(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7516425号
(P7516425)

(45)発行日　令和6年7月16日(2024.7.16)

(24)登録日　令和6年7月5日(2024.7.5)

(51)国際特許分類
H04L 9/32 (2006.01)

FI
H04L 9/32 200C
H04L 9/32 200Z

請求項の数 27 （全69頁）

(21)出願番号 特願2021-569311(P2021-569311)
(86)(22)出願日 令和2年4月21日(2020.4.21)
(65)公表番号 特表2022-533752(P2022-533752A)
(43)公表日 令和4年7月25日(2022.7.25)
(86)国際出願番号 PCT/IB2020/053762
(87)国際公開番号 WO2020/240289
(87)国際公開日 令和2年12月3日(2020.12.3)
審査請求日 令和5年3月22日(2023.3.22)
(31)優先権主張番号 1907397.2
(32)優先日 令和1年5月24日(2019.5.24)
(33)優先権主張国・地域又は機関
英国(GB)

(73)特許権者 318001991
エヌチェーン ライセンシング アーゲー
スイス・６３００・ツーク・グラーフェナウヴェーク・６
(74)代理人 100107766
弁理士　伊東 忠重
(74)代理人 100070150
弁理士　伊東 忠彦
(74)代理人 100135079
弁理士　宮崎 修
(72)発明者 ワハブ，ジェド
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

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）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、共同r値r_{joint}に基づき定義されるチャレンジを評価するための参照データを含む、ステップと、

第１ECDSA署名i=１,２のペアのうちのそれぞれのECDSA署名のそれぞれのr部分r_i及びs部分s_iを少なくとも含む情報を含む１つ以上の第２トランザクションを受信するステップであって、前記第１ECDSA署名のうちのそれぞれは、それぞれの第１公開鍵P_iに対応するそれぞれの第１秘密鍵V_iに基づき前記１つ以上の第２トランザクションのうちの１つの第２トランザクションの部分に署名する、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、前記第１トランザクション内の前記参照データ及び前記１つ以上の第２トランザクションの中で受信された前記r部分r_iに基づき、前記チャレンジが満たされるかどうかを検証し、それを条件として真の値を返すよう構成され、前記チャレンジは、以下の基準：

【数６２】

$$r_1 + r_2 = \lambda^2 - r_{joint} \, mod \, p,$$

を含み、ここで、r_1+r_2はスカラー加算を示し、

【数６３】

$r_{joint} = [R_{joint}]_x$ ，楕円曲線点加算により $R_{joint} = R_1 + R_2$、

pは素数モジュラスであり、

【数６４】

$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \, mod \, p, \; R_i = k_i \cdot G, \; x_i = [R_i]_x, \; y_i = [R_i]_y$、

k_iは一時鍵であり、Gは楕円曲線生成元であり、$[]_x$は[…]のx座標を示し、$[]_y$は[…]のy座標を示し、「・」は楕円曲線スカラー乗算を示す、ステップと、

を含む方法。

【請求項２】

λ^2を示すデータが、前記１つ以上の第２トランザクションのうちの少なくとも１つの第２トランザクションの中で受信される情報の部分として受信され、前記の決定は、λ^2を示す受信されたデータに基づき実行される、請求項１に記載の方法。

【請求項３】

前記第１トランザクション内の前記参照データは、p及びλ^2-rの値を含む、請求項１に記載の方法。

【請求項４】

前記方法は、

前記第１ECDSA署名のうちの各々の第１ECDSA署名のそれぞれの第１公開鍵を取得し、ECDSAの検証関数を適用して、前記それぞれの第１公開鍵及び署名済み部分に基づき、各第１ECDSA署名を検証するステップであって、前記コードは、前記第１ECDSA署名のうちの各第１ECDSA署名の前記検証を更なる条件として、真の値を返すよう構成される、ステップを含む請求項１、２、又は３に記載の方法。

【請求項５】

前記第１公開鍵を取得する前記ステップは、前記１つ以上の第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４に記載の方法。

【請求項６】

前記コードは、前記第１ECDSA署名のうちの１つ、一部、又は全部のそれぞれの第１公開鍵として、誰の公開鍵が使用されたかに関係なく、真の結果を出力するよう構成される、請求項１～５のいずれかに記載の方法。

【請求項７】

ECDSAは、$y^2=x^3+7$の形式の楕円曲線に基づく、請求項１～６のいずれかに記載の方法。

【請求項８】

ECDSAはsecp２５６k１アルゴリズムである、請求項１～７のいずれかに記載の方法。

【請求項９】

前記第１ECDSA署名の各々は、異なるそれぞれの第２パーティの署名であり、前記チャレンジは少なくとも部分的に第１パーティにより定義される、請求項１～８のいずれかに記載の方法。

【請求項１０】

前記第１ECDSA署名の各々の前記r部分及びs部分は、それぞれの一時鍵k_iを用いてそれぞれの第２パーティにより生成される、請求項９に記載の方法。

【請求項１１】

それぞれの一時鍵は、前記第１パーティによりそれぞれの第２パーティに与えられる、又はその逆である、請求項１０に記載の方法。
【請求項１２】
【数６５】

$$P_i = V_i \cdot G,$$

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

$$R_i = k_i \cdot G,$$

$$r_i = [R_i]_x, \text{ and}$$

$$s_i = k_i^{-1}\left(H_{sig}(m) + r_i V_i\right) mod\ n,$$

ここで、k_iはそれぞれの一時鍵であり、m_iは前記第２トランザクションのそれぞれの署名済み部分であり、H_{sig}はそれぞれの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、nは前記生成元の素数位数である、請求項１１に記載の方法。
【請求項１３】
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、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７～１２のいずれかに記載の方法。
【請求項１４】
前記１つ以上の第２トランザクションを受信する前記ステップは、第２パーティのうちの１つから前記第２トランザクションのそれぞれを受信するステップを含む、請求項７～１３のいずれかに記載の方法。
【請求項１５】
前記第２トランザクション内で受信された前記情報は、第２パーティの更なる秘密鍵を用いる前記第２パーティのうちの少なくとも１つ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７～１４のいずれかに記載の方法。
【請求項１６】
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少なくとも１つの第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１５に記載の方法。
【請求項１７】
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１５又は１６に記載の方法。
【請求項１８】
前記第２トランザクション内で受信された前記情報は、第１パーティの秘密鍵を用いる前記第１パーティの暗号署名を更に含む、請求項７～１７のいずれかに記載の方法。
【請求項１９】
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署名と同じそれぞれの秘密鍵を用いるが、前記第１ECDSA署名と異なるr部分の値を有する、第２パーティのうちの１つ、一部、又は全部の各々の追加ECDSA署名を含み、
前記コードは、それぞれの前記第１公開鍵を用いて前記追加ECDSA署名の各々を検証し、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７～１８のいずれかに記載の方法。
【請求項２０】
前記コードは、r部分の値の大きな集合の中から任意の２つのr部分の値がr_1及びr_2として使用可能にするよう構成される、請求項１～１９のいずれかに記載の方法。

【請求項２１】
前記集合は、r部分の値の少なくとも３個のペアを含み、
k値の異なるそれぞれのペアは、少なくとも３つの第２パーティのそれぞれに分配されて、それらが集合内のペアのそれぞれを生成できるようにし、
r_1は前記第２パーティのうちの１つの第２パーティからのr部分の値のうちの１つであり、r_2は前記第２パーティのうちの別の第２パーティからのr部分の値であり、
前記コードは、前記少なくとも３つの第２パーティのうちの任意の２つが前記チャレンジを満たすことを可能にする、請求項７～１９のいずれかに従属する請求項２０に記載の方法。
【請求項２２】
前記情報は、同じ第２トランザクション内で受信される、請求項１～２１のいずれかに記載の方法。
【請求項２３】
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と含み、
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２２に記載の方法。
【請求項２４】
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、請求項１～２２のいずれかに記載の方法。
【請求項２５】
前記情報は、複数の第２トランザクションに渡り受信される、請求項２４に記載の方法。
【請求項２６】
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１～２５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２７】
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理機器に請求項１～２５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。
【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き、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の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ことに伴う問題は、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「・」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、Tx_2の部分に署名し、Tx_1のアウトプットをアンロックするために、Tx_2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】
$$r = [R]_x, where\ R = k \cdot G$$
$$s = k^{-1}\left(H_{sig}(m) + rV\right) mod\ n$$

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されるハッシュ関数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】

本開示は、「rパズル」と呼ばれる本願明細書で開示される新しい形式の知識証明に関する。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、Tx_1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、Tx_2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。

【００１９】

特に、本開示は、複数のr部分を結合するrパズルの形式を請求する。例えば、各r部分は、異なるパーティの一時鍵kに対応してよく、当該方法は、一緒に動作する複数の異なるパーティによってのみ償還可能な第１トランザクションを生成するために使用されてよい。

【００２０】

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）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。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、共同r値r_{joint}に基づき定義されるチャレンジを評価するための参照データを含む、ステップと、

第１ECDSA署名i=１,２のペアのうちのそれぞれのECDSA署名のそれぞれのr部分r_i及びs部分s_iを少なくとも含む情報を含む１つ以上の第２トランザクションを受信するステップであって、前記第１ECDSA署名のうちのそれぞれは、それぞれの第１公開鍵P_iに対応するそれぞれの第１秘密鍵V_iに基づき前記１つ以上の第２トランザクションのうちの１つの第２トランザクションの部分に署名する、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、前記第１トランザクション内の前記参照データ及び前記１つ以上の第２トランザクションの中で受信された前記r部分r_iに基づき、前記チャレンジが満たされるかどうかを検証し、それを条件として真の値を返すよう構成され、前記チャレンジは、以下の基準：

【数３】

$$r_1 + r_2 = \lambda^2 - r_{joint} \bmod p$$

を含み、ここで、r_1+r_2はスカラー加算を示し、

【数４】

$r_{joint} = [R_{joint}]_x$、楕円点加算により、$R_{joint} = R_1 + R_2$

pは素数モジュラスであり、

【数５】

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p,\ R_i = k_i \cdot G,\ x_i = [R_i]_x,\ y_i = [R_i]_y,$$

k_iは一時鍵であり、Gは楕円曲線生成元であり、$[]_x$は[…]のx座標を示し、$[]_y$は[…]のy

座標を示し、「・」は楕円曲線スカラー乗算を示す、ステップと、
を含む。
【００２１】
知識証明に基づく開示されるrパズルは、単純に、Ｐ２ＰＫＨの知識証明に基づく従来のハッシュパズルの代替として使用され得る。しかしながら、実施形態において利用され得るアプローチに基づくrパズルの追加の利点がある。つまり、Ｐ２ＰＫＨにおけるチャレンジと異なり、第１トランザクション（例えば、Tx_1）内のチャレンジの基礎を形成するr部分は、任意の特定のアイデンティティにリンクされない。同時に、rパズルを解くことが検証者にkを開示することなくkの知識を証明するという事実は、rパズルが脆弱性の影響を受けないことを意味する。この脆弱性によると、マイナー又はノードのオペレータは、解を観察でき、それを第２トランザクションの彼／彼女自身のバージョンに挿入できる。
【００２２】
従って、実施形態では、前記複数のＥＣＤＳＡ署名のうちの１つ、一部、又は全部について、前記コードは、誰の公開鍵がそれぞれの公開鍵として使用されるかに関係なく、真の結果を出力するよう構成される。
【００２３】
上述の脆弱性を回避するために、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、Tx_1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）。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のまま、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。
【００２４】
例えば、Aliceは、彼女がシークレット鍵を与える任意の人々のグループが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、彼女はそれらの人々が誰であるか予め指定したくないとする。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。例えば、トランザクションのペアは、グループによる、契約条項への同意を示すため及び／又は合意により支払いを行うために使用できる。Aliceは、１つ以上の信頼できる人々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。
【図面の簡単な説明】
【００２５】
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システムの別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１０】共同加算rパズルの概略図である。
【図１１】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１２】ECDSA署名の例示的なフォーマットを概略的に示す。
【図１３】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。
【発明を実施するための形態】
【００２６】
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、プリイメージ）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、ハッシュ関数が多くの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。
【００２７】
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。
【００２８】
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。
【００２９】
＜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。

【００３０】

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。

【００３１】

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。

【００３２】

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。

【００３３】

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。

【００３４】

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。

【００３５】

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。

【００３６】

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を一緒に形成する。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。

【００３７】

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。

【００３８】

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試みることにより、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。

【００３９】

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。

【００４０】

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。

【００４１】

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。

【００４２】

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。

【００４３】

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。

【００４４】

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。純粋に例示として、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という言及は、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。

【００４５】

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。

【００４６】

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。

【００４７】

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。

【００４８】

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。

【００４９】

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設備であるため、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いに対応し、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)。

【００５０】

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。

【００５１】

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。

【００５２】
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。

【００５３】
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。

【００５４】
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ず、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）。

【００５５】
＜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。

【００５６】
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。

【００５７】

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。

【００５８】

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タ(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。

【００５９】

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$UTXO_0$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。

【００６０】

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。

【００６１】

従って、図示の例では、Ｔｘ０のアウトプット２０３のUTXO$_0$は、ロックスクリプト[ChecksigP$_A$]を含み、これは、UTXO$_0$が償還されるために(厳密には、UTXO$_0$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SigP$_A$を必要とする。[ChecksigP$_A$]は、Aliceの公開－秘密鍵ペアからの公開鍵P$_A$を含む。Ｔｘ$_1$のインプット２０２は、Ｔｘ$_1$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Ｔｘ$_0$全体のハッシュであるTxID$_0$)を含む。Ｔｘ$_1$のインプット２０２は、Ｔｘ$_0$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、Ｔｘ$_0$内のUTXO$_0$を識別するインデックスを含む。Ｔｘ$_1$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〈SigP$_A$〉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。

【００６２】

新しいトランザクションＴｘ$_1$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６】

$$<\text{Sig } P_A> <P_A> \,||\, [\text{Checksig } P_A]$$

【００６３】

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、Ｔｘ$_0$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PAを使用して、Ｔｘ$_1$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Ｔｘ$_0$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、Ｔｘ$_0$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)。

【００６４】

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ければなら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。

【００６５】

Ｔｘ$_1$内のアンロックスクリプトが、Ｔｘ$_0$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Ｔｘ$_1$内で提供され、認証されている場合)、ノード１０４は、Ｔｘ$_1$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Tx_1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$UTXO_0$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。

【００６６】

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。

【００６７】

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$UTXO_0$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$UTXO_0$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。

【００６８】

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、$UTXO_0$へのポインタが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$UTXO_1$しか持っていないとする。$UTXO_0$で指定されたデジタルアセットの量が$UTXO_1$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できることは除外されない。

【００６９】

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に最も近いか、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。

【００７０】

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 P_A]を[ChecksigP_A]=OP_DUPOP_HASH１６０〈H(P_A)〉OP_EQUALVERIFYOP_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。

【００７１】

署名P_Aは、デジタル署名である。実施形態において、これは楕円曲線secp２５６k１を使用するECDSAに基づく。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)。

【００７２】

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。

【００７３】

＜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い。

【００７４】

サイドチャネル３０１は、Ｐ２Ｐオーバレイネットワーク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。

【００７５】

＜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。

【００７６】

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とを含む。

【００７７】

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。

【００７８】

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。

【００７９】

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）。

【００８０】

＜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）。

【００８１】

第０トランザクションTx_0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Tx_1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Tx_2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Tx_0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Tx_2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Tx_1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。

【００８２】

図５に示すように、ソーストランザクションTx_0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２０３_0（例えば、Tx_0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Tx_0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、Tx_0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Tx_0のインプットをポイントするかは（それらが、Tx_0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。

【００８３】

本発明の場合には、ソーストランザクションTx_0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Tx_1（チャレンジトランザクション）である。従って、Tx_1は、Tx_0の関連するアウトプット（図示の例ではTx_0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Tx_0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２０２_1（例えば、Tx_1のインプット０）を有する。Tx_0のロックスクリプトにより要求されるAliceからの署名は、Tx_1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Tx_1の部分は、Tx_1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Tx_1.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Tx_1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、Tx_1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２０３_1を含む（以下を参照、本例ではTx_1のアウトプット０）。

【００８４】

第１トランザクションTx_1は、少なくとも１つのアウトプット２０３_1（例えば、ここでもアウトプットがUTXOと呼ばれてよいTx_1のアウトプット０）を有する。第１トランザクションTx_1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。Tx_0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、Tx_1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。

【００８５】

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202_2（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。これの例は、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、Tx１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。

【００８６】

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202_2（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。

【００８７】

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203_2、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、Tx２のアウトプット０）、を含む。

【００８８】

実施形態では、Tx_1内のアウトプット203_1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202_2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。

【００８９】

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。

【００９０】

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もよい。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ョンをネットワーク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Tx_1に構成してネットワーク１０６へ転送できる。Tx_1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。

【００９１】

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）Tx_2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Tx_2に構成してネットワークへ転送できる。

【００９２】

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。

【００９３】

「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。

【００９４】

＜楕円曲線デジタル署名アルゴリズム（ELIPTICAL 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。

【００９５】

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。

【００９６】

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）。
【００９７】
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、逆方向に実行され、秘密鍵を用いてのみ実行可能である。
【００９８】
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。
【００９９】
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。
【０１００】
「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。
【０１０１】
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。
【０１０２】
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。
【０１０３】
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７】
$\mathbb{R}^2,\ \mathbb{R}^2$

及び、
【数８】
$\mathbb{R}^2$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９】
$$\mathcal{E}\colon\ y^2 = x^3 + ax + b$$

【０１０４】
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。

【０１０５】

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)

【０１０６】

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)=∞　　∀A∈E

【０１０７】

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ε上の任意の点Aについて、A+∞=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）。

【０１０８】

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１０】

$$v \cdot A = \underbrace{A + \cdots + A}_{v\text{回}}.$$

【０１０９】

つまり、Aのそれ自体とのv回の楕円曲線点加算である。

【０１１０】

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。

【０１１１】

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：

$vv^{-1}=1$

【０１１２】

図６Aは、上述の演算の直感的視覚化を提供する。ここで、εは、全部の実数：

【数１１】

$$\mathbb{R}.$$

を含む無限体に渡り定義される。

【０１１３】

図６Bは、次式により定義される楕円曲線ε_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１２】

$$\mathcal{E}_n\colon\ y^2 = x^3 + ax + b \mod p$$

【０１１４】

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。

【０１１５】

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。

【０１１６】

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。

【０１１７】

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),…０,…,(p-１)の中にある。

【０１１８】

正式には、ε_nは、両端を含む０とp-1との間の整数（[０,p-1]と示される）及びアイデンティティ要素∞の集合[０,p-1]∪{∞}を有する有限体：

【数１３】

$$\mathbb{F}_p$$

に渡り定義される楕円曲線と言える。基本的に、上述の算術演算のセットである（幾つかの変更を伴う、以下を参照）。

【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１４】

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

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

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

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

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

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

ここで、

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

及び

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

【０１２２】
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１５】

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

【０１２３】
つまり、整数vの乗算反数は、v mod pのモジュラ反数である。
【０１２４】
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)=∞である。B=∞の場合も特別であり、上述のようにA+∞=Aと表記して解決される。
【０１２５】
楕円曲線スカラー乗算の定義は、楕円曲線加算のこの定義を採用し、その他の場合には同じままである。
【０１２６】
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１６】

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

【０１２７】
乗算反数がmod n又はmod pに関して定義されるかは、文脈上明らかである。
【０１２８】
実際に、数がmod n又は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、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、mod n
両方のチェックが肯定的結果を与えたる場合があることに留意する。この場合、その数はmod p且つmod nでなければならない。
【０１２９】
＜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）。
【０１３０】
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ε_n上の点である「生成元（generator point）」Gから導出される、楕円曲線ε_n上の点Pであり、次式の通りである：
【数１７】

$$P = V \cdot G = \underbrace{G + \cdots + G}_{V\ 回}$$

【０１３１】
ここで、「・」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ε_n上の楕円曲線スカラー乗算を示す。
【０１３２】

楕円曲線ε_nの次数nは、次式を満たすスカラーとして、生成元Gに関して定義される。

$n \cdot G = \infty$

従って、nは、楕円曲線の選択、及び生成元Gの選択、の両方により決定される。nは簡単に楕円曲線ε_nの次数と呼ばれてよいが、より正確には、nは楕円曲線に関する生成元G（により生成されたサブグループ）の次数である。実際のECCの文脈では、Gは、nが大きな素数であるように選択されるべきである。従って、実施には、p及びnは、両方とも素数であるが、通常は互いに等しくない。

【０１３３】

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。

【０１３４】

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、「力ずくで」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。

【０１３５】

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。

【０１３６】

＜楕円曲線デジタル署名検証アルゴリズム（ECDSA)）＞

ブロックチェーンシステムでは、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。

【０１３７】

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："RFC６９７９-Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", Tools.ietf.org, ２０１９。

【０１３８】

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。

【０１３９】

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。

【０１４０】

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～n-１の範囲で生成する。

【０１４１】

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。

$R = k \cdot G$

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）。

$r=[R]_x$

上式は署名のr部分である。

【０１４２】

s部分生成器６０６は、k mod 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１８】

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

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１９】

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

【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ε_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２０】

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

【０１４７】

$[R']_x$=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を肘する人物のみが、（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Tx_1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Tx_1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Tx_1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パズルの回は、ネットワーク１０６にTx_2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Tx_2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Tx_2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、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が使用され得る）。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. TxID1 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、それをハッシングし、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Add_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、彼らが成功し、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。
【表２】

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

【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、Tx_1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：
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：Tx_2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_Pと等しいかどうかをチェックし、更に、Tx_2の署名済み部分の知識が与えられた場合に、Tx_2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。

【０１５８】

従って、アウトプットは、Tx_2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。

【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】

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、$r=[k・G]_x$.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り、ここで、kは、rに対応する一時鍵である。

【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できるので（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）。

【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ったために使用されたものと同じであるか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。

【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、Tx_2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、Tx_2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Tx_2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。

【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。Tx_1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、Tx_2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。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。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、s及びP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２１】

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

【０１６８】

ここで、r'は、Tx_1のロックスクリプトから取り入れられ、s及びmは、Tx_2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、Tx_2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。UTXO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、Tx_2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、Tx_2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Tx_2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（Tx_1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）。

【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Checksig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幾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、数学的に言うと追加ステップを含むが、実際には、両方ともTx$_2$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し示す。

【０１７１】

全部の可能な実施形態においてTx$_2$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Tx$_2$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UTXO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Tx$_2$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、Tx$_2$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。Tx$_1$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、Tx$_2$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。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及びP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２２】

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、Tx$_1$のロックスクリプトから取り入れられ、s、r、及びmは、Tx$_2$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、Tx$_2$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Checksig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Checksig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。その後に、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Tx_1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Tx_0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更に詳細な説明は図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】

*TxID*1	
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）Tx_1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、Tx_2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。Tx_1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、$h=H_{puz}(r')$。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、Tx_2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。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及びP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２３】

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、Tx_1のロックスクリプトから取り入れられ、s、r、及びmは、Tx_2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、Tx_2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Checksig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Tx_1内のロックスクリプトの例は以下に示される。

【表４】

*TxID*1	
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。

〈P_B〉〈sig_r〉

【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、$r=[k・G]_x$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、解を見付けた者が誰でもプライバシ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、動作すると考えることができる。
【０１９７】
＜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。
【数２４】
$$s = k^{-1}(H(m) + rV) \bmod n$$

【０１９８】
Vについて解くと、以下を得る：
【数２５】
$$V = r^{-1}(sk - H(m)) \bmod n$$

【０１９９】
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。
【０２００】
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るべきである。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。
【０２０１】
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。
【０２０２】

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。

【０２０３】

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。

【０２０４】

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。

【０２０５】

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、この更なる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）。

【０２０６】

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。

【０２０７】

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。

【０２０８】

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。

【０２０９】

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Ｐ２PKH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。

【０２１０】

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。

【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 ⟨*r*⟩ OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIGVERIFY OP_1 ⟨*Alice's PK*⟩ ⟨*Bob's* P_2⟩ OP_2 OP_CHECKMULTISIG

【０２１１】

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。

【０２１２】

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V_2、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。

【０２１３】

上述の例では、Tx_2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P_2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、次の章で更に詳細に議論される。

【０２１４】

しかしながら、P_2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Tx_1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P_2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。

【０２１５】

＜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$P_{puzzle}=V_{puzzle}\cdot G$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。

【表６】

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

【０２１６】
対応するアンロックスクリプトは、以下に示される。
【数２６】

$$\langle sigP_{ID}\rangle\langle P_{ID}\rangle\langle sigP_{puzzle}\rangle\langle P_{puzzle}\rangle$$

【０２１７】
〈sigP_{ID}〉〈P_{ID}〉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〈sigP_{ID}〉〈P_{ID}〉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。
【０２１８】
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７】

$TxID$	
Inputs	Outputs
Any spending inputs	OP_DUP OP_HASH160 $\langle H(P_{puzzle})\rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_HASH160 $\langle H(r_{puzzle})\rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

【０２１９】
ロックスクリプト内で、
【数２７】

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

【０２２０】
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、〈H(P_{puzzle})〉=〈H(r_{puzzle})〉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。

【表８】

TxID	
Inputs	Outputs
Any spending inputs	OP_DUP OP_HASH160 $\langle H(P_{puzzle})\rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_PUSHDATA1 $\langle r-prefix\rangle$ OP_SWAP OP_CAT OP_HASH160 $\langle H(r_{puzzle})\rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

【０２２１】

（ここで、H(...)は、ロックスクリプトの概略表現において三角括弧〈〉で示され、これは実際には、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２８】

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

【０２２２】

＜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じ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。

【０２２３】

P=V・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。

【数２９】

$$r = [k \cdot G]_x, \text{ 及び}$$

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

【０２２４】

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。

〈P〉〈r,s〉

【０２２５】

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〉

【０２２６】

ここで、P'=V'・Gは、公開／秘密鍵ペアであり、以下の通りである。

【数３０】

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

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

【０２２７】

マイナーは（Vを知らないので）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３１】

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

【０２２８】

署名は、(R ’x)=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。

【０２２９】

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３２】

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

【０２３０】

（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。）

【０２３１】

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〉

【０２３２】

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。

【表９】

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_CHECKSIG

【０２３３】
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。
【０２３４】
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。
【０２３５】
＜共同加算（JOINT ADDITIVE）rパズル＞
rパズル方法は、基本的に２つの単一のrパズルの結合である共同加算rパズルをカバーするよう拡張することもできる（$R_{joint}=R_1+R_2$、ここで、「+」は楕円曲線点加算を意味する）。（１人以上の）証明者は、従って、ロックされた資金を使用するために、両方のパズルを解く必要がある。しかしながら、この場合には、上述の方法と同様に、２つのパズルr_1及びr_2が生成され、それぞれそれら自体の原像（pre-image、プリイメージ）又はそれぞれの解k_1及びk_2を有し、ここで、$r_i=[k_i \cdot G]_x$である。方法は以下のように動作する。
Tx_1内のアンロックスクリプトは、k_1を用いる１つの署名、及びk_2を用いるもう１つの署名を要求する。私たちは、各署名からr_iを抽出し、それらのrとの関係を検証しなければならない。r_1及びr_2は２個の楕円曲線点のx座標なので、それらを単に加算してはならないことに留意することが重要である。代わりに、２個の値の和が２個の点の和のx座標と同じにならないので、和を得るために、点加算を行う必要がある。
楕円曲線点加算に関する幾つかの事実を思い出し、次式により定義される素数位数nの楕円曲線を考える：
【数３３】
$$y^2 = x^3 + ax + b \ mod \ p.$$

ここで、pは素数である。曲線のグループ構造に従い、曲線上の２個の点
【数３４】
$$P_1 = (x_1, y_1) \text{ 及び } P_2 = (x_2, y_2)$$

が加算されて、以下の通り第３の点
【数３５】
$$P_{joint} = (x_3, y_3)$$

を得る。Psは、ここでは、楕円曲線上の一般的な点の表記として使用されることに留意する（それらは公開鍵Pを表さない）。
【０２３６】

【数３６】

$$P_{joint} = P_1 + P_2.$$

点P_{joint}は、Ｐ１及びＰ２を結ぶ線を通過し次にｘ軸に関して反射される楕円曲線上の点であると定義される。点加算の式は次の通りである。

【数３７】

$$x_3 = \lambda^2 - x_1 - x_2 \mod p \qquad (1)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \mod p, \qquad (2)$$

ここで、$P_1 \neq P_2$の場合、以下を得る：

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p \qquad (3)$$

$P_1 = P_2$ の場合、以下を得る：

$$\lambda = \frac{3x_1^2 + a}{2y_1} \mod p. \qquad (4)$$

例えば、幾つかの一般的なブロックチェーンは、特に、楕円曲線パラメータがａ＝０及びｂ＝７により与えられるｓｅｃｐ２５６ｋ１協定を使用する。

【０２３７】

r_iについて同じ値を使用しない限り、点は同じではないので、式（４）を削除できる。更に、主にｘ座標に関心があるので（ｒを扱うので）、式（２）も削除できる。焦点を当てるべき主な式は式（１）である。モジュラ反数：

【数３８】

$$\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

及びモジュラ平方（λ^2）を行う必要を除去するために、トランザクションを生成し及びスクリプト内でその値を使用する前に、λ^2をオフチェーンで計算できる。基本的に、ｘをｒに置き換える状況では、式（１）は以下のようになる：

【数３９】

$$r_{joint} \equiv \lambda^2 - r_1 - r_2 \mod p \qquad (5)$$

【０２３８】

並べ替えると、以下を得る：

【数４０】

$$r_1 + r_2 \equiv \lambda^2 - r_{joint} \mod p \qquad (6)$$

【０２３９】

これは、第１トランザクションTx_1のロックスクリプトが、第２トランザクションTx_2にチャレンジして、以下を満たすr_1及びr_2の値をそのアンロックスクリプトに含めることにより、ｒパズルの中で使用できる：

【数４１】

$$r_1 + r_2 = \lambda^2 - r_{joint} \bmod p$$

望ましくは、Tx_1のロックスクリプトは、共同rパズルの解の２個の部分r_1及びr_2の各々が有効な署名に対応することを確認するために、それぞれの署名(r_1,s_1)及び(r_2,s_2)を更に検証する。

【０２４０】

図１０は、例示的な実装を示す。第２トランザクションTx_2のアンロックスクリプトは、それぞれ秘密鍵V_1及びV_2により生成された少なくとも２個の署名(r_1,s_1)及び(r_2,s_2)を含む。例えば、署名は、２人の異なるパーティBob及びCharlieのそれぞれの秘密鍵に基づく、該２人の異なるパーティの署名であり得る。アンロックスクリプトは、それぞれV_1及びV_2に対応する彼らそれぞれの公開鍵P_1及びP_2も含んでよい。第１トランザクションTx_1のロックスクリプトは、r_{joint}の参照として動作する参照データD_{ref}の部分を含む。これは、共同r値r_{joint}自体の直接の値、又は(λ^2-r)の値、又は$(\lambda^2-r) \bmod n$の値のいずれかを含んでよい。

【０２４１】

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、少なくとも、２個の署名のr部分r_1&r_2及びs部分s_1&s_2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４２】

I) $r_1 + r_2 = \lambda^2 - r_{joint} \bmod p$, をチェックし、

II) $R_1' = H_{sig}(m)s_1^{-1} \cdot G + r_1 s_1^{-1} \cdot P_1$, を計算し、

III) $[R_1']_x = r_1$, をチェックし、

IV) $R_2' = H_{sig}(m)s_2^{-1} \cdot G + r_2 s_2^{-1} \cdot P_2$, を計算し、

V) $[R_2']_x = r_2$, をチェックする。

【０２４２】

注：式の末尾にある「mod p」という記述は、等式を含む式中の全部の演算がmod pの下で行われることを意味する。

【０２４３】

公開鍵P_1及びP_2も、Tx_2内のアンロックスクリプトから取り入れられてよい。代替として、上述のように、任意的にTx_2に含まれる１ビットフラグの助けにより、公開署名Pをそのそれぞれの署名部分（r,s）から導出することが可能である。別の代替案として、公開鍵の一方又は両方が何らかの他の手段により検証者に既に知られていること、検索され得ることが排除されない。

【０２４４】

ロックスクリプトがアンロックスクリプトと一緒に実行されるとき、それらは、３つの条件Ｉ）、ＩＩＩ）、及びＶ）の全部が真であることを条件として、「真」の結果を出力する。これらのチェックは任意の順序で実行でき、上述の参照符号は必ずしも順序を課すことを意味しない（しかし、勿論、チェックＩＩＩは、対応する計算ＩＩの後に実行される必要があり、チェックＶは対応する計算ＩＶの後に実行される必要がある）。上述の表記は、必ずしも、２個の異なる署名の中で使用されるH_{sig}が同じ形式のハッシュ関数でなければならないことを意味しない。

【０２４５】

Tx_1のロックスクリプト内の参照データD_{ref}は、チェックＩ）で使用されるr_{joint}自体

の直接の値を含んでよい。この場合、λ^2及びpの一方又は両方は、それぞれのノード１０４にあるメモリに予め格納され、チェックで使用するためにメモリから取り込まれてよい。代替として、λ^2及びpの一方又は両方は、Tx_1のロックスクリプト内の参照データD_{ref}の中の別個の値として含まれ、チェックで使用するうためにそこから取り入れられてよい。別の可能性として、λ^2及びpの一方又は両方は、証明トランザクションTx_2のアンロックスクリプトに含まれ、チェックＩ）で使用されるべき解の部分として提示され得る。λ^2及びpは、同じソースに由来する必要はなく、これらのソースの任意の組合せから取り入れられ得ることにも留意する。λ^2は、λの形式で等価に格納され又は提示されることができ、平方はチェックＩ）においてその場で（ad hoc）計算されることにも留意する。

【０２４６】

別の代替として、Tx_1のロックスクリプト内の参照データD_{ref}は、(λ^2-r)の値、又は$(\lambda^2-r) \mod p$の値を含んでよい。前者の場合には、他の場所から取得される必要があるのは、pだけである。ここでも、それは、ノード１０４に予め格納されたデフォルト値であってよく、又はTx_1のロックスクリプトに含まれるD_{ref}の別の要素であってよく、又は証明トランザクションTx_1のアンロックスクリプトにより提示される解の部分として含まれてよい。

【０２４７】

＜セキュアなrパズルのハンドオフ＞

更に、任意的な拡張は、共同rパズルを使用して、秘密鍵漏洩脆弱性を防ぐ（任意的なセキュリティ機能１で前述した）。多数のエンティティがkの知識を有し、彼らのうちの１つが署名でkを使用する場合、kの知識を有する全員が、署名に署名するために使用される秘密鍵を導出できる。考え方は、共同rパズルを使用して、$k_1+k_2=k$となるように及び$R_i=k_i \cdot G$であるので同様に（EC点加算を用いて）$R_1+R_2=R$に、kを２個の部分k_1及びk_2に分けることである。これが行われる方法は、共同rパズル（第４章）とほぼ同一であるが、k_1及びk_2は事前に設定されない。これが行われる方法は、スクリプト内点加算技術と同様である。ロックスクリプトには、Rのx及びy座標が設定される。UTXOをアンロックする又は使用するために、支払者は、R_1及びR_2の座標と共にλを提供しなければならないだろう。次に、ロックスクリプトにおいて、Rのx及びy座標と一緒に、ロックスクリプトは、R_1及びR_2座標が楕円曲線上にあることを、次式を用いてチェックしなければならない：

【数４３】

$$y^2 = x^3 + ax + b.$$

【０２４８】

その後、スクリプトは、式１及び式４（以下に示す）の両方が維持されるかをチェックしなければならない。r値は点Rのx座標のみを含むので、式２の検証は必要ない。

【数４４】

$$x_3 = \lambda^2 - x_1 - x_2 \mod p \qquad (1)$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p \qquad (3)$$

$$\lambda(x_2 - x_1) = y_2 - y_1 \mod p \qquad (4)$$

【０２４９】

（１人以上の）証明者は、$k_1+k_2=k$となる任意の値のk_1及びk_2を選択し、その結果、kを知っている任意の他者は、Bobが選択した値k_1及びk_2を知らないので、彼らはBob（又はCharlie等）の秘密鍵を導出することができないことに留意する。彼ら全員が、そ

れらの２個の値の和がkに等しくなることを知っている。アンロックスクリプトは、以下に示す要素を含まなければならない：

【数４５】

$$\langle sig_1' \rangle \langle P_{B1} \rangle \langle sig_{r1} \rangle \langle y_{r_1} \rangle \langle sig_2' \rangle \langle P_{B2} \rangle \langle sig_{r2} \rangle \langle y_{r_2} \rangle \langle x_{inv} \rangle$$

【０２５０】

＜例示的な使用例＞

例として、〈simplified_r〉と呼ばれるλ^2-rの値を取り入れると、以下のように、これをTx$_1$のロックスクリプトに直接追加できる。

【表１０】

TxID	
Inputs	Outputs
Any unspent inputs	OP_DUP OP_4 OP_PICK OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_SWAP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_ADD ⟨*n*⟩ OP_MOD ⟨*simplified_r*⟩ OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIGVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG

【０２５１】

スクリプトは以下のように動作する。

第１行は、スタックの一番上にある両方の署名〈sig$_{r1}$〉及び〈sig$_{r2}$〉の複製を得る。

第２行は、〈sig$_{r1}$〉からr$_1$を抽出する。

第３行は、スタックの１番上の２個のアイテムをスワップして、スタックの１番上の〈sig$_{r2}$〉を得て、それからr$_2$を抽出する。

第４行は、r$_1$をr$_2$に加算し、mod pをして、（６）に示すように、その値が〈simplified_r〉に等しいかどうかをチェックする。

第５行は、第１署名ペアが、それらの対応する公開鍵に対して有効であることをチェックする。

第６行は、第２署名ペアが、それらの対応する公開鍵に対して有効であることをチェックする。

【０２５２】

Tx$_2$内の対応するアンロックスクリプトは、r$_1$及びr$_2$の両方（これらはRのx座標である）、それらのそれぞれのy座標y$_1$及びy$_2$、及びx$_{inv}$=(r$_1$-r$_2$)$^{-1}$の値を含む。与えられた例では、以下に示すものと同様である。ここで、署名sig$_{r1}$P$_{B1}$はr$_1$を使用し、及びsig$_{r2}$P$_{B2}$はr$_2$を使用し、証明者Bob（B）は任意の秘密／公開鍵ペアを使用して署名を計算できる。

【数４６】

$$\langle sig_1' \rangle \langle P_{B1} \rangle \langle sig_{r1} \rangle \langle y_{r_1} \rangle \langle sig_2' \rangle \langle P_{B2} \rangle \langle sig_{r2} \rangle \langle y_{r_2} \rangle \langle x_{inv} \rangle .$$

【０２５３】

追加の署名sig'は、セキュリティのための追加特徴である（任意的なセキュリティ特徴＃２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。sig$_1$'はr$_1$と異なるr値を使用し、sig$_2$'はr$_2$と異なるr値を使用することに留意する。上述の章で指摘したように、これは、他者がトランザクションを変更し、署名を偽造することを防ぐためのものである。

【０２５４】
この方式は、$r=r_1+\ldots+r_l$のように、l個の加算をカバーするよう拡張できる。この方式は、２個の点の加算について行うことができるので、帰納法により、これは、複数回の２個の点の加算のための技術を繰り返すことにより、l個の加算をカバーするよう拡張できる。

【０２５５】
加算rパズルの例示的な使用例として、考え方は、参加者のグループがトランザクションからの何らかのアウトプットをアンロックできるようにするために利用できる。

【数４７】
$R_1+R_2=R_3+R_4=R_5+R_6=R$ になるように、 楕円曲線点の集合 $\{R_1,R_2,R_3,R_4,R_5,R_6\}$ を考える。ここで、$R_i=k_i\cdot G$

【０２５６】
３人の人々：Bob、Charlie、及びDoraのグループを考える。次に、以下を行うことにより、Rパズルトランザクションによりエミュレートされる２-out-of-３マルチシグUTXOを構成できる。

１．k_1及びk_3をAliceに配信する。
２．k_1及びk_5をBobに配信する。
３．k_4及びk_6をCharlieに配信する。
４．以下のステップを実行するロックスクリプトを有するｔトランザクションを構成する：
（ｉ）sig_{r1}及びsig_{r2}からr_1及びr_2を抽出するセクション、
（ｉｉ）x_{inv}が実際に反数であることをチェックする、つまり：

【数４８】
$(r_1-r_2)^{-1}\cdot(r_1-r_2)==1$

（ｉｉｉ）点がEC上にあるｋとおをチェックする、つまり：

【数４９】
$y_{r_1}^2=(r_1^3+ar_1+b)\ mod\ p$ 及び $y_{r_2}^2=(r_2^3+ar_2+b)\ mod\ p$

（ｉｖ）以下を計算するセクション：

【数５０】
$\lambda\coloneqq(y_{r_1}-y_{r_2})(r_1-r_2)^{-1}\ mod\ p$

（ｖ）以下をチェックする：

【数５１】
$r=\lambda^2-r_1-r_2\ \ mod\ n$

ここで、rは事前に指定される。

【０２５７】
パーティの各々が２個のksをそれぞれ使用する理由は、２個のR点の各々の和が３人の人々の部分集合をカバーするからである。例えば、R_1+R_2は（２ of ３マルチシグで）Alice及びBobの部分集合をカバーし、一方、R_1+R_2はAlice及びCharlieの部分集合をカバーする、等である。

【０２５８】
ステップ４に記載されたロックスクリプトは、２個の署名(r',s')及び(r'',s'')を要求し、ここで、r'+r''=simplified_rであることに留意する（この文脈におけるプライム表記は異

なる署名を表し、プライムが知識署名における提出されたインスタンスと対照的に使用された前の章における参照インスタンスではないことに留意する）。これらの署名のうちの１つ目は１つのパーティからである。２つ目は別のパーティからである。

【０２５９】

従って、スクリプトをアンロックする３つの可能な方法がある：

１．Alice及びBobは、それぞれk_1及びk_2により署名する：

【数５２】

$$\langle sig_1' \rangle \langle P_{B1} \rangle \langle sig_{r1} \rangle \langle y_1 \rangle \langle sig_2' \rangle \langle P_{B2} \rangle \langle sig_{r2} \rangle \langle y_2 \rangle \langle x_{inv} \rangle$$

２．Alice及びCharlieは、それぞれk_3及びk_4により署名する：

【数５３】

$$\langle sig_3' \rangle \langle P_{B3} \rangle \langle y_3 \rangle \langle sig_{r3} \rangle \langle sig_4' \rangle \langle P_{B4} \rangle \langle sig_{r4} \rangle \langle y_4 \rangle \langle x_{inv} \rangle$$

３．Bob及びCharlieは、それぞれk_5及びk_6により署名する：

【数５４】

$$\langle sig_5' \rangle \langle P_{B5} \rangle \langle y_5 \rangle \langle sig_{r5} \rangle \langle sig_6' \rangle \langle P_{B6} \rangle \langle sig_{r6} \rangle \langle y_6 \rangle \langle x_{inv} \rangle$$

【０２６０】

全部のシークレットを生成しそれらをグループに配信する信頼できる第三者が存在すると仮定する。この仮定は、例えば、組織設定、例えば事業体において正当性を示すことができる。

【０２６１】

適用は、R_i個の点の異なる組み合わせを選択することにより生成できる。例えば、$R=R_1+R_2+R_4$を設定することは、３人全員のメンバーが署名することを強制する。

【０２６２】

更に、幾つかのアカウントに基づくモデルでは、１つのトランザクションTx_1^{acc}のスマートコントラクトが、２個（以上）の異なる第２トランザクション$Tx_{2,1}^{acc}$、$Tx_{2,2}^{acc}$からインプットを受信することが可能であってよい。従って、第１トランザクションTx_1のスマートコントラクト内に設定されたチャレンジは、２個（以上）の異なる第２トランザクション$Tx_{2,1}^{acc}$、$Tx_{2,2}^{acc}$からのデータの組み合わせにより満たされるべきである。例えば、１つの第２トランザクション$Tx_{2,1}^{acc}$は、Bobの署名(r_1,s_1)（必要に応じて公開鍵P_1又はフラグflg_1である）を含み、もう１つの第２トランザクション$Tx_{2,2}^{acc}$はCharlieの署名(r_2,s_2)（必要に応じて、彼の公開鍵P_2又はフラグflg_2である）を含み得る。これらは、Tx_1のスマートコントラクト内に設定された共同加算rパズルを解くために一緒に使用され得る。

【０２６３】

＜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。

【０２６４】

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、その後にスマートコントラクトに署名を提供し得る。

【０２６５】

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。

【０２６６】

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。

【０２６７】

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。

【０２６８】

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。

【０２６９】

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。

【０２７０】

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。

【０２７１】

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。

【０２７２】

図１１は、図７～１０に関して上述したものと同じ又は同様のrパズル機能を実装し得るトランザクションTx_1^{acc}及びTx_2^{acc}のペアも示す。それぞれは、（ソースアドレスフィールド内の）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Tx_1^{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Tx_2^{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Tx_1^{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Tx_2^{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Tx_1^{acc}及びTx_2^{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。

【０２７３】

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。

【０２７４】

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」、送信されると言える。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た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、TX_1^{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、TX_2^{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、TX_2^{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TX_2^{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、TX_2^{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、TX_2^{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）。

【０２７５】

プロトコルエンジン４０１accは、TX_2^{acc}を受信すると、それが有効であることを条件に、TX_2^{acc}内の宛先アドレス１１０３bと一致するアカウントを探す。Tx_1^{acc}が処理され、有効であるとすると、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たらし得る（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）。

【０２７６】

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。

【０２７７】

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$Tx_{1,\alpha}^{acc}$により生成されてよい。別個の「中間」トランザクション$Tx_{1,\beta}^{acc}$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。

【０２７８】

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。

【０２７９】

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。

【０２８０】

＜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。

【０２８１】

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）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。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、共同r値r_{joint}に基づき定義されるチャレンジを評価するための参照データを含む、ステップと、

第１ECDSA署名i=１,２のペアのうちのそれぞれのECDSA署名のそれぞれのr部分r_i及びs部分s_iを少なくとも含む情報を含む１つ以上の第２トランザクションを受信するステップであって、前記第１ECDSA署名のうちのそれぞれは、それぞれの第１公開鍵P_iに対応するそれぞれの第１秘密鍵V_iに基づき前記１つ以上の第２トランザクションのうちの１つの第２トランザクションの部分に署名する、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、前記第１トランザクション内の前記参照データ及び前記１つ以上の第２トランザクションの中で受信された前記r部分r_iに基づき、前記チャレンジが満たされるかどうかを検証し、それを条件として真の値を返すよう構成され、前記チャレンジは、以下の基準：

【数５５】

$$r_1 + r_2 = \lambda^2 - r_{joint} \bmod p,$$

を含み、ここで、r_1+r_2はスカラー加算を示し、

【数５６】

$r_{joint} = [R_{joint}]_x$ ，楕円曲線点加算により $R_{joint} = R_1 + R_2$ 、

pは素数モジュラスであり、

【数５７】

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p,\ R_i = k_i \cdot G,\ x_i = [R_i]_x,\ y_i = [R_i]_y,$$

k_iは一時鍵であり、Gは楕円曲線生成元であり、$[]_x$は[…]のx座標を示し、$[]_y$は[…]のy座標を示し、「・」は楕円曲線スカラー乗算を示す、ステップと、

を含む。

【０２８２】

参照データは、共同r値r_{joint}自体の直接の値、又は(λ^2-r_{joint})の値、又は(λ^2-r_{joint}

)modnの値を含んでよい。

【０２８３】

実施形態では、第１トランザクション内の参照データは、共同r値r_{joint}自体を含んでよい。この場合、pは、検証ノードにおいて予め格納されてよく、又は参照データに含まれてもよく、又は１つ以上の第２トランザクションのうちの少なくとも１つの中で受信された情報の更なる部分として受信されてよい。同様に、λ又はλ^2は、検証ノードにおいて予め格納されてよく、又は参照データに含まれてもよく、又は１つ以上の第１トランザクションのうちの少なくとも１つの中で受信された情報の更なる部分として受信されてよい。本願明細書で、r_{joint}、p、λ、又はλ^2等のような何らかの値が参照データに含まれる、又は格納される若しくは他の場所から提供される、等と示される場合、値は、直接又は等価的に表現されることができ、逆変換することによりノードにより値を決定させることができる変換又は符号化形式で表現され得ることにも留意する。

【０２８４】

本開示の教示の第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、λ^2を示すデータが、１つ以上の第２トランザクションのうちの少なくとも１つの中で受信される情報の部分として受信され、前記の決定は、λ^2を示す受信されたデータに基づき実行される、方法が提供され得る。

【０２８５】

代替として、第１トランザクション内の参照データはλ^2-rを含んでよい。この場合、pは、検証ノードにおいて予め格納されてよく、第１トランザクション内の参照データの部分であってよく、又は１つ以上の第２トランザクションの中で受信された情報の更なる部分として受信されてよい。代替として、第１トランザクションに含まれる参照データは、$(\lambda^2\text{-}r_{joint})\ \mathrm{mod}\ p$を含んでよく、この場合、追加要素が任意の第２トランザクション内で受信される又は検証ノードに予め格納されている必要がない。

【０２８６】

本開示の第３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、前記第１トランザクション内の前記参照データは、p及びλ^2-rの値を含む、方法が提供され得る。

【０２８７】

第４の任意的な具体化によると、第１、第２、又は第３の具体化による方法であって、前記方法は、

前記第１ECDSA署名のうちの各々の第１ECDSA署名のそれぞれの第１公開鍵を取得し、ECDSAの検証関数を適用して、前記それぞれの第１公開鍵及び署名済み部分に基づき、各第１ECDSA署名を検証するステップであって、前記コードは、前記第１ECDSA署名のうちの各第１ECDSA署名の前記検証を更なる条件として、真の値を返すよう構成される、ステップを含む方法が提供され得る。

【０２８８】

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。

【０２８９】

本開示の第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、前記第１公開鍵を取得する前記ステップは、前記１つ以上の第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。

【０２９０】

代替として、前記取得するステップは、前記それぞれのECDSA署名の前記r部分及びs部分の組み合わせから、前記第１公開鍵の各々を導出するステップを含んでよい。

【０２９１】

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付けられ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、又はデータソース内の前記第１公開鍵を検索するステップ、を含み得ることが排除されない。

【０２９２】

第６の任意的な具体化によると、第１～第５の具体化のいずれかによる方法であって、前記コードは、前記第１ECDSA署名のうちの１つ、一部、又は全部のそれぞれの第１公開鍵として、誰の公開鍵が使用されたかに関係なく、真の結果を出力するよう構成される、方法が提供され得る。

【０２９３】

第７の任意的な具体化によると、第１～第６の具体化のいずれかによる方法であって、前記ECDSAは以下の形式：$y^2=x^3+7$の楕円曲線に基づく、方法が提供され得る。

【０２９４】

第８の任意的な具体化によると、第１～第７の具体化のいずれかによる方法であって、前記ECDSAはsecp２５６k１アルゴリズムである、方法が提供され得る。

【０２９５】

第９の任意的な具体化によると、第１～第８の具体化のいずれかによる方法であって、前記第１ECDSA署名の各々は、異なるそれぞれの第２パーティの署名であり、前記チャレンジは少なくとも部分的に第１パーティにより定義される、方法が提供され得る。

【０２９６】

各パーティは、例えば、それぞれの個人、又はそれぞれの組織であってよい。

【０２９７】

第１０の任意的な具体化によると、第９の具体化のによる方法であって、前記第１ECDSA署名の各々の前記r部分及びs部分は、それぞれの一時鍵k_iを用いてそれぞれの第２パーティにより生成される、方法が提供され得る。

【０２９８】

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、それぞれの一時鍵は前記第１パーティによりそれぞれの第２パーティに与えられるか、又はその逆である、方法が提供され得る。

【０２９９】

本開示の第１２の任意的な具体化によると、第１１の具体化による方法であって、

【数５８】

$$P_i = V_i \cdot G\ ,\quad k_i \in [1,n-1]\ ,\quad R_i = k_i \cdot G\ ,\quad r_i = [R_i]_x\ ,\quad 及び\ s_i = k_i^{-1}\left(H_{sig}(m) + r_i V_i\right) mod\ n,$$

【０３００】

ここで、k_iはそれぞれの一時鍵であり、m_iは前記第２トランザクションのそれぞれの署名済み部分であり、H_{sig}はそれぞれの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、nは前記生成元の素数位数である、方法が提供され得る。前記署名済みメッセージmは、前記２つの異なる署名について異なる又は同じであってよい。nの値及びハッシュ関数H_{sig}の形式は、両方の署名について同じであってよい。

【０３０１】

第１３の任意的な具体化によると、第７～第１２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】

第１４の任意的な具体化によると、第７～第１３のいずれかの具体化による方法であって、前記１つ以上の第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パーティのうちの１つから前記第２トランザクションの各々を受信するステップを含む、方法が提供され得る。

【０３０５】

代替として、前記１つ以上の第２トランザクションは、代わりに、全部の第２パーティから前記署名を受信した第三者から受信され得る。

【０３０６】

第１５の任意的な具体化によると、第７～第１４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パーティのうちの少なくとも１つの更なる秘密鍵を用いて前記第２パーティのうちの該少なくとも１つ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。

【０３０７】

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。

【０３０８】

第１６の任意的な具体化によると、第１５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少なくとも１つの第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。

【０３０９】

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。

【０３１０】

第１７の任意的な具体化によると、第１５又は第１６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。

【０３１１】

第１８の任意的な具体化によると、第７～第１７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。

【０３１２】

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書名を検証し、前記第１パーティの前記暗号書名を更なる条件として真の結果を返すよう構成される。

【０３１３】

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。

【０３１４】

第１９の任意的な具体化によると、第７～第１８のいずれかの具体化による方法であって、前記第２トランザクション内で受信された前記情報は、前記第１ECDSA署名と同じそれぞれの秘密鍵を用いるが、前記第１ECDSA署名と異なるr部分の値を有する、前記第２パーティのうちの１つ、一部、又は全部の各々の追加ECDSA署名を含み、

前記コードは、それぞれの前記第１公開鍵を用いて前記追加ECDSA署名の各々を検証し、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。

【０３１５】

実施形態では、前記追加ECDSA署名は、それぞれの第１ECDSA署名と異なるメッセージに署名してよい。

【０３１６】

第１２の任意的な具体化によると、第１～第１９のいずれかの具体化による方法であって、前記コードは、r部分の値の大きな集合の中から任意の２つのr部分の値がr_1及びr_2として使用可能にするよう構成される、方法が提供され得る。

【０３１７】

第２１の任意的な具体化によると、第７～第１９のいずれかの具体化に従属する第１２の具体化による方法であって、前記集合は、r部分の値の少なくとも３個のペアを含み、

k値の異なるそれぞれのペアは、少なくとも３つの第２パーティのそれぞれに分配されて、それらが集合内のペアのそれぞれを生成できるようにし、

r_1は前記第２パーティのうちの１つの第２パーティからのr部分の値のうちの１つであり、r_2は前記第２パーティのうちの別の第２パーティからのr部分の値であり、

前記コードは、前記少なくとも３つの第２パーティのうちの任意の２つが前記チャレンジを満たすことを可能にする、方法が提供され得る。

【０３１８】

第２２の任意的な具体化によると、第１～第２１の具体化のいずれかによる方法であって、前記情報は同じ第２トランザクション内で受信される、方法が提供され得る。つまり、前記１つ以上の第２トランザクションは、たった１つの第２トランザクションである。

【０３１９】

第２３の任意的な具体化によると、第２２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と含み、

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：

前記の検証コードにより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、

を含む方法が提供され得る。

【０３２０】

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。

【０３２１】

第２４の任意的な具体化によると、第１～第２２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。

【０３２２】

第２５の任意的な具体化によると、第２４の具体化による方法であって、前記情報は複数の第２トランザクションに渡り受信される、方法が提供され得る。

【０３２３】

例えば、前記複数の第２トランザクションは、前記第２パーティの各々から受信されるそれぞれのトランザクションを含んでよい。各第２トランザクション内で受信される前記情報は、それぞれの第２パーティのr部分r_iを含んでよい。各第２トランザクション内で受信される前記情報は、それぞれの第２パーティの公開鍵P_iを含んでよい。各第２トランザクション内で受信される前記情報は、それぞれの第２パーティの更なる署名及び／又は追加署名を含んでよい。

【０３２４】

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。

【０３２５】

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１ECDSA署名の少なくとも前記s部分（及び、実施形態では、前記第１ECDSA署名の前記ｒ部分の前記提出されたインスタンス及び／又は前記データ要素）に基づき、第三者により生成される。

【０３２６】

本開示の第２６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れ、ネット

ワークのノード上で実行すると第１～第２５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。

【０３２７】

本開示の教示の第２７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２５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。

【０３２８】

本開示の教示の第２８の具体化によると、ブロックチェーンに記録するための第１トランザクションであって、前記第１トランザクションは、１つ以上のコンピュータ可読媒体上に具現化され、実行可能コードを含む、第１トランザクションが提供される。前記コードは、第１ECDSA署名のペアi=１,２のそれぞれのr部分r_i及びs部分s_iを含む情報を少なくとも含む１つ以上の第２トランザクションに基づき、共同r値に基づき定義されたチャレンジを評価するための参照データを含み、前記第１ECDSA署名の各々は、それぞれの第１公開鍵P_iに対応するそれぞれの第１秘密鍵V_iに基づき、前記１つ以上の第２トランザクションのうちの１つの部分に署名する。前記コードは、前記第１トランザクション内の前記参照データ及び前記１つ以上の第２トランザクション内で受信されたr部分r_iに基づき、前記チャレンジが満たされるかどうかを検証し、それを条件として真の結果を返すよう構成され、前記チャレンジは、以下の基準を含み：

【数５９】

$$r_1 + r_2 = \lambda^2 - r_{joint} \; mod \; p,$$

ここで、r_1+r_2はスカラー加算を示し、

【数６０】

$r_{joint} = [R_{joint}]_x$ ，楕円曲線点加算により$R_{joint} = R_1 + R_2$、

pは素数モジュラスであり、

【数６１】

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad mod \; p, \; R_i = k_i \cdot G, \; x_i = [R_i]_x, \; y_i = [R_i]_y$$、

k_iは一時鍵であり、Gは楕円曲線生成元であり、$[]_x$は[…]のx座標を示し、$[]_y$は[…]のy座標を示し、「・」は楕円曲線点乗算を示す。

【０３２９】

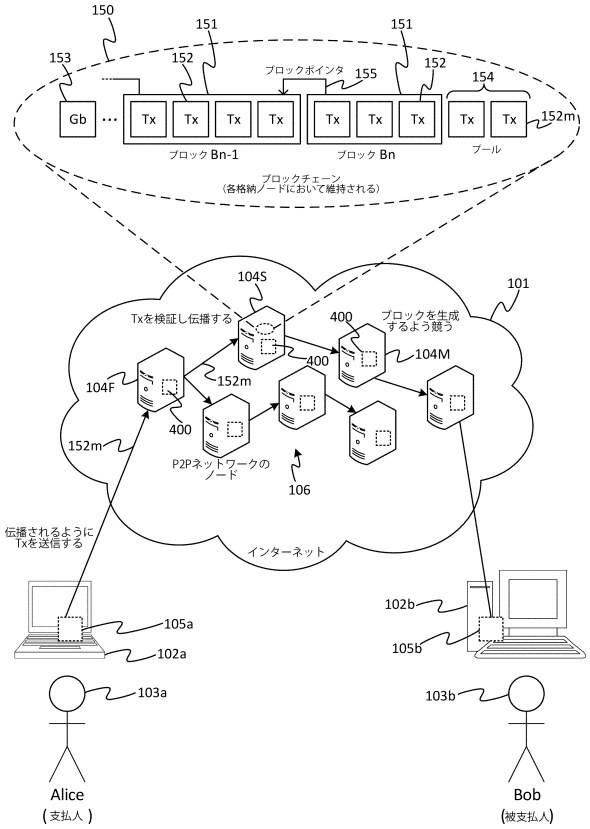
実施形態では、前記第１トランザクションは、本願明細書に開示された具体化又は他の特徴のいずれかにより構成されてよい。

【０３３０】

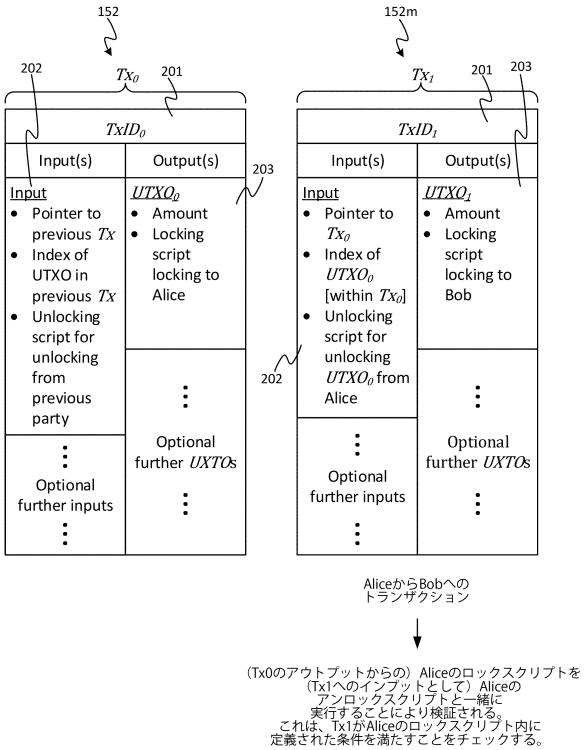
開示された技術の他の変形例又は使用事例は、本明細書で開示されると、当業者に明らかになり得る。本開示の範囲は、記載された実施形態によって限定されるものではなく、添付の特許請求の範囲によってのみ限定される。

【図面】

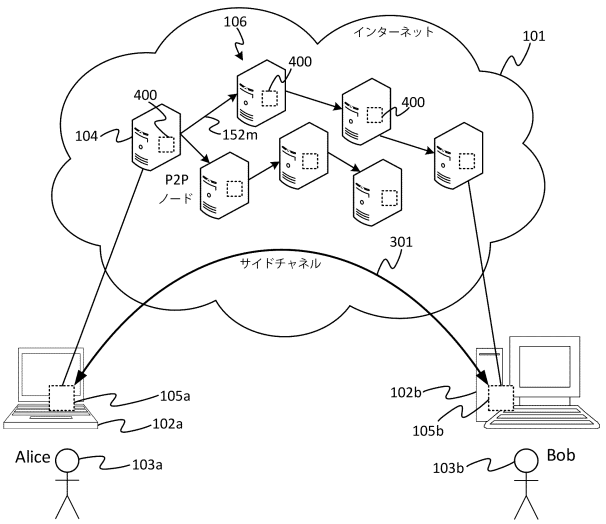
【図１】


150
151
151
153
152
ブロックポインタ
155
152
154
Gb
Tx
Tx
Tx
Tx
Tx
Tx
Tx
Tx
Tx
152m
ブロック Bn-1
ブロック Bn
プール
ブロックチェーン
（各格納ノードにおいて維持される）
104S
Txを検証し伝播する
400
ブロックを生成するよう競う
101
104M
104F
152m
400
400
152m
P2Pネットワークのノード
106
伝播されるように
Txを送信する
インターネット
105a
102a
102b
105b
103a
103b
Alice
（支払人）
Bob
（被支払人）

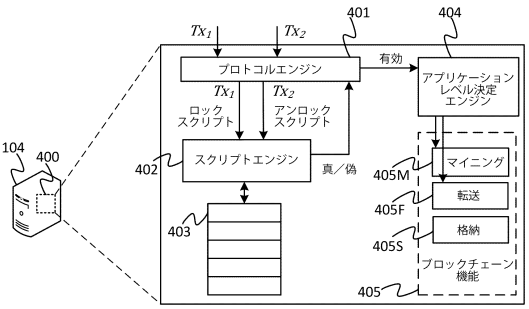
【図２】


152
202
Tx_0
201
$TxID_0$
Input(s)
Output(s)
203
Input
• Pointer to previous Tx
• Index of UTXO in previous Tx
• Unlocking script for unlocking from previous party
$UTXO_0$
• Amount
• Locking script locking to Alice
Optional further inputs
Optional further $UXTOs$
152m
Tx_1
201
203
$TxID_1$
Input(s)
Output(s)
202
Input
• Pointer to Tx_0
• Index of $UTXO_0$ [within Tx_0]
• Unlocking script for unlocking $UTXO_0$ from Alice
$UTXO_1$
• Amount
• Locking script locking to Bob
Optional further inputs
Optional further $UXTOs$
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。

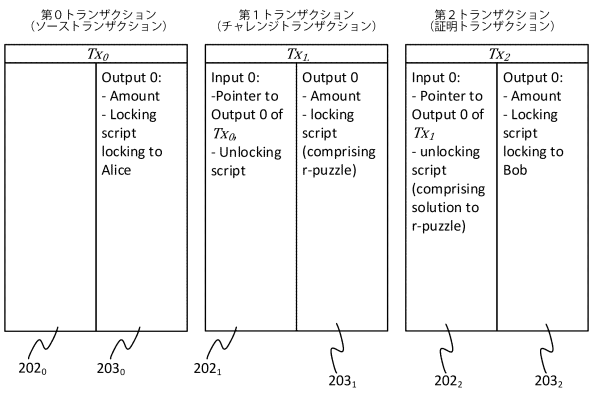
【図３】


106
インターネット
101
400
400
400
104
152m
P2P
ノード
301
サイドチャネル
105a
102a
102b
105b
Alice
103a
103b
Bob

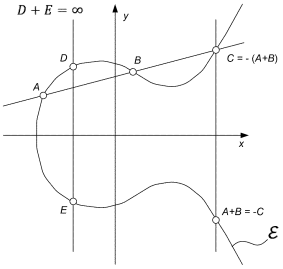
【図４】


Tx_1
Tx_2
401
404
プロトコルエンジン
有効
アプリケーションレベル決定エンジン
Tx_1
ロックスクリプト
Tx_2
アンロックスクリプト
104
400
スクリプトエンジン
真／偽
402
403
マイニング
405M
転送
405F
格納
405S
ブロックチェーン機能
405

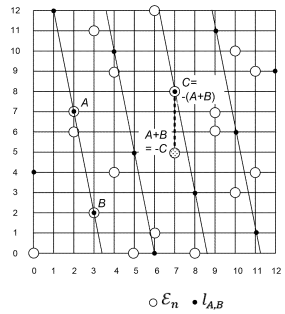
【図５】



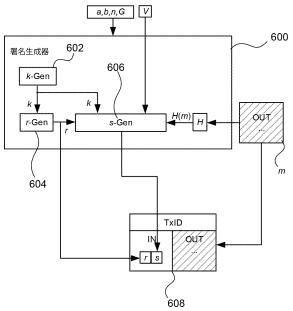
【図６Ａ】



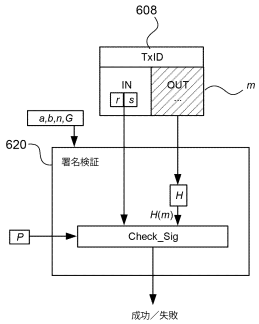
【図６Ｂ】



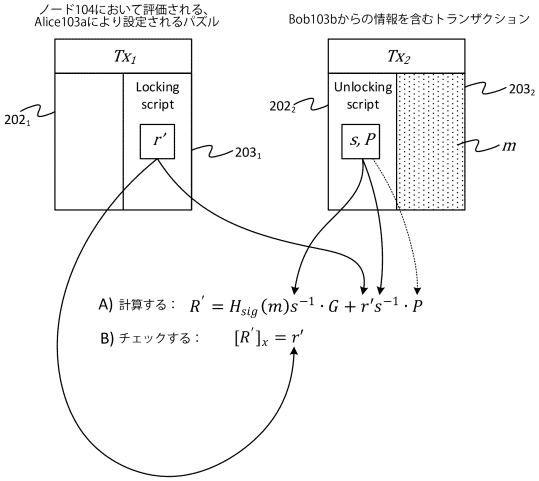
【図６Ｃ】



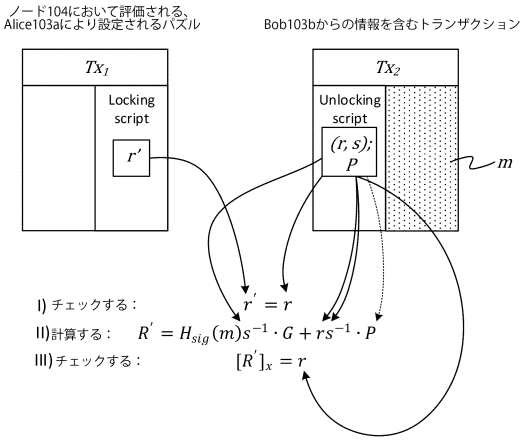
【図６Ｄ】



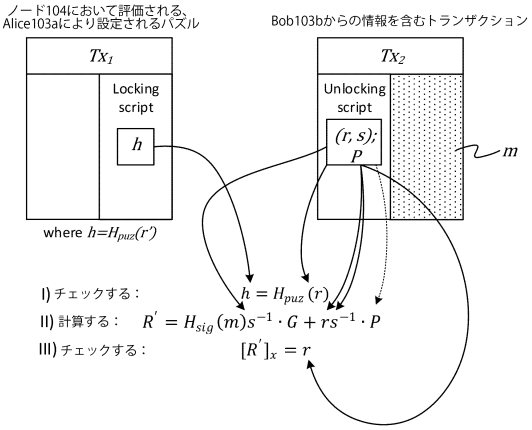
【図７】



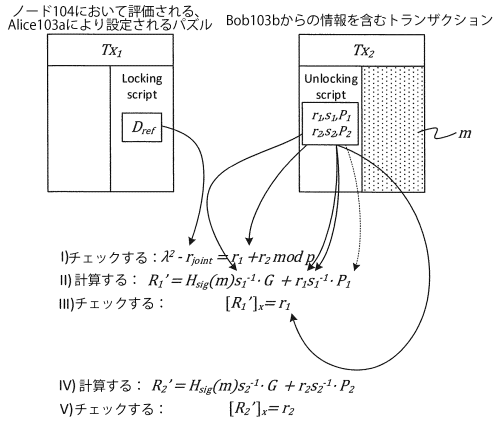
【図８】


ノード104において評価される、
Alice103aにより設定されるパズル
Bob103bからの情報を含むトランザクション
Tx_1
Locking script
r'
Tx_2
Unlocking script
(r; s);
P
m
I) チェックする：
II)計算する：
III)チェックする：

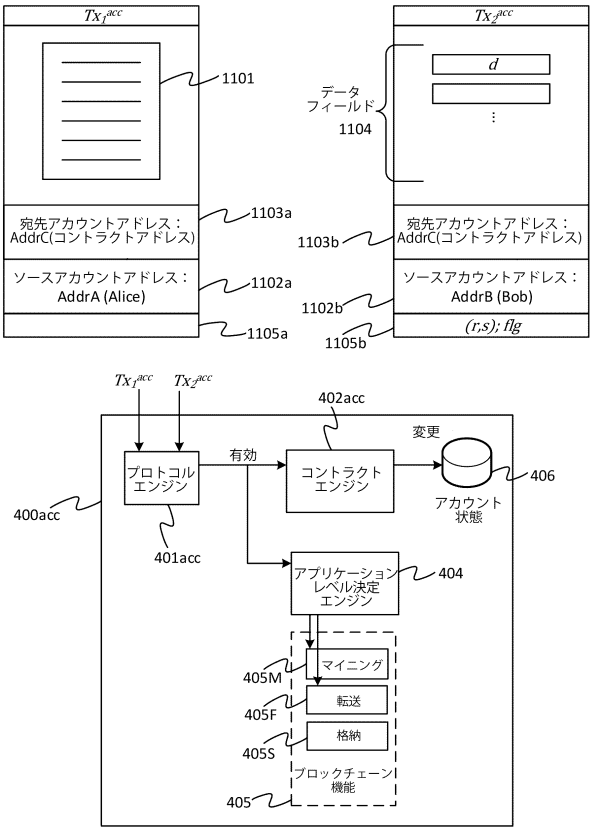
【図９】


ノード104において評価される、
Alice103aにより設定されるパズル
Bob103bからの情報を含むトランザクション
Tx_1
Locking script
h
where $h=H_{puz}(r')$
Tx_2
Unlocking script
(r, s);
P
m
I) チェックする：
II) 計算する：
III) チェックする：

【図１０】


ノード104において評価される、
Alice103aにより設定されるパズル
Bob103bからの情報を含むトランザクション
Tx_1
Locking script
D_{ref}
Tx_2
Unlocking script
r_1,s_1,P_1
r_2,s_2,P_2
m
I)チェックする：$\lambda^2 - r_{joint} = r_1 + r_2 \bmod p$
II) 計算する：$R_1' = H_{sig}(m)s_1^{-1} \cdot G + r_1 s_1^{-1} \cdot P_1$
III)チェックする：$[R_1']_x = r_1$
IV) 計算する：$R_2' = H_{sig}(m)s_2^{-1} \cdot G + r_2 s_2^{-1} \cdot P_2$
V)チェックする：$[R_2']_x = r_2$

【図１１】


Tx_1^{acc}
1101
宛先アカウントアドレス：
AddrC(コントラクトアドレス)
1103a
ソースアカウントアドレス：
AddrA (Alice)
1102a
1105a
Tx_2^{acc}
データ
フィールド
1104
d
1103b
宛先アカウントアドレス：
AddrC(コントラクトアドレス)
1102b
ソースアカウントアドレス：
AddrB (Bob)
(r;s); flg
1105b
402acc
変更
有効
プロトコル
エンジン
コントラクト
エンジン
406
アカウント
状態
400acc
401acc
アプリケーション
レベル決定
エンジン
404
マイニング
405M
転送
405F
格納
405S
ブロックチェーン
機能
405

【図１２】

【図１３】

Stack	Script	Description
empty	$\langle P_A \rangle \langle sig_A \rangle$ OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	scriptSig及びScriptPubKeyが結合される
$\langle P_B \rangle \langle sig_A \rangle$	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ 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
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A \rangle$ OP_3	OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ 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
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A[:3] \rangle \langle sig_A[3:] \rangle$	OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS(sig)が、len(r)の直前の第３バイトで分割される
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A[3:] \rangle$ OP_1	OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS7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
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A[4:] \rangle \langle sig_A[3:4] \rangle$	OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToSがバイトを分割し、(sigの上位２つのアイテムをスワップする
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A[4:36/37] \rangle$ $\langle sig_A[36/37:] \rangle$	OP_DROP $\langle r \rangle$ 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(r)= $\langle sig_A[3:4] \rangle$ バイトで分割され、これは必ず32又は33バイトになるので36又は37になる
$\langle P_B \rangle \langle sig_A \rangle \langle sig_A[4:36/37] \rangle$ $\langle r \rangle$	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS（sigの残り）をトロップし、<r>をスタックにプッシュする
$\langle sig_A \rangle \langle P_B \rangle$	OP_CHECKSIG	$\langle sig_A[4:36/37] \rangle = \langle r \rangle$ 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

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　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　米国特許出願公開第２０１０／０３０８９７８（ＵＳ，Ａ１）
特開２０１８－０９３４３４（ＪＰ，Ａ）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ　９／０８，９／３２
Ｇ０９Ｃ　１／００