(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7620687号
(P7620687)

(45)発行日　令和7年1月23日(2025.1.23)

(24)登録日　令和7年1月15日(2025.1.15)

(51)国際特許分類
H04L 9/32 (2006.01)

FI
H04L 9/32 200Z
H04L 9/32 200B

請求項の数 8 外国語出願 （全19頁）

(21)出願番号 特願2023-190064(P2023-190064)
(22)出願日 令和5年11月7日(2023.11.7)
(62)分割の表示 特願2021-518541(P2021-518541)の分割
原出願日 令和1年10月9日(2019.10.9)
(65)公開番号 特開2023-184657(P2023-184657A)
(43)公開日 令和5年12月28日(2023.12.28)
審査請求日 令和5年11月7日(2023.11.7)
(31)優先権主張番号 1816936.7
(32)優先日 平成30年10月17日(2018.10.17)
(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 ライト，クレイグ スティーヴン
イギリス国 シーエフ１０ ２エイチエイチ カーディフ チャーチル ウェイ チャーチル ハウス ７ス フロア アーカート－ダイクス アンド ロード エルエルピー

最終頁に続く

(54)【発明の名称】 公開鍵結合検証を含む、コンピュータにより実施されるシステム及び方法

(57)【特許請求の範囲】

【請求項１】

コンピュータにより実施される方法であって、

公開鍵結合検証関数を含むアウトプットを有する第１ブロックチェーントランザクションを取得するステップであって、前記関数は、複数の公開鍵と前記公開鍵の結合から導出されるグループ公開鍵とを含み、前記第１ブロックチェーントランザクションは、リソースへのアクセスを許可し又はリソースの制御を移転するために償還可能なように構成される、ステップと、

前記公開鍵に関連する少なくとも１つの勾配値を決定するステップと、

前記公開鍵と、前記少なくとも１つの勾配値と、前記グループ公開鍵とを含むインプットを含む第２ブロックチェーントランザクションを提供するステップと、

を含み、

前記第１ブロックチェーントランザクションは、前記グループ公開鍵が前記公開鍵の結合から導出されたことを検証するために、前記インプットへの前記公開鍵結合検証関数の適用が成功すると、リソースへのアクセスを許可し又はリソースの制御を移転するために償還可能なように構成される、方法。

【請求項２】

前記公開鍵結合検証関数の適用の成功は、

（ｉ）第１公開鍵及び第２公開鍵に関連する勾配値を用いて前記第１及び第２公開鍵に前記関数を適用して、第１結果を得るステップと、

（ｉｉ）前記第１結果及び第３公開鍵に関連する勾配値を用いて前記第１結果及び前記第３公開鍵に前記関数を適用して、第２結果を得るステップと、
を含む、請求項１に記載の方法。
【請求項３】
各反復で更なる公開鍵を用いて、前記（ｉｉ）を少なくとも１回繰り返すステップ、を更に含む請求項２に記載の方法。
【請求項４】
前記第２ブロックチェーントランザクションをブロックチェーンへ提出するステップ、を更に含む請求項１～３のいずれかに記載の方法。
【請求項５】
前記第１ブロックチェーントランザクションの前記アウトプットは、
前記グループ公開鍵に対応するグループ署名と、
マークルルートと、
マークルパス検証関数と、
を更に含み、
前記第２ブロックチェーントランザクションの前記インプットは、
前記複数の公開鍵に関連付けられたマークルパスと、
前記グループ署名に対応するグループ秘密鍵と、
を含む、請求項１～４のいずれかに記載の方法。
【請求項６】
前記第１ブロックチェーントランザクションの前記アウトプットは、
前記複数の公開鍵に対応する複数の署名と、
マークルルートと、
マークルパス検証関数と、
を更に含み、
前記第２ブロックチェーントランザクションの前記インプットは、
前記複数の公開鍵に関連付けられたマークルパスと、
前記複数の署名に対応する複数の秘密鍵と、
を含む、請求項１～４のいずれかに記載の方法。
【請求項７】
システムであって、
プロセッサと、
前記プロセッサによる実行の結果として、前記システムに請求項１～６のいずれか一項に記載のコンピュータにより実施される方法を実行させる実行可能命令を含むメモリと、
を含むシステム。
【請求項８】
実行可能命令を記憶した非一時的コンピュータ可読記憶媒体であって、前記実行可能命令は、コンピュータシステムのプロセッサにより実行された結果として、前記コンピュータシステムに、請求項１～６のいずれか一項に記載のコンピュータにより実施される方法を少なくとも実行させる、非一時的コンピュータ可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、概して、リソース制御及び／又はアクセスの移転に関し、より詳細には、ブロックチェーン上で暗号マルチシグネチャ方法を用いる、このような制御及び／又はアクセスの移転に関する。本開示は、限定ではなく、Ｂｉｔｃｏｉｎブロックチェーン又はＢｉｔｃｏｉｎプロトコルの任意の変形において使用することに特に適する。
【背景技術】
【０００２】
本願明細書では、私たちは、全ての形式の電子的な、コンピュータに基づく、分散型台

帳を包含するために用語「ブロックチェーン」を使用する。これらは、総意に基づくブロックチェーン及びトランザクションチェーン技術、許可及び未許可台帳、共有台帳、並びにこれらの変形を含む。他のブロックチェーン実装が提案され開発されているが、ブロックチェーン技術の最も広く知られているアプリケーションは、Ｂｉｔｃｏｉｎ台帳である。Ｂｉｔｃｏｉｎは、ここでは、便宜上及び説明の目的で参照されることがあるが、本開示はＢｉｔｃｏｉｎブロックチェーンと共に使用することに限定されず、代替のブロックチェーン実装及びプロトコルが本開示の範囲に包含されることに留意すべきである。用語「ユーザ」は、ここでは、人間又はプロセッサに基づくリソースを表してよい。

【０００３】

ブロックチェーンは、コンピュータに基づく非集中型の分散型システムとして実装されるピアツーピアの電子台帳であり、ブロックにより構成され、ブロックはまたトランザクションにより構成される。各トランザクションは、ブロックチェーンシステムの中の参加者間でデジタルアセット又はリソース（例えば、暗号通貨又はトークン化されたアイテム）の制御の移転を符号化するデータ構造であり、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。各ブロックは前のブロックのハッシュを含み、これらのブロックは一緒に繋げられて、起源以来ブロックチェーンに書き込まれている全てのトランザクションの永久的な変更不可能な記録を生成する。トランザクションは、スクリプトとして知られている小さなプログラムを含む。スクリプトは、それらのインプット及びアウトプットを埋め込まれ、トランザクションのアウトプットがどのように及び誰によりアクセス可能であるかを指定する。Ｂｉｔｃｏｉｎプラットフォームでは、これらのスクリプトはスタックに基づくスクリプト言語を用いて記述される。

【０００４】

トランザクションがブロックチェーンに書き込まれるためには、検証されなければならない。ネットワークノード（マイナー）は、無効なトランザクションがネットワークから拒否され、各トランザクションが有効であることを保証するために作業を実行する。ノードにインストールされたソフトウェアクライアントは、未使用トランザクション（unspent transaction, UTXO）のロック及びアンロックスクリプトを実行することにより、UTXOに対してこの検証作業を実行する。ロック及びアンロックスクリプトの実行が真（TRUE）と評価する場合、トランザクションは有効であり、トランザクションはブロックチェーンに書き込まれる。したがって、トランザクションがブロックチェーンに書き込まれるためには、（ｉ）トランザクションを受信した第１ノードにより検証され、トランザクションが有効な場合には、ノードが該トランザクションをネットワーク内の他のノードに中継する、（ｉｉ）マイナーにより構築された新しいブロックに追加される、（ｉｉｉ）マイニングされる、つまり過去のトランザクションのパブリック台帳に追加される、ことが必要である。

【０００５】

ブロックチェーン技術は、暗号通貨の実装の使用のために最も広く知られているが、デジタル事業家が、Ｂｉｔｃｏｉｎの基づく暗号セキュリティシステム及び新しいシステムを実装するためにブロックチェーンに格納できるデータの両方の使用を開発し始めている。ブロックチェーンが、暗号通貨の分野に限定されない自動化タスク及びプロセスのために使用できれば、非常に有利になる。このようなソリューションは、ブロックチェーンの利益（例えば、永久性、イベントの記録の耐タンパ性、分散型処理、等）を利用しながら、それらの用途をより多様化し得る。

【０００６】

別の領域のロックチェーンに関連する関心事項は、ブロックチェーンを介して現実世界のエンティティを表現し及び移転するための、「トークン」（又は「カラードコイン（coloured coin）」）の使用である。潜在的に極秘の又は秘密のアイテムは、識別可能な意味又は値を有しないトークンにより表すことができる。したがって、トークンは、現実世界のアイテムがブロックチェーンから参照されることを可能にする識別子として機能する。

【０００７】

ブロックチェーントランザクションは、全部でＮ個のうちのＭ個の署名が、トランザクションを償還する（redeem）するためにＲｅｄｅｅｍスクリプトへのインプットとして提示される必要があるように、トランザクションを制限する組み込み（built－in）マルチシグネチャプロトコルを利用できる。例えば、トランザクションは、３－ｏｆ－５マルチシグネチャ方法を用いてロックされてよい。その結果、トランザクションは、５個のうちの任意の３個の署名に対応する３個の秘密鍵を用いることによってのみ、ロック解除され得る。

【０００８】

Ｍ－ｏｆ－Ｎ方法について、Ｂｉｔｃｏｉｎプロトコルの変形で使用されるコマンドを参照すると、オペコードＯＰ＿ＭＵＬＴＩＳＩＧは、Ｎ個の公開鍵及びＭ個の署名をインプットとして取り入れる。Ｎ個の公開鍵は、本例では、Ｒｅｄｅｅｍスクリプト自体に格納される。オペコードは、最初の署名から開始して、Ｎ個の公開鍵を検索して、署名が該公開鍵により生成されたかどうかを調べる。署名が一致しない全ての鍵をドロップする。この理由から、署名の順序は、公開鍵の提供された順序に一致しなければならない。マルチシグＲｅｄｅｅｍスクリプトの一例は以下に与えられる。

［表１］

【０００９】

【表１】

Redeem script
<PubKey 1><PubKey 2>...<PubKey *N*> OP_CHECKMULTISIG

これは、アンロックするために、以下のインプットの提示が必要である。

［表２］

【００１０】

【表２】

Input
<sig 1><Sig 2>...<Sig *M*>

以上から、Ｒｅｄｅｅｍスクリプトのサイズは、必要な署名の数Ｍ、及び参加者の数Ｎの両方と共に線形にスケーリングすることが明らかである。従って、Ｍ及びＮが増大するにつれ、トランザクションを償還するためにＲｅｄｅｅｍトランザクションにおいて必要な署名の数は、増大し、公開鍵の数も増大する。その結果、伝搬するために必要なネットワーク速度、及び格納するために必要な空間、後のＲｅｄｅｅｍトランザクションが増大する。更に、Ｒｅｄｅｅｍトランザクションのアンロックスクリプトをスクリプトと結合するときに実行される必要のある演算の数は、Ｎ及びＭで線形に増大する。

【００１１】

従って、マルチシグネチャプロトコルを利用するブロックチェーントランザクション、特に膨大な数の公開鍵の記憶を必要とするトランザクションのようなブロックチェーントランザクションの送信、処理、及び記憶に関連付けられた演算及び記憶要件を低減するソリューションを提供することが望ましい。

【００１２】

このような改良されたソリューションがここで考案される。したがって、本開示によると、添付の請求項において定められる方法が提供される。

【発明の概要】

【００１３】

本開示により、コンピュータにより実施される方法が提供される。当該方法は、セキュ

リティ方法として記載され得る。
【００１４】
当該方法は、
公開鍵結合検証関数を含むブロックチェーントランザクションを設けるステップであって、前記ブロックチェーントランザクションは、前記ブロックチェーントランザクションにインプットを提供することにより、リソースへのアクセスを許可し又はリソースの制御を移転するために償還可能なように構成され、前記インプットは、
複数の公開鍵と、
前記複数の公開鍵のうちのそれぞれの２個に関連する少なくとも１つの勾配値と、
前記複数の公開鍵及び前記少なくとも１つの勾配値の結合から導出されるグループ鍵と、を含む、ステップと、
を含んでよく、前記ブロックチェーントランザクションは、前記トランザクションの償還が成功すると、前記グループ鍵が前記複数の公開鍵の前記結合から導出されることを検証するために、前記公開鍵検証関数を前記インプットに適用するよう構成される、方法。
【００１５】
このような方法は、セキュアな信頼できる公に検証可能な方法で、複数の秘密鍵により提示され及び／又は制御されるリソースの制御又はそれへのアクセスを移転する方法を提供する。
【００１６】
当該方法は、既知の値から導出公開鍵を導出するステップと、
前記導出公開鍵を２次公開鍵のシーケンスとして構成するステップと、
を含んでよく、前記２次公開鍵のシーケンスは、前記複数の公開鍵の部分集合である。
【００１７】
これは、ユーザが、方法の既知の要件を、方法で使用される公開鍵に変換することを可能にし、それにより、方法の多様性を向上するという利点を提供する。シーケンスの係数が予め計算され得るという更なる利点が提供される。それにより、方法が実行可能な速度を増大する。
【００１８】
当該方法は、前記導出公開鍵を前記２次公開鍵のシーケンスとして構成するステップの前に、前記既知の値に制約関数を適用するステップであって、それにより前記部分集合のサイズを制限する、ステップ、を含んでよい。
【００１９】
これは、方法の効率を更に向上するという利点を提供する。
【００２０】
当該方法は、前記複数の公開鍵のうちの更なる公開鍵及び前記導出公開鍵に少なくとも部分的に基づき、前記複数のうちの１つの公開鍵を計算するステップ、を含んでよい。
【００２１】
これは、１つの格納された公開鍵から、複数のうちの１つ以上が導出可能であるならば、所与の複数の公開鍵を格納するための要件を除去するという利点を提供する。
【００２２】
前記ブロックチェーントランザクションは、
前記グループ公開鍵に対応するグループ署名と、
マークルルートと、
マークルパス検証関数と、
を更に含んでよく、前記ブロックチェーントランザクションは、前記ブロックチェーントランザクションに、
前記複数の公開鍵に関連付けられたマークルパスと、
前記グループ署名に対応するグループ秘密鍵と、
を提供することにより償還可能なように構成される。
【００２３】

このような方法は、所与のレベルのセキュリティを達成するのに少ない処理ステップしか必要としない。それにより、セキュリティを維持しながら、方法の効率を向上するという利点を提供する。

【００２４】

本開示は、プロセッサと、プロセッサによる実行の結果として、システムに本願明細書に記載のコンピュータにより実施される方法のいずれかの実施形態を実行させる実行可能命令を含むメモリと、を含むシステムも提供する。

【００２５】

本開示は、実行可能命令を記憶した非一時的コンピュータ可読記憶媒体であって、前記実行可能命令は、コンピュータシステムのプロセッサにより実行された結果として、少なくとも、前記コンピュータシステムに、本願明細書に記載のコンピュータにより実施される方法を実行させる、非一時的コンピュータ可読記憶媒体も提供する。

【００２６】

本開示のこれらの及び他の態様は、本願明細書に記載の実施形態から明らかであり及びそれらを参照して教示される。本開示の実施形態は、単なる例を用いて及び添付の図面を参照して以下に説明される。

【図面の簡単な説明】

【００２７】

【図１】非圧縮公開鍵のデータ符号化を示すテーブルである。

【図２】本開示の実施形態を実行するスタックの進展を示すテーブルである。

【図３】本開示の実施形態を実行するスタックの進展を示すテーブルである。

【図４】異なるマルチシグネチャ方法の、Ｍ、Ｎ、及びスクリプトサイズの間の関係を示すテーブルである。

【図５】本開示を具現化する２つのトランザクションを概略的に示す。

【図６】本開示を具現化するステップのシーケンスを示すフローチャートである。

【図７】種々の実施形態が実装できるコンピューティング環境を示す概略図である。

【発明を実施するための形態】

【００２８】

Ｂｉｔｃｏｉｎスクリプトの中の２つの楕円曲線点の加算を検証する方法が開示される。２個の点P_1、P_2、及び候補解P_3が与えられると、開示の方法は、スクリプトの中で、$P_3 = P_1 + P_2$を検証する。一連の点加算へと前述の方法を拡張する更なる方法が開示される。

【００２９】

当該方法は、多数の参加者のための、マルチシグネチャ方法のような署名方法で使用されてよい。当該方法は、楕円曲線デジタル署名アルゴリズム（Elliptic Curve Digital Signature Algorithm (ECDSA)）プロトコルを使用してよい。

【００３０】

素数の楕円曲線は、次式により定義される。

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

曲線のグループ構造に従い、曲線上の２個の点$P_1 = (x_1, y_1)$及び$P_2 = (x_2, y_2)$が加算されて、第３の点$P_3 = (x_3, y_3)$、つまり$P_3 = P_1 + P_2$を得る。

【００３１】

点P_3は、P_1及びP_2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。

【００３２】

【数１】

$x_3 = \lambda^2 - x_1 - x_2 \mod p$

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

ここで、$P_1 \neq P_2$の場合、勾配 λ は次式により与えられる：

$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p$

$P_1 = P_2$の場合、勾配 λ は次式により与えられる：

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

Ｂｉｔｃｏｉｎプロトコルは、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。

【００３３】

ブロックチェーントランザクションのスクリプトの中で拡張ユークリッドの互除法（extended Euclidean algorithm）を適用することは実用的ではない。これは、モジュラ逆数の計算が必要であり、従って、上述の式で定義された楕円曲線点加算をスクリプト内に実装できないからである。

【００３４】

以下では、２個の点Ｐ$_1$及びＰ$_2$の加算の解を検証する方法が開示される。解Ｐ$_3$＝Ｐ$_1$＋Ｐ$_2$自体の計算は、スクリプトの外部で実行されるが、答えはスクリプト内で検証される。これは、ブロックチェーンノードに課され得る計算上の要件を低減する。

【００３５】

この検証を達成するために、解Ｐ$_3$、及びλにより示されるＰ$_1$とＰ$_2$との間の直線の勾配の値、が必要である。λの計算は、スクリプトの外部で実行されるが、λの検証はスクリプト内で実行できる。

【００３６】

新たな公開鍵検証関数＜Ｐｏｉｎｔ　Ａｄｄ　Ｐ$_1$，Ｐ$_2$，λ，Ｐ$_3$＞は、以下に詳細に定義され、それに提示されるλがＰ$_1$とＰ$_2$との間の直線の勾配である場合、及びＰ$_3$＝Ｐ$_1$＋Ｐ$_2$である場合、真（ＴＵＲＥ）を返すよう構成される。これら２つの検証は、スクリプト内で直接達成され、トランザクションの償還が成功すると、発行される。

【００３７】

関数の実行は以下のステップを含む。

【００３８】

【数２】

1. λ の値は以下を調べることにより検証される:

   a) $P_1 \neq P_2$の場合、

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

   b) $P_1 = P_2$ の場合、

   $2\lambda y_1 \mod p = 3x_1^2 \mod p.$

2. P_3の値は、座標$P_3 = (x_3, y_3)$が以下を満たすことを調べることにより検証される

   $x_3 = \lambda^2 - x_1 - x_2 \mod p$ 及び

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

公開鍵検証関数の連続的反復を適用することにより、任意の数ｎの公開鍵の和が検証さ

れ得る。ここで、唯一の制約は、ブロックチェーンプロトコルが使用されるという要件、例えばＢｉｔｃｏｉｎスクリプトでは、オペコードの数（２０１）又はサイズ（単位：バイト）（１００００）である。

【００３９】

上述の公開鍵検証関数は、公開鍵のうちの２個が結合して第３の公開鍵を生じることを検証するために、３個の公開鍵で動作する。これは３個の公開鍵の加算Ｐ＝Ｐ$_1$＋Ｐ$_2$＋Ｐ$_3$、及びそれより多くを検証するために格納できる。拡張された関数は、ＰｏｉｎｔＡｄｄＭｕｌｔｉと示され、以下に定義される。ここで、検証手順は、Ｐ$_1$、Ｐ$_2$、及びＰ$_3$が結合してＰを生じることを検証する。

【００４０】

【数３】

$$<\text{Point Add Multi } P_1, P_2, P_3, \lambda', \lambda, P> := <\text{Point Add } P_1, P_2, \lambda', P'> <\text{Point Add } P', P_3, \lambda, P>$$

ここで、$P' = P_1 + P_2$，　値λ' はP_1とP_2との間の勾配であり、値λ はP'とP_3との間の勾配である。

一般的には、つまり、結合して公開鍵Ｐを生じるｎ個の公開鍵Ｐ$_i$が与えられると、ＰｏｉｎｔＡｄｄＭｕｌｔｉは以下のように定義できる。

【００４１】

【数４】

$$<\text{PointAddMulti } P_i, \lambda_i, P> :=$$

$$<\text{PointAdd } P_1, P_2, \lambda_1, P_1'><\text{PointAdd} P_1', P_3, \lambda_2, P_2' > \ldots <\text{PointAdd } P_{n-2}', P_n, \lambda_{n-1}, P>$$

点加算関数ＰｏｉｎｔＡｄｄは、以下に開示する方法を用いて、Ｂｉｔｃｏｉｎスクリプト言語で構成できる。

【００４２】

図１に、非圧縮公開鍵のデータ符号化が示される。ここで、データのダミー値が、よく知られた書籍であるA. Antonopoulos著、Mastering Bitcoin、2nd Edition, O’Reilly Media (２０１７)から取り入れられる。

【００４３】

非圧縮公開鍵Ｐが与えられると、ｘ及びｙ座標は、演算子ＯＰ＿ＳＰＬＩＴを用いてＢｉｔｃｏｉｎスクリプトから以下のように直接抽出できる。

【００４４】

【数５】

$$<P> \text{ OP\_1 OP\_SPLIT OP\_NIP 32 OP\_SPLIT } = <x> <y>$$

この演算を用いて、ｘ及びｙ座標はインプットＰ$_3$、λ、Ｐ$_1$、Ｐ$_2$から抽出できる。

【００４５】

【数６】

$$<P_3> <\lambda> <P_1> <P_2> \longrightarrow <x_3> <y_3> <\lambda> <x_1> <y_1> <x_2> <y_2>$$

留意すべき事に、上式の右辺にある項は、複製され、再配置されて、基本演算を用いて任意の所望の結合を生成し得る。

【００４６】

図２を参照すると、関数ＰｏｉｎｔＡｄｄの構成における第１のステップは、λのインプット値を検証することである。Ｐ$_1$≠Ｐ$_2$の場合、以下の式が真のままであるかが調べ

られる。
【００４７】
【数７】

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

これは、＜ｙ２＞＜ｙ１＞＜λ＞＜ｘ２＞＜ｘ１＞をインプットとして取り入れ、このインプットに対して以下の演算を行うことにより達成される。
【００４８】
【数８】

OP_SUB OP_MUL <p> OP_MOD OP_3 OP_ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL

これは、次式が満たされる場合、及びその場合のみ、真を返す。
【００４９】
【数９】

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

図２のテーブルは、上述の演算が実行されるとき、スタックの進展を示す。
【００５０】
Ｐ１＝Ｐ２の場合、以下の演算が真であることを調べる、上述の式と同様の演算セットが構成できる。
【００５１】
【数１０】

$$2\lambda y_1 \mod p = 3x_1^2 \mod p$$

図３を参照すると、関数ＰｏｉｎｔＡｄｄの構成における第２のステップは、Ｐ３のインプット値がＰ１＋Ｐ２の和であることを検証することである。これを行うために、次式の各々が真であることが調べられる。
【００５２】
【数１１】

$$x_3 = \lambda^2 - x_1 - x_2 \mod p, \text{ and}$$

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

λは既に検証済みなので、以下の計算で使用できる。上述の２つの式のうちの最初の式が成り立つことを調べるために、以下がインプットとして提示され：＜ｘ３＞＜λ＞＜ｘ１＞＜ｘ２＞
以下の演算が行われる：
【００５３】
【数１２】

OP_3 OP_ROLL OP_DUP OP_MUL OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL.

ここで、ｐは楕円曲線のベースフィールド（base field）の次数である。

【００５４】
図３のテーブルは、上述の演算が実行されるとき、スタックの進展を示す。
【００５５】
新しい公開鍵Ｐ３は、既存の公開鍵Ｐ１を、楕円曲線生成元（generator point）Ｇを数値Ｓで乗算して計算される公開鍵に加算することにより、生成できる。つまり、Ｐ３＝Ｐ１＋Ｓ・Ｇ、
【００５６】
【数１３】

$$S \in \mathbb{Z}_n^*$$

Ｓは、元の公開鍵Ｐ１から導出される公開鍵Ｐ３を生成するために使用される決定鍵であってよい。
【００５７】
Ｐ３＝Ｐ１＋Ｓ・Ｇの点加算の検証は、生成元Ｇの定数倍（fixed multiples）を含む一連の点加算を検証することにより、達成できる。
【００５８】
一例として、点加算を以下のシリーズに分解するために、倍及び加算（double－and－add）方法が使用されてよい。
【００５９】
【数１４】

$$P_3 = P_1 + s_0G + s_12G + s_24G + s_38G + \cdots + s_{256}2^{256}G$$

ここで、ｓ０、．．．．、ｓ２５６∈｛０，１｝は、次式のようにＳのバイナリ表現の係数である。
【００６０】
【数１５】

$$S = s_0 + s_12 + s_24 + s_38 + \cdots + s_{256}2^{256}$$

生成元を複数倍することにより得られる点Ｇ、２Ｇ、４Ｇ、．．．、2^{256}Ｇは、一般に知られており、予め計算できる。これは、最終結果を得るために検証される必要のある２５５個の可能な個々の点加算が存在することを意味する。
【００６１】
従って、Ｐ３＝Ｐ１＋Ｓ・Ｇは、最大で２５５＋１＝２５６個の点加算により検証されてよい。
【００６２】
倍及び加算方法の代替として、スライドウインドウ又はモンゴメリラダー法のような他の点加算方法アルゴリズムが使用されてよい。
【００６３】
計算された鍵による点加算の計算を更に管理し易くするために、Ｓが更に狭い範囲に制限されてよい。範囲は、アプリケーション固有であってよい。
【００６４】
Ｓは、４３億個の値の範囲を表す以下：
【００６５】
【数１６】

$$\mathbb{Z}_n^*$$

の範囲内の３２ビット範囲の中の値を取る必要があってよい。しかしながら、最終結果を

検証するためには、３１個の個々の点加算だけでよい。これは、一例としてＢｉｔｃｏｉｎでは、完全な２５６ビット範囲の中のＳで開始し、次に該範囲を３２ビットに制限するためにモジュラ演算ＯＰ＿ＭＯＤを適用することにより、達成されてよい。和Ｐｘ＋Ｓ・Ｇを取ることにより、範囲を難読化するために、ランダムな公開鍵P_xが使用されてよい。

【００６６】

一例として、中古車オークションの最高入札額が１６０００米ドルである。数値１６０００の形式のｔ米ドルの入札は、公開鍵P_1に加算されて、加法鍵（additive key）Ｐ＝P_1＋ｇ・Ｇを得ることができる。

【００６７】

これは、１４個の点加算により、スクリプト内で検証されてよい。

【００６８】

これが入札処理でどのように使用され得るかを見るために、車に対する現在の最高入札が９０００米ドルであると仮定する。これは、特定のＵＴＸＯにより追跡されている。このＵＸＴＯは、以下のＲｅｄｅｅｍスクリプトを有する。

【００６９】

【数１７】

```
IF  t > 9000
    <Point Add Multi  P1, t · G, P>
    <CheckSig  P>
ELSE
    OP_RETURN
```

これは、アンロックのためにインプットとして以下を要求する。

【００７０】

【数１８】

```
<Sig  P> <P> <Verification data> <P1> <t>
```

上述の点加算検証方法は、次に、Ｐ＝P_1＋ｔ・Ｇを検証するために使用できる。これは、１４個の個々の点加算の検証を含むことに留意する。これらの検証のためのデータは、上述のインプットの中で「検証データ（verification data）」としてラベル付けされる。

【００７１】

ＵＴＸＯは、ｔ＞９０００の新しい入札が提供された場合に、アンロックできる。新しい入札者は、彼らの公開鍵P_1、及び和Ｐ＝P_1＋ｔ・Ｇに対する解Ｐを提供しなければならない。彼らが加法鍵Ｐによりトランザクションに署名したという事実は、彼らをｔ米ドルの特定の入札に常にリンクする。

【００７２】

別の例として、現在ブロックのようなブロックチェーン内のブロックによりハッシュされると、特定の閾値より下の値を生成するノンス（数値）を探すＢｉｔｃｏｉｎマイナーを考える。マイナーは、トランザクションの公開鍵及びノンスの最初の９桁でロックされた該トランザクションを生成してよい。つまり、Ｐ＝P_1＋（ノンスの最初の９桁）・Ｇである。数値の最初の９桁について１０億個もの組合せが存在する。従って、公開鍵Ｐは、３０個の点加算を用いて検証できる。これは、鍵Ｐを、マイナーの公開鍵、及びマイニングされたブロックのノンスに常にリンクする。

【００７３】
上述のスクリプト内の点加算検証手順は、Ｍ－ｏｆ－Ｎマルチシグネチャ方法を用いてトランザクションに署名する効率的な方法を実施するために使用されてよい。それにより、ブロックチェーンを用いてリソースの制御又はリソースへのアクセスを移転するより効率的且つセキュアな方法を可能にする。
【００７４】
Ｍ－ｏｆ－Ｎ方法では、ＢｉｔｃｏｉｎオペコードＯＰ＿ＭＵＬＴＩＳＩＧは、インプットとしてＮ個の公開鍵及びＭ個の署名を取り入れる。オペコードは、最初の署名から開始して、Ｎ個の公開鍵を検索して、署名が該公開鍵により生成されたかどうかを調べる。署名が一致しない全ての鍵をドロップする。この理由から、署名の順序は、公開鍵の提供された順序に一致しなければならない。
【００７５】
マルチシグＲｅｄｅｅｍスクリプトは以下の通りである。
【００７６】
【数１９】

<PubKey 1><PubKey 2>...<PubKey *N*> OP_CHECKMULTISIG

これは、アンロックのためにインプットとして以下を要求する：
＜ｓｉｇ１＞＜Ｓｉｇ２＞．．．＜ＳｉｇＭ＞。
【００７７】
以上から、スクリプトのサイズは、必要な署名の数Ｍ、及び参加者の数Ｎの両方と共に線形にスケーリングすることが分かる。
【００７８】
図４～６を参照すると、公開鍵の各々のＭ－ｏｆ－Ｎ組合せに対応するリーフを有するマークル木が生成され、これは全部でＮ個のリーフからＭ個の選択することを含む。次に、マークルルートを含むＲｅｄｅｅｍスクリプトが生成される。該Ｒｅｄｅｅｍスクリプトは、マークルパス及び署名検証方法を含むインプットのアンロック表現を必要とする。
【００７９】
リソースの制御又はリソースへのアクセスを移転する２つの方法が後述される。各々は、それら自体の利点及び欠点を有する。
１．個別署名方法これは、トランザクションに個々に署名するためにそれぞれの署名を使用し、マルチシグと同じ機能を有するが、スクリプトサイズがより小さい。
２．グループ署名方法これは、署名グループの中のＭ任意の参加者の単一の署名を提供する。これは、共有シークレットから導出される鍵、又はグループの公開鍵を加算することにより形成される鍵のような、参加者のうちのＭ人の各集合について予め合意された任意の公開鍵であってよい。後者の場合、鍵が個々のメンバの鍵の和であることを検証するための追加オプションは、前述の検証方法を使用する。
【００８０】
各方法は、図４に示されたテーブルに纏めたように、異なるスケーリング特性を有する。
【００８１】
４－ｏｆ－５方法を考える。この場合、マークル木は、以下のリーフを有する。
【００８２】
【数２０】

5 choose 4(つまり５－元集合から４個の元を選ぶ方法)
$= \binom{5}{4} = \frac{5!}{4!(5-4)!} = 5$ リーフ

マークル木自体は、３レベルの深さである。ルート＜Ｒ＞を有するマークルパスを検証する演算は、以下の通りである。

【００８３】
【数２１】

<Verify Merkle Path> =
6 OP_PICK OP_SHA256 (OP_SWAP OP_IF OP_SWAP OP_ENDIF OP_CAT OP_SHA256)*3 <R> OP_EQUALVERIFY
これは、以下の形式のマークルパスについて真を返す
<Leaf> <Grandparent Sib> <0,1> <Parent Sib> <0,1> <Sib> <0,1>
ここで、兄弟ノード（sibling）が左ノードである場合、０であり、兄弟ノードが右ノードである場合、１である。

マークルパス及びマークルパスを検証するための演算は、ＮからＭを選ぶ（Ｎ choose Ｍ）署名グループの選択数と共に、対数的にサイズが増大する。
【００８４】
個別署名方法では、グループの各メンバは、トランザクションの個々の署名を提供する。この方法の機能は、標準的なマルチシグ方法と同じであるが、スクリプトのサイズは縮小し、スクリプトが実行される効率を向上する。スクリプトは、以下の形式を取る:
【００８５】
【数２２】

<Verify Merkle Path> <CheckSig>*M

これは、アンロックするために、以下のインプットの提示が必要である。
【００８６】
【数２３】

<Sig P_1>…<Sig P_M> <P_1>…<P_M> <Merkle Path $P_1, \ldots, P_M$>

償還者（redeemer）は、インプットとして、Ｍ個の署名をそれらの対応する公開鍵と共に、及びＰのマークルパスを提供しなければならない。
【００８７】
この方法でのＰ１、．．．、ＰＭのマークル木リーフへのマッピングは、例えば、ＯＰ＿ＣＡＴを用いるＰ１、．．．、ＰＭの連結のハッシュであってよい。
【００８８】
この方法の主要な特徴は、各メンバが特定のトランザクションに署名することである。彼らは、彼らの秘密鍵に関する情報を互いに開示しないので、かれらの秘密鍵のセキュリティを損なわない。
【００８９】
図４を参照すると、この方法は、従来のマルチシグ方法に勝る有利なスケーリング特性を有することが分かる。これは、マルチシグネチャ制度が小さなＭ及び大きなＮ、例えば５－ｏｆ－１０００、にあるとき最も顕著である。
【００９０】
グループ署名方法では、トランザクションに署名するＭ人のメンバのグループについて、１つの署名しか必要ない。スクリプトは、以下の形式を取る。
【００９１】
【数２４】

<Verify Merkle Path> <CheckSig>

これは、アンロックするために、以下のインプットの提示が必要である。

【００９２】

【数２５】

<Sig P><P> <Merkle Path>

ここで、Ｐは、Ｍ人の参加者の集合についてのグループ公開鍵である。

【００９３】

この方法の利点は、１つの署名しか必要ないので、アンロックスクリプトのサイズが非常に小さいことである。スクリプト内の公開鍵の加算の数は、参加者の数Ｍと共に線形にスケーリングする。従って、この方法は、個々の署名を調べるより、処理パワーの観点でコストの少ない演算を提供する。

【００９４】

この方法は、参加者が互いに彼らの秘密鍵を共有することを要求する。これは、この方法で彼らの使用する鍵ペアが、１回限りの使用の鍵ペアであることが、本方法のセキュリティを向上するために有用であることを意味する。別の特手用は、グループ秘密鍵へのアクセスを有する任意の数の参加者が、彼らの望む任意のトランザクションに署名し得ることである。これは、方法の多様性を向上する。これらの特徴が特に望ましい用途がある。例えば、ＵＴＸＯとしてブロックチェーンに記録された、良好に確立された最後の手段の条項（last－resort clause）を有効にしたいと望む取締役のグループである。これは、本方法を用いて特定のＵＴＸＯをアンロックすることにより達成されてよい。

【００９５】

上述の方法では、外部観察者は全く認識されない。つまり、実際には、グループ署名方法である。署名及び対応する公開鍵がユーザのグループに関連するという事実を公にするために、本方法は、スクリプト内で、グループ公開鍵が個々の参加者の公開鍵の和であることＰ＝Ｐ$_1$＋・・・＋Ｐ$_M$が検証されることを要求するよう変更されてよい。この適応された方法は、上述の鍵加法検証方法を使用してよい。

【００９６】

【数２６】

<Verify Merkle Path> <Point Add Multi $P_1, \ldots, P_M, P$> <CheckSig>

これは、アンロックするために、以下のインプットの提示が必要である。

【００９７】

【数２７】

<Sig P><P> <Verification data><P><P_1>…<P_M> <Merkle Path>

ここで、＜Ｐｏｉｎｔ Ａｄｄ Ｍｕｌｔｉ Ｐ$_1$，．．．，Ｐ$_M$，Ｐ＞関数は、前に導入されたものであり、＜Ｖｅｒｉｆｉｃａｔｉｏｎ ｄａｔａ＞は、各々の個々の点加算の間の勾配を含む。

【００９８】

図５は、トランザクションＴｘ１及びＴｘ２のペアを示す。Ｔｘ１は、Ｖｅｒｉｆｙ Ｍａｒｋｌｅ Ｐａｔｈ関数、特定のＭ－ｏｆ－Ｎ要件に調整されたＰｏｉｎｔＡｄｄＭｕｌｔｉ関数、及びＣｈｅｃｋＳｉｇ関数を含むＲｅｄｅｅｍスクリプトを有するＵＴＸＯを定義する。Ｔｘ２は、Ｔｘ１へのインプットの中でグループ署名、グループ署名に関連するグループ公開鍵、ＰｏｉｎｔＡｄｄＭｕｌｔｉ関数により要求される勾配、ユーザのグループの公開鍵、及び関連付けられたＭａｒｋｌｅ Ｐａｔｈを提示することにより、Ｔｘ１のＵＴＸＯを償還することを目的とする、ブロックチェーンに後に提出されるトランザクションを定義する。

【００９９】
図６は、上述の個別署名方法及びグループ署名方法に従う方法を実行するために行われるステップを示すフローチャートを示す。図６で、Ｔｘ１及びＴｘ２は、いずれかの方法に従い使用されるトランザクションのペアを表し得る。
【０１００】
図７を参照すると、本開示の少なくとも一実施形態を実施するために使用され得るコンピューティング装置２６００の説明のための簡略ブロック図が提供される。種々の実施形態で、コンピューティング装置２６００は、上述の図示のシステムのうちのいずれかを実装するために使用されてよい。例えば、コンピューティング装置２６００は、データサーバ、ウェブサーバ、ポータブルコンピューティング装置、パーソナルコンピュータ、又は任意の電子コンピューティング装置として使用するために構成されてよい。図７に示すように、コンピューティング装置２６００は、主メモリ２６０８及び永久記憶装置２６１０を含む記憶サブシステム２６０６と通信するよう構成され得る１つ以上のレベルのキャッシュメモリ及びメモリ制御部（集合的に２６０２とラベル付けされる）を備える１つ以上のプロセッサを含んでよい。主メモリ２６０８は、図示のように、動的ランダムアクセスメモリ（ＤＲＡＭ）２６１８及び読み出し専用メモリ（ＲＯＭ）２６２０を含み得る。記憶サブシステム２６０６及びキャッシュメモリ２６０２は、本開示で説明されたようなトランザクション及びブロックに関連付けられた詳細事項のような情報の記憶のために使用されてよい。プロセッサ２６０２は、本開示で説明されたような任意の実施形態のステップ又は機能を提供するために利用されてよい。
【０１０１】
プロセッサ２６０２は、１つ以上のユーザインタフェース入力装置２６１２、１つ以上のユーザインタフェース出力装置２６１４、及びネットワークインタフェースサブシステム２６１６とも通信できる。
【０１０２】
バスサブシステム２６０４は、コンピューティング装置２６００の種々のコンポーネント及びサブシステムが意図した通りに互いに通信できるようにするメカニズムを提供してよい。バスサブシステム２６０４は、単一のバスとして概略的に示されるが、バスサブシステムの代替の実施形態は、複数のバスを利用してよい。
【０１０３】
ネットワークインタフェースサブシステム２６１６は、他のコンピューティング装置及びネットワークへのインタフェースを提供してよい。ネットワークインタフェースサブシステム２６１６は、幾つかの実施形態では、コンピューティング装置２６００の他のシステムからデータを受信し及びそれへデータを送信するインタフェースとして機能してよい。例えば、ネットワークインタフェースサブシステム２６１６は、データ技術者が、装置をネットワークに接続することを可能にする。その結果、データ技術者は、データセンタのような遠隔地にいがなら、データを装置へ送信し、データを装置から受信できる。
【０１０４】
ユーザインタフェース入力装置２６１２は、キーボード、統合型マウス、トラックボール、タッチパッド、又はグラフィックタブレットのような指示装置、スキャナ、バーコードスキャナ、ディスプレイに組み込まれたタッチスクリーン、音声認識システム、マイクロフォンのようなオーディオ入力装置、及び他の種類の入力装置のような、１つ以上のユーザ入力装置を含んでよい。通常、用語「入力装置」の使用は、コンピューティング装置２６００に情報を入力する全ての可能な種類の装置及びメカニズムを含むことを意図する。
【０１０５】
１つ以上のユーザインタフェース出力装置２６１４は、ディスプレイサブシステム、プリンタ、又は音声出力装置のような非視覚ディスプレイ、等を含んでよい。ディスプレイサブシステムは、陰極線管（CRT）、液晶ディスプレイ（LCD）、発光ダイオード（LED）ディスプレイ、又はプロジェクションのような平面装置、又は他のディスプレイ装置を含んでよい。通常、用語「出力装置」の使用は、コンピューティング装置２６００から情

報を出力する全ての可能な種類の装置及びメカニズムを含むことを意図する。１つ以上のユーザインタフェース出力装置２６１４は、例えば、ユーザインタフェースを提示して、ここに記載したプロセス及び変形を実行するアプリケーションとのユーザ相互作用が適切であるとき、そのような相互作用を実現するために使用されてよい。

【０１０６】

記憶サブシステム２６０６は、本開示の少なくとも１つの実施形態の機能を提供する基本プログラミング及びデータ構造を記憶するコンピュータ可読記憶媒体を提供してよい。アプリケーション（例えば、プログラム、コードモジュール、命令）は、１つ以上のプロセッサにより実行されると、本開示の１つ以上の実施形態の機能を提供し、記憶サブシステム２６０６に格納されてよい。これらのアプリケーションモジュール又は命令は、１つ以上のプロセッサ２６０２により実行されてよい。記憶サブシステム２６０６は、更に、本開示に従い使用されるデータを格納するレポジトリを提供する。例えば、主メモリ２６０８及びキャッシュメモリ２６０２は、プログラム及びデータのための揮発性記憶を提供できる。永久記憶装置２６１０は、プログラム及びデータの永久（不揮発性）記憶を提供でき、磁気ハードディスクドライブ、取り外し可能媒体に関連付けられた１つ以上のフロッピディスクドライブ、取り外し可能媒体に関連付けられた１つ以上の光ドライブ（例えば、ＣＤ－ＲＯＭ、又はＤＶＤ、又はＢｌｕｅ－Ｒａｙ）ドライブ、及び他の同様の記憶媒体を含んでよい。このようなプログラム及びデータは、本開示に記載した１つ以上の実施形態のステップを実行するためのプログラム、及び本開示に記載したトランザクション及びブロックに関連付けられたデータを含み得る。

【０１０７】

コンピューティング装置２６００は、ポータブルコンピュータ装置、タブレットコンピュータ、ワークステーション、又は後述する任意の他の装置を含む種々のタイプのものであってよい。さらに、コンピューティング装置２６００は、１つ以上のポート（例えば、USB、ヘッドフォンジャック、光コネクタ、等）を通じてコンピューティング装置２６００に接続可能な別の装置を含み得る。コンピューティング装置２６００に接続され得る装置は、光ファイバコネクタを受けるよう構成される複数のポートを含んでよい。したがって、この装置は、光信号を、処理のために装置を接続するポートを通じてコンピューティング装置２６００に送信される電気信号に変換するよう構成されてよい。コンピュータ及びネットワークの絶えず変化する特性により、図７に示したコンピューティング装置２６００の説明は、装置の好適な実施形態を説明する目的の特定の例としてのみ意図される。図７に示したシステムより多くの又は少ないコンポーネントを有する多くの他の構成が可能である。

【０１０８】

上述の実施形態は、本発明を限定するのではなく、説明すること、及び当業者は添付の特許請求の範囲により定められる本開示の範囲から逸脱することなく多くの代替的実施形態を考案できることに留意すべきである。特許請求の範囲において、括弧内の任意の参照符号は、請求項を限定することを意図しない。用語「有する」及び「含む」（comprising、comprises）等は、任意の請求項又は明細書全体に列挙されたもの以外の要素又はステップの存在を排除しない。本願明細書では、「有する」は「有する又は構成される」を意味し、「含む」は「含む又は構成される」を意味する。要素の単数の参照は、該要素の複数の参照を排除しない。逆も同様である。本開示は、幾つかの別個の要素を含むハードウェアにより、及び適切にプログラムされたコンピュータにより、実装できる。幾つかの手段を列挙する装置クレームでは、これらの手段のうちの幾つかは、１つの同じハードウェアアイテムにより具現化されてよい。単に特定の手段が相互に異なる従属請求項に記載されるという事実は、これらの手段の組み合わせが有利に使用されないことを示さない。

【図面】

【図１】

Table 1: Public Key Data Encoding

Data Structure	Length in bytes	Data
Uncompressed key flag	1	04
X	32	F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
Y	32	07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB

【図２】

Table 2: Check λ when $P_1 \neq P_3$

Stack	Script	Description
	<y_2> <y_1> <λ> <x_2> <x_1> OP_SUB OP_MUL <p> OP_MOD OP_3 OP_ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL	Constants added to the stack
<y_2> <y_1> <λ> <x_2> <x_1>	OP_SUB OP_MUL <p> OP_MOD OP_3 OP_ ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL	Subtract
<y_2> <y_1> <λ> <x_2 - x_1>	OP_MUL <p> OP_MOD OP_3 OP_ ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL	Multiply
<y_2> <y_1> <$\lambda(x_2 - x_1)$>	<p> OP_MOD OP_3 OP_ ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL	Constant added to the stack
<y_2> <y_1> <$\lambda(x_2 - x_1)$> <p>	OP_MOD OP_3 OP_ ROLL OP_3 OP_SUB <p> OP_MOD OP_EQUAL	Mod
<y_2> <y_1> <$\lambda(x_2 - x_1) \bmod p$>	OP_3 OP_ ROLL OP_3 OP_PICK OP_SUB <p> OP_MOD OP_EQUAL	Roll 3rd item of stack
<y_1> <$\lambda(x_2 - x_1) \bmod p$> <$y_2$>	OP_3 OP_ ROLL OP_SUB <p> OP_MOD OP_EQUAL	Roll 3rd item of stack
<$\lambda(x_2 - x_1) \bmod p$> <$y_2$> <$y_1$>	OP_SUB <p> OP_MOD OP_EQUAL	Subtract
<$\lambda(x_2 - x_1) \bmod p$> <$y_2 - y_1$>	<p> OP_MOD OP_EQUAL	Constant added to the stack
<$\lambda(x_2 - x_1) \bmod p$> <$y_2 - y_1$> <$p$>	OP_MOD OP_EQUAL	Mod
<$\lambda(x_2 - x_1) \bmod p$> <$y_2 - y_1 \bmod p$>	OP_EQUAL	Check equal
TRUE		

【図３】

Table 3: Check x coordinate of P_3

Stack	Script	Description
	<x_3> <λ> <x_1> <x_2> OP_3 OP_ ROLL OP_DUP OP_MUL OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	Constants pushed to stack
<x_3> <λ> <x_1> <x_2>	OP_3 OP_ ROLL OP_DUP OP_MUL OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	
<x_3> <x_1> <x_2> <λ>	OP_DUP OP_MUL OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	Roll 3rd item
<x_3> <x_1> <x_2> <λ> <λ>	OP_MUL OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	Duplicate
<x_3> <x_1> <λ^2> <x_2>	OP_SWAP OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	Multiply
<x_3> <x_1> <$\lambda^2 - x_2$>	OP_SUB OP_SWAP <p> OP_MOD OP_EQUAL	Swap
<x_3> <$\lambda^2 - x_2$> <x_1>	OP_SWAP <p> OP_MOD OP_EQUAL	Subtract
<x_3> <$\lambda^2 - x_2 - x_1$>	<p> OP_MOD OP_EQUAL	Constant pushed to the stack
<x_3> <$\lambda^2 - x_2 - x_1$> <p>	OP_MOD OP_EQUAL	Mod
<x_3> <$\lambda^2 - x_2 - x_1 \bmod p$>	OP_EQUAL	Swap
TRUE		Check equal

【図４】

スケーリング特性	標準的なマルチシグ	個別署名方法	グループ署名方法	鍵加算を伴うグループ署名方法
署名	線形 M	線形 M	0	0
公開鍵	線形 N	線形 M	0	線形 M
マークルパス	N/A	Log $\binom{N}{M}$	Log $\binom{N}{M}$	Log $\binom{N}{M}$

【図５】

Tx1

Redeem script:

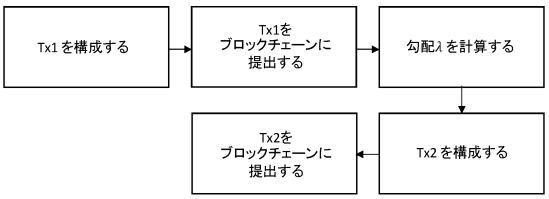
```
<Verify Merkle path>
<PointAddMulti P1,P2,…,PM,P>
<CheckSig>
```

Tx2

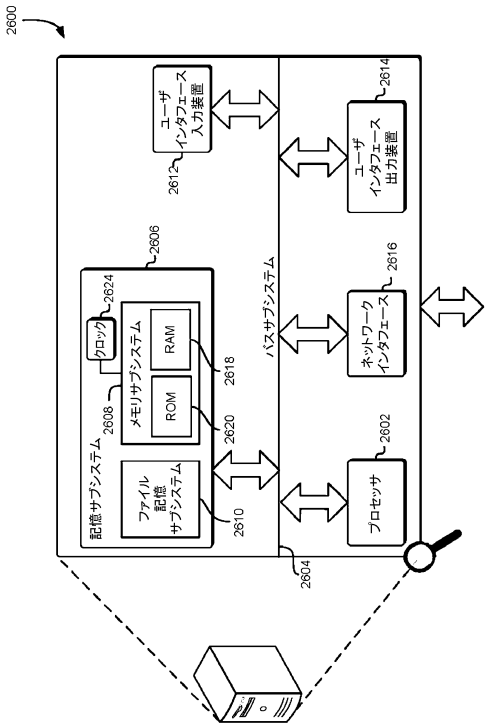
Input (unlocking script):

```
<Sig P><P>
<Gradients><P><P1><P2>…<PM>
<Merkle path>
```

【図６】



【図７】


2600
ユーザ
インタフェース
入力装置
2612
2614
ユーザ
インタフェース
出力装置
2606
2624
クロック
メモリサブシステム
RAM
ROM
2618
2620
2608
記憶サブシステム
ファイル
記憶
サブシステム
2610
バスサブシステム
2616
ネットワーク
インタフェース
2602
プロセッサ
2604

フロントページの続き

内
(72)発明者 ヴォーン，オーウェン
イギリス国 シーエフ１０ ２エイチエイチ カーディフ チャーチル ウェイ チャーチル ハウス ７ス フロア アーカート－ダイクス アンド ロード エルエルピー 内
審査官 中里 裕正
(56)参考文献 国際公開第２０１８／１８５７２４（WO，Ａ１）
MAXWELL, G. et al.，Simple Schnorr Multi-Signatures with Applications to Bitcoin，Cryptology ePrint Archive，Report 2018/068 ver:20180520:191909，[online]，2018年05月20日，pp.1-34，〈URL:https://eprint.iacr.org/2018/068/20180520:191909〉
ZHAO, Y.，Aggregation of Gamma-Signatures and Applications to Bitcoin，Cryptology ePrint Archive，Report 2018/414 ver:20181205:162615，[online]，2018年12月05日，pp.1-22，〈URL:https:// eprint.iacr.org/2018/414/20181205:162615〉
(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｈ０４Ｌ ９／３２
ＪＳＴＰｌｕｓ／ＪＭＥＤＰｌｕｓ／ＪＳＴ７５８０（ＪＤｒｅａｍＩＩＩ）
ＩＥＥＥ Ｘｐｌｏｒｅ