(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7707209号
(P7707209)

(45)発行日　令和7年7月14日(2025.7.14)

(24)登録日　令和7年7月4日(2025.7.4)

(51)国際特許分類

Ｈ０４Ｌ　9/32　(2006.01)		Ｈ０４Ｌ	9/32　２００Ｂ
Ｇ０６Ｆ　21/64　(2013.01)		Ｈ０４Ｌ	9/32　２００Ｚ
		Ｇ０６Ｆ	21/64

請求項の数　28　（全46頁）

(21)出願番号　特願2022-575356(P2022-575356)
(86)(22)出願日　令和3年5月12日(2021.5.12)
(65)公表番号　特表2023-528649(P2023-528649A)
(43)公表日　令和5年7月5日(2023.7.5)
(86)国際出願番号　PCT/EP2021/062620
(87)国際公開番号　WO2021/249717
(87)国際公開日　令和3年12月16日(2021.12.16)
審査請求日　令和6年4月15日(2024.4.15)
(31)優先権主張番号　2008951.2
(32)優先日　令和2年6月12日(2020.6.12)
(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　パガーニ，アレッシオ
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

最終頁に続く

(54)【発明の名称】　ファイル検証システムおよび方法

(57)【特許請求の範囲】

【請求項１】

ブロックチェーン上にオーバーレイされたツリー構造を使用する方法であって、前記方法は、ユーザのコンピュータ機器上のオペレーティングシステムまたはファイルシステムによって実行され、前記ツリー構造は、複数のノードとノード間のエッジとを含み、各ノードは、前記ブロックチェーン上に記録された異なるトランザクションであり、各エッジは、それぞれの子ノードからそれぞれの親ノードに接続し、前記エッジは、トランザクションＩＤを含む各トランザクションによって形成され、各子ノードは、前記子ノードのそれぞれのペイロードにおいて前記それぞれの親ノードの前記トランザクションＩＤを指定し、前記親ノードのうちの１つは、前記ツリー構造のルートノードであり、前記方法は、

前記ブロックチェーンを検査して、前記ツリー構造の少なくとも一部を識別するステップであって、ターゲット子ノードの前記それぞれのペイロードにおいてファイルの記録を含む前記子ノードのうちの前記ターゲット子ノードを少なくとも識別することと、前記ツリー構造を通って前記ターゲット子ノードから前記ルートノードに戻る１つまたは複数のエッジを含むパスを識別することとを含む、ステップと、

チェックを実行するステップであって、

Ａ）前記ターゲット子ノードから前記ルートノードに戻る前記識別されたパスに沿ったエッジごとに、前記それぞれの子ノードが前記それぞれの親ノードに関連付けられた鍵によって署名されていることをチェックすることと、

Ｂ）前記ファイルの現在のインスタンスが前記ターゲット子ノードに含まれる前記

記録と一致することをチェックすることと
を含む、ステップと、
少なくともチェックＡ）およびＢ）が肯定的であるという条件で、前記ファイルの前記現在のインスタンスを検証するステップと
を含む方法。
【請求項２】
前記検証するステップを条件として、前記ユーザが前記ファイルの前記現在のインスタンスに対して要求されたアクションを実行することを可能にするステップを含む、請求項１に記載の方法。
【請求項３】
前記オペレーティングシステムまたはファイルシステムは、前記ユーザが前記コンピュータ機器を使用して前記要求されたアクションを実行することを可能にする前に前記チェックを実施するように構成される、請求項２に記載の方法。
【請求項４】
前記要求されたアクションは、前記ファイルを読み出すこと、前記ファイルを修正すること、前記ファイルを実行すること、または前記ファイルを削除することのうちの１つを含む、請求項２または３に記載の方法。
【請求項５】
前記ターゲット子ノードの前記ペイロードは、１つまたは複数の許可されたアクションの指示をさらに含み、前記オペレーティングシステムまたはファイルシステムは、前記要求されたアクションが前記ターゲット子ノードにおいて示された前記許可されたアクションのうちの１つであるというさらなる条件でのみ、前記要求されたアクションを可能にすることを実行するように構成される、請求項２から４のいずれか一項に記載の方法。
【請求項６】
前記１つまたは複数の許可されたアクションは、前記ファイルを読み出すこと、前記ファイルを修正すること、前記ファイルを実行すること、または前記ファイルを削除することのうちの１つまたは複数を含む、請求項５に記載の方法。
【請求項７】
前記ターゲット子ノードの前記ペイロードは、１つまたは複数の許可されたユーザの指示をさらに含み、前記オペレーティングシステムまたはファイルシステムは、前記アクションを要求する前記ユーザが前記ターゲット子ノードの前記ペイロードにおいて示された許可されたユーザであるというさらなる条件でのみ、前記要求されたアクションを可能にすることを実行するように構成される、請求項２から６のいずれか一項に記載の方法。
【請求項８】
ブロックチェーン上にオーバーレイされたツリー構造を使用する方法であって、前記方法は、ファイルの現在のインスタンスの完全性を検証するために、アンチウィルスソフトウェアまたは別のアプリケーションによって実行され、前記ツリー構造は、複数のノードとノード間のエッジとを含み、各ノードは、前記ブロックチェーン上に記録された異なるトランザクションであり、各エッジは、それぞれの子ノードからそれぞれの親ノードに接続し、前記エッジは、トランザクションＩＤを含む各トランザクションによって形成され、各子ノードは、前記子ノードのそれぞれのペイロードにおいて前記それぞれの親ノードの前記トランザクションＩＤを指定し、前記親ノードのうちの１つは、前記ツリー構造のルートノードであり、前記方法は、
前記ブロックチェーンを検査して、前記ツリー構造の少なくとも一部を識別するステップであって、ターゲット子ノードの前記それぞれのペイロードにおいてファイルの記録を含む前記子ノードのうちの前記ターゲット子ノードを少なくとも識別することと、前記ツリー構造を通って前記ターゲット子ノードから前記ルートノードに戻る１つまたは複数のエッジを含むパスを識別することとを含む、ステップと、
チェックを実行するステップであって、
Ａ）前記ターゲット子ノードから前記ルートノードに戻る前記識別されたパスに沿っ

たエッジごとに、前記それぞれの子ノードが前記それぞれの親ノードに関連付けられた鍵によって署名されていることをチェックすることと、
　　Ｂ）前記ファイルの現在のインスタンスが前記ターゲット子ノードに含まれる前記記録と一致することをチェックすることと
　を含む、ステップと、
　少なくともチェックＡ）およびＢ）が肯定的であるという条件で、前記ファイルの前記現在のインスタンスを検証するステップと
　を含む方法。
【請求項９】
　それぞれの親ノードに関連付けられた前記鍵は、前記親ノードのペイロードに含まれることによって前記親ノードに関連付けられる、請求項１から８のいずれか一項に記載の方法。
【請求項１０】
　各トランザクションは、ロックスクリプトを含む少なくとも１つの出力と、それぞれの他のトランザクションの出力を指し示し、かつ前記それぞれの他のトランザクションの前記出力をロック解除するためのロック解除スクリプトを含む少なくとも１つの入力とを含む、請求項１から９のいずれか一項に記載の方法。
【請求項１１】
　各子ノードの前記入力は、前記それぞれの親ノードの前記鍵によって署名され、Ａ）は、前記パスに沿った各子ノードの前記入力が、前記それぞれの親ノードの前記鍵によって署名されていることをチェックすることを含む、請求項１０に記載の方法。
【請求項１２】
　各子ノードの前記ペイロードは、前記それぞれの子ノードの前記出力のうちの１つまたは複数に含まれる、請求項１０または１１に記載の方法。
【請求項１３】
　前記記録は、少なくとも前記ファイルの明示的なコピーを含み、Ｂ）は、前記ファイルの前記現在のインスタンスが、前記ターゲット子ノードの前記ペイロードに記録された前記コピーと同じであることをチェックすることを含む、請求項１から１２のいずれか一項に記載の方法。
【請求項１４】
　前記記録は、少なくともプレイメージのハッシュを含み、前記プレイメージは前記ファイルを含み、Ｂ）は、前記ファイルの現在のインスタンスを含む前記プレイメージの前記現在のインスタンスの前記ハッシュが、前記ターゲット子ノードの前記ペイロードに記録された前記ハッシュと同じであることを少なくともチェックすることを含む、請求項１から１３のいずれか一項に記載の方法。
【請求項１５】
　前記記録は、ハッシュツリーのリーフとして複数のファイルから生成された、前記ハッシュツリーのハッシュルートを含み、前記ファイルは、前記複数のファイルのうちの１つであり、Ｂ）は、前記複数のファイルの現在のインスタンスから計算された前記ハッシュルートが、前記ターゲット子ノードに記録された前記ハッシュルートと同じであることをチェックすることを含む、請求項１から１４のいずれか一項に記載の方法。
【請求項１６】
　前記親ノードのうちの少なくとも１つは、前記親ノードのうちの別の親ノードの子ノードである中間親ノードである、請求項１から１５のいずれか一項に記載の方法。
【請求項１７】
　前記少なくとも１つの中間親ノードは、前記パスが２つ以上のエッジを含むように、前記ターゲット子ノードと前記ルートノードとの間の前記パスに沿った少なくとも１つのノードを含む、請求項１６に記載の方法。
【請求項１８】
　前記少なくとも１つの中間親ノードは、前記ファイルを含むフォルダを表すフォルダノ

ードを含む、請求項１７に記載の方法。
【請求項１９】
前記少なくとも１つの中間親ノードは、ユーザノードを含み、前記ルートノードに関連付けられた前記鍵は、システム管理者の鍵であり、前記ユーザノードに関連付けられた前記鍵は、ユーザの鍵である、請求項１７に記載の方法。
【請求項２０】
前記チェックは、Ｃ）前記ルートノードが信頼できるエンティティによって署名されていることをチェックするという追加のチェックを含み、前記検証することは、前記チェックＣ）が肯定的であることをさらに条件とする、請求項１から１９のいずれか一項に記載の方法。
【請求項２１】
前記信頼できるエンティティはシステム管理者である、請求項２０に記載の方法。
【請求項２２】
前記ルートノードの入力は、前記信頼できるエンティティによって署名され、前記チェックＣ）は、前記ルートノードの入力が前記信頼できるエンティティによって署名されていることをチェックすることを含む、少なくとも請求項１０に従属する請求項２０または２１に記載の方法。
【請求項２３】
前記チェックは、Ｄ）前記ターゲット子ノードがグラフ構造の他のどの子ノードの親ノードでもないことをチェックするというさらなるチェックを含み、前記検証することは、前記チェックＤ）が、前記ターゲット子ノードが前記他の子ノードのどの子ノードの親でもないと決定することをさらに条件とする、請求項１から２２のいずれか一項に記載の方法。
【請求項２４】
新しいエッジを介して前記ターゲット子ノードに新しい子ノードを付加することによって、前記ブロックチェーン上の前記グラフ構造に記録された前記ファイルを後に更新するステップを含み、前記新しい子ノードは、前記新しい子ノードのペイロードにおいて前記更新されたファイルの記録を含む、請求項２３に記載の方法。
【請求項２５】
前記ターゲット子ノードは、前記記録の終了時間をさらに指定し、前記チェックは、Ｅ）現在の時間が前記ターゲット子ノードにおいて指定された前記終了時間よりも遅くないという点で前記記録が失効していないことをチェックするという別のチェックを含み、前記検証は、前記チェックＥ）が、前記現在の時間が失効していないと決定することをさらに条件とする、請求項１から２４のいずれか一項に記載の方法。
【請求項２６】
前記ツリー構造はメタネットグラフである、請求項１から２５のいずれか一項に記載の方法。
【請求項２７】
コンピュータシステムであって、
１つまたは複数の処理ユニットを含む処理装置と、
１つまたは複数のメモリユニットを含むメモリと
を備え、
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、前記処理装置上で実行されると、請求項１から２６のいずれか一項に記載の動作を実行するように構成されている、システム。
【請求項２８】
コンピュータ可読ストレージ上に具現化されたコンピュータプログラムであって、前記コンピュータプログラムは、１つまたは複数の処理ユニット上で実行されると、請求項１から２６のいずれか一項に記載の動作を実行するように構成されたコードを含む、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、ブロックチェーン上のトランザクションのペイロードにデータコンテンツの記録が記憶されるブロックチェーンの応用に関する。
【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（以下では「ブロックチェーンネットワーク」と呼ばれる）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る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プール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。
【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。
【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、ノードが有効なプルーフオブワークの解を識別しようとするブロックテンプレートにそれらを挿入する。有効な解が見つかると、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実施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。

【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「コインベー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実施される。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も、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。この場合、各トランザクションは、過去のトランザクションのシーケンス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。

【０００９】

ブロックチェーンネットワークは、すでにインターネットのような基礎となるネットワークの上にオーバーレイされたオーバーレイネットワークの一種である。しかしながら、ブロックチェーン上にオーバーレイネットワークのさらなる層をオーバーレイすることも

可能である。この例はメタネットとして知られている。メタネットの各ノードは、ブロックチェーン上の異なるトランザクションである（「ノード」は、ここでは異なる意味で使用されており、ブロックチェーンネットワークのノードを指すのではなく、メタネットのノードを指すことに留意されたい）。データコンテンツおよびメタネットメタデータは、ＯＰ＿ＲＥＴＵＲＮによってトランザクションの使用不可能な出力において、そのような各トランザクションのペイロードに記憶される。データコンテンツは、例えば、テキスト、画像、ビデオまたはオーディオコンテンなどを記憶するためにメタネットが使用されている実際のユーザコンテンツであり、メタデータは、メタネットノード間のリンクを定義する。メタネットノード間のリンクまたはエッジは、ブロックチェーン層における使用エッジに必ずしも対応しない。すなわち、所与のメタネットトランザクションの入力が、ブロックチェーン層における別のファンディングトランザクションの出力を指し示す場合、メタネット層におけるその同じトランザクションまたはメタネットノードは、必ずしも、ファンディングトランザクションと同じトランザクションではない。代わりに、メタネット層におけるリンクまたはエッジは、メタネットのデータコンテンツ間のリンクを定義する。

【発明の概要】

【００１０】

オペレーティングシステムなどは、外部の脅威（例えば、ウイルス、マルウェア、スパイウェア）および意図しないユーザ挙動（例えば、意図しないファイル削除、構成変更）から常に保護する必要がある。さらに、特に企業環境では、ライセンス契約によりソフトウェアの使用が制限または制御されている場合がある。

【００１１】

本開示は、ソフトウェアセキュリティおよびデータ完全性を保証し、ファイル動作および変更を監視および管理するために使用することができる、メタネットベースのグラフなどの、ブロックチェーン上にオーバーレイされたグラフ構造に基づくソリューションを提示する。例えば、実施形態では、メタネットベースのシステムは、本明細書でメタネットトリップワイヤプロトコル（ＭＴＰ：Metanet Tripwire Protocol）と呼ばれる新しいメタネットプロトコルを使用して設計することができる。ＭＴＰは、所与のファイルに対して許可された動作（例えば、読み出し、書き込み、実行）およびファイルの有効性を指定する。メタネットトリップワイヤを使用するシステムは、メタネットグラフを使用してシステム管理者によって証明されたファイルを開くまたは実行することのみ行う。

【００１２】

本開示の一態様によれば、ブロックチェーン上にオーバーレイされたツリー構造を使用する方法であって、ツリー構造は、複数のノードとノード間のエッジとを含み、各ノードは、ブロックチェーン上に記録された異なるトランザクションであり、各エッジは、それぞれの子ノードからそれぞれの親ノードに接続し、エッジは、トランザクションＩＤを含む各トランザクションによって形成され、各子ノードは、子ノードのそれぞれのペイロードにおいてそれぞれの親ノードのトランザクションＩＤを指定し、親ノードのうちの１つは、ツリー構造のルートノードである、方法が提供される。この方法は、ブロックチェーンを検査して、ツリー構造の少なくとも一部を識別するステップを含み、このステップは、ターゲット子ノードのそれぞれのペイロードにおいてファイルの記録を含む子ノードのうちのターゲット子ノードを少なくとも識別することと、ツリー構造を通ってターゲット子ノードからルートノードに戻る１つまたは複数のエッジを含むパスを識別することとを含む。次いで、この方法は、チェックを実行するステップを含み、このステップは、Ａ）ターゲット子ノードからルートノードに戻る識別されたパスに沿ったエッジごとに、それぞれの子ノードがそれぞれの親ノードに関連付けられた鍵によって署名されていることをチェックすることと、Ｂ）ファイルの現在のインスタンスがターゲット子ノードに含まれる記録と一致することをチェックすることとすることとを含む。ファイルの現在のインスタンスは、少なくともチェックＡ）およびＢ）が肯定的であるという条件で検証される。

【図面の簡単な説明】

【００１３】
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】ブロックチェーン上にオーバーレイされたネットワークの概略図である。
【図４】ブロックチェーン上にメタネットなどのネットワークをオーバーレイするための例示的なプロトコルを示す概略的なトランザクション図である。
【図５】本明細書に開示される実施形態によるメタネットシステムツリーを概略的に示し、メタネットシステムツリーは、システムルートノードおよび３つのファイルハッシュノードを表す。
【図６】プログラムを実行するために必要とされるすべてのファイルがマークルツリーにおいてリンクされ、マークルツリールートがファイルハッシュノードに記憶される、本明細書の実施形態によるマークルツリーの使用を概略的に示す。
【図７】本明細書に開示される実施形態による、ファイルハッシュノード（ＦＨ）を記憶および更新するために使用されるメタネットツリーを概略的に示し、ノードは、子を有さない場合のみ有効である。
【図８】本明細書に開示される実施形態による、ファイルシステムフォルダ構造（左側）と、メタネットフォルダノードを使用した複製構造（右側）とを概略的に示す。ドライブＣ：およびＹ：は、フォルダノードとしてオンチェーンで記憶される。
【図９】保護されたファイルが管理者によって管理される、本明細書の実施形態によるマルチユーザシステムを概略的に示す。各ユーザは、プライベートファイルの異なるサブセットへのアクセスを有し、ファイルは、（場合によっては異なる許可を有する）複数のユーザによってアクセスされ得る。
【図１０】本明細書に開示される実施形態による企業シナリオを概略的に示し、会社のシステム管理者が各システムルートを初期化する。ユーザノードは、ファイルまたはフォルダのサブセットを管理するためにユーザに割り当てることができる。
【図１１】本明細書に開示される実施形態によるメタネットトリップワイヤノードの例示的な構造を概略的に示す。オプションの特徴のリストは網羅的ではない。
【図１２】本明細書に開示される実施形態による例示的な方法を示す概略フローチャートである。
【発明を実施するための形態】
【００１４】
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。
【００１５】
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）などの他の機器を備える処理装置を含む。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。

【００１６】

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、データがプ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ソリューションを必要とする）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。

【００１７】

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序を定義するために、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発生ブロック（Ｇｂ：genesis block）１５３までずっと戻る。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発生ブロック１５３を指し示していた。

【００１８】

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していない。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。

【００１９】

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」されるべきであることを指定する。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。

【００２０】

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先のユーザ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。場合によっては、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。場合によっては、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。

【００２１】

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に求める。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１５２ｉ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１５２ｉ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、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く、または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１５２ｊ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１５２ｊ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。

【００２２】

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（例えば、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前方のトランザクション１５２ｊ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１５２ｉ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１５２ｊ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。

【００２３】

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、「プルーフオブワーク」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、プ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。ハッシュ関数の特性は、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。

【００２４】

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証明として提供する（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閾値コンセンサスに伝搬し、プロトコルルールを実施する。次いで、トランザクション１５４の順序付きセットは、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１５１ｎ－１を指し示す新しいブロック１５１ｎ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、別名二重支出としても知られている、前に妥当性確認されたトランザクションと同じ出力の割当てを行う場合、トランザクションを有効として受け入れないことを含む。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。

【００２５】

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１５４のプール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１５１ｎ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１５４のプール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プロングが最も長く成長しても、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。

【００２６】

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「生成トランザクジョン（generation transaction）」と呼ばれることもある。これは典型的に、新しいブロック１５１ｎ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１５１ｎ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「マイニング手数料」と呼ばれ、以下に説明する。

【００２７】

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を含む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。

【００２８】

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。

【００２９】

消費ユーザの役割を果たす複数の当事者１０３の各々のコンピュータ機器１０２もネットワーク１０１に接続されており、それぞれが個々のユーザまたは組織であり得る。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、（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）ブロックチェーン１５０のコピーを記憶するストレージエンティティとして動作し得る。

【００３０】

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求められる役割を果たさないので、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、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１０３ａおよびそのそれぞれのコンピュータ機器１０２ａ、ならびに第２の当事者１０３ｂおよびそのそれぞれのコンピュータ機器１０２ｂ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１０３ａは、本明細書ではアリスと呼ばれ、第２の当事者１０３ｂ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。

【００３１】

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。

【００３２】

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記憶媒体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。

【００３３】

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。

【００３４】

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ず、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。

【００３５】

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 with）、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。

【００３６】

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を含み、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込み特徴であってもよく、または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。

【００３７】

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１５４の順序付きセット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へと前方に伝搬する。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。

【００３８】

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１５４の順序付きプールに承認されると、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１５４の異なるプール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。

【００３９】

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ず、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）。

【００４０】

いくつかのブロックチェーンネットワークによって動作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場合、各トランザクションは、過去のトランザクションのシーケンス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することができる。このデータフィールドは、例えば、前のトランザクションＩＤがデータフィールドに含まれている場合、前のトランザクションを指し示し得る。

【００４１】

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。

【００４２】

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。

【００４３】

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「Tx_1」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取り、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「Tx_0」とラベル付けされている。Tx_0およびTx_1は、単なる任意のラベルである。それらは、Tx_0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Tx_1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意味するものではない。Tx_1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。

【００４４】

先行するトランザクションTx_0は、アリスが新しいトランザクションTx_1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、この場合、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、Tx_0およびTx_1を作成してネットワーク１０６に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、Tx_0をTx_1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「後続の」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」および「後の」、「親」および「子」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。

【００４５】

先行するトランザクションTx_0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$UTXO_0$とラベル付けされた特定のＵＴＸＯを含む。各ＵＴＸＯは、ＵＴＸＯによって表されるデジタル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、後続の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、後続の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、後続の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。

【００４６】

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分である。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分である。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。

【００４７】

つまり、図示の例では、Tx_0の出力２０３内の$UTXO_0$は、$UTXO_0$が償還されるために（厳密には、$UTXO_0$を償還しようとする後続のトランザクションが有効となるために）アリスの署名Sig P_Aを必要とするロックスクリプト［Checksig P_A］を含む。［Checksig P_A］は、アリスの公開鍵－秘密鍵ペアの公開鍵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秘密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＜Sig P_A＞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。

【００４８】

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：

〈Sig P_A〉　〈P_A〉　¦¦　[Checksig P_A]

ここで、「¦¦」は連結を表し、「〈…〉」はデータをスタックに置くことを意味し、「[…]」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き、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したアリスの署名を含むことを認証する。データの予想される部分自体（「メッセージ」）はまた、この認証を実行するために含まれる必要がある。実施形態では、署名されたデータは、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、含まれる必要はない）。

【００４９】

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、アリスが自身の秘密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。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。

【００５０】

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１５４の順序付きプールに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ことが重要である１つの理由である。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。

【００５１】

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。したがって、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であろう。

【００５２】

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ても、「後に残す」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$UTXO_0$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、$UTXO_0$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。

【００５３】

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、$UTXO_0$へのポインタがTx_1への唯一の入力であり、Tx_1は唯一の出力$UTXO_1$を有するとする。$UTXO_0$において指定されたデジタル資産の額が$UTXO_1$において指定された額よりも大きい場合、その差分は、$UTXO_1$を含むブロックを生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。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。

【００５４】

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、別の前方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。

【００５５】

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。

【００５６】

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）。

【００５７】

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。

【００５８】

レイヤ２オーバーレイネットワーク

ブロックチェーンネットワーク１０６は、すでに、インターネット１０１などのネットワーク上にオーバーレイされたオーバーレイネットワークの形態である。しかしながら、ブロックチェーン上にオーバーレイネットワークの別の層を重ねることも可能である。これは、例として図３に示されている。一例はメタネットである。そのようなネットワークは、基礎となるネットワークインフラストラクチャとしてのベースネットワーク１０１（例えば、インターネット）と、ベースネットワーク上にオーバーレイされたオーバーレイネットワークの第１の層としてのブロックチェーンネットワーク１０６とに対して、オーバーレイネットワークの第２の層であるという意味で、「レイヤ２」ネットワークとも呼ばれ得る。

【００５９】

オーバーレイネットワーク３００のこの第２の層は、ノード３０１およびエッジ３０２のネットワークを含む。ここで、ノード３０１は、図１および図２に関連して前述したようなブロックチェーンネットワーク１０６の層におけるノード１０４ではなく、メタネット（またはブロックチェーン上にオーバーレイされた他のそのようなネットワーク）の層におけるノードを指すことに留意されたい。メタネットネットワーク（または同様のもの）の各ノード３０１は、ブロックチェーン１５０上の異なるそれぞれのトランザクション１５２であり、それぞれが、それぞれのトランザクションのペイロードにデータを記憶する。したがって、メタネットネットワーク３００（または同様のもの）のノード３０１は、本明細書では、データ記憶ノードまたはデータ記憶トランザクションとも呼ばれ得る。そこに記憶されるデータは、データコンテンツおよび／またはメタデータ、典型的には両方を含み得る。出力ベースのモデルでは、それぞれのトランザクションの使用不可能な出力２０３に記憶され得る。この出力は、実行時にスクリプトを終了させるロックスクリプト内の１つまたは複数のオペコードによって使用不可能にされ得る。例えば、スクリプト言語を採用するシステムでは、これは、使用されるプロトコルに応じて、ＯＰ＿ＲＥＴＵＲＮオペコード、またはＯＰ＿ＲＥＴＵＲＮが後に続くＯＰ＿ＦＡＬＳＥであり得る。しかしながら、これは限定されず、当業者は、他のブロックチェーンシステム、例えば、アカウントベースのモデルを採用するシステムにおいてトランザクションに任意のペイロードデータを記憶するための他の技法を認識するであろう。以下では、出力ベースのモデルに関して例示し得るが、これに限定されるものではない。

【００６０】

レイヤ２オーバーレイネットワーク３００は、純粋にデータから構成され、完全に仮想であってもよいことに留意されたい。すなわち、ブロックチェーン１５０のトランザクション１５２上にオーバーレイされたオーバーレイネットワークとして、メタネットなどの

ノード３０１およびエッジ３０２は、基礎となるブロックチェーンネットワーク１０６または基礎となるネットワークインフラストラクチャ１０１の任意の特定の物理的アクタまたはエンティティに必ずしも対応しない。

【００６１】

データコンテンツは、例えば、テキスト、オーディオ、静止画もしくは動画、または他の文書を記憶するためにメタネット（または同様のもの）が使用されている実際のデータである。これは、ユーザコンテンツまたはユーザデータとも呼ばれ得る。メタデータは、ブロックチェーン１５０上にネットワークを重ねるためのプロトコルを実装する。トランザクション１５２の少なくともいくつかでは、それはデータコンテンツ間のリンクを定義する。これらは、ノード３０１間のエッジ３０２として説明することもできる。リンクまたはポインタは、例えば、親ノードのトランザクションＩＤ、$\mathrm{TxID}_{\mathrm{parent}}$、を含み得る。本明細書で参照される場合の「リンク」は、１つの可能性ではあるが、必ずしもハイパーテキストリンクを意味するものではないことに留意されたい。より一般的には、リンクは、現在のノード３０１がメタネット層（または、ブロックチェーン１５０の上に重ねられた他のそのようなオーバーレイ層）において関連している別のノード３０１を指し示す任意の形態のポインタを指すことができる。

【００６２】

便宜上、以下では、例としてメタネットに関して説明するが、これに限定されず、より一般的には、メタネットを参照する本明細書の箇所はいずれも、ブロックチェーン上にオーバーレイされた任意のオーバーレイネットワークで置き換えられ得ることが理解されよう。同様に、メタネットノードへのいかなる参照も、任意のオーバーレイネットワークノードまたはオーバーレイネットワークのデータストレージノードへの参照と置き換えられ得、メタネットリンクまたはエッジへのいかなる参照も、当該オーバーレイネットワークの層における任意のオーバーレイネットワークエッジまたはリンクへの参照と置き換えられ得る。

【００６３】

メタネットプロトコルは、公開ブロックチェーン上に記憶され、多くの使用事例のために様々なアプリケーションで使用されることができるオンチェーンデータを構造化するための方式および規格を定義する。プロトコルは、ノードおよびエッジを含むグラフ構造をブロックチェーントランザクションのセットから構築することができること、およびこれらの構造を使用して任意の性質のデータ（「コンテンツ」）を記憶し、伝達し、表現し、配信することができることを規定する。トランザクションをノードとして扱い、署名をトランザクション間に作成されたエッジとして扱うことによって、メタネットプロトコルは、図３に示すようなオンチェーングラフ構造の作成を可能にする。

【００６４】

図から分かるように、メタネット３００のノード３０１およびエッジ３０２はツリー構造を形成する。すなわち、親ノード３０１は、１つまたは複数の子ノード３０１にリンクされ、任意の所与の子３０１はそれ自体が、それ自体の１つまたは複数の子にリンクされた親であり得、以下同様である。本目的上、当該ツリー構造は、より広いツリーまたはグラフのサブセットにすぎない可能性があることに留意されたい。

【００６５】

図３はまた、ノード３０１およびその関連付けられたエッジ３０２がどのように更新され得るかを示す。トランザクションはブロックチェーン１５２上に不変的に記録されるので、メタネットノード３０１に対する更新は、新しいトランザクション１５２によって新しいインスタンス３０１’および対応するエッジ３０２’を作成することを必要とする。

【００６６】

図３の構造は、ネストされたドメイン、例えば、ウェブサイトおよびそのページの構造を含み得、ここで、「トップレベルドメイン」は、その下のサブドメインをカプセル化し、以下同様である。１つの機能鍵ドメイン（後述する、例えば、書き込み鍵、ファンディング鍵（funding key）または暗号化鍵のドメイン）は、これらの構造ドメインの多くに

及ぶことができる。

【００６７】

図３の円は、メタネットプロトコルのルールセットにしたがって作成される単なるトランザクションであるノードを表す。そのルールセットにしたがって作成されフォーマットされたトランザクション１５２Ｎの例を図４に示す。

【００６８】

図４の右側にあるトランザクション１５２Ｃは、メタネットの所与のノード３０１Ｃ（子）を実装するブロックチェーン１５０のトランザクション１５２を表す。図４の左上にあるトランザクション１５２Ｐは、メタネット層における子ノード１５２Ｃの親を実装するブロックチェーン１５０のトランザクションを表す。子ノードのトランザクション１５２Ｃは、ロック解除スクリプトを含み、ブロックチェーン１５０のファンディングトランザクション１５２Ｆの出力２０３を指し示す入力２０２を有する。言い換えれば、ファンディングトランザクション１５２Ｆの出力は、メタネットノード１５２Ｃの入力によって使用される。ファンディングトランザクション１５２ＦおよびメタネットＰ親トランザクション１５２Ｐは、必ずしも同じトランザクションではないことに留意されたい（ただし、それも除外されない）。

【００６９】

子トランザクション１５２Ｃは、ペイロード（ブロックチェーン層の観点からのペイロード）を保持する、例えばＯＰ＿ＲＥＴＵＲＮによって使用不可能にされた、使用不可能な出力２０３を含む。このペイロードは、メタネットのデータコンテンツ（「Data」）を含み得、それは、ハッシュおよび／または暗号化されるか、または単に生のデータ（「平文の」）であり得る。

【００７０】

子トランザクション１５２Ｃのペイロードは、メタネットネットワーク層のメタデータも含む。このメタデータは、少なくとも親トランザクション１５２Ｐのトランザクション識別子を含む。これは、メタネット層でリンク（エッジ）３０２を作成する。また、子ノード３０１Ｃに関連付けられた鍵P_{node}を含むことがメタネットプロトコルによって求められ得る。

【００７１】

ファンディングトランザクション１５２Ｆの出力２０３のロックスクリプトはまた、署名が子ノード１５２Ｃの入力２０２内のロック解除スクリプトに含まれることを必要とする。具体的には、この署名は、メタネット親に関連付けられた鍵P_{parent}を使用して署名された署名（すなわち、その鍵によって署名されたメッセージ）である必要がある。これは、ブロックチェーン層においてエッジ４０２（使用エッジと呼ばれることもある）を作成する。必要とされる署名が子トランザクション１５２Ｃの入力２０２内のロック解除スクリプトに含まれていない場合、子トランザクション１５２Ｃは、ブロックチェーンネットワーク１０６のノード１０４によって妥当性確認されないので、ブロックチェーンネットワーク１０６を通して伝搬されることも、ブロックチェーン１５０に記録されることもない。しかしながら、ファンディングトランザクション１５２Ｆは必ずしもメタネット親トランザクション１５２Ｐと同じブロックチェーントランザクション１５２ではないので、ブロックチェーン層使用エッジ４０２は、必ずしもメタネット層エッジ３０２と同じではないことに再度留意されたい。

【００７２】

図４は、トランザクション全体の抽象化として、メタネットトランザクションの特定の関連のある構成要素のみを概説する。これらの構成要素は、プロトコル識別子フラグに加えて、以下を含む：

- 公開鍵P_{node}
- 親公開鍵P_{Parent}の署名$SigP_{Parent}$
- ノード自体のトランザクションＩＤ　$TxID_{node}$
- ノードの親のトランザクションＩＤ　$TxID_{Parent}$

【００７３】
プレースホルダ＜Data＞は、一般に、メタネットノードトランザクションに含まれ得る任意のコンテンツデータを指す。また、いくつかのアプリケーションでは、暗号化鍵ｅｋでデータを暗号化することを望むことも予想され、この場合、トランザクションに含まれるデータは＜e（Data，ek）＞としてキャストされ、ここで、ｅ（　）は適切な暗号化関数である。
【００７４】
各メタネットノード３０１は、強力なバージョニングおよび許可制御がメタネットグラフによって継承されることを可能にするインデックスであるペア（P_{node}，$TxID_{node}$）によって一意に識別され得る。各メタネットノードは、それ自体（P_{node}，$TxID_{node}$）およびその親（P_{Parent}，$TxID_{parent}$）を識別するのに十分な情報を含むことも理解されたい。
【００７５】
メタネット子ノード３０１Ｃのトランザクションが親ノード３０１Ｐからの正しい入力署名$SigP_{Parent}$を含むことを保証するために、多くの場合、これを容易にする１つまたは複数のファンディングトランザクション１５２Ｆを作成することが望ましい場合があり、これを図４の左下に示す。
【００７６】
親鍵P_{parent}および／または子ノード鍵P_{node}は、ブロックチェーン１５０への子ノード３０１Ｃのデータの書き込みを認可する書き込み鍵とみなされ得る。
【００７７】
したがって、メタネットは、ブロックチェーン自体の基礎となる技術のみを使用して、そのようなデータのための許可および書き込みアクセス制御を符号化するような方法で、オンチェーンデータが構造化されることを可能にするプロトコルを提供する。したがって、メタネットプロトコルは、ユーザがユーザのオンチェーンコンテンツを証明可能に所有することを可能にするソリューションである。
【００７８】
メタネットプロトコルは、メタネットＤＡＧ（Metanet Direct Acyclic Graph）の作成を可能にするルールのセットを定義する。メタネットＤＡＧの単一のインスタンスは、メタネットツリーと呼ばれる。各メタネットツリーは、ルートノード（トップレベルノード）を有し、ルートノードを含む各メタネットノードは、１つまたは複数の子ノードを有することができる（例えば、再び図３を参照）。
【００７９】
このように、メタネットＤＡＧは、ツリーのグローバルコレクションになり、各ツリーは、それ自体のルートノードから開始し、それ自体の局所化された許可構造を有することができる。
【００８０】
メタネットノード３０１は、単に、メタネットプロトコルのルールセットに従うトランザクションである。親を有さないルートノードと、所与の子ノードが厳密に１つの親を有する子ノードという２つのタイプのノードが存在する。一実装形態によれば、メタネットノードの最も基本的なあるとライン構造は、トランザクションが以下の基準を満たすことを求める：
・ トランザクションは、少なくとも１つのＯＰ＿ＲＥＴＵＲＮ出力を有する。
・ ＯＰ＿ＲＥＴＵＲＮペイロードは以下を含む：
　－ メタネットフラグ
　－ ノードアドレスP_{node}
　－ 親トランザクションＩＤ　$TxID_{parent}$
・ 各トランザクションは、ルートノードを除き、親ノードによって署名された入力を含む。
【００８１】

上述したように、メタネットノードは、４つの要素を含むトランザクション１５２である：

・ P_{node}－ノードのアドレス
・ $TxID_{node}$－ノードのバージョン
・ P_{parent}－ノードの親のアドレス
・ $TxID_{parent}$－ノードの親のバージョン

【００８２】

メタネットエッジ３０２は、署名によって作成される。親ノードから子ノードへのエッジを作成するために、子ノードは、その親に関連付けられた鍵のペアを使用して署名されなければならず、$Sig\ P_{parent}$は、子ノードの入力に現れなければならない。

【００８３】

ファイル検証システム

以下では、ファイルの検証を可能にするためにメタネット（またはブロックチェーン上にオーバーレイされた他のそのようなグラフ構造）を使用することができる方法について説明する。例えば、これは、外部からの改ざん（ハッカー、ウイルス、または他のマルウェアなどによる）に対して、または正当なユーザによる意図しない挙動（例えば、意図しないファイル削除または修正）に対して保護を提供し得る。例えば、開示された技法は、ファイルに制限を設けて、ファイルの読み出し、修正、削除、または実行など、他のユーザによる特定のアクションを防止または許可するために、システム管理者によって使用され得る。

【００８４】

一例として、図５を参照されたい。図示のように、ルートノード３０１Ｒおよび複数のリーフノード３０１Ｌを含むツリー構造が作成される。ツリー構造は、例えば、複数の他のユーザ１０３のそれぞれのコンピュータ機器１０２による使用のための中央リソースとして、システム管理者のコンピュータ機器によって作成（および必要に応じて更新）され得る。代替的に、ツリー構造は、所与のユーザ自身のオペレーティングシステムまたはファイルシステムによって、ユーザ自身のコンピュータ機器１０２のプライベート記録として使用するために作成（および／または更新）されて得る。

【００８５】

ルート３０１Ｒ以外の各ノードは、１つのエッジ３０２によって親ノードに接続され、親ノードは、ルートノード３０１、または（例えば、図８～図１０に示すように）それ自体が別の親の子である中間親ノード３０１Ｉであり得る。すなわち、ツリーには２つ以上のレベルが存在し得る。各リーフノード３０１Ｌは、子ノード３０１Ｃである。図５に示すように２つの層のみの場合、ルートノード３０１Ｒは、リーフノード３０１Ｌの親ノード３０１Ｐである。ツリーに２つ以上のレベルがある場合（例えば、図８、図９または図１０のように）、中間レベルノード３０１Ｉは、各リーフノード３０１Ｌの親３０１Ｐであり、各中間レベルノード３０１Ｉの親は、それ自体が、ツリーのレベルの数に応じて、別のより高いレベルの中間ノード３０１Ｉまたはルートノード３０１Ｒであり得る。

【００８６】

各ノード３０１は、例えば、図３および図４に関連して説明したように、ブロックチェーン１５０の異なるトランザクション１５２である。各エッジ３０２は、ペアのノード３０１の間のリンクである。エッジ３０２は、それぞれの親ノードに関連付けられた秘密鍵で子ノードに暗号署名することによって作成され、それは、親の対応する公開鍵を使用して認証することができる。実施形態では、これらのエッジは、図３および図４に関連して説明したように生成される。すなわち、各ノード１５２は、少なくとも１つの入力２０２と少なくとも１つの出力２０３とを含む出力ベースモデル（例えばＵＴＸＯベースモデル）のトランザクションであり、親ノード３０１Ｐの秘密鍵で子ノード３０１Ｃの入力２０２に署名することによってエッジ３０２が生成される。子３０１Ｃをチェーン上に記録するためには、子３０１Ｃの入力は、ロックスクリプトが、ロックを解除し、したがってブロックチェーン上に記録するためにブロックチェーンネットワーク１０６によって子ノー

ドトランザクション３０１Ｃ／１５２Ｃが妥当性確認されるために親の署名を必要とするファンディングトランザクション１５２Ｆの出力２０３を指し示す。親ノードの鍵は、親ノード３０１Ｐの出力２０３内のペイロードに含まれることによって、それぞれの親ノード３０１Ｐに関連付けられ得る。また、親３０１ＰのトランザクションＩＤは、子３０１Ｃの出力内のペイロードに含まれ得る。再び例として図４を参照する。ペイロードは、それぞれのトランザクションの使用不可能な出力に含まれ得、例えば、使用されているプロトコルに応じて、ＯＰ＿ＲＥＴＵＲＮまたはＯＰ＿ＦＡＬＳＥおよびＯＰ＿ＲＥＴＵＲＮによって使用不可能にされ得る。実施形態では、オーバーレイプロトコルは、メタネットプロトコルであり得、したがって、ツリー構造は、メタネットグラフまたはその一部の形態をとり得る。

【００８７】

しかしながら、他のオーバーレイプロトコルでは、トランザクション１５２間にオーバーレイエッジ３０２を作成し、したがって、それらのトランザクションがツリーのノードを形成するツリー構造を形成するために他の方法が使用され得ることは除外されない。また、ツリー構造は、他のタイプのトランザクションモデルを使用して、例えば、アカウントベースのモデルにおけるスマートコントラクトによって、形成され得る。

【００８８】

本明細書で開示されるファイル検証規定によれば、少なくとも１つのそれぞれのファイルの記録が、リーフノード３０１Ｃのうちの１つ、複数、またはすべてのそれぞれのペイロードに記憶される。各リーフノード３０１Ｌはまた、オプションで、そのそれぞれのファイルに関連付けられたメタデータを含み得る。実施形態では、このメタデータは、１つまたは複数の許可のセットの表示、１つまたは複数の許可されたユーザの表示、および／または終了時間（例えば、有効期限）の表示を含み得る。

【００８９】

出力ベースのモデルでは、本目的のための所与のトランザクションの「ペイロード」は、トランザクション１５２（すなわち、ノード３０１）の１つまたは複数の出力（例えば、使用不可能な出力）に含まれ得ることに留意されたい。この用語は、単一の出力に限定されない。例えば、親トランザクションＩＤおよびファイルの記録は、必ずしも所与のリーフノード３０１Ｌの同じ出力２０３に含まれる必要はなく（ただし、これは１つの可能性である）、またはメタデータおよびファイル記録は、必ずしも同じ出力２０３に含まれる必要はない（ただし、これも確かに１つの可能な実装形態である）。

【００９０】

クライアントソフトウェア１０５は、少なくとも１つのユーザ１０３（例えば、アリスまたはボブ）のコンピュータ機器１０２上で実行され、リーフノード３０１Ｌのうちの１つにおけるファイルの記録を使用して、ユーザの機器上にローカルに保持されたファイルの何らかの現在の（意図された）インスタンスがファイルの真のコピーであるかどうかを検証するように構成される。より一般的には、本明細書に開示される方法は、任意のコンピュータ機器上の任意の当事者（必ずしもエンドユーザまたは消費者のみではない）によって実行され得るが、例示の目的上、本明細書の実施形態に関連してそのように説明され得る。実施形態では、検証は、ユーザのコンピュータ機器１０２上のオペレーティングシステムまたはファイルシステムなどのシステムソフトウェアの低レベルの信頼できる部分によって実施され得る。この場合、図１に示すクライアントソフトウェア１０５は、統合されたブロックチェーンアクセス機能を有するオペレーティングシステムまたはファイルシステムを含み得る。

【００９１】

所与のリーフノード３０１Ｌに含まれるファイルの記録は、ファイルの明示的なコピー、すなわち、ファイルの生の（変換されていない）コピー（「平文の」）を含み得る。代替的に、それは、ファイルの暗号化されたバージョンのような、ファイルの変換を含んでもよい。別の代替として、記録として使用される変換は、ファイルを含むプレイメージのハッシュを含んでもよく、プレイメージは、ファイルのみ、または別の要素と連結された

ファイルを含み得る。したがって、ファイルの「記録」は、記録からのファイルの回復が可能であることを必ずしも意味しないことに留意されたい。本発明の目的のために、記録は、ファイルのいくつかの現在の（意図される）インスタンス（ユーザのコンピュータ機器１０２上のローカルコピーなどの検証されているインスタンス）が、記録を作成するために使用されたファイルのインスタンスと同じであるかどうかを後にチェックすることを可能にする任意の指示である。例えば、ハッシュの場合、現在のインスタンスをハッシュし、そのハッシュをリーフノード３０１Ｌ内の記録上のファイルのハッシュと比較することができる。

【００９２】

所与のリーフノード３０１Ｌ内のツリー構造または個々のファイル記録は、例えば、システム管理者によって、またはユーザのオペレーティングシステムもしくはファイルシステムによって、最初作成され、および／またはその後更新され得る。

【００９３】

クライアントソフトウェア１０５は、実行されると、またはブロックチェーンネットワーク１０６から切断されている期間後にオンラインに戻ると、ブロックチェーン１５０を自動的に検査して、最新バージョンのツリー構造をダウンロードする。それは、ダウンロードされた構造、またはリーフノードのうちの少なくとも１つを使用して、コンピュータ機器１０２上でローカルに保持されるインスタンスなどのファイルの所与のインスタンス（「現在のインスタンス」）が正当であるかどうかを検証するように構成される。例えば、これは、定期的に行われてもよいし、ファイルに対して特定の動作（例えば、読み出し、書き込み、もしくは実行）またはウイルススキャンなどを行うためのユーザ１０３による要求などの特定のイベントに応答して行われてもよい。

【００９４】

検証を実行するために、ソフトウェア１０５は、少なくとも２つの構成チェックを実行し、検証されるべきファイルの現在のインスタンスについて両方が肯定的であることを要求する。

【００９５】

第１のチェックは、Ａ）ツリー構造が、有効グラフのオーバーレイプロトコル（例えば、メタネットプロトコル）の要件を満たすことである。これは、各子ノード３０１Ｃがそのそれぞれの親ノード３０１Ｐの鍵で署名されている（すなわち、それぞれの親に関連付けられた鍵から生成された署名を含む）ことを少なくともチェックすることを含む。つまり、図５に示す２レベルの場合、各リーフノード３０１Ｌは、ルートノード３０１Ｒの鍵によって署名されなければならない。または、図８～図１０のような３つ以上のレベルを有するツリーでは、各リーフノード３０１Ｌは、そのそれぞれの中間親３０１Ｉの鍵によって署名され、各中間親（それ自体がツリーのより上位の別の親３０１Ｐの子３０１Ｃである）は、階層内のその位置に応じて、ルートノード３０１Ｒまたは別の中間親３０１Ｉのいずれかの鍵によって署名される。

【００９６】

出力ベース（例えばＵＴＸＯベース）のトランザクションモデルの場合、署名の要件は、子トランザクション３０１Ｃ／１５２Ｃの入力２０２がそれぞれの親３０１Ｐの鍵によって署名されること、およびそれぞれの親３０１Ｐの鍵が親３０１Ｐ／１５２Ｐの出力２０３内のペイロードに含まれること（例えば、使用されているトランザクションプロトコルおよびスクリプトに応じて、ＯＰ＿ＲＥＴＵＲＮまたはＯＰ＿ＦＡＬＳＥおよびＯＰ＿ＲＥＴＵＲＮによって使用不可能にされるような、使用不可能な出力）であり得る。親鍵自体（および署名）もまた、子３０１Ｃ／１５２Ｃの入力に含まれることが要求され得る。

【００９７】

実施形態では、第１のチェックＡ）はまた、オーバーレイプロトコルの１つまたは複数のさらなる要件が満たされていること、例えば、子３０１Ｃ／１５２Ｃのペイロードがそれぞれの親３０１Ｐ／１５２Ｐのトランザクション ＩＤを含むことをチェックすることを含み得る。

【００９８】
第２のチェックは、Ｂ）ファイルの現在のインスタンス、すなわち、当該の現在のファイルの意図されたインスタンスが、関連するリーフノード３０１Ｌに保持されたファイルの記録と一致することのチェックである。記録がファイル自体の明示的なコピー（生のファイル「平文の」）を含む場合、このチェックは、ファイルを比較してそれらが同一であることをチェックすることを含む。または、記録がファイルのハッシュを含む場合、チェックは、現在の（例えば、ローカル）インスタンスをハッシュし、ハッシュを比較してそれらが同一であることをチェックすることを含む。または、記録がファイルの暗号化されたバージョンを含む場合、チェックは、同じ暗号化を使用して現在の（例えば、ローカル）インスタンスを暗号化し、暗号化されたバージョンが同一であることをチェックすること、または記録されたインスタンスを復号し、これが現在の（例えば、ローカル）インスタンスと一致することをチェックすることを含み得る。場合によっては、暗号化とハッシングの両方を使用することができる。
【００９９】
２つのチェックは、いずれの順序でも実行することができ、Ａ）およびＢ）ならびに「第１」および「第２」は、単なる任意のラベルにすぎないことに留意されたい。
【０１００】
チェックＡ）およびＢ）の両方が真であることが分かった場合にのみ、当該ファイルは、検証済み、すなわち有効であると宣言され得る（ただし、ファイルを検証するこの文脈における「有効」という用語は、ブロックチェーンネットワーク１０６のノード１０４によって適用されるノードプロトコルによるトランザクションの妥当性確認と混同されるべきではなく、両方が必要とされるが、それらは異なる概念である）。例えば、これは、ファイルが検証され、したがって使用するのに安全であることを宣言する、ユーザのコンピュータ機器１０２上のオペレーティングシステムまたはファイルシステムを含み得る。例えば、実施形態では、ユーザは、ファイルの読み出しまたは実行など、ファイルに対して何らかのアクションを実行することを要求している可能性があり、オペレーティングシステムまたはファイルシステムは、開示された検証にパスすることを条件としてのみ、このアクションの実行を許可する。
【０１０１】
実施形態では、オペレーティングシステムまたはファイルシステムは、それぞれのファイルの記録が保持される同じノード３０１Ｌから許可されたアクションを読み出すように構成される。この場合、オペレーティングシステムまたはファイルシステムは、追加の条件として、要求されたアクション（例えば、読み出し、書き込み、削除、または実行）がそれぞれのノードで指定された許可されたアクションの中にある場合にのみ、その要求されたアクションがファイルに対して実行されることを許可する。代替的にまたは追加的に、実施形態では、オペレーティングシステムまたはファイルシステムは、それぞれのファイルの記録が保持されているノード３０１Ｌから許可されたユーザを読み出すように構成される。この場合、オペレーティングシステムまたはファイルシステムは、要求されたアクション（例えば、読み出し、書き込み、削除または実行）を実行することを要求するユーザが許可されたユーザの中にある場合にのみ、そのアクションがファイルに対して実行されることを許可する。
【０１０２】
実施形態では、ファイルを検証するためのさらなる条件として、１つまたは複数の追加のチェックを適用する（例えば、オペレーティングシステムまたはファイルシステムによって実施する）こともできる。これらは、例えば、ルートノードが特定の信頼できるエンティティ、例えば、システム管理者によって署名されなければならないという第３のチェックＣ）、ファイルが記録された場合の記録ある子ノード３０１Ｃ自体がツリー内の別の子ノード３０１Ｃの親３０１Ｉであってはならない（すなわち、リーフノード３０１Ｌでなければならない）という第４のチェックＤ）、および／または記録に含まれる終了時間が現在の時間より前であってはならない（すなわち、記録が失効（expired）していない

）という第５のチェックＥ）を含み得る。チェックＣ）は、プロセスにおける追加の信頼レベルを提供する。チェックＤ）は、以下でより詳細に説明するように、新しいリーフノードを古いノードに付加することによって古い記録を無効化または更新するための機構を可能にする。チェックＥ）は、例えば、ユーザのコンピュータ１０２がオフラインになり、ブロックチェーン１５０から最新バージョンのツリーにしばらくの間アクセスできない場合に、ファイルを無制限に検証するために記録が使用されるのを防ぐ。

【０１０３】

すべてのこれらのチェックは、任意の順序で実行することができ、Ａ～Ｅ）および「第１」～「第５」は、単なる任意のラベルにすぎないことに留意されたい。

【０１０４】

図６は、本開示の技法の実施形態において採用され得る別の例示的な変形例を示す。図６は、マークルツリーとも呼ばれることがあるハッシュツリーを示す。ハッシュツリーのノード６０１は、図５などにおけるオーバーレイネットワークのツリー構造のノード３０１に対応しないことに留意されたい。ハッシュツリーは複数のハッシュリーフ６０１Ｌを含み、各々がツリーにおいて１つの親に対する子である。ツリーのルートにはハッシュルート６０１Ｒがある。

【０１０５】

ハッシュツリーを決定するために、リーフ６０１Ｌは、１つまたは複数のリーフレベルのセットに配置され、各セット内で、そのセットのリーフが互いに組み合わされ（例えば、連結され）、次いで、この組合せがハッシュされる。リーフレベルのセットが１つしかない場合（すなわち、リーフレベルのセットがすべてのリーフである場合）、このセットのハッシュは、単にハッシュルート６０１Ｒである。一方、リーフのセットが２つ以上存在する（すなわち、図示のように、リーフが異なるリーフレベルのセットに分割される）場合、結果として得られるハッシュは、１つまたは複数の第２のレベルのセットに配置される。第２のレベルの各セット内で、下のレベルからのハッシュが互いに組み合わされ（例えば、連結され）、次いで、この組合せがハッシュされる。第２のレベルのセットが１つしかない場合、結果として得られるハッシュはハッシュルート６０１である。第２のレベルのセットが２つ以上存在する場合、第３のレベルにおいて同じプロセスが繰り返され、以下同様に、ハッシュルートが決定されるまでツリーを上に進む。ハッシュツリーは、マークルツリーと呼ばれることもある（この場合、ハッシュルートはマークルルートと呼ばれ得、ハッシュリーフはマークルリーフと呼ばれる）。「マークルツリー」は、各セットのサイズが正確に２つのメンバーである（すなわち、ツリーの全体にわたってハッシュがペアで行われる）ことを意味すると解釈されることがあるが、この制限は、本明細書では必ずしも課されない。

【０１０６】

本明細書に開示されるいくつかの実施形態によれば、ハッシュツリー内の各リーフ６０１Ｌは、関連するファイル、例えば、特定のソフトウェアのファイル（実行可能ファイルおよび関連するデータファイルなど）のグループからの異なるそれぞれのファイルを含むプレイメージのハッシュである。次いで、ハッシュルート６０１Ｒが、このファイルのグループから計算される。このような実施形態では、オーバーレイネットワークツリー（例えば、メタネットツリー）のノード３０１のうちの所定の１つに記憶されたファイル記録は、複数のファイルのハッシュルート６０１Ｒであり得る（したがって、図６に示すツリー全体は、図５などに示すツリーのノード３０１のうちの１つに記憶される記録の算出方法を表していることになる）。次いで、チェックＢ）は、ユーザのコンピュータ機器１０２上のファイルの現在のローカルコピーから計算された対応するハッシュルートが、ブロックチェーン１０５上の対応するオーバーレイネットワークノード３０１に記憶されたハッシュルート６０１Ｒと同一であることをチェックすることを含む。このようにして、グループ内のファイルのうちのたった１つのインスタンスがユーザのコンピュータ機器１０２上で変更された場合であっても、ファイルのグループ全体がユーザに対して無効化される。例えば、この場合も同様に、このテストは、オペレーティングシステムまたはファイ

ルシステムによって実施され得る。
【０１０７】
そのような実施形態では、例えば、図５または図７～図１０のオーバーレイネットワークツリー構造内のノード３０１のうちの少なくともいくつかはそれぞれ、（ノード３０１ごとの個々のファイルではなく）ファイルの異なるそれぞれのグループを記録し得る。
【０１０８】
図７は、ツリー内のファイル記録を更新するための例示的な機構を示す。１つの可能性として、新しいリーフノード３０１Ｌ’を、前の記録を含むリーフノード３０１Ｌと同じ親ノード３０１Ｐ／Ｒ／Ｉに付加することができる。すなわち、新しいエッジ３０２’が、古い記録の親と同じ親ノード３０１Ｐ／Ｉ／Ｒから新しいノード３０１Ｌ’に作成される。この場合、リーフ記録３０１Ｌ、３０１Ｌ’の両方が有効のままである。しかしながら、別の可能性では、新しいリーフノード３０１Ｌ’は、古い記録を含むノード３０１（以前はリーフ３０１Ｌ）に付加される。すなわち、古い記録のノード３０１から新しいノード３０１Ｌ’に新しいエッジ３０２’が作成される。この場合、上述のチェックＤ）と組み合わせて、古い記録が無効にされ、新しい記録が唯一の有効なバージョンとして残される。
【０１０９】
図８～図１０は、３つ以上のレベルを有するツリー構造のいくつかの例を示す。図８では、ツリー構造は、ファイルフォルダ構造を模倣するために使用される。図９では、ルートノードは、システム管理者の鍵に関連付けられ、中間親３０１Ｉは、個々のユーザの鍵に関連付けられ得る。例えば、ユーザの鍵は、ツリーの特定のサブセットに対する許可を与えるために、システム管理者によってユーザに割り当てられ得る。図１０は、ルートノード３０１Ｒが会社全体のシステムに関連付けられ、異なる中間親３０１Ｉが会社内の異なるサブシステムに関連付けられる例を示す。
【０１１０】
上記の実施形態は、検証機構がオペレーティングシステムまたはファイルシステムの一部として組み込まれるファイルの管理（読み出し、実行など）に関して説明されているが、これは限定的ではないことに留意されたい。ユーザがファイルの完全性をチェックすることのみを望む可能性がある他の可能な使用事例が存在し、この場合、単純なアプリケーションまたは他のソフトウェア（例えば、ある種のアンチウィルス）もこの方法を実施することができる。このソフトウェアは、他のファイルの完全性を検証するように構成され、ソフトウェアは、望ましくないファイル変更、またはファイルもしくはプログラムを危険にさらしたハッカーの場合、ユーザに警告する。プロセスは、上述したものと同じであるが、オペレーティングシステムまたはファイルシステムに統合される必要はない。むしろ、それは、ユーザがインストールするだけのプログラムであって、ユーザが新しいツリーを初期化し、後にファイルの完全性をチェックする（および、ファイル変更の場合に警告する）ことを可能にするプログラムであり得る。
【０１１１】
図１１は、ＵＴＸＯベースのトランザクションモデルにおいて子ノード３０１Ｃを実装するための例示的なトランザクションを示す。
【０１１２】
図１２は、本明細書に開示される実施形態による例示的な方法のフローチャートである。例えば、方法は、オペレーティングシステムまたはファイルシステムによって自動的に実行され得る。ステップ１２１０において、方法は、ブロックチェーンネットワーク１０６を介してブロックチェーン１５０を検査して、オーバーレイネットワークツリー構造（例えば、メタネットグラフ）にアクセスすることを含む。このステップは、オペレーティングシステムまたはファイルシステムによって、定期的に、および／またはユーザが特定のファイルにアクセスすることを要求したとき、および／またはブロックチェーンネットワーク１０２にアクセスすることができない期間（例えば、１０６がインターネットに接続されていなかったため）の後にコンピュータ機器１０２がオンラインに戻ったときなど、特定のイベントに応答して実行され得る。オペレーティングシステムまたはファイルシ

ステムがツリー構造にアクセスするのが初めてではない場合、コンピュータ機器１０２上にローカルに記憶されたツリーの以前にダウンロードされたバージョンを更新し得る。ダウンロードされたバージョンは、コンピュータ機器１０２が再びオフラインになった場合に、後でファイルを検証するために使用され得る。したがって、いくつかのシナリオでは、ステップ１２１０は、ユーザのコンピュータ機器１０２のローカルストレージからローカルにダウンロードされたツリーのコピーにアクセスすることによって置き換えられ得る。

【０１１３】

ステップ１２２０において、方法は、アクセスされたツリー構造がオーバーレイネットワークプロトコル（例えば、メタネットプロトコル）の要件を満たすことをチェックすることを含む。ステップ１２３０において、方法は、検証されるべき何らかの所望のターゲットファイル、例えば、ユーザが何らかのアクション（例えば、読み出し、書き込み、削除、または実行）を実行するためにアクセスすることを要求しているファイルの記録を、ツリー構造のどのノード３０１が含むかを識別することを含む。ステップ１２２０および１２３０は、互いに対してどの順序で実行されてもよいことに留意されたい。

【０１１４】

ステップ１２４０において、方法は、識別されたノード３０１内のファイルの記録が、当該の現在のインスタンス、例えば、ユーザのコンピュータ機器１０２上のローカルバージョンと一致することをチェックすることを含む。例えば、これは、記録内のファイルハッシュを現在のインスタンスのハッシュと比較することを含み得る。

【０１１５】

ステップ１２５０において、方法は、実装形態に応じて課され得る任意の追加のオプションのチェックを実行することを含む。これは、ユーザがファイルにアクセスする許可を有すること、および／またはユーザが要求しているアクションが許可されていることをチェックすることを含み得る。さらなる代替または追加の例として、このステップは、前述のチェックＣ）～Ｅ）のいずれか１つ、いくつか、またはすべてを含み得る。

【０１１６】

チェック１２２０、１２４０、および１２５０は、互いに対してどの順序で実行されてもよいことに留意されたい。

【０１１７】

ステップ１２６０において、方法は、適用されたチェックのすべてが満たされたかどうかを決定する。これらすべてのチェックが満たされていることを条件としてのみ、方法はステップ１２７０に進み、使用するファイルの現在のインスタンスを検証する。例えば、これは、要求されたアクション（例えば、読み出し、書き込み、削除、または実行）がそれに対して実行されることを可能にすることを含み得る。しかしながら、チェックのいずれかが満たされない場合、方法は代わりにステップ１２８０に分岐し、ファイルは検証されていないと宣言される。好ましくは、これは、ファイルの現在の（意図された）インスタンスがこれ以上使用されないようにすることを含む。

【０１１８】

さらなる例示として、以下では、メタネットベースの実装形態の文脈において、上記の概念のいくつかの例をより詳細に説明する。

【０１１９】

メタネットベースのシステム

本明細書に開示される実施形態によれば、メタネットは、オンチェーンソフトウェア妥当性確認およびデータ完全性を可能にする方法を提供し、システムセキュリティを保証するために使用され得る。メタネットで保護されたシステムを使用したいシステム管理者は、メタネットルートノードを作成し、それをそのシステム（例えば、ターミナルまたはサーバ）に一意に関連付けることができる。システムごとに新しいメタネットルートが提供され得る。各ファイルは、標準プロセス（例えば、ｓｈａ２５６）を使用してハッシュされ得、ハッシュは、システムメタネットルートノード（例えば、図５）のメタネットノード子に記憶され得る。これは、本明細書ではファイルハッシュ（ＦＨ）ノードと呼ばれ得

る。各メタネットノードはまた、ファイルに対して許可された動作（読み出し、書き込み、実行）と、それらを実行することを許可されたユーザとを指定し得る。

【０１２０】

ファイル有効性

ファイルは、それに関連付けられたファイルハッシュノードが以下の３つの条件を満たす（加えて、ツリー自体がメタネットプロトコルを満たすものとして検証されている）場合に有効であるとみなされる：

Ｉ．ファイルのハッシュは、ファイルハッシュノードに示されるものである、

ＩＩ．ファイルハッシュノードはリーフノードである（子を有さない）、

ＩＩＩ．ファイルハッシュノードが失効していない。

【０１２１】

ファイルハッシュノードは、そのノードを表すトランザクションに書かれた有効期限が経過したとき、または、そのトランザクションに含まれるＵＴＸＯが使用された場合、失効する。有効期限を有するファイルハッシュノードは、ＵＴＸＯがその有効期限の前に使用されない限り、設定された有効期限まで有効である。有効期限のないファイルハッシュは、そのＵＴＸＯが使用されない限り有効である。

【０１２２】

第２の条件（ｉｉ）は、ノードの無効化または更新を可能にすることである。実施形態では、無効化のための少なくとも２つの可能な機構が存在することに留意されたい。１つは、トランザクションレベルでメタネットノードの出力２０３を割り当てる（例えば、使用する）ことであり、もう１つは、メタネットレベルでメタネットエッジ３０２によってそのノードに取り付けられた別のリーフノードを作成することである。

【０１２３】

ファイルを無効に設定するには、２つの方法が可能である：

ｉ．新しいノードを現在のファイルハッシュノードに付加する。このノードは、ハッシュもしくはランダムなハッシュ（例えば、プライバシーの理由で）の代わりに空のフィールドを含むことができる；または

ｉｉ．ファイルハッシュトランザクションに含まれるＵＴＸＯを使用する新しいトランザクション（必ずしもメタネットトランザクションではない）を生成する。

【０１２４】

すでに述べたように、ファイル有効性は、有効期限を設定して構成され得、この場合、ファイルは、（人間の時間またはブロックの高さで）設定された時間に達するまで有効である。その後、ファイルは、失効したノードに、同じハッシュであるが異なる有効期限を有する新しいファイルハッシュノードを付加することで、システム管理者によって更新されなければならない。追加のセキュリティ対策として、オペレーティングシステムは、失効したファイルハッシュノードに関連付けられたファイルを自動的に削除することができる。

【０１２５】

好ましくは、オペレーティングシステムは、ファイルハッシュノード有効性をチェックすることができるように、ツリーの更新されたバージョンをローカルに維持するであろう。ＵＴＸＯも使用する有効性チェックの場合には、保留中のトランザクションプール１５４の更新されたコピーも使用可能であるべきである。

【０１２６】

ソフトウェア有効性：ファイルハッシュノード（単一のファイルを有するソフトウェアの場合）またはマークルルート６０１Ｒを記憶するマークルルートノード（複数のファイルまたは依存性を有するソフトウェアの場合）が一般的なファイルの同じ条件を尊重する場合、ソフトウェアの一部は有効であると考えられる。加えて、マークルルートノードは、マークルツリー有効性を構築し、検証する。

【０１２７】

ファイル更新：保護されたファイルを更新する必要があるとき、システム管理者は新し

いファイルハッシュノードを生成する（マークルツリーノードを使用するソフトウェアのためのプロセスは同じである）。２つの可能なケースが識別され得（図７参照）、第１のケースは、ファイルの古いバージョンが依然として有効であることであり、第２のケースは、ファイルの古いバージョンが無効になる（そのファイルに対する動作が以降許可されない）ことである。

【０１２８】

第１の可能性：ファイルの更新されたバージョンのみが有効である。ファイルの最新バージョンのみが有効であるべき場合、新しいファイルハッシュノードは、ノードの古いバージョンに付加されなければならない。分岐の最新ノード（すなわち、ツリーの最深部）のみが有効であるため、中間バージョンは無効とみなされ、それらに基づいて試みられたあらゆる動作が無効化される。より高いレベルのセキュリティを実施するために、オペレーティングシステムは、無効になったファイルハッシュノードに関連付けられたファイルを自動的に削除することができる。

【０１２９】

第２の可能性：ファイルのより古いバージョンは有効なままである。ファイルの２つ以上のバージョンが有効であるべき場合、新しいファイルハッシュノードは、古いファイルハッシュノードではなくシステムルートノードに付加されなければならない（これらの２つのノードは兄弟である）。

【０１３０】

フォルダ構造の複製

これまで、説明のこの部分では、ファイルまたはファイルハッシュ（またはソフトウェアもしくはマークルツリー）ノード結合は純粋に論理的であり、ＳｙｓＭｅｔツリー内に有効なファイルハッシュノードがある場合、同じハッシュを有するファイルは、ファイルハッシュノードのプロパティおよび制限を継承する。しかしながら、このセクションでは、メタネットツリー構造にシステムフォルダ構造も結合することで、さらなるレベルのセキュリティを導入する。例えば、図８を参照されたい。これは、より高いレベルのセキュリティを保証する：ファイルが有効でなければならない（有効ファイルハッシュノード）だけでなく、使用されるために特定のフォルダ内になければならない。

【０１３１】

この実装形態では、メタネットツリーは、ツリー構造内のファイル位置を記述するシステムフォルダ構造を反映する。この技法が使用される場合、ファイルハッシュノードは、フォルダおよびサブフォルダを有するフォルダ構造にしたがって作成される。この技法は、フォルダノードという新しいタイプのメタネットノードの使用を必要とする。

【０１３２】

フォルダノード：フォルダノードは、システム内のフォルダ構造を模倣するために使用されるメタネットノードである。このタイプのノードはまた、ハッシュされた絶対フォルダパスおよび名前（例えば、ｓｈａ２５６（「C://folder/path/folder_name」））を含み得、ファイルハッシュノード（そのフォルダに含まれるファイル）および他のフォルダノード（そのサブフォルダ）にリンクされ得る。

【０１３３】

最も単純な形態では、フォルダノードは、システムフォルダ構造を複製するためのプレースホルダとして使用される。しかしながら、これらのノードは、フォルダおよびその内容に関する情報をメタデータの形態で含めることで、強化され得る。いくつかの実施形態では、ユーザＩＤリストおよび許可された動作リストなどの制御フィールドは、フォルダノードへの割当てができない場合があり、したがって、ファイル許可は、代わりに、ファイルハッシュまたはマークルルートノードを使用して管理され得る。より高度な許可構造は、ユーザノード（後述する）を使用して達成することができる。

【０１３４】

この実装形態では、ファイル有効性は以下のようにチェックされる：

Ｉ．ファイルが有効なファイルハッシュノードを有するかどうかをチェックする（前述

したように）、および
ＩＩ．ハッシュされたファイル絶対パスが、ファイルハッシュ親ノード（フォルダノード）に記憶されたものと一致することをチェックする。
【０１３５】
マルチユーザシステム
いくつかのユーザは、同じシステムまたはシステムの一部へのアクセスが与えられ、特定のファイルまたは特定のフォルダに対する動作を実行することを認可され得る。ファイルを読み出す許可を与えるような、より単純な場合、すべてのユーザＩＤをファイルハッシュまたはマークルルートノードにリストすることができる。しかしながら、より洗練された場合では、特別な管理およびより構造化された許可方法を必要とし得る。これらの場合、システム管理者は、ユーザ固有のメタネットノードを作成して、それらにノード所有権を譲渡することができる。これらのユーザ固有のノードは、以下ではユーザノードと呼ばれ、特定のユーザに関連付けられる。このタイプのファイルは、ファイルまたはフォルダ管理をユーザに転送するために使用される（Ｕｎｉｘコマンドｃｈｏｗｎと同様に）。
【０１３６】
ユーザノード：本明細書で開示される場合、ユーザノードは、毎回システム管理者に明示的な認可を要求することなく、ユーザがシステム上でいくつかの動作を実行することを認可するために使用され得るメタネットノードである。ユーザノードは、システム管理者によって作成され、システムルートノード３０１Ｒ（図９参照）またはフォルダノードに付加される。実施形態では、システム管理者は、ユーザノードにいくつかの制限を課すことができ、ユーザノードから作成された後続のノード（それらの子）は、同じ制限を継承する。これらの制限は、許可される動作のタイプ、有効期限、許可されるファイルタイプ、およびファイルロケーションを含み得る。
【０１３７】
ユーザは、ユーザノードで指定された制限を尊重することを条件に、新しいファイルおよびプログラムをそのシステムに追加することができる。例えば、ユーザノードは、ファイルの読み出しおよび書き込みを許可され得るが、それらの実行は許可されない可能性があり、または、ファイルは、所定のドライブまたはフォルダにおいてのみ実行され得る。これらの制限は、システムのセキュリティを維持するのを助ける。例えば、これらの制限なしに攻撃者がユーザ秘密鍵を盗んだ場合、攻撃者は、マシン全体を危険にさらす完全なシステム制御を行うことができるであろう。
【０１３８】
図９に示すようなユーザノードの概念は、例えば、図５に示すような単純なユーザ許可の概念とは異なる。図５では、システム管理者は、ユーザがこのノードのための特定の許可を有することを指定するだけであるが、図８に示すようなユーザノードのシステムでは、システム管理者は、ユーザが自身の新しいサブノードを作成することを可能にすることができる。例えば、ユーザは、特定のフォルダを所有すること、新しいファイルまたはサブフォルダを追加すること、および／またはそれらを再び削除することを許可され得る。
【０１３９】
実施形態では、ユーザノードが更新されると、そのすべての子ノードはデフォルトで無効にされ、それらはユーザノードの新しいバージョンに再作成され、付加される必要がある。これは非能率的に見えるかもしれないが、ユーザノードから作成されたノードが常に、親ユーザノードの同じプロパティ（例えば、許可、有効期限）を反映することを保証する。
【０１４０】
ユーザノードは、以下のように作成され得る：P_{parent}（P_{parent}はシステム管理者に属する）によって所有されるメタネットノードが、新しい所有者P_{node}（P_{node}はユーザＵに属する）として指定する新しいメタネットトランザクションを作成する。この時点から、Ｕは、新しいメタネットトリップワイヤノードを作成し、それらをその分岐に付加することできる。

【０１４１】
企業のための複数システム管理
企業では、会社のシステム管理者のみがすべてのマシンのファイルおよびプログラムを妥当性確認することができるような、より厳格なシステム管理が好まれ得る。このシステム管理は、提案されたアーキテクチャに新しい層を追加することによって達成され得る。この場合、各システムのシステムルートノードは、会社のシステム管理者が会社システム管理者ノードを使用することで作成される（図１０参照）。
【０１４２】
このシナリオでは、会社のシステム管理者は、任意の会社システムでファイルを更新または無効にすることを許可された唯一の者である。ユーザによって所有される公開鍵に関連付けられたユーザノードを作成することによって、特定のシステムまたはその一部をユーザに割り当てることができる。代替的に、ユーザはシステムにファイルを追加し、会社のシステム管理者に承認を求めることができる。システム管理者は、ファイルを検証および妥当性確認し、新しいファイルハッシュまたはマークルツリーノードをシステムツリーに追加する。新しいノードが追加されるとすぐに、システムはファイルを妥当性確認し、必要とされる動作を実行することができる。
【０１４３】
オンチェーンでのファイル記憶
ファイルハッシュノードは、そのハッシュだけではなく、ファイル全体をオンチェーンで記憶するように修正することができる。これらのノードは、以下ではファイルノードと呼ばれる。ファイルハッシュは妥当性確認および完全性を保証するが、安全な保管も必要な場合または他の手法によるファイルの更新が複雑で実行不可能な場合は、完全なファイルノードが有用であり得る。
【０１４４】
これは、システム管理者が、チェーン上にファイルをリリースすることによって一度だけファイルを公開し、次いで、すべてのユーザのコンピュータからデータをダウンロードできるので、有用である。リリースは、ユーザのコンピュータ１０２がオンラインになったときにオペレーティングシステムによって自動的に検出され、更新され得る（管理者は、ユーザに更新を警告する電子メールなどのメッセージを送信する必要がない）。例えば、これにより、外部サーバの必要なしに、ドライバー、ライブラリなどの更新方法としてブロックチェーンを利用することができる。ユーザのオペレーティングシステムは、既知のメタネットツリーに関連付けられたファイルの以前のバージョンを有するので、チェーン上のどこを見るべきかを知っている。つまり、ツリーが更新されたかどうかをチェックするために最新バージョンをダウンロードするとき、ファイルの最新バージョンを新しいリーフとみなす。
【０１４５】
安全な記憶：重要なデータは、ファイルノードを使用して記憶され、取り出されることができるので、ハードウェア損傷または窃盗の場合でも、ファイルを常に取り出すことができる。ブロックチェーンは公開されているため、オンチェーンにアップロードされたデータは、最先端の暗号化技法で暗号化され、鍵は安全に保管され得る。重要な機密ファイルは、好ましくは、少なくとも暗号化されていない形態では、オンチェーンに記憶されるべきではない。
【０１４６】
ファイル更新：いくつかのシステムを管理するシステム管理者は、ファイル更新を直接オンチェーンでリリースすることができる。システムは、最新のメタネットシステムツリーを取り出すたびに、これらのファイルも自動的に更新する。これは、例えば、セキュリティフィックスをリリースするか、または構成ファイルを更新するのに有用であり得る。システム管理者のみがシステムプリファレンスを構成および更新することができる場合、システム管理者は、ファイルノードにおいてｃｏｎｆｉｇファイルをオンチェーンでリリースすることによってそれを行うことができる。各システムは、システムツリーのそのコ

ピーを自動的に更新し、古いｃｏｎｆｉｇファイルを無効にし、有効なファイルノード（子を有さずかつ失効していないノード）内で見つけられた最新のバージョンをダウンロードする。

【０１４７】

メタネットトリップワイヤプロトコル

このセクションでは、プロトコルの特定の例として、メタネットトリップワイヤプロトコル（ＭＴＰ）を紹介する。次いでメタネットで保護されたシステムを作成するためにそれをどのように使用するかを説明する。ＭＴＰは、メタネットプロトコルの上に構築され、メタネットノードを特殊化し、オペレーティングシステムを保護するためのルールのセットを追加する。メタネットツリー構造は、任意の一般的なシステム（例えば、ラップトップ、サーバ、スマートフォン）内のファイルおよびフォルダを制御および妥当性確認するために使用される。

【０１４８】

ＭＴＰを統合するオペレーティングシステムは、任意のファイル動作（すなわち、読み出し、書き込み、実行）の前にそのメタネットシステムツリーをチェックし、ファイルが有効である場合にのみその動作を実行する。前のセクションですでに説明したように、ファイル自体またはそのハッシュが有効なメタネットトリップワイヤノードに記憶されている場合、ファイルは有効である（ノードは、前述したプロパティを尊重する場合に有効である）。ファイルのコピーまたはそのハッシュは常にブロックチェーンに記憶されているので、ファイル内の単一の変更がＭＴＰによって検出され、動作の発生を拒否するイベントフラグがトリガされる。

【０１４９】

複数のファイル（例えば、実行可能ファイル、構成ファイル、ライブラリ、データ）が実行されるのを必要とするソフトウェアの場合のように、洗練されたトリップワイヤ機構が可能である。これらの場合、各ファイルハッシュは、マークルルートノードに記憶されるマークルツリーを作成するために使用される。マークルツリーは「トリップワイヤ」として機能し、これらのファイルのいずれかにおける単一の変更がそれをオフに設定し、ソフトウェアが実行されるのを防ぐイベントフラグをトリガする。これは、マルウェア攻撃を防ぐためだけでなく、例えば、ライセンス契約が成立しているとき、またはソフトウェアおよびライブラリの特定のセットを使用して特定の結果が得られることをユーザが証明したいときにも特に有用である。

【０１５０】

メタネットトリップワイヤノードおよびエッジ

メタネットノードは、ファイルのハッシュまたはファイル自体を記憶するために使用される。それらはまた、ユーザに許可を与え、ファイルおよびフォルダ情報を記憶するために使用される。これらのノードは、メタネットエッジを使用してリンクされ、階層制御構造を形成する。新しいメタネットトリップワイヤノードおよび新しいメタネットトリップワイヤエッジを作成するためのプロセスについてここで説明する。

【０１５１】

メタネットトリップワイヤノード：メタネットトリップワイヤ（ＭＴ）ノードは、ファイルを妥当性確認または記憶し、許可、ユーザ、およびフォルダを管理するように特殊化されたメタネットノードである。メタネットノードは、メタネットプロトコルに従うブロックチェーントランザクションである。メタネットノードは、ＯＰ＿ＲＥＴＵＲＮの後にメタネットフラグ（４バイトのプレフィックス）をトランザクションに含めることによって作成される。メタネットフラグの後に、トランザクションは、ハッシュのようなファイルの有効性に関する情報と、有効期限または許可された動作のリストなどの任意の他のオプションのパラメータとを含む。各ノードには、システム管理者によって制御される新しい公開鍵P_{node}が割り当てられ、P_{node}に関連付けられた秘密鍵は、そのノードの子を作成する必要がある。したがって、秘密鍵の所有者のみが新しいファイルを更新または追加することができる。ユーザノードを作成することは、公開鍵P_{node}がシステムユーザによ

って制御される新しいＭＴノードを作成することを意味し、これにより、ユーザは、新しいファイルまたはフォルダを作成することができる。異なるタイプのＭＴノードが上記で説明されている。

【０１５２】

メタネットトリップワイヤエッジ：メタネットトリップワイヤエッジは、標準メタネットエッジのルールに従う：メタネットエッジは、親メタネットノードおよび子メタネットノードという２つのメタネットノード間の関連付けである（ＭＴＰでは、これらはＭＴノードである）。エッジは、親の署名が別のメタネットノード（子）の入力に現れるときに作成される。親のみがエッジを作成することができ、したがって、それ自体を子にリンクすることができる。エッジは、異なるノード間のリンクを作成するために使用される。例えば、システムルートノードおよびそのファイルハッシュとユーザノードとの間、およびファイルハッシュノードと同じファイルのより新しいバージョンとの間には常にリンクがある。メタネットトリップワイヤトランザクションの例を図１１に示す。

【０１５３】

ノードタイプ

このセクションでは、ＭＴノードがリストされ、詳細に説明される。すべてのＭＴノードは、ＯＰ＿ＲＥＴＵＲＮ（例えば、図１１）に含まれる追加のフィールドを有するメタネットノードである。

【０１５４】

システムルートノード：システムルートノードは、親公開鍵を含まない。しかしながら、スプーフィング攻撃を防ぐために、システム管理者によって作成される。＜file hash＞フィールドには、システムハッシュまたはシステムを記述する一意の識別子（例えば、ＭＡＣアドレス、ドライブシリアル番号、ＣＰＵ情報などの組合せ）が記憶される。システムルートノードの親は、システム管理者またはシステム所有者である。

【０１５５】

ユーザノード：ユーザノードは、システム管理者からユーザに、またはユーザから別のユーザにノード所有権を譲渡するために使用される。新しいユーザは＜user＞フィールドで指定され、許可された動作のリストは＜permitted operations list＞フィールドで指定される。ユーザノードの親は、システムルートノード、別のユーザノード、またはフォルダノードであり得る。

【０１５６】

ファイルハッシュノード：ファイルハッシュノードは、ファイルのハッシュを含む。＜file hash＞フィールドには、ハッシュが記憶される。追加的に、＜path hash＞フィールドでパスを指定することができる。ノードを有するファイルの親は、システムルートノード、フォルダノードもしくはユーザノード、または更新の場合には別のファイルハッシュノードであり得る。

【０１５７】

マークルルートノード：互いにリンクされる必要があるファイルのグループ（例えば、ソフトウェアの場合、実行可能ファイル、ｃｏｎｆｉｇファイル、いくつかのライブラリおよびデータが必要とされ得る）は、マークルツリーにおいて互いにハッシュされ得、マークルルートは、マークルルートノードに記憶される（ファイルがノードを有するのと同様であるが、個々のファイルのハッシュだけではなくファイルのグループのマークルルートを記憶する）。マークルルートは＜ｆｉｌｅ ｈａｓｈ＞フィールドに記憶される。マークルツリーに含まれるファイルの順序付きリスト（絶対パスを含む）は、＜file list＞フィールドに記憶される。

【０１５８】

マークルルートノードの親は、システムルートノード、フォルダノード、もしくはユーザノード、または更新の場合にはファイルハッシュノードもしくは別のマークルルートノードであり得る。

【０１５９】

ファイルノード：ファイルノードは、オンチェーンでファイル全体を記憶する。＜file＞フィールドは、ファイルのバイトコードを含み、オプションで、＜file hash＞フィールドは、ファイルのハッシュを記憶する（ＦｉｌｅＨａｓｈノードと同様に）。

【０１６０】

ファイルノードの親は、システムルートノード、フォルダノード、もしくはユーザノード、または更新の場合には別のファイルノードであり得る。

【０１６１】

フォルダノード：フォルダノードは、システムフォルダ構造を記述するために使用される。＜path hash＞フィールドには、絶対パスのハッシュが記憶される。このタイプのノードは、システムフォルダ構造がシステムツリーにおいて複製される場合に使用される。フォルダノードの親は、システムルートノード、別のフォルダノード、またはユーザノードであり得る。

【０１６２】

ツリー実装

ＭＴプロトコルは、ツリー初期化とノード管理という２つのフェーズで構成される。以下では、これらのフェーズについて説明し、それらの完了および実行に関わる段階的なプロセスについて論じる。

【０１６３】

ツリー初期化：ツリー初期化フェーズでは、メタネットシステムツリーが作成され、システムルートが初期化される。以下のステップが実行される：

１．システム管理者が選択される。

２．システム管理者がルート鍵P_{admin}を選択する。

３．P_{admin}は、システムルートノードを作成するために使用され、以下を含む：

Ｉ．システムの一意の識別子、および

ＩＩ．初期ノードを作成するために使用されることとなる新しい鍵P_{root}（相対的な秘密鍵はシステム管理者によって制御される）。

４．システム管理者は、システムルートノードのトランザクションＩＤおよび鍵P_{admin}（代替的に、P_{root}が使用され得る）を使用してシステムを初期化する。

【０１６４】

ノード管理：ノード管理フェーズは、ＭＴノード挿入、更新、および削除という３つの主要な動作を含む。

【０１６５】

挿入：新しいＭＴノードは、上述したルールにしたがってシステムルートまたは他のノードに付加することができる。新しいノードは次のように作成される：

１．新しいトランザクションが作成され、ＯＰ＿ＲＥＴＵＲＮの後に必要なフィールド（例えば、メタネットフラグ、ユーザｉｄリスト、有効期限、…）が挿入される。

２．親ノードは、新しい鍵P_{node}、親ノードのトランザクションＩＤを追加し、トランザクションに署名する。P_{node}に対する秘密鍵は、新しいノードがユーザノードである場合を除き、システム管理者によって制御される。この場合、秘密鍵はユーザによって制御され、このノードの将来の子はユーザによって署名される。

【０１６６】

更新：既存のノードのより新しいバージョンを作成する（例：ハッシュを更新する）ことでノードが更新される。ノードが、より古いバージョンに付加される場合（これは、より古いバージョンの子である）、新しいバージョンのみが有効である（通常、この場合、親の同じ鍵P_{node}が使用される）。新しいバージョンが、より古いノードの同じ親に付加される場合（それらは兄弟である）、両方のバージョンが有効である（通常、この場合、新しい鍵P_{node}が使用される）。

【０１６７】

削除：ノードは、無効なハッシュを持つ新しいノードを付加するか、そのＵＴＸＯを使用することによって削除される（図１１のｘ　ＢＳＶを参照）。P_{node}に関連付けられた

秘密鍵を知っているシステム管理者またはユーザのみが、より新しいバージョンのノードを付加することまたはＵＴＸＯを使用することができる。

【０１６８】

オペレーティングシステムでのツリー管理

オペレーティングシステムは、ルートノードの鍵P_{root}およびトランザクションＩＤを（安全な方法で）記憶することによってツリーをセットアップするシステム管理者によって初期化される。システムが初期化されるとき、１つまたは複数のマイニングノードからメタネットシステムツリーの最新バージョンをダウンロードする。

【０１６９】

ファイルに対する動作を実行する必要があるたびに、ファイルがシステムツリーと照合され、ファイルが有効な場合にのみ動作が完了する。システム管理者またはユーザがファイルまたはソフトウェアを更新すると、メタネットシステムツリーの最新バージョンをダウンロードする必要がある。セキュリティとスピードを高めるために、システムツリーを定期的に更新し、すべてのファイルを予防的にチェックすることができる。

【０１７０】

実施例

１．新しいシステムが初期化される。これは、メタネットシステムツリーをダウンロードする。

２．ユーザがファイルを開こうとする。

３．ファイルがメタネットシステムツリーと照合される。

ａ．ファイルが有効であり、ユーザがそのファイルを開く許可を有している場合：

－　ファイルが開かれる。

ｂ．ファイルが有効でない場合（例えば、撃者がマルウェアを挿入した場合）、またはユーザがファイルを開く許可を有していない場合：

－　システムはイベントフラグをトリガするトリップワイヤを作動させ、動作は拒否される。

【０１７１】

上記の実施形態は、例としてのみ記載されていることが理解されよう。

【０１７２】

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、上記の説明は概して任意のブロックチェーンに適用され得ることが理解されよう。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への上記のあらゆる言及は、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への言及に置き換えることができる。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に、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の説明されたプロパティのうちのいくつかまたはすべてを共有し得る。

【０１７３】

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を作成し、公開し、伝搬し、記憶するという説明された機能の少なくともすべてを実行する。これらの機能のすべてではないが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公開することはしないが、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は、好ましいビットコインネットワーク１０６のノードとはみなされないことを想起されたい）。

【０１７４】
本発明の非優先実施形態では、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を作成し、公開し、伝搬し、記憶する機能のすべてではないが、少なくとも１つまたはいくつかを実行し得る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、「ノード」は、ブロック１５１を作成および公開するが、それらのブロック１５１を記憶および／または他のノードに伝搬しないように構成される、ネットワークエンティティを指すために使用され得る。
【０１７５】
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」という用語に置き換えることができ、そのようなエンティティ／要素は、ブロックを作成し、公開し、伝搬し、記憶する役割の一部またはすべて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述した方法と同じ方法でハードウェアに実装され得る。
【０１７６】
さらにより一般的には、下記ステートメントのうちのいずれか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。
【０１７７】
ステートメント１：ブロックチェーン上にオーバーレイされたツリー構造を使用する方法であって、ツリー構造は、複数のノードとノード間のエッジとを含み、各ノードは、ブロックチェーン上に記録された異なるトランザクションであり、各エッジは、それぞれの子ノードからそれぞれの親ノードに接続し、エッジは、トランザクションＩＤを含む各トランザクションによって形成され、各子ノードは、子ノードのそれぞれのペイロードにおいてそれぞれの親ノードのトランザクションＩＤを指定し、親ノードのうちの１つは、ツリー構造のルートノードであり、方法は、ブロックチェーンを検査して、ツリー構造の少なくとも一部を識別するステップであって、ターゲット子ノードのそれぞれのペイロードにおいてファイルの記録を含む子ノードのうちのターゲット子ノードを少なくとも識別することと、ツリー構造を通ってターゲット子ノードからルートノードに戻る１つまたは複数のエッジを含むパスを識別することとを含む、ステップと、チェックを実行するステップであって、Ａ）ターゲット子ノードからルートノードに戻る識別されたパスに沿ったエッジごとに、それぞれの子ノードがそれぞれの親ノードに関連付けられた鍵によって署名されていることをチェックすることと、Ｂ）ファイルの現在のインスタンスがターゲット子ノードに含まれる記録と一致することをチェックすることとを含む、ステップと、少なくともチェックＡ）およびＢ）が肯定的であるという条件で、ファイルの現在のインスタンスを検証するステップとを含む、方法。
【０１７８】
ステートメント２：上記検証するステップを条件として、ユーザがファイルの現在のインスタンスに対して要求されたアクションを実行することを可能にするステップを含む、ステートメント１の方法。
【０１７９】
ステートメント３：方法は、ユーザのコンピュータ機器上のオペレーティングシステムまたはファイルシステムによって実行され、オペレーティングシステムまたはファイルシステムは、ユーザがコンピュータ機器を使用して要求されたアクションを実行することを可能にする前に上記チェックを実施するように構成される、ステートメント２の方法。
【０１８０】
ステートメント４：要求されたアクションは、ファイルを読み出すこと、ファイルを修正すること、ファイルを実行すること、またはファイルを削除することのうちの１つを含む、ステートメント２または３の方法。
【０１８１】

ステートメント５：ターゲット子ノードのペイロードは、１つまたは複数の許可されたアクションの指示をさらに含み、オペレーティングシステムまたはファイルシステムは、要求されたアクションがターゲット子ノードにおいて示された許可されたアクションのうちの１つであるというさらなる条件でのみ、要求されたアクションを可能にすることを実行するように構成される、ステートメント２、３または４の方法。

【０１８２】

ステートメント６：１つまたは複数の許可されたアクションは、ファイルを読み出すこと、ファイルを修正すること、ファイルを実行すること、またはファイルを削除することのうちの１つまたは複数を含む、ステートメント５の方法。

【０１８３】

ステートメント７：ターゲット子ノードのペイロードは、１つまたは複数の許可されたユーザの指示をさらに含み、オペレーティングシステムまたはファイルシステムは、アクションを要求するユーザがターゲット子ノードのペイロードにおいて示された許可されたユーザであるというさらなる条件でのみ、要求されたアクションを可能にすることを実行するように構成される、ステートメント２から６のいずれかの方法。

【０１８４】

ステートメント８：方法は、ファイルの現在のインスタンスの完全性を検証するために、アンチウィルスソフトウェアまたは別のアプリケーションによって実行される、ステートメント１または２の方法。

【０１８５】

ステートメント９：それぞれの親ノードに関連付けられた鍵は、親ノードのペイロードに含まれることによって親に関連付けられる、先行ステートメントのいずれかの方法。

【０１８６】

ステートメント１０：各トランザクションは、ロックスクリプトを含む少なくとも１つの出力と、それぞれの他のトランザクションの出力を指し示し、かつそれぞれの他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を含む少なくとも１つの入力とを含む、先行請求項のいずれかにの方法。

【０１８７】

ステートメント１１：各子の入力は、それぞれの親の鍵によって署名され、Ａ）は、上記パスに沿った各子ノードの入力が、それぞれの親の鍵によって署名されていることをチェックすることを含む、ステートメント１０の方法。

【０１８８】

ステートメント１２：各子ノードのペイロードは、それぞれの子ノードの出力のうちの１つまたは複数に含まれる、ステートメント１０または１１の方法。

【０１８９】

ステートメント１３：上記記録は、少なくともファイルの明示的なコピーを含み、Ｂ）は、ファイルの現在のインスタンスが、ターゲット子ノードのペイロードに記録されたコピーと同じであることをチェックすることを含む、先行ステートメントのいずれかの方法。

【０１９０】

ステートメント１４：上記記録は、少なくともプレイメージのハッシュを含み、プレイメージはファイルを含み、Ｂ）は、ファイルの現在のインスタンスを含むプレイメージの現在のインスタンスのハッシュが、ターゲット子ノードのペイロードに記録されたハッシュと同じであることを少なくともチェックすることを含む、先行ステートメントのいずれかの方法。

【０１９１】

ステートメント１５：上記記録は、ハッシュツリーのリーフとして複数のファイルから生成された、ハッシュツリーのハッシュツリールートを含み、上記ファイルは、複数のファイルのうちの１つであり、Ｂ）は、複数のファイルの現在のインスタンスから計算されたハッシュルートが、ターゲット子ノードに記録されたハッシュルートと同じであることをチェックすることを含む、先行ステートメントのいずれかの方法。

【０１９２】
ステートメント１６：上記親ノードのうちの少なくとも１つは、上記親ノードのうちの別の親ノードの子ノードである中間親ノードである、先行ステートメントのいずれかの方法。
【０１９３】
ステートメント１７：上記少なくとも１つの中間親ノードは、上記パスが２つ以上のエッジを含むように、ターゲット子ノードとルートノードとの間の上記パスに沿った少なくとも１つのノードを含む、ステートメント１６の方法。
【０１９４】
ステートメント１８：上記少なくとも１つの中間親ノードは、ファイルを含むフォルダを表すフォルダノードを含む、ステートメント１７の方法。
【０１９５】
ステートメント１９：上記少なくとも１つの中間親ノードは、ユーザノードを含み、ルートノードに関連付けられた鍵は、システム管理者の鍵であり、ユーザノードに関連付けられた鍵は、ユーザの鍵である、ステートメント１７の方法。
【０１９６】
ステートメント２０：上記チェックは、Ｃ）ルートノードが信頼できるエンティティによって署名されていることをチェックするという追加のチェックを含み、上記検証することは、チェックＣ）が肯定的であることをさらに条件とする、先行ステートメントのいずれかの方法。
【０１９７】
ステートメント２１：上記信頼できるエンティティはシステム管理者である、ステートメント２０の方法。
【０１９８】
ステートメント２２：ルートノードの入力は、信頼できるエンティティによって署名され、チェックＣ）は、ルートノードの入力が信頼できるエンティティによって署名されていることをチェックすることを含む、少なくともステートメント１０に従属するステートメント２０または２１の方法。
【０１９９】
ステートメント２３：上記チェックは、Ｄ）ターゲット子ノードがグラフ構造の他の子ノードの親ノードでもないことをチェックするというさらなるチェックを含み、上記検証することは、チェックＤ）が、ターゲット子ノードが他の子ノードのどの子ノードの親でもないと決定することをさらに条件とする、先行ステートメントのいずれかの方法。
【０２００】
ステートメント２４：新しいエッジを介してターゲット子ノードに新しい子ノードを付加することによって、ブロックチェーン上のグラフ構造に記録されたファイルを後に更新するステップを含み、新しい子ノードは、新しい子ノードのペイロードにおいて更新されたファイルの記録を含む、ステートメント２３の方法。
【０２０１】
ステートメント２５：ターゲット子ノードは、記録の終了時間をさらに指定し、上記チェックは、Ｅ）現在の時間がターゲット子ノードにおいて指定された終了時間よりも遅くないという点で記録が失効していないことをチェックするという別のチェックを含み、上記検証は、上記チェックＥ）が、現在の時間が失効していないと決定することをさらに条件とする、先行ステートメントのいずれかの方法。
【０２０２】
ステートメント２６：ツリー構造はメタネットグラフである、先行ステートメントのいずれかの方法。
【０２０３】
ステートメント２７：コンピュータシステムであって、１つまたは複数の処理ユニットを含む処理装置と、１つまたは複数のメモリユニットを含むメモリとを備え、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、処理装置上で実行されると、先行ステートメントのいずれかの動作を実行するように構成されている、システム。

【０２０４】

ステートメント２８：コンピュータ可読ストレージ上に具現化されたコンピュータプログラムであって、コンピュータプログラムは、１つまたは複数の処理ユニット上で実行されると、ステートメント１から２７のいずれかの動作を実行するように構成されたコードを含む、コンピュータプログラム。

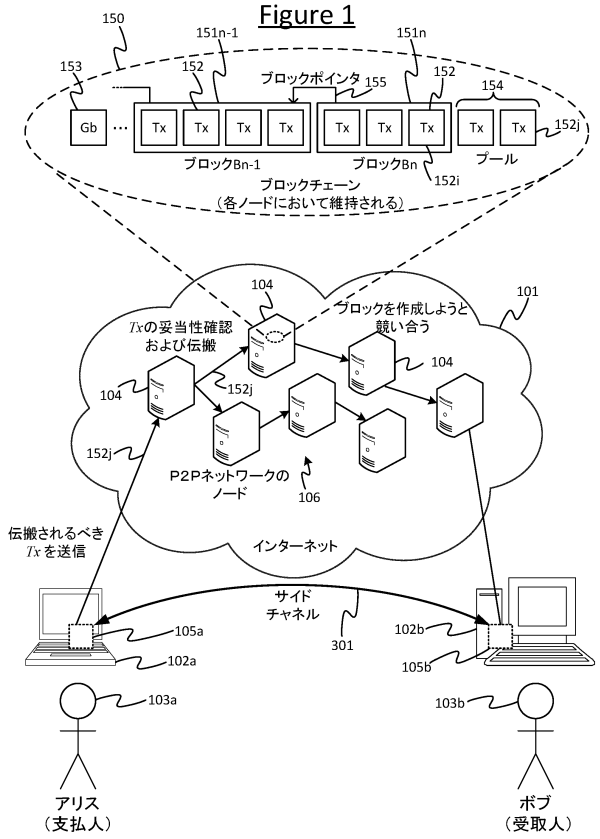
【０２０５】

開示された技術の他の変形または応用は、本明細書の開示が与えられると、当業者に明らかになるであろう。本開示の範囲は、上述の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。

【図面】

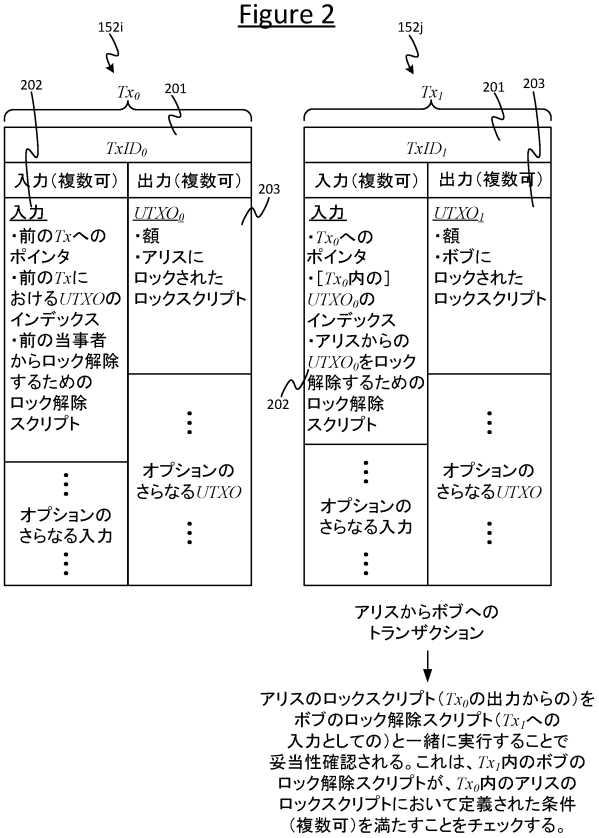
【図１】

Figure 1



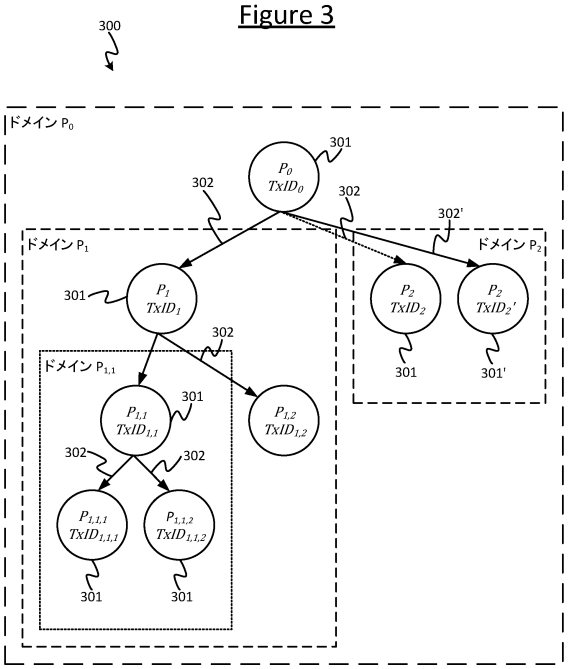
【図２】

Figure 2



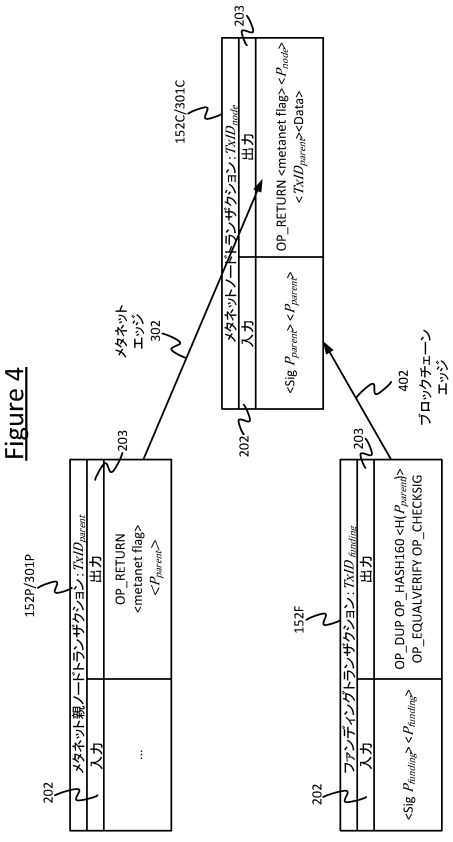
【図３】

Figure 3



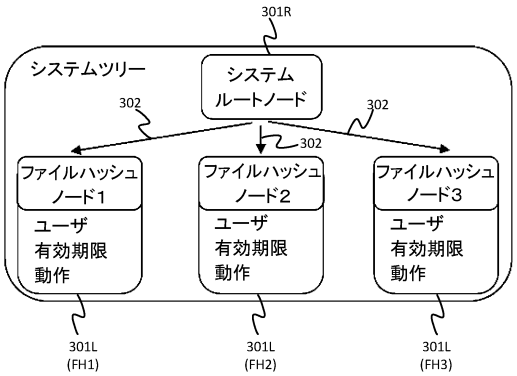
【図４】

Figure 4



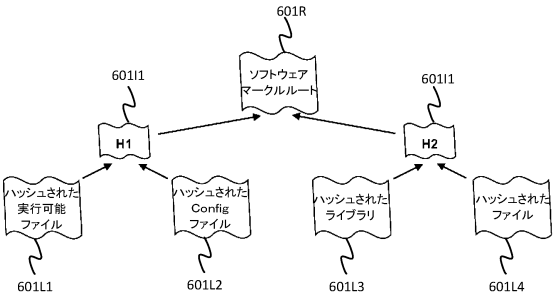
【図５】

Figure 5



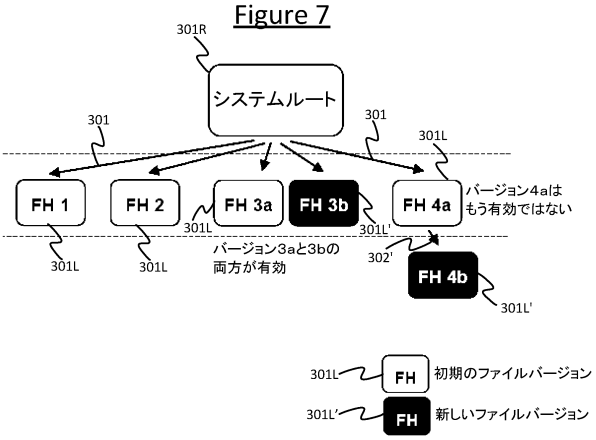
【図６】

Figure 6



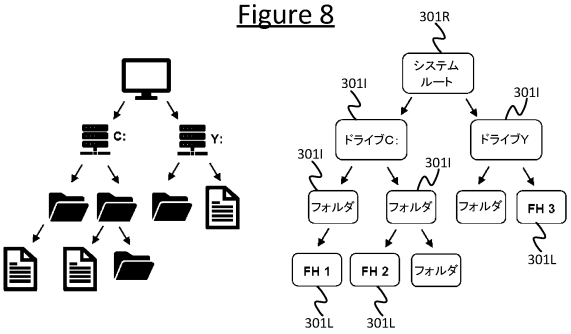
【図７】

Figure 7



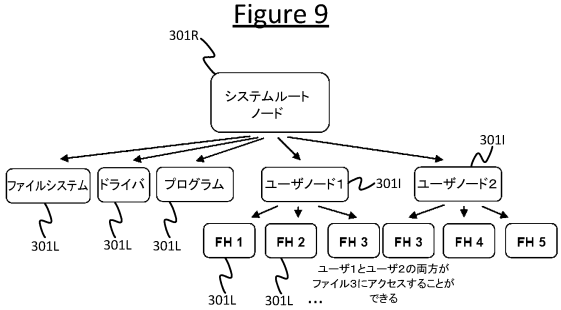
【図８】

Figure 8



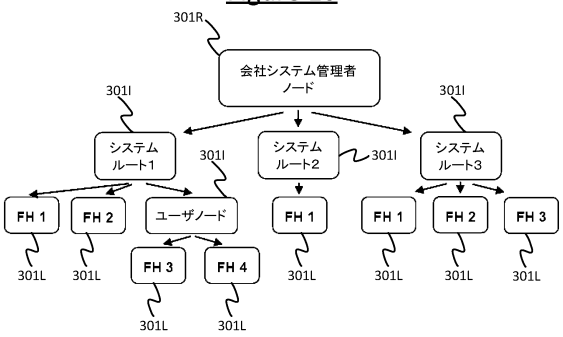
【図９】

Figure 9



【図１０】

Figure 10



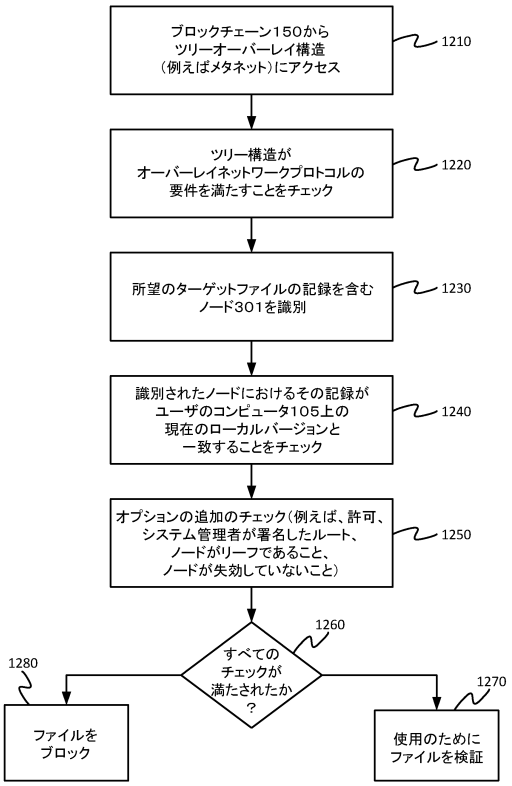
【図１１】

Figure 11

$TxID_{node}$			
入力		出力	
値	スクリプト	値	スクリプト
x BSV	$<Sig\ P_{parent}>$ $<P_{parent}>$	0 BSV	OP_FALSE OP_RETURN <metanet flag> $<P_{node}>$ $<TxID_{parent}>$ ノードを特殊化するために使用される オプションのパラメータ。例えば: <expiry date> <user ids list> <permitted operations list> <file hash> <file> <path hash> <…>
		x BSV	OP_DUP OP_HASH160 $<H_{160}(P_{node})>$ OP_EQUALVERIFY OP_CHECKSIG

【図１２】

Figure 12



フロントページの続き

(72)発明者　ライト，クレイグ，スティーヴン
イギリス ダブリュー１ダブリュー ８エーピー ロンドン マーケット プレイス ３０ エヌチェーン ライセンシング アーゲー 内

審査官　平井 誠

(56)参考文献　国際公開第２０２０／１０９９０７（WO，Ａ１）
国際公開第２０１９／１３３５６８（WO，Ａ１）
国際公開第２０２０／０２１３９４（WO，Ａ２）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０９Ｃ　１／００－ ５／００
Ｈ０４Ｌ　９／００－４０
Ｇ０６Ｆ　２１／００－８８