDがデータフィールドに含まれている場合、前のトランザクションを遡ってポイントしてよい。

【0037】

どちら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を実行したい場合、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のノードの1つ104(現在は、通常、サーバ又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)に新しいトランザクションを送信する。このノード104は、各ノード104に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どうかをチェックする。ノードプロトコルの詳細は、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、全体のトランザクションモデルと一緒に形成する。ノードプロトコルは、典型的には、ノード104に、新しいトランザクション152j内の暗号署名が、トランザクション152の順序付けされたシーケンスの中の前のトランザクション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ことを要求する。アウトプットベースの場合、これは、新しいトランザクション152jのインプット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消費する先行するトランザクション152jのアウトプット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んでよく、この条件は、典型的には、新しいトランザクション152jのインプット内の暗号署名が、新しいトランザクションのインプットがポイントする前のトランザクション152iのアウトプットをアンロックすることを少なくともともチェックすることを含む。幾つかのトランザクションプロトコルでは、条件は、少なくとも部分的に、インプット及び/又はアウトプットに含まれるカスタムスクリプトによって定義されてもよい。或いは、単にノードプロトコルだけで固定することもできるし、或いは、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が有効であれば、現在のノードは新しいトランザクションを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転送する。これらのノード104の少なくともいくつかは、同じノードプロトコルに従って同じテストを適用し、新しいトランザクション

10

20

30

40

50

152jを1つ以上のさらなるノード104に転送するなど、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のネットワーク全体に伝播される。

【0038】

アウトプットベースのモデルでは、与えられたアウトプット(例えば、UTXO)が消費されるかどうかの定義は、ノードプロトコルに従って別の今後の(onward)トランザクション152jのインプットによって既に有効に償還されているかどうかである。トランザクションが有効であるための別の条件は、それが消費又は償還を試みる先行するトランザクション152iのアウトプットが、別の有効なトランザクションによって未だ消費/償還されていないことである。ここでも、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播又は記録されない。これは、支払者が同じトランザクションのアウトプットを複数回消費しようとする二重支出を防ぐ。一方、アカウントベースのモデルは、口座残高を維持することによって、二重支出を防ぐ。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、口座残高は、一度に単一の定義された状態を有する。

【0039】

検証に加えて、ノード104Mのうちの少なくともいくつかは、「proof of work」に支えられているマイニングと呼ばれるプロセスで、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合する。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。そして、マイナーは、暗号パズルを解決しようとする試みるにより、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を組み立てるために競争する。これは、典型的には、ノンスがトランザクションのプール154と連結され、ハッシュされるときに、ハッシュのアウトプットが所定の条件を満たすような「ノンス」値を探すことを含む。例えば、所定の条件は、ハッシュのアウトプットが、所定の数の先行ゼロを有することであってもよい。ハッシュ関数の特性は、インプットに関して予測不可能なアウトプットを持つことである。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。

【0040】

パズルを解く最初のマイナーノード104Mは、これをネットワーク106に通知し、その解を証明として提供する。この解は、ネットワーク内の他のノード104によって簡単にチェックすることができる(ハッシュが対する解が与えられれば、ハッシュのアウトプットが条件を満たすことを確認することは簡単である)。勝者がパズルを解いたトランザクションのプール154は、各ノードで勝者が発表した解をチェックしたことに基づいて、記憶ノード104Sとして機能するノード104のうちの少なくともいくつかによってブロックチェーン150の新しいブロック151として記録される。また、新しいブロック151nにはブロックポインタ155が割り当てられ、チェーン内で前に作成されたブロック151n-1を指すようになっている。proof-of-workは、新しいブロックを151作成するのに多大な労力を要し、二重の支出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く、従ってマイニングノード104Mが二重支払いを彼らのブロックに含まないようにする動機が働くので、二重の支出のリスクを減じるのを助ける。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内の各格納ノード104Siで認識され、維持されるため、変更することはできない。また、ブロックポインタ155は、ブロック151に順序を課す。トランザクション152は、P2Pネットワーク106内の各記憶ノード104Sで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を提供する。

【0041】

パズルを解決するために常に競争している異なるマイナー104Mは、いつ解を探し始めたかによって、いつでもマイニングされていない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パズルを解いているかもしれないことに留意する。パズル

10

20

30

40

50

を解く者は誰でも、最初に次の新しいブロック151nに含まれるトランザクション152を定義し、現在のマイニングされていないトランザクションのプール154が更新される。そして、マイナー104Mは、新たに定義された未解決のプール154からブロックを作り出すために、競争を続ける。また、生じ得る「分岐（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人のマイナー104Mが互いに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矛盾したビューが伝播する場合である。要するに、分岐の枝が伸びるときは常に、最長のものが最終的なブロックチェーン150になる。

【0042】

ほとんどのブロックチェーンでは、勝ったマイナー104Mは、何も無いものから新しい量のデジタルアセットを生み出す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ている(通常のトランザクションでは、あるユーザから別のユーザにデジタルアセットの量を移転する)。したがって、勝ったノードは、ある量のデジタルアセットを「マイニング」したと言える。この特殊なタイプのトランザクションは「生成 (generation)」トランザクションと呼ばれることもある。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、マイナー104Mがproof-of-work競争に参加する動機を与える。多くの場合、通常の(非生成)トランザクションは、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ー104Mにさらに報酬を与えるために、追加のトランザクション手数料をそのアウトプットの1つにおいて指定する。

【0043】

マイニングに含まれる計算リソースのために、典型的には、少なくともマイナーノード104Mの各々は、1つ以上の物理的サーバユニットを含むサーバ、又はデータセンタ全体の形態をとる。各転送ノード104M及び/又は記憶ノード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、任意の所与のノード104は、ユーザ端末又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を含むことができる。

【0044】

各ノード104のメモリは、それぞれの1つ以上の役割を実行し、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を格納する。ノード104に属するいずれの動作も、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、アプリケーションレイヤにおける1つ以上のアプリケーション、又はオペレーティングシステムレイヤ若しくはプロトコルレイヤのような下位レイヤ、又はこれらの任意の組合せの中に実装されてよい。また、本明細書で使用される「ブロックチェーン」という用語は、一般的な技術の種類を指す一般的な用語であり、任意の特定の専有のブロックチェーン、プロトコル又はサービスに限定されない。

【0045】

また、ネットワーク101には、消費者ユーザの役割を果たす複数のパーティ103の各々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及び被支払人の役割を果たすが、他のパーティの代わりにトランザクションをマイニングし又は伝播することに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つのパーティ103及びそれぞれの機器102が説明のために示されており、第1パーティ103a及びそのそれぞれのコンピュータ機器102a、幾つかの第2パーティ103b及びそのそれぞれのコンピュータ機器102bである。より多くのこのようなパーティ103及びそれらのそれぞれのコンピュータ機器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。各パーティ103は、個人又は組織であってもよい。純粹に例示として、第1パーティ103aは、本明細書においてAliceと称され、第2パーティ103bは、Bobと称されるが、これは限定的なものではなく、本明細書においてAlice又はBobという言葉及

10

20

30

40

50

は、それぞれ「第1パーティ」及び「第2パーティ」と置き換えることができることは理解されるであろう。

【0046】

各パーティ103のコンピュータ機器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。各パーティ103のコンピュータ機器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又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、例えば、ハードディスクなどの磁気媒体、SSD、フラッシュメモリ又は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学的媒体を使用する1つ以上のメモリユニットを含んでもよい。各パーティ103のコンピュータ機器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記憶する。所与のノード104に属するいずれの動作も、それぞれのコンピュータ機器102の処理装置上で実行されるソフトウェアを使用することにより実行され得ることが理解されよう。各パーティ103のコンピュータ機器102は、少なくとも1つのユーザ端末、例えばデスクトップ又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、又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。所与のパーティ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接続されたリソースを含んでもよい。

【0047】

クライアントアプリケーション105は、最初に、1つ以上の適切なコンピュータ読取可能な記憶媒体、例えばサーバからダウンロードされたもの、又はリムーバブルSSD、フラッシュ・メモリ・キー、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、光ディスク、例えばCD又はDVD ROM、又はリムーバブル光学ドライブなどのリムーバブル記憶装置上で、任意の所与のパーティ103のコンピュータ機器102に提供され得る。

【0048】

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」機能を備える。これには主に2つの機能を有する。これらのうちの1つは、それぞれのユーザパーティ103が、ノード104のネットワーク全体にわたって伝播され、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アセットの量をそれぞれのパーティに報告することである。アウトプットベースのシステムでは、この第2の機能は、当該パーティに属するブロックチェーン150全体に散在する様々なトランザクション152のアウトプットの中で定義される量を照合することを含む。

【0049】

注：種々のクライアント機能が所与のクライアントアプリケーション105に統合されるところとして説明されることがあるが、これは、必ずしも限定的ではなく、代わりに、本願明細書に記載される任意のクライアント機能が2つ以上の異なるアプリケーションのスーツに実装されてよく、例えばAPIを介してインタフェースし、又は一方が他方へのプラグインである。より一般的には、クライアント機能は、アプリケーションレイヤ、又はオペレーティングシステムのような下位レイヤ、又はこれらの任意の組合せにおいて実装され得る。以下は、クライアントアプリケーション105の観点で説明されるが、これは限定的ではないことが理解される。

【0050】

各コンピュータ機器102上のクライアントアプリケーション又はソフトウェア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、記憶ノード104のうちの1つ、一部、又は全部にコンタクトして、それぞれのパーティ103が受領者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に問い合わせることができる(又は、実施形態では、ブロックチェーン150は、部分的にその公開視認性を通じてトランザクションの信頼を提供する公開的设备であるため、実際には、ブロックチェーン150内の他のパーティのトランザクションを検査する)。各コンピュータ機器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、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することができる)。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)。

【0051】

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、前述のように、proof-of-work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152のセットを含む。各ブロック151は、また、ブロック151への逐次的順序を定義するように、チェーン内の先に生成されたブロック151を遡ってポイントするブロックポインタ155を含む。ブロックチェーン150はまた、proof-of-work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シーケンスに順序を定義するために、前のトランザクションへのポインタを含む(注：トランザクション152のシーケンスは、分岐することが許される)。ブロック151のチェーンは、チェーンの最初のブロックであったジェネシスブロック (genesis block (Gb)) 153にまで戻る。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、先行するトランザクションではなくジェネシスブロック153を指し示した。

【0052】

所与のパーティ103、例えばAliceがブロックチェーン150に含まれる新たなトランザクション152jを送信したいと望む場合、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のウォレット機能を使用して)新たなトランザクションを定式化する(formulate)。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、Aliceのコンピュータ102に最も近いか又は最も良好に接続されている転送ノード104Fであってもよい。任意の所与のノード104が新しいトランザクション152jを受信すると、ノードプロトコル及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」であるための特定の条件を満たしているかどうかをチェックすることを含み、その例については、簡単に詳述する。幾つかのトランザクションプロトコルでは、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構成可能であってもよい。或いは、条件は単にノードプロトコルの組み込み機能であってもよく、或い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。

【0053】

新たに受信されたトランザクション152jが、有効であると見なされるテストに合格したという条件で(すなわち、「有効である」という条件で)、トランザクション152jを受

10

20

30

40

50

信した任意の記憶ノード104Sは、そのノード104Sに維持されているブロックチェーン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、検証済みトランザクション152をP2Pネットワーク106内の1つ以上の他のノード104に伝播する。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全体に間もなく伝播されることを意味する。

【0054】

ひとたび1つ以上の記憶ノード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、マイナーノード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proof-of-workパズルを解決するために競争を開始する(他のマイナー104Mは、依然として、プール154の古いビュー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、Alice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proof-of-workが行われると、ブロックチェーン150内のブロック151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を含むので、トランザクションの順序もまた、不変的に記録される。

【0055】

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可能性があり、従って、1つのインスタンスがマイニングされてブロック150になる前に、どのインスタンスが「有効」であるかについて矛盾するビューを有することがあり、その時点で、全部のノード104は、マイニングされたインスタンスのみが有効なインスタンスであることに合意する。ノード104が1つのインスタンスを有効であるとして受け入れ、次に第2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、該ノード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だマイニングされていないインスタンスを破棄する(つまり、無効であるとして扱う)。

【0056】

<UTXOに基づくモデル>

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、アウトプットベース又は「UTXO」ベースのプロトコルを参照して説明される。しかし、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。

【0057】

UTXOベースのモデルでは、各トランザクション(「Tx」)152は、1つ以上のインプット202及び1つ以上のアウトプット203を含むデータ構造を含む。各アウトプット203は、未使用トランザクションアウトプット(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インプット202のソースとして使用することができる(UTXOが未だ償還されていない場合)。UTXOは、デジタルアセット(値のストア)の量を指定する。それは、情報の中でも特にその元となったトランザクションのトランザクションIDを含む。トランザクションデータ構造はまた、ヘッダ201も含んでもよく、ヘッダ201は、インプットフィールド202及びアウトプットフィールド203のサイズの指示子を含んでもよいヘッダ201を含んでもよい。ヘッダ201は、トランザクションのIDも含んでもよい。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、マイナー104Mに提出された未処理トランザクション152のヘッダ201に格納される。

【0058】

例えばAlice103aは、問題のデジタルアセットの量をBob103bに移転するトラン

10

20

30

40

50

ザクション152jを作成したいと考えているとする。図2において、Aliceの新しいトランザクション152jは「Tx1」とラベル付けされている。これは、Aliceにロックされているデジタルアセットの量を、シーケンス内の先行するトランザクション152iのアウトプット203に取り入れ、その少なくとも一部をBobに移転する。先行するトランザクション152iは、図2において「Tx0」とラベル付けされている。Tx0と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の最初のトランザクションであること、又は、Tx1がプール154の直ぐ次のトランザクションであることを意味しない。Tx1は、まだAliceにロックされた未使用アウトプット203を有する任意の先行する（つまり祖先）トランザクションのいずれかを指し示すことができる。

【0059】

先行するトランザクションTx0は、Aliceがその新しいトランザクションTx1を作成するとき、又は少なくとも彼女がそれをネットワーク106に送信するときまでに、既に検証され、ブロックチェーン150に含まれていてもよい。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、或いは、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。或いは、Tx0及びTx1が生成されネットワーク102に送信されることができ、或いは、ノードプロトコルが「孤児（orphan）」トランザクションのバッファリングを許容する場合にはTx1の後にTx0が送信されることもできる。ここでトランザクションのシーケンスの文脈で使用される「先行する」及び「後の」という用語は、トランザクション内で指定されたトランザクションポイント(どのトランザクションがどの他のトランザクションを指すかなど)によって定義されるシーケンス内のトランザクションの順序を指す。それらは、「先行する」及び「相続する」又は「祖先」及び「子孫」、「親」及び「子」、等により、等しく置き換えられ得る。これは、必ずしも、それらが作成され、ネットワーク106に送られ、又は任意の所与のノード104に到達する順序を意味しない。それにもかかわらず、先行するトランザクション(祖先トランザクション又は「親」)を指す後続のトランザクション(子孫トランザクション又は「子」)は、親トランザクションが検証されない限り、検証されない。親の前にノード104に到着した子は孤児とみなされる。それは、ノードプロトコル及び/又はマイナーの行動に応じて、親を待つために特定の時間、破棄又はバッファリングされることがある。

【0060】

先行するトランザクションTx0の1つ以上のアウトプット203のうちの1つは、本明細書でUTXO₀とラベル付けされた特定のUTXOを含む。各UTXOは、UTXOによって表されるデジタルアセット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に、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インプット202の中のアンロック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クスクリプトとを含む。典型的には、ロックスクリプトは、特定のパーティ(それが含まれているトランザクションの受益者)に量をロックする。すなわち、ロックスクリプトは、標準的に以下のようなアンロック条件を定義する：後続のトランザクションのインプット内のアンロックスクリプトは、先行するトランザクションがロックされたパーティの暗号署名を含む。

【0061】

ロックスクリプト(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, capital S)と呼ばれる。ロックスクリプトは、トランザクションアウトプット203を消費するために必要な情報、例えば、Aliceの署名の必要条件を指定する。トランザクションのアウトプットには、アンロックスクリプトが現れる。アンロックスクリプト(別名：scriptSig)は、ロック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、Bobの署名を含んでもよい。アンロックスクリプトは、トランザクションのインプット202に現れる。

【0062】

10

20

30

40

50

従って、図示の例では、Tx₀のアウトプット203のUTXO₀は、ロックスクリプト[Checks_{sig} 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)、Aliceの署名Sig P_Aを必要とする。[Checks_{sig} P_A]は、Aliceの公開-秘密鍵ペアからの公開鍵P_Aを含む。Tx₁のインプット202は、Tx₁を指すポインタ(例えば、そのトランザクションID、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TxID₀)を含む。Tx₁のインプット202は、Tx₀の任意の他の可能なアウトプット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のインプット202は、さらに、Aliceが鍵ペアからのAliceの秘密鍵をデータの所定の部分(暗号において「メッセージ」と呼ばれることもある)に適用することによって作成された、Aliceの暗号署名を含むアンロックスクリプト<Sig P_A>を含む。有効な署名を提供するためにAliceが署名する必要があるデータ(又は「メッセージ」)は、ロックスクリプトにより、又はノードプロトコルにより、又はこれらの組み合わせによって定義され得る。

【0063】

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、ロックスクリプトとアンロックスクリプトを一緒に実行して、アンロックスクリプトがロックスクリプトで定義されている条件(この条件は1つ以上の基準を含むことができる)を満たしているかどうかをチェックすることを含む。実施形態では、これは、2つのスクリプトの連結を含む。

【数3】

<Sig P_A> <P_A> || [Checks_{sig} P_A]

【0064】

ここで、「||」は連結を表し、「<...>」はスタックにデータを配置することを意味し、「[...]」はアンロックスクリプトに含まれる機能である(本例では、スタックベースの言語)。同等に、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により1つずつ実行されてよい。いずれの方法でも、一緒に実行する場合、スクリプトは、Tx₀のアウトプット内のロックスクリプトに含まれるAliceの公開鍵P_Aを使用して、Tx₁のインプット内のロックスクリプトが、データの期待部分に署名するAliceの署名を含むことを認証する。また、データの期待部分(「メッセージ」)も、この認証を実行するためにTx₀に含まれる必要がある。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すでに本質的に存在するので、データの署名された部分の指定にクリアに含まれる必要がある)。

【0065】

公開-秘密暗号法による認証の詳細は、当業者には周知であろう。基本的に、Aliceが彼女の秘密鍵により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、Aliceの公開鍵とそのメッセージが明らか(暗号化されていないメッセージ)ならば、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Alice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、実施形態では、特定のデータ片又はトランザクションの部分等に署名するという言及は、データ片又はトランザクションの部分のハッシュに署名することを意味し得る。

【0066】

Tx₁内のアンロックスクリプトが、Tx₀のロック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、Aliceの署名がTx₁内で提供され、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。それが記憶ノード104Sである場合、これは、proof-of-work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。それが転送ノード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、それによって、それがネットワ

10

20

30

40

50

ーク全体に伝播されることになる。一旦、 Tx_1 が検証され、ブロックチェーン150に含まれると、これは、 Tx_0 からのUTXO₀を消費したものとして定義する。 Tx_1 は、未使用トランザクションアウトプット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によって既に消費されたアウトプットを消費しようとする場合、 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、先行するトランザクション Tx_0 において参照されたUTXOが既に使用されているかどうか(既に別の有効なトランザクションへの有効なインプットを形成しているかどうか)もチェックする必要がある。これが、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が消費されたUTXO₂₀₃をマークする別個のデータベースを維持することができるが、最終的には、UTXOが消費されたかどうかを定義するのは、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インプットを既に形成しているかどうかである。

【0067】

所与のトランザクション152の全部のアウトプット203の中で指定された総量が全部のそのインプット202により指される総量より大きい場合、これは、殆どのトランザクションモデルにおいて無効の別の基礎である。従って、このようなトランザクションは、伝播されず、マイニングされてブロック151にされることもない。

【0068】

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うち、別の分量が消費されている一方で、分量を「残しておく」ことはできない。ただし、UTXOからの量は、次のトランザクションの複数のアウトプットに分割できる。例えば、 Tx_0 のUTXO₀で定義された量は、 Tx_1 の複数のUTXOに分割できる。したがって、AliceがBobにUTXO₀で定義された量のすべ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のアウトプットの中で自分自身にお釣りを与えるか、又は別のパーティに支払うことができる。

【0069】

実際には、Aliceは通常、勝ったマイナーのための手数料も含める必要がある。なぜなら、今日では、生成トランザクションの報酬だけでは、マイニングを動機付けるには通常十分ではないからである。Aliceがマイナーのための手数料を含まない場合、 Tx_0 はマイナーのノード104Mによって拒否される可能性が高く、したがって、技術的には有効であるが、それは依然として伝播されず、ブロックチェーン150に含まれない(マイナーのプロトコルは、マイナーが望まない場合には、マイナー104Mにトランザクション152を受け入れるよう強制しない)。一部のプロトコルでは、マイニング料金は、独自の別個のアウトプット203を必要としない(すなわち、別個のUTXOを必要としない)。代わりに、インプット202によって指される総量と、所与のトランザクション152のアウトプット203の中で指定される総量との間の差は、勝ったマイナー104に自動的に与えられる。例えば、UTXO₀へのポイントが Tx_1 への唯一のインプットであり、 Tx_1 は1つのアウトプットUTXO₁しか持っていないとする。UTXO₀で指定されたデジタルアセットの量がUTXO₁で指定された量より多い場合、その差は自動的に勝ったマイナー104Mへ行く。しかし、代替的又は追加的に、必ずしも、トランザクション152のUTXO₂₀₃のうちの独自のものにおいて、マイナー手数料を明示的に指定することは除外されない。

【0070】

Alice及びBobのデジタルアセットは、ブロックチェーン150内の任意のトランザクション152の中で彼らにロックされた未使用UTXOで構成されている。従って、典型的には、所与のパーティ103のアセットは、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、所与のパーティ103の総残高を定義する1つの数値は記憶されていない。各パーティへのロックされた、別の将来の(onward)トランザクションに未だ消費されていない全て

10

20

30

40

50

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、記憶ノード104Sのいずれかに記憶されたブロックチェーン150のコピーを、例えば、各パーティのコンピュータ機器02に最も近いが、又は最も良好に接続されている記憶ノード104Sに問い合わせることによって行うことができる。

【0071】

スクリプトコードは、概略的に表現されることが多い(すなわち、正確な言語ではない)ことに注意する。例えば、`[Checksig PA]`を`[ChecksigPA]=OP_DUP OP_HASH160 <H(PA)> OP_EQUALVERIFY OP_CHECKSIG`を意味するように記述し得る。「OP_....」は、スクリプト言語の特定のオペコードを表す。OP_CHECKSIG(「Checksig」とも呼ばれる)は、2つのインプット(署名と公開鍵)を取り込み、楕円曲線デジタル署名アルゴリズム(Elliptic Curve Digital Signature Algorithm (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。ランタイムでは、署名(「sig」)の発生は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」インプット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格納することができ、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使用不可能アウトプット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格納することが望ましいドキュメントを含むことができる。

【0072】

署名PA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に基づく。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インプットの一部、及びトランザクションアウトプットの全部又は一部に署名する。署名するアウトプットの特定の部分は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、どのアウトプット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)。

【0073】

ロックスクリプトは、それぞれのトランザクションがロックされているパーティの公開鍵を含んでいることを表す「scriptPubKey」と呼ばれることがある。アンロックスクリプトは、対応する署名を提供することを表す「scriptSig」と呼ばれることがある。しかし、より一般的には、UTXOが償還される条件が署名を認証することを含むことは、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必須ではない。より一般的には、スクリプト言語は、1つ以上の条件を定義するために使用され得る。したがって、より一般的な用語「ロックスクリプト」及び「アンロックスクリプト」が好ましい。

【0074】

<任意的なサイドチャネル>

図3は、ブロックチェーン150を実装するための更なるシステム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。Alice及びBobのコンピュータ機器102a、102bの各々に存在するクライアントアプリケーションは、それぞれ、追加通信機能を含む。つまり、それは、Alice103aが、(いずれかのパーティ又は第三者の勧誘で)Bob103bとの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ネットワークと別個にデータの交換を可能にする。このような通信は、時に「オフチェーン」と呼ばれる。例えば、これは、パーティのうちの一方がトランザクションをネットワーク106にブロードキャストすることを選択するまで、トランザクションがネットワークP2P106に(未だ)発行されることなく、又はチェーン150上でそのようにすることなく、AliceとBob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てよい。代替又は追加で、サイドチャネル301は、任意の他のトランザクションに関連するデータ、例えば、鍵、交渉される量又は条項、データコンテンツ、等を交換するために使用されてよ

10

20

30

40

50

い。

【0075】

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もよい。代替又は追加で、サイドチャネル301は、モバイルセルラネットワーク、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AliceとBobの装置102a、102bの間の直接有線若しくは無線リンクのような異なるネットワークを介して確立されてよい。一般に、本願明細書のどこかで言及されるサイドチャネル301は、「オフチェーン」で、つまりP2Pオーバーレイネットワーク106と別個にデータを交換するための1つ以上のネットワーキング技術又は通信媒体を介する任意の1つ以上のリンクを含んでよい。1つより多くのリンクが使用されるとき、全体としてのオフチェーンリンクのバンドル又は集合がサイドチャネル301と呼ばれてよい。従って、Alice及びBobが特定の情報又はデータ片等をサイドチャネル301を介して交換すると言われる場合、これは、必ずしも全部のこれらのデータ片が正確に同じリンク又は同じ種類のネットワークを介して送信される必要があることを意味しないことに留意する。

【0076】

<ノードソフトウェア>

図4は、UTXO又はアウトプットに基づくモデルの例における、P2Pネットワーク106の各ノード104で実行され得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の決定エンジン404、及び1つ以上のブロックチェーン関連機能モジュールのセット405を含む。任意の所与のノード104で、これらは、(ノードの1つ以上の役割に依存して)マイニングモジュール405M、転送モジュール405F、及び格納モジュール405S、のうちの任意の1、2、又は3個全部を含んでよい。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、それらをノードプロトコルに従い処理するよう構成される。トランザクション152m (Tx_m) が受信され、別の先行するトランザクション152m-1 (Tx_{m-1}) のアウトプット (例えばUTXO) をポイントするインプットを有するとき、プロトコルエンジン401は、 Tx_m 内のアンロック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。プロトコルエンジン401は、更に、 Tx_m のインプットの中のポインタに基づき、 Tx_{m-1} を識別し検索する。それは、 Tx_{m-1} が未だブロックチェーン150上にない場合、それぞれのノード自身の保留中トランザクションのプール154から、又は Tx_{m-1} が既にブロックチェーン150上にある場合、それぞれのノード若しくは別のノード104に格納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} を検索してよい。いずれの方法も、スクリプトエンジン401は、 Tx_{m-1} のポイントされるアウトプットの中のロックスクリプトを識別し、これをスクリプトエンジン402に渡す。

【0077】

スクリプトエンジン402は、従って、 Tx_{m-1} のロックスクリプト、及び Tx_m の対応するインプットからのアンロックスクリプトを有する。例えば、 Tx_1 及び Tx_2 が図4に示されるが、 Tx_0 及び Tx_1 等のようなトランザクションの任意のペアについても同様である。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し、これらは、使用されているスタックに基づくスクリプト言語 (例えばScript) に従い、スタック403にデータを置くことと、データを検索することを含む。

【0078】

スクリプトを一緒に実行することにより、スクリプトエンジン402は、アンロックスクリプトがロックスクリプトの中で定義された1つ以上の基準を満たすか否か、つまり、それがロックスクリプトが含まれるアウトプットを「アンロック」するか否かを決定する。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に返す。スクリプトエンジン402は、アンロックスクリプトは対応するロックスクリプトの中で指定された1つ以上の基準を満たすと決定した場合、結果「真」を返す。その他の場合、そ

れは結果「偽」を返す。

【0079】

アウトプットに基づくモデルでは、スクリプトエンジン402からの結果「真」は、トランザクションの有効性についての条件のうちの1つである。標準的に、同様に満たされなければならない、プロトコルエンジン401により評価される1つ以上の更なるプロトコルレベルの条件が更にあり、 Tx_m のアウトプットの中で指定されたデジタルアセットの総量がそのインプットによりポイントされる総量を超えないこと、 Tx_{m-1} のポイントされるアウトプットは別の有効なトランザクションにより未だ使用されていないこと、等である。プロトコルエンジン401は、1つ以上のプロトコルレベルの条件と一緒にスクリプトエンジン402からの結果を評価し、それら全部が真である場合、トランザクション Tx_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有効であるかどうかの指示を、アプリケーションレベル決定エンジン404に出力する。 Tx_m が実際に検証されたことのみを条件として、決定エンジン404は、マイニングモジュール405M及び転送モジュール405Fの一方又は両方を、それらのそれぞれのブロックチェーンに関連する機能を Tx_m に関して実行するよう制御することを選択してよい。これは、マイニングモジュール405Mがマイニングしてブロック151にするために Tx_m をノードのそれぞれのプール154に追加すること、及び／又は、転送モジュール405Fが Tx_m をP2Pネットワーク106内の別のノード104へ転送することを含んでよい。しかしながら、実施形態では、決定エンジン404は無効なトランザクションを転送し又はマイニングすることを選択しないが、逆に言えば、これは必ずしも、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又は転送をトリガしなければならないことを意味するものではないことに留意する。任意的に、実施形態では、決定エンジン404は、これらの機能のうち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してよい。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ること、及び十分なマイニング手数料が残されることの両方を条件としてのみ、トランザクションをマイニングすることを選択してよい。

【0080】

用語「真 (true)」及び「偽 (false)」は、本願明細書では、必ずしも単一の2進数字 (ビット) のみの形式で表現される結果を返すことに限定しないが、それは勿論1つの可能な実装であることに留意する。より一般的には、「真」は、成功又は肯定的な結果を示す任意の状態を表すことができ、「偽」は、不成功又は非肯定的な結果を示す任意の状態を表すことができる。例えば、アカウントに基づくモデルでは (図4に示されない)、「真」の結果は、ノード104による署名の暗示的なプロトコルレベルの検証と、スマートコントラクトの追加の肯定的なアウトプットとの組合せにより示され得る (全体の結果は、両方の個々の結果が真である場合に、真を伝達すると考えられる)。

【0081】

<例示的なトランザクションセット>

図5は、本願明細書に開示される実施形態に従い使用するためのトランザクション152のセットを示す。セットは、第0トランザクション Tx_0 、第1トランザクション Tx_1 、及び第2トランザクション Tx_2 を含む。「第0」、「第1」、及び「第2」は、単なる便宜上のラベルであることに留意する。それらは、必ずしも、これらのトランザクションが直ちに1つずつブロック151又はブロックチェーン150内に置かれること、又は第0トランザクションがブロック151又はブロックチェーン150内の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、それらのトランザクションがネットワーク106へ送信される順序に関して何も示唆しない。それらは、単に、1つのトランザクションのアウトプットが次のトランザクションのインプットによりポイントされる論理的シリーズを表す。幾つかのシステムでは、親をその子の後にネットワーク106へ送信することが可能であることを思い出してほしい (この場合、「親のない」子がある期間の間、1つ以上のノード104においてバッファされ、その間、親が到着するのを待っている)。

【0082】

第0トランザクションTx₀は、本発明の目的のためにソーストランザクションと呼ばれてもよく、Alice103aへのロックされたデジタルアセットの量のソースとして機能する。第1トランザクションTx₁は、本発明の目的のためにチャレンジトランザクション又はパズルトランザクションと呼ばれてもよい。それは、第2トランザクションTx₂がrパズルに対する解を提供することに依存して、ソーストランザクションTx₀からデジタルアセットの量を条件付きで移転するための仲介として機能する。第2トランザクションTx₂は、証明トランザクション又は支払いトランザクションと呼ばれてもよく、第1トランザクションTx₁内に設定されたrパズルに対する解を提供し、結果として生じる証明者（又は場合によっては、証明者が代表を務める受益者）への支払いをロックするトランザクションである。実施形態は、証明者（第2パーティ）がBobになるが、後の議論に基づき認められ、実際にrパズルが、任意の第2パーティがrパズルを解く有効な署名を提供する限り、アイデンティティに関係なく、彼らが証明者になることを許容する例を用いて説明され得る。

10

【0083】

図5に示すように、ソーストランザクションTx₀は、デジタルアセットの量を指定する少なくとも1つのアウトプット2030（例えば、Tx₀のアウトプット0）を含み、及びAlice103aへのこのアウトプットをロックするロックスクリプトを更に含む。これは、ソーストランザクションTx₀のロックスクリプトが、少なくとも1つの条件が満たされることを要求することを意味する。この条件は、アウトプットをアンロックしようとする（従って、デジタルアセットの量を償還する）任意のトランザクションのインプットが、そのアンロックスクリプト内にAliceの暗号署名（つまり、Aliceの公開鍵を使用する）を含まなければならないことである。この意味で、Tx₀のアウトプット内で定義される量は、Aliceにより所有されると言える。アウトプットは、UTXOと呼ばれてよい。どの先行するトランザクションのアウトプットがTx₀のインプットをポイントするかは（それらが、Tx₀の合計アウトプットをカバーするのに十分である限り）、本発明の目的のために特に重要ではない。

20

【0084】

本発明の場合には、ソーストランザクションTx₀のアウトプット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、Tx₀の関連するアウトプット（図示の例ではTx₀のアウトプット0）へのポインタを含み及び該アウトプットのロックスクリプト内で定義された条件に従いTx₀のポイントされたアウトプットをアンロックするよう構成される、少なくともAliceの署名を要求するアンロックスクリプトを更に含む、少なくとも1つのインプット2021（例えば、Tx₁のインプット0）を有する。Tx₀のロックスクリプトにより要求されるAliceからの署名は、Tx₁の特定の部分に署名することが要求される。幾つかのプロトコルでは、署名される必要のあるTx₁の部分は、Tx₁のアンロックスクリプト内で定義された設定であり得る。例えば、これは、署名に付加される1バイトであるSIGHASHフラグにより設定されてよい。従って、データの観点では、アンロックスクリプトは次の通りである：〈Sig P_A〉〈sighashflag〉〈P_A〉代替として、署名される必要のある部分は、単にTx₁の固定又はデフォルト部分であってよい。いずれの方法も、署名されるべき部分は、標準的に、アンロックスクリプト自体を除き、及びTx₁のインプットの一部又は全部を除いてよい。しかしながら、Tx₁の署名済み部分は、少なくとも、rパズルを含むアウトプット2031を含む（以下を参照、本例ではTx₁のアウトプット0）。

30

40

【0085】

第1トランザクションTx₁は、少なくとも1つのアウトプット2031（例えば、ここでもアウトプットがUTXOと呼ばれてよいTx₁のアウトプット0）を有する。第1トランザクションTx₁のアウトプットは、任意の1つのパーティにロックされない。Tx₀と同様に、それは、転送されるべきデジタルアセットの量を指定する少なくとも1つのアウトプット（例えば、Tx₁のアウトプット0）を有する。該アウトプットは、該アウトプットをアンロックする、従って該量を償還するために何が必要かを定義するロックスクリプトを

50

更に含む。しかしながら、このロックスクリプトは、 r パズルの解を提供する任意のパーティによりアンロックされることが可能である。

【0086】

第2トランザクション（支払いトランザクション） Tx_2 は、少なくとも1つのインプット2022（例えば、 Tx_2 のインプット0）を有し、該インプットは、 Tx_1 の上述のアウトプット（図示の例では Tx_1 のアウトプット0）へのポインタを含み、 Tx_1 のロックスクリプトの中で定義されたアンロックスクリプトの1つ以上の要件を満たすことに基づき Tx_1 の該アウトプットをアンロックするよう構成されるアンロックスクリプトも更に含む。本願明細書に開示される実施形態によると、アンロック条件は、少なくとも、対応するアンロックスクリプトが r パズルに対する解を含むという要件を含む。 r パズルは、楕円曲線暗号（elliptical curve cryptography (ECC)）署名の r 部分に基づき Tx_1 のロックスクリプト内で定義されるチャレンジを含む。これは、 Tx_2 のアンロックスクリプトに彼らの署名（又は少なくともその s 部分）を含む任意のパーティ（この場合にはたまたまBobである）により満たされることができる。 Tx_0 のロックスクリプトと異なり、任意のパーティの署名は、それが r チャレンジ（つまり、 r パズル）を満たす有効な署名である限り、 Tx_1 のロック条件をアンロックするために使用できる。この例は、間もなく更に詳細に議論される。Bobは、単に、証明者又は第2パーティの例としてここで選択されたが、 r パズルは、実際には、任意の第2パーティ、例えば、Charlie、Dora、Ezekiel、等が証明者になることを許容する。幾つかの実施形態では、 Tx_1 内のアンロック条件は、1つ以上の更なる条件を条件としても生成され得る。例えば、Aliceの署名が Tx_2 のアンロックスクリプトにも含まれることを要求する。

【0087】

第2トランザクション Tx_2 は、少なくとも1つのアウトプット2022（例えば、 Tx_2 のアウトプット0）を有し、該アウトプットは、Bobへ移転すべきデジタルアセットの量、及びこれをBobに対してロックするロックスクリプトを指定する（つまり、これは、更なる、今後のトランザクションが、使用するためにアンロックスクリプトの中にBobの署名を含むことが要求される）。この意味で、ターゲットトランザクション Tx_2 のアウトプットは、Bobにより所有されると言える。このアウトプットは、ここでも UTXOと呼ばれてよい。

【0088】

証明者の署名（例えば、Bobの場合にはSig P_B ）により署名された Tx_2 の部分は、少なくとも、このアウトプット2032、つまり証明者への支払いをロックするアウトプット（本例では、 Tx_2 のアウトプット0）、を含む。

【0089】

実施形態では、 Tx_1 内のアウトプット2031内のロックスクリプトがアウトプットをアンロックするための複数の代替条件、例えば複数の代替の r パズルを定義することが可能である。この場合、 Tx_2 のインプット2022内のアンロックスクリプトは、それが代替アンロック条件のうちのいずれか1つを満たす場合、 Tx_1 のアウトプットをアンロックする。

【0090】

第0（つまり、ソース）トランザクション Tx_0 は、Alice、証明者（例えば、Bob）、又は第三者により生成されてよい。それは、標準的に、Aliceが Tx_0 のインプット内に定義された量を取得した先行するパーティの署名を要求する。それは、Alice、Bob、先行するパーティ、又は別の第三者によりネットワーク106へ送信されてよい。

【0091】

第1トランザクション（つまり、チャレンジトランザクション） Tx_1 は、Alice、証明者（例えば、Bob）、又は第三者により生成されてよい。実施形態では、それはAliceの署名を要求するので、それはAliceにより生成されてよい。代替として、それは、Bob又は第三者によりテンプレートとして生成され、次に署名のためにAliceへ送信されてよく、例えばサイドチャネル301を介して送信される。Aliceは、次に、署名付きトランザクシ

ンをネットワーク106へ彼女自身で送信し、又はそれをBob若しくは第三者へ送信してそれらをネットワーク106へ転送し、又は単に彼女の署名をBob若しくは第三者へ送信して、署名付きTx₁に構成してネットワーク106へ転送できる。Tx₁をネットワーク106へ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を介して実行されてよい。

【0092】

第2トランザクション（つまり、証明又は支払いトランザクション）Tx₂は、Alice、証明者（例えば、Bob）、又は第三者により生成されてよい。第1バージョンは証明者の署名及び／又はデータを要求するので、Bobにより生成されてよい。代替として、それは、Alice又は第三者によりテンプレートとして生成され、次に、署名するためにBobへ送信されてよく、例えば、サイドチャネル301を介してBobへ送信される。Bobは、次に、署名付きトランザクションをネットワーク106へ彼女自身で送信し、又はそれをAlice若しくは第三者へ送信してそれらをネットワーク106へ転送し、又は単に彼の署名をAlice若しくは第三者へ送信して、署名付きTx₂に構成してネットワークへ転送できる。

【0093】

トランザクションの異なる要素が生成され構成され得る種々の位置があり、それが直接に又は間接的にP2Pネットワーク106の最終的な宛先へと送信される種々の方法があることが理解される。開示の技術の実装の範囲は、これらのいずれに関しても限定されない。

【0094】

「Aliceにより」、「Bobにより」、及び「第三者により」のような語句は、本願明細書では、それぞれ「Aliceのコンピュータ機器102aにより」、「Bobのコンピュータ機器102bにより」、及び「第三者のコンピュータ機器102により」の短縮表現として使用されることがある。また、所与のパーティの機器は、該パーティにより使用される1つ以上のユーザ装置、又は該パーティにより利用されるクラウドリソースのような幾つかのサーバリソース、又はそれらの任意の組合せを含み得ることに留意する。それは、必ずしも動作が単一のユーザ装置上で実行されることに限定しない。

【0095】

<楕円曲線デジタル署名アルゴリズム (ELLIPTICAL CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA)) >

公開鍵暗号法は、多数の異なる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をセキュアにするための基礎として使用される。公開鍵暗号法の使用は、公開鍵暗号化及びデジタル署名方式を含む。公開鍵暗号法は、特定の関数が、計算するのは容易だが、何からの特定の知識が無くては逆算 (reverse) することが困難であるという原理に基づいている。そのような関数は、落とし戸関数 (trapdoor function) と呼ばれ、それを逆算するために必要な特定の知識は該関数の落とし戸 (トラップドア) と呼ばれる。計算するのが容易であることは、所与の入力 (又は入力のセット) について妥当な時間枠内で落とし戸関数を計算することが計算上実現可能であることを意味し、逆算が困難であることは、落とし戸の知識を有しないで結果から該入力 (又は複数の該入力) を推定することが計算上不可能であることを意味する。

【0096】

公開鍵暗号法の文脈では、鍵ペアは、公開鍵 (これは誰にでも自由に入手可能にできる) 及び対応する秘密鍵 (これは、特定のエンティティ又はグループにのみ知られているという意味で、秘密であると想定される) を意味する。公開鍵は落とし戸関数を定義し、対応する秘密鍵は該関数を逆算するために必要な落とし戸である。

【0097】

公開鍵暗号の文脈では、暗号化は、落とし戸関数に基づき (つまり、暗号化は「順方向」に実行され)、一方で、復号は、落とし戸が分かっているときにのみ実現可能である落とし戸関数の逆算に基づく (つまり、復号は、「逆方向」に実行される)。

【0098】

デジタル署名の文脈では、署名検証は、公開鍵を用いて順方向に実行され、署名生成は

10

20

30

40

50

、逆方向に実行され、秘密鍵を用いてのみ実行可能である。

【0099】

ブロックチェーンの文脈では、公開鍵暗号法に基づくデジタル署名は、トランザクションに暗号法で署名するため及び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。

【0100】

ECCは、楕円曲線の数学的特性を利用する形式の公開鍵暗号法であり、DSA (Digital Secure Algorithm) のような他の暗号方式に対して種々の利点を有する。

【0101】

「楕円曲線デジタル署名アルゴリズム (Elliptic Curve Digital Signature Algorithm (ECDSA))」は、デジタル署名生成及び検証のための基礎としてECCを使用するクラスのデジタル署名方式を表す。ECDSAの特定の原理は以下に概説される。

【0102】

数学的用語では、ECCは、素数位数の有限体上の楕円曲線の代数的構造を利用する。有限体は、要素の有限の集合、並びに、集合内の要素に適用されるとき通常の算術規則（結合法則、可換性、等）を満たす乗算、加算、減算、及び除算の関連する演算のセットを意味する。つまり、「通常」の意味で、加算、乗算、等を必要としないが、本質的に同じように振る舞う、演算である。

【0103】

楕円曲線演算：

ECCの文脈では、加算、減算、及び乗算演算は、それぞれ、本願明細書で「+」で示される楕円曲線点加算、本願明細書で「-」で示される楕円曲線点減算、及び本願明細書で「・」で示される楕円曲線点乗算である。加算及び減算演算は、それぞれ、楕円曲線上の2個の点に適用され、楕円曲線上の第3の点を返す。しかしながら、乗算演算は、スカラー及び楕円曲線上の単一の点に適用され、楕円曲線上の第2の点を返す。これに対して、除算は、スカラー上で定義される。

【0104】

説明を目的として、図6Aは、全ての実数値の2次元座標の集合である

【数4】

$\mathbb{R}^2, \mathbb{R}^2$

及び、

【数5】

$\mathbb{R}^2$ の要素を示す $(x, y) \in \mathbb{R}^2$

における楕円曲線 ε を示す。楕円曲線 ε は、次式を満たす点の集合である：

【数6】

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

【0105】

加算： ε の数学的特性は、楕円曲線 ε 上の任意の2個の点A、Bが与えられると、A及びBと交差する直線は ε とCと示される1個の追加の点でのみ再び交差し；A及びBの楕円曲線加算、つまりA+Bは、Cの「反射 (reflection)」として定義され、Cと交差する水平線を取ると、Cの反射は、該直線が交差する楕円曲線上の他方の点である。この定義は、A=B場合には、Aにおける ε の接線が ε と再び交差する点がCであるという変更により、当てはまる。この定義は、 ∞ と示される無限遠点を楕円曲線上の点として及びそこで任意の垂直線が楕円曲線と交差すると定義することにより（例えば、D及びEとラベル付けされた点が垂直方向及び水平方向に揃えられ、従ってD+E= ∞ である）、2個の点と交差する線が垂直である場合に当てはまる。

【0106】

減算／加算反数 (Subtraction/additive inverse)：上述の反射の定義は、任意の点に適用され、楕円曲線点減算の定義を提供する。 $A-B$ は、 A と B の反射との和である。 B の反射は、より正式には B の「加算反数」と呼ばれ、 $-B$ と示される。この表記を用いて、楕円曲線減算は、数学的表記で定義できる：

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

【0107】

ここで、図6Bにおいて、 $C=-(A+B)$ 及び $(A+B)=-C$ 。この定義の下で、 $D=-E$ であり、代数構造の一般的規則を反映すること、つまり、楕円曲線上の任意の点とその加算反数との楕円点加算が無限遠点であることにも留意する。つまり、

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

【0108】

無限遠点 ∞ は、より正式には、「単位元」と呼ばれる（通常の算術計算の平行と偏差の両方に留意する：通常の算術計算では、任意の数 a とその加算反数 $-a$ との和は0であり、0は通常の算術計算の単位元である）。単位元、通常の算術計算をミラーリングする ∞ の別の特性は、 ∞ 自体を含む E 上の任意の点 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、記述 $a+0=0$ と同様である）。

【0109】

乗算：楕円曲線点加算の定義から、楕円曲線スカラー乗算の定義は以下の通りである。楕円曲線点 A の整数 v との乗算は次式のように定義される：

【数7】

$$v \cdot A = \underbrace{A + \cdots + A}_v.$$

【0110】

つまり、 A のそれ自体との v 回の楕円曲線点加算である。

【0111】

注：楕円曲線スカラー乗算は、従来、楕円曲線点乗算とも呼ばれている。これらの2つの用語は、本開示において同じ意味を有する。

【0112】

除算／乗算反数 (Division/multiplicative Inverse)：除算の延安は、スカラーに関して定義される。スカラー v が与えられると、「乗算反数」はスカラー v^{-1} として定義され、その結果、以下の通りである：

$$vv^{-1}=1$$

【0113】

図6Aは、上述の演算の直感的視覚化を提供する。ここで、 E は、全部の実数：

【数8】

$$\mathbb{R}.$$

を含む無限体に渡り定義される。

【0114】

図6Bは、次式により定義される楕円曲線 E_n を示すので、上述の演算が、ECCの文脈で実際にどのように適用されるかをより厳密に表す：

【数9】

$$E_n: y^2 = x^3 + ax + b \pmod{p}$$

【0115】

ここで、 p は素数（素数モジュラス）であり、 mod はモジュロ演算を示す。上式を満たす点の集合は有限であり、それらの点のうちの1つを除き全部が白丸として図6Bに示され、残りの点は単位元 ∞ である。

10

20

30

40

50

【0116】

素数 p は、楕円曲線の定義の部分を形成し、自由に選択できる。楕円曲線が良好な暗号特性を有するためには、 p は十分に大きくなければならない。例えば、265ビットの p が特定のブロックチェーンモデルにおいて指定される。

【0117】

添え字「 n 」は、本願明細書では、以上に定義された点加算の下で楕円曲線点により形成されるグループの次数を表す（簡単に言うと、これは、楕円曲線 ε_n の次数と呼ばれてよい）。以下を参照する。

【0118】

言い換えると、 n はグループの次数であり、 p はフィールドの次数である。全部で n 個の楕円曲線点がある。楕円曲線上の各点は、2個の数／座標 (x, y) により表され、ここで、 x 及び y は、全部、範囲 $-(p-1), \dots, 0, \dots, (p-1)$ の中にある。

【0119】

図6Bの ε_n は図6Aの ε_n と同様に水平対称性であることが分かる。これは、素数ファイル（prime file）に対する楕円曲線の一般的特性であり、従って、 ε_n 上の点の加算反数の定義が依然として当てはまる。幾つかの点は水平方向に揃えられた対称点を有しない（例えば、 $(0, 0)$ ）。そのような点は、それら自体の加算反数である。

【0120】

ε_n 上の2点A及びBと交差する「直線」 $l_{A,B}$ は、有限点集合になり、小さな黒丸により示され、同様の幾何学的要件を満たし、楕円曲線スカラー乗算の定義が依然として当てはまる。図6Aと同様に、図6Bは、点 $C=-(A+B)$ の加算反数である点 $A+B=-C$ を示し、ここで直線 $l_{A,B}$ が ε_n と再び交差する。

【0121】

ε_n 上の2点の楕円曲線加算 $A+B=-C$ は、次式により代数的に定義できる：

【数10】

$$\begin{aligned} A &= (x_A, y_A), \\ B &= (x_B, y_B), \\ C &= (x_C, y_C) = -(A+B), \\ x_C &= (\lambda^2 - x_A - x_B) \bmod p, \\ y_C &= (\lambda(x_C - x_A) + y_A) \bmod p, \\ &= (\lambda(x_C - x_B) + y_B) \bmod p, \end{aligned}$$

ここで、

$$\lambda = (y_A - y_B)(x_A - x_B)^{-1} \bmod p \quad \text{if } A \neq B,$$

及び

$$\lambda = (2y_A)^{-1}(3x_A^2 + a) \bmod p \quad \text{if } A = B.$$

上述の目的のために、整数 v の乗算反数 v^{-1} の定義は、以下のように変更される：

【数11】

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

【0122】

つまり、整数 v の乗算反数は、 $v \bmod p$ のモジュラ反数である。

【0123】

$B=-A$ の場合は特別であり、単位元 ∞ の導入により解決され、上述のように、この場合、 $A+B=A+(-A)=\infty$ である。 $B=\infty$ の場合も特別であり、上述のように $A+\infty=A$ と表記して解決

される。

【0124】

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、その他の場合には同じままである。

【0125】

他の文脈では、関連するスカラー v の乗算反数 v^{-1} の定義は：

【数12】

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

10

【0126】

乗算反数が $\text{mod } n$ 又は $\text{mod } p$ に関して定義されるかは、文脈上明らかである。

【0127】

実際に、数が $\text{mod } n$ 又は $\text{mod } p$ として扱われるべきかを識別するために、以下のチェックが適用されてよい。

(1) EC点の座標を表す数であるか？

a) Yesの場合、 $\text{mod } p$

(2) EC点を乗算するために使用される数であるか？

a) Yesの場合、 $\text{mod } n$

両方のチェックが肯定的結果を与えた場合があることに留意する。この場合、その数は $\text{mod } p$ 且つ $\text{mod } n$ でなければならない。

20

【0128】

<楕円曲線暗号法 (Elliptic Curve Cryptography (ECC)) >

楕円曲線演算は、シークレット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する。特に、有限体に渡る楕円曲線点のスカラー乗算を逆算することは、至難問題である（実行することが計算上不可能である）。

【0129】

秘密鍵 V は整数の形式をとり、対応する公開鍵 P は、楕円曲線 E_n 上の点である「生成元 (generator point)」 G から導出される、楕円曲線 E_n 上の点 P であり、次式の通りである：

30

【数13】

$$P = V \cdot G = \underbrace{G + \dots + G}_V$$

【0130】

ここで、「 $\cdot$ 」は、 a 、 b 、及び n （楕円曲線パラメータ）により定められる、楕円曲線 E_n 上の楕円曲線スカラー乗算を示す。

【0131】

十分に大きな V について、 P を導出するために実際に V 回の楕円曲線加算を実行することは困難である、つまり計算上不可能である。しかしながら、 V が知られている場合、 P は、楕円曲線演算の代数特性を利用することにより、遙かに効率的に計算できる。 P を計算するために使用できる効率的なアルゴリズムの例は、「Double and Add」アルゴリズムである。重大なことに、これは V が知られている場合にのみ実施できる。

40

【0132】

反対に、 V が分からない場合、 G 及び P の両方が分かっていたとしても、 V を導出する（つまり、スカラー乗算を逆算する）計算上実行可能な方法は存在しない（これは、所謂「離散対数問題」である）。攻撃者は、 G から開始して、 P を得るまで楕円曲線点加算を繰り返し実行することにより、「力づくで」 P を破ろうとし得る。この点において、攻撃者は、 V が、彼が実行すべき楕円曲線点加算の数であることを知っているが、それは計算上不可能であることが分かる。従って、 V は、上述の意味において落とし戸の要件を満たす。

50

【0133】

ECCでは、公開鍵P、生成鍵G、及び楕円曲線 ε_n は、公開されており、知られていると想定されるが、秘密鍵Vは秘密である。

【0134】

<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では、ユーザ又は他のエンティティは、標準的に、彼らのアイデンティティを証明するために使用される秘密鍵Vを保持し、対応する公開鍵Pは次式により計算され得る：

$$P = V \cdot G$$

秘密鍵Vは、ECDSAを用いてデータのピースm（「メッセージ」）に署名するために使用できる。

10

【0135】

ECDSAの更なる詳細は、参照によりその全体が本願明細書に組み込まれる次の文献で与えられる："RFC 6979-Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", Tools.ietf.org, 2019。

【0136】

図6Cは、公開鍵－秘密鍵ペア (V,P) についてECDSA署名 (r,s) を生成する署名生成機能（署名生成器600）の概略機能ブロック図を示す。EDSA署名は、本願明細書ではそれぞれr部分 (r-par (r)) 及びs部分 (s-part (s)) と呼ばれる値のペアである。

20

【0137】

署名生成は、公開鍵Pを導出するために使用された同じ楕円曲線 ε_n 及び生成元Gに基づく。従って、楕円曲線パラメータa、b、及びn、並びに生成元Gは、署名生成器600への入力として示される。

【0138】

署名生成器600の一時鍵生成器602は、「一時」鍵 $k \in [1, n-1]$ を、つまり両端を含む $1 \sim n-1$ の範囲で生成する。

【0139】

r部分生成器604は、kから対応する一時鍵を以下のように計算する。

【0140】

$$R = k \cdot G$$

そして次に、計算される点のx座標を取り入れる（楕円曲線点のx座標を取り入れる処理を示す[]_xによる）。

30

【0141】

$$r = [R]_x$$

上式は署名のr部分である。

【0142】

s部分生成器606は、 $k \bmod n$ のモジュラ反数 k^{-1} を用いて署名のs部分を計算し（つまり、次式であり、先の説明を参照する）：

【数14】

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

40

$H(m)$ と示される（必要な場合にはトランケートされる）、メッセージmのハッシュは以下の通りである：

【数15】

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

【0143】

本例では、メッセージmは、トランザクション608に含まれるべき日を含む（本例で

50

は1つ以上のトランザクションアウトプット)。これは、メッセージmに署名する処理と呼ばれてよく、メッセージmは、トランザクションの署名済み部分と呼ばれてよい。

【0144】

メッセージm及び署名(r,s)は、また、トランザクション608の部分を形成する。本例では、署名(r,s)は、アンロックスクリプトの部分としてトランザクション608のインプットに含まれる。

【0145】

図6Dは、トランザクション608を検証する署名検証機能(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により実行される計算は、留意すべきことに公開されている同じ楕円曲線 E_n 及び生成元Gに基づく。

【0146】

署名は秘密鍵Vをインプットとして要求するが、つまり有効な署名を生成するためにその知識を要求するが、署名(r,s)を検証するためには、署名ペア(r,s)、メッセージm、及び公開鍵Pしか必要ない。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の署名済み部分をハッシングする(署名(r,s)を生成するために使用されたのと同じハッシュ関数Hを適用する)。検証処理は、次に、以下の計算を用いて実行される。

【数16】

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

【0147】

[R']=rの場合及びその場合にのみ、署名は有効である(つまり、署名検証器が成功する)。その他の場合、無効である(つまり署名検証が失敗する)。本例では、rは、トランザクション608に含まれる署名のr部分を示す。

【0148】

署名検証器処理で使用される公開鍵Pは、例えば、先行するトランザクションのロックスクリプト内で指定される。署名検証は、この場合には、先行するトランザクションのロックスクリプト内で指定された公開鍵、(後の)トランザクション608の署名された部分m及び署名(r,s)を用いて実行され、署名(r,s)が先行するトランザクション内で指定された公開鍵Pに対応する秘密鍵V及びのりのトランザクション608の署名された部分mに基づき生成されたものでない限り、失敗する。従って、秘密鍵Vを保持する人物のみが、(標準的には、彼ら自身の公開鍵を後のトランザクション608のアウトプットに含めることにより)先行するトランザクションのアウトプットを請求でき、後のトランザクション608の署名された部分mは署名(R,S)を無効にすることなく変更できない。

【0149】

Rパズル

以下は、ECDSAに基づく知識証明の新しい形式を記載する。説明により、チャレンジャーは、彼女自身でTx₁を生成しP2Pブロックチェーンネットワークに発行することにより、又はTx₁を構成し発行するための必要な詳細を第三者に提供することにより、第1トランザクションTx₁内でrパズルを設定する第1パーティAliceである。検証者(実際に証明を実行するパーティ)は、ネットワークのノード104のオペレータ、例えばマイナーである。rパズルの解は、ネットワーク106にTx₂を発行することにより、提供される。rパズルがアイデンティティに本質的に結び付けられないので、証明者は任意の第2パーティであり得る。しかし、例として、以下は、証明者がたまたまBobであるシナリオの観点で記載されることがある。証明者は、彼自身でTx₂を生成し発行するか、又は必要な詳細を第三者に提供してそれらをTx₂に構成し発行するようにしてよい。

【0150】

暗号ハッシュ関数は、インプットの小さな変更がアウトプットの予測できない変更をもたらす場合に、インプットを決定論的に不明確にする手段を提供する。従来のハッシュ関数は、MD5、RIPEMD-160、SHA-1、及びSHA-256[5]を含む。これらはそれぞれ

10

20

30

40

50

れ、衝突耐性（同じアウトプットを生成する2つのインプットを見付ける確率が極めて小さい）、及びプリイメージ耐性（ハッシュ値 $h=H(d)$ が与えられた場合に、インプット d を見付けることが極めて困難である）。

【0151】

従来のハッシュパズルは以下のように設定できる。考え方は、第1トランザクション Tx_1 を設定することである。第1トランザクション Tx_1 は、第2トランザクション Tx_2 がそのインプットに何らかの特定のデータのピースを含むことを条件として、自身のアウトプットが第2トランザクション Tx_2 により償還されることを可能にする。

【0152】

ブロックチェーントランザクションでは、単純に、以下のように、ロックスクリプト内でハッシュ値 h を用いて、第1パーティ（Alice）は、非標準トランザクション Tx_1 を生成し得る：

OP_HASH160 <h> OP_EQUALVERIFY

ここで、 $h=H_{puz}(d)$ であり、 H_{puz} は、パズルの中で使用されるハッシュ関数である（上述の例では、ロックスクリプトによると、このハッシュ関数はHASH160でなければならないが、他の実装では他の形式のハッシュ関数を使用され得る）。このロックスクリプトが含まれるUTXOを償還することは、後のトランザクションのアンロックスクリプト内にあるハッシュパズル解を要求する。従って、アドレスAddr_Bobを有する第2パーティのための支払いトランザクション Tx_2 は、 d を含むだけでよいアンロックスクリプトにより構成され得る。

【表1】

$TxID_2$	
Input	Output
0. $TxID_1$ Unlocking script: < d >	0. Address: $Addr_Bob$ Amount: {VALUE}

【0153】

ここで、 $TxID_i$ は Tx_i のトランザクションIDである。ロックスクリプトは以下を記述する： Tx_2 のインプット内のアンロックスクリプトからデータ値 d を取り込み、それをハッシュし、それが Tx_1 のアウトプット内のロックスクリプトに含まれるハッシュ値 h と等しいかどうかをチェックする。従って、アウトプットは、 Tx_2 のアンロックスクリプト内の d を提供することにより、アンロックされる。

【0154】

この素朴な例では、 Tx_2 内のハッシュパズル解を有するユーザのトランザクションが分かった後に、このトランザクションを最初に受信したマイナーは、悪意をも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解を有するが、彼ら自身のアドレスAddr_Minerにアウトプットを変更している、新しい柔軟な（malleated）バージョン Tx_2^* を生成することができる。悪意あるマイナーは、次に、彼／彼女自身で Tx_2^* をマイニングしてブロック151にすることができる。 Tx_2 がマイニングされる前に、彼らが Tx_2^* をマイニングすることに成功すれば、悪意あるマイナーはBobの代わりに支払いを受け取るだろう。

10

20

30

40

50

【表 2】

<i>TxID₂</i> *	
Input	Output
1. TxID1 Unlocking script: <d>	1. Address: <i>Addr_Miner</i> Amount: {VALUE}

10

【0155】

デジタル署名は、所有権を証明し未使用トランザクションアウトプット (unspent transaction output (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の中で一般的に使用されている。これは、Tx₁のようなトランザクションのアウトプットが、特定のパーティにロックされることを可能にする。最も一般的な例は、トランザクションのアウトプットが公開鍵の特定のハッシュ（これは、そのパーティのアドレスとしても機能する）にロックされるP2PKH (pay-to-public-key-hash) トランザクションである。公開鍵Pのロックスクリプトは：

20

OP_DUP OP_HASH160 <h_p> OP_EQUALVERIFY OP_CHECKSIG

【0156】

ここで、h_p=H_{sig}(P)であり、H_{sig}は、署名の中で使用されるハッシュ関数である（上述の例では、ロックスクリプトによると、このハッシュ関数はHASH160でなければならないが、他の実装では他の形式のハッシュ関数が使用され得る）。このUTXOを別のトランザクションへのインプットとして使用可能にするためには、Pを用いて有効なECDSA署名を有するアンロックスクリプトを提供する必要がある：

<sig> <P>

【0157】

文字列全体（アンロック+ロックスクリプト）は、マイナーにより評価される。マイナーは、正しい公開鍵が提供されるか、及び署名が有効でありPに対応するか、をチェックする。ロックスクリプトは、基本的に以下を記述する：Tx₂のインプット内のアンロックスクリプトから公開鍵Pを取り入れ、それをハッシングし、それがTx₁のアウトプット内のロックスクリプトに含まれるハッシュ値h_pと等しいかどうかをチェックし、更に、Tx₂の署名済み部分の知識が与えられた場合に、Tx₂のアンロックスクリプトから公開鍵Pを用いて、ECDSA検証関数に基づき署名sigを検証する。ECDSA検証関数はOP_CHECKSIGオペコードにより呼び出される。

30

【0158】

従って、アウトプットは、Tx₂のアンロックスクリプト内で、Pに対応する秘密鍵Vに基づき署名された有効な署名sigを提供することによってのみ、アンロックできる。

40

【0159】

これをハッシュパズルと一緒にすると、上述の脆弱性は、ハッシュパズル解と一緒に、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り、修正できる。ロックスクリプトは以下のように構成され得る：

OP_HASH160 <h> OP_EQUALVERIFY OP_DUP OP_HASH160 <h_p> OP_EQUALVERIFY P_CHECKSIG

【0160】

対応するアンロックスクリプトは次のようになる：

<sig> <P> <d>

【0161】

50

しかしながら、これは、それを償還できる人を、公開鍵Pの所有者に制限する。ここで、これは幾つかの適用では、例えば、Aliceがパズルを設定した後にのみ署名権限を選定する能力を保持したい場合、望ましくないことがあることが分かる。

【0162】

ここで、ハッシュパズル機能は、一時的なランダム値であってよいECDSA署名の中のr部分を利用することにより、エミュレートできることが分かる。ECDSA署名は、2つの部分r及びsで構成される。以上から分かるように、 $r=[k \cdot G]_x$ である。従来のハッシュパズル $h=H(d)$ の代わりに、楕円曲線加算の逆算の困難さは、本願明細書でrパズルと呼ばれる類似のパズルを形成できる。パズルを解くためには、解の値kを取得する必要がある、ここで、kは、rに対応する一時鍵である。

10

【0163】

従来のハッシュパズルでは、パズルを解くときに、ブロックチェーン上にdを開示するリスクがある。しかしながら、rパズルでは、kは決して開示されない。その代わり、rが開示され、署名と共にrから、kの知識が証明できる。

【0164】

kは固定サイズでなければならないが、ハッシュパズルのプリイメージデータは任意の長さになるので（そして、ハッシュ関数の1つの特徴は、インプットデータの長さに拘わらず、固定長の値を出力することである）、ハッシュパズル機能をエミュレートするために、rパズルの生成者は、先ず、何らかの多のプリイメージデータをハッシングして値kを取得してよい。例えば、256ビット長の秘密／一時鍵を使用する場合、kを得るために、rパズルに対するプリイメージデータはハッシングされなければならない。代替として、しかしながら、何らかの適切な長さ値のkは、単に選択され、それ自体の能力で直接シークレット値として使用され得る（つまり、何らかの他の選考するプリイメージからそれを導出する必要がない）。

20

【0165】

この方法は、支払い（spending）のためにECDSA署名を使用する任意のブロックチェーンシステムと共に使用できる。説明のために、以下は、UTXOに基づくモデルにおける例示的な実装を説明する。スクリプト言語では、OP_CHECKSIGオペコードは、スタック上で署名及び公開鍵を要求する（スタックの一番上には公開鍵があり、その直下に署名がある）。rパズルについて、スクリプトは、提供された署名の中のr値がrパズルチャレンジののために使用されたものと同じであることをチェックするよう構成される。言い換えると、スクリプトは、（OP_CHECKSIGを通じて）署名が公開鍵に基づき有効であることをチェックするだけでなく、署名が、ブロックチェーン上で事前に発行されているrパズルのr値を用いて生成されたことも確かめる。

30

【0166】

rパズルの幾つかの例示的な実装は、図7～10を参照して以下に議論される。それぞれの場合に、証明者、例えばBobは、Tx₂の部分に署名することにより、署名(r,s)を生成している。この形式の署名は、時に、「sig」と呼ばれることもある。暗号署名の文脈では、署名済み部分は「メッセージ」（m）とも呼ばれる。署名済み部分（メッセージ）mは、少なくとも、結果として生じるBobへの支払いをロックする、Tx₂のアウトプットを含む。1つより多くのアウトプットが存在する場合、mは、アウトプットのうちの一部又は全部を含んでよい。mは、使用される場合には、ロックタイム（locktime）のような他の部分も含んでよい。しかしながら、それは、アンロックスクリプト自体を除く（及び、勿論、少なくとも、署名自体を除く）。メッセージmとして署名されるべきTx₂の部分は、Sighashにより設定され、又はデフォルトであり、マラはプロトコルの固定的特徴であることもできる。

40

【0167】

おそらく、rパズルの最も簡単な実装は図7に示される。Tx₁内のロックスクリプトは、ここではr'とラベル付けされる、参照インスタンス又はr部分を含む。この方法では、Tx₂内のアンロックスクリプトは、少なくとも、Bobの署名のs部分（s）のみを含めばよい。

50

それは、Bobがmに署名するために使用した秘密鍵Vに対応する公開鍵Pも含んでよい。Tx₁のロックスクリプトは、ノード104でスクリプトエンジン402により実行されると、s及びPをTx₂のアンロックスクリプトから取り入れ、以下の演算を実行するように構成される。

【数17】

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

II) $[R']_x = r'$ であることをチェックする

10

【0168】

ここで、r'は、Tx₁のロックスクリプトから取り入れられ、s及びmは、Tx₂のアンロックスクリプトから取り入れられる。Bobの公開鍵Pも、Tx₂のアンロックスクリプトから取り入れられてよく、又は他の手段により知られていてよい。H_{sig}は、第1ECDSA署名を生成する際にmをハッシングするために使用されたハッシュ関数である。それは、任意の形式のハッシュ関数であってよい。それがどんな形式であっても、このハッシュ関数の形式（タイプ）は、所定の両者に知られているものであると考えられてよい。Gは固定の公衆に知られたベクトル値である。

【0169】

ロックスクリプトは、前記のチェックが真であることを条件に「真」の結果を返し、その他の場合に「偽」の結果を返すよう構成される。UTX0の場合には、アンロックスクリプトと一緒にロックを実行した真（つまり、成功）の結果は、トランザクションの有効性の要件である。従って、Tx₂の有効性は、rパズルの結果のプロキシとして使用できる。或いは、別の方法では、Tx₂の有効性は、rパズルに対する解を提供することを条件とする。つまり、Bobがrパズルに合格しない場合、彼のトランザクションTx₂は、ネットワーク106に渡り伝播されず、ブロックチェーン150に記録もされない（Tx₁のアウトプット内に定義されたいかなる支払いも償還されない）。

20

【0170】

図7の例は数学的意味において最も単純であるが、これは、必ずしも、任意の所与のノードプロトコル又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意味しない。支払者（spender）が、〈r,s〉及び〈P〉ではなく、アンロックスクリプトの中で〈s〉及び〈P〉しか提供しない場合、スクリプトはこれに対応しなければならない。演算I)~II)は、標準的なChecksigtタイプのオペコードの演算ではない。OP_CHECKSIGオペコードは、署名がDERフォーマットであることを期待しているので、〈s〉値のみがアンロックスクリプト内で提供された場合、有効な署名をDERフォーマットで生成するために、ロックスクリプト内にいくつかの追加のオペコード（連結するためにOP_CAT等）が存在する必要がある。図8は、数学的に言うとは追加ステップを含むが、実際には、両方ともTx₂のインプットから取り入れられるr及びsに基づくECDSA署名検証を呼び出すための専用オペコードを既に有するScriptのようなスクリプト言語と統合しているより簡単な代替の例を簡単に記載示す。

30

【0171】

全部の可能な実施形態においてTx₂にPを含むことは本質的ではないことにも留意する。実際に、メッセージm及び(r,s)又はこの場合には(r',s)の知識から、公開鍵の2つの可能な値P及び-P（しかしどちらがどちらかは分からない）を計算することが可能である。2つの検証は、次に、どちらが正しい方かを識別するために使用できる。又は代替として、2つの可能な解のうちのどちらを使用すべきかをシグナリングするための1ビットフラグがTx₂に含まれることが可能である。この後者のアプローチは、幾つかのアカウントに基づくプロトコルで現在使用されている。しかしながら、それは、スクリプト言語（例えば、Script）がP及び-Pを(r,s)及びmから計算する演算のためのオペコードを有しない現在のUTX0に基づくプロトコルでは使用されない傾向にある。しかしながら、演算

40

50

がロックスクリプト内に明示的にコーディングされること、又は導入される可能性を排除すべきではない。別の可能性は、Aliceが既に知っている、或いはPへのアクセスを有するか又はそれをサイドチャンネル301を介して受信することである。しかしながら、それは、PをTx₂にマッピングするために別のルックアップを必要とする。

【0172】

別の例示的な実装は図8に示される。ここで、rパズルは、Tx₂のアンロックスクリプトがr部分の提出されたインスタンスrを明示的に含むことを要求する。Tx₁のロックスクリプトは、r部分についてのテストを含み、該テストは、r部分の参照インスタンスr'が提出されたインスタンスrに対して比較されることを含む。この方法では、Tx₂内のアンロックスクリプトは、少なくとも、Bobの署名のr部分(r)及びs部分(s)を含まなければならない。それは、Bobがmに署名するために使用した秘密鍵Vに対応する公開鍵Pも含んでよい。Tx₁のロックスクリプトは、ノード104でスクリプトエンジン402により実行されると、r、s及びPをTx₂のアンロックスクリプトから取り入れ、以下の演算を実行するように構成される。

【数18】

I) $r' = r$ であることをチェックし、

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

III) $[R']_x = r$ であることをチェックする

【0173】

ここで、r'は、Tx₁のロックスクリプトから取り入れられ、s、r、及びmは、Tx₂のアンロックスクリプトから取り入れられる。Bobの公開鍵Pも、Tx₂のアンロックスクリプトから取り入れられてよく、又は他の手段により知られていてよく、例えば(r,s)及びmから導出され、又は前述のような(r,s)及びmであってよい。

【0174】

ロックスクリプトは、ステップI)及びII)の両方のチェックが真であることを条件に「真」の結果を返し、その他の場合に「偽」の結果を返すよう構成される。UTXOに基づく場合にも、これは、トランザクションの有効性がrパズル知識証明の結果に依存して決定されることを可能にする。番号I~IIIは、必ずしも順序を意味しないことに留意する。III)はII)の後に実行される必要があるが、チェックI)はII)~III)の前又は後に実行できる。

【0175】

図8の方法では、ステップII)及びIII)は、単独で、ECDSA検証関数により実行される従来の演算である。大部分のプロトコルでは、それらは、従って、Scriptにおける既存のChecksigtオペコード(OP_CHECKSIG)のような専用オペコードにより呼び出される。ステップI)は、汎用オペコードを用いてロックスクリプトに別個にコーディングできる(例は後述する)。ステップII)及びIII)がChecksigtのような専用オペコードを用いる代わりに、汎用オペコードを用いて明示的に符号化されることも原理的に除外されない。

【0176】

1つの例示的なトランザクションプロトコルでは、図12に示すように、トランザクションECDSA署名は、ASN.1(Abstract Syntax Notation One)DER(Distinguished Encoding Rules)符号化フォーマットを使用する。第1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。次のバイトフィールドは、シーケンスの長さを16進数で含む。第3バイトフィールドは、ASN.1整数(integer)を示すフラグ0x02を含む。その後、ECDSA署名のr値が、次の32又は33バイトに含まれる。フィールドは32バイトであるべきだが、rの第1バイトが0x7fより大きい場合(第1ビットは1である)、0の追加バイトがr値の前に追加され、33バイトの長さにする。これは、整数

の第1ビットを署名として解釈するDERフォーマット符号化の結果として行われる。0の追加バイトは値の始めに追加され、その結果、負の値として解釈されない。同じことが、ECDSA署名のs値に行われる。最終的に、1バイトフィールド、ハッシュタイプ(ht)が、DER符号化に追加され、これはトランザクション内のビットコイン署名のタイプ(SIGHASH_ALL、SIGHASH_NONE、等)に対応する。

【0177】

Alice (A) が、パズルの解を取得した者が誰でも使用できるrパズルトランザクションを生成したい場合を検討する。これを達成するために、彼女は、以下に一例を示すような新しいトランザクションTx₁を生成する。インプット部分は、使用されている前のトランザクションTx₀のアンロックスクリプトを含む。簡単のため、それは、Aliceの署名及び公開鍵を用いて使用される標準的なP2PKHであると仮定する。アウトプット部分は、ロックスクリプト(スクリプト公開鍵)、又は言い換えるとrパズルチャレンジを含む。図12に示すように、署名は、幾つかのプロトコルではDER符号化フォーマットを使用してよい。従って、スクリプトは、符号化署名からrの値を抽出し、次に、それが<r>に等しいかをチェックしなければならない。その後、スクリプトは、署名が公開鍵に基づき有効であることをチェックしなければならない。スクリプトがどのように動作するかの詳細な説明は図5に示される。太字のオペコードは、基本的に、署名からrを抽出する方法である。

【表3】

TxID1	
Inputs	Outputs
Any spending inputs	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

【0178】

対応するアンロックスクリプトは、以下に示される。ここで、署名sig_rはrを使用し、支払者Bob (B) は任意の秘密／公開鍵ペアを用い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>

【0179】

図13は、ステップ毎のスクリプト分析を示す。

【0180】

一時鍵kは、Aliceにより生成され、Bob (及び任意的に1人以上の他の可能な証明者) に与えられてよい。代替として、kは、Bobにより生成され、Bob (又はkを共有するためにBobが選択した誰か) だけが解くことができるrパズルを設定するためにAliceに与えられてよい。いずれの場合にも、証明者Bobは、送信者Aliceはrパズルの解(k)を知っているので、彼女がトランザクションを彼女自身で使用しないことを信じなければならない。これを防ぐために、証明者Bobは、パズルを生成し、Aliceがrパズルトランザクションを生成するときに使用するために、Aliceにr値を送信し得る。その後、Bobは、彼がrパズルの解であり鍵の一形態として見られる値kを保持している限り、任意の秘密／公開鍵ペアを用いて後日、アウトプットを償還できる。他方で、幾つかの場合には、Aliceがkを知っているという事実は、有利な特徴になり得る。例えば、これは、秘密鍵パズルを生成するために、それを通じて一般化されたアトミックスワップ(atomic swap)を生成するために使用できる。

【0181】

図9は、rパズルの別の例を示す。これは、P2PKH (pay to public key hash) と同様に、本願明細書で「P2RPH (pay to r-puzzle hash)」と命名される。追加されるセキュリティ及びプライバシーのために、r値は、(ネットワーク106のノード104を

通じて伝播されブロックチェーン150上に置かれる) Tx₁内に置かれる前に、ハッシングされ得る。P2PKHと同様に、公開鍵自体ではなく、公開鍵のハッシュのみがブロックチェーン上にある場合、rパズルにより同じことが行われる。

【0182】

ここでも、rパズルは、Tx₂のアンロックスクリプトがr部分の提出されたインスタンスrを含むことを要求する。Tx₁のロックスクリプトは、ここでも、r部分についてのテストを含むが、今回は、r'のハッシュの形式のr部分の圧縮されたインスタンスの形式である。つまり、 $h = H(r')$ 。これは、提出されたインスタンスrに対して比較される。この方法では、Tx₂内のアンロックスクリプトは、ここでも、少なくとも、Bobの署名のr部分(r)及びs部分(s)を含まなければならない。それは、Bobがmに署名するために使用した秘密鍵Vに対応する公開鍵Pも含んでよい。Tx₁のロックスクリプトは、ノード104でスクリプトエンジン402により実行されると、r、s及びPをTx₂のアンロックスクリプトから取り入れ、以下の演算を実行するように構成される。

【数19】

I) $h = H_{puz}(r)$ であることをチェックし、

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

III) $[R']_x = r$ であることをチェックする

【0183】

ここで、hは、Tx₁のロックスクリプトから取り入れられ、s、r、及びmは、Tx₂のアンロックスクリプトから取り入れられる。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(hash-of-r puzzle)で使用されるハッシュ関数である。それは、任意の形式のハッシュ関数であってよい。それは、 H_{sig} と同じ又は異なる形式のハッシュ関数であってよい。それがどんな形式であっても、 H_{puz} の形式は、所定の両者に知られているものであると考えられてよい。Bobの公開鍵Pも、Tx₂のアンロックスクリプトから取り入れられてよく、又は他の手段により知られていてよく、例えば(r,s)及びmから導出され、又は前述のような(r,s)及びmであってよい。

【0184】

ロックスクリプトは、ステップI)及びII)の両方のチェックが真であることを条件に「真」の結果を返し、その他の場合に「偽」の結果を返すよう構成される。III)はII)の後に実行される必要があるが、チェックI)はII)～III)の前又は後に実行できる。

【0185】

ここでも、図8の場合のように、ステップII)及びIII)は、単独で、ECDSA検証関数により実行される従来の演算である。大部分のプロトコルでは、それらは、従って、scriptにおける既存のChecksigtオペコード(OP_CHECKSIG)のような専用オペコードにより呼び出される。ステップI)は、汎用オペコードを用いてロックスクリプトに別個にコーディングできる。

【0186】

トランザクションチャレンジTx₁内のロックスクリプトの例は以下に示される。

【表4】

TxID1	
Inputs	Outputs
Any spending inputs	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_HASH160 (h) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

【0187】

送信者及び受信者の両方のパーティの間で一貫している任意のタイプのハッシュ関数が使用され得る。しかしながら、P2PKH標準と調和していながら、私たちは、OP_HASH160、SHA-256のダブルハッシュ、及びRIPEMD-160を使用する。

【0188】

対応するアンロックスクリプトは、以下に示される（前の章と同じである）。ここで、署名 sig_r は r を使用し、支払者Bob（B）は任意の秘密／公開鍵ペアを用いて署名を計算できる。

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

【0189】

従って、図9の例は、 r の未変換インスタンスの代わりに、 r 部分のハッシュを r チャレンジの基礎として使用することを除き、図8と同様である。

【0190】

これらの場合のいずれでも、 Tx_1 のアンロックスクリプトが「真」の結果について追加基準を課すことに留意する。例えば、例は、ロックタイムまたは追加署名に対する要件であり得る。

【0191】

上述の技術のいずれかの例示的な使用例は、汎用知識チャレンジである。何らかの解 k を有する任意のチャレンジ、又はハッシングされて k になる何らかの解を検討する。Aliceは、パズルに結合される r パズルを生成する。つまり、彼女は、 $r = [k \cdot G]_x$ を定義できる。

【0192】

例として、Alice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 を構成できる。ここで、基礎にある k 値は、学生が解くように促される数学の問題に対する解である。解を考え出すことができた者は誰でも、署名 (r, s) を生成するためにそれを使用でき、従って報酬を請求できる。ここで、 r はロックスクリプト内の値と一致する。署名は、真正さを提供するだけでなく、解を誰か他の者に開示することなく、解の知識証明としても機能する。従って、 r パズルは、何からの解の知識又は情報を、それを公開するリスクを伴わずに一般的に証明するためのセキュアなメカニズムを提供する。それは、アンロックスクリプト内で要求される署名をエレガントに再利用し、任意の公開鍵 P が使用できるので、解を見つけた者が誰でもプライバシーと共に報酬を請求できるようにする。

【0193】

この方式は、トークン又はデジタルチケットの形式として使用されることもできる。例えば、イベント主催者は、デジタルチケットとして k の異なる値を、出席者に発行できる。出席者がイベントに出席したいとき、彼らは、 r パズルの使用を通じてシークレットトークンの知識を証明できる。

【0194】

別の例示的な使用例として、 r パズルは、あるパーティが別のパーティに署名する権利を委任できる署名権限付与方式として使用できる。ロックスクリプトに一致する r 値を有する署名が提供された場合にのみアンロックできる r パズルトランザクション Tx_1 を検討する。これは、値 k 、ここで $[k \cdot G]_x = r$ 、を知っている人物だけがそのような署名を生成できることを意味する。しかしながら、人物が k の知識を誰か他の者に渡した場合、これは、事実上、彼又は彼女の代わりに署名する権利を他の人物に与える。

【0195】

例えば、Aliceは配達を受け取りたいとする。彼女は彼女が配達を受け取るためにそこに居ないかもしれないことを心配している。彼女は、Bob及びCharlieの両者に k のコピーを与え、彼らは彼女の代わりに配達を受け取ることができる。Daveが小包を配達している場合、彼女は、Bobに小包を解放するために、期待される r 値を有する署名を得なければならない。

【0196】

このようなシナリオでは、 k は一時秘密鍵として、 r は一時公開鍵として、 k 及び r が特定のアイデンティティにリンクされないことを除き、それぞれ V 及び P と同様に、動作すると

10

20

30

40

50

考えることができる。

【0197】

<共同値 (JOINT-VALUE) rパズル>

図9のハッシングされたrパズル (P2PKH) への拡張として、($h = H_{\text{puz}}(r \parallel d)$ を得るために) ハッシングの前にrと連結される追加の値dを含むことが可能である。その場合、証明者 (例えば、Bob) は、rパズルを解くだけでなく、dも知っていなければならない。この例は図10に示される。

【0198】

Tx_1 のロックスクリプトは、ノード104でスクリプトエンジン402により実行されると、r、s、P及びdを Tx_2 のアンロックスクリプトから取り入れ、以下の演算を実行するように構成される。

【数20】

I) $h_{\text{joint}} = H_{\text{puz}}(r \parallel d)$ であることをチェックし、

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

III) $[R']_x = r$ であることをチェックする

【0199】

$r \parallel d$ は、いずれかの順序で (rが最初、又はdが最初)、r及びdの連結を表す。チャレンジトランザクション Tx_1 内のロックスクリプトの例は以下に示される。

【表5】

$TxID$	
Inputs	Outputs
Any spending inputs	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT
	OP_SWAP OP_SPLIT OP_DROP OP_2 OP_ROLL OP_CAT
	OP_HASH160 $\langle h_{\text{joint}} \rangle$ OP_EQUALVERIFY
	OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG

【0200】

対応するアンロックスクリプトは、以下に示される (dが含まれることを除き、前の章と同じである)。署名 $\text{sig}_r P_B$ はrを使用し、証明者Bob (B) は任意の秘密／公開鍵ペアを用いて署名を計算できる。

$\langle \text{sig}' \rangle \langle B \rangle \langle d \rangle \langle \text{sig}_r \rangle$

【0201】

追加の署名 sig' は、セキュリティのために追加された特徴である (後述する任意的なセキュリティ特徴についての章を参照する)。しかし、これは、全ての可能な実施形態で必要とされるものではない。

【0202】

例示的な使用例は、CLTVにリンクされたrパズル (CLTV Linked R-Puzzle) であってよい。この場合、データ値dは、CLTV (Check Lock Time Verify) トランザクションアウトプットにリンクされた時間値tであり得る。この背後にある動機は時間tを隠すことであり、この時間tの前には、P2RPHハッシュの中でアウトプットは使用できず、rパズルにリンクする。その場合、証明者 (例えば、Bob) は、rパズルを解くだけでなく、tも知っていなければならず、それを使用するためには特定の時間まで待機しなければならない

10

20

30

40

50

らない。トランザクション内のロックスクリプトの例は以下に示される。
【表 6】

TxID1	
Inputs	Outputs
Any spending inputs	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP
	OP_SPLIT OP_DROP OP_2 OP_ROLL OP_CHECKLOCKTIMEVERIFY
	OP_CAT OP_HASH160 $\langle h_{joint} \rangle$ OP_EQUALVERIFY OP_OVER
	OP_CHECKSIGVERIFY OP_CHECKSIG

【0203】

10

対応するアンロックスクリプトは、以下に示される。ここで、署名 sig_{rP_B} は r を使用し、支払者Bob (B) は任意の秘密／公開鍵ペアを用いて署名を計算できる。

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

【0204】

追加の署名 sig' は、セキュリティのために追加された特徴である（後述する任意的なセキュリティ特徴についての章を参照する）。しかし、これは、全ての可能な実施形態で必要とされるものではない。

【0205】

以上は、連結の観点で説明された。しかしながら、これを特定の関数 $f(r,d)$ へと一般化することも可能である。例えば、 f は r 及び d の加算であってよく、例えば $\langle r \rangle \langle d \rangle \text{OP_ADD}$ として実装できる。

【0206】

$\langle \text{複数の } r \text{ 値のステートメント} \rangle$

別の可能性は、 r の複数の所定の値、例えば、異なるステートメントに関連付けられそれらをアンロックする r_1 、 r_2 、及び r_3 を有することである。ステートメント S_i をそれぞれの r_i に割り当てた場合、私たちは、署名の中で対応する r_i を用いることにより、特定のステートメントに肯定応答できる。例えば、これは、合意した1又は複数の代替の可能な項目を承諾することを示すよう、署名するために使用されてよい。

【0207】

どの r 値がアンロックスクリプト内で使用されるかをチェックするロックスクリプトを構成することが可能であり、 r の値に解釈を割り当てることができる。上述の思想を実装するロックスクリプトは以下のようであってよい。

【数21】

OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP

OP_DUP OP_HASH160 $\langle H(r_1) \rangle$ OP_EQUAL

OP_IF

$\langle \text{Statement 1} \rangle$ OP_DROP OP_OVER OP_CHECKSIGVERIFY

OP_ELSE

OP_DUP OP_HASH160 $\langle H(r_2) \rangle$ OP_EQUAL

OP_IF

$\langle \text{Statement 2} \rangle$ OP_DROP OP_OVER OP_CHECKSIGVERIFY

OP_ELSE

OP_HASH160 $\langle H(r_3) \rangle$ OP_EQUAL

OP_IF

$\langle \text{Statement 3} \rangle$ OP_DROP OP_OVER OP_CHECKSIGVERIFY

OP_ENDIF

OP_ENDIF

OP_ENDIF

OP_CHECKSIG

【0208】

全ての $\langle \text{statement } i \rangle$ は、対応する r パズルを解いた後にのみアクセスできる異なるロック条件により置き換えられるべきである。アンロックスクリプトが以下に示され、ここで、 r_i は、設定されたステートメントにアクセスするために必要な異なる r 値である。

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

【0209】

追加の署名 sig' は、ここでも、セキュリティのための、任意的に追加された特徴である（以下を参照する）。しかし、これは、全ての可能な実施形態で必要とされるものではない。

【0210】

$\langle r$ パズル順方向チェーン $\rangle$

トランザクション $Tx(i+1)$ を消費するための十分条件が、トランザクション $Tx(i)$ の消費条件により要求されるものと同じ知識、及び知識のもう1つの追加のピースである、一連のトランザクションを検討する。例えば、 P_0, R_0 （の秘密鍵部分の）知識が、 $Tx0$ のアウトプットを消費するために必要とされる場合、 P_0, R_0, R_1 （の秘密鍵部分の）知識は、 $Tx1$ のアウトプットを消費するために十分である。ある意味、トランザクションは、順方向チェーンの中で互いにリンクされる。 $[R_i]_x = r_i$ であることに留意する。例示的な使用例は、トークン r_i が何かに対するアクセストークンである場合であってよい。ここで、順序が重要であり、例えば、各トークンは、例えばメディアストリームの異なるそれぞれの部分へのアクセスを可能にする。

【0211】

図14は、一連のトランザクションの例を示す。

【0212】

例えば、Aliceがトークン $r_0, r_1, r_2, \dots$ を所有し、トークン発行者はハッシュ $H(r_0), H(r_1), H(r_2), \dots$ の知識を有するとする。トークン r_i は、それをブロックチェーン上で開示すること、及び k_i の知識証明を提供するにより、アクティブ化されることが合意されている。

【0213】

Aliceは、上述のようなトランザクションのシリーズを生成することにより、彼女のトー

10

20

30

40

50

クンをアクティブ化してよい。AliceがトランザクションTx0をブロードキャストするとき、彼女は次のトランザクションの中でトークン r_0 をアクティブ化することを約束する。彼女がTx1をブロードキャストするとき、彼女は、 r_0 を開示し、及びトークンをアクティブ化する k_0 の知識証明を提供する。

【0214】

この方法の利点は、他の誰も、 k_0 の知識を証明できないので、Aliceのトークン r_0 をアクティブ化できないことである。また、Aliceの公開鍵 P_0 と各トークンのアクティブ化との間に階層的な認証リンクがある。更に、セキュリティを維持しながら、トランザクションのサイズは絶対的な最小値に保たれる。これは、彼女のトークンのアクティブ化が彼女のデジタル署名に組み込まれることにより、達成される。

10

【0215】

トークン r_i は予め生成される。 r_i の実際の値同士の間特定の関係が存在する必要はない。また、図14Aに示される公開鍵 $P_{i+1} = P_i + R_i$ 同士の間関係は必須ではないことに留意する。代替として、これは、任意の又は自由に選択された公開鍵により行うことができる。図14Aに示した関係は、公開鍵を生成するための単に便利な方法である。

【0216】

<任意的なセキュリティ特徴#1>

k に基づく署名が公開された場合、 k の値を知っている者は誰でも、署名を生成するために使用されたシークレット鍵 V の値を導出できる。これは、以下の署名式の中の V について解くことにより行うことができる。

20

【数22】

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

【0217】

V について解くと、以下を得る：

【数23】

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

【0218】

多くの場合にトランザクションの受信者は k を知っている者だけなので、これは有意なリスクを引き起こさない。他の場合には、支払者は、 r パズルの解に署名するために使用された秘密鍵 V を決して再利用しないよう注意しなければならない。良好なセキュリティの慣習は、ユーザが公開／秘密鍵ペア (P, V) を決して再利用しないことが望ましく、むしろ、新しい金銭を受け取るとき、常に新鮮な新しい公開／秘密鍵ペアを使用する。

30

【0219】

原則的に、公開－秘密鍵ペア (P, V) は「永久的」である。つまり、それは、何回も使用できる。ランダム一時鍵 k の使用はこれを保証すべきである。しかしながら、乱数生成器の実装が不十分である場合には、問題が起こる。

【0220】

同じ一時鍵 k 及び同じ秘密鍵を用いて2つの異なるメッセージに署名した場合、2つの署名から秘密鍵 V を導出できる。つまり、所与の (r, s) 及び k が与えられると、 V を算出できる。ここで、 $r = [k \cdot G]_x$ であり、 V は署名で使用された公開鍵 P に対する秘密鍵である。乱数生成器が署名処理中に障害になった場合、最後に同じ乱数を生成することがあり、従って、秘密鍵が公衆に漏洩する。この問題を解決するために、人々は、乱数生成器を固定する代わりに、公開鍵を再利用することを避け始めた。

40

【0221】

本例では、Aliceが k を知っているが、彼女が、Bobの公開鍵に対する秘密鍵である V を知らない場合。AliceがBobに k を渡すとき。Bobは、彼の秘密鍵を用いて (r, s) を提供することにより、 r パズルを解くことができる。Aliceは署名を見ると、彼女は k を知ってい

50

るので、彼女はVを導出できる。これは、Bobにとって望ましくない場合がある。従って、Bobは、望ましくは(P,V)の再利用を回避すべきである。

【0222】

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

【0223】

これを解決するために、本願明細書に開示される実施形態によると、Bobは、対応する公開鍵 P_2 を有する別個の秘密鍵 V_2 を用いて、 Tx_2 にBobの追加署名 sig_2 を含めてよい。彼は、追加署名と一緒に P_2 も含める。従って、2種類の公開-秘密鍵ペアがある。第1種類は、1回限りの使用のためにオンザフライで生成されるものである。他の種類は、何らかの追加プロトコル、例えばHDウォレットに従い生成されるものである。Bobは、rパズル署名のために第1種類の鍵ペアを使用し、第2署名のために第2種類を使用できる。

【0224】

Aliceは、公開鍵とアイデンティティとの間のマッピングに基づき、Bobのアイデンティティ、例えばBobの正式名、ユーザ名、又はネットワークアドレスを検索するために、第2公開鍵を使用できる。マッピングは、例えば、公開鍵をアイデンティティにマッピングする公開データベースの中で利用可能にされ得る。或いは、マッピングは、単にAliceとBobとの間で予め合意され得る（例えば、Aliceのコンピュータ機器102aにプライベートに格納される）。

【0225】

ここで再び、署名権限の使用例を検討する。例えば、Aliceは、配達を受け取りたいが、彼女自身で配達を受け付けることが可能ではないかも知れない。彼女は、Bob及びCharlieの両者にkのコピーを与え、彼らは彼女の代わりに配達を受け取ることができる。Daveは、小包を配達している。彼は、期待されるr値を有する署名を得なければならない。ここで、彼の記録又は規則対応では、Daveは受取人のアイデンティティを検証する必要もあることを考える。

【0226】

Bobは配達を受け付けるためにそこに居るとする。Bobが彼の公開鍵及び署名をkに基づき生成した場合、Alice及びCharlieの両者は、Bobの秘密鍵Vを算出できる。これは、公開鍵が1回限りの使用のために指定された場合には、問題ではない。しかしながら、Bobがこの公開鍵を将来に彼のアイデンティティを証明するために必要とする場合には、理想的ではない。

【0227】

この問題を解決するために、実施形態は、 Tx_2 に、Bobを識別するために使用できるBobからのrパズルと独立した1つ以上の署名を含めてよい。例えば、追加署名及び対応する公開鍵 P_2 を、Daveが受け付けるのと同じトランザクション内のOP_RETURNアウトプット（未使用アウトプット）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クスクリプト内に追加OP_CHECKSIGを含めることである。トランザクション及び追加署名のために使用された公開鍵を閲覧することにより、Aliceは、誰が彼女の代わりに署名したかを教えることができる。

【0228】

幾つかの他の場合には、値kが使用する前に漏洩する可能性があるという心配があり得る。これを解決するために、Aliceはrパズルトランザクションに P_2 PKHを追加して、それをよりセキュアにすることができる。Aliceは彼女の署名権限をBobに委任したいとする。AliceがBobから1回限り公開鍵 P_2 を取得し、r値を指定するだけでなく追加公開鍵 P_2 も指定するrパズルトランザクションを生成する。

【0229】

Alice自身も署名できるようにするために、任意的に、Aliceは1-out-of-2 MultiSigを生成できる。ロックスクリプトの一例は以下に与えられる。

10

20

30

40

50

【表 7】

TxID	
Inputs	Outputs
Any spending inputs	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP <r> OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIGVERIFY OP_1 <Alice's PK> <Bob's P ₂ > OP_2 OP_CHECKMULTISIG

【0230】

Aliceはrパズルの解、つまり署名権をBobに渡すべきときを選択できるので、rパズルが一層の柔軟性を提供することに留意する。彼女は、トランザクションがマイニングされた後でも、渡すか渡さないかを決定できる。

10

【0231】

kが漏洩した場合、人々は、漏洩したkにより署名に署名するために使用される秘密鍵を発見できる。しかしながら、別の秘密鍵V₂、つまりBobを識別するために使用できる公開鍵にリンクされる秘密鍵がある。アウトプットを損傷させるために、攻撃者は、2つの独立したシークレットを取得しなければならない。これは、それらのうちの1つのみを損傷させることより遙かに可能性が低い。

【0232】

上述の例では、Tx₂のロックスクリプトは、従来のP2PKHを用いて、Bobの追加公開鍵P₂にロックされる（rパズルで使用されたものではなく、追加署名によりアンロックされる）。rパズル技術は、ユーザにとって追加の選択肢を可能にする。幾つかの適用では、証明者がアイデンティティに関係なくチャレンジを満たすことができるように、rパズルを使用することが望ましい場合がある。他方で、幾つかの他の適用では、ハッシュパズルとP2PKHの組合せが、依然として望ましい場合があり、rパズルはそれに関連して任意的に使用できる。これは、間もなく更に詳細に議論される。

20

【0233】

しかしながら、P₂に対応する追加署名がアイデンティティ検索及び／又はセキュリティのために必要であるが、P2PKHにおけるようにTx₁のロックスクリプトが特定の証明者のアイデンティティに予め結び付けられることがない場合、上述のロックスクリプトが相応して採用できる。つまり、対応する公開鍵P₂にOP_EQUALVERIFYではなく、追加署名にChecksigを単に含めることができる。

30

【0234】

<任意的なセキュリティ特徴#2>

上述の方法における別の可能なセキュリティ脆弱性は、署名の鍛造性（forgeability）である。これは、（ハッシュパズルと同様に）資金を請求しようとするマイナーにより利用されることがある。トランザクションを（支払者から）受信したマイナーは、支払者が元のトランザクションで使用したものと同一署名を使用しながら、トランザクションを変更して、自分自身に資金を送ることができる。これは以下のように行われる。

【0235】

$P = V \cdot G$ を、mにより示される元のトランザクションに署名して、以下のように署名（r, s）を得るために使用された公開／秘密鍵ペアとする。

40

【数24】

$$r = [k \cdot G]_x, \text{ 及び}$$

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

【0236】

そのトランザクションを使用するために、支払者は、以下のアンロックスクリプトを使用する。

50

〈P〉 〈r,s〉

【0237】

このトランザクションを受信したマイナーは、トランザクションを、以下の新しいアンロックスクリプトを使用して自分自身に資金を送信する、 m' により示される新しいものに変更できる。

〈P'〉 〈r,s〉

【0238】

ここで、 $P' = V' \cdot G$ は、公開／秘密鍵ペアであり、以下の通りである。

【数25】

$$V' = V + r^{-1}[H(m) - H(m')], \text{ 及び}$$

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

【0239】

マイナーは（ V を知らないのに） V' を知る必要がないことに留意する。検証処理は、次に、以下の計算を用いて実行される。

【数26】

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

【0240】

署名は、 $(R', s) = r$ の場合及びその場合にのみ有効であり、その他の場合に無効である。

【0241】

新しいトランザクション m' 及び新しいアンロックスクリプトにより、検証処理は以下の通りである。

【数27】

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

【0242】

（プライム表記（primed notation）は以前と異なる意味を持ち、この文脈では、プライム符号を付されることが参照インスタンスを表さないことに留意する。）

【0243】

この潜在的な脆弱性を解決するために、実施形態では、マイナーがシークレット鍵 V を知らない限り提供することができない別のメッセージ m_{sighash} に対するアンロックスクリプト内に別の追加署名 sig' を含めてよい。その場合、アンロックスクリプトは、以下の通りであってよい。

〈sig'〉 〈P〉 〈sig〉

【0244】

sig' は、異なるメッセージ m_{sighash} に対する署名であってよい。従って、メッセージを変更するために、私たちは、元のものとは異なる sighash フラグを使用するだけである（例えば、デフォルトフラグである SIGHASH_ALL の代わりに SIGHASH_NONE ）。また、 sig' は異なる値の r を使用しなければならない。その結果、それは秘密鍵を漏洩しない（何故なら、秘密鍵は、同じ一時鍵を使用する2つの署名から導出できるからである）。最後に、トランザクションは、以下に示すように末尾に別の OP_CHECKSIG を含む必要がある。

10

20

30

40

50

【表 8】

TxID	
Inputs	Outputs
Any spending inputs	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_CHECKSIG

【0245】

これは、同じ公開鍵Pをrパズルとして使用しなければならない。その結果、公開鍵Pに対する秘密鍵Vを知っている者だけが別の署名を生成でき、上述の攻撃は不可能である。

10

【0246】

攻撃者は、公開鍵を、攻撃者が秘密鍵の知識を有しない別の公開鍵で置き換えようとする。この攻撃を防ぐために、チャレンジは、秘密鍵の知識についても尋ねる。この場合、1つの署名は十分ではない。従って、2つの署名が要求される。両方の署名は、同じ秘密鍵の知識の証明として考えられる。チャレンジはそれらが異なる一時鍵を有することを要求するので、これはセキュアである。

【0247】

<P2PKH+P2PRPH>

rパズルが知識証明として使用されるのと同様に、P2PKHアウトプットも、P2PKHアウトプット内の公開鍵に対応する秘密鍵の知識の知識証明である。これは、基本的に、パズルkを、P2PKHアウトプット内の公開鍵 $P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G$ にマッピングする秘密鍵 V_{puzzle} で置き換えることにより行われる。rパズルの実施形態では、知識証明は、証明者により選択可能でありアイデンティティにリンクするために使用可能である公開鍵を伴う。P2PKHでは、通常の支払い署名及び公開鍵（これらは、シークレットパズルの知識を署名する）は、特定のアイデンティティにリンクするために、別の署名及び公開鍵を伴わなければならない。些細なことであるが、証明者は、以下に示すように、別のOP_CHECKSIGに対応するP2PKHアンロックに対する別の署名及び公開鍵を、ロックスクリプトに追加できる。

20

【表 9】

TxID	
Inputs	Outputs
Any spending inputs	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_CHECKSIG

30

【0248】

対応するアンロックスクリプトは、以下に示される。

【数 28】

40

$$\langle \text{sig} P_{ID} \rangle \langle P_{ID} \rangle \langle \text{sig} P_{\text{puzzle}} \rangle \langle P_{\text{puzzle}} \rangle$$

【0249】

$\langle \text{sig} P_{ID} \rangle \langle P_{ID} \rangle$ とパズル又はトランザクションとの間に暗号リンクが存在しないことに留意する。マイナー及び任意の者は、実際にそれらを、別の公開鍵に基づく別の署名で置き換えることができる。これは、マイナーが公開ハッシュパズルを傍受できることに少し似ている。マイナー又は傍受者は、（公開ハッシュパズルの場合のように）自分自身に資金を送信するようメッセージを変更することができない。しかしながら、彼らが $\langle \text{sig} P_{ID} \rangle \langle P_{ID} \rangle$ を、彼ら自身のアイデンティティにリンクされていたかのようにするよう彼

50

ら自身のものに置き換えることを止めるものは何もない。

【0250】

他方で本願明細書に開示される技術に基づき、P2PKH及びP2RPHの両方を同じスクリプト内で一緒に使用して、第2署名に暗号リンクを強制することが可能である。その結果、それは、上述の場合に傍受され置き換えられることができなくなる。

【表10】

TxID	
Inputs	Outputs
Any spending inputs	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_SWAP OP_SPLIT OP_DROP OP_HASH160 $\langle H(r_{\text{puzzle}}) \rangle$
	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

10

【0251】

ロックスクリプト内で、

【数29】

$$P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G \text{ であると同時に、 } r = [R]_x = [V_{\text{puzzle}} \cdot G]_x$$

【0252】

従って、それらは基本的に等価である。公開鍵が圧縮され又は圧縮されず、いずれの方法でもプレフィックスが先頭に追加されるので、それらは実際には等価ではない。以下に示すように、署名からr値を抽出するとき、スクリプトにプレフィックスを明示的に追加して、 $\langle H(P_{\text{puzzle}}) \rangle = \langle H(r_{\text{puzzle}}) \rangle$ を生成することも可能である。しかしながら、これは、実際にはあまり多くの利益を達成しない。

20

30

【表11】

TxID	
Inputs	Outputs
Any spending inputs	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_SWAP OP_SPLIT OP_DROP OP_PUSHDAT1 $\langle r - \text{prefix} \rangle$ OP_SWAP OP_CAT OP_HASH160 $\langle H(r_{\text{puzzle}}) \rangle$
	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

40

【0253】

(ここで、 $H(\dots)$ は、ロックスクリプトの概略表現において三角括弧 $\langle \rangle$ で示され、これ

50

は実際には、ハッシュ関数ではなく、ハッシュの値を表すことに留意する。三角括弧〈〉は、スタック内のこの値の位置を意味する。)

両方のトランザクションに対する対応するアンロックスクリプトは、以下に示される。

【数 30】

$$\langle P_{ID} \rangle \langle r_{puzzle}, s \rangle \langle sig P_{puzzle} \rangle \langle P_{puzzle} \rangle$$

【0254】

＜アカウントに基づくモデルにおける代替の実装＞

以上は、大まかに、アウトプットに基づくモデル（例えば、UTXOに基づくモデル）における実装の観点で説明された。しかしながら、これは限定的ではないことが理解される。図 11 は、アカウントに基づくモデルを使用する可能な代替の実装を示す。

【0255】

要するに、アカウントに基づくモデルでは、rパズル機能は、ユーザにより呼び出されるスマートコントラクト関数に含まれ得る。あるパーティは、スマートコントラクト内にrパズル値（又はハッシングされたrパズル値）を設定できる。次に、他のパーティは、その後スマートコントラクトに署名を提供し得る。

【0256】

UTXOブロックチェーンアーキテクチャでは、第1トランザクションのアンロックスクリプト内に埋め込まれた要件は、第2トランザクションが有効であるとして受け入れられブロックチェーンに記録されるためには、第2トランザクションのロックスクリプトにより満たされなければならない。この状況で、トランザクション検証処理の部分としてマイナーにより既に行われた作業を利用するので、これは有利である。この状況における具体例として、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、ブロックチェーンネットワーク全体に渡るノードにより検証されたことを意味し、また、そのロックスクリプトが特定の有用な要件を満たすことを意味する。関心のあるパーティは、彼ら自身のために、それらの要件が満たされるかどうかをチェックする必要がなく、トランザクションがブロックチェーンに記録されることに成功しているという事実により、彼らは、単に、それらの要件が満たされていると想定できる。これは、トランザクションが有効であるためには、スクリプトが完了すると「真」の結果を返さなければならないという事実に起因し（トランザクションが有効であるための他の要件が存在してもよい）、スクリプトが「偽」の結果を返した場合には（これは、本願明細書で使用される用語によると、例えばOP_VERIFYオペコードがスクリプトを終了したために、スクリプトが失敗した場合を含む）、トランザクションは無効である。

【0257】

しかしながら、他のブロックチェーンモデル（例えば、特定のアカウントに基づくアーキテクチャ）では、トランザクション有効性と実行トランザクションコードの結果との間の相互依存性は必ずしもミラーリングされない。例えば、特定のスマートコントラクトブロックチェーンでは、トランザクションは、それらがブロックチェーンプロトコルにより課される「基本的」有効性要件のセットを満たすならば、有効であり、従って、ブロックチェーンに記録するために受け入れられてよい。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埋め込まれた特定の要件を満たさない場合でも、依然として有効であるとして受け入れられ、ブロックチェーンに記録されてよい。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。

【0258】

第2トランザクションが、第1トランザクションにより生成されたスマートコントラクトアカウントにアドレスされているとすると、それは次に、該トランザクションにどのように応答するかを決定するためにスマートコントラクトコードへと落とされる。例えば、何らかの要件が乱されない場合にそれを無視し（又は偽の結果を返し）、一方で、要件が正しい場合には、スマートコントラクトアカウントの残額から減額されクレジットされた

10

20

30

40

50

デジタルアセットの額で、証明者に報酬を与える（又は真の値を返す）。ある意味、これは、ノードにより「暗示的に」実行される「プロトコルレベル」の処理、つまり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り課される有効性の要件を満たすかどうかを決定するトランザクションに対して実行される処理から、スマートコントラクト（エージェント）による、つまりスマートコントラクトコード内に明示的にコーディングされた、「エージェントレベル」の処理を抽象化する。従って、そのようなブロックチェーンアーキテクチャでは、トランザクションそれぞれのプロトコルレベルにおけるノードによる「有効／無効」の決定は、スマートコントラクトによりエージェントレベルで該トランザクションに関して返される「真／偽」の結果から分離されてよい。ここで、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てよいが、エージェントレベルでは偽の結果を返してよい。

10

【0259】

これは、トランザクションが有効であるために「スクリプトが真」の結果を返すことが必要であるUTXOアーキテクチャと対照的に、トランザクションは、スクリプトが終了した又はスタック上に真以外のものを残して完了した場合に、無効である。

【0260】

トランザクション有効性のための基本的要件のうちの1つは、トランザクションが有効な署名を含むことであってよい。従って、上述のUTXIの例では、署名はチャレンジトランザクション自体のコードにより（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、又は同じ方法で署名をチェックし、更に結果が真であることを検証し、否である場合にスクリプトが終了するOP_CHECKSIGVERIFYオペコードを用いて）検証されたが、代替のブロックチェーンアーキテクチャでは、署名は、上述の意味では暗示的に処理ノードにより検証されてよく、これは、トランザクションコード自体に署名チェックをコーディングする必要を回避できる。

20

【0261】

本願の文脈では、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有効であると考えられる。しかし、例えば、何らかの他の要件が満たされないので、アプリケーションレベルでは依然として偽の結果を返してよい。

【0262】

図11は、アカウントに基づくモデルによる、トランザクションを処理するノードソフトウェア400の代替案を示す。ノードソフトウェアはここでは400accとラベル付けされる。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のアカウントに基づくバージョンのノード104の各々に実装されてよい。アカウントに基づくノードソフトウェア400accは、アカウントに基づくプロトコルエンジン401acc、コントラクトエンジン402acc(スクリプトエンジン402と同様のもの)、アプリケーションレベルの決定エンジン404、及び1つ以上のブロックチェーン関連機能モジュールのセット405を含む。任意の所与のノード104で、これらは、（ノードの1つ以上の役割に依存して）マイニングモジュール405M、転送モジュール405F、及び格納モジュール405S、のうちの任意の1、2、又は3個全部を含んでよい。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、それらをノードプロトコルに従い処理するよう構成される。ノードソフトウェア400accは、それぞれのノード104のメモリに、複数のアカウントの各々のアカウント状態406も維持している。これらは、例えば、Alice、証明者（例えば、Bob）、及び／又はAliceと証明者との間で制定されるコントラクトに依存して借り方又は貸し方である別のパーティのアカウントを含み得る。コントラクトエンジン402accは、トランザクション内で受信したスマートコントラクトの結果に依存して、アカウント状態を変更するよう構成される。スマートコントラクトは、「エージェント」とも呼ばれる。

30

40

【0263】

図11は、図7～10に関して上述したものと同一又は同様のrパズル機能を実装し得るトランザクションTx₁^{acc}及びTx₂^{acc}のペアも示す。それぞれは、（ソースアドレスフィ

50

ールド内の) ソースアカウントアドレス1102、及び(宛先アドレスフィールド内の) 宛先アカウントアドレス1103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及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及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、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、Aliceによるチャレンジ(パズル)を含んでよい。それは、Aliceにより、又はAliceにより提供された詳細を用いてAliceに代わる第三者により生成されてよい。第2トランザクションTx₂^{acc}は、任意的に、ユーザの指定したペイロードデータを運ぶ1つ以上の手数料データフィールド1104を含んでよい。これ／これらは、証明者、例えばBobにより提供されたパズルに対する解の少なくとも部分を含んでよい。トランザクションTx₁^{acc}及びTx₂^{acc}は、更にAlice及び証明者によりそれぞれ署名される。各トランザクションは、それぞれのパーティの署名1105a、1105bも含む。

【0264】

トランザクションは、ネットワーク106に渡り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が有効か否かを自動的に検証する。つまり、これは、プロトコルエンジン401accの固有の機能であり、スマートコントラクト1101内で指定される必要がない。プロトコルエンジン401accは、従って、少なくともそれぞれの署名が有効であることを条件に、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。それは、満たされるべき、有効性のための1つ以上の追加条件を要求してもよい。有効ならば、アプリケーションレベル決定エンジン404は、それぞれトランザクションをマイニング及び／又は転送するよう、マイニングモジュール405M及び／又は転送モジュール405Fを制御するかを選択できる。

【0265】

そのようなアカウントに基づくモデルでは、Alice、Bob、及びスマートコントラクト自体は、異なるアカウントアドレスを有する別個のアカウントを割り当てられる。トランザクションは、そのソースアドレスフィールド内のアドレス「から」、その宛先アドレスフィールド内のアドレス「へ」、送信されとも言える。スマートコントラクト用にアカウントを生成するために、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、トランザクションの中でブロックチェーンにアップロードされる。そのようなアカウント生成トランザクションでは、宛先フィールド内の宛先アドレス1103は、ブロックチェーンにおいて以前に使用されることがないアドレスでなければならない。一旦、トランザクションが受け付けると、そのアドレスは、新たに生成されたスマートコントラクトアカウントのアドレスになる。その後、更なるトランザクションは、スマートコントラクトを「呼び出す」ためにそのアドレスへ送信されることができ、つまり、更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。「宛先」アドレス1103は、コントラクトを制定するために中間アドレスとして動作する。Aliceは、1つ以上の要件を指定するスマートコントラクトを生成するために、Tx₁^{acc}をそのアドレスへ送信する。Bobは、スマートコントラクトを呼び出すために、Tx₂^{acc}をその同じアドレスへ送信する。また、スマートコントラクトに、Tx₂^{acc}がそれらの指定された要件を満たすか否かを検証させる。「ソース」アドレス1102は、コントラクトに対するパーティであるユーザのアカウントを指定する。スマートコントラクトがTx₂^{acc}は指定された要件を満たすと決定した場合、スマートコントラクトは、自身の口座(アカウント)残高からデジタルアセットの額を控除し、Tx₂^{acc}内のソースアドレス1102bを有するアカウント(つまり、Bobのアカウント)の残高をその額だけクレジットさせる(credit、貸し方に記入する)(直感的には、Tx₂^{acc}を送信することにより、Bobは、事実上、(ソースアドレスフィールド内で識別される)彼のアカウントをクレジットするよう(宛先アドレスフィールド内で識別される)スマートコントラクトに依頼する)。

【0266】

プロトコルエンジン401accは、Tx₂^{acc}を受信すると、それが有効であることを条件に、Tx₂^{acc}内の宛先アドレス1103bと一致するアカウントを探す。Tx₁^{acc}が処理さ

10

20

30

40

50

れ、有効であるとする、そのアカウントは、 TX_1^{acc} のお陰で存在し、 TX_1 内で提供されたスマートコントラクトコードに関連付けられる。応答して、プロトコルエンジン401accは、 TX_1^{acc} からのスマートコントラクト1101を実行するよう、コントラクトエンジン402accを制御し、コントラクト内にどんな基準が定義されているかに依存して、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入れる。オペランドデータは、例えば、自由データフィールド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_2^{acc} からのオペランドデータが TX_1^{acc} のスマートコントラクト1101内に定義された1つ以上の基準を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内に定義された変更に従い、1つ以上のパーティ（Alice、証明者、及び／又は1人以上の第三者）のアカウント状態406を変更する。その他の場合、アカウント状態406に対するこの変更は行われない。しかしながら、幾つかのアカウントに基づくシステムでは、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ための条件ではない。従って、 TX_2^{acc} が TX_1^{acc} のスマートコントラクト1101内に設定された基準を満たさない場合、 TX_2^{acc} は、失敗したトランザクションのレコードとして、依然として伝播されブロックへとマイニングされる。それは、依然としてマイニング手数料ももらし得る（従って、プロトコルエンジン401は、依然として、パーティのうちの1つ及び勝者であるマイナーのアカウント状態406を変更してよい）。

【0267】

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内にコーディングされることができ、解は、 TX_2^{acc} のデータフィールド1104の1つ以上の中で提示できる。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の暗示的署名検証機能の一部は、例えば、図8～10の変形のうちの1つを実装するために、利用され得る。図8～10の場合には、ステップII）及びIII）は、 TX_2^{acc} の署名を検証するとき、プロトコルエンジン401accの陰関数であってよい（署名検証自体はプロトコルエンジン401accにより実装されるノードプロトコルの固有機能であることを思い出してほしい）。従って、 TX_1^{acc} のスマートコントラクト1101では、これの上にステップI）を積み重ねるだけでよい。スマートコントラクトは、I）の結果が真であるかどうか、及びプロトコルエンジン401accが TX_2^{acc} は有効であると示すかどうか、をチェックする。両方がYesである場合、それは、検証について「真」の全体の結果を宣言する。つまり、Bobは、rパズルにより設定されたチャレンジを満たすことに成功する。図8～10の実装のうち、図9及び10の場合にデータ値dのみが、自由データフィールド1104に含まれる必要があるだけであることに留意する。署名情報は、署名フィールド1105bに含まれる。

【0268】

スマートコントラクトアカウントは、アカウントに関連付けられた（論理的）データ記憶要素である「データレジスタ」（図示しない）ともインデックスを付される。以上に概説したUTXOモデルでは、値は、ロックスクリプト自体に埋め込まれ、同じことがスマートコントラクトコード1101の特定のピースについても言える。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、代替として又は追加で、そのアカウントレジスタのうちの1つ以上に格納されたデータで実行されてよい。更に、通常、スマートコントラクトアカウントが生成された後に、スマートコントラクトアカウントレジスタに値を格納することが可能である。従って、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チャレンジトランザクション TX_1, α^{acc} により生成されてよい。別個の「中間」トランザクション TX_1, β^{acc} は、次に、（今や存在する）スマートコントラクトアカウントへ送信されてよく、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を格納する効果を有する。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、例えば第1の場所（Alice）でスマートコントラクトを生成した同じパーティからのそのようなデータのみを受け付けるよう構成されてよい。 TX_2^{acc} が受信されると、コントラクトエンジン402accにより実行される演

算（例えば、「レジスタ \$R にアクセスし、値を Tx_2^{acc} のデータフィールド内の値 \$D と比較する」）は、チャレンジトランザクション $Tx_{1,\alpha}^{acc}$ 内で提供されるスマートコントラクトバイトコードにより定義される。しかし、\$R に格納された値は、中間トランザクション $Tx_{1,\beta}^{acc}$ により設定されている。本願明細書で使用される用語によると、 $Tx_{1,\alpha}^{acc}$ は、依然として、1 つ以上の要件を設定するチャレンジトランザクションと言える。今や、それらの要件のみが、1 つ以上の中間トランザクション（例えば、 $Tx_{1,\beta}^{acc}$ ）内で提供されるデータを参照して定義されてよい。

【0269】

従って、幾つかの実装では、チャレンジトランザクション $Tx_{1,\alpha}^{acc}$ は、 r パズルの演算（例えば、証明トランザクション Tx_2^{acc} の署名の r 部分をレジスタ \$R 内の値と比較し、それらが一致するかを調べる、等）を定義してよい。しかし、証明トランザクション Tx_2^{acc} の r 部分と比較される \$R 内の値は、中間トランザクション $Tx_{1,\beta}^{acc}$ により送信されてよい。

【0270】

幾つかのアカウントに基づくモデルは、署名 1105 と一緒に公開鍵 P が含まれることを必要としないことにも留意する。代わりに、単に 1 ビットフラグ flg を含む。上述のように、 (r,s) 及びメッセージから 2 つの可能な鍵 P 及び $-P$ を導出することが可能である。フラグ flg は、これらの 2 つの可能な解のうちのどちらが、実際に、 Tx_2^{acc} においてメッセージに署名するために証明者により使用された秘密鍵 V に対応する公開鍵であるかをシグナリングするために使用される。プロトコルエンジン 401acc は、この (r,s) 及び flg を使用して、 Tx_2^{acc} の中で明示的に受信する代わりに、署名者の公開鍵 P を導出する。この技術は、アウトプットに基づくモデルでも可能であり、アカウントに基づくモデル専用ではない。しかし、多くの現在のアウトプットに基づくモデルで使用されるスクリプト言語では、 r 及び s から P を導出するための専用オペコードが存在しない。従って、この機能を、スタックに基づく言語の既存の汎用オペコードを用いてアンロックスクリプト内に明示的にコーディングすることは複雑になるだろう。更に、特定のアカウントに基づくモデルは、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該トランザクションのソースアドレスを導出することに留意する。従って、ソースアドレスは、必ずしも、トランザクション内で別個に符号化されず、公開鍵が署名から導出される場合には、これは、ソースアドレスも署名から間接的に導出され得ることを意味する。

【0271】

<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。

【0272】

より一般的には、本願明細書に開示される技術の第 1 の具体化によると、楕円曲線デジタル署名アルゴリズム (elliptic curve digital signature algorithm, ECDSA) 検証関数に基づき知識証明を実行する、コンピュータにより実施される方法が提供される。方法は、ブロックチェーンネットワークの検証ノードにおいて、

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であって、該コードは第 1 ECDSA 署名の r 部分の参照インスタンスを指定する要素を含む、ステップと、

第 1 ECDSA 署名の少なくとも s 部分を含む情報を含む第 2 トランザクションを受信するステップと、

第 1 公開鍵を取得するステップであって、第 1 ECDSA 署名は第 1 公開鍵に対応する第 1 秘密鍵に基づきメッセージに署名し、メッセージは第 2 トランザクションの部分である、ステップと、

前記第 1 トランザクションからの前記コードを実行するステップと、を含む。前記コードは、第 1 秘密鍵として誰の公開鍵が使用されたかに関係なく、ECDSA 検証関数が、第 1 ECDSA 署名に適用されるとき、第 2 トランザクション内で受信されたメッセージ及び取得された第 1 公開鍵が与えられると、第 2 トランザクション内で受信された s 部分が第 1 トランザクションにより指定された r 部分の参照インスタンスに対応することを検証すること

を条件として、真の結果を返すよう構成される。

【0273】

アウトプットに基づくモデル（例えば、UTXOに基づくモデル）では、ECDSA検証関数は、第1トランザクションのアウトプット（例えば、UTXO）のロックスクリプト内のオペコードにより呼び出されてよい。オペコードは、検証ノードに予め格納されたECDSA検証関数のインスタンスを呼び出してよい。代替として、アカウントに基づくモデルにおけるように、ECDSA検証関数は、ノードの陰関数（implicit function）であってよい。これは、第1トランザクション（アカウントに基づく場合にはスマートコントラクトであってよい）内のコードにより明示的に呼び出される必要があるのではなく、ノードプロトコルの部分として自動的に実行される。別の代替案は、ECDSA検証がコードに明示的にコーディングされ得ることを排除しない。

10

【0274】

本願明細書に開示される技術の第2の任意的な具体化によると、第1の具体化による方法であって、前記の要素が第1 ECDSA署名のr部分の参照インスタンスである方法が提供される。

【0275】

本開示の第3の任意的な具体化によると、第2の具体化による方法であって、前記ECDSA検証関数は、以下を実行し：

【数31】

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

20

$[R']_x = r'$ であることをチェックする

ここで、 r' は前記第1 ECDSA署名の前記r部分の参照インスタンスであり、 s は前記第1 ECDSA署名の前記s部分であり、 P は前記第1公開鍵であり、 m は前記第1 ECDSA署名により署名された前記第2トランザクションの一部であり、 H_{sig} は前記第1 ECDSA署名を生成する際に m をハッシングするために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' のx座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示す、方法が提供され得る。

【0276】

30

この場合、コードは、前記のチェックが真であることを条件に真の結果を返し、その他の場合に偽の結果を返すよう構成される。

【0277】

第4の任意的な具体化によると、第1又は第2の具体化による方法であって、前記第インスタンストランザクション内で受信された前記情報は、前記第1 ECDSA署名の前記r部分の提出されたインスタンスを更に含み、前記コードは、前記r部分の前記提出されたインスタンスが前記参照インスタンスに等しいことをチェックするよう構成される、方法が提供され得る。

【0278】

第5の任意的な具体化によると、第4の具体化による方法であって、前記ECDSA検証関数は、前記第2トランザクション内で受信された前記s部分が前記r部分の前記提出されたインスタンスに対応することをチェックし、両方のチェックを一緒に行うことにより、前記参照インスタンスに対応することを検証する、方法が提供され得る。

40

【0279】

本開示の第6の任意的な具体化によると、第2の具体化に依存する第5の具体化による方法であって、前記コードは、以下を実行し：

【数32】

50

$r' = r$ であることをチェックし、

ここで、ECDSA 検証関数は以下を実行する：

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

$[R']_x = r$ であることをチェックする

ここで、 r は前記第1 ECDSA署名の前記 r 部分の前記提出されたインスタンスであり、 r' は前記第1 ECDSA署名の前記 r 部分の参照インスタンスであり、 s は前記第1 ECDSA署名の前記 s 部分であり、 P は前記第1公開鍵であり、 m は前記第1 ECC署名により署名された前記第2トランザクションの一部であり、 H_{sig} は前記第1 ECDSA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 は楕円生成元であり、「 R' 」 $_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示す、方法が提供され得る。

【0280】

この場合、コードは、両方の前記のチェックが真であることを条件に真の結果を返し、その他の場合に偽の結果を返すよう構成される。

【0281】

第7の任意的な具体化によると、第4又は第5の具体化による方法であって、前記の要素が第1 ECDSA署名の r 部分の参照インスタンスである方法が提供され得る。この場合、コードは、前記提出されたインスタンスに対して同じ変換を実行することにより、前記提出されたインスタンスが前記参照インスタンスと等しいことの前記チェックを実行するよう構成される。

【0282】

第8の任意的な具体化によると、第7の具体化による方法であって、前記要素は、前記第1 ECDSA署名の前記 r 部分の前記参照インスタンスを含むコンポーネントのハッシュであるハッシュ値である（前記変換はハッシュである）、方法が提供され得る。

【0283】

本開示の第9の任意的な具体化によると、第5の具体化に依存する第8の具体化による方法であって、前記コンポーネントは前記第1 ECDSA署名の前記 r 部分の前記参照インスタンス r' であり、前記コードは、以下を実行するよう構成され、

【数33】

$h = H_{puz}(r)$ であることをチェックし、

ここで、ECDSA 検証関数は以下を実行する：

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

$[R']_x = r$ であることをチェックする

ここで、 r は前記第1 ECDSA署名の前記 r 部分の前記 r 座標のインスタンスであり、 s は前記第1 ECDSA署名の前記 s 部分であり、 h はハッシュ値であり、 H_{puz} は r をハッシュして h を生成するために使用されたハッシュ関数であり、 P は前記第1公開鍵であり、 m は前記第1 ECC署名により署名された前記第2トランザクションの一部であり、 H_{sig} は前記第1 ECDSA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 は楕円生成元であり、「 R' 」 $_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示す、方法が提供され得る。

【0284】

この場合、コードは、両方の前記のチェックが真であることを条件に真の結果を返し、その他の場合に偽の結果を返すよう構成される。

【0285】

H_{puz} は、ハッシュパズル内で使用されるハッシュ関数である。 H_{sig} は、ECC署名及び検証で使用されるハッシュ関数である。 H_{puz} 及び H_{sig} は、同じ形式のハッシュ関数であってよく又はそうでなくてよい。

【0286】

本開示の第10の任意的な具体化によると、第5の具体化に依存する第8の具体化による方法であって、前記コンポーネントは、前記第1 ECDSA署名の前記 r 部分の前記参照インスタンス r' のデータ値の参照インスタンス d' との結合 $f(r', d')$ であり、前記第2トランザクション内で受信された情報は、記第1 ECDSA署名の前記 r 部分の提出されたインスタンス r 及び前記データ値の提出されたインスタンス d を更に含む、方法が提供され得る。コードは、以下を実行するよう構成される。

10

【数34】

$$h_{joint} = H_{puz}(f(r, d)) \text{ であることをチェックし、}$$

ここで、ECDSA 検証関数は以下を実行する：

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

$$[R']_x = r \text{ であることをチェックする}$$

20

ここで、 s は前記第1 ECDSA署名の前記 s 部分であり、 h_{joint} はハッシュ値であり、 H_{puz} は、 $f(r', d')$ をハッシングして h_{joint} を生成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1 ECC署名により署名された前記第2トランザクションの部分であり、 H_{sig} は、 m をハッシングして前記第1 ECDSA署名を生成するために使用されたハッシュ関数であり、 G は楕円生成元であり、 $[R']_x$ は R' の x 座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示す。この場合、コードは、両方の前記のチェックが真であることを条件に真の結果を返し、その他の場合に偽の結果を返すよう構成される。

【0287】

H_{puz} 及び H_{sig} は、同じ形式のハッシュ関数であってよく又はそうでなくてよい。

【0288】

30

本開示の第11の任意的な具体化によると、第10の具体化による方法であって、前記結合 f は連結であり、以下である方法が提供され得る：

【数35】

$$f(r', d') \text{ は } r' || d' \text{ であり、 } H_{puz}(f(r, d)) \text{ は } H_{puz}(r || d) \text{ である。}$$

【0289】

本開示の第12の任意的な具体化によると、第10又は第11の具体化による方法であって、前記データ値は、目標時点を指定するロックタイム値を含み、前記コードは、前記目標時点に達することを更なる条件として、真の結果を返すよう構成される、方法が提供され得る。

40

【0290】

前記ロックタイム値は、人間の時間の観点で、例えば、ミリ秒、秒、分、時間、日、週、月又は年で、目標時点を指定してよい。代替として、ロックタイム値は、ブロック数の観点で目標時点を指定してよい。いずれの方法も、指定される時点は、絶対時間、又は所定の点、例えば最初のトランザクションがブロックチェーンのブロックへとマイニングされた点からの経過時間量として指定されてよい。

【0291】

本開示の第13の任意的な具体化によると、第1～第12のいずれかの具体化による方法であって、前記第1公開鍵を取得する前記ステップは、前記第2トランザクション内の情報の部分として、前記第1公開鍵を受信するステップを含む、方法が提供され得る。

50

【0292】

代替として、前記第2トランザクション内の情報は、前記第1 ECDSA署名の前記s部分及び前記r部分の提出されたインスタンスの両方を含み、前記取得するステップは、前記第1 ECDSA署名の前記s部分及び前記r部分の提出されたインスタンスから、前記第1公開鍵を導出するステップを含む、方法が提供され得る。

【0293】

別の代替案として、取得する前記ステップが、例えば、前記第2トランザクションに関連するサイドチャネルを介して、前記第1公開鍵を受信するステップ、又は、前記第2トランザクション内で又は前記サイドチャネルを介して、前記第1公開鍵のインデックス又は前記第1公開鍵の所有者のアイデンティティを受信し、これを使用して前記第1公開鍵を検索するステップ、を含み得ることを除外しない。

10

【0294】

第14の任意的な具体化によると、第1～第13のいずれかの任意的な具体化による方法であって、前記コードは、第2 ECDSA署名のr部分の参照インスタンスを更に指定し、前記コードは、複数の代替条件のうちのいずれか1つが満たされた場合に、真の結果を返すよう構成される、方法が提供され得る。この場合、前記代替条件は、

i) 前記第1 ECDSA署名により署名された前記第2トランザクションの一部及び前記第1公開鍵が与えられると、前記第1 ECDSA署名の前記s部分が前記第1 ECDSA署名の前記r部分の参照インスタンスに対応することを検証することと、

ii) 前記第2 ECDSA署名により署名された前記第2トランザクションの一部及び前記第2 ECDSA署名を生成するために使用された第2秘密鍵に対応する第2公開鍵が与えられると、前記第2 ECDSA署名の前記s部分が前記第2 ECDSA署名の前記r部分の参照インスタンスに対応することを検証することと、

20

を含んでよい。

【0295】

例えば、異なる条件は、合意の異なる代替条項のような異なる代替ステートメントにマッピングされてよい。

【0296】

第15の任意的な具体化によると、第1～第14のいずれかの任意的な具体化による方法であって、前記第1 ECDSA署名の少なくとも前記s部分は、第2パーティにより、第1パーティにより前記第2パーティに与えられた若しくは逆も同様である一時鍵、及び前記第2パーティの秘密鍵である第1秘密鍵を用いて生成される、方法が提供され得る。

30

【0297】

第16の任意的な具体化によると、第15の任意的な具体化による方法であって、

【数36】

$$P = V \cdot G, \quad k \in [1, n-1], \quad R = k \cdot G, \quad r = [R]_x, \quad \text{and} \quad s = k^{-1}(H_{sig}(m) + rV) \bmod n$$

ここで、Pは前記第1公開鍵であり、Vは前記第1秘密鍵であり、kは前記一時鍵であり、Gは楕円生成元であり、nは素数モジュラスであり（生成元の素数位数）、mは前記第1 ECDSA署名により署名された前記第2トランザクションの一部であり、 H_{sig} は、前記第1 ECDSA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座標を示し、「 $\cdot$ 」は楕円曲線スカラー乗算を示す、方法が提供され得る。

40

【0298】

第17の任意的な具体化によると、少なくとも第14の具体化に依存する第15又は第16の任意的な具体化による方法であって、前記第1 ECDSA署名の前記r部分の前記提出されたインスタンスrも前記第2パーティにより生成される、方法が提供され得る。

【0299】

第18の任意的な具体化によると、少なくとも第10、第11、又は第12の具体化に依存する第17の任意的な具体化による方法であって、前記データ値の前記提出されたイ

50

ンスタンスdも前記第2パーティにより生成される、方法が提供され得る。

【0300】

本開示の第19の任意的な具体化によると、第15～第18のいずれかの具体化による方法であって、前記第2トランザクションを受信する前記ステップは、前記第2パーティから前記第2トランザクションを受信するステップを含む、方法が提供され得る。

【0301】

本開示の第20の任意的な具体化によると、第15～第19のいずれかの具体化による方法であって、前記コードにより返された結果が真であることを条件として、前記第1パーティのためのサービスをトリガするステップを含む方法が提供され得る。

【0302】

例えば、前記サービスは、前記第1パーティにより委任されてよく、前記第1パーティの代わりに実行されてよく、及び／又は前記第1パーティの利益のために実行されてよい。前記サービスは、コンピュータ化されたサービスであってよく、トリガする前記ステップは、前記サービスを自動的にトリガするステップを含んでよい。

【0303】

前記一時鍵を前記第2パーティ（「Bob」）に与えることにより、これは、前記第1パーティ（「Alice」）がBobに彼女の代わりに前記サービスに署名させることを可能にするが、Bobが一時鍵kをノードに開示する又はブロックチェーン上で発行する必要はない。更に、処理は任意の特定の秘密鍵（又はその対応する公開鍵）に結び付けられないので、これは、Aliceがまた第三者（「Charlie」）に一時鍵のコピーを与え得ること、及びBob及びCharlieのいずれかが前記サービスに署名することに成功し得ることを意味する。これは、r部分がチャレンジの基礎として使用され、r部分が任意の特定のアイデンティティにマッピングされないので、可能である。

【0304】

本開示の第21の任意的な具体化によると、第15～第20のいずれかの具体化による方法であって、前記第2トランザクション内で受信された前記情報は、前記第2パーティの更なる秘密鍵を用いて前記第2トランザクションの部分に署名する、前記第2パーティの更なる暗号署名を含み、前記更なる秘密鍵は更なる公開鍵に対応する、方法が提供され得る。

【0305】

前記更なる署名は、ECC署名又は別のタイプ、例えばRSA署名であり得る。

【0306】

本開示の第22の任意的な具体化によると、第21の具体化による方法であって、前記第1パーティ及び／又は第三者が前記更なる公開鍵に基づき前記第2パーティのアイデンティティを検索できるようにするマッピングが利用可能である、方法が提供され得る。

【0307】

例えば、前記アイデンティティは、第2パーティの個人名、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。前記第三者は、例えば、前述のサービスの提供者であり得る。

【0308】

本開示の第23の任意的な具体化によると、第21又は第22の具体化による方法であって、前記コードは、前記更なる公開鍵を用いて前記更なる暗号署名を検証し、前記更なる暗号鍵が検証されたことを更なる条件として真の値を返すよう構成される、方法が提供され得る。

【0309】

本開示の第24の任意的な具体化によると、第15～第23のいずれかの具体化による方法であって、前記第2トランザクション内で受信された前記情報は、前記第1パーティの秘密鍵を用いて前記第2トランザクションの部分に署名する、前記第1パーティの暗号署名を含む、方法が提供され得る。

【0310】

実施形態では、前記方法は、前記第1パーティの前記秘密鍵に対応する公開鍵を取得するステップを含んでよく、前記コードは、前記第2パーティの前記暗号署名を検証し、前記第1パーティの前記暗号署名を更なる条件として真の結果を返すよう構成される。

【0311】

実施形態では、前記第2パーティ及び／又は第三者が前記第1パーティの前記公開鍵に基づき前記第1パーティのアイデンティティを検索できるようにするマッピングが利用可能であってよい。例えば、前記第1パーティの前記アイデンティティは、前記第1パーティの個人名、会社名、ユーザ名又はネットワークアドレスであってよい。

【0312】

本開示の第25の任意的な具体化によると、第15～第24のいずれかの具体化による方法であって、前記第2トランザクション内で受信された前記情報は、前記第1ECDSA署名と同じ第1秘密鍵を用いるが、前記r部分の異なる値を有する追加ECDSA署名を含み、

10

前記コードは、前記第1公開鍵を用いて前記追加ECDSA署名を検証し、前記追加ECDSA署名が検証されたことを更なる条件として真の結果を返すよう構成される、方法が提供され得る。

【0313】

前記追加ECDSA署名は、前記第2トランザクションの、前記第1ECDSA署名と異なる部分に署名してよい。

【0314】

本開示の第26の任意的な具体化によると、第1～第25のいずれかの具体化による方法であって、前記第2トランザクションは、デジタルアセットの額を指定し、前記額は、前記コードが前記真の結果を返すことを少なくとも条件として、アンロックされる、方法が提供され得る。

20

【0315】

本開示の第27の任意的な具体化によると、第1～第26のいずれかの具体化による方法であって、前記トランザクションの各々は、1つ以上のインプットと1つ以上のアウトプットとを含むデータ構造を有し、各アウトプットはロックスクリプトを含み、各インプットはアンロックスクリプトと別のトランザクションのアウトプットへのポインタを含み、

前記コードは、前記第1トランザクションの前記ロックスクリプトに含まれ、前記第2トランザクション内で受信される前記情報は、前記第2トランザクションのインプット内の前記アンロックスクリプトに含まれ、前記第2トランザクションの前記インプット内の前記ポインタは、前記第1トランザクションの前記アウトプットを指す、方法が提供され得る。この場合、前記方法は、前記コードが前記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、前記検証に応答して、以下：前記検証ノードにより1つ以上のブロックへとマイニングするために前記第2トランザクションをトランザクションのプールに含めるステップ、及び／又は前記第2トランザクションを前記ブロックチェーンネットワークの少なくとも1つの他のノードへ転送するステップ、のうちの少なくとも1つを含む。

30

【0316】

前記第1及び第2トランザクションを含む複数のトランザクションの各々について、前記ネットワークのノードのうちの少なくとも幾つかは、前記トランザクションが有効であることを条件として、各トランザクションを伝播させるよう構成され、前記ネットワークの前記ノードのうちの少なくとも幾つかは、前記トランザクションが有効であることを条件として、前記ブロックチェーンの少なくとも一部のコピーに各トランザクションを記録するよう構成される。前記第2トランザクションの有効性は、少なくとも前記コードが前記真の結果を返すことを条件とする。

40

【0317】

本開示の第28の任意的な具体化によると、第27の具体化による方法であって、前記第2トランザクションの前記アウトプットのうちの1つは、更新された第1ECDSA署名の前記r部分の参照インスタンスを指定する前記コードの更新バージョンを含む、方法が提供

50

され得る。この場合、前記方法は、

前記更新された第1 ECDSA署名の少なくともs部分を含む情報を含むトランザクションのセットのうちの第3トランザクションを受信し、更新された第1公開鍵を取得するステップであって、前記更新された第1 ECDSA署名は、前記更新された公開鍵に対応する更新された第1秘密鍵に基づき、前記第3トランザクションの部分に署名する、ステップと、

前記第3トランザクションから前記コードの更新されたバージョンを実行するステップであって、前記更新されたコードは、前記第2トランザクションの署名済み部分及び前記取得された第1公開鍵が与えられると、どの秘密鍵が前記更新された第1秘密鍵として使用されたかに関係なく、前記更新された第1 ECDSA署名に適用された前記更新された第1 ECDSA署名が、前記更新されたs部分が前記第2トランザクション内で指定された前記r部分の参照インスタンスに対応することを検証することを条件として、真の値を返すよう構成される、ステップと、

を含んでよい。

【0318】

実施形態では、前記第3トランザクションの前記アウトプットのうちの1つは、更なる更新された第1 ECDSA署名の前記r部分の参照インスタンス等を指定し、従って順方向チェーン又はrパズルを生成する、前記コードの更なる更新バージョンを含んでよい。

【0319】

本開示の第29の任意的な具体化によると、第1～第26の具体化による方法であって、前記トランザクションは、アカウントに基づくモデルに従い構成され、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含まれる、方法が提供され得る。

【0320】

実施形態では、いずれのモデルでも、検証ノードは、ブロックチェーンの少なくとも一部を格納しているマイニングノード、転送ノード、及び／又は記憶ノードであってよい（例えば、フルコピー記憶ノードはブロックチェーンの完全なコピーを格納している）。実施形態では、前記第1トランザクションを取得する前記ステップは、第1パーティ、例えば前述の第1パーティから、前記第1トランザクションの少なくとも部分を受信するステップを含んでよい。実施形態では、前記第1バージョンを取得する前記ステップは、前記第1パーティから前記第1トランザクションを受信するステップを含んでよい。代替として、実施形態では、前記第1トランザクションを取得する前記ステップは、前記検証ノードにおいて前記第1トランザクションを策定するステップを含んでよい。実施形態では、前記第1トランザクションを受信する前記ステップは、少なくとも前記第1パーティから前記r部分の前記参照インスタンスを受信するステップと、前記ノードのうちの前記1つにおいて前記第1トランザクションへと策定するステップと、を含んでよい。実施形態では、前記第1トランザクションを取得する前記ステップは、前記検証ノードにおいて、前記r部分を生成することを含む前記第1トランザクションを策定するステップを含んでよい。

【0321】

実施形態では、前記第2トランザクションを受信する前記ステップは、第2パーティ、例えば前述の第2パーティから、前記第2トランザクションを受信するステップを含んでよい。実施形態では、前記第2トランザクションは、前記第2パーティにより少なくとも部分的に生成される。実施形態において、前記第2トランザクションは、前記第2パーティにより生成される。実施形態では、前記第2トランザクションを受信する前記ステップは、前記第2パーティから直接に、又は前記第1パーティ若しくは第三者を介して、前記第2トランザクションを受信するステップを含んでよい。実施形態では、前記第2トランザクションは、前記第2パーティにより前記第三者に提供される前記第1 ECDSA署名の少なくとも前記s部分（及び、実施形態では、前記第1 ECDSA署名の前記r部分の前記提出されたインスタンス及び／又は前記データ値）に基づき、第三者により生成される。

【0322】

本願明細書の教示の第13の具体化によると、コンピュータ可読記憶装置上に具現化さ

10

20

30

40

50

れ、ネットワークのノード上で実行すると第1～第29のいずれかの具体化の方法を実行するよう構成される、コンピュータプログラムが提供される。

【0323】

第31の具体化によると、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機器と、
を含み、

前記メモリは、前記処理機器上で実行するよう構成されるコードを格納し、前記コードは、前記処理機器に第1～第29の具体化のうちのいずれかの方法を実行させるよう構成される、ノードが提供される。

10

【0324】

本願明細書の教示の第32の具体化によると、楕円曲線デジタル署名アルゴリズム（ECDSA）検証関数に基づき知識証明を設定する、コンピュータにより実施される方法であって、前記方法は、第1パーティのコンピュータ機器を用いて、

第2パーティへ一時鍵を送信するか、又は前記第2パーティから前記一時鍵を受信するステップと、

前記一時鍵に基づき、第1ECDSA署名のr部分の参照インスタンスを決定するステップと、

実行可能コードを含む第1トランザクションを形成するステップであって、前記コードは前記第1ECDSA署名の前記r部分の参照インスタンスを指定する要素を含む、ステップと、

20

を含み、

前記コードは、第2トランザクションからの前記第1ECDSA署名の少なくともs部分、及び第1公開鍵、に作用するよう構成され、前記第1ECDSA署名は、前記第1公開鍵に対応する第1秘密鍵に基づき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り、

前記コードは、どの秘密鍵が前記第1秘密鍵として使用されたかに関係なく、以下：

前記第2トランザクション内で受信されたメッセージ及び前記第1公開鍵が与えられると、前記ECDSA検証関数が、前記第1ECDSA署名に適用されるとき、前記第2トランザクション内で受信された前記s部分が前記第1トランザクションにより指定された前記r部分の参照インスタンスに対応することを検証する、

30

ことを条件として、真の結果を返すよう構成される、方法が提供される。

【0325】

実施形態では、方法は、本願明細書に記載された任意の具体化又は他の実施形態に従い前記第1パーティのコンピュータ機器において実行されるステップを更に含んでよい。

【0326】

本願明細書の教示の第33の具体化によると、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、1つ以上のコンピュータ可読データ媒体上に具現化され、

第1トランザクションは実行可能コードを含み、該コードは第1ECDSA署名のr部分の参照インスタンスを指定する要素を含み、

40

第2トランザクションは第1ECDSA署名の少なくともs部分を含む情報を含み、

第1ECDSA署名は第1公開鍵に対応する第1秘密鍵に基づきメッセージに署名し、メッセージは第2トランザクションの部分である。前記コードは、ブロックチェーンネットワークのノードにおいて実行すると、第1秘密鍵として誰の公開鍵が使用されたかに関係なく、ECDSA検証関数が、第1ECDSA署名に適用されるとき、第2トランザクション内で受信されたメッセージ及び取得された第1公開鍵が与えられると、第2トランザクション内で受信されたs部分が第1トランザクションにより指定されたr部分の参照インスタンスに対応することを検証することを条件として、真の結果を返すよう構成される。

【0327】

50

実施形態では、第 1 及び／又は第 2 トランザクションは、本願明細書に開示された任意の具体化又は他の実施形態による特徴を更に含んでよい。

【 0 3 2 8 】

開示された技術の他の変形例又は使用事例は、本明細書で開示されると、当業者に明らかになり得る。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、添付の特許請求の範囲によってのみ限定される。

10

20

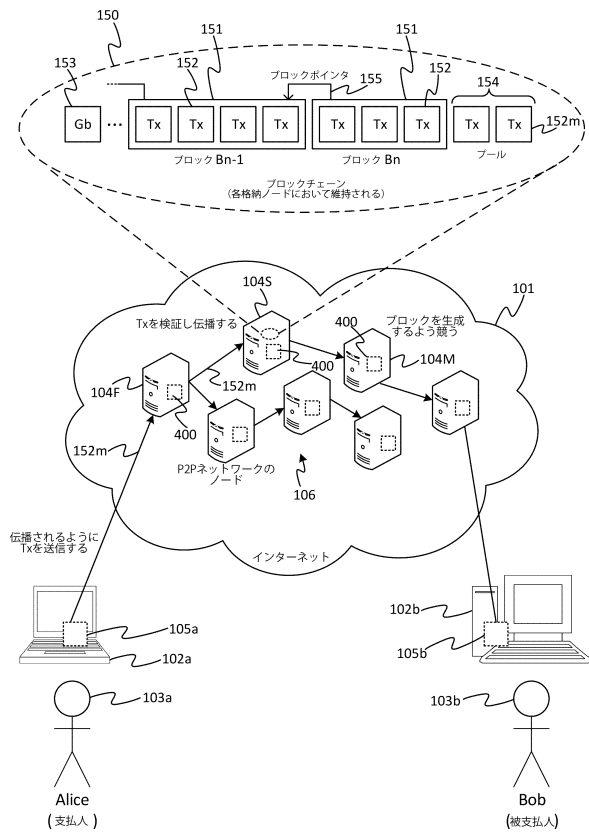
30

40

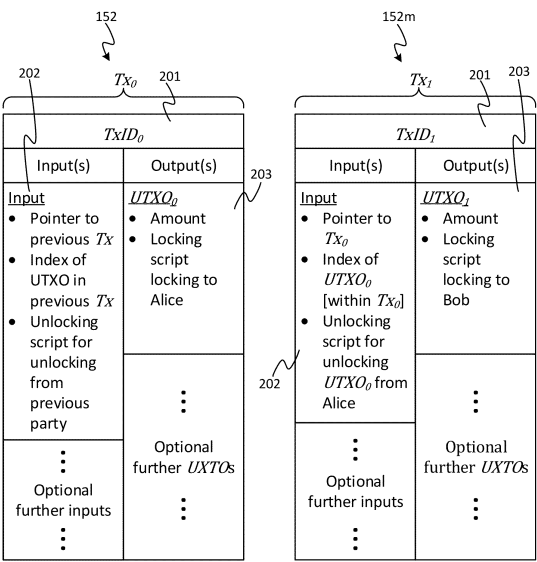
50

【図面】

【図 1】



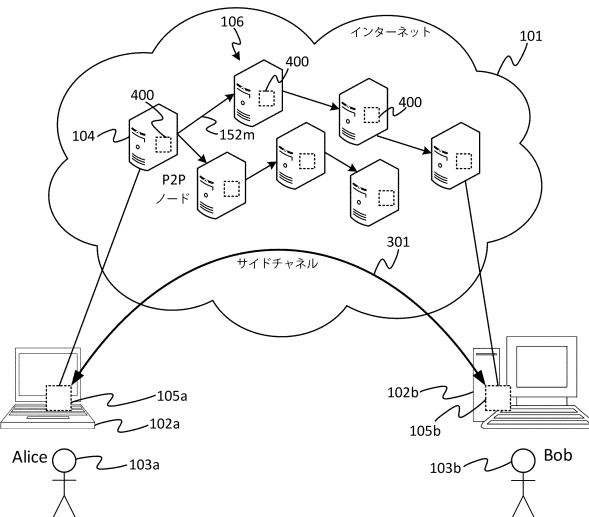
【図 2】



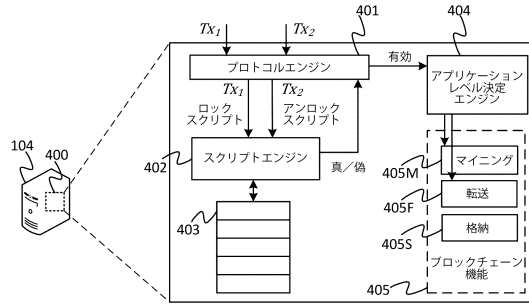
AliceからBobへの
トランザクション

(Tx0のアウトプットからの) Aliceのロックスクリプトを
(Tx1へのインプットとして) Aliceの
アンロックスクリプトと一緒に
実行することにより検証される。
これは、Tx1がAliceのロックスクリプト内に
定義された条件を満たすことをチェックする。

【図 3】



【図 4】



10

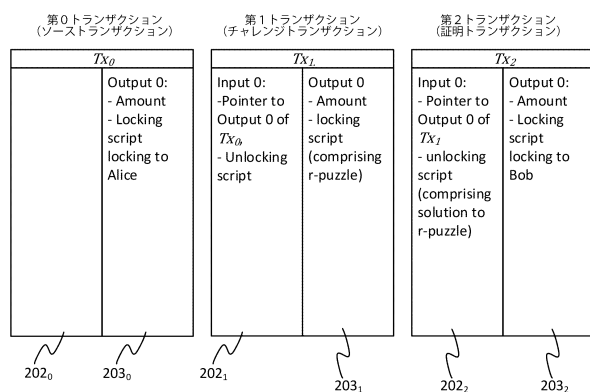
20

30

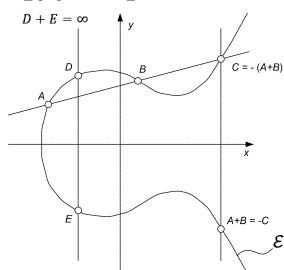
40

50

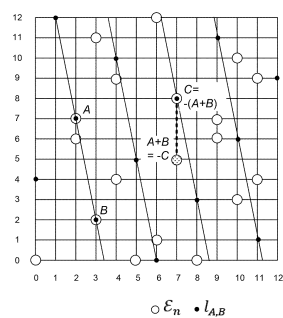
【図 5】



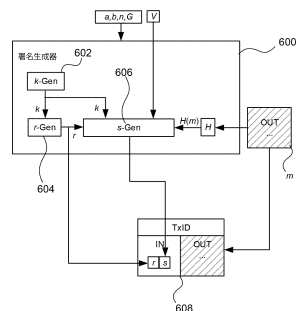
【図 6 A】



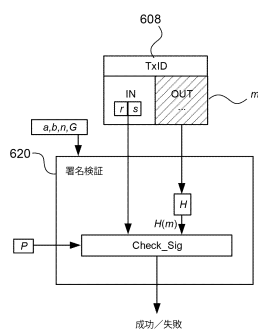
【図 6 B】



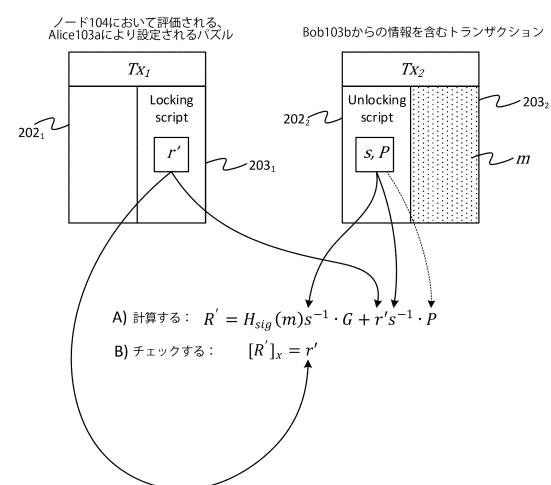
【図 6 C】



【図 6 D】



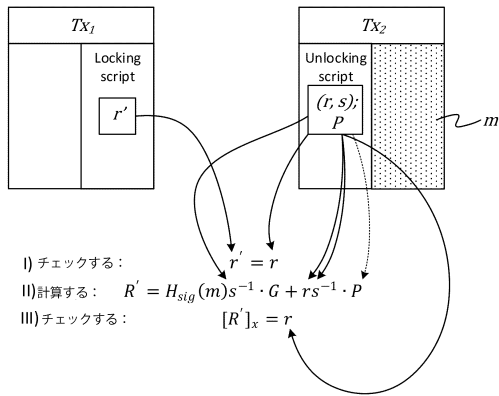
【図 7】



【図 8】

ノード104において評価される。
Alice103aにより設定されるハズル

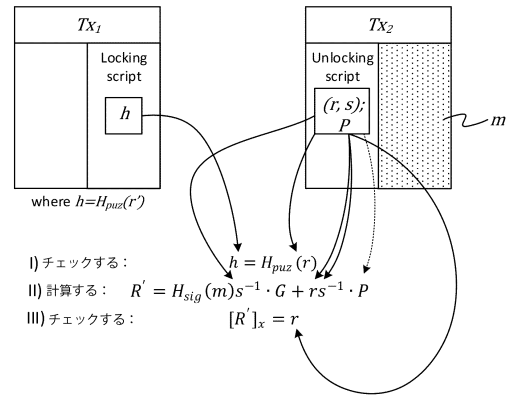
Bob103bからの情報を含むトランザクション



【図 9】

ノード104において評価される。
Alice103aにより設定されるハズル

Bob103bからの情報を含むトランザクション

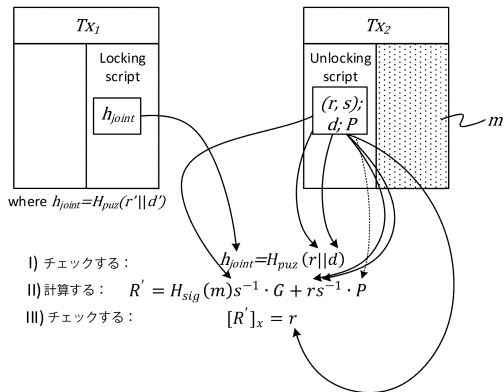


10

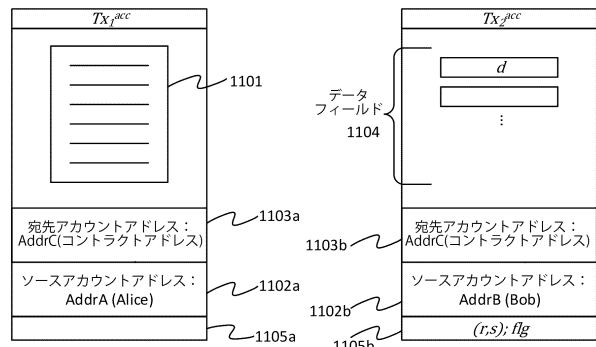
【図 10】

ノード104において評価される。
Alice103aにより設定されるハズル

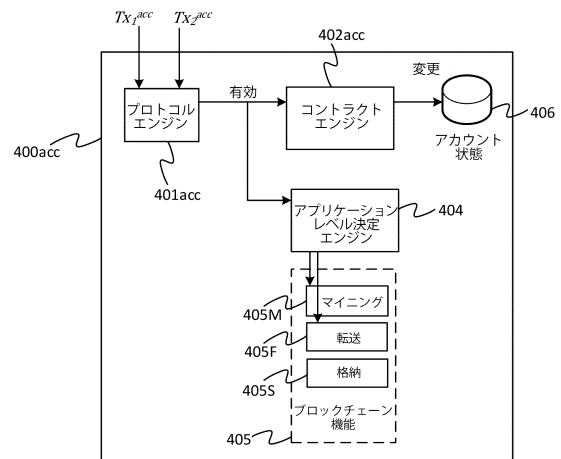
Bob103bからの情報を含むトランザクション



【図 11】



20

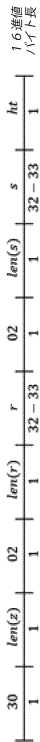


30

40

50

【図 1 2】



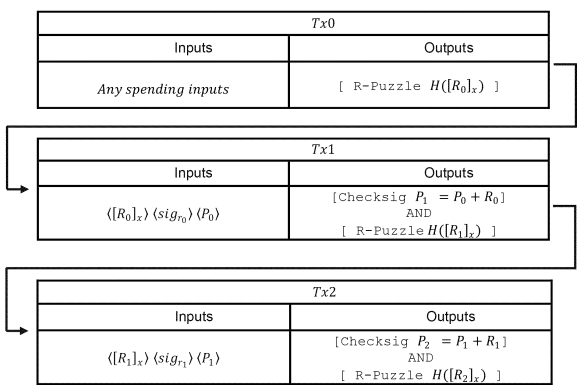
【図 1 3】

Stack	Script	Description
empty	$(P_2) \langle sig_r \rangle$ OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	scriptSig及びScriptPubKeyが結合される
$(P_2) \langle sig_r \rangle$	OP_DROP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	アンロックスクリプトの定数がスタックにプッシュされる
$(P_2) \langle sig_r \rangle \langle sig_r \rangle$	OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS(sig)が複製され、定数(3)がスタックにプッシュされる
$(P_2) \langle sig_r \rangle \langle sig_r \rangle [3:]$	OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS(sig)が、len()の直前の第3バイトで分割される
$(P_2) \langle sig_r \rangle \langle sig_r \rangle [3:]$ OP_1	OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToSアイテムの2番目が除去され、定数(1)がスタックにプッシュされる
$(P_2) \langle sig_r \rangle \langle sig_r \rangle [4:] \langle sig_r \rangle [3:]$	OP_SPLIT OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToSが1バイトで分割し、(sig)の2つのバイトをスタックする
$(P_2) \langle sig_r \rangle \langle sig_r \rangle [4:36/37] \langle sig_r \rangle [36/37:]$	OP_DROP OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToSアイテムの2番目 (インデックス4で開始するsig) が len()=(sig[3:4])バイトで分割され、(sig[3:4])バイトの2つのバイトをスタックする
$(P_2) \langle sig_r \rangle \langle sig_r \rangle [4:36/37] \langle sig_r \rangle [36/37:]$	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS (sig)の第3バイトをスタックにプッシュする
$(sig_r) (P_2)$	OP_CHECKSIG	$(sig_r [4:36/37]) = (r)$ かどうかをチェックする
True	empty	上位2個のスタックアイテムについて、署名がチェックされる

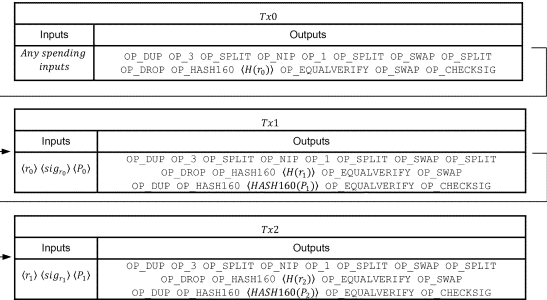
10

20

【図 1 4 A】



【図 1 4 B】



30

40

50

フロントページの続き

- (72)発明者 ジャン, ウェイ
 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 ドイロン, 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 ヴォーガン, オーウェン
 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 金沢 史明
- (56)参考文献 国際公開第2018/215876 (WO, A1)
 国際公開第2019/034984 (WO, A1)
- (58)調査した分野 (Int.Cl., DB名)
- G09C 1/00
 H04L 9/00- 9/40