

(19)日本国特許庁(JP)

(12)公表特許公報(A)

(11)公表番号
特表2024-536271
(P2024-536271A)

(43)公表日 令和6年10月4日(2024.10.4)

(51)国際特許分類

F I

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

G 0 6 F 21/64 (2013.01) G 0 6 F 21/64

審査請求 未請求 予備審査請求 未請求 (全53頁)

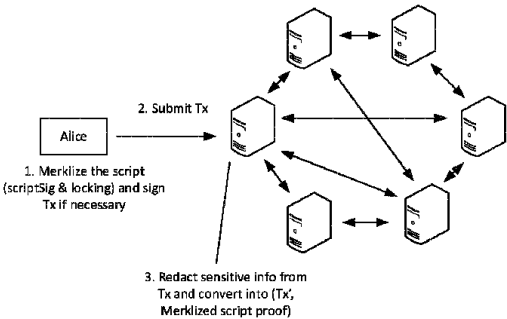
(21)出願番号	特願2024-519817(P2024-519817)	(71)出願人	318001991
(86)(22)出願日	令和4年9月6日(2022.9.6)		エヌチェーン ライセンシング アーゲー
(85)翻訳文提出日	令和6年4月1日(2024.4.1)		スイス・6 3 0 0・ツーク・グラーフエ
(86)国際出願番号	PCT/EP2022/074687		ナウヴェーク・6
(87)国際公開番号	WO2023/057149	(74)代理人	100107766
(87)国際公開日	令和5年4月13日(2023.4.13)		弁理士 伊東 忠重
(31)優先権主張番号	2114286.4	(74)代理人	100229448
(32)優先日	令和3年10月6日(2021.10.6)		弁理士 中横 利明
(33)優先権主張国・地域又は機関	英国(GB)	(72)発明者	キラズ, メーメット, サビール
(81)指定国・地域	AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA, RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR, GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC, 最終頁に続く	(72)発明者	イギリス ダブリュー1ダブリュー 8エービー ロンドン マーケット プレイス 3 0 エヌチェーン ライセンシング アーゲー 内
			デイヴィーズ, ジャック, オーウェン
			イギリス ダブリュー1ダブリュー 8エー 最終頁に続く

(54)【発明の名称】 ブロックチェーントランザクションからのコンテンツの編集

(57)【要約】

ブロックチェーントランザクションのデータを編集するコンピュータ実装方法であって、第1の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、第1のブロックチェーントランザクションは、編集対象のそれぞれのターゲットデータを含む1つまたは複数のそれぞれのスクリプトを含む、ステップと、1つまたは複数のそれぞれのスクリプトの少なくとも1つについて、それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの1つまたは複数にわたって分割される、ステップと、少なくとも1つのそれぞれのスクリプトをそれぞれのマークルツリーのそれぞれのマークルルートに置き換えることによって、第1のブロックチェーントランザクションの編集済みバージョンを生成するステップとを含む方法。

Figure 6



【特許請求の範囲】**【請求項 1】**

ブロックチェーンランザクションのデータを編集するコンピュータ実装方法であって、

第 1 のブロックチェーンランザクションを取得するステップであって、前記第 1 のブロックチェーンランザクションは、編集対象のそれぞれのターゲットデータを含む 1 つまたは複数のそれぞれのスクリプトを含む、ステップと、

前記 1 つまたは複数のそれぞれのスクリプトの少なくとも 1 つについて、前記それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、前記それぞれのターゲットデータは、前記それぞれのマークルツリーの前記それぞれのリーフのうちの 1 つまたは複数にわたって分割される、ステップと、

前記少なくとも 1 つのそれぞれのスクリプトを前記それぞれのマークルツリーのそれぞれのマークルルートに置き換えることによって、前記第 1 のブロックチェーンランザクションの編集済みバージョンを生成するステップとを含む方法。

【請求項 2】

各それぞれのスクリプトは、1 つまたは複数の関数を含み、前記それぞれのマークルツリーを構築する前記ステップは、連続する関数の 1 つまたは複数のそれぞれのシーケンスを前記マークルツリーのそれぞれのリーフとしてグループ化することを含む、請求項 1 に記載の方法。

【請求項 3】

各それぞれのスクリプトは、実行されたときに前記それぞれのターゲットデータを示すように構成されたそれぞれのデータ関数を含み、前記それぞれのマークルツリーを構築する前記ステップは、前記それぞれのマークルツリーのそれぞれのリーフの一部として前記それぞれのデータ関数を含めることを含む、請求項 2 に記載の方法。

【請求項 4】

各それぞれのスクリプトは、前記それぞれのターゲットデータのそれぞれのデータ長を含み、前記それぞれのマークルツリーを構築する前記ステップは、前記それぞれのデータ長に基づく、請求項 3 に記載の方法。

【請求項 5】

前記それぞれのマークルツリーを構築する前記ステップは、前記それぞれのデータ関数と同じ前記それぞれのマークルツリーのそれぞれのリーフの一部として前記それぞれのデータ長を含めることを含む、請求項 4 に記載の方法。

【請求項 6】

前記それぞれのターゲットデータは、前記それぞれのマークルツリーの複数の前記それぞれのリーフにわたって分割される、請求項 1 に記載の方法。

【請求項 7】

前記第 1 のブロックチェーンランザクションは、それぞれのターゲットデータを含む複数のそれぞれのスクリプトを含み、前記方法は、

前記複数のそれぞれのスクリプトの各々について、前記それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、前記それぞれのターゲットデータは、前記それぞれのマークルツリーの前記それぞれのリーフのうちの 1 つまたは複数にわたって分割される、ステップと、

各それぞれのスクリプトを前記それぞれのマークルツリーのそれぞれのマークルルートで置き換えることによって、前記第 1 のブロックチェーンランザクションの前記編集済みバージョンを生成するステップとを含む、請求項 1 に記載の方法。

【請求項 8】

前記ブロックチェーンランザクションの前記編集済みバージョンをハッシュすることによって、前記第 1 のブロックチェーンランザクションの前記編集済みバージョンの一

10

20

30

40

50

次トランザクション識別子を生成するステップを含む、請求項 1 に記載の方法。

【請求項 9】

前記第 1 のブロックチェーントランザクションに対する要求に応答して、前記ブロックチェーントランザクションの前記編集済みバージョンをエンティティに送信するステップを含む、請求項 1 に記載の方法。

【請求項 10】

前記第 1 のブロックチェーントランザクションの前記編集済みバージョンをメモリに記憶するステップを含む、請求項 1 に記載の方法。

【請求項 11】

前記方法は、トランザクションバリデータによって実行される、請求項 1 に記載の方法

10

【請求項 12】

前記第 1 のブロックチェーントランザクションの入力から署名を抽出するステップと、メッセージに対する前記署名を検証するステップであって、前記メッセージの 1 つまたは複数の部分は、前記第 1 のブロックチェーントランザクションの前記編集済みバージョンに基づく、ステップと

請求項 11 に記載の方法。

【請求項 13】

前記入力、それぞれのターゲットデータを含むそれぞれのスクリプトを含む前のブロックチェーントランザクションの出力を参照し、前記メッセージの一部は、前記前のブロックチェーントランザクションの前記出力に基づき、前記前のブロックチェーントランザクションの前記出力に基づく前記メッセージの前記一部は、前記それぞれのスクリプトに基づいて構築されたマークルツリーのマークルルートに基づき、前記それぞれのターゲットデータは、前記それぞれのマークルツリーの前記それぞれのリーフのうちの 1 つまたは複数にわたって分割される、請求項 12 に記載の方法。

20

【請求項 14】

前記それぞれのスクリプトのうちの少なくとも 1 つは、前記第 1 のブロックチェーントランザクションの第 1 の出力の第 1 のロックスクリプトであり、前記方法は、

前記修正された一次トランザクションにマッピングされた前記それぞれのマークルルートと、前記第 1 の出力のそれぞれのインデックスとをデータベースに記憶するステップを含む、請求項 8 に記載の方法。

30

【請求項 15】

第 2 の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、前記第 2 のブロックチェーントランザクションは、前記第 1 のブロックチェーントランザクションの前記修正された一次トランザクション識別子および前記第 1 の出力の前記インデックスを参照する第 1 の入力を含み、前記第 1 の入力は、第 1 のロック解除スクリプトを含む、ステップ

を含み、前記方法は、

前記データベースから、前記修正された一次トランザクション識別子および前記第 1 の出力の前記それぞれのインデックスにマッピングされた前記それぞれのマークルルートを取り出すことと、

40

前記第 1 の入力の前記第 1 のロック解除スクリプトに基づいて、前記第 1 の出力の部分ロックスクリプトを決定することと、

マークル包含証明を実行して、前記部分ロックスクリプトが前記それぞれのマークルルートに対応する前記それぞれのマークルツリーのそれぞれのリーフであることを確認することと、

前記部分ロックスクリプトに照らして前記第 1 のロック解除スクリプトを妥当性確認することと

によって前記第 2 のブロックチェーントランザクションを妥当性確認するステップ

を含む、請求項 14 に記載の方法。

50

【請求項 16】

前記それぞれのスクリプトのうちの少なくとも1つは、前記第1のブロックチェーントランザクションの第1の出力の第1のロックスクリプトであり、前記方法は、

前記修正された一次トランザクション識別子および前記第1の出力のそれぞれのインデックスにマッピングされた部分ロックスクリプトをデータベースに記憶するステップであって、前記部分ロックスクリプトは、前記第1のロックスクリプトの少なくとも一部を含むが、前記ターゲットデータは含まない、ステップ

請求項8に記載の方法。

【請求項 17】

第2の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、前記第2のブロックチェーントランザクションは、前記第1のブロックチェーントランザクションの前記修正された一次トランザクション識別子および前記第1の出力の前記インデックスを参照する第1の入力を含み、前記第1の入力は、第1のロック解除スクリプトを含む、ステップ

を含み、前記方法は、

前記データベースから、前記修正された一次トランザクション識別子および前記第1の出力の前記それぞれのインデックスにマッピングされた前記それぞれの部分ロックスクリプトを取り出すことと、

前記部分ロックスクリプトに照らして前記第1のロック解除スクリプトを妥当性確認することと

によって前記第2のブロックチェーントランザクションを妥当性確認するステップを含む、請求項16に記載の方法。

【請求項 18】

前記編集済みトランザクションは複数のフィールドを含み、前記方法は、

トランザクションマークルツリーを生成するステップであって、前記トランザクションマークルツリーのそれぞれの複数のそれぞれのリーフは、前記編集済みトランザクションの1つまたは複数のフィールドから形成される、ステップと、

前記編集済みトランザクションの二次トランザクション識別子を生成するステップであって、前記二次トランザクション識別子は、前記トランザクションマークルツリーのマークルルートを含む、ステップと

を含む、請求項1に記載方法。

【請求項 19】

コンピュータ機器であって、

1つまたは複数のメモリユニットを備えるメモリと、

1つまたは複数の処理ユニットを備える処理装置であって、前記メモリは、前記処理装置上で実行されるように構成されたコードを記憶し、前記コードは、前記処理装置上にあるときに、請求項1から18のいずれか一項に記載の方法を実行するように構成される、処理装置と

を備えるコンピュータ機器。

【請求項 20】

コンピュータ可読ストレージ上に具現化され、1つまたは複数のプロセッサ上で実行されると、請求項1から18のいずれか一項に記載の方法を実行するように構成されたコンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本開示は、ブロックチェーントランザクションに記憶されたコンテンツを編集する方法に関する。例えば、本方法は、機密データ、プライベートデータ、または違法データを隠すために使用され得る。

【背景技術】

【0002】

ブロックチェーンは、分散型データ構造の形態を指し、ブロックチェーンの複製コピーが、分散型ピアツーピア（P2P）ネットワーク（以下、「ブロックチェーンネットワーク」と呼ぶ）内の複数のノードの各々において維持され、広く公開されている。ブロックチェーンはデータブロックのチェーンを含み、各ブロックは1つまたは複数のトランザクションを含む。いわゆる「コインベーストランザクション」以外の各トランザクションは、1つまたは複数のコインベーストランザクションまで遡る1つまたは複数のブロックにまたがり得るシーケンス内の先行するトランザクションを指し示す。コインベーストランザクションについては以下でさらに説明する。ブロックチェーンネットワークにサブミットされたトランザクションは、新しいブロックに含まれる。新しいブロックは、「マイニング（mining）」と呼ばれることが多いプロセスによって作成され、このプロセスは、複数のノードの各々が「プルーフオブワーク（proof-of-work）」を実行しようと競うこと、すなわち、ブロックチェーンの新しいブロックに含まれるのを待っている、順序付けられ妥当性確認（validate）された保留中のトランザクションの表現に基づいて暗号パズルを解くことを伴う。ブロックチェーンはいくつかのノードでプルーニング（prune）され得、ブロックの公開は、単なるブロックヘッダの公開を通して達成され得ることに留意されたい。

10

【0003】

ブロックチェーン内のトランザクションは、デジタル資産（すなわち、いくつかのデジタルトークン）を伝達すること、仮想台帳またはレジストリにおけるエントリのセットを順序付けること、タイムスタンプエントリを受信および処理すること、および／またはインデックスポインタを時間順にすることという目的のうちの1つまたは複数のために使用され得る。ブロックチェーンは、ブロックチェーンの上に追加の機能を重ねるために活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、トランザクションにおける追加のユーザデータまたはデータへのインデックスの記憶を可能にし得る。単一のトランザクション内に記憶することができる最大データ容量に対してあらかじめ指定された制限はなく、したがって、より複雑なデータを組み込むことができる。例えば、これを使用して、ブロックチェーンに電子文書を記憶したり、オーディオまたはビデオデータを記憶したりすることができる。

20

【0004】

ブロックチェーンネットワークのノード（「マイナー（miner）」と呼ばれることが多い）は、以下でより詳細に説明する分散型のトランザクション登録および検証プロセスを実行する。要するに、このプロセスの間に、ノードは、トランザクションを妥当性確認し、有効なプルーフオブワークの解の識別を試みるブロックテンプレートにそれらを挿入する。有効な解が見つかり、ネットワークの他のノードに新しいブロックが伝搬され、これにより、各ノードが、新しいブロックをブロックチェーンに記録することができる。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、ユーザ（例えば、ブロックチェーンクライアントアプリケーション）は、ネットワークのノードのうちの1つにトランザクションを送信して、伝搬させる。トランザクションを受信したノードは、妥当性確認済みトランザクションを新しいブロックに組み込むプルーフオブワークの解を見つけようと競い合い得る。各ノードは、同じノードプロトコルを強制するように構成され、そのノードプロトコルには、トランザクションが有効であるための1つまたは複数の条件が含まれる。無効なトランザクションは、伝搬もブロックへの組み込みもされない。トランザクションが妥当性確認され、それによってブロックチェーン上に受け入れられたと仮定すると、トランザクション（任意のユーザデータを含む）は、不変の公開記録としてブロックチェーンネットワーク内のノードの各々において登録およびインデックス付けされたままになる。

30

40

【0005】

プルーフオブワークパズルを解くのに成功して最新のブロックを作成したノードは、典型的には、ある額のデジタル資産、すなわちいくつかのトークンを配布する「コインベ

50

ストランザクション」と呼ばれる新しいトランザクションで報酬が与えられる。無効なトランザクションの検出および拒否は、ネットワークのエージェントとして働き、不正を報告および阻止するようにインセンティブが与えられる競合ノードのアクションによって強制される。情報の広範な公開により、ユーザは、ノードの性能を連続的に監査することができる。単なるブロックヘッダの公開により、参加者は、ブロックチェーンの継続的な完全性を確実にすることができる。

【0006】

「出力ベースの」モデル（UTXOベース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つまたは複数の入力と1つまたは複数の出力とを含む。任意の使用可能な出力は、トランザクションの先行するシーケンスから導出可能なデジタル資産の額を指定する要素を含む。使用可能な出力は、UTXO（「未使用トランザクション出力」）と呼ばれることがある。出力は、出力を将来償還するための条件を指定するロックスクリプトをさらに含み得る。ロックスクリプトは、デジタルトークンまたは資産を妥当性確認および転送するために必要な条件を定義する述語（predicate）である。トランザクション（コインベーストランザクション以外）の各入力は、先行するトランザクションにおけるそのような出力へのポインタ（すなわち、参照）を含み、指し示された出力のロックスクリプトをロック解除するためのロック解除スクリプトをさらに含み得る。そのため、トランザクションのペアを考慮して、それらを第1のトランザクションおよび第2のトランザクション（または「ターゲット」トランザクション）と呼ぶ。第1のトランザクションは、デジタル資産の額を指定する少なくとも1つの出力を含み、出力をロック解除する1つまたは複数の条件を定義するロックスクリプトを含む。第2のターゲットトランザクションは、第1のトランザクションの出力へのポインタを含む少なくとも1つの入力と、第1のトランザクションの出力をロック解除するためのロック解除スクリプトとを含む。

【0007】

そのようなモデルでは、第2のターゲットトランザクションがブロックチェーンネットワークに送信されて伝搬されブロックチェーンに記録されるとき、各ノードで適用される有効性の基準のうちの1つは、ロック解除スクリプトが第1のトランザクションのロックスクリプトで定義された1つまたは複数の条件のすべてを満たすことである。もう1つは、第1のトランザクションの出力が別の先の有効なトランザクションによってまだ償還されていないということである。これらの条件のいずれかにしたがってターゲットトランザクションが無効であることを発見したノードは、（無効なトランザクションを登録するために伝搬する可能性はあるが、有効なトランザクションとしては）それを伝搬すること、ブロックチェーンに記録されるように新しいブロックにそれを含めることもしない。

【0008】

トランザクションモデルの別のタイプは、アカウントベースのモデルである。このケースでは、各トランザクションは、過去のトランザクションのシーケンスにおける先行するトランザクションのUTXOを参照することによってではなく、絶対アカウント残高を参照することによって転送されるべき額を定義する。すべ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にノードによって記憶され、絶えず更新される。

【発明の概要】

【0009】

ブロックチェーンは、不変であるように設計されており、これは、一旦トランザクションがブロックチェーンに書き込まれると、変えることが不可能であるとみなされることを意味する。しかしながら、トランザクションデータには、不正アクセスから保護する必要があるプライベート情報または機密情報が含まれている可能性がある。別の例として、トランザクションデータには、少なくともいくつかの管轄区域において違法であるコンテンツが含まれている可能性がある。プライベート、機密、または違法と見なされ得るデータは、本明細書では「秘密データ」と呼ばれ、そのようなデータへのアクセスの配布または許可を回避する必要があることから、「秘密」データと呼ぶ。したがって、秘密デ

10

20

30

40

50

タがすでにブロックチェーンに書き込まれている場合、それをどのように扱うかという問題が生じる。ブロックチェーン上での秘密データの共有を防ぐこと、例えば、プライベートデータを適切に保護し、データ保護法に準拠することも重要である。この問題は、任意の残りの（すなわち、非秘密）データがトランザクションに属すること、およびトランザクションが妥当性確認され将来のトランザクションで使用することができることを証明することが可能であることを確実にしつつ、ブロックチェーントランザクションに記憶された秘密データへのアクセスを除去または制限する方法と言い換えることもできる。

【0010】

本明細書に開示される一態様によれば、ブロックチェーントランザクションのデータを編集するコンピュータ実装方法であって、第1の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、第1のブロックチェーントランザクションは、編集対象のそれぞれのターゲットデータを含む1つまたは複数のそれぞれのスクリプトを含む、ステップと、1つまたは複数のそれぞれのスクリプトの少なくとも1つについて、それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの1つまたは複数にわたって分割される、ステップと、少なくとも1つのそれぞれのスクリプトをそれぞれのマークルツリーのそれぞれのマークルルートに置き換えることによって、第1のブロックチェーントランザクションの編集済みバージョンを生成するステップとを含む方法が提供される。

【0011】

本方法は、マークルツリーを使用してブロックチェーントランザクションからのターゲットデータ（例えば、秘密データ）を編集（または除去、または隠蔽）することを可能にする。トランザクションは、少なくとも1つのスクリプトを含む。スクリプトは、トランザクションの入力（ロック解除スクリプト）または出力（ロックスクリプト）にあり得る。スクリプトは、ターゲットデータ（編集対象のデータ）を含む。ターゲットデータは、上述したように、プライベート、機密、または違法である秘密データであってもよい。一般に、ターゲットデータは、トランザクションから除去されることとなるデータであり得、必ずしもプライベート、機密、または違法である必要はない。例えば、ターゲットデータは、そのサイズのために、例えばターゲットデータが大きな画像または音声ファイルを含み得るという理由で、トランザクションから除去される必要があってもよい。別の例として、ターゲットデータは、それがバグまたは他の有害なデータ（データを記憶または処理するコンピュータに損害を与え得るという意味で有害である）を含むという理由で、トランザクションから除去される必要があってもよい。

【0012】

マークルツリーは、スクリプトに基づいて、すなわちスクリプトを使用して構築される。スクリプトの異なる部分は、マークルツリーの異なるリーフとして使用される。マークルツリーの少なくとも1つのリーフは、ターゲットデータの一部を含む。いくつかの例では、単一のリーフノードがターゲットデータ全体を含む。他の例では、ターゲットデータは、複数のリーフにわたって分割される。当技術分野で知られているように、マークルツリーのリーフはハッシュされてリーフハッシュを形成し、リーフハッシュのペアが連結されてハッシュされ、それぞれの内部ハッシュを形成する。内部ハッシュのペアを連結してハッシュするプロセスは、単一のハッシュ（マークルルート）が残るまで繰り返される。スクリプトはマークルルートで置き換えられ、トランザクションの編集済みバージョン（「編集済みトランザクション」）を形成する。したがって、ターゲットデータは、ターゲットデータを含むスクリプトをマークルルートで置き換えることによって、トランザクションから編集または除去される。

【0013】

編集済みトランザクションが、トランザクションの元のバージョンの代わりに記憶および／または共有され得、したがって、ターゲット（例えば、秘密）データの記憶および／または配布を防止する。

10

20

30

40

50

【0014】

本方法は、トランザクションジェネレータ、例えばユーザによって実行され得る。例えば、本方法は、編集済みトランザクションに基づいて署名を生成するように実行され得る。本方法は、トランザクションバリデータ、例えばブロックチェーンノードによって実行されてもよい。例えば、本方法は、編集済みトランザクションに基づいて署名を検証し、編集済みトランザクションのトランザクション識別子を生成し、および／または編集済みトランザクションを妥当性確認するように実行されてもよい。

【0015】

いくつかの実施形態では、本方法は、選択的開示機構を実装するために使用され得、これにより、証明者は、検証者に開示されるべきトランザクションコンテンツのサブセットを選択することができる一方で、一部のコンテンツは隠されたままにすることができる。この機構は、スクリプトにとらわれず、任意のスクリプト機構によって埋め込まれたコンテンツの選択的開示を容易にする。

【図面の簡単な説明】

【0016】

本開示の実施形態の理解を助け、そのような実施形態がどのように実施され得るかを示すために、単なる例として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得るトランザクションのいくつかの例を概略的に示す。

【図3】使用不可能な出力に埋め込まれたデータを含む例示的なブロックチェーントランザクションの構造を概略的に示す。

【図4】pay-to-public-keyハッシュ（P2PKH）スクリプトをマークル化（Merkelize）する例を概略的に示す。

【図5】使用不可能な出力スクリプトをマークル化する例を概略的に示す。

【図6】マークル化されたトランザクションを使用するための例示的な大まかなアーキテクチャの概略ブロック図であり、前のマークル化された（Merkalized）スクリプトを使用するトランザクションが作成され、トランザクションがブロードキャストされ、ノードが、編集済みの前のトランザクションを使用して、前のマークル化されたスクリプトに照らしてトランザクションを妥当性確認する。

【図7】編集済みのフィールドが破線付きの実線ボックスによって示される、例示的な編集済みトランザクションを概略的に示す。

【図8】編集済みトランザクションの二次識別子を生成する例を概略的に示す。

【発明を実施するための形態】

【0017】

1. 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などの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で構成され得る。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）ネットワーク106を形成するように構成され得る複数のブロックチェーンノード104を含む。図示されていないが、ブロックチェーンノード104は、ほぼ完全なグラフとして構成され得る。したがって、各ブロックチェーンノード104は、他のブロックチェーンノード104に高度に接続される。

【0018】

各ブロックチェーンノード104は、ピアのコンピュータ機器を含み、ノード104の異なるものは、異なるピアに属する。各ブロックチェーンノード104は、1つまたは複数のプロセッサ、例えば、1つまたは複数の中央処理装置（CPU）、アクセラレータプロセッサ、特定用途向けプロセッサおよび／またはフィールドプログラマブルゲートアレイ（FPGA）、ならびに特定用途向け集積回路（ASIC）などの他の機器を含む処理

装置を備える。各ノードはまた、メモリ、すなわち、1つまたは複数の非一時的コンピュータ可読媒体の形態のコンピュータ可読ストレージを備える。メモリは、1つまたは複数のメモリ媒体、例えば、ハードディスクなどの磁気媒体、ソリッドステートドライブ（SSD）、フラッシュメモリもしくはEEPROMなどの電子媒体、および／または光ディスクドライブなどの光学媒体を採用する1つまたは複数のメモリユニットを備え得る。

【0019】

ブロックチェーン150は、データブロック151のチェーンを含み、ブロックチェーン150のそれぞれのコピーは、分散型またはブロックチェーンネットワーク106内の複数のブロックチェーンノード104の各々で維持される。上述したように、ブロックチェーン150のコピーを維持することは、ブロックチェーン150を完全に記憶することを必ずしも意味しない。代わりに、ブロックチェーン150は、各ブロックチェーンノード150が各ブロック151のブロックヘッダ（後述する）を記憶している限り、データがブルーニングされ得る。チェーン内の各ブロック151は、1つまたは複数のトランザクション152を含み、この文脈におけるトランザクションは、データ構造の一種を指す。データ構造の性質は、トランザクションモデルまたは方式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、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つの入力および少なくとも1つの出力を含む。各出力は、プロパティとしてデジタル資産の量を表す額を指定し、その例は、出力が暗号的にロックされている（ロック解除され、それによって償還または使用されるためにはそのユーザの署名または他の解決策を必要とする）ユーザ103である。各入力は、先行するトランザクション152の出力を指し示し、それによってトランザクションをリンクする。

【0020】

各ブロック151はまた、ブロック151への順番を定義するために、チェーン内の前に作成されたブロック151を指し示すブロックポインタ155を含む。各トランザクション152（コインベーストランザクション以外）は、トランザクションのシーケンスへの順番を定義するために、前のトランザクションへ戻るポインタを含む（注意：トランザクション152のシーケンスは分岐することが可能である）。ブロック151のチェーンは、チェーン内の最初のブロックであったジェネシスブロック（Gb：genesis block）153まで戻る。チェーン150内の早期にある1つまたは複数の元のトランザクション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を指し示していた。

【0021】

ブロックチェーンノード104の各々は、トランザクション152を他のブロックチェーンノード104にフォワードし、それによってトランザクション152をネットワーク106全体に伝搬させるように構成される。各ブロックチェーンノード104は、ブロック151を作成し、同じブロックチェーン150のそれぞれのコピーをそれらのそれぞれのメモリに記憶するように構成される。各ブロックチェーンノード104はまた、ブロック151に組み込まれるのを待っているトランザクション152の順序付きセット（またはプール）154を維持する。順序付きプール154は、「メムプール（mempool）」と呼ばれることが多い。本明細書におけるこの用語は、任意の特定のブロックチェーン、プロトコル、またはモデルに限定することを意図するものではない。これは、ノード104が有効であるとして受け入れたトランザクションの順序付きセットを指し、それに対して、ノード104は、同じ出力を使用しようとする他のトランザクションを受け入れないように義務付けられている。

【0022】

所与の現在のトランザクション152jにおいて、その入力（または各入力）は、トランザクションのシーケンスにおける先行するトランザクション152iの出力を参照するポインタを含み、この出力が現在のトランザクション152jにおいて償還または「使用

10

20

30

40

50

」されるべきであることを指定する。使用することまたは償還することは、必ずしも金融資産の移転を意味するものではないが、それは確かに1つの一般的な用途である。より一般的には、使用することは、出力を消費すること、または別の以降の（onward）トランザクションにおける1つまたは複数の出力にそれを割り当てることとして説明され得る。一般に、先行するトランザクションは、順序付きセット154または任意のブロック151内の任意のトランザクションであり得る。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となるためには存在および妥当性確認される必要があるが、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152jが作成されるときまたはネットワーク106に送信されるときに必ずしも存在する必要はない。したがって、本明細書における「先行する（preceding）」は、ポインタによってリンクされた論理シーケンスにおける先行するものを指し、必ずしも時間シーケンスにおける作成または送信の時間を指すものではなく、したがって、トランザクション152i、152jが順不同に作成または送信されることを必ずしも除外するものではない（オーファントランザクションに関する以下の説明を参照）。先行するトランザクション152iは、同様に、先のトランザクション（antecedent transaction）または先行したトランザクション（predecessor transaction）とも呼ばれる。

10

【0023】

現在のトランザクション152jの入力はまた、入力認可、例えば、先行するトランザクション152iの出力がロックされている対象のユーザ103aの署名を含む。次に、現在のトランザクション152jの出力は、新しいユーザまたはエンティティ103bに暗号的にロックされ得る。したがって、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の入力において定義された額を、現在のトランザクション152jの出力において定義されたように、新しいユーザまたはエンティティ103bに転送することができる。いくつかのケースでは、トランザクション152は、複数のユーザまたはエンティティ（そのうちの1つは残り（change）を与えるために元のユーザまたはエンティティ103aであり得る）間で入力額を分割するために複数の出力を有し得る。いくつかのケースでは、トランザクションはまた、1つまたは複数の先行するトランザクションの複数の出力からの額をまとめ、現在のトランザクションの1つまたは複数の出力に再分配するために複数の入力を有することができる。

20

【0024】

ビットコインなどの出力ベースのトランザクションプロトコルによれば、個々のユーザまたは組織などの当事者103が（手動でまたは当事者によって採用される自動プロセスによって）新しいトランザクション152jを制定することを望むとき、制定を行う当事者は、新しいトランザクションをそのコンピュータ端末102から受信者に送信する。制定を行う当事者または受信者は、最終的に、このトランザクションをネットワーク106のブロックチェーンノード104の1つまたは複数（これは、現在では、典型的にはサーバまたはデータセンタであるが、原則として他のユーザ端末であってもよい）に送信する。新しいトランザクション152jを制定する当事者103がトランザクションをブロックチェーンノード104の1つまたは複数に直接送信し、いくつかの例では、受信者に送信しないことも除外されない。トランザクションを受信するブロックチェーンノード104は、ブロックチェーンノード104の各々で適用されるブロックチェーンノードプロトコルにしたがって、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。ブロックチェーンノードプロトコルは、典型的には、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の順序付きシーケンス内で前のトランザクション152iに依存する予想される署名と一致することをチェックするようにブロックチェーンノード104に要求する。そのような出力ベースのトランザクションプロトコルでは、これは、新しいトランザクション152jの入力に含まれる当事者103の暗号署名または他の認可が、新しいトランザクションが使用する（または「割り当てる」）先行するトランザクション152iの出力において定義される条件と一致することをチェックすることを含み得、この条件は、典型的には、新しいトランザクション152jの入力における暗号署名ま

30

40

50

たは他の認可が、新しいトランザクションの入力がリンクされている前のトランザクション152 iの出力をロック解除することを少なくともチェックすることを含む。条件は、先行するトランザクション152 iの出力に含まれるスクリプトによって少なくとも部分的に定義され得る。代替的に、それは、単にブロックチェーンノードプロトコルのみによって固定されてもよいし、これらの組合せによるものであってもよい。いずれにしても、新しいトランザクション152 jが有効である場合、ブロックチェーンノード104は、それをブロックチェーンネットワーク106内の1つまたは複数の他のブロックチェーンノード104にフォワードする。これらの他のブロックチェーンノード104は、同じブロックチェーンノードプロトコルにしたがって同じテストを適用し、そして、新しいトランザクション152 jを1つまたは複数のさらなるノード104にフォワードし、以下同様である。このようにして、新しいトランザクションはブロックチェーンノード104のネットワーク全体に伝搬される。

10

【0025】

出力ベースのモデルでは、所与の出力（例えば、UTXO）が割り当てられる（または、「使用される」）かどうかの定義は、それがブロックチェーンノードプロトコルにしたがって別の以降のトランザクション152 jの入力によって有効に償還されたかどうかである。トランザクションが有効であるための別の条件は、それが償還しようとする先行するトランザクション152 iの出力が、別のトランザクションによってまだ償還されていないことである。この場合も同様に、有効ではない場合、トランザクション152 jは、（無効としてフラグ付けされ、警告のために伝搬されない限り）伝搬されることも、ブロックチェーン150内に記録されることもない。これは、トランザクタ（transactor）が同じトランザクションの出力を複数回割り当てようとする二重支出を防止する。一方、アカウントベースのモデルは、アカウント残高を維持することによって二重支出を防止する。ここでも、トランザクション順序が定義されているので、アカウント残高は常に単一の定義された状態にある。

20

【0026】

トランザクションを妥当性確認することに加えて、ブロックチェーンノード104はまた、「ブルーフオブワーク」によって支持される、一般にマイニングと呼ばれるプロセスにおいてトランザクションのブロックを最初に作成しようと競い合う。ブロックチェーンノード104において、新しいトランザクションが、ブロックチェーン150上に記録されたブロック151内にまだ現れていない有効なトランザクションの順序付きプール154に追加される。次いで、ブロックチェーンノードは、暗号パズルを解こうとすることで、トランザクションの順序付きセット154からトランザクション152の新しい有効なブロック151を組み立てようと競い合う。典型的には、これは、ナンスが保留中のトランザクションの順序付きプール154の表現と連結されハッシュされたときにハッシュの出力が所定の条件を満たすような「ナンス」値を探索することを含む。例えば、所定の条件とは、ハッシュの出力が特定の所定の数の先行ゼロを有することであり得る。これは、ブルーフオブワークパズルの1つの特定のタイプにすぎず、他のタイプが除外されないことに留意されたい。ハッシュ関数のプロパティは、その入力に対して予測不可能な出力を持つことである。したがって、この探索は、総当たりでしか実行することができないので、パズルを解こうとしている各ブロックチェーンノード104でかなりの量の処理リソースを消費する。

30

40

【0027】

最初にパズルを解いたブロックチェーンノード104は、これをネットワーク106に公表し、後にネットワーク内の他のブロックチェーンノード104によって容易にチェックすることができるその解をブルーフとして提供する（ハッシュに対する解が与えられると、ハッシュの出力が条件を満たすことをチェックすることは簡単である）。この最初のブロックチェーンノード104は、ブロックを、このブロックを受け入れる他のノードのしきい値コンセンサスに伝搬し、プロトコルルールを強制する。次いで、トランザクションの順序付きセット154は、ブロックチェーンノード104の各々によってブロックチ

50

チェーン150内に新しいブロック151として記録されるようになる。ブロックポインタ155はまた、チェーン内の前に作成されたブロック151 $n-1$ を指し示す新しいブロック151 n に割り当てられる。プルーフオブワークの解を作成するために必要とされる、例えばハッシュの形態の、かなりの量の労力は、ブロックチェーンプロトコルのルールに従うという最初のノード104の意図を示す。そのようなルールは、前に妥当性確認されたトランザクションと同じ出力の使用または割当てを行った場合にトランザクションを有効として受け入れること（別名二重支出としても知られている）を行わないことを含む。ブロック151は、一旦作成されると、ブロックチェーンネットワーク106内のブロックチェーンノード104の各々において認識および維持されるので、修正することができない。ブロックポインタ155はまた、ブロック151に順番を付与する。トランザクション152は、ネットワーク106内の各ブロックチェーンノード104において順序付き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。

10

20

30

40

50

【0028】

任意の所与の時間にパズルを解こうと競い合う異なるブロックチェーンノード104は、それらがいつ解を探索し始めたかまたはトランザクションが受信された順序に応じて、任意の所与の時間に、まだ公開されていないトランザクションのプール154の異なるスナップショットに基づいてそれを行っていてもよいことに留意されたい。誰がそれぞれのパズルを最初に解いても、どのトランザクション152が次の新しいブロック151 n にどの順序で含まれるかを定義し、公開されていないトランザクションの現在のプール154が更新される。次いで、ブロックチェーンノード104は、新たに定義された、公開されていないトランザクションの順序付きプール154からブロックを作成しようと競い合い続け、以下同様である。2つのブロックチェーンノード104が互いに非常に短い時間内にパズルを解いて、ブロックチェーンの相反する見解がノード104間で伝搬される場合に発生し得る任意の「フォーク」を解決するためのプロトコルも存在する。要するに、フォークのどちらのブランチでも最も長く成長した方が、確定的なブロックチェーン150となる。同じトランザクションが両方のフォークに現れるので、これがネットワークのユーザまたはエージェントに影響を与えないことに留意されたい。

【0029】

ビットコインブロックチェーン（およびほとんどの他のブロックチェーン）によれば、新しいブロック104の構築に成功したノードには、（あるエージェントまたはユーザから別のエージェントまたはユーザにある額のデジタル資産を転送するエージェント間またはユーザ間のトランザクションとは対照的に）追加の定義された量のデジタル資産を分配する新しい特別な種類のトランザクションにおいて、受け入れられた追加の額のデジタル資産を新たに割り当てる能力が与えられる。この特別なタイプのトランザクションは、通常、「コインベーストランザクション」と呼ばれるが、「開始トランザクション（initiation transaction）」または「生成トランザクション（generation transaction）」と称されることもある。これは典型的に、新しいブロック151 n の最初のトランザクションを形成する。プルーフオブワークは、新しいブロックを構築するノードが、この特別なトランザクションが後に償還されることを可能にするプロトコルルールに従うという意図を示す。ブロックチェーンプロトコルルールは、この特別なトランザクションが償還され得る前に、満期期間、例えば100個のブロックを必要とし得る。多くの場合、通常の（非生成）トランザクション152はまた、そのトランザクションが公開されたブロック151 n を作成したブロックチェーンノード104にさらに報酬を与えるために、その出力のうちの1つにおいて追加のトランザクション手数料を指定する。この手数料は通常「トランザクション手数料」と呼ばれ、以下で説明される。

【0030】

トランザクション妥当性確認および公開に関与するリソースに起因して、典型的には、ブロックチェーンノード104の少なくとも各々は、1つまたは複数の物理サーバユニットで構成されるサーバの形態をとるか、さらにはデータセンタ全体の形態をとる。しかし

ながら、原則として、任意の所与のブロックチェーンノード 104 は、ユーザ端末または互いにネットワーク化されたユーザ端末のグループの形態をとることができる。

【0031】

各ブロックチェーンノード 104 のメモリは、そのそれぞれの 1 つまたは複数の役割を実行し、ブロックチェーンノードプロトコルにしたがってトランザクション 152 を処理するために、ブロックチェーンノード 104 の処理装置上で実行されるように構成されたソフトウェアを記憶する。本明細書においてブロックチェーンノード 104 に帰する任意のアクションは、それぞれのコンピュータ機器の処理装置上で実行されるソフトウェア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は、アプリケーション層、またはオペレーティングシステム層もしくはプロトコル層などの下位層、またはこれらの任意の組合せにおける 1 つまたは複数のアプリケーションにおいて実装され得る。

10

【0032】

消費ユーザの役割を担う複数の当事者 103 の各々のコンピュータ機器 102 もネットワーク 101 に接続されている。これらのユーザは、ブロックチェーンネットワーク 106 と対話し得るが、トランザクションの妥当性確認にもブロックの構築にも参加しない。これらのユーザまたはエージェント 103 のうちのいくつかは、トランザクションの送信者および受信者として動作し得る。他のユーザは、必ずしも送信者または受信者として動作することなくブロックチェーン 150 と対話し得る。例えば、いくつかの当事者は、ブロックチェーン 150 のコピーを記憶する（例えば、ブロックチェーンノード 104 からブロックチェーンのコピーを取得した）ストレージエンティティとして動作し得る。

20

【0033】

当事者 103 のうちのいくつかまたはすべては、異なるネットワーク、例えば、ブロックチェーンネットワーク 106 の上にオーバーレイされたネットワークの一部として接続され得る。ブロックチェーンネットワークのユーザ（「クライアント」と呼ばれることが多い）は、ブロックチェーンネットワーク 106 を含むシステムの一部であるといえるが、これらのユーザは、ブロックチェーンノードに要求される役割を果たさないで、ブロックチェーンノード 104 ではない。代わりに、各当事者 103 はブロックチェーンネットワーク 106 と対話してもよく、それによって、ブロックチェーンノード 106 に接続する（すなわち通信する）ことでブロックチェーン 150 を利用し得る。2 つの当事者 103 およびそれらのそれぞれの機器 102、すなわち、第 1 の当事者 103 a およびそのそれぞれのコンピュータ機器 102 a、ならびに第 2 の当事者 103 b およびそのそれぞれのコンピュータ機器 102 b は、例示の目的で示されている。そのような当事者 103 およびそれらのそれぞれのコンピュータ機器 102 ははるかに多く存在し、システム 100 に参加し得るが、便宜上、それらは図示されていないことが理解されよう。各当事者 103 は、個人または組織であり得る。純粋に例示として、第 1 の当事者 103 a は、本明細書ではアリスと呼ばれ、第 2 の当事者 103 b はボブと呼ばれるが、これは限定的なものではなく、本明細書におけるアリスまたはボブへのいかなる言及も、それぞれ「第 1 の当事者」および「第 2 の当事者」と置き換えられ得ることが理解されよう。

30

【0034】

各当事者 103 のコンピュータ機器 102 は、1 つまたは複数のプロセッサ、例えば、1 つまたは複数の 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、および／または FPGA を含むそれぞれの処理装置を備える。各当事者 103 のコンピュータ機器 102 は、メモリ、すなわち、1 つまたは複数の非一時的コンピュータ可読媒体の形態のコンピュータ可読ストレージをさらに備える。このメモリは、1 つまたは複数のメモリ媒体、例えば、ハードディスクなどの磁気媒体、SSD、フラッシュメモリもしくは EEPROM などの電子媒体、および／または光ディスクドライブなどの光学媒体を採用する 1 つまたは複数のメモリユニットを備え得る。各当事者 103 のコンピュータ機器 102 上のメモリは、処理装置上で実行されるように構成された少なくとも 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。本明細書において所与の当事者 103 に帰する任意のアクションは、それぞれの

40

50

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して実行され得ることが理解されよう。各当事者 103 のコンピュータ機器 102 は、少なくとも 1 つのユーザ端末、例えば、デスクトップもしくはラップトップコンピュータ、タブレット、スマートフォン、またはスマートウォッチなどのウェアラブルデバイスを含む。所与の当事者 103 のコンピュータ機器 102 はまた、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などの 1 つまたは複数の他のネットワーク化されたリソースを含み得る。

【0035】

クライアントアプリケーション 105 は、最初に、1 つまたは複数の適切なコンピュータ可読ストレージ上で任意の所与の当事者 103 のコンピュータ機器 102 に提供され得、例えば、サーバからダウンロードされ得るか、またはリムーバブル 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もしくはテープ、CD もしくは DVD ROM などの光ディスク、またはリムーバブル光学ドライブなどのリムーバブル記憶デバイス上で提供され得る。

【0036】

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。これは 2 つの主要な機能を有する。これらのうちの 1 つは、それぞれの当事者 103 が、トランザクション 152 を作成し、認可し（例えば署名し）、1 つまたは複数のビットコインノード 104 に送信することを可能にして、トランザクション 152 をブロックチェーンノード 104 のネットワーク全体に伝搬させ、それによってブロックチェーン 150 に含まれるようにすることである。もう 1 つは、それぞれの当事者に、その当事者が現在所有しているデジタル資産の額を報告することである。出力ベースのシステムでは、この第 2 の機能は、当該当事者に属するブロックチェーン 150 全体に散在している様々なトランザクション 152 の出力において定義された額を照合することを含む。

【0037】

様々な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 に統合されるものとして説明され得るが、必ずしもこれに限定されるものではなく、代わりに、本明細書で説明される任意のクライアント機能は、例えば、API を介してインターフェースする、または一方が他方へのプラグインである 2 つ以上の別個のアプリケーション一式において実装され得ることに留意されたい。より一般的には、クライアント機能は、アプリケーション層もしくはオペレーティングシステムなどの下位層、またはこれらの任意の組合せにおいて実装され得る。以下では、クライアントアプリケーション 105 に関して説明するが、これに限定されないことが理解されよう。

【0038】

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またはソフトウェア 105 のインスタンスは、ネットワーク 106 のブロックチェーンノード 104 のうちの少なくとも 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、それぞれの当事者 103 が受信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 にクエリを行うためにブロックチェーンノード 104 にコンタクトすることができる（または、実施形態では、ブロックチェーン 150 は、部分的にその公開性（public visibility）を通じてトランザクションにおける信頼を提供する公共施設であるので、実際にブロックチェーン 150 における他の当事者のトランザクションを検査する）。各コンピュータ機器 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したがってトランザクション 152 を定式化し、送信するように構成される。上述したように、各ブロックチェーンノード 104 は、ブロックチェーンノードプロトコルにしたがってトランザクション 152 を妥当性確認し、トランザクション 152 をフォワードして、それらをブロックチェーンネットワーク 106 全体に伝搬するように構成されたソフトウェアを実行する。トランザクションプロトコルおよび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に従い（go w

10

20

30

40

50

ith)、一緒に所与のトランザクションモデルを実装する。ブロックチェーン150内のすべてのトランザクション152に対して同じトランザクションプロトコルが使用される。ネットワーク106内のすべてのノード104によって同じノードプロトコルが使用される。

【0039】

所与の当事者103、例えばアリスが、ブロックチェーン150に含まれるべき新しいトランザクション152jを送信することを望むとき、アリスは、関連トランザクションプロトコルにしたがって(アリスのクライアントアプリケーション105内のウォレット機能を使用して)新しいトランザクションを定式化する。次いで、アリスは、クライアントアプリケーション105から、アリスが接続されている1つまたは複数のブロックチェーンノード104にトランザクション152を送信する。例えば、これは、アリスのコンピュータ102に最良に接続されたブロックチェーンノード104であり得る。任意の所与のブロックチェーンノード104は、新しいトランザクション152jを受信すると、ブロックチェーンノードプロトコルおよびそのそれぞれの役割にしたがってそれを処理する。これには、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すかを最初にチェックすることが含まれ、その例については、以下でより詳細に説明する。いくつかのトランザクションプロトコルでは、妥当性確認のための条件は、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り得る。代替的に、条件は、単にノードプロトコルの組み込み特徴であってもよいし、スクリプトとノードプロトコルとの組合せによって定義されてもよい。

【0040】

新たに受信されたトランザクション152jが、有効であるとみなされるためのテストにパスすることを条件として(すなわち、それが「妥当性確認される」ことを条件として)、トランザクション152jを受信する任意のブロックチェーンノード104は、そのブロックチェーンノード104において維持されるトランザクションの順序付きセット154に新たな妥当性確認済みトランザクション152を追加する。さらに、トランザクション152jを受信する任意のブロックチェーンノード104は、妥当性確認済みトランザクション152をネットワーク106内の1つまたは複数の他のブロックチェーンノード104に伝搬する。各ブロックチェーンノード104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、ネットワーク106全体にわたってすぐに伝搬されることを意味する。

【0041】

所与のブロックチェーンノード104において維持される保留中のトランザクションの順序付きプール154に承認されると、そのブロックチェーンノード104は、新しいトランザクション152を含むそれぞれのプール154の最新バージョンに対してプルーフオブワークパズルを解こうと競い始める(他のブロックチェーンノード104が、トランザクションの異なるプール154に基づいてパズルを解こうと試みている可能性があるが、どのノードでも最初に解いたものが、最新のブロック151に含まれるトランザクションのセットを定義することを想起されたい。最終的に、ブロックチェーンノード104は、アリスのトランザクション152jを含む順序付きプール154の一部についてパズルを解くことになる。)新しいトランザクション152jを含むプール154に対してプルーフオブワークが行われると、それは普遍的にブロックチェーン150内のブロック151のうちの1つの一部となる。各トランザクション152は、先のトランザクションへ戻るポインタを含むので、トランザクションの順序も不変的に記録される。

【0042】

異なるブロックチェーンノード104は、最初、所与のトランザクションの異なるインスタンスを受信し得るので、1つのインスタンスが新しいブロック151において公開される(この時点で、公開されたインスタンスが唯一の有効なインスタンスであることにすべてのブロックチェーンノード104が同意している)までは、どのインスタンスが「有効」であるかについて相反する見解を有する。ブロックチェーンノード104が1つのイ

ンスタンスを有効として受け入れ、次いで、別のインスタンスがブロックチェーン150に記録されていることを発見した場合、そのブロックチェーンノード104は、これを受け入れなければならない、最初に受け入れたインスタンス（すなわち、ブロック151で公開されていないもの）を破棄する（すなわち、無効として扱う）。

【0043】

いくつかのブロックチェーンネットワークによって運用される代替タイプのトランザクションプロトコルは、アカウントベースのトランザクションモデルの一部として、「アカウントベース」プロトコルと呼ばれ得る。アカウントベースのケースでは、各トランザクションは、過去のトランザクションのシーケンスにおける先行するトランザクションのUTXOを参照することによってではなく、絶対アカウント残高を参照することによって転送されるべき額を定義する。すべ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にそのネットワークのノードによって記憶され、絶えず更新される。そのようなシステムでは、トランザクションは、アカウントの実行中のトランザクションタリー（「ポジション」とも呼ばれる）を使用して順序付けられる。この値は、送信者によってそ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。加えて、トランザクションにおけるオプションのデータフィールドも署名され得る。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指し示し得る。

【0044】

2. UTXOベースのモデル

図2は、例示的なトランザクションプロトコルを示す。これは、UTXOベースのプロトコルの一例である。トランザクション152（「Tx」と略記される）は、ブロックチェーン150の基本的なデータ構造である（各ブロック151は1つまたは複数のトランザクション152を含む）。以下では、出力ベースまたは「UTXO」ベースのプロトコルを参照して説明する。しかしながら、これはすべての可能な実施形態に限定されるものではない。例示的なUTXOベースのプロトコルは、ビットコインを参照して説明されるが、他の例示的なブロックチェーンネットワーク上でも等しく実装され得ることに留意されたい。

【0045】

UTXOベースのモデルでは、各トランザクション（「Tx」）152は、1つまたは複数の入力202および1つまたは複数の出力203を含むデータ構造を含む。各出力203は、未使用トランザクション出力（UTXO）を含み得、これは、（UTXOがまだ償還されていない場合）別の新しいトランザクションの入力202のソースとして使用され得る。UTXOは、デジタル資産の額を指定する値を含む。これは、分散型台帳上のトークンの設定数を表す。UTXOはまた、他の情報の中でも、元となるトランザクションのトランザクションIDを含み得る。トランザクションデータ構造は、入力フィールド（複数可）202および出力フィールド（複数可）203のサイズを示すインジケータを含み得るヘッダ201も含み得る。ヘッダ201はまた、トランザクションのIDを含み得る。実施形態では、トランザクションIDは、（トランザクションID自体を除く）トランザクションデータのハッシュであり、ノード104にサブミットされる生のトランザクション152のヘッダ201に記憶される。

【0046】

アリス103aが、当該デジタル資産の額をボブ103bに転送するトランザクション152jを作成することを望むとする。図2では、アリス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、シーケンス内の先行するトランザクション152iの出力203においてアリスにロックされたデジタル資産の額をとり、これのうちの少なくとも一部をボブに転送する。先行するトランザクション152iは、図2では「Tx₀」とラベル付けされている。Tx₀およびTx₁は、単なる任意のラベルである。それらは、Tx₀がブロックチェーン151内の最初のトランザクションであること、またはTx₁がプール154内のすぐ次のトランザクションであることを必ずしも

10

20

30

40

50

意味するものではない。Tx₁は、アリスにロックされた未使用の出力203を依然として有する任意の先行する（すなわち先の）トランザクションを指し示すことができる。

【0047】

先行するトランザクションTx₀は、アリスが新しいトランザクションTx₁を作成した時点では、または少なくともアリスがそれをネットワーク106に送信する時点までには、すでに妥当性確認されブロックチェーン150のブロック151に含まれている可能性がある。それは、その時点でブロック151のうちの1つにすでに含まれていてもよいし、順序付きセット154で依然として待機していてもよく、このケースでは、すぐに新しいブロック151に含まれることになる。代替的に、Tx₀およびTx₁を作成してネットワーク106と一緒に送信することもできるし、ノードプロトコルが「オーファン」トランザクションのバッファリングを可能にする場合には、Tx₀をTx₁の後に送信することさえもできる。トランザクションのシーケンスの文脈において本明細書で使用される「先行する」および「後続する」という用語は、トランザクション内で指定されている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し示すかなど）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、同様に、「先行するもの（predecessor）」および「後続するもの（successor）」、または「先の（antecedent）」および「後の（descendant）」、「親（parent）」および「子（child）」などと置き換えられ得る。これは、それらの作成、ネットワーク106への送信、または任意の所与のブロックチェーンノード104への到着の順序を必ずしも意味するものではない。それにもかかわらず、先行するトランザクション（先のトランザクションまたは「親」）を指し示す後続するトランザクション（後のトランザクションまたは「子」）は、親トランザクションが妥当性確認されない限り、妥当性確認されない。親より前にブロックチェーンノード104に到着する子は、オーファンとみなされる。それは、ノードプロトコルおよび／またはノード挙動に応じて、親を待つために特定の時間バッファされるかまたは破棄され得る。

【0048】

先行するトランザクションTx₀の1つまたは複数の出力203のうちの1つは、本明細書ではUTXO₀とラベル付けされた特定のUTXOを含む。各UTXOは、UTXOによって表されるデジタル資産の額を指定する値と、ロックスクリプトとを含み、ロックスクリプトは、後続するトランザクションが妥当性確認され、したがってUTXOが正常に償還されるために、後続するトランザクションの入力202内のロック解除スクリプトが満たさなければならない条件を定義する。典型的には、ロックスクリプトは、その額を特定の当事者（それが含まれるトランザクションの受益者）にロックする。すなわち、ロックスクリプトは、典型的には、後続するトランザクションの入力内のロック解除スクリプトに、先行するトランザクションがロックされる当事者の暗号署名が含まれるという条件を含むロック解除条件を定義する。

【0049】

ロックスクリプト（通称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、ブロックチェーンネットワークによって使用される「スクリプト（Script）」（大文字S）と呼ばれる。ロックスクリプトは、トランザクション出力203を使用するためにどの情報が必要とされるか、例えばアリスの署名の必要性、を指定する。ロック解除スクリプトはトランザクションの出力に現れる。ロック解除スクリプト（通称scriptSig）は、ロックスクリプト基準を満たすのに必要な情報を提供するドメイン固有言語で書かれたコードの一部である。例えば、それはボブの署名を含み得る。ロック解除スクリプトは、トランザクションの入力202に現れる。

【0050】

つまり、図示の例では、Tx₀の出力203内のUTXO₀は、UTXO₀が償還されるために（厳密には、UTXO₀を償還しようとする後続するトランザクションが有効となるために）アリスの署名Sig_{PA}を必要とするロックスクリプト[Checksig_{PA}]を含

む。 $[Checksig\ P_A]$ は、アリスの公開鍵－私有鍵ペアの公開鍵 P_A の表現（すなわち、ハッシュ）を含む。 Tx_1 の入力202は、（例えば、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そのトランザクションID、 $TxID_0$ によって） Tx_1 を指し示すポインタを含む。 Tx_1 の入力202は、 Tx_0 の任意の他の可能な出力の中から $UTXO_0$ 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 の入力202は、アリスが鍵ペアのアリスの私有鍵をデータの所定の部分（暗号では「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、アリスの暗号署名を含むロック解除スクリプト $\langle Sig\ P_A \rangle$ をさらに含む。有効な署名を提供するためにアリスによって署名される必要があるデータ（または「メッセージ」）は、ロックスクリプトによって、またはノードプロトコルによって、またはこれらの組合せによって定義され得る。

10

【0051】

新しいトランザクション Tx_1 がブロックチェーンノード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、ロックスクリプトおよびロック解除スクリプトを一緒に実行して、ロック解除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つまたは複数の基準を含み得る）を満たす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つのスクリプトを連結することを含む：

$\langle Sig\ P_A \rangle \quad \langle P_A \rangle \quad | \quad [Checksig\ P_A]$

ここで、「 $|$ 」は連結を表し、「 $\langle \dots \rangle$ 」はデータをスタックに置くことを意味し、「 $[\dots]$ 」はロックスクリプト（この例ではスタックベースの言語）で構成される関数である。同等に、スクリプトは、スクリプトを連結するのではなく、共通スタックを用いて次々に実行され得る。いずれにしても、一緒に実行されると、スクリプトは、 Tx_0 の出力内のロックスクリプトに含まれるようなアリスの公開鍵 P_A を使用して、 Tx_1 の入力内のロック解除スクリプトが、データの予想される部分に署名するアリスの署名を含むことを認証する。この認証を実行するためには、データの予想される部分自体（「メッセージ」）も含まれる必要がある。実施形態では、署名されるデータは Tx_1 の全体を含む（つまり、平文のデータの署名された部分を指定する別個の要素は、すでに本質的に存在するので、含まれる必要がない）。

20

【0052】

公開－秘密暗号法による認証の詳細は、当業者によく知られている。基本的に、アリスが自身の私有鍵を使用してメッセージに署名した場合、アリスの公開鍵および平文のメッセージが与えられると、ノード104などの別のエンティティは、メッセージがアリスによって署名されたものに違いないことを認証することができる。署名は、典型的には、メッセージをハッシュし、ハッシュに署名し、これを署名としてメッセージにタグ付けすることを含み、これにより、公開鍵の任意の保持者が署名を認証することができるようになる。したがって、データの特定の部分またはトランザクションの一部などに署名することへの本明細書におけるいかなる参照も、実施形態では、データのその部分またはトランザクションの一部のハッシュに署名することを意味し得ることに留意されたい。

30

【0053】

Tx_1 内のロック解除スクリプトが、 Tx_0 のロックスクリプト内で指定されている1つまたは複数の条件を満たす場合（つまり、図示の例では、アリスの署名が Tx_1 内で提供され、認証された場合）、ブロックチェーンノード104は、 Tx_1 が有効であるとみなす。これは、ブロックチェーンノード104が、保留中のトランザクションの順序付きプール154に Tx_1 を追加することとなることを意味する。ブロックチェーンノード104はまた、トランザクション Tx_1 をネットワーク106内の1つまたは複数の他のブロックチェーンノード104にフォワードして、トランザクション Tx_1 がネットワーク106全体に伝搬されるようにする。 Tx_1 が妥当性確認されてブロックチェーン150に含まれると、これは、 Tx_0 からの $UTXO_0$ を使用済みとして定義する。 Tx_1 は、未使用トランザクション出力203を使用する場合にのみ有効になり得ることに留意されたい。別のトランザクション152によってすでに使用された出力を使用しようとする場

40

50

合、 Tx_1 は、他のすべての条件が満たされたとしても無効になる。したがって、ブロックチェーンノード104はまた、先行するトランザクション Tx_0 内の参照されたUTXOがすでに使用済みであるかどうか（すなわち、それが別の有効なトランザクションへの有効な入力をすでに形成したかどうか）をチェックする必要がある。これは、ブロックチェーン150がトランザクション152に定義された順序を課することが重要である理由の1つである。実際には、所与のブロックチェーンノード104は、どのトランザクション152内のどのUTXO203が使用されたかをマーキングする別個のデータベースを維持し得るが、最終的には、UTXOが使用されたかどうかを定義するものは、ブロックチェーン150内の別の有効なトランザクションへの有効な入力をすでに形成しているかどうかである。

10

【0054】

所与のトランザクション152のすべての出力203において指定された総額が、そのすべての入力202によって指し示された総額よりも大きい場合、これは、ほとんどのトランザクションモデルにおいて無効性の別の根拠となる。そのため、そのようなトランザクションは、伝搬されることも、ブロック151に含まれることもない。

【0055】

UTXOベースのトランザクションモデルでは、所与のUTXOが全体として使用される必要があることに留意されたい。UTXOにおいて使用済みとして定義された額の一部は、別の一部が使用されている間、「残す」ことはできない。しかしながら、次のトランザクションの複数の出力間でUTXOからの額を分割することはできる。例えば、 Tx_0 内のUTXO₀において定義された額を、 Tx_1 内の複数のUTXO間で分割することができる。したがって、アリスが、UTXO₀において定義された額のすべてをボブに与えたくない場合、アリスは、リマインダを使用して、 Tx_1 の第2の出力において自分自身に残りを与えるか、または別の当事者に支払うことができる。

20

【0056】

実際には、アリスはまた、通常、アリスのトランザクション104をブロック151に成功裏に含めるビットコインノード104に対する手数料を含める必要がある。アリスがそのような手数料を含めない場合、 Tx_0 は、ブロックチェーンノード104によって拒否され得、したがって、技術的に有効であっても、伝搬されず、ブロックチェーン150に含まれない可能性がある（ノードプロトコルは、ブロックチェーンノード104が望まない場合にトランザクション152を受け入れることを強制しない）。いくつかのプロトコルでは、トランザクション手数料は、それ自体の別個の出力203を必要としない（すなわち、別個のUTXOを必要としない）。代わりに、所与のトランザクション152の入力（複数可）202によって指し示される総額と出力（複数可）203で指定されている総額との間の任意の差が、トランザクションを公開するブロックチェーンノード104に自動的に与えられる。例えば、UTXO₀へのポイントが Tx_1 への唯一の入力であり、 Tx_1 は唯一の出力UTXO₁を有するとする。UTXO₀において指定されたデジタル資産の額がUTXO₁において指定された額よりも大きい場合、その差分は、UTXO₁を含むブロックを作成するためのプルーフオブワーク競争に勝つノード104によって割り当てられ得る（または、使用され得る）。しかしながら、代替的にまたは追加的に、トランザクション手数料がトランザクション152のUTXO203のうちのそれ自体の1つにおいて明示的に指定され得ることは必ずしも除外されない。

30

40

【0057】

アリスおよびボブのデジタル資産は、ブロックチェーン150内のどこかで任意のトランザクション152においてそれらにロックされたUTXOから構成される。したがって、典型的には、所与の当事者103の資産は、ブロックチェーン150全体にわたる様々なトランザクション152のUTXO全体に散在している。ブロックチェーン150内のどこにも、所与の当事者103の総残高を定義する数字は記憶されない。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は、それぞれの当事者にロックされ、別の以降のトランザクションでまだ使用されていない様々なUTXOのすべての値を一

50

緒に照合することである。これは、ビットコインノード 104 のいずれかに記憶されたブロックチェーン 150 のコピーにクエリを行うことによって行うことができる。

【0058】

スクリプトコードは、多くの場合、概略的に（すなわち、正確な言語を使用せずに）表されることに留意されたい。例えば、特定の機能を表すためにオペレーションコード（オペコード）が使用され得る。「OP_...」は、スクリプト言語の特定のオペコードを指す。例として、OP_RETURN は、ロックスクリプトの最初に OP_FALSE が先行するときに、トランザクション内にデータを記憶することができ、それによってデータをブロックチェーン 150 内に不変的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な出力を作成するスクリプト言語のオペコードである。例えば、データは、ブロックチェーンに記憶することが望まれる文書を含み得る。

10

【0059】

典型的には、トランザクションの入力は、公開鍵 P_A に対応するデジタル署名を含む。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、データの特定の一部分に署名する。いくつかの実施形態では、所与のトランザクションについて、署名は、トランザクション入力の一部、およびトランザクション出力の一部または全部に署名する。署名された出力の特定の部分は、SIGHASH フラグに依存する。SIGHASH フラグは、通常、どの出力が署名されるかを選択するために署名の最後に含まれる 4 バイトコードである（したがって、署名時に固定される）。

【0060】

20

ロックスクリプトは、典型的には、それぞれのトランザクションがロックされる当事者の公開鍵を含むという事実を指して、「scriptPubKey」と呼ばれることがある。ロック解除スクリプトは、典型的には、それが対応する署名を供給するという事実を指して「scriptSig」と呼ばれることがある。しかしながら、より一般的には、UTXO が償還されるための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 のすべ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、スクリプト言語を使用して、任意の 1 つまたは複数の条件を定義することができる。したがって、より一般的な用語「ロックスクリプト」および「ロック解除スクリプト」が好まれ得る。

【0061】

30

3. サイドチャネル

図 1 に示すように、アリスおよびボブのコンピュータ機器 102 a、120 b の各々上のクライアントアプリケーションは、それぞれ、追加の通信機能を含み得る。この追加の機能により、（いずれかの当事者または第三者の扇動で）アリス 103 a は、ボブ 103 b と別個のサイドチャネル 107 を確立することができる。サイドチャネル 107 は、ブロックチェーンネットワークとは別でのデータの交換を可能にする。そのような通信は、「オフチェーン」通信と呼ばれることがある。例えば、これは、当事者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（まだ）ブロックチェーンネットワーク 106 に登録されることなく、またはチェーン 150 上に進むことなく、アリスとボブとの間でトランザクション 152 を交換するために使用され得る。このようにトランザクションを共有することは、「トランザクションテンプレート」の共有と呼ばれることがある。トランザクションテンプレートは、完全なトランザクションを形成するために必要とされる 1 つまたは複数の入力および／または出力を欠いていてもよい。代替的にまたは追加的に、サイドチャネル 107 は、鍵、交渉された額または条件、データコンテンツなどの任意の他のトランザクション関連データを交換するために使用され得る。

40

【0062】

サイドチャネル 107 は、ブロックチェーンネットワーク 106 と同じパケット交換ネットワーク 101 を介して確立され得る。代替的にまたは追加的に、サイドチャネル 301 は、モバイルセルラーネットワークなどの異なるネットワーク、またはローカルワイヤレスネットワークなどのローカルエリアネットワーク、またはアリスのデバイス 102 a

50

とボブのデバイス102bとの間の直接のワイヤードまたはワイヤレスリンクを介して確立され得る。一般に、本明細書のどこでも、参照されるサイドチャンネル107は、「オフチェーン」すなわちブロックチェーンネットワーク106とは別でデータを交換するための1つまたは複数のネットワーキング技術または通信媒体を介した任意の1つまたは複数のリンクを含み得る。2つ以上のリンクが使用される場合、全体としてのオフチェーンリンクの束または集合は、サイドチャンネル107と呼ばれることがある。したがって、アリスおよびボブがサイドチャンネル107上で情報またはデータの特定の部分などを交換すると言われている場合、これは、これらのデータの部分のすべてが全く同じリンクまたは同じタイプのネットワーク上で送信されなければならないことを必ずしも意味するものではないことに留意されたい。

10

【0063】

4. トランザクションコンテンツの編集

ブロックチェーントランザクションは、入力および出力を含み、その各々がスクリプトを含む。ブロックチェーンスクリプト(Pay-to-Public-Key(P2PK)、Pay-to-Public-Key-Hash(P2PKH)、マルチ署名、OP_RETURNなど)を使用して、トランザクションに任意のデータを記憶することができる。出力は、複雑なロック条件を符号化することができるスクリプトを含み、加えて、ロック条件の一部であってもなくてもよい追加のデータ項目を記憶する。特定の例では、OP_FALSE OP_RETURN出力として知られる使用不可能なスクリプトパターンを使用して、任意のデータがトランザクションに追加され得る。台帳上に任意のデータを記憶するために使用されるスクリプトオペコードの組合せOP_FALSE OP_RETURNは、ブロックチェーンのサイズの増加を招き、ブロックチェーンの完全なデータを記憶することを望むブロックチェーンノード104のディスク空間を消費する。ブロックチェーン上に記憶されるデータの中には、不正、違法、機密、またはプライベートなものなどがあり得る。データの中には、ブロックチェーンに記録された時点で、または後日、これらのカテゴリのうちの1つに分類されるものがあり得る。データの中には、一部の管轄区域ではこれらのカテゴリの1つに分類されるが、他の管轄区域では分類されないものがあり得る。したがって、他の理由の中でもとりわけ、トランザクションのサイズを減少させ、および／または(管轄区域に依存する)ルールおよび規制に準拠するためには、トランザクションからのデータを編集することができる必要がある。図3は、OP_FALSE OP_RETURNペイロードを有する例示的なブロックチェーントランザクションの構造を示す。

20

30

【0064】

本開示の実施形態は、ブロックチェーントランザクションに記憶されたデータ、すなわち、トランザクションの入力または出力にスクリプトの一部として含まれるデータの編集を可能にする。編集対象のデータは、本明細書では「ターゲットデータ」と呼ばれる。ターゲットデータは、入力(ロック解除)スクリプトの一部として含まれてもよい。例えば、図3に示されるトランザクションは、データ<Sig_A> <P_A>(署名と公開鍵)を含むロック解除スクリプトを含む。ターゲットデータは、出力(ロック)スクリプトの一部として含まれてもよい。例えば、同じトランザクションが2つのロックスクリプトを含み、その両方がデータを含む。第1のロックスクリプトは、データP_B(公開鍵)を含む。第2のロックスクリプトは、データ<data>(メディアコンテンツ、個人の詳細(例えば、名前および住所)、請求書情報などの任意のデータ)を含む。ターゲットデータは、使用可能な(すなわち、転送可能および割当て可能な)出力または使用不可能な(すなわち、転送不可能および割当て不可能な)出力の一部として含まれてもよい。

40

【0065】

ターゲットデータの編集は、データを含む任意のトランザクションであり得る元のブロックチェーントランザクションへのアクセスを有する任意のエンティティによって実行され得る。簡潔にするために、編集プロセスを実行するエンティティを「リダクタ」と呼ぶ。いくつかの実施形態では、リダクタは、ユーザ、例えばアリス103aまたはボブ103bなどの当事者、または、むしろそれぞれのコンピュータ機器102a、102bであ

50

り得る。他の実施形態では、リダクタはブロックチェーンノード104を含む。これらの実施形態について以下に説明する。

【0066】

リダクタは、ターゲットトランザクション、すなわち、編集対象のターゲットデータをスクリプトの一部として含むトランザクションを取得する。リダクタは、ターゲットトランザクションを生成してもよい。例えば、リダクタは、トランザクションを生成するユーザ（例えば、アリス103a）であり得る。この場合、リダクタがターゲットデータも生成してもよいし、ターゲットデータは他の場所から取得（例えば、受信）されてもよい。他の例では、リダクタは、ターゲットトランザクションを生成したエンティティでなくともよい。例えば、ターゲットトランザクションは、ブロックチェーンから取得されてもよいし、ブロックチェーンノード104からもしくはトランザクションジェネレータから受信されてよい。例えば、リダクタは、ターゲットトランザクションを生成したアリス103aからターゲットトランザクションを受信するブロックチェーンノード104であり得る。

10

【0067】

ターゲットトランザクションを取得すると、リダクタは、ターゲットデータを含むスクリプト（例えば、ロック解除スクリプトまたはロックスクリプト）を使用してマークルツリーを生成する。スクリプト全体を使用して、マークルツリーを生成する。ターゲットデータは、その生の形式で、マークルツリーの1つまたは複数のリーフを形成する。すなわち、マークルツリーの1つまたは複数のリーフは、ターゲットデータの少なくとも一部を含む（またはターゲットデータの少なくとも一部から構成される）。いくつかの例では、単一のリーフがターゲットデータを含み得（またはターゲットデータから構成され得）、これは、ターゲットデータ全体がそのリーフの一部として含まれることを意味する。代替的に、ターゲットデータは、マークルツリーの複数のリーフにわたって分割されてもよく、これは、異なるリーフがターゲットデータの異なる部分を含むことを意味する。例えば、ターゲットデータは、等しいサイズのチャンクに分割されてもよいし、ターゲットデータは、所定数のチャンクに分割されてもよい。それどころか、ターゲットデータは、任意に分割されてもよい。

20

【0068】

「マークルツリー（Merkle tree）」という用語は一般にバイナリハッシュツリーを指すことに留意されたい。しかしながら、概して、本開示の実施形態は、任意のタイプのハッシュツリー、例えば三次ハッシュツリーを利用し得る。したがって、「マークルツリー」への任意の参照は、代わりに、より一般的な「ハッシュツリー」に置き換えられてもよい。

30

【0069】

いくつかの例では、スクリプトはターゲットデータのみを含み得る。例えば、ロック解除スクリプトは、前のロックスクリプトのロック解除スクリプトをロック解除するために使用されたターゲットデータのみを含み得る。その場合、マークルツリーは、ターゲットデータに完全に基つき得る。しかしながら、実際には、スクリプトは、ターゲットデータだけでなく、他の構成要素を含む可能性が高い。例えば、スクリプトは、それぞれの演算を実行するように構成された1つまたは複数の関数を含み得る。ビットコインブロックチェーン上では、これらの関数はオペコードとして知られている。追加的または代替的に、スクリプトは、編集対象ではないデータを含んでもよい。例えば、スクリプトは、編集される必要のない公開鍵ハッシュと、編集される必要のある画像ファイルとを含み得る。これらの例では、ターゲットデータは、マークルツリーの一部、すなわち、マークルツリーの1つまたは複数のリーフのみを形成する。関数（複数可）および／または他のデータもマークルツリーの一部を形成する。すなわち、マークルツリーの1つまたは複数のリーフは、1つまたは複数の関数を含み得（例えば、1つまたは複数の関数から構成され得）、マークルツリーの1つまたは複数のリーフは、非ターゲットデータを含み得る（例えば、非ターゲットデータから構成され得る）。

40

50

【0070】

スクリプトの異なる構成要素（例えば、関数、ターゲットデータ、他のデータなど）は、例えば左から右への順序を有し、その順序は、マークルツリーを構築する際に尊重される。マークルツリーのリーフは、スクリプトの構成要素を使用して、それらが現れる順序で形成される。例えば、スクリプトは、1つまたは複数の関数の第1のセットと、その後続くターゲットデータと、その後続く1つまたは複数の関数の第2のセットとを含み得る。マークルツリーのリーフの第1のセットは、関数の第1のセットから形成され、マークルツリーのリーフの第2のセットは、ターゲットデータから形成され、マークルツリーのリーフの第3のセットは、関数の第2のセットから形成される。

【0071】

いくつかの例では、1つまたは複数の連続する関数はグループ化され、マークルツリーの単一のリーフを形成し得る。他の例では、各関数がマークルツリーの単一のリーフを形成し得る。

【0072】

いくつかの例では、スクリプトは、実行されたときにターゲットデータを示すように構成されたデータ関数（例えば、OP_PUSHDATA、OP_RETURNなど）を含み得る。ターゲットデータを示すことは、実行されたときにターゲットデータをメモリ（例えば、スタックベースのメモリ）に出力することを含み得る。これらの例では、データ関数は、マークルツリーのそれぞれのリーフに含まれる唯一の関数となるように、他の関数から分離され得る。データ関数を含むリーフは、例えばバイト単位でターゲットデータの長さなどの他のデータを含み得る。代替的に、ターゲットデータの長さは、マークルツリーの異なるリーフとして含まれてもよい。データの長さを使用して、マークルツリーのリーフの数を決定し得る。

【0073】

以下でより詳細に説明する図4および図5は、編集対象のターゲットデータを含むスクリプトに基づいてマークルツリーを生成するプロセスを示す。図4に示されるように、左から右に、関数（オペコード）の第1のグループは第1のリーフを形成し、データ関数およびターゲットデータ（〈pubkeyhash〉）が第2のリーフを形成し、関数の第2のグループが第3のリーフを形成し、第3のリーフを複製して第4のリーフが形成される。図5では、第1のデータ関数（OP_RETURN）が第1のリーフの一部を形成し、第2のデータ関数（OP_PUSHDATA）がデータ長（[X-bytes]）とともに第2のリーフの一部を形成し、ターゲットデータは6つのリーフノードにわたって分割される。

【0074】

スクリプトに基づいてマークルツリーが生成されると、スクリプトをマークルツリーのマークルルートで置き換えることによってターゲットトランザクションの編集済みバージョンが生成される。したがって、ターゲットデータは、マークルルートを使用して符号化される（そして隠される）。

【0075】

ターゲットトランザクションは、それぞれのターゲットデータを含む複数のそれぞれのスクリプトを含み得る。スクリプトに基づいてマークルツリーを生成する上記で説明したプロセスは、それぞれのスクリプトのうちの1つ、いくつか、またはすべてについて実行され得る。それぞれのスクリプトは、それぞれのマークルツリーのそれぞれのマークルルートに置き換えられてもよい。

【0076】

リダクタは、編集済みトランザクションを1つまたは複数のエンティティ、例えばユーザ103またはブロックチェーンノード104に送信し得る。例えば、リダクタは、ターゲットトランザクション（必ずしもその編集済みバージョンではない）についての要求をユーザ（例えば、ボブ103b）から受信するブロックチェーンノード104であってもよい。

【0077】

10

20

30

40

50

リダクタは、編集済みトランザクションの一次トランザクション識別子を生成し得る。一部のブロックチェーン上では、一次トランザクション識別子（TxID）は、シリアル化されたトランザクション（例えば、32ビットVersion、32ビットLockTime、トランザクション入力のリスト、トランザクション出力のリストなど）のダブルSHA256ハッシュ（またはSHA256d=SHA256（SHA256（.）））として生成される。編集済みトランザクションの一次トランザクション識別子TxID'は、編集済みトランザクションをハッシュすることによって、同様の方法で生成されてもよい。一次トランザクション識別子TxID'を生成するために使用されるハッシュ関数は、SHA256またはダブルSHA256ハッシュ関数であり得る。他のハッシュ関数、例えばSHA512が代わりに使用されてもよい。

10

【0078】

リダクタがトランザクションジェネレータ（例えば、ユーザ、ウォレットアプリケーションなど）である場合、リダクタは、編集済みトランザクションに基づいて署名を生成してもよい。具体的には、署名はメッセージに基づき（すなわち、メッセージの関数であり）、メッセージは編集済みトランザクションに基づく。例えば、メッセージは、トランザクションのシリアル化されたバージョンであってもよい。通常、署名は、トランザクションのスクリプトに基づく（それらのスクリプトが生じる形式であるかハッシュされているかにかかわらず）。ここで、署名は、1つまたは複数のスクリプトを置き換えるマークルルート（複数可）に基づく。署名は、編集済みトランザクションの1つまたは複数の入力および／または出力に署名し、編集済みトランザクションの入力に含まれる。署名は、1つまたは複数の追加の項目、例えば私有鍵の関数であってもよい。例えば、署名はECDSA署名であってもよい。別の署名方式を使用してもよい。編集済みトランザクション、署名は、ターゲットトランザクション（すなわち、トランザクションの元の、未編集バージョン）に含まれ、妥当性確認のためにブロックチェーンネットワーク106にサブミットされる。すなわち、編集済みトランザクションは、1つまたは複数のスクリプトをそれぞれのマークルルートで置き換えることによって生成され、署名は、編集済みトランザクションに基づいて生成され、トランザクションの元のバージョンに含まれ、次いで、署名された元のトランザクションがブロックチェーンネットワーク106に送信される。

20

【0079】

上述したように、署名は、ターゲットトランザクションの入力に含まれる。入力、前のトランザクションの出力（「前の出力」）を参照する。出力は、ターゲットデータを含むスクリプトを含み得る。通常、署名されるべきメッセージの一部は、前の出力に基づく。ここで、メッセージを構築する際、リダクタは、（上記で説明した方法を使用して）前の出力のスクリプトに基づいてマークルツリーを構築し、スクリプトをマークルルートで置き換える。それにより、署名されるべきメッセージは、前の出力のスクリプトではなく、マークルルートに基づく。

30

【0080】

リダクタがトランザクションバリデータ（例えばブロックチェーンノード104）である場合、トランザクションバリデータは、署名を含むターゲットトランザクションを取得し、編集済みトランザクションに基づいて署名がメッセージに対して有効であることを検証し得る。メッセージは、上記で説明した方法と同じ方法で生成される。

40

【0081】

ターゲットトランザクションのスクリプトのうちの少なくとも1つは、ロックスクリプト（第1のロックスクリプト）であり得る。ここで、「第1の」は、ラベルとして使用され、必ずしもロックスクリプトのリスト内の第1のロックスクリプトを意味するわけではない。リダクタがトランザクションバリデータである場合、リダクタは、第1のロックスクリプトに基づいて生成されたマークルルートを、ターゲットトランザクションの修正された一次トランザクション識別子にマッピングされたデータベースに記憶し得る。例えば、修正された一次トランザクション識別子は、鍵値ペア（key-value pair）の鍵であってもよく、第1のロックスクリプトのマークルルートが値である。ターゲットトランザク

50

クションが複数の出力を有する場合、第1のロックスクリプトが属する出力のインデックスは、鍵の一部を形成し得る。リダクタは、複数の鍵値ペアを記憶し得、各鍵は、それぞれのトランザクションのそれぞれの出力に関連する。データベースは、後述するように、トランザクションを妥当性確認するために使用され得る。

【0082】

トランザクションバリデータは、ターゲットトランザクションの出力（例えば第1のロックスクリプトを含む第1の出力）を参照する入力（第1の入力）を含む第2のブロックチェーントランザクションを取得し得る。ここでも、「第1の」は、単に、ターゲットトランザクションの出力を参照する入力のラベルとして使用される。第2のトランザクションの第1の入力は、第1のロック解除スクリプトを含む。ターゲットトランザクションの第1の出力は、ターゲットトランザクションの修正された一次トランザクション識別子と第1の出力のインデックスとを使用して参照される。トランザクションバリデータは、いくつかの方法のうちの1つで第2のトランザクションを妥当性確認し得る。

10

【0083】

第1のオプションとして、トランザクションバリデータは、参照された出力にマッピングされたマークルートをデータベースから取得し、第1のロック解除スクリプトに基づいて第1のロックスクリプトの部分ロックスクリプトを決定する。部分ロックスクリプトは、ロックスクリプトの全部ではなく一部、例えば、ターゲットデータまでのロックスクリプトの全部を含む。部分ロックスクリプトはまた、マークルルートに対応するマークルツリーの1つまたは複数のハッシュを含む。ハッシュは、リーフハッシュおよび／または内部ハッシュであり得る。各ハッシュは、ターゲットデータに直接的に（リーフハッシュの場合）または間接的に（内部ハッシュの場合）基づく。例えば、図3を参照すると、部分ロックスクリプトは、以下の形式をとることができる：OP_DUP OP_HASH160 H₁ H₂₃。トランザクションバリデータは、部分ロックスクリプトを生成してもよいし、他の場所、例えば、第2のトランザクションをブロックチェーンに提出したエンティティ、またはブロックチェーンノード104からそれを取得してもよい。

20

【0084】

トランザクションバリデータは、第1のロック解除スクリプトに含まれる情報（例えば、データおよび／または機能）を使用して、部分ロックスクリプトの形式、例えば、部分ロックスクリプトに必ず存在している（または少なくとも存在する可能性が高い）関数を決定する。部分ロックスクリプトは、第1のロック解除スクリプトの構文解析（syntactic analysis）に基づいて決定され得る。一例として、第1のロック解除スクリプトが署名とその後に続く公開鍵とを含む場合、トランザクションバリデータは、第1のロックスクリプトがpay-to-public-key-hashスクリプトを含むと仮定し得る。部分ロックスクリプトは、pay-to-public-key-hashスクリプトとして設定され得る。別の例として、第1のロック解除スクリプトは、部分ロックスクリプトの形式の指示を含み得る。トランザクションバリデータは包含証明（proof of inclusion）を実行して、部分ロックスクリプトがマークルルートに対応するマークルツリーに属することを確認し得る。部分ロックスクリプトがマークルツリーに属し、したがって第1のロックスクリプトの一部であることを確認すると、トランザクションバリデータは、部分ロックスクリプトに照らして第1のロック解除スクリプトを妥当性確認する。妥当性確認に失敗すると、第2のトランザクションは拒否される。

30

40

【0085】

第2のオプションとして、第1の出力のマークルルートは鍵値ペアの値として記憶するのではなく、トランザクションバリデータは、第1の出力の部分ロックスクリプトを、修正されたトランザクション識別子および第1の出力のインデックスにマッピングされた鍵値ペアの値として記憶し得る。上述したように、部分ロックスクリプトは、（ターゲットデータを含まない）生の形式の第1のロックスクリプトの一部と、第1のロックスクリプトに基づいて生成されたマークルツリーの1つまたは複数のリーフまたは内部ハッシュとを含む。例えば、部分ロックスクリプトは、関数および／または非ターゲットデータのセ

50

ットと、ターゲットデータのハッシュとを含み得る。ターゲットトランザクションの第1の出力を参照する第2のブロックチェーンを取得すると、トランザクションバリデータは、データベースから部分ロックスクリプトを取り出し、部分ロックスクリプトに照らして第1のロック解除スクリプトを妥当性確認する。言い換えると、残りの生の形式の第1のロックスクリプトは、第1のロック解除スクリプトと一緒に実行される。妥当性確認に失敗すると、第2のトランザクションは拒否される。

【0086】

第3のオプションとして、第1のオプションと第2のオプションとを組み合わせ得る。すなわち、トランザクションバリデータは、修正された一次トランザクション識別子およびターゲットトランザクションの第1の出力のインデックスにマッピングされたデータベースに、マークルルートおよび部分ロックスクリプトの両方を記憶し得る。第2のトランザクションを受信したことに応答して、トランザクションバリデータは、（例えば、第1のロック解除スクリプトの分析に基づいて）部分ロックスクリプトの生の部分を決定し、生の部分が実際に第1のロックスクリプトの一部であることを確認するために、データベースから取り出された部分ロックスクリプトを使用して、マークル包含証明（Merkle proof of inclusion）を実行し得る。次いで、第1のロック解除スクリプトに照らして部分ロックスクリプトの生の部分を妥当性確認する。妥当性確認に失敗すると、第2のトランザクションは拒否される。

【0087】

上記の説明は、編集済みトランザクションを生成するためにターゲットトランザクションの個々のスクリプトに基づいてマークルツリーを生成することに主に焦点を当てている。リダクタ（ユーザ103、ブロックチェーンノード104、または他のエンティティであってもよい）は、編集済みトランザクションに基づいてマークルツリーを生成し得る。すなわち、トランザクションレベルのマークルツリーが生成され、そのリーフは、編集済みトランザクションの部分から形成される。編集済みトランザクションのフィールドのうちの少なくとも1つがマークルルート、例えば図3または図4のマークルルートを含む例が図8に示されている。トランザクションマークルツリーのマークルルートは、編集済みトランザクションの第2のトランザクション識別子として機能する。編集済みトランザクションは、複数のフィールド、例えば、バージョン番号、ロック時間、1つまたは複数の入力、1つまたは複数の出力などを含む。1つまたは複数のリーフは、単一のそれぞれのフィールドの一部またはすべてを含み得る。1つまたは複数のリーフは、複数のそれぞれのフィールドの一部またはすべてを含み得る。

【0088】

したがって、本開示は、編集済みトランザクションの個々のデータフィールドの検証を可能にするためにマークルツリーを利用することができるさらなる方法を認識する。編集済みトランザクションは、マークルツリーを生成するためにデータフィールドのセットに分割（またはパース）され、すなわち、トランザクションの異なる部分が別個のデータフィールドとして識別される。マークルルートは、トランザクションの新規の二次識別子として機能する。例えば、軽量クライアントのみを操作している可能性がある、クエリを行うユーザは、完全なトランザクションデータを取得してハッシュする必要なく、トランザクションに（小さな）データフィールドが存在することを証明するために、二次識別子を使用して証明を実行することができる。

【0089】

5. マークル化されたスクリプトを用いた選択的開示

このセクションは、ブロックチェーンコンテンツを編集するために上記で説明した実施形態をどのようにして実装することができるかの例を提供する。

【0090】

このセクションでは、ブロックチェーンスクリプトのマークル化を利用するトランザクションデータを選択的に開示するために使用することができる新しい気候が提供される。大まかに言えば、マークルツリーは、スクリプトが入力または出力であり得るトランザク

10

20

30

40

50

ションにおける特定のスクリプトについて計算される。スクリプトは、ハッシュ関数（例えば、SHA256）を用いて連続オペコードの各シーケンスがハッシュされてリーフノードが作成されるように、左から右にパースされる。〈data push〉に遭遇したときはいつでも、データを同様にハッシュしてリーフノードを作成することができ、またはデータをn個のチャンクに分割し、各チャンクをハッシュしてn個のリーフノードを作成することができる。すべてのリーフノードが作成されると、スクリプトを表すマークルツリーを計算することができる。

【0091】

マークルツリーの特徴により、署名、セキュアなマルチパーティ計算、紛失通信（oblivious transfer）、およびゼロ知識証明のようなコストのかかる非対称演算ではなくハッシュ計算によって、情報の選択的開示を非常に効率的に達成することができる。

【0092】

5.1 マークル化されたスクリプト

諸実施形態では、トランザクション内のすべてのスクリプトは、「スクリプトマークルツリー」としてマークル化された形式で表される。この手法を要約すると、連続オペコードの各文字列をリーフノードとして扱い、各データプッシュをリーフノードとして扱い、マークルツリーのリーフノードを、それらがスクリプトに現れる順序にしたがって順序付けることによって、スクリプトマークルツリーがスクリプトから生成される、とすることができる。スクリプトscriptMerkelize(SCRIPT)をマークル化するための例示的なアルゴリズムを以下に説明する：

1. スクリプトを左から右にパースする：

a. 連続オペコードの各シーケンスがグループ化され、バイト単位の基礎となる生の表現がリーフとしてインデックス付けされる。

b. 〈Data〉=OP_PUSHDATA [X bytes] [Data]の形式の各データプッシュは、別々に扱われ、リーフとしてインデックス付けされ、ここで、OP_PUSHDATA XはOP_PUSHDATA1、OP_PUSHDATA2、OP_PUSHDATA4のうちの1つである。Xは、[X bytes]のサイズである。

c. リーフは、それらのインデックスにしたがって順序付けられ、最も左のリーフは、スクリプトの最も左の部分に対応する。

2. 各リーフをハッシュして、マークルツリーにおけるリーフノードを作成する。代替的に、各リーフのダブルハッシュ（SHA256d=SHA256（SHA256（x）））、ここでxはプリメッセージである）を代わりにとることができる。

3. マークルツリーTが構築され、リーフからルートRが計算され、ここで、通常の方法でパディングが行われる。

a. ツリーは常に偶数のリーフを有する。リーフが奇数である場合、最後のリーフが複製され得る。例えば、ツリー内の最後のリーフが左ブランチである場合、その最後のリーフが右リーフ位置に複製される。

【0093】

OP_PUSHDATA Xおよび[X bytes]は、それぞれ、セクション4において説明されたデータ関数およびデータ長の例である。データ長は、編集済みデータの長さを示すために使用される。

【0094】

上記で示したステップに加えて、特定のスクリプトにおける各データプッシュの異なる部分のより粒度の細かい選択的開示を可能にするために、スクリプトの〈Data〉要素をさらに分解することができる。例えば、データは、当然、より小さなブロックに分割することができる固定サイズのフレームのメディアコンテンツを含む。このシナリオでは、単に以下の追加部分を本発明の（our）scriptMerkelize（）アルゴリズムに追加することができる。

【0095】

〈data push〉に遭遇したときはいつでも、最初にn個のチャンクに分割し、各チャ

10

20

30

40

50

ンクのハッシュを計算して n 個のリーフを作成することができる。 n の値は固定とすることができ、スクリプトによって異なることもできる。一般に、チャンクの各々は、等しいサイズである必要はないことにも留意されたい。

【 0 0 9 6 】

下記データを分割しない場合、

【数 1】

$Leaf = OP_PUSHDATA [X\text{-Bytes}] DATA$

$OP_PUSHDATA [X\text{-bytes}]$ は、以下のように、 n 個のデータリーフからの別個のリーフとして $\langle Data \rangle$ から分離することもでき、ここで、1つの $\langle data\ push \rangle$ は、ツリー内で $n + 1$ 個のリーフになる。

【数 2】

$Leaf_0 = OP_PUSHDATA [X\text{-Bytes}]$

$Leaf_1 = Data[1]$

...

$Leaf_n = DATA[n]$

【 0 0 9 7 】

備考：データをより小さなチャンクに分割することにはトレードオフがあるが、ハッシュ関数圧縮およびマークルツリー計算の増加による計算コストがかかる。コンテンツデータが 4 4 8 0 ビットであるとする。SHA256(content data)は、9つの圧縮関数を必要とし、1つのリーフのみを有する。しかしながら、コンテンツデータを 4 4 8 ビットに分割する場合、10個のデータチャンクを有し、各チャンクは1つの圧縮関数を必要とし、したがって、合計で10回の圧縮を必要とする。第2のオプションもツリーの深さを増加させる。

【 0 0 9 8 】

このアルゴリズムは、任意の所与のスクリプトからマークルツリーを生み出す。このマークルツリーのルート R は、以下で説明される更新された $T \times ID$ 生成および $SIGHASH$ アルゴリズムにおいて使用され得る。次に、上記のスクリプトマークル化アルゴリズムをいくつかの例示的なスクリプトに適用することを考える。

【 0 0 9 9 】

5. 1. 1 実施例

実施例 1：Pay-to-Public-Key-Hash (P2PKH)

【 0 1 0 0 】

単純な「Pay-to-Public-Key-Hash」出力を考えると、そのスクリプトを次のように書くことができる：

$SCRIPT = OP_DUP\ OP_HASH160\ OP_PUSHDATA\ \langle pubkeyhash \rangle\ OP_EQUALVERIFY\ OP_CHECKSIG$

ここで、 $SCRIPT$ は次のようにリーフにチャンクされる：

【数 3】

$Leaf_0 = OP_DUP\ OP_HASH160$

$Leaf_1 = OP_PUSHDATA\ \langle pubkeyhash \rangle$

$Leaf_2 = OP_EQUALVERIFY\ OP_CHECKSIG$

$Leaf_3 = Leaf_2$

【 0 1 0 1 】

この例を図 4 に示す。

【0102】

最後のリーフは、ツリーの計算の一部として複製されることに留意されたい。また、スクリプトをマークル化しても、トランザクションからスクリプトが除去または「隠される」ことはないことに留意されたい。その代わり、「置き換える (replace)」という用語を使用するときに意味するのは、スクリプト自体ではなくマークルルートを使用するということである。ここで強調しておきたいのは、マークルツリーのリーフにおけるスクリプトおよびそのデータ要素の順序付けを、これを使用するトランザクションの以降の妥当性確認のため、保持する必要があるということである。ここで、「以降の妥当性確認 (onward validation)」とは、編集され、そのスクリプトの1つがマークル化で置き換えられたトランザクションの出力を使用する任意の将来のトランザクションの妥当性確認を指すことに留意されたい。

10

【0103】

ここでの微妙な点は、バリデータが、スクリプトの未編集部分を取得して実行することに加えて、完全性チェック (すなわち、マークル包含証明) を実行して、未編集のスクリプト部分が、部分的に編集される前の元のスクリプトの一部であったことを証明する必要もあり得ることである。このタイプの以降のトランザクション妥当性確認のための例示的なプロセスを以下に詳述する。

【0104】

可能な編集の例として、図4のP2PKH出力は、別のトランザクションにおいて使用されている (すなわち、割り当てられている)。後に、このP2PKH出力を編集することができる (例えば、Leaf₁のスクリプトが編集される)。H(Leaf₁)を有するので、編集済みトランザクションが有効であることが分かり、それにより、スクリプトのマークルルートと、すでにマイニングされたトランザクションに一致するTxIDとを計算することができる。さらに、pubkeyを取得し、手動でLeaf₁、OP_PUSHDATA pubkeyhashを再作成し、これをハッシュして、H(Leaf₁)に一致するハッシュ値を計算することによって、支出トランザクションのscriptSig内のpubkeyが有効であることを証明することさえできる。

20

【0105】

ここで強調しておきたいのは、これが、完全性を証明するためにゼロ知識ベースのプロトコルを呼び出し得る、同じ効果を達成する代替のより多くの方法と比較したときに、編集済みトランザクションの以降の使用 (onward spend) を効率的に妥当性確認することを可能にすることである。マークルツリー検証がゼロ知識証明を実行するよりも効率的である傾向があるという事実は、この方法により、編集済みトランザクションのより効率的な妥当性確認が可能になることを意味する。

30

【0106】

実施例2: OP_FALSE OP_RETURN

何らかのデータコンテンツがOP_RETURNオペコードを有するトランザクションに挿入されており、そこには何らかの機密情報 (例えば、個人情報または違法情報) が含まれており、ある部分を選択的に開示する一方で他の部分を完全に隠すことを必要とすると仮定する。以下とする:

40

SCRIPT=OP_FALSE OP_RETURN <DATA>

ここで、<DATA>=OP_PUSHDATAX [X-bytes] [Data]であり、[X-bytes]は[DATA]のバイト長である。

【0107】

次に、ツリーのリーフを構築し、各リーフ (左から右) は、オペコードのセットまたはデータプッシュのいずれかである。これは、スクリプト内のすべてのデータプッシュがリーフであり、オペコードの各クラスタもプッシュであることを意味する。例えば、SCRIPTは、以下のようにリーフにチャンク化され、データプッシュは、上記で説明したように粒度の細かいチャンクに分割される。

【数4】

50

$Leaf_0 = OP_FALSE\ OP_RETURN$
 $Leaf_1 = OP_PUSHDATA\ [X\text{-bytes}]$
 $Leaf_2 = Data_0$

...

$Leaf_{n+1} = Data_{n-1}$

ここで、 $[DATA] = [Data_0 || Data_1 || \dots Data_{n-1}]$ である。

【0108】

データは様々なサイズとすることができる。この技法を使用するシステムは、異なる基準に基づいて n を構成することができる。データのサイズが 4000 ビットであると仮定する。次いで、以下のルールのいずれかを使用してデータを分割することができる：

- ・ システムは n を固定し、事前に定義する：システムが $n = 6$ を使用すると仮定する。すると、各データリーフは、

【数5】

$\lfloor 4000/6 \rfloor = 666$ ビット

(最後のチャンクに加算される剰余を丸める) となる。これを SHA256 でハッシュするために、各データリーフ入力は、(512 ビットの 2 つのブロックを埋めるために) 1024 ビットにパディングされる。

- ・ システムはリーフに関連付けられたデータ量を制限する：システムがハッシュ関数として SHA256 を使用したとすると、データは、512 ビットのサイズの k 個のブロックから構成されるようにゼロビットでパディングされ得る [3]。データが 4000 ビットである場合、各チャンクが 512 ビットであるので、データをパディングする必要があり、圧縮するリーフノードは $n = \lfloor 4000 / 512 \rfloor = 8$ 個である。

【0109】

図5は、8つのリーフを有するマークル化されたスクリプトの例を示し、第1のリーフは、オペコード「OP_FALSE OP_RETURN」のシーケンスであり、残りのリーフは、6つのチャンクに分割されたデータである。

【0110】

5.2 セキュリティ分析

ハッシュ関数は、オフライン (ブルートフォース) 攻撃を受けやすく、その使用は、プレイメージが十分なエントロピーを有するかどうか依存する。トランザクション内に秘密データがある場合、ブルートフォース攻撃が可能にならないように、秘密部分が十分なエントロピーを有することを確実にすることが必要であり得る。より具体的には、敵対者は、難読化された (ハッシュされた、または暗号化された) メッセージから秘密を取り出すことができてはならない。ここでエントロピーとは、ランダムデータを必要とするアルゴリズムで使用するためにシステムによって収集されるランダム性を指す。良好なエントロピーの欠如は、セキュリティシステムを脆弱にし、データを安全に暗号化／難読化することができない。一般的な経験則として、128 ビットは、最も感度の高いアプリケーションを除くすべてに対して十分であると考えられる。

【0111】

以下では、各ケースを別々に考慮することによって、本発明の機構がどのように使用され得るかを分類する。

- ・ 秘密データコンテンツのエントロピーが 128 ビットよりも大きい場合：

— この秘密コンテンツの個々のチャンク (すなわち、マークルツリーのリーフ) も十分なエントロピーを有するかどうかをチェックする。プライバシーは、十分なエントロピーを有するまで隣接するデータと連結することによって依然として維持することができ、次いで、それらのブロックのマークルルートのみを開示する。

1. 例えば、 $Leaf_2$ が十分なエントロピーを有さないが、 $Leaf_1$ および $Leaf_3$ と組み合わせられたときに十分なエントロピーを有する場合、 H_{4567} および $Leaf_4, \dots, Leaf_N$ のみを開示する（すなわち、 $Leaf_0, Leaf_1, Leaf_2$ 、および $Leaf_3$ は共有されない）。

- ・ 秘密データコンテンツのエントロピーは128ビット未満であるが、データコンテンツ全体のエントロピーが128ビットよりも大きい場合：

- － ブロックされた（隠された）チャンクは、秘密データコンテンツのみを隠す場合、オフライン攻撃を受けやすい。

- － 以下のように依然としてプライバシーを維持することができる：

- 1. これらの128ビット（公開部分および秘密部分）を抽出し、それらを連結してエントロピーを増加させる、

- 2. この新しい値を秘密にする、

- 3. 公開部分の残りを公開のままにする、

- ・ スクリプト全体のエントロピーが128ビット未満である場合

- － それはオフライン攻撃を受けやすい。

- － それでも、完全を期すために、

トランザクションを妥当性確認するのに十分であるので、データのマークルルート以外のデータのいかなる部分も開示しない。

【0112】

5.3 マークル化されたスクリプトのレイヤ1適用

ここで、マークル化されたスクリプトの概念をBitcoin Coreなどのレイヤ1ブロックチェーンシステムの設計にどのように適用することができるかを考える。このセクションの目標は、ノード104が、編集されたものに関連する将来のトランザクションを検証する機能を犠牲にすることなく、ブロックチェーン150のコピーからトランザクション情報を編集することを可能にするために、どのようにスクリプトマークル化がビットコインのようなブロックチェーンのプロトコルレベルで採用され得るかを調査することである。

【0113】

前のセクションでは、所与のスクリプトについてどのようにマークルルートを生成するかを一般的な意味で説明した。次いで、このマークルルートは、S I G H A S HアルゴリズムまたはT x I D計算など、スクリプト自体をハッシュすることを典型的に含む既存の計算において、スクリプトの代わりに使用され得る。このセクションにおけるレイヤ1の提案は、T x I Dがトランザクションデータからどのように計算されるかに影響を及ぼすが、トランザクションメッセージ自体に含まれるものには影響を及ぼさない。後のセクションで説明するように、T x I Dの構築は、生のオペコードの代わりにスクリプトマークルルートを使用するが、トランザクション自体は、依然としてスクリプトを完全に含む。

【0114】

スクリプトのマークル化は、典型的に、元のトランザクションがブロードキャストされた後で、データの一部を編集する必要があるときに行われることに留意されたい。

【0115】

本方法は、既存のブロックチェーン実装の以下の態様への変更を必要とする：

- ・ トランザクションID（T x I D）構成
- ・ S I G H A S Hアルゴリズム
- ・ トランザクション妥当性確認

【0116】

これらの変更により、完全な元のトランザクションスクリプトデータがない場合でもトランザクションを妥当性確認することができる。これは、トランザクションがブロック151に含まれた後のある時点でトランザクションデータの一部が不正であると識別された場合に有用である。なぜなら、これにより、ノード104は、違法データを含むトランザクションアウトポイントの以降の使用を妥当性確認することができ（例えば、ノードをブ

10

20

30

40

50

ートストラップしている最中に編集済みスクリプト妥当性確認に失敗すると、トランザクションは、それがブロックチェーンにおいて十分に「深い」場合、有効であると仮定され得る)、エンドユーザは、トランザクション内の残りの非違法データの存在証明を実行することができるからである。

【0117】

図6は、トランザクションの例示的なフローを示す。ステップ1において、ターゲットトランザクションのスクリプトがマークル化される。トランザクションは、例えば、以下で説明する修正されたS I G H A S Hアルゴリズムを使用して署名され得る。ステップ2において、元のターゲットトランザクション(ステップ1で生成された署名を含み得る)がブロックチェーン150にサブミットされる。ステップ3において、トランザクションからのターゲットデータは、スクリプトをマークル化することによって編集される。

10

【0118】

5.3.1 TxID構築プロトコルの修正

ビットコインの現在の実装では、トランザクション識別子(TxID)は、生のトランザクションデータTxのSHA256dとして計算され、以下のように書くことができる:

【数6】

$$TxID := SHA256d(Tx)$$

20

【0119】

しかしながら、TxIDの計算には、完全な生のトランザクションデータが必要であるが、処理することが望ましくない可能性がある不正なまたは大きなデータを扱う場合にはこれを避けることが望まれる。マークル化されたスクリプトに対応するために、すべてのトランザクションのTxIDは、新しいアルゴリズムcomputeTxID()を使用して以下のように計算され得る:

1. トランザクションTxの完全なトランザクションデータを取得する。
2. 各スクリプトのスクリプトデータをブランクアウトして、Tx'と表される簡略化されたトランザクションを生み出す。
3. ブランクアウトされた各スクリプトの代わりに、scriptMerkelize(SCRIPT)アルゴリズムによって以前に計算された、そのスクリプトのマークルルートを配置する。その結果はTx''である。
4. TxID'' = SHA256d(Tx'')を計算する。

30

【0120】

これは、編集されたスクリプトデータに依存するTxIDを再構築するリスクなしに、トランザクションを一意に参照するために使用され得る新しい(一次)トランザクションIDを定義する。

【0121】

別の方法で説明すると、トランザクションは、スクリプトの一部が後で編集される場合であっても、同じTxIDを有する。

40

【0122】

この例ではSHA256ハッシュ関数を使用することを選択したことに留意されたい。これは、置き換えられるスクリプトのサイズにかかわらず、すべてのスクリプトが32バイトのハッシュダイジェストに置き換えられることを意味する。scriptMerkelize()アルゴリズムに対して異なるハッシュ関数が選択された場合、置き換えダイジェストの長さは異なり得る。

【0123】

5.3.2 S I G H A S Hアルゴリズムの修正

ビットコインの既存の実装は、ECDSA署名によって署名され、ロック解除スクリプトで使用されるべきメッセージへのビットコイントランザクションのシリアル化を定義す

50

る S I G H A S H アルゴリズムを含む。S I G H A S H アルゴリズムの重要な特徴は、特定のトランザクション入力に署名するためのシリアル化されたメッセージを生成するとき、そのメッセージが、その入力において消費されている前のロックスクリプトおよび前のアウトポイントも含むことを確実にすることである。これは、署名されるべきシリアル化されたメッセージが、前のロックスクリプトに明示的に依存することを意味する。重要なことに、この前のロックスクリプトが実際に前のトランザクションの一部であったことを確実にすることが必要であり、これは、前のロックスクリプトが実際に、ダブルハッシュが $T x I D_{prev}$ を生み出す生のトランザクション $T x_{prev}$ の一部であったことを検証することによって行うことができる。完全な S I G H A S H アルゴリズムは、データタイプが括弧に与えられて、次のように書かれる：

10

1. $T x_{current}$ 内の $nVersion$ (4 バイトのリトルエンディアン)

2. すべての入力アウトポイントのシリアル化の S H A 2 5 6 d (32 バイトハッシュ)

・ ANYONECANPAY フラグが設定されている場合、これは 32 バイトのゼロでなければならない。

3. すべての入力の $nSequence$ のシリアル化の S H A 2 5 6 d (32 バイトハッシュ)

・ ANYONECANPAY フラグが設定されている場合、これは 32 バイトのゼロでなければならない。

4. 使用されているアウトポイント (トランザクション ID の 32 バイト + インデックスの 4 バイトのリトルエンディアン)。

20

5. $subScript$ のバイト長 (ビッグエンディアン)。

6. $subScript$ (以下に定義)

7. サトシ単位での出力量 (8 バイトのリトルエンディアン)。

8. このアウトポイントの $nSequence$ (4 バイトのリトルエンディアン)

9. すべての出力量および $scriptPubKeys$ のシリアル化の S H A 2 5 6 d。これらは $T x_{current}$ における出力から取り出される。

・ S I N G L E フラグが設定されており、入力インデックスが出力の数より小さい場合、これは、入力と同じインデックスの $scriptPubKey$ を有する出力のダブル S H A 2 5 6 であるべきである。

30

・ N O N E フラグが設定されている場合、これは 32 バイトのゼロでなければならない。

10. トランザクション $T x_{current}$ の $nLocktime$ (4 バイトのリトルエンディアン)

11. 署名の sighash タイプ (4 バイトのリトルエンディアン)

【0124】

上記のアルゴリズムにおけるステップ 6 は、前のロックスクリプトおよび $prevOuts$ を使用して生成される $subScript$ に依存する。前のトランザクション間の関係は以下の通りである：

「新しい $subScript$ が $previousScriptPubKey$ から作成される。 $subScript$ は、最新の OP_CODESEPARATOR ([being executed] である OP_CHECKSIG の直前のもの) から開始し $previousScriptPubKey$ の終わりまでである。OP_CODESEPARATOR が存在しない場合、 $previousScriptPubKey$ が $subScript$ になる。

40

【0125】

S I G H A S H アルゴリズムの生の $previousScriptPubKey$ へのこの依存性は、使用されている前のロックスクリプトが、違法データなど、共有しないことを望むデータを含み得る場合に問題を提示する。したがって、編集された場合、使用することができなくなり、ステップ 6 は失敗する。

【0126】

編集済みスクリプトを用いた以降の妥当性確認の例：

50

Tx_1 の任意の出力を使用する任意の後続のトランザクション Tx_2 の妥当性確認を指すために、トランザクション T_1 の「以降の妥当性確認」または「以降の使用」という用語を使用する。例えば、 Tx_1 および Tx_2 をブロックチェーンにおいてすでにマイニングされたトランザクションとする：

【数 7】

Tx_1		Tx_2	
OUTPUT		INPUT	OUTPUT
LOCKING		UNLOCKING	LOCKING
ScriptPubKey -->		scriptSig -->	OP_RETURN CONTENT

10

【0127】

以下のシナリオは、既存のSIGHASHアルゴリズムが編集済みスクリプトデータの扱いにどのように失敗することになるかを強調している。

- Tx_2 を妥当性確認する、ここで、SIGHASHが、この時点では欠落データである編集済み出力scriptPubKeyをハッシュすることを含むので、 Tx_2 は後に編集される Tx_1 の出力を使用している。

- Tx_2 を妥当性確認する、ここで、SIGHASHが、この時点では欠落データであるOP_RETURN出力をハッシュすることを含むので、 Tx_2 は後に編集される出力に送信している。

20

【0128】

この問題は、SIGHASHアルゴリズムで使用されるpreviousScriptPubKeyを、scriptMerkelize(previousScriptPubKey)を使用して生成されるロックスクリプトのためのマークルルートで置き換えることによって解決される。これは、前のロックスクリプトのマークルルートのみを使用して、SIGHASHアルゴリズム全体を実行し、したがって署名を妥当性確認することができることを意味する。より一般的には、すべてのスクリプトは、それらに署名するときに、それらのscriptMerkelize()ルートで置き換えられ得る。これは、ECDSA署名によって署名された任意のスクリプトがメッセージ構築プロセス中にそのマークルルートの代わりとなるように、SIGHASHアルゴリズムの新しいバージョンで実施され得るルールである。

30

【0129】

5.3.3 トランザクションの以降の妥当性確認

上記で説明したようにトランザクションに署名するとき、SIGHASHアルゴリズムは（SIGHASHフラグに応じて）、典型的に、出力スクリプト（および他のデータ）をハッシュして、後で署名されるダイジェストにすることを伴う。本発明の新構築では、メッセージのための新しいシリアル化アルゴリズムが、署名する前に使用される（例えば、ECDSA）。違いは、SIGHASHアルゴリズムがスクリプト自体ではなくマークルルートを使用することである。これは、署名が署名した可能性のある編集済みトランザクションを受信しただけの当事者による署名の以降の妥当性確認を可能にする。この修正されたSIGHASHアルゴリズムは、通常の場合でのみ呼び出される、すなわち、OP_CHECKSIG、OP_CHECKSIGVERIFY、OP_CHECKMULTISIG、またはOP_CHECKMULTISIGVERIFY演算が呼び出されたときに呼び出されることに留意されたい。

40

【0130】

例えば、OP_RETURNデータが編集済みである、OP_FALSE OP_RETURN <data>を含む出力を有するトランザクションのケースを考える。元のビットコイントランザクションでは、ハッシュするためにUTXOの出力スクリプトを必要とするが、編集済みトランザクションでは、このデータが利用不可能であり、それを再現することはできない。したがって、署名されたダイジェストを再現することはできない。本発明の新構築では、署名のためのハッシュプロセスの目的のために、トランザクションを妥当性確認しなければならない受信側の当事者にスクリプトのマークルルートが提供された場合、以降の妥

50

当性確認が可能となる。トランザクション内の署名は、同じ S I G H A S H アルゴリズムを実行してダイジェストを生成することによって検証することができる。

【0131】

S I G H A S H アルゴリズムに対するこの提案された修正は、E C D S A 署名の作成および妥当性確認がスクリプトの編集によって妨げられないことを確実にするように特に設計されることに留意されたい。本発明の修正は、すべての署名について、これらの署名が編集済み出力を使用しようと試みるかどうかにかかわらず、これを達成する。

【0132】

次のセクションでは、1つのトランザクションの一部の編集が将来のトランザクションがその出力を使用することを妨げないように、このシナリオにおいてトランザクション妥当性確認を実行するのに必要なステップを説明する。

【0133】

5.3.4 更新されたトランザクション妥当性確認

前のセクションでは、S I G H A S H アルゴリズムが生形式の入力の前のロックスクリプトに依存しないように、どのようにして S I G H A S H アルゴリズムを更新することができるかを示した。これにより、完全な前の生のロックスクリプトを知ることなく署名を検証することができる。しかしながら、トランザクションを妥当性確認するとき、ロック解除スクリプト内の E C D S A 署名（複数可）を検証することは、プロセスの一部分にすぎない。支出トランザクションが、部分的に編集されたロックスクリプトを使用しようと試みるとき、支出トランザクションによって満たされている前のロックスクリプトの部分が実際に元のロックスクリプトの一部であったことを確認することも必要である。これは、前のロックスクリプトを前の $TxID_{prev}$ にマッピングすることによって行うことができ、これは、マークル化されたスクリプトも符号化する。

【0134】

これを行うための1つの方法は、形式 $(TxID_{prev}, i_{prev})$ のアウトポイントを
【数8】

$$(TxID_{prev}, i_{prev}) \mapsto [Locking\ script]$$

としてロックスクリプトにマッピングすることである。次に、 $[Locking\ script]$ を使用して、使用されている部分スクリプトを取り出し、スクリプトマークル化を使用して、使用されている部分が実際に完全なスクリプトの一部であったことを、メモリからそれを取り出すだけでなく、証明することができる。例えば、 $[Checksig\ P_A] \langle Data \rangle OP_DROP$ という形式のスクリプトの場合、 $\langle Data \rangle$ 構成要素を知らずに出力を使用したいと望む可能性がある。これは、 $\langle Data \rangle$ 構成要素にアクセスすることなく、 $[Checksig\ P_A]$ 構成要素が実際に完全な元のロックスクリプトの一部であったことを確認する必要があることを意味する。

【0135】

既存のシステムでは、UTXO データベースは、 $(TxID_{prev}, i_{prev}) : [Locking\ script]$ という形式の鍵値ストアのように振る舞い、ここで、鍵は、既存のトランザクション識別子 $TxID_{prev}$ および出力インデックス i_{prev} から構成されるアウトポイントであり、アウトポイントに対して記憶される値は、それに対応するロックスクリプトである。この出力の将来の使用を妥当性確認するとき、ノードソフトウェアで実施されるプロセスの大まかなステートメントは、以下の通りである：

1. 支出トランザクション $Tx_{current}$ をパースし、その入力によって参照されるアウトポイント $TxID_{prev}, i_{prev}$ を取得する。

2. 取得された各アウトポイント j について、UTXO データベースから対応する $[Locking\ script]$ を取り出す。

3. $Tx_{current}$ の各入力 k について、その $[Unlocking\ script]$ を、対応する取り出された $[Locking\ script]$ に対して妥当性確認する。

上記のプロセスにおける第2のステップは、UTXOデータベースが正しく維持されており、したがって、前のロックスクリプトの完全性が健全であるという仮定の下で、前のロックスクリプトが参照されたアウトポイントに接続される場合である。

【0136】

ノード104が、前のロックスクリプト（複数可）の全体を記憶するのではなく、提案されたマークル化されたスクリプトシステムを実装することを望む場合、上記のプロセスを変更して、そのデータのすべてを知る必要なく、TxCurrentにおいて使用されている部分ロックスクリプトの完全性を確実にすることができる。部分ロックスクリプトは、いくつかのデータリーフを提供し、他のデータリーフを隠し、場合によっては、隠されたデータリーフのハッシュのみを含み得る。例えば、前のロックスクリプトがOP_CHECKSIG OP_FALSE OP_RETURN 〈data〉である場合、この完全なスクリプトを記憶する代わりに、ノード104は、OP_CHECKSIGと、このスクリプトのマークル化された形式のOP_RETURNデータリーフに対する対応するリーフハッシュとを記憶するのみでよい。

10

【0137】

この目的のためにUTXOデータベース実装を変更しようとする方法は、大まかに3つあり、それらは、1）マークル化されたスクリプトルート（複数可）の記憶、2）所望される部分スクリプト、または3）両者の組合せである。

【0138】

5.3.4.1 マークルルートの記憶

20

UTXOデータベースがスクリプトマークルルートを記憶する場合、それは、(TxIDprev, iprev) : [Merkle root]の形式の鍵値ストアのように動作し、ここで、[Merkle root]は、scriptMerkelize()アルゴリズムを[Locking script]に適用した結果のルートRである。

【0139】

次いで、トランザクション妥当性確認プロセスは、TxCurrentにおいて消費されている前のロックスクリプト（複数可）の完全性を確実にするために、次のように変更され得る。

1. 支出トランザクションTxCurrentをパースし、その入力によって参照されるアウトポイント(TxIDprev, iprev)を取得する。

30

2. 取得された各アウトポイントjについて：

2.1 UTXOデータベースから対応する[Merkle root]を取り出す。

2.2 TxCurrentの[Unlocking script]をパースして、トランザクションによって使用されている[Partial Locking Script]を決定する。

2.3 マークル包含証明を実行して、[Partial Locking Script]が[Merkle Root]の下集合のメンバであることを確認する。

3. TxCurrentの各入力kについて、その[Unlocking script]を、取り出されかつ完全性チェックされた対応する[Partial Locking script]に対して妥当性確認する。

【0140】

40

ステップ2.2において、前の部分ロック解除スクリプトを決定するために構文解析を使用したことに留意されたい。しかしながら、これは、例えば、TxCurrentの入力に、部分スクリプトが満たされていることの明示的な指示が含まれる場合、いくつかの方法で実装において置き換えられ得る。この部分スクリプトは、トランザクションサブミッタ(submitter)とバリデータとの間のメッセージングプロトコルの一部として配信することもできる。

【0141】

5.3.4.2 部分スクリプトの記憶

UTXOデータベースが部分スクリプトを記憶する場合、UTXOデータベースは、「TxIDprev, Indexprev : [Partial locking script]」という形式の鍵値ストアのよ

50

うに動作する。ここで、[Partial locking script]は、他のスクリプトデータがブルーニングおよび除去される際にビットコインノードが保持することを選択するロックスクリプトの部分（複数可）である。

【0142】

次いで、トランザクション妥当性確認プロセスは、TxCurrentにおいて消費されている前のロックスクリプト（複数可）の完全性を確実にするために、次のように変更され得る：

1. 支出トランザクションTxCurrentをパースし、その入力によって参照されるアウトポイントTxIDPrev,IndexPrevを取得する。

2. 取得された各アウトポイントjについて、UTXOデータベースから対応する[Partial locking script]を取り出す。

3. TxCurrentの各入力kについて、その[Unlocking script]を、対応する取り出された[Partial locking script]に対して妥当性確認する。

【0143】

このケースは、取り出されたスクリプトが完全な前のロックスクリプトの一部のみであることを除いて、既存の実装形態と同一である。ここでも、UTXOデータベースの、したがって[Partial locking script]の完全性は、ノード104がブロックチェーン1の全体の履歴を以前に処理したことによって保証されるという仮定を使用している。

【0144】

コンテンツをブロックするように求められたノードは、例えば、管轄区域、裁判所の命令、および選好に応じて異なるデータをブロックし得るので、異なる方法でブロックされた前のロックスクリプトの複数のバージョンが存在し得ることに留意されたい。UTXOデータベースが同じアウトポイントに対して複数の部分的な前のロックスクリプト、例えば「TxIDPrev,IndexPrev : [Partial script 1] [Partial script 2]」を記憶する場合、TxCurrent上の使用するロック解除スクリプトに対応するためにはどの部分スクリプトが取り出されるべきかを示す必要があり得る。例えば、トランザクションが複数の異なるフィールドA、B、Cを有する医療記録を含むと仮定する。ある管轄区域（例えば、米国）ではフィールドAおよびCをブロックし、別の管轄区域（例えば、英国）ではフィールドAおよびBのみをブロックすることが必要な場合がある。したがって、米国および英国の両方にサービスを提供するマイナーは、異なるブロック基準に基づいてトランザクションの両方のビューを有する必要がある。

【0145】

これは、前のロック解除スクリプトが、独立して満たすことができる複数の異なるブランチを有し得る場合に特に関連する。すなわち、OP_IFブランチを有するトランザクションは、非常に異なるロック解除スクリプトで妥当性確認することができ、おそらく、あるマイナーは、OP_IF…部分を保持するが、別のマイナーは、OP_ELSE…部分を保持するであろう。各マイナーは、どちらがブロックされているかに応じて、Txの2つの「ビュー」のうちの1つに基づいてこのスクリプトの使用を妥当性確認することしかできない。マークルルートストレージの場合のように、これは、どの前のロックスクリプト部分がTxCurrentの入力によって満たされているかの指示を供給することによって行うことができる。

【0146】

5.3.4.3 複合ストレージ

最後のケースでは、ノード104は、UTXOセット内のアウトポイントに対してスクリプトマークルルートおよび部分スクリプトの両方を記憶することによってUTXOデータベースを実装したいと望み得る。これにより、ノードは、オンザフライで支出トランザクションを妥当性確認するために必要とされるマークル証明を、外部ソースから調達するのではなく、再生成することができる。

【0147】

次いで、トランザクション妥当性確認プロセスは、TxCurrentにおいて消費さ

10

20

30

40

50

れている前のロックスクリプト（複数可）の完全性を確実にするために、次のように変更され得る：

1. 支出トランザクション $Tx_{current}$ をパースし、その入力によって参照されるアウトポイント ($TxID_{prev}, i_{prev}$) を取得する。
2. 取得された各アウトポイント j について：
 - 2.1 UTXO データベースから対応する [Merkle root] を取り出す。
 - 2.2 $Tx_{current}$ の [Unlocking script] をパースして、トランザクションによって使用されている [Partial Locking Script] を決定する。
 - 2.3 残りの部分ロックスクリプトによって生成されたリーフを使用することによって、特定の [Partial Locking Script] のマークル証明を取得する。
 - 2.4 マークル包含証明を実行して、[Partial Locking Script] が [Merkle Root] の下の集合のメンバであることを確認する。
3. $Tx_{current}$ の各入力 k について、その [Unlocking script] を、取り出されかつ完全性チェックされた対応する [Partial Locking script] に対して妥当性確認する。

10

【0148】

この更新された妥当性確認方法は、マークル証明がここでは UTXO データベースに記憶された他の部分スクリプトデータから生成されるステップ 2. 3 の追加を除いて、ノードがスクリプトマークルルートのみを記憶する場合と同一であることに留意されたい。

【0149】

20

5. 3. 5 データリーフの選択的開示によるトランザクションコンテンツのブロック

このセクションでは、マークルルート、リーフ（すなわち、平文の公開データ）、および（ブロックされたデータの）リーフのハッシュ値が与えられると、トランザクションの正しさを容易に妥当性確認することができる、ブロックされるべき機密データの包含証明について説明する。

【0150】

以下では、特定のスクリプト（OP_FALSE OP_RETURN）を提供するが、任意のビットコインスクリプトに容易に一般化することができる。以下とする：

【数 9】

$$\begin{aligned} Leaf_0 &= OP_FALSE\ OP_RETURN\ OP_PUSHDATA \\ Leaf_1 &= Data_0 \\ Leaf_2 &= Data_1 \\ &\dots \\ Leaf_n &= Data_{n-1} \end{aligned}$$

30

【0151】

OP_RETURN データが秘密または違法コンテンツを含むと仮定する。そのような場合、マークルルートが依然として計算される可能性がある間、秘密コンテンツについてのいかなる情報も開示しないように、最大数のデータブロックおよび子ハッシュを共有する。例えば、トランザクション $TxID$ が以下を含むと仮定する：

40

【数 10】

$$Data = Leaf_1 || Leaf_1 || \dots || Leaf_n$$

ここで、 $Leaf_2$ および $Leaf_5$ は、共有すべきでない秘密コンテンツを含む。プライベート部分をブロックしながら非秘密部分の存在を証明するために、 $TxID$ が与えられると、 $Leaf_1$ 、 $SHA256(Leaf_2)$ 、 $Leaf_3$ 、 $Leaf_4$ 、 $SHA256(Leaf_5)$ 、 $Leaf_6$ 、 $\dots$ 、 $Leaf_n$ を共有する場合、秘密コンテンツを開示することなく、データのマークルルートを計算し、（ $TxID$ を介して）トランザクシ

50

ョンを妥当性確認することが依然として可能である。leaf₂およびleaf₅が欠落しているとき、検証者はOP_RETURNデータのマークルートを計算することができないことに留意されたい。本発明の新しいTxIDフォーマットはマークルートを必要とするので、OP_RETURNデータのマークルルートなしでは、TxIDを計算することができない。しかしながら、H(leaf₂)およびH(leaf₅)が提供されると、検証者はマークルートを容易に計算することができる。したがって、H(leaf₂)、H(leaf₅)は、検証を実行するために必要とされる部分ロックスクリプトの部分である。

【0152】

図7は、使用可能な出力および使用不可能な出力を含む例示的なトランザクションを示す。この例は、トランザクションがTxID計算およびトランザクション妥当性確認の両方の観点からどのように見られるかを示す。

【0153】

5.4 トランザクションマークルツリー

上述したように、例えば、そのトランザクションが「候補データフィールド」と呼ばれる特定のデータを含むか否かを決定する際に使用するための、編集済みトランザクションの二次トランザクション識別子を生成するために、編集済みトランザクションのフィールドを使用してマークルツリーが生成され得る。二次トランザクション識別子がこのように呼ばれるのは、ほとんどのブロックチェーンプロトコルでは、各トランザクションが、通常、トランザクション全体をハッシュ（またはダブルハッシュ）することによって生成される一次トランザクション識別子に関連付けられているからである。

【0154】

例示的な技法は、マークルツリーのリーフとして使用することができるデータパケット（またはデータフィールド）のセットに、トランザクションをフィールド単位で分割することを含み、マークルツリーのルートは二次トランザクション識別子に対応する。ここで、分割は、トランザクションのデータフィールドを識別することと同等である。言い換えると、トランザクションは、実際には「分割」される必要はない。代わりに、トランザクションの異なる部分が、データフィールドのセットの異なるものとして識別され得る（例えば、割り当てられ得る）。二次トランザクション識別子は、所与のトランザクションに対して一意である（例えば、所与のトランザクションの一意の256ビット数値表現（ハッシュ値））。このハッシュ値は、セット全体（すなわち、完全なトランザクション）を取得することなく、任意の個々のフィールドがマークルツリーの有効なリーフであったかどうかを検証するために使用することができる。

【0155】

トランザクションTxが与えられると、それがブロックチェーンのブロックに含まれていることを検証する1つの方法は、その対応するTxID（一次トランザクション識別子）がブロックに現れることを検証することである。このチェックは、マークルツリー証明（例えば、マークル証明）を実行して、TxIDに対応するトランザクションが、ブロックヘッダ内のハッシュルートによって表されるトランザクションセットの一部であることを検証することによって行うことができる。しかしながら、このチェックは、検証者が最初に完全なトランザクションメッセージ $m = Tx$ を取得し、 $TxID = H(Tx)$ が実際に所与のTxおよび想定されるTxIDに対して成立することを確認することを必要とし、ここで、Hはハッシュ関数である。これは、特に、ユーザが軽量クライアントを実装し、および／またはメッセージmが大きいときに、ブロックチェーンの一部のユーザにとって問題となり得る。

【0156】

例では、以下の定義に従う二次トランザクション識別子MTxIDが生成され得る：

【数11】

$$MTxID := F(Tx, TxID)$$

10

20

30

40

50

アルゴリズム F は、2つの入力メッセージ Tx および $TxID$ から二次トランザクション識別子を生成する一方向関数として作用する。 $TxID$ をトランザクションメッセージ Tx の関数として

【数 1 2】

$$TxID := H^2(Tx)$$

と書くことができるという事実を考えると、これは、 $MTxID$ を、次のように単一のメッセージ $m = Tx$ の関数として書くことができることを意味する：

【数 1 3】

$$MTxID := F(Tx, H^2(Tx)) := F(Tx)$$

アルゴリズム F は、マークルツリージェネレータ (Merkle tree generator) を含む。アルゴリズムは、完全なトランザクションを入力 Tx としてとり、ハッシュダイジェスト $MTxID$ を返し、これをトランザクションの二次識別子として使用することができる。二次識別子は、256ビットハッシュダイジェストであってもよい。二次識別子 $MTxID$ は、この方式において、一次識別子 $TxID$ に取って代わらないことを理解されたい。これを強化するために、アルゴリズム F の設計は、例えば、二次識別子を一次識別子に結び付ける典型的なダブルハッシュ関数 $H^2(Tx)$ を使用した $TxID$ の生成を含む。

【0 1 5 7】

二次トランザクション識別子 $MTxID$ を生成するための方法は、以下に要約されるように、3つの主要な段階に分けることができる。

入力： Tx

F(Tx):

1) $TxID := H^2(Tx)$ を計算する (これはオプションのステップであることに留意されたい)。

2) Tx を、 $N = 2^k$ 個の順序付けられたデータパケットのセット D 、

【数 1 4】

$$D := \{D_1, D_2, \dots, D_N\}$$

に分離する。

3) セット D のパケットをリーフとして使用してバイナリマークルツリー T を生成し、そのルート R を計算する。

出力： $MTxID = R$

【0 1 5 8】

5. 4. 1 段階 1: $TxID$ の計算

本方法は、トランザクションをハッシュすることによって一次トランザクション識別子を生成することを含み得る。第 1 の段階は、 Tx に対して実行される $SHA-256$ ダブルハッシュ計算であり、これは一次トランザクション識別子 $TxID$ を生成する。他のハッシュ関数を使用されてもよく、いくつかの例では、単一のハッシュ計算のみが実行される。通常、 Tx メッセージ自体は、 $TxID$ を明示的に含まない。したがって、後続の段階は、 $TxID$ のこの明示的な事前計算を必要とし、その結果、マークルツリーは、この一次識別子を明示的に符号化するように構築することができる。

【0 1 5 9】

5. 4. 2 段階 2: Tx の順序付けられたデータセット D への分離

メッセージ Tx を含むトランザクションデータは、マークルツリー T のリーフとして使用することができる個別パケットに分離される。トランザクションは、その既存のフィールドに分割され得る。これらのフィールドのほとんどは、一般にサイズが非常に小さい、

典型的には1～32バイトの範囲である単純な数値データを含む。したがって、トランザクションのほとんどは、それぞれ入力（ロック解除）および出力（ロック）に関するsigScriptおよびscriptPubKeyフィールドに関係する。トランザクションのフィールドは、入力フィールド、出力フィールド、および他のフィールドという3つのカテゴリを使用して区別され得る。いくつかの例では、トランザクションは、これらの3つのカテゴリ、例えば、すべての入力フィールドを含む1つのデータフィールド、すべての出力フィールドを含む1つのデータフィールド、およびその他フィールドを含む1つのデータフィールドに分割される。入力フィールドおよび出力フィールドは、非スクリプトフィールドとスクリプトフィールドとに分離することができ、非スクリプトフィールドは数値データを含み、スクリプトフィールドはスクリプトデータを含む。

10

【0160】

以下の表は、トランザクションがデータフィールドのセットに分割され得る方法を示す。

【表1】

フィールド	サイズ (バイト)	入力、出力、または その他のどれか?	スクリプトまたは 非スクリプトのど れか?
version	4	その他	
txin_count	1-9	その他	
tx_in []			
txid_prev	32	入力	非スクリプト
vout	4	入力	非スクリプト
scriptSigLen	変数	入力変数	非スクリプト
scriptSig	変数	入力	スクリプト
sequence	4	入力	非スクリプト
txout_count	1-9	その他	
txout []			
value	8	出力	非スクリプト
scriptPubKeyLen	変数	出力	非スクリプト
scriptPubKey	変数	出力	スクリプト
locktime	4	その他	

20

30

40

【0161】

この表は、Txメッセージをその構成要素フィールドに分割して、いくつかの方法でパ

50

ケットDのセットを形成することができることを示している。

【0162】

いくつかの例では、トランザクションは、トランザクションの入力データ（例えば、txid_prev）を含む少なくとも1つのデータフィールドと、トランザクションの出力データ（例えば、値）を含む少なくとも1つのデータフィールドと、トランザクションの非入力および非出力データ（すなわち、その他のデータ、例えば、バージョン）を含む少なくとも1つのデータフィールドとに分割され得る。各データフィールドは、1つのタイプのデータのみ、例えば入力データのみから構成されてもよい。

【0163】

他の例では、トランザクションは、より多くのデータフィールドに分割され得る。例えば、データフィールドのセットは、トランザクションのスクリプト入力データ（例えば、scriptSig）を含む少なくとも1つのデータフィールドと、トランザクションの非スクリプト入力データ（例えば、vout）を含む少なくとも1つのデータフィールドと、トランザクションのトランザクションのスクリプト出力データ（例えば、scriptPubkey）を含む少なくとも1つのデータフィールドと、トランザクションのトランザクションの非スクリプト出力データ（例えば、scriptPubKeyLen）を含む少なくとも1つのデータフィールドとを含み得る。

【0164】

本開示によれば、各scriptSigおよびscriptPubKeyは、それぞれのスクリプトに基づいて生成されたマークルツリーの対応するマークルルートで置き換えられ得ることに留意されたい。

【0165】

トランザクションTxは、順序付けられたデータパケットの以下のセットに分割される：

$D_1 = \langle \text{version} \rangle$ 、

$D_2 = \langle \text{txin_count} \rangle$ 、

$D_3 = \langle \text{txout_count} \rangle$ 、

$D_4 = \langle \text{locktime} \rangle$ 、

$D_5 = \langle \text{txid_prev} \rangle \parallel \langle \text{vout} \rangle \parallel \langle \text{scriptSigLen} \rangle \parallel \langle \text{sequence} \rangle$ 、

$D_6 = \langle \text{scriptSig} \rangle$ のマークルルート、

$D_7 = \langle \text{value} \rangle \parallel \langle \text{scriptPubKeyLen} \rangle$ 、

$D_8 = \langle \text{scriptPubKey} \rangle$ のマークルルート。

【0166】

このようにデータを分割する選択は、各パケット D_i がバイナリマークルツリーのリーフである場合、いくつかの利点を有する。まず、すべての非スクリプト入力に連結されてパケット D_5 を形成し、同様に、すべての非スクリプト出力に連結されて D_7 を形成する。これは、トランザクションのすべての入力および出力が、非スクリプト構成要素およびスクリプト構成要素（もっと厳密に言えば、そのマークルルート）という厳密に2つの部分に分割されることを意味する。これは、すべての入力および出力を兄弟のリーフとしてペアにできることを意味するので、有利である。したがって、入力および出力を他のフィールドから完全に分離することができ、各入力または出力のスクリプト構成要素および非スクリプト構成要素も分離することができる。

【0167】

これらのフィールドに加えて、TxIDはまた、マークルツリーTのリーフとして含まれ得る。上記の例に続いて、この手段は、データフィールド $D_9 = \langle \text{TxID} \rangle$ を含む。一般に、トランザクションには多くの入力と多くの出力があり、それぞれがアルゴリズムFによって2つの構成要素に分割される。したがって、このツリーTにおけるデータブロックの総数 $N = |D|$ は以下の式によって与えられる。

【数15】

10

20

30

40

50

$$N = \underbrace{4}_{\text{他のフィールド}} + \underbrace{(2 \times n_{in})}_{\text{入力フィールド}} + \underbrace{(2 \times n_{out})}_{\text{出力フィールド}} + \underbrace{1}_{TxID}$$

ここで、 n_{in} 、 n_{out} は、それぞれトランザクション入力および出力の数である。いくつかの例では、すべての他のデータフィールドを連結して、単一のデータフィールドを形成し得る。これにより、リーフの総数は $N = 2 + 2(n_{in} + n_{out})$ に減少する。

【0168】

$N = 2^k$ となるような

【数16】

$k \in \mathbb{Z}^*$

10

がない場合、マークルツリーの十分なリーフがあることを確実にするために、ヌルパディングデータの $2^k - N > 0$ 個のデータパッケージが追加され得る。このパディングは、ヌルデータとすることもできるし、一次トランザクション識別子と最終的な二次識別子 $MTxID$ との間のリンクを強化するために $TxID$ の $2^k - N$ 個のコピーとすることもできる。

【0169】

5. 4. 3 段階3：Dを使用したマークルツリーTの生成およびルートRの計算

20

本方法は、データフィールドのセットをトランザクションマークルツリーのリーフとして使用することを含む。トランザクションマークルツリーは、リーフ層と、1つまたは複数の内部層と、ルート層とを含む。リーフ層は、複数のリーフノード（各ノードがハッシュダイジェストであるため、リーフハッシュとも呼ばれる）を含む。各リーフノードは、トランザクションのそれぞれのデータフィールドをハッシュすることによって生成される。リーフハッシュのうちの少なくとも1つは、一次トランザクション識別子に基づく。いくつかの例では、一次トランザクション識別子をハッシュすることによって、1つまたは複数のリーフハッシュが生成される。いくつかの例では、トランザクションの1つまたは複数のフィールドは、一次トランザクション識別子と連結され、次いで、ハッシュされて、それぞれのリーフハッシュを生成する。各内部層は、複数の内部ノード（または内部ハッシュ）を含む。所与の内部層内の各内部ハッシュは、下位層からの少なくとも2つのハッシュの連結をハッシュすることによって生成される。例えば、最初のまたは最下位の内部層（すなわち、リーフ層に直接接続された内部層）は、少なくとも2つのリーフハッシュの連結をハッシュすることによって生成された内部ノードを含む。ルート層は、トランザクションツリーのマークルルート、すなわち二次トランザクション識別子を含む。二次トランザクション識別子は、1つまたは複数の内部層のうちの最上位の内部層（すなわち、ルート層に直接接続された内部層）の内部ハッシュの連結をハッシュすることによって生成される。

30

【0170】

二次トランザクション識別子（トランザクションマークルツリーのマークルルート）は、ブロックの生成トランザクションに含まれ得る。その後、ブロックがブロックチェーンに記録され得る。代替的に、後述するように、二次トランザクション識別子は、ネットワークの1つまたは複数のノードを介してノードに送信されるトランザクションに含まれ得る。

40

【0171】

アルゴリズムFは、データパッケージDの順序付きセットを取得し、これらのパッケージをリーフとして使用することによってマークルツリーTを構築する。図8は、バイナリマークルツリーの例示的な構成を示す。この例では、最初の4つのデータフィールド $D_1, \dots, D_4$ は、トランザクションのその他フィールドであり、次の $2 \times (n_{in} + n_{out})$ データフィールドは、スクリプト D_5, D_6 および非スクリプト D_7, D_8 フィールドデ

50

ータのペアとして表される入力フィールドおよび出力フィールドである。残りのデータフィールドは、 $T \times I D$ を含む少なくとも1つのフィールド D_9 、 D_{10} と、何らかの整数 k について $N = 2^k$ に達するために必要とされる任意のパディング D_N とを含む。

【0172】

6. さらなる備考

開示される技法の他の変形形態またはユースケースは、本明細書の開示が与えられると、当業者には明らかになり得る。本開示の範囲は、説明された実施形態によって限定されず、添付の特許請求の範囲によってのみ限定される。

【0173】

例えば、上記のいくつかの実施形態は、ビットコインネットワーク106、ビットコインブロックチェーン150およびビットコインノード104に関して説明されている。しかしながら、ビットコインブロックチェーンはブロックチェーン150の1つの特定の例であり、上記の説明は一般に任意のブロックチェーンに適用されてもよいことが理解されるであろう。すなわち、本発明はビットコインブロックチェーンに限定されるものではない。より一般的には、ビットコインネットワーク106、ビットコインブロックチェーン150およびビットコインノード104への上記の任意の参照は、それぞれブロックチェーンネットワーク106、ブロックチェーン150およびブロックチェーンノード104への参照に置き換えられてもよい。ブロックチェーン、ブロックチェーンネットワークおよび／またはブロックチェーンノードは、上記で説明したようなビットコインブロックチェーン150、ビットコインネットワーク106およびビットコインノード104の説明された特性の一部またはすべてを共有してもよい。

【0174】

本発明の好ましい実施形態では、ブロックチェーンネットワーク106はビットコインネットワークであり、ビットコインノード104は、ブロックチェーン150のブロック151を作成、発行、伝搬、および記憶する説明した機能を少なくともすべて実行する。これらの機能のすべてではなく1つまたはいくつかのみを実行する他のネットワークエンティティ（またはネットワーク要素）が存在し得ることは除外されない。すなわち、ネットワークエンティティは、ブロックを作成および発行することなしに、ブロックを伝搬および／または記憶する機能を実行し得る（これらのエンティティが好ましいビットコインネットワーク106のノードとみなされないことを想起されたい）。

【0175】

本発明の他の実施形態では、ブロックチェーンネットワーク106はビットコインネットワークでなくてもよい。これらの実施形態では、ノードが、ブロックチェーン150のブロック151を作成、発行、伝搬、および記憶する機能のうちの少なくとも1つまたはすべてではないがいくつかを実行してもよいことは除外されない。例えば、それらの他のブロックチェーンネットワーク上では、「ノード」は、ブロック151を作成および発行はするが、それらのブロック151を記憶および／または他のノードへの伝搬はしないように構成されたネットワークエンティティを指すために使用され得る。

【0176】

さらにより一般的には、上記の「ビットコインノード」104という用語へのいかなる言及も、「ネットワークエンティティ」または「ネットワーク要素」という用語と置き換えられ、そのようなエンティティ／要素は、ブロックを作成、発行、伝搬、および記憶する役割の一部または全部を実行するように構成される。そのようなネットワークエンティティ／要素の機能は、ブロックチェーンノード104を参照して上記で説明したものと同一方法でハードウェアに実装されてもよい。

【0177】

上記の実施形態は、単なる例として説明されていることが理解されるであろう。より一般的には、以下のステートメントのうちの任意の1つまたは複数による方法、装置、またはプログラムが提供され得る。

【0178】

＜ステートメント 1＞ブロックチェーンランザクションのデータを編集するコンピュータ実装方法であって、

第 1 のブロックチェーンランザクションを取得するステップであって、第 1 のブロックチェーンランザクションは、編集対象のそれぞれのターゲットデータを含む 1 つまたは複数のそれぞれのスクリプトを含む、ステップと、

1 つまたは複数のそれぞれのスクリプトの少なくとも 1 つについて、それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの 1 つまたは複数にわたって分割される、ステップと、

少なくとも 1 つのそれぞれのスクリプトをそれぞれのマークルツリーのそれぞれのマークルルートに置き換えることによって、第 1 のブロックチェーンランザクションの編集済みバージョンを生成するステップと

を含む方法。

【0179】

＜ステートメント 2＞各それぞれのスクリプトは、1 つまたは複数の関数を含み、それぞれのマークルツリーを構築する上記ステップは、連続する関数の 1 つまたは複数のそれぞれのシーケンスをマークルツリーのそれぞれのリーフとしてグループ化することを含む、ステートメント 1 に記載の方法。

【0180】

＜ステートメント 3＞各それぞれのスクリプトは、実行されたときにそれぞれのターゲットデータを示すように構成されたそれぞれのデータ関数を含み、それぞれのマークルツリーを構築する上記ステップは、それぞれのマークルツリーのそれぞれのリーフの一部としてそれぞれのデータ関数を含めることを含む、ステートメント 2 に記載の方法。

【0181】

＜ステートメント 4＞各それぞれのスクリプトは、それぞれのターゲットデータのそれぞれのデータ長を含み、それぞれのマークルツリーを構築する上記ステップは、それぞれのデータ長に基づく、ステートメント 3 に記載の方法。

【0182】

＜ステートメント 5＞それぞれのマークルツリーを構築する上記ステップは、それぞれのデータ関数と同じそれぞれのマークルツリーのそれぞれのリーフの一部としてそれぞれのデータ長を含めることを含む、ステートメント 4 に記載の方法。

【0183】

＜ステートメント 6＞それぞれのターゲットデータは、それぞれのマークルツリーの複数のそれぞれのリーフにわたって分割される、先行するステートメントのいずれかに記載の方法。

【0184】

＜ステートメント 7＞第 1 のブロックチェーンランザクションは、それぞれのターゲットデータを含む複数のそれぞれのスクリプトを含み、方法は、

複数のそれぞれのスクリプトの各々について、それぞれのスクリプトに基づいてそれぞれのマークルツリーを構築するステップであって、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの 1 つまたは複数にわたって分割される、ステップと、

各それぞれのスクリプトをそれぞれのマークルツリーのそれぞれのマークルルートで置き換えることによって、第 1 のブロックチェーンランザクションの編集済みバージョンを生成するステップと

を含む、先行するステートメントのいずれかに記載の方法。

【0185】

＜ステートメント 8＞ブロックチェーンランザクションの編集済みバージョンをハッシュすることによって、第 1 のブロックチェーンランザクションの編集済みバージョンの一次ランザクション識別子を生成するステップを含む、先行するステートメントのい

10

20

30

40

50

いずれかに記載の方法。

【0186】

＜ステートメント9＞第1のブロックチェーントランザクションに対する要求に応答して、ブロックチェーントランザクションの編集済みバージョンをエンティティに送信するステップを含む、先行するステートメントのいずれかに記載の方法。

【0187】

＜ステートメント10＞第1のブロックチェーントランザクションの編集済みバージョンをメモリに記憶するステップを含む、先行するステートメントのいずれかに記載の方法。

【0188】

＜ステートメント11＞方法は、トランザクションジェネレータによって実行される、先行するステートメントのいずれかに記載の方法。

【0189】

＜ステートメント12＞メッセージに対する署名を生成するステップであって、メッセージの1つまたは複数の部分は、第1のブロックチェーントランザクションの編集済みバージョンに基づく、ステップと、

署名を第1のブロックチェーントランザクションの入力に含めるステップと、

第1のブロックチェーントランザクションを妥当性確認のためにブロックチェーンネットワークにサブミットするステップと

を含む、ステートメント11に記載の方法。

【0190】

＜ステートメント13＞署名は、楕円曲線デジタル署名アルゴリズム（ECDSA）署名である、ステートメント12に記載の方法。

【0191】

＜ステートメント14＞入力は、それぞれのターゲットデータを含むそれぞれのスクリプトを含む前のブロックチェーントランザクションの出力を参照し、メッセージの一部は、前のブロックチェーントランザクションの出力に基づき、前のブロックチェーントランザクションの出力に基づくメッセージの一部は、それぞれのスクリプトに基づいて構築されたマークルツリーのマークルルートに基づき、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの1つまたは複数にわたって分割される、ステートメント12またはステートメント13に記載の方法。

【0192】

＜ステートメント15＞方法は、トランザクションバリデータによって実行される、ステートメント1から10のいずれかに記載の方法。

【0193】

＜ステートメント16＞第1のブロックチェーントランザクションの入力から署名を抽出するステップと、

メッセージに対する署名を検証するステップであって、メッセージの1つまたは複数の部分は、第1のブロックチェーントランザクションの編集済みバージョンに基づく、ステップと

ステートメント15に記載の方法。

【0194】

＜ステートメント17＞入力は、それぞれのターゲットデータを含むそれぞれのスクリプトを含む前のブロックチェーントランザクションの出力を参照し、メッセージの一部は、前のブロックチェーントランザクションの出力に基づき、前のブロックチェーントランザクションの出力に基づくメッセージの一部は、それぞれのスクリプトに基づいて構築されたマークルツリーのマークルルートに基づき、それぞれのターゲットデータは、それぞれのマークルツリーのそれぞれのリーフのうちの1つまたは複数にわたって分割される、ステートメント16に記載の方法。

【0195】

10

20

30

40

50

<ステートメント 18> 第 1 のブロックチェーントランザクションの編集済みバージョンをブロックチェーンに記憶するステップを含む、ステートメント 15 またはそれに従属する任意のステートメントに記載の方法。

【0196】

<ステートメント 19> それぞれのスクリプトのうちの少なくとも 1 つは、第 1 のブロックチェーントランザクションの第 1 の出力の第 1 のロックスクリプトであり、方法は、修正された一次トランザクションにマッピングされたそれぞれのマークルルートと、第 1 の出力のそれぞれのインデックスとをデータベースに記憶するステップを含む、ステートメント 8 およびステートメント 15 から 18 のいずれかに記載の方法

。

【0197】

<ステートメント 20> 第 2 の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、第 2 のブロックチェーントランザクションは、第 1 のブロックチェーントランザクションの修正された一次トランザクション識別子および第 1 の出力のインデックスを参照する第 1 の入力を含み、第 1 の入力は、第 1 のロック解除スクリプトを含む、ステップ

を含み、方法は、

データベースから、修正された一次トランザクション識別子および第 1 の出力のそれぞれのインデックスにマッピングされたそれぞれのマークルルートを取り出すことと、

第 1 の入力の第 1 のロック解除スクリプトに基づいて、第 1 の出力の部分ロックスクリプトを決定することと、

マークル包含証明を実行して、部分ロックスクリプトがそれぞれのマークルルートに対応するそれぞれのマークルツリーのそれぞれのリーフであることを確認することと、

部分ロックスクリプトに照らして第 1 のロック解除スクリプトを妥当性確認することと

によって第 2 のブロックチェーントランザクションを妥当性確認するステップ

を含む、ステートメント 19 に記載の方法。

【0198】

<ステートメント 21> それぞれのスクリプトのうちの少なくとも 1 つは、第 1 のブロックチェーントランザクションの第 1 の出力の第 1 のロックスクリプトであり、方法は、

修正された一次トランザクション識別子および第 1 の出力のそれぞれのインデックスにマッピングされた部分ロックスクリプトをデータベースに記憶するステップであって、部分ロックスクリプトは、第 1 のロックスクリプトの少なくとも一部を含むが、ターゲットデータは含まない、ステップ

を含む、ステートメント 8 およびステートメント 15 から 18 のいずれかに記載の方法

。

【0199】

<ステートメント 22> 第 2 の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、第 2 のブロックチェーントランザクションは、第 1 のブロックチェーントランザクションの修正された一次トランザクション識別子および第 1 の出力のインデックスを参照する第 1 の入力を含み、第 1 の入力は、第 1 のロック解除スクリプトを含む、ステップ

を含み、方法は、

データベースから、修正された一次トランザクション識別子および第 1 の出力のそれぞれのインデックスにマッピングされたそれぞれの部分ロックスクリプトを取り出すことと、

部分ロックスクリプトに照らして第 1 のロック解除スクリプトを妥当性確認することと

によって第 2 のブロックチェーントランザクションを妥当性確認するステップ

を含む、ステートメント 21 に記載の方法。

10

20

30

40

50

【0200】

＜ステートメント23＞データベースから、修正された一次トランザクション識別子および第1の出力のそれぞれのインデックスにマッピングされたそれぞれのマークルルートおよび部分ロックスクリプトを取り出すステップと、

第1の入力の第1のロック解除スクリプトに基づいて、第1の出力の部分ロックスクリプトを決定するステップと、

取り出された部分ロックスクリプトに基づいて生成された1つまたは複数のそれぞれのリーフを使用して、決定された部分ロックスクリプトのためのマークル証明を取得するステップと、

マークル包含証明を実行して、決定された部分ロックスクリプトがそれぞれのマークルルートに対応するそれぞれのマークルツリーのそれぞれのリーフであることを確認するステップと、

決定された部分ロックスクリプトに照らして第1のロック解除スクリプトを妥当性確認するステップと

を含む、ステートメント19およびステートメント21に記載の方法。

【0201】

＜ステートメント24＞編集済みトランザクションは複数のフィールドを含み、方法は、

トランザクションマークルツリーを生成するステップであって、トランザクションマークルツリーのそれぞれの複数のそれぞれのリーフは、編集済みトランザクションの1つまたは複数のフィールドから形成される、ステップと、

編集済みトランザクションの二次トランザクション識別子を生成するステップであって、二次トランザクション識別子は、トランザクションマークルツリーのマークルルートを含む、ステップと

を含む、先行するステートメントのいずれかに記載の方法。

【0202】

＜ステートメント25＞トランザクションマークルツリーの1つまたは複数のリーフは、編集済みトランザクションの複数のフィールドから形成される、ステートメント24に記載の方法。

【0203】

＜ステートメント26＞トランザクションマークルツリーの1つまたは複数のリーフは、編集済みトランザクションの単一のフィールドから形成される、ステートメント24またはステートメント25に記載の方法。

【0204】

＜ステートメント27＞トランザクションマークルツリーの1つまたは複数のリーフは、編集済みトランザクションのそれぞれのフィールドのそれぞれの部分から形成されるが、完全なそれぞれのフィールドからは形成されない、ステートメント24またはそれに従属する任意のステートメントに記載の方法。

【0205】

＜ステートメント28＞コンピュータ機器であって、

1つまたは複数のメモリユニットを備えるメモリと、

1つまたは複数の処理ユニットを備える処理装置であって、メモリは、処理装置上で実行されるように構成されたコードを記憶し、コードは、処理装置上にあるときに、ステートメント1から27のいずれかに記載の方法を実行するように構成される、処理装置とを備えるコンピュータ機器。

【0206】

＜ステートメント29＞コンピュータ可読ストレージ上に具現化され、1つまたは複数のプロセッサ上で実行されると、ステートメント1から27のいずれかに記載の方法を実行するように構成されたコンピュータプログラム。

10

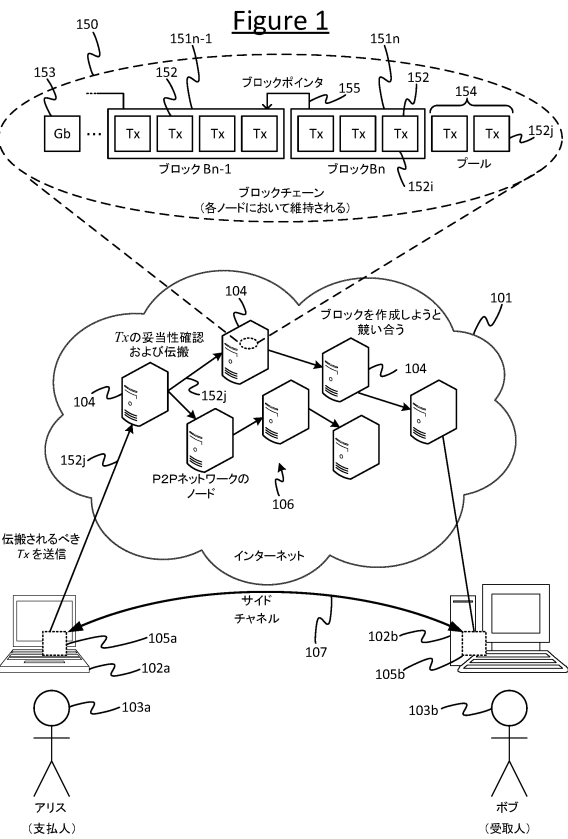
20

30

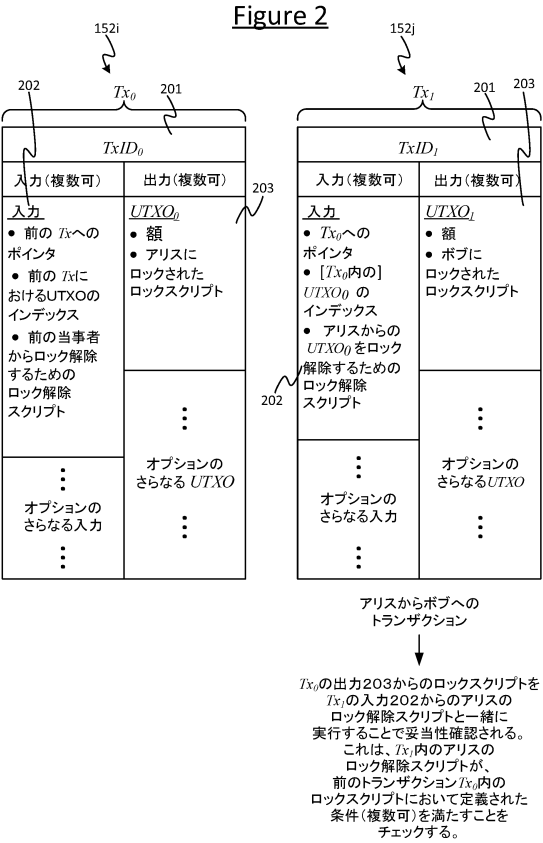
40

50

【図面】
【図 1】



【図 2】

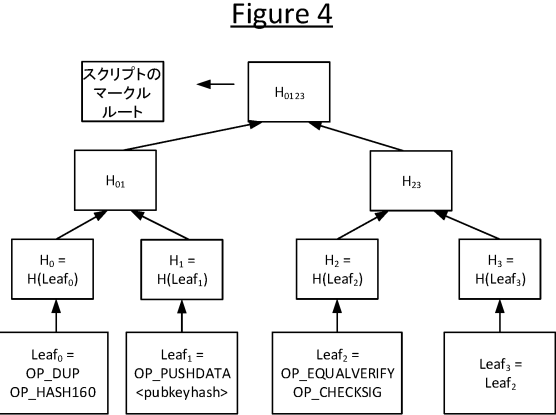


【図 3】

Figure 3

$TxID_1$			
バージョン	1	ロック時間	0
In-count	1	Out-count	2
入力リスト		出力リスト	
アウトポイント	ロック解除スクリプト	値	ロックスクリプト
$TxID, Index$	$< Sig_A > < P_A >$	x satoshi	$[P2PKH P_B]$
		0	OP_FALSE OP_RETURN $< data >$

【図 4】



10

20

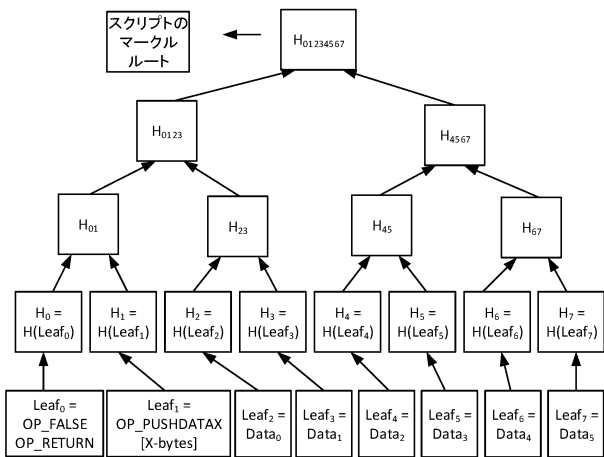
30

40

50

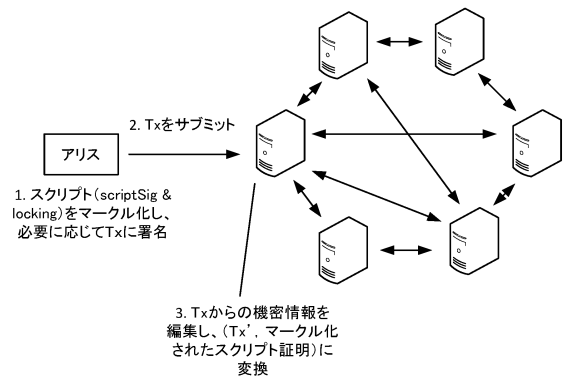
【図 5】

Figure 5



【図 6】

Figure 6



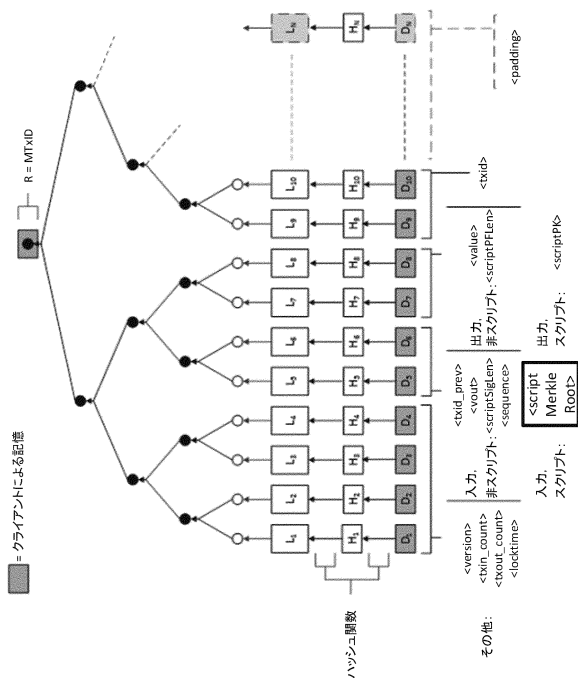
【図 7】

Figure 7

TxID ₁			
バージョン	1	ロック時間	0
In-count	1	Out-count	2
入カリスト		出カリスト	
アウトポイント	ロック解除スクリプト	値	ロックスクリプト
TxID, Index	< Sig _A > < P _A > の マーケルルート	x satoshi	[P2PKH P _B] の マーケルルート
		0	OP_FALSE OP_RETURN < data > のマーケルルート

【図 8】

Figure 8



10

20

30

40

50

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2022/074687

A. CLASSIFICATION OF SUBJECT MATTER INV. H04L9/00 H04L9/32 H04L9/40 ADD. According to International Patent Classification (IPC) or to both national classification and IPC												
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) H04L Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EPO-Internal, WPI Data												
C. DOCUMENTS CONSIDERED TO BE RELEVANT												
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.										
X	Anonymous: "bips/bip-0114.mediawiki at master . bitcoin/bips . GitHub", , 21 August 2020 (2020-08-21), XP093000751, Retrieved from the Internet: URL:https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki [retrieved on 2022-11-22] the whole document ----- -/--	1-29										
☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.												
* Special categories of cited documents : <table><tr><td>"A" document defining the general state of the art which is not considered to be of particular relevance</td><td>"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention</td></tr><tr><td>"E" earlier application or patent but published on or after the international filing date</td><td>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone</td></tr><tr><td>"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</td><td>"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art</td></tr><tr><td>"O" document referring to an oral disclosure, use, exhibition or other means</td><td>"&" document member of the same patent family</td></tr><tr><td>"P" document published prior to the international filing date but later than the priority date claimed</td><td></td></tr></table>			"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art	"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family	"P" document published prior to the international filing date but later than the priority date claimed	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention											
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone											
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art											
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family											
"P" document published prior to the international filing date but later than the priority date claimed												
Date of the actual completion of the international search 22 November 2022		Date of mailing of the international search report 30/11/2022										
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016		Authorized officer Kufer, Léna										

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2022/074687

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Anonymous: "Bitcoin Optech - Schnorr and Taproot workshop (2019-09-27) Bitcoin Transcripts", / 27 September 2019 (2019-09-27), XP093000580, Retrieved from the Internet: URL:https://btc transcripts.com/misc/2019-09-27-schnorr-taproot-workshop/ [retrieved on 2022-11-22] the whole document -----	1-29
X	Anonymous: "2018-03-06-merkleized-abstract-syntax-trees-mast", / 6 March 2018 (2018-03-06), XP093000542, Retrieved from the Internet: URL:https://diyhl.us/wiki/transcripts/bitcoin-core-dev-tech/2018-03-06-merkleized-abstract-syntax-trees-mast/ [retrieved on 2022-11-22] the whole document -----	1-29
A	DOMINIC DEUBER ET AL: "Redactable Blockchain in the Permissionless Setting", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 10 January 2019 (2019-01-10), XP081498795, DOI: 10.1109/SP.2019.00039 the whole document -----	1-29

フロントページの続き

MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,N
E,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,
CV,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IQ,IR,IS,IT,J
M,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY
,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,T
H,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,ZA,ZM,ZW
ーピー ロンドン マーケット プレイス 30 エヌチェーン ライセンシング アーゲー 内
(72)発明者 リウ, ワイ
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 ライセンシング アーゲー 内