

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号

特許第7273053号

(P7273053)

(45)発行日 令和5年5月12日(2023.5.12)

(24)登録日 令和5年5月1日(2023.5.1)

(51)国際特許分類

H 0 4 L 9/32 (2006.01)

F I

H 0 4 L 9/32 2 0 0 Z

請求項の数 14 (全32頁)

(21)出願番号 特願2020-547298(P2020-547298)
 (86)(22)出願日 平成30年11月28日(2018.11.28)
 (65)公表番号 特表2021-505095(P2021-505095 A)
 (43)公表日 令和3年2月15日(2021.2.15)
 (86)国際出願番号 PCT/EP2018/082832
 (87)国際公開番号 WO2019/106006
 (87)国際公開日 令和1年6月6日(2019.6.6)
 審査請求日 令和3年11月16日(2021.11.16)
 (31)優先権主張番号 17425121.5
 (32)優先日 平成29年12月1日(2017.12.1)
 (33)優先権主張国・地域又は機関
 欧州特許庁(EP)
 (31)優先権主張番号 102017000145294
 (32)優先日 平成29年12月15日(2017.12.15)
 最終頁に続く

(73)特許権者 520189980
 クアント ネットワーク リミテッド
 イギリス グレーター ロンドン ロンドン
 N1 7 G U ウェンロック ロード 2 0
 - 2 2
 (74)代理人 110002860
 弁理士法人秀和特許事務所
 (72)発明者 ベルディアン, ギルバート
 イギリス グレーター ロンドン ロンドン
 N1 7 G U ウェンロック ロード 2 0
 - 2 2 クアント ネットワーク リミテ
 ッド内
 (72)発明者 バターソン, コリン
 イギリス グレーター ロンドン ロンドン
 N1 7 G U ウェンロック ロード 2 0
 最終頁に続く

(54)【発明の名称】 ブロックチェーン通信と順序付け

(57)【特許請求の範囲】

【請求項1】

1つ以上のブロックチェーン上に現れるトランザクションの順序の記録を維持するためのコンピュータ実装方法であって、前記方法が

複数のトランザクションを識別することであって、前記複数のトランザクションの各トランザクションが前記1つ以上のブロックチェーンのいずれかに含まれていることと、

前記複数のトランザクションの記録をデータストアに保存することであって、前記記録が前記複数のトランザクションの相対順序を示すこととを含む、方法。

【請求項2】

前記複数のトランザクションを識別することが、

前記1つ以上のブロックチェーンの各々に追加された新しい各ブロックの内容を読み取ることを含む、請求項1に記載の方法。

【請求項3】

前記複数のトランザクションを識別することが、

前記新しい各ブロック内のトランザクションを関連性基準と比較することであって、前記識別された複数のトランザクションが、前記関連性基準を満たすトランザクションを含むことをさらに含む、請求項2に記載の方法。

【請求項4】

前記複数のトランザクションの記録をデータストアに保存することは、トランザクシ

ンが新しいブロックで識別される度に、その識別されたトランザクションの記録を前記データストアに保存することを含み、前記複数のトランザクションの相対順序が、前記複数のトランザクションの各々が識別された相対順序である、請求項 2 または請求項 3 に記載の方法。

【請求項 5】

前記複数のトランザクションの記録が複数の連続した検証セットを含み、
前記複数の連続した検証セットの少なくともいくつかは、前記複数のトランザクションのそれぞれ 1 つ以上に対応する 1 つ以上のトランザクション識別子を含む、
請求項 1 ～ 4 のいずれか一項に記載の方法。

【請求項 6】

前記 1 つ以上のブロックチェーンの少なくとも 1 つに、前記複数の連続する検証セットの直近の検証セットの記録を保存すること
をさらに含む、請求項 5 に記載の方法。

【請求項 7】

前記複数の検証セット内の先行する検証セットの記録が前記 1 つ以上のブロックチェーンの少なくとも 1 つに存在することを確認することと、

前記先行する検証セットの記録が前記 1 つ以上のブロックチェーンの少なくとも 1 つに存在しない場合には、

前記先行する検証セットの記録を前記 1 つ以上のブロックチェーンの少なくとも 1 つで再度行うことと

をさらに含む、請求項 6 に記載の方法。

【請求項 8】

前記先行する検証セットの記録が前記 1 つ以上のブロックチェーンの少なくとも 1 つに存在しない場合には、

前記先行する検証セットで識別されたトランザクションの各々が、前記 1 つ以上のブロックチェーンのその対応するブロックチェーンに存在することを確認することと、

前記先行する検証セットで識別された 1 つ以上のトランザクションが、前記 1 つ以上のブロックチェーンのそれらの対応するブロックチェーンに存在しない場合には、

前記 1 つ以上のトランザクションをそれらの対応するブロックチェーンに再度行うことと

をさらに含む、請求項 7 に記載の方法。

【請求項 9】

前記直近の検証セットで識別されたトランザクションの各々が前記 1 つ以上のブロックチェーンの少なくとも 1 つに存在することを確認することと、

前記先行する検証セットで識別された 1 つ以上のトランザクションが前記 1 つ以上のブロックチェーンのそれらの対応するブロックチェーンに存在しない場合には、

前記 1 つ以上のトランザクションをそれらの対応するブロックチェーンに再度行うことと
をさらに含む、請求項 6 ～ 8 のいずれか一項に記載の方法。

【請求項 10】

前記直近の検証セットの記録が、前記直近の検証セットの内容に少なくとも部分的に基づいて決定される検証セット識別子を含む、請求項 6 ～ 9 のいずれか一項に記載の方法。

【請求項 11】

前記直近の検証セットの前記検証セット識別子が、前記直近の検証セットの内容のハッシュを含む、請求項 10 に記載の方法。

【請求項 12】

前記複数のトランザクションの記録は、各々がそれぞれのトランザクションの内容に少なくとも部分的に基づいて決定される複数のトランザクション識別子を含む、請求項 1 ～ 11 のいずれか一項に記載の方法。

【請求項 13】

1 つ以上のプロセッサと、

10

20

30

40

50

ソフトウェアプログラムを保存するメモリであって、前記ソフトウェアプログラムが前記1つ以上のプロセッサによって実行されると請求項1～12のいずれか一項に記載の方法を前記1つ以上のプロセッサに実行させる、メモリとを含むシステム。

【請求項14】

1つ以上のプロセッサによって実行されると請求項1～12のいずれか一項に記載の方法を前記1つ以上のプロセッサに実行させる1つ以上の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、1つ以上のブロックチェーン上に現れるトランザクションの順序の記録を維持するための方法、システム、コンピュータプログラムおよびSDKに関する。また、ブロックチェーン通信に関連する方法、システム、コンピュータプログラムおよびSDKにも関する。

【背景技術】

【0002】

ブロックチェーン技術は、広範な用途の可能性を秘め、中央のデータ認証機関を必要とせずに安全かつ確実にデータを保存し得る方法を提供する。その結果、ブロックチェーン技術は、例えば、データの透明性の向上、特定のストレージエンティティに依存しないより堅牢なデータストレージ、データセキュリティの向上、不正への耐性の向上など、幅広いセクターにわたる多くの技術的メリットを可能にし得て、電気エネルギーの分配やピアツーピアのクラウドストレージなど、様々な分野における用途がすでに見出されている。

【0003】

ブロックチェーンは、サトシ・ナカモトの名前で公開された「ビットコイン：ピアツーピア電子キャッシュシステム」というタイトルのホワイトペーパーで2009年に最初に概念化された。ホワイトペーパーのコピーは、<https://bitcoin.org/bitcoin.pdf>で見出され得る。このホワイトペーパーでは、ビットコインのピアツーピア転送を記録するための、以降ブロックチェーンと呼ばれているものを利用した「ビットコイン」と呼ばれるピアツーピアバージョンの電子キャッシュが提案された。

【0004】

ブロックチェーンは非中央集権化された分散型デジタル台帳であり、これを使用して、効果的に不変のデータ記録が保存され得る。それらは、データをブロックチェーンに追加したり、ブロックチェーンの整合性を維持したりするために中央の権限を必ずしも必要としないという点で非中央集権化されている。データは、ブロックチェーンに参加するノードのネットワークに「トランザクション」をブロードキャストするエンティティによってブロックチェーンに追加され得て、トランザクションは、ブロックチェーンに追加されるデータと、ブロックチェーンプロトコルに準拠する暗号化要素とを含む。次に、ブロックチェーンに参加している各ノードは、トランザクションの有効性を検証し、承認された場合、作業中の新しいブロックにトランザクションを追加する。

【0005】

各ノードは、その新しいブロックに、ノードのネットワークにブロードキャストされた他の新しいトランザクションの作業中であることを追加する場合もある。しばらくすると、ノードは、その新しいブロックをネットワーク内の他のノードに発行し、その新しいブロックには、様々な新しいトランザクションと、新しいブロックをブロックチェーン内の前のブロックに結び付ける暗号化データとが含まれている。このように、連続するブロックを暗号で結び付けることによって、ブロックチェーン内の1つのブロックが後で改変される場合（例えば、誰かがブロックチェーンに保存されているデータを不正に変更したい場合）に容易に検出可能であり得る。

【0006】

ノードが新しいブロックを発行した後、ネットワーク内の他のノードは新しいブロック

10

20

30

40

50

の内容を確認し、承認された場合、次の新しいブロックで作業を開始し、それがまたその前のブロックに暗号で結び付けられる。したがって、新しいブロックが発行されて承認される度にブロックチェーンが成長し、新しい承認された各ブロックはその前のブロックに暗号で結び付けられていることがわかる。ノードが、発行された新しいブロックを確認し、それをブロックチェーンに受け入れるこのプロセスは、「コンセンサス」メカニズムと呼ばれることがよくある。

【0007】

ホワイトペーパー「ビットコイン：ピアツーピア電子キャッシュシステム」では、ビットコイン用に特定のプロトコルが提案されている。プロトコルは、トランザクションに含めるべき情報の形式と種類、新しいブロックに含まれる情報の形式と種類、トランザクションおよび新しいブロックに使用すべき暗号化技術とレシピ、ならびに使用されるコンセンサスメカニズムの特定の種類（例えば、ビットコインでは、コンセンサスメカニズムは、ブロックチェーンの各ブロックでハッシュベースの作業証明を利用し、これは、連続する各ブロック間の暗号化タイドでもある）のような事柄をカバーする。ブロードキャストトランザクションは、プロトコルに適切に準拠していない場合、いずれの新しいブロックにも含まれない。同様に、新しいブロックがプロトコルに適切に準拠していない場合、ノードのネットワークによってブロックチェーンに受け入れられない。

【0008】

ブロックチェーンは分散型台帳技術（DLT）とも呼ばれ、最初は暗号通貨ビットコインを有効にするために提案されたが、その後、他の多くの用途が見出されている。最初に、イーサリアム、リップル、ネームコインなどの新しい暗号通貨が発売され、各々独自の特定のブロックチェーンで動作し、独自の特定のプロトコルを利用している。それ以来、ブロックチェーン技術は暗号通貨よりもはるかに広範に活用され得て、かつデータの効果的に不変の記録が役立つあらゆるシナリオで利用され得ることが認識されている。これにより各々特定の使用例に合わせて作られた現存する様々なブロックチェーンの数が急速に拡大することになり、技術がさらに勢いを増し認知度が高まるにつれて様々なブロックチェーンの数が増え続けると予想される。

【発明の概要】

【0009】

本開示の第1の態様では、1つ以上のブロックチェーン（DLT）上に現れるトランザクションの順序の記録を維持するためのコンピュータ実装方法が提供され、この方法は、複数のトランザクションを識別することであって、複数のトランザクションの各トランザクションが1つ以上のブロックチェーンのいずれかに含まれていることと、複数のトランザクションの記録をデータストアに保存することであって、記録が複数のトランザクションの相対順序を示すこととを含む。

【0010】

トランザクションの順序の記録を維持することにより、順序が乱れたメッセージに関連するセキュリティおよび安定性の弱点が低減され得るため、1つ以上のブロックチェーンを利用するシステムのセキュリティおよび安定性が向上する。複数のブロックチェーンにまたがるトランザクションの順序付けは特に問題が多く、セキュリティおよび不正の問題にさらされているため、この技術的な利点は、2つ以上（複数）のブロックチェーンにまたがって現れるトランザクションの順序が記録される場合に特に明白である。

【0011】

複数のトランザクションを識別することは、1つ以上のブロックチェーンの各々に追加された新しい各ブロックの内容を読み取ることを含み得る。

【0012】

複数のトランザクションを識別することは、新しい各ブロック内のトランザクションを関連性基準と比較することであって、識別された複数のトランザクションが、関連性基準を満たすトランザクションを含むことをさらに含み得る。関連性基準と比較することにより、関連するトランザクションのみをデータストアで識別すればよく、これにより、デー

10

20

30

40

50

タストアが将来レビューされる場合の速度が向上し、必要なデータストアのサイズが削減され得る。

【0013】

好ましくは、データストアに複数のトランザクションの記録を保存することは、新しいブロックでトランザクションが識別される度にその識別されたトランザクションの記録をデータストアに保存することであって、複数のトランザクションの相対順序は、複数のトランザクションの各々が識別された相対順序である。

【0014】

好ましくは、複数のトランザクションの記録は、複数の連続した検証セットを含み、複数の連続した検証セットの少なくともいくつかは、複数のトランザクションのそれぞれ1つ以上に対応する1つ以上のトランザクション識別子を含む。

10

【0015】

この方法はさらに、1つ以上のブロックチェーンの少なくとも1つに、複数の連続した検証セットの直近の検証セットの記録を保存することを含み得る。このようにして、記録の整合性および否認防止が強化され、それによって方法のセキュリティと安定性が向上し得る。

【0016】

この方法はさらに、複数の検証セット内の先行する検証セットの記録が1つ以上のブロックチェーンの少なくとも1つに存在することを確認することと、先行する検証セットの記録が1つ以上のブロックチェーンの少なくとも1つに存在しない場合には、先行する検証セットの記録を1つ以上のブロックチェーンの少なくとも1つで再度行うこととを含み得る。このようにして、トランザクションの記録された順序の信頼性を維持しつつ、ブロックチェーンの初期のフォークがすばやく識別され得る。

20

【0017】

先行する検証セットの記録は、直近の検証セットの記録を1つ以上のブロックチェーンの少なくとも1つに保存する前に、1つ以上のブロックチェーンの少なくとも1つで再度行ってもよい。

【0018】

この方法はさらに、先行する検証セットの記録が1つ以上のブロックチェーンの少なくとも1つに存在しない場合には、先行する検証セットで識別されたトランザクションの各々が、1つ以上のブロックチェーンのその対応するブロックチェーンに存在することを確認することと、先行する検証セットで識別された1つ以上のトランザクションが、1つ以上のブロックチェーンのそれらの対応するブロックチェーンに存在しない場合には、1つ以上のトランザクションをそれらの対応するブロックチェーンに再度行うこととを含み得る。

30

【0019】

この方法はさらに、直近の検証セットで識別されたトランザクションの各々が1つ以上のブロックチェーンの少なくとも1つに存在することを確認することと、先行する検証セットで識別された1つ以上のトランザクションが1つ以上のブロックチェーンのそれらの対応するブロックチェーンに存在しない場合には、1つ以上のトランザクションをそれらの対応するブロックチェーンに再度行うこととを含み得る。

40

【0020】

1つ以上のトランザクションは、直近の検証セットの記録を1つ以上のブロックチェーンの少なくとも1つに保存する前に、それらの対応するブロックチェーンに再度行われ得る。

【0021】

直近の検証セットの記録は、直近の検証セットの内容に少なくとも部分的に基づいて決定される検証セット識別子を含み得る。

【0022】

直近の検証セットの検証セット識別子は、直近の検証セットの内容のハッシュを含み得

50

る。

【0023】

直近の検証セットの記録を保存することは、少なくとも1つのブロックチェーンに含めるために1つ以上のブロックチェーンの少なくとも1つのブロックチェーンにトランザクションをブロードキャストすることを含み得て、ブロードキャストされるトランザクションは直近の検証セットの一意の識別子を含む。

【0024】

複数のトランザクションの記録は、各々がそれぞれのトランザクションの内容に少なくとも部分的に基づいて決定される複数のトランザクション識別子を含み得る。

【0025】

複数のトランザクション識別子は、複数のトランザクションの内容の複数のハッシュを含み得る。

【0026】

複数のトランザクションの記録は、複数のトランザクションの少なくともいくつかの内容を含み得る。

【0027】

本開示の第2の態様では、1つ以上のプロセッサと、ソフトウェアプログラムを保存するメモリであって、ソフトウェアプログラムが1つ以上のプロセッサによって実行されると先行する請求項のいずれかに記載の方法を1つ以上のプロセッサに実行させるメモリを含むシステムが提供される。

【0028】

本開示の第3の態様では、1つ以上のプロセッサによって実行されると第1の態様による方法を1つ以上のプロセッサに実行させる1つ以上のコンピュータプログラムが提供される。

【0029】

本開示の第4の態様では、第3の態様の1つ以上のコンピュータプログラムを開発するためのソフトウェア開発ツールのセットを含むソフトウェア開発キットSDKが提供される。

【0030】

本開示の第5の態様では、アプリケーション機能層と複数の異なるブロックチェーンとの間の相互運用性を提供する方法が提供され、ここで、アプリケーション機能層は複数の異なるブロックチェーンの1つ以上を利用する機能的オペレーションを含み、方法は、ブロックチェーン通信層が、アプリケーション機能層からブロックチェーン要求を受信することであって、ブロックチェーン要求が複数の異なるブロックチェーンの1つ以上のブロックチェーン上で実行されるブロックチェーン動作に関することと、ブロックチェーン動作を実行するために1つ以上のブロックチェーンのプロトコルに従って1つ以上のブロックチェーンとインターフェースすることを含む。したがって、アプリケーション機能層はブロックチェーンプロトコルから切り離され得て、アプリケーションがより簡単にブロックチェーンをまたいで動作し、かつブロックチェーン間で切り替わることを可能にし、それにより、特定のブロックチェーンにおける任意のセキュリティや安定性の脅威をより迅速に軽減することを可能にする。

【0031】

ブロックチェーン要求は、1つ以上のブロックチェーン上のデータを読み取る要求を含み得る。1つ以上のブロックチェーンからデータを読み取る要求は、1つ以上のブロックチェーンから読み取られるデータを示すデータ識別子を含み得る。

【0032】

1つ以上のブロックチェーンとインターフェースすることは、1つ以上のブロックチェーンで要求されたデータを読み取ることを含み、方法は、ブロックチェーン通信層が、1つ以上のブロックチェーンから読み取られたデータをアプリケーション機能層に通信することをさらに含む。

10

20

30

40

50

【0033】

ブロックチェーン要求は、1つ以上のブロックチェーンにデータを書き込む要求を含み得る。

【0034】

データの書き込み要求は、1つ以上のブロックチェーンに書き込むデータを含み得る。

【0035】

ブロックチェーン要求は、複数のブロックチェーンのうちの第1のブロックチェーンと複数のブロックチェーンのうちの第2のブロックチェーンとの間で、クロスレジャートランザクションXLTを実行する要求を含み得て、XLTを実行する要求は、XLTによって第2のブロックチェーンに転送される第1のブロックチェーン上のデータを示すXLT識別子を含む。

10

【0036】

ブロックチェーン通信層とアプリケーション機能層との間の通信は、ブロックチェーンプログラミングインターフェースBPIによって定義された方法に従って実行され得る。

【0037】

本開示の第6の態様では、1つ以上のプロセッサと、ソフトウェアプログラムを保存するメモリであって、ソフトウェアプログラムが1つ以上のプロセッサによって実行されると先行する請求項のいずれかに記載の方法を1つ以上のプロセッサに実行させるメモリとを含むシステムが提供される。

【0038】

本開示の第7の態様では、1つ以上のプロセッサによって実行されると第5の態様による方法を1つ以上のプロセッサに実行させる1つ以上のコンピュータプログラムが提供される。

20

【0039】

本開示の第8の態様では、第7の態様の1つ以上のコンピュータプログラムを開発するためのソフトウェア開発ツールのセットを含むソフトウェア開発キットSDKが提供される。

【0040】

本開示の第9の態様では、第1のブロックチェーンと第2のブロックチェーンとの間でクロスレジャートランザクションXLTを実行するための方法が提供され、この方法は、第1のブロックチェーン上でXLTに関連するプロポーズ（*propose*）XLTトランザクションを識別することと、データストアのトランザクション記録にプロポーズXLTトランザクションの記録を保存することと、第2のブロックチェーン上でレディ（*ready*）XLTトランザクションを識別することと、レディXLTトランザクションの記録をデータストア内のトランザクション記録に保存することと、この記録がプロポーズXLTトランザクションとレディXLTトランザクションの相対順序を示すことと、プロポーズXLTトランザクションとレディXLTトランザクションがXLT基準を満たすかどうかを判定することと、プロポーズXLTトランザクションとレディXLTトランザクションがXLT基準を満たしている場合には、第1のブロックチェーンと第2のブロックチェーンとの間でXLTを実行するために第1のブロックチェーンおよび第2のブロックチェーンにコミット（*commit*）XLTトランザクションをブロードキャストすることと、XLT基準が、レディXLTトランザクションがプロポーズXLTトランザクションに対応するという第1の基準と、プロポーズXLTトランザクションがレディXLTトランザクションに先行することをトランザクション記録が示すという第2の基準とを含むこととを含む。したがって、1つ以上のブロックチェーンにフォークが発生した場合でも、2つ以上のブロックチェーンにまたがるトランザクションの順序が確実に維持および理解され得るため、クロスレジャートランザクションのセキュリティと信頼性が向上し、不正のリスクが低減される。

30

40

【0041】

この方法はさらに、1つ以上のトランザクションを第1のブロックチェーンおよび／ま

50

たは第2のブロックチェーンにブロードキャストすることであって、1つ以上のトランザクションがそれぞれ1つ以上の記録識別子を含み得て、各記録識別子はトランザクション記録の内容の少なくとも一部に少なくとも部分的に基づいて決定され、1つ以上の記録識別子は共にプロポーズXLTトランザクションおよびレディXLTトランザクションの記録を示す。

【0042】

1つ以上の記録識別子は、トランザクション記録の内容の少なくとも一部の1つ以上のハッシュを含み得る。

【0043】

XLT基準は、1つ以上のトランザクションが第1のブロックチェーンおよび／または第2のブロックチェーンに含まれ、その結果、1つ以上の記録識別子が第1のブロックチェーンおよび／または第2のブロックチェーンに保存されるという第3の基準を含み得る。

【0044】

プロポーズXLTトランザクションの記録はプロポーズXLTトランザクションの内容のハッシュを含み得て、レディXLTトランザクションの記録はレディXLTトランザクションの内容のハッシュを含む。

【0045】

本開示の第10の態様では、1つ以上のプロセッサと、ソフトウェアプログラムを保存するメモリであって、ソフトウェアプログラムが1つ以上のプロセッサによって実行されると第9の態様による方法を1つ以上のプロセッサに実行させるメモリとを含むシステムが提供される。

【0046】

本開示の第11の態様では、1つ以上のプロセッサによって実行されると第9の態様による方法を1つ以上のプロセッサに実行させる1つ以上のコンピュータプログラムが提供される。

【0047】

本開示の第12の態様では、第11の態様の1つ以上のコンピュータプログラムを開発するためのソフトウェア開発ツールのセットを含むソフトウェア開発キットSDKが提供される。

【0048】

本開示の第13の態様では、2つ以上のブロックチェーン上で実行されたトランザクションの相対順序の記録を維持するための方法が提供される。このようにして、異なるブロックチェーンのトランザクションの順序を確実に順序付けることができるため、特にブロックチェーンフォークおよび／またはXLTの場合に、セキュリティと信頼性の向上を可能にする。

【0049】

本開示の第14の態様では、1つ以上のプロセッサと、ソフトウェアプログラムを保存するメモリであって、ソフトウェアプログラムが1つ以上のプロセッサによって実行されると第9の態様による方法を1つ以上のプロセッサに実行させるメモリとを含むシステムが提供される。

【0050】

本開示の第15の態様では、1つ以上のプロセッサによって実行されると第9の態様による方法を1つ以上のプロセッサに実行させる1つ以上のコンピュータプログラムが提供される。

【0051】

本開示の第16の態様では、第11の態様の1つ以上のコンピュータプログラムを開発するためのソフトウェア開発ツールのセットを含むソフトウェア開発キットSDKが提供される。

【図面の簡単な説明】

【0052】

10

20

30

40

50

本開示の態様は、以下の図面を参照して、例としてのみ説明される。

【0053】

【図1】図1は、本開示の一態様によるオーバーレジャー通信層の例示的な表現を示す図である。

【0054】

【図2】図2は、図1のオーバーレジャー通信層を利用するシステムの例示的な表現を示す図である。

【0055】

【図3】図3は、図1のオーバーレジャー通信層を利用するさらなるシステムの例示的な表現を示す図である。

【0056】

【図4】図4は、図3のシステムを使用して実行される例示的な方法のステップの表現を示す図である。

【0057】

【図5】図5は、図1のオーバーレジャー通信層を利用するさらなるシステムの例示的な表現を示す図である。

【0058】

【図6】図6は、ブロックチェーン内のフォークの例示的な表現を示す図である。

【0059】

【図7】図7は、本開示の一態様による、1つ以上のブロックチェーン上に現れるトランザクションの順序の記録を維持するための方法の例示的な表現を示す図である。

【0060】

【図8】図8は、図7の方法が実行される2つのブロックチェーンの例示的な表現を示す図である。

【0061】

【図9】図9は、図7の方法が実行される2つのブロックチェーンのさらなる例示的な表現を示す図である。

【0062】

【図10】図10は、本開示の態様に従ってクロスレジャートランザクションが実行される2つのブロックチェーンの例示的な表現を示す図である。

【0063】

【図11】図11は、本開示の一態様によるさらなるシステムの例示的な表現を示す図である。

【0064】

【図12】図12は、図11のシステムによって実行される方法の例示的な表現を示す図である。

【発明を実施するための形態】

【0065】

本発明者らは、ブロックチェーン技術が成長および発展するにつれて、様々なエンティティが安全、安心、かつ確実にそれを利用することがますます困難になる可能性があることを認識した。例えば、現在、エンティティがブロックチェーン技術を使用する新しいソフトウェアアプリケーションを開発したい場合、独自の新しいブロックチェーンを開始するか（非常に時間がかかり、複雑であり、まったく新しいブロックチェーンプロトコルの確立が必要である）、またはビットコインやイーサリアムなどの既存のブロックチェーン技術を利用しなければならない。

【0066】

本発明者らは、既存のブロックチェーン技術を利用するために、エンティティが、そのブロックチェーンのプロトコルに完全に準拠する、例えばプロトコルによって定められる暗号化技術、トランザクション形式および通信チャネルを使用して、ソフトウェアアプリケーションを開発しなければならないことに気付いた。したがって、使用するブロックチ

10

20

30

40

50

チェーン技術の選択は、ソフトウェア開発プロセスの非常に早い段階で、場合によってはソフトウェアアプリケーションの要件と使用法を完全に理解する前に行う必要がある決定である。

【0067】

各ブロックチェーンには、ブロックチェーンの特定のニーズを満たすために策定された特定のプロトコルがある。ブロックチェーンでハードフォークを発生させずにブロックチェーンプロトコルを変更することはできず、これはブロックチェーンが必要な変更に対してあまり適応性がないことを意味する。その結果、選択したブロックチェーン技術がニーズを満たさなくなったとエンティティが後で判断した場合（例えば、選択したブロックチェーン技術が不安定になったり、安全でなくなったりしたため、または異なるブロックチェーン技術が有利であるようにソフトウェアアプリケーションの要件が変わったため、または最初に選択されたが今では古くなったブロックチェーン技術よりも新しいブロックチェーンが優れているため）、代替となるブロックチェーン技術の使用を開始するには、代替となるブロックチェーン技術のプロトコルに準拠するようにソフトウェアを修正しなければならない。ソフトウェアは最初に選択されたブロックチェーン技術のプロトコルによって設定された基盤上に構築されるため、実際には、これは不可能ではないにしても非常に困難であり得る。したがって、最初から正しいブロックチェーン技術を選択することが非常に重要であるが、ブロックチェーン技術は急速に発展しているため、実際には達成するのが非常に難しい可能性がある。

【0068】

さらに、以前にブロックチェーンに保存された任意のデータは、ブロックチェーンプロトコル間の違いにより、代替となるブロックチェーンに直接転送されない場合がある。

【0069】

本発明者らによって認識されたこれらの長期的な困難に鑑みて、アプリケーションの機能をそれらが利用しているブロックチェーンから分離するために新しい通信層を導入する技術的解決策が本発明者らによって考案され、本開示において提供されている。アプリケーションが動作しているブロックチェーンの様々な要件からアプリケーションの機能を切り離すことにより、アプリケーションとブロックチェーンとの間の相互運用性は改善され得る。以下でより詳細に説明する本開示の態様から理解されるように、これは、将来のブロックチェーン技術の変化に、より効率的かつ柔軟に対応し得ることを意味する可能性がある。これは、セキュリティの脅威への対応（例えば、1つのブロックチェーン技術にセキュリティの欠陥があると認識された場合、アプリケーションがより安全なブロックチェーン技術に簡単に切り替わり得る）、セキュリティおよび信頼性の向上（アプリケーションが複数のブロックチェーン間でより容易に動作し、ブロックチェーン間で重複データを保存し、および／またはブロックチェーン間でデータを転送する可能性があるため）、および／または効率および速度の向上（例えば、より新しいブロックチェーン技術でトランザクションの処理速度が上昇した場合、アプリケーションがそのより新しいブロックチェーン技術に簡単に切り替わり得る）に役立つ可能性がある。

【0070】

以下の開示全体を通じて、「ブロックチェーン」、「ブロックチェーン技術」、「分散型台帳技術（DLT）」という用語は互換的に使用される。さらに、以下の開示全体を通じて、「トランザクション」という用語はブロックチェーンに関連して使用される。当業者は、この文脈における「トランザクション」が、例えば通貨など、何らかの所有権の移転に限定されないことを理解するであろう。むしろ、「トランザクション」は、何らかの形の情報をブロックチェーン上に置くことができる手段である。例えば、エンティティは、ブロックチェーン上に置かれる「トランザクション」をブロードキャストする場合があります、トランザクションには、ブロックチェーン上に置かれる情報と、トランザクションがブロックチェーンに承認されるために必要とされるその他のデータ項目（公開鍵、署名、ハッシュなど）が含まれる。ブロックチェーン上に置かれる情報は、通貨の金融送金に関連する場合があるが、代わりに、任意の他の種類の情報（例えば、エンティティが後で

10

20

30

40

50

自分自身が参照するためにブロックチェーンに保存したいデータなど）の場合もある。

【0071】

図1は、アプリケーション機能層30とDLT層10との間に位置する「オーバーレジャー通信層」20の導入の例示的な視覚化を示す。「オーバーレジャー通信層」という用語は、本開示全体を通じて利用され、DLTのプロトコル要件を、DLTを利用するアプリケーションの機能的オペレーション（「ビジネスロジック」と呼ばれることもある）から切り離す通信層を意味する。

【0072】

図1では、DLT層10の一部として2つのDLT（第1のDLT12および第2のDLT14）が示されているが、オーバーレジャー通信層20は、任意の数の異なるDLT、例えばビットコイン、イーサリアム、コーダ、オープンチェーン、ハイパーレジャーなどとインターフェース可能であり得ることが理解されよう。各DLTには、ブロックチェーンからのデータの読み取り、ブロックチェーンへのデータの書き込み、および／または同じブロックチェーン上のあるエンティティから別のエンティティへ、または別のブロックチェーンへ（クロスレジャートランザクション）のデータの所有権の移転など、ブロックチェーン動作の独自の特定のプロトコルがある（様々なブロックチェーンでは、ブロックチェーンが実行するように構成されている目的に応じて、ブロックチェーン上で読み取り、書き込みおよび移転動作の任意の1つ以上が可能であり得る）。

【0073】

オーバーレジャー通信層20は、ブロックチェーン動作が第1のDLT12および第2のDLT14上で実行され得る異なる方法を含む。例えば、第1のDLT12はビットコインであり得て、新しいトランザクションをそのブロックチェーンに書き込むための特定のプロトコルを有する。ビットコインの特定のプロトコルは、当業者にはよく理解されているが、簡単に言えば、受取人の公開鍵（受取人の「財布」）、受取人に転送される金額、現在転送中のビットコインを支払者が取得した以前のトランザクションのハッシュ、および（支払者の秘密鍵を使用して署名された）支払者の署名などの情報を含み得る。プロトコルは、これらの各項目の許容される形式（長さやオブジェクトタイプなど）、ハッシュおよび署名を生成するために使用しなければならない暗号技術（SHA-256など）、ならびに各項目がトランザクション内に表示される順序を定義する。また、プロトコルは、ビットコインのブロックチェーンから情報がどのように読み取られ得るか（例えば、各トランザクションおよびブロックチェーンの各ブロックにどの情報を保持するかを識別する）、情報の形式と順序、ならびに（ブロックチェーンの読み取り時に各々が検証され得るように）ハッシュ、署名および作業証明の生成に使用される暗号技術を定義する場合がある。対照的に、第2のDLT14は、読み取り動作のみなど、異なるブロックチェーン動作の実行を許可する完全に異なるプロトコルを備えている場合があり（例えば、それは誰でも読み取ることができる閉じたブロックチェーンであるが、許可されたエンティティのみが書き込みできる）、各ブロックは、異なる情報を異なる順序で含み得る、および／または第1のDLT12に対して異なる暗号技術を使用し得る。オーバーレジャー通信層20は、第1のDLT12および第2のDLT14の各々で実行される様々な許容可能な動作を可能にする方法を含む。

【0074】

図1に示すアプリケーション機能層30は、1つ以上のDLTを利用しているアプリケーションの機能的オペレーション（「ビジネスロジック」と呼ばれることもある）を含む。アプリケーションのアプリケーション機能層30は、ブロックチェーン動作を実行したい場合、関連情報（ブロックチェーン要求）をオーバーレジャー通信層20に渡し（例えば、DLTに書き込みたいデータ、またはDLTから読み取りたい情報の性質を含む）、次に、ブロックチェーン動作を実行するために関連するDLTのプロトコルに従って関連するDLTとインターフェースする。次に、オーバーレジャー通信層20は、任意の関連する結果をアプリケーション機能層30に返してもよい。この動作は、本開示で後に説明されるより具体的な例からより完全に理解され得る。

10

20

30

40

50

【0075】

オーバーレジャー通信層20に保持されている方法を様々な異なるエンティティが利用できるように、方法は、例えば、ソフトウェア開発キット（SDK）にカプセル化されてもよく、これを以降オーバーレジャーSDKと呼ぶ。オーバーレジャー通信層20に保持されている方法は、任意の他の適切な同等の方法でカプセル化されてもよいことが理解されよう。オーバーレジャーSDKを取得することにより、開発者は、開発している特定のソフトウェアアプリケーション用に利用したいDLTをより簡単に選択してもよく、選択したDLTとの相互作用に必要な方法およびツールをオーバーレジャーSDKに提供させてもよい。

【0076】

図2は、SDKプロバイダ210と、アプリケーション開発者220と、開発されたアプリケーション230と、第1のDLTネットワーク240と、第2のDLTネットワーク250とを含むシステム200の例示的な表現を示す。SDKプロバイダ210は、オーバーレジャーSDKを構築および維持する責任があり得て、オーバーレジャーSDKは、この例では少なくとも第1のDLT12および第2のDLT14とインターフェース（相互作用）する方法を備えているが、任意の数のDLTのため（すなわち、複数のDLTのため）の相互作用の方法を備えていてもよい。アプリケーション開発者220は、第1のDLT12および／または第2のDLT14を利用するアプリケーションを開発したい場合、SDKプロバイダ210から（例えば、SDKプロバイダ210から、または任意の他のオーバーレジャーSDKホスティングエンティティからダウンロードすることによって）オーバーレジャーSDKを取得し得る。次に、オーバーレジャーSDKを利用してアプリケーション230を開発してもよく、次に、アプリケーション230は、オーバーレジャーSDKで定義された方法に従って、第1のDLT12および／または第2のDLT14と相互作用することができる。図2では、第1のDLTネットワーク240は第1のDLT12の参加ノードのネットワークを表し、第2のDLTネットワーク250は第2のDLT14の参加ノードのネットワークを表す。図2に示す特定の例では、アプリケーション230は、第1のDLTネットワーク240および第2のDLTネットワーク250の両方と相互作用できることが示されているが、アプリケーション開発者は、代わりにオーバーレジャーSDKを利用して、第1のDLTネットワーク240のみ、または第2のDLTネットワーク250のみと相互作用するようにアプリケーション230を構成してもよいことが理解されよう。

【0077】

特定の一態様では、アプリケーション230の開発の一部として、アプリケーション開発者230は、アプリケーション230のオーバーレジャー通信層20とアプリケーション機能層30との間の通信を処理するプログラミングインターフェースを開発してもよい。これ以降、このインターフェースをブロックチェーン・プログラミング・インターフェース（BPI）と呼び、BPIは、アプリケーション機能層30とオーバーレジャー通信層20との相互作用を簡素化するのに役立つように構成されている。例えば、アプリケーション230のオーバーレジャー通信層20は、第1のDLTネットワーク240および／または第2のDLTネットワーク250間の通信を処理するためにオーバーレジャーSDKで定義された方法に従って動作してもよい。アプリケーション開発者220は、BPIを開発して、1つ以上のブロックで実行されるブロックチェーン動作に関連するブロックチェーン要求の中でアプリケーション230のアプリケーション機能層30がオーバーレジャー通信層20に渡し得る特定の種類の情報、および／またはオーバーレジャー通信層20から受信し得る特定の種類の情報を定義してもよい。例えば、「第1のDLT12への書き込み」コマンドを提供してもよく、これは、第1のDLT12に書き込むデータの任意の関連パラメータ（最大長、オブジェクトタイプなど）を指定してもよく、および／または「第2のDLT14からの読み取り」コマンドを提供してもよく、これは、第2のDLT14から情報を読み取るために必要な任意の関連データを指定してもよい（例えば、特定のエンティティによって第2のDLT14に書き込まれた第2のDLT14のデ

10

20

30

40

50

ータを読み取りたい場合、「第2のDLT14からの読み取り」コマンドは、公開鍵など、特定のエンティティのブロックチェーン識別子が必要であることを指定してもよく、これを使用して、オーバーレジャー通信層20がそのブロックチェーン識別子を含むトランザクションについて第2のDLT14の内容を検索してもよい)。その結果、オーバーレジャー通信層20が、オーバーレジャーSDKによってサポートされているDLTのいずれかとインターフェースできるように、アプリケーション230のアプリケーション機能層30は、BPIによって提供される様々なコマンドを利用し得ることがわかる。

【0078】

したがって、アプリケーション230のアプリケーション機能層30は、オーバーレジャーSDKに従って動作するオーバーレジャー通信層20によって、第1のDLT12および第2のDLT14の特定のDLT層10プロトコル要件から切り離される。アプリケーション機能層30は、第1のDLT12および／または第2のDLT14と相互作用するために、BPIによって定義された方法でオーバーレジャー通信層20にデータを送受信するだけでよい。その結果、第1のDLT12または第2のDLT14のいずれかで将来的にそのプロトコルが変更される場合、これはオーバーレジャーSDKへの変更によりオーバーレジャー通信層20によって処理され得る。アプリケーション230のアプリケーション機能層30は、BPIで定義されているようにオーバーレジャー通信層20とデータを送受信するため、不変のままであり得るか、またはDLTプロトコル変更によりBPIを何らかの方法で変更しなければならない場合にせいで最小限変更される(例えば、DLTに書き込むことができるオブジェクトタイプが変更される場合、BPIも変更される可能性があり、それにより、異なるオブジェクトタイプのデータを提供することがアプリケーション機能層30に要求される可能性がある)。それにもかかわらず、アプリケーション機能層30に変更が必要な場合であっても、それは最小限の変更であり、アプリケーション機能層30の基本的な態様は不変のままであり得ることは明らかである。

【0079】

同様に、アプリケーション開発者220が、別のDLTの使用を開始するように決定を下した場合(例えば、最初に選択されたDLTが今では不安定であるか、セキュリティの欠陥があるため、あるいは速度および／または効率が向上した新しいDLTが発売されたため)、同様のことが見られる。その異なるDLTがオーバーレジャーSDKによってサポートされる場合、アプリケーション開発者220は、BPIを単に更新して、その異なるDLTと相互作用するためのコマンドを提供してもよく、その後アプリケーション230のアプリケーション機能層30はその異なるDLTを利用することができる。この場合も、アプリケーション機能層30は、1つの特定のDLTおよびそのDLTが必要とする特定のプロトコルと相互作用するように開発されておらず、代わりに、アプリケーションの機能要件を実行し、何らかの形のブロックチェーンの相互作用が必要なときはいつでもBPIを利用するように開発されているため、アプリケーション230のアプリケーション機能層30に必要な変更は最小限のみであるか皆無であり得ることがわかる。

【0080】

オーバーレジャー通信層20を使用してアプリケーション機能層30をDLT層10から分離することの利点は、アプリケーション開発者220が他の「クライアント」アプリケーション開発者が利用するサービスオフリングを開発するシナリオを考慮することによってさらに理解され得る。サービスオフリングは、例えば、特定の種類の情報をブロックチェーンに書き込むことを可能にするサービス(これについては「メッセージングの態様」のセクションで詳しく説明する)および／またはクロスレジャートランザクションを可能にするサービス(これについては「クロスレジャートランザクションの態様」のセクションで詳しく説明する)および／またはDLTから特定の種類の情報を読み取ることを可能にするサービスなど、他のアプリケーション開発者が利用したい可能性がある、あらゆる種類のブロックチェーンサービスに関連し得る。

【0081】

図3は、「クライアント」アプリケーション開発者によって利用され得るサービスオフ

10

20

30

40

50

アリングを実証するための例示的なシステム 300 を示す。システム 300 は、SDK プロバイダ 210（前述）、サービスオフアリング開発者 310、クライアントアプリケーション開発者 320、クライアントアプリケーション 330、第 1 の DLT ネットワーク 240（前述）、および第 2 の DLT ネットワーク 250（前述）を含む。

【0082】

図 4 は、サービスオフアリング開発者がオーバーレジャー SDK をどのように利用し得るかを例示する方法ステップの例を示す。

【0083】

ステップ S410 において、サービスオフアリング開発者 310 は、SDK プロバイダ 210 からオーバーレジャー SDK を取得し得る（例えば、SDK プロバイダ 210 からダウンロードするか、または任意の他のオーバーレジャー SDK ホスティングエンティティからダウンロードすることによる）。

【0084】

ステップ S420 において、サービスオフアリング開発者 310 は、提供しているサービスの BPI を開発し、それを適切にカプセル化し得る（例えば BPI 構成ファイルにカプセル化されるが、代わりに任意の他の適切な代替が使用されてもよい）。BPI は、先に説明したように、1 つ以上の DLT と相互作用するためにクライアントアプリケーション 330 のアプリケーション機能層 30 によって使用され得る機能、コマンドなどを指定する。

【0085】

ステップ S430 において、クライアントアプリケーション開発者 320 は、オーバーレジャー SDK および BPI 構成ファイルを取得し得る（例えば、サービスオフアリング開発者 310 からダウンロードするか、または任意の他の適切なホスティングエンティティからダウンロードすることによる）。

【0086】

ステップ S440 において、クライアントアプリケーション開発者 320 は、クライアントアプリケーション 330 のアプリケーション機能層 30 を開発する。アプリケーション機能層 30 が何らかの方法で DLT と相互作用する必要がある場合、アプリケーション機能層 30 は単にブロックチェーン要求をオーバーレジャー通信層 20 に通信するために BPI のコマンドまたは機能を適切に利用すればよく、ブロックチェーン要求は、1 つ以上のブロックチェーンで実行されるブロックチェーン動作に関する。したがって、サービスオフアリング開発者 310 によって提供されるオーバーレジャー SDK および BPI は、アプリケーション機能層 30 を DLT 層 10 から切り離すことがわかる。したがって、オーバーレジャー SDK はオーバーレジャー通信層 20 の機能を実行すると見なしてもよく、BPI は、サービスオフアリング開発者によって提供される特定のサービスを実行するためにアプリケーション機能層 30 がオーバーレジャー通信層 20 とどのようにインターフェースし得るかを定義すると見なしてもよい。

【0087】

結果として、クライアントアプリケーション開発者 320 が後日、クライアントアプリケーション 320 に使用する DLT を変更したいと決定した場合（例えば、最初にクライアントアプリケーション 330 の機能アプリケーション層 30 は、BPI の機能およびコマンドを使用して第 1 の DLT 22 と相互作用するように構成されている場合があるが、第 2 の DLT 24 を代わりに使用すべきであると後に決定される）、機能アプリケーション層 20 が使用する特定の BPI の機能およびコマンドを変更する必要があるのみである（オーバーレジャー SDK および BPI が、第 1 の DLT 22 および第 2 の DLT 24 の両方をサポートすると想定）。同様に、オーバーレジャー SDK によってサポートされている任意の DLT のプロトコルについて何かに変更された場合、オーバーレジャー SDK が変更される可能性があり、サービスオフアリング開発者 310 は結果として必要に応じて BPI を変更し得る。その結果、多くの場合、クライアントアプリケーション開発者は、クライアントアプリケーション 330 の機能アプリケーション層 30 が使用する特定の

10

20

30

40

50

B P I の機能およびコマンドを変更するだけでよく、クライアントアプリケーションの機能アプリケーション層 3 0 の基本的な態様はいずれも変更する必要がない。

【 0 0 8 8 】

オプションで、クライアントアプリケーション開発者 3 2 0 が、取得した B P I 構成ファイルが有効であることを確認し得るように、サービスオフリング開発者 3 1 0 は、B P I 構成ファイルを作成した後に標準の公開秘密鍵暗号化技術を使用してそれに署名してもよい。

【 0 0 8 9 】

オーバーレジャー S D K は、特定の種類の D L T 相互作用の実行を可能にする多くの異なる機能を含み得る。これらの特定の機能の 2 つである「メッセージングの態様」と「クロスレジャートランザクション」の態様について、以下で詳しく説明する。

【 0 0 9 0 】

アプリケーション機能層 3 0 およびオーバーレジャー通信層 2 0 は両方とも「層」として上記で説明されているが、それらは代替的にソフトウェアモジュールまたはパッケージと見なしてもよいことが理解されよう。例えば、それらは、アプリケーション 2 3 0 またはクライアントアプリケーション 3 3 0 内の 2 つのモジュールであり得る。あるいは、それらは各々異なるアプリケーション内に配置され、異なる電子デバイスで動作する可能性がある。例えば、アプリケーション機能層 3 0 は、クライアント電子デバイス上で動作するソフトウェアモジュールまたはパッケージであり得て、オーバーレジャー通信層 2 0 は、サービスオフリング電子デバイス上で動作する異なるソフトウェアモジュールまたはパッケージであり得て、この 2 つの層は、任意の適切な通信インターフェースを介して B P I に従って通信する。さらに、オーバーレジャー通信層 2 0 は、特定のブロックチェーン動作を任意の 1 つ以上の異なる D L T で実行できるように構成されたソフトウェアモジュールまたはパッケージと見なしてもよく、サービスオフリング開発者 3 1 0 は、クライアントアプリケーション開発者がオーバーレジャー通信層 2 0 をダウンロードし、B P I を介してそれを利用するか、B P I を介してリモートで利用することを可能にする。

【 0 0 9 1 】

メッセージングの態様

ブロックチェーンが保存するように必ずしも設計されていない情報の不変の記録をブロックチェーン上に維持することがしばしば有用であり得ることが発明者らによって認識された。以下の全体を通して、「メッセージ」という用語が使用されているが、「メッセージ」には、ブロックチェーンを使用して記録を保持したいあらゆる種類の情報が包含されることを理解すべきである。

【 0 0 9 2 】

ブロックチェーンを使用してメッセージの記録を保持することにより、メッセージの内容を将来検証できるように、メッセージの事実上不変の記録を保持することができる。ブロックチェーンは、各ブロックに含まれるトランザクション内のいくつかの種類の情報を承認するように装備されているが、通常、その情報が何であるかについて非常に規範的である。例えば、人気のある多くのブロックチェーン（ビットコインなど）は元来、暗号通貨と単純な金融送金のために設計された。他のブロックチェーンは他の特定の目的のために開発されており、将来さらに多くのブロックチェーンがさらなる特定の目的のために開発される可能性がある。オーバーレジャー通信層 2 0 を開発する過程で、本発明者らは、その目的のために設計されなかったブロックチェーン上のいくつかの種類の情報を保存することが非常に有用であり得ることに気付いた。具体的な使用例の 1 つは、後で説明するように、1 つ以上のブロックチェーン上に現れるトランザクションの順序の記録を維持する一環としてであるが、他にも多くの潜在的な用途と利点がある。

【 0 0 9 3 】

その目的のために設計されなかったブロックチェーンにメッセージを保存することの潜在的な利点に気付いたため、本発明者らは既存のブロックチェーンアーキテクチャ内のトランザクションの詳細を調査した。多くのブロックチェーンアーキテクチャにはトランザ

10

20

30

40

50

クション内のフィールドが含まれており、その中に必須ではないデータを追加できることに気付いた。例えば、ビットコインにはO P - R E T U R Nフィールドがあり、イーサリアムにはスクリプトバイトコードに使用されるフィールドがあり、リップルには「メモ」フィールドがある。これらのフィールドは、一般に「非必須データフィールド」と呼ばれることがある。ただし、これらの各フィールドのサイズ、したがって有用性は、比較的限定されている。

【0094】

本発明者らによって特定された解決策は、そのようなフィールドを使用して、あるメッセージ（ハッシュまたはメッセージの暗号化など）に少なくとも部分的に基づき、かつそれを一意に示す値を保存し、次にそのメッセージを別の場所（オフチェーン）に保存することである。メッセージを一意に示す値は、オフチェーンに保存されている完全なメッセージへのポインタ（例えば、ハッシュポインタ）として機能し得る。メッセージを一意に示す値をブロックチェーンに保存することにより、オフチェーンに保存されたメッセージに基づいてテスト値を生成し、それをブロックチェーンに保存された値に対して確認するだけで、メッセージの整合性はいつでも確認され得る。テスト値はブロックチェーンに保存された値と同じ方法で生成されているため（例えば、同じハッシュレシピを使用して）、テスト値がブロックチェーンに保存された値と一致する場合、オフチェーンに保存されたメッセージには整合性がある（すなわち、改変されていない）。それらが一致しない場合には、オフチェーンに保存されたメッセージは何らかの方法で改ざんされている。これは、ブロックチェーンの内容が事実上不変であるため、ブロックチェーンに保存されている値はどのようにしても改変することができないためである。

【0095】

当業者は、メッセージを一意に示す値が、任意の適切な方法で生成され得ることを理解するであろう。例えば、この値は、S H A - 2（例えば、S H A - 2 5 6、S H A - 5 1 2など）またはS H A - 3などのハッシュアルゴリズムを使用してメッセージの内容をハッシュすることにより生成され得て、メッセージを一意に示す値はハッシュであり、ブロックチェーン内でハッシュポインタとして機能する。あるいは、値は、任意の適切な暗号化アルゴリズムを使用してメッセージの内容を暗号化することによって生成され得る。メッセージを一意に示す値が生成され得る特定の方法は、メッセージが保存される特定のブロックチェーンの要件、および／または満たす必要がある特定の暗号要件（例えば、アルゴリズムによって提供される整合性のレベルなど）に基づいて選択され得る。例えば、値が保存されるトランザクションフィールドは、特定の最大長を有する場合があります、特定のハッシュアルゴリズムは、生成されるハッシュが長すぎる可能性があるために除外される。オーバーレジャーS D Kは、それがサポートするD L Tに適した特定のハッシュアルゴリズムおよび／または暗号化アルゴリズムを含み得て、これにより、アプリケーション機能層はオーバーレジャー通信層20にメッセージを渡すだけでよく、次に、オーバーレジャー通信層20は、D L Tにメッセージの記録を保存するためにメッセージを適切に処理してもよい。次に、アプリケーションはメッセージのコピーをオフチェーンの別の場所（例えば、アプリケーションが動作しているデバイスのローカルメモリ、またはリモートデータベースやクラウドストレージなどのリモートメモリ）に保存してもよい。

【0096】

前述のように、上記のメッセージングの態様には多くの潜在的な用途があり得る。特定の非限定的な使用例の1つを、図5を参照して以下で説明する。

【0097】

図5は、サービスオフアリング開発者310と、クライアントアプリケーション開発者320と、クライアントアプリケーションバックエンド510と、第1のクライアントアプリケーション520と、第2のクライアントアプリケーション530と、第1のD L Tネットワーク240とを含むシステム500を示す。サービスオフアリング開発者310は、上述のメッセージングの態様を使用して1つ以上のD L Tにメッセージを記録するためのB P I機能を提供するメッセージングサービス用のB P Iを作成することができる。

次に、クライアントアプリケーション開発者320は、オーバーレジャーSDKおよびサービスオフリング開発者310のBPI構成ファイルを使用して、メッセージングアプリケーションのユーザが互いに安全にメッセージを送信できるようにするメッセージングアプリケーションを開発することができる。第1のクライアントアプリケーション520および第2のクライアントアプリケーション530は、第1および第2のユーザの電子デバイス、例えばモバイル電子デバイス（スマートフォン、またはタブレットコンピュータ、またはラップトップコンピュータなど）またはデスクトップコンピュータ上でそれぞれ動作するクライアントアプリケーションを複製することができる。第1のクライアントアプリケーション520および第2のクライアントアプリケーション530は、第1および第2のユーザの電子デバイス上で、デバイス上のソフトウェアインストールとして、またはウェブブラウザベースのアプリケーションなどとして動作し得る。

10

【0098】

第1のユーザが第2のユーザにメッセージを伝えたい場合、メッセージと第2のユーザのID（IDは、例えばクライアントアプリケーションの第2のユーザのユーザ名など、任意の適切な形式を取り得る）を第1のクライアントアプリケーション520に入力してもよい。クライアントアプリケーションの機能アプリケーション層30は、サービスオフリング開発者310が作成したBPIの適切な機能／コマンドを使用して、第1のDLT12に追加するために、第1のクライアントアプリケーション520のオーバーレジャー通信層20にメッセージを渡すように構成され得る。次に、オーバーレジャー通信層20は、第1のDLT12と相互作用するためのSDKの方法に従ってメッセージをハッシュし、メッセージのハッシュを含む第1のDLT12のための適切なトランザクションを構築し、トランザクションを第1のDLTネットワーク240に通信してもよい。したがって、メッセージのハッシュは第1のDLT12に含まれる。また、第1のクライアントアプリケーション520のオーバーレジャー通信層20は、オプションで、トランザクションが第1のDLT12に首尾よく追加されたと判定したときに、アプリケーション機能層30に通知を返すように構成されてもよい。メッセージ全体のコピーとメッセージのハッシュはまた、例えば、アプリケーション機能層30がオーバーレジャー通信層20からメッセージのハッシュを受信し、それをメッセージと共にクライアントアプリケーションバックエンド510に通信することにより、クライアントアプリケーションバックエンド510（これは例えば、メッセージを受信、保存および検索するためにクライアントアプリケーション開発者320が提供するサーバ、データベース、またはクラウドベースのストレージシステムなどであり得る）に通信されてもよい。第1のクライアントアプリケーション520とクライアントアプリケーションバックエンド510との間の通信は、オプションで、任意の適切な暗号化技術を使用して保護されてもよい。

20

30

【0099】

次に、第2のユーザは、メッセージが待機していることに気が付き得る。これが発生し得る多くの異なる方法が存在し、例えば、第2のユーザ識別子は、第2のユーザのDLT公開鍵またはDLTウォレットであり得るか、または第2のユーザのDLT公開鍵またはDLTウォレットを取得するために第1のクライアントアプリケーション520によって使用されていてもよく、DLT12上のトランザクションは、第2のユーザの公開鍵またはDLTウォレットをトランザクションの受信者として識別してもよい。この場合、第2のクライアントアプリケーション530は、第1のDLT12における新しいトランザクションで第2のユーザが識別されることに気付いてもよい（第2のクライアントアプリケーション530が、第1のDLT12に追加される新しい各ブロックの内容を監視して、第2のユーザのDLT公開鍵またはDLTウォレットを識別する任意の新しいトランザクションを探すように構成されている場合）。あるいは、サードパーティのサービスがこの監視プロセスを実行して、第2のクライアントアプリケーション530に通知してもよい。次に、第2のクライアントアプリケーション530のアプリケーション機能層30は、適切なBPI「読み取り」機能を使用して、第1のDLT12からメッセージのハッシュを取得することができる。

40

50

【0100】

第2のクライアントアプリケーション530がメッセージハッシュのコピーを有すると、アプリケーション機能層30は、クライアントアプリケーションバックエンド510に対して第2のユーザを認証するために、何らかの形式の認証と共にそれをクライアントアプリケーションバックエンド510に通信してもよい（例えば、これは、第2のクライアントアプリケーション530およびクライアントアプリケーションバックエンド510が確立した認証プロセスの任意の形式を使用するか、または第2のユーザのDLT秘密鍵によって署名されたデジタル署名を提供することにより、クライアントアプリケーションバックエンド510は、第1のDLT240に保存されたメッセージハッシュに関連付けられた公開鍵を使用して確認することができる）。認証が渡された場合、メッセージハッシュに対応する完全なメッセージが第2のクライアントアプリケーション530に返されてもよい。次に、第2のクライアントアプリケーション530は、第1のクライアントアプリケーション520で使用されたものと同じハッシュアルゴリズムを介してそれを渡すことによって、返却されたメッセージの整合性を検証して（例えば、適切な「チェックメッセージ」BPI機能を使用するか、またはサービスオファリングプロバイダ310および／またはクライアントアプリケーション開発者320が適合すると見なす任意の他の方法で）、テストハッシュを作成し、次にそのテストハッシュを、第1のDLT12から取得したメッセージハッシュと比較してもよい。

10

【0101】

この特定のプロセス例の多くの利点は直ちに明らかであろう。まず、暗号化された電子メールや、2人のユーザ間でパスワードの合意を必要とするパスワードで保護されたドキュメントなど、複雑で確実に実行するのが難しいことが多い技術を行う必要がなく、メッセージの内容が第1のユーザから第2のユーザに安全に渡る可能性がある。さらに、第2のユーザ530は、さもなければテストハッシュが第1のDLT12に保存されたメッセージハッシュと一致しないため、クライアントアプリケーションバックエンド510が何らかの方法でメッセージを改ざんしていないことを確信することができる。さらに、将来の任意の時点で、メッセージの内容に関して第1のユーザと第2のユーザとの間に何らかの形の紛争がある場合、第1のDLT12に保存されたメッセージハッシュを使用して、紛争を完全に解決することができる。

20

【0102】

第2のユーザは同様に、類似の方法でメッセージに返信してもよい。

30

【0103】

これは、本開示のメッセージングの態様の単なる1つの特定の使用例であり、多くの他のものが可能であり得ることが理解されよう。さらに、上記の特定の使用例の多くの代替的な実装が可能である。例えば、アプリケーションバックエンドは代替的にサービスオファリング開発者310によって提供されてもよく、BPIは単にメッセージを受信するように構成されてもよく、オーバーレジャーコミュニケーション層20は、メッセージハッシュを第1のDLT12に置くこと、またアプリケーションバックエンド510にメッセージおよびメッセージハッシュを通信することを処理してもよい。この場合、クライアントアプリケーションのアプリケーション機能層30は、GUIおよびクライアントアプリケーションによって提供される他の機能など、他の機能のみに関係し得る。さらに、アプリケーション機能層30は、BPIによって提供される機能またはコマンドを利用して、1つだけでなく多くのDLTにメッセージハッシュを置いてよく、それによって、メッセージハッシュのさらなる冗長性およびバックアップを提供する。

40

【0104】

トランザクションの順序付けの態様

信頼できるブロックチェーン動作の重要な要件は、トランザクションの順序付けである。例えば、二重使用の暗号通貨詐欺を防止するために、新しいトランザクションがブロックチェーンネットワークにブロードキャストされるときに、その新しいトランザクションで識別された暗号通貨が以前に使われたかどうかを判定することが可能でなければならず

50

、新しいトランザクションが以前に使われていない場合にのみ新しいトランザクションを新しいブロックに含める（すなわち、二重の使用を防ぐ）ことが従来認識されている。多くの既存のブロックチェーンは、ブロックチェーン上のブロックの年代順の性質のおかげでこれを達成し、各ブロックにはタイムスタンプが含まれ、新しいトランザクションで識別された暗号通貨がまだ使われていないことを既存のブロックの中で確認する。

【0105】

しかしながら、本発明者らは、トランザクションの信頼できる順序付けが、暗号通貨ブロックチェーンだけでなく、より一般的には全ての種類のブロックチェーンにとって重要であり得ることを認識した。例えば、様々なインターネットおよび電気通信分野およびプロトコルにおいて、誤って順序付けられたメッセージングがセキュリティおよび安定性の弱点として悪用され得ることが理解されよう。例えば、TCP/IPパケットの順序と同期がシステムのセキュリティに影響を与え得る方法は数多くあり、例えば、メッセージが正しい順序から外れたり、メッセージセットが不完全であったりすると、整合性または可用性のいずれかの点でシステムのセキュリティが低下する。そのため、本発明者らは、ブロックチェーン内でトランザクションの正しい順序を維持することが、例えば、上記のメッセージングの態様に従ってメッセージを運ぶために使用する場合だけでなく任意の他の目的で使用する場合に、1つ以上のDLTを利用する安全、確実かつ不正に強いシステムを獲得する上で非常に重要であり得ると認識した。

【0106】

DLTで正しいトランザクション順序を維持する上での2つの特定の課題が本発明者らによって特定されている。

【0107】

1. ブロックチェーンが分岐する場合、フォークの短い方のブロングに現れた1つ以上のトランザクションがブロックチェーンから失われる可能性がある。当業者は、DLTが分岐する可能性、およびその場合のDLTの挙動の特徴を容易に理解するであろう。ただし、伝統的に、フォークによってトランザクションが失われた場合、そのトランザクションは、トランザクションが失われたと認識された場合に、参加しているノードのネットワークによってフォークのより長いブロングに後で追加される可能性があることに注意されたい。

【0108】

図6は、ブロックチェーン内のフォークの例示的な表現を示している。トランザクション1は、ブロックチェーンの「ブロング1」に現れているブロックB1の一部である。ただし、フォークの結果、ブロックチェーンがより長いブロングである「ブロング2」で続いているため、トランザクション1はブロックチェーンから事実上失われている。トランザクション2はブロックB3の一部であり、「ブロング2」に現れている。しばらくすると、トランザクション1は、「ブロング2」のブロックB8の一部としてブロックチェーンに追加される。したがって、トランザクション1は、ブロックチェーンにおいてトランザクション2よりも後に現れる。しかし、トランザクション1がトランザクション2に先行することが重要である場合（例えば、それらが連続したメッセージに関連している場合）、これはセキュリティまたは安定性の欠陥を引き起こす可能性があるか、あるいはブロックチェーンを利用しているアプリケーションにおいて不正の発生を可能にし得る。

【0109】

2. 前に説明したように、いくつかのシナリオでは、1つのアプリケーションが2つ以上の異なるDLTを利用することが望ましい場合がある。この一部として、あるDLTから別のDLTヘデータ（例えば、メッセージ（前述）、または暗号通貨の一部、またはDLTに保存されている他の種類の別の情報）を移動するために、クロスレジャートランザクション（XLT）を実行することが場合によっては必要であり得る。しかし、各ブロックチェーンは、そのブロックチェーン上の異なるブロック間の順序を維持するように開発されている場合があり（例えば、タイムスタンプを使用して）、各DLTが異なるタイムスタンプ手順を利用している可能性があるため、および／またはタイムスタンプに使用さ

れるクロックが同期していない可能性があるため、異なるブロックチェーンに現れるトランザクション間で順序を確実に識別することは非常に難しい場合がある。したがって、2つ以上（複数）のDLTを利用する場合、DLT上に現れるトランザクションの順序を決定および維持することは、DLTを使用しているアプリケーションのセキュリティと安定性にとって非常に重要であり得る。

【0110】

図7は、「検証セット」を使用してトランザクションの順序を維持するために本発明者らによって開発されたプロセスの例示的な方法ステップを示す。図8は、図7の方法ステップを理解するのに役立つ2つのDLTと検証セットの視覚化の例を示す。

【0111】

ステップ710において、第1のDLT12および第2のDLT14は、対象の新しいトランザクションが第1のDLT12または第2のDLT14に追加されたブロックに含まれるときを識別するために監視される。第1のDLT12および第2のDLT14は、それがDLTに追加されるときに各新しいブロックの内容を読み取ることによって監視され得る。対象のトランザクションを構成するものの関連性基準は、この方法の特定の実装の問題である。例えば、上記で開示したBPIは、BPIを利用してDLTに置かれている任意のトランザクションが、任意の他の手段によって（例えば、そのBPIを利用していない他のエンティティによって）DLTに置かれたトランザクションとそれを区別する特定の識別子を含むように構成され得る。特定の識別子は、例えば、トランザクションに使用される公開鍵、またはトランザクション内の何らかの他の種類の情報であり得る。その結果、関連性基準は、サービスオフリング開発者310によって提供されるサービスを使用している第1のDLT12または第2のDLT14に含まれる任意のトランザクション（例えば、第1のクライアントアプリケーション520または第2のクライアントアプリケーション530によって追加されるか、さらに言えば、必要に応じてDLTにトランザクションを置くためにサービスオフリング開発者が操作する何らかのサービスアプリケーションによって追加されるトランザクション）が関連性基準を満たし、かつ関連性があると識別され得るように、BPIによって含まれる識別子であり得る。これは、第1のDLT12および／または第2のDLT14が、多くの異なるエンティティに利用され得る汎用DLTである場合に特に有用であり得るが、サービスオフリング開発者310は、トランザクションの適切な順序を保証して、サービスオフリングのセキュリティと安定性を保護するために、そのサービスに関連するトランザクションを識別する必要がある。

【0112】

図8において、トランザクションA、C、E、Oは全て対象のトランザクションである。「トランザクションX」というラベルの付いたトランザクションは対象外である。

【0113】

新しいブロックが第1のDLT12または第2のDLT14のいずれかに追加され、ステップ710で関連するトランザクションを含むことが識別されると、ステップ720において、トランザクションの内容に少なくとも部分的に基づいて識別された各々についてトランザクション識別子（例えば、ハッシュ）が決定され得る。次に、この識別子が直近または「進行中」の検証セットに追加される。検証セットは、任意の適切なメモリ、データベース、データアレイ、クラウドストレージなどのデータストアに保存されている関連するトランザクションのオフチェーン記録の一部である。トランザクション識別子は、DLT内の関連するトランザクションへのポインタ（例えば、ハッシュポインタ）として機能し得る。新しい各トランザクションIDは、検出された順に「進行中」の検証セットに追加される。したがって、識別されたトランザクションの記録は、トランザクションが第1のDLT12に現れたか、第2のDLT14に現れたかに関係なく、トランザクションの相対順序を示す。その結果、両方のDLTを監視し、第1のDLT12または第2のDLT14に追加されるとすぐに関連する各トランザクションのトランザクション識別子を追加することにより、トランザクションの相対順序が1つ以上の異なるDLTをまたが

10

20

30

40

50

って記録され得ることがわかる。当業者は、「進行中」の検証セットが、関連するトランザクションの順序を維持する任意の適切なデータストレージ技術を使用して保存され得ることを認識するであろう。新しい関連するトランザクションが「進行中」の検証セットに追加されると、プロセスはステップ S 7 1 0 に戻り、第 1 の D L T 1 2 および第 2 の D L T 1 4 の監視を続けてもよい。

【0114】

特定の時間に、プロセスはステップ S 7 3 0 に進み、「進行中」の検証セット（例えば、「進行中」の検証セットの内容に少なくとも部分的に基づくハッシュまたは暗号化）の記録（検証セット識別子など）を第 1 の D L T 1 2 および第 2 の D L T 1 4 に保存するプロセスを開始してもよい。プロセスがステップ S 7 3 0 に進む特定の時間は実装の問題であり、例えば、直近の以前の検証セットの記録が D L T に追加されてから特定の時間が D L T の監視に費やされた後、または「進行中」の検証セットに特定の数の関連するトランザクションが追加された後、または直近の以前の検証セットの記録が D L T に追加されてから特定の数のブロックが第 1 の D L T 1 2 または第 2 の D L T 1 4 に追加された後などであってもよい。

【0115】

ステップ S 7 3 0 において、直近の先行する検証セットの内容の記録（例えば、ハッシュまたは暗号化）が第 1 の D L T 1 2 および第 2 の D L T 1 4 の両方に存在するか否かが判定される（例えば、どちらの D L T にもその記録が失われる原因となるフォークがないことを確認するため）。記録が両方の D L T に現れている場合（図 7 の「Y e s」）、プロセスはステップ S 7 5 0 に進む。記録が両方の D L T に現れていない場合（図 7 の「N o」）、プロセスはステップ S 7 4 0 に進む。この確認は、両方の D L T で直前の検証セットの記録を検索することにより実行され得る。

【0116】

ステップ S 7 4 0 において、欠落している記録は、それが欠落している D L T 上で再度行われ得る（本質的には、記録が最初に D L T 上に置かれたトランザクションが、例えば上記の「メッセージングの態様」を使用して、再度ブロードキャストされ、D L T に追加される）。ただし、欠落している記録を再度行う前に、直前の検証セット内で識別されたトランザクションのいずれかが、それが表れているべき対応する D L T から欠落しているかどうかを最初に判定してもよい。これは、トランザクションの記録に保存されている直前の検証セット内のトランザクション識別子のポインタの性質（例えば、ハッシュポインタ）を使用することにより、すばやく判定され得る。その後、任意の欠落しているトランザクションは、直前の検証セットの欠落している記録を再度行う前に、関連する D L T 上で再度行われ得る。任意の再度行われたトランザクションはここで、欠落していたトランザクションが先行していた（欠落していなかった場合の）トランザクションよりも後の D L T に現れる可能性があるが、検証セットはトランザクションの正しい順序の適切な記録を依然として維持しているであろう。結果として、欠落している検証セットの記録、およびそれらに関連付けられている任意の欠落しているトランザクションは、トランザクションが発生した適切な順序を失うことなく、D L T に再び追加され得る。このステップの詳細は、以下の図 8 および図 9 の説明からさらに完全に理解され得る。

【0117】

ステップ S 7 5 0 において、「進行中」の検証セットで識別された全てのトランザクションが関連する D L T に依然として現れているどうか判定される（例えば、D L T の一方または両方のフォークによって失われたものがないことを確認する）。それらが関連する D L T に全て依然として現れている場合（図 7 の「Y e s」）、プロセスはステップ S 7 7 0 に進む。それらが関連する D L T に全て現れていない場合（図 7 の「N o」）、プロセスはステップ S 7 6 0 に進む。この確認は、「進行中」の検証セット内の遷移識別子のポインタの性質によって実行され得る。

【0118】

ステップ S 7 6 0 において、現在欠落していると識別されたトランザクションは、それ

が現れるべきであった対応するDLT上で再度行われ得る（本質的に、再度ブロードキャストされ、DLTに追加される）。任意の再度行われたトランザクションはここで、欠落していたトランザクションが先行していた（欠落していなかった場合の）トランザクションよりも後のDLTに現れる可能性があるが、「進行中」の検証セットはトランザクションの正しい順序の適切な記録を依然として維持しているであろう。結果として、欠落したトランザクションは、トランザクションが発生した適切な順序を失うことなく、DLTに再び追加され得る。

【0119】

ステップS770において、「進行中」の検証セットの内容に少なくとも部分的に基づいて「進行中」の検証セットの記録が決定され（例えば、検証セットの内容に少なくとも部分的に基づくハッシュまたは暗号化などの検証セット識別子の決定）、第1のDLT12および第2のDLT14の両方に追加される。記録は、上記の「メッセージングの態様」で説明した手法を利用することにより両方のDLTに追加され得る。結果として、上記の「メッセージングの態様」を使用することにより「進行中」の検証セットの内容の記録を両方のDLTに追加でき、それにより、その検証セットの内容の効果的に不変の記録が提供され、それを使用して、関連するトランザクションの記録のデータ整合性、したがってセキュリティと信頼性の向上が実現され得ることがわかる。次に、プロセスはステップS710に戻り、DLTの監視を続け、次の「進行中」の検証セットの作業を開始し得る。

【0120】

図8に戻ると、検証セット $n-1810$ 、検証セット $n820$ 、および検証セット $n+1830$ を含む、オフチェーンのデータストア（任意の適切な種類のデータストア、データベース、クラウドストレージ、または任意の他の種類のメモリ）に保存された関連するトランザクションの記録840を見ることができる。検証セット $n-1810$ は、トランザクションYのハッシュとトランザクションZのハッシュとを含み、これらは、第1のDLT12または第2のDLT14上の以前のトランザクションであり、単純化のために図8では別段示されていない。検証セット $n820$ は、トランザクションAのハッシュ822'と、トランザクションOのハッシュ824'と、トランザクションCのハッシュ826'とを含む。トランザクションA822およびトランザクションC826は、第1のDLT12に現れ、トランザクションO824は第2のDLT14に現れる。それらが検証セット $n820$ に現れる順序は、トランザクションO824が、トランザクションA822の後に検出されたが、トランザクションC826の前に検出されたことを示す。検証セット $n+1830$ はトランザクションEのハッシュ832'を含み、この例では「進行中」の検証セットと見なすことができる。

【0121】

見てわかるように、検証セット $n-1810$ 'のハッシュおよび検証セット $n820$ 'のハッシュは両方とも、上記のステップS770に従って、両方のDLTに以前に追加されている。検証セット $n820$ が「進行中」の検証セットであった場合、それが上記のステップS730に来ると、検証セット $n-1810$ 'のハッシュが両方のDLTに存在するため、プロセスはステップS750に直接進むことができると理解される。同様に、それがステップS750に来ると、トランザクションA822、トランザクションO824およびトランザクションC826は全て関連するDLTにまだ現れているため、プロセスはステップS770に直接進み、両方のDLTに検証セット $n820$ 'のハッシュが追加され得る。

【0122】

このプロセスの結果として、2つのブロックチェーンに現れるトランザクションの順序のオフチェーン記録が、トランザクションの記録840内の検証セットによって保持されることがわかるであろう。オフチェーン記録は、それらが関連するトランザクションの相対順序を示す。上記の「メッセージングの態様」も使用して各検証セットのハッシュポイントを両方のDLTに保存することにより、各検証セットの記録がDLTに保持され得て、これは検証セットのオフチェーン記録のデータの整合性およびセキュリティを維持する

10

20

30

40

50

上で、また以下で説明するように第1のDLT12および／または第2のDLT14でフォークが発生した場合に特に役立ち得る。

【0123】

図9は、図7の方法ステップ、特にステップS730～S760の理解を助けるための2つのDLTおよび検証セットのさらなる例示的な視覚化を示す。図9は図8に類似しているが、図9には第1のDLT12のフォークの表現が含まれており、これにより2つのトランザクションおよび検証セットの1つの記録が第1のDLT12から失われている。

【0124】

図9では、トランザクションC826を含める前に、第1のDLT12がフォークする。トランザクションC826、検証セット n のハッシュ820'、およびトランザクションE832は全て、フォークの「ブロング1」に現れている。ただし、この例では、DLT12として前進するのはフォークの「ブロング2」であり、「ブロング1」の内容は（少なくとも一時的に）実質的に失われる。検証セット n のハッシュ820'をDLTに追加すると、検証セット $n+1$ 930で作業が開始され、これは見てわかるようにトランザクションEのハッシュ832'およびトランザクションGのハッシュ934'をこの順序で含む。図9で識別されるポイントPで、図7のプロセスはステップS730に進み、その時点で、検証セット n 820'およびトランザクションC826のハッシュが第1のDLT12にもはや現れないことが識別される（それらは両方とも「ブロング1」に現れ、これは失われたフォークのブロングであるため）。その結果、ステップS740において、トランザクションCが再度行われ（図9では参照番号926で識別されているが、元のトランザクションC826と再度行われたトランザクションC926の内容は同じであることが理解されるであろう）、次に検証セット n のハッシュが再度行われる（図9では参照番号920'で識別されているが、検証セット820'の元のハッシュは、再度行われた検証セット920'のハッシュと同じであることが理解されるであろう）。その後、ステップS750において、トランザクションE832（「進行中」の検証セット $n+1$ 930で識別されている）は「ブロング1」に含まれていたため、第1のDLT12にもはや現れないことが決定される。したがって、ステップS760において、トランザクションEが再度行われる（図9では参照番号932で識別されているが、元のトランザクションE832と再度行われたトランザクションE932の内容は同じであることが理解されるであろう）。その後、ステップS770において、検証セット $n+1$ のハッシュ930'が両方のDLTに追加される。したがって、トランザクションC826、検証セット n のハッシュ820'、およびトランザクションE832が第1のDLT12から全て失われたとしても、それらはトランザクションの真の順序の信頼できる記録を失うことなくトランザクションG934（トランザクションC826およびトランザクションE832の後に最初に行われた）の後に第1のDLT12に再び追加され、それによってシステムのセキュリティと安定性が保証されることがわかる。その結果、検証セットのハッシュをDLTに含めることはオプションであるが（別の方法では、トランザクションのオフチェーン記録は、検証セットのいずれのハッシュもDLTに保存せずに単に保持され得る）、各検証セットの記録をDLTに保存することは、オフチェーン検証セットのデータの整合を可能にすることによってセキュリティと信頼性の向上を可能にするだけでなく、フォークをすばやく識別し、欠落している記録および／またはトランザクションに関連するDLTに再度追加することも可能にすることが理解されよう。

【0125】

図9に示す各ブロックは単一のトランザクションのみを含むが、実際には各ブロックが多くのトランザクションを含む可能性が高いことが理解されよう。さらに、上記は2つのDLTを使用するプロセスを説明しているが、プロセスは単一のDLT、または3つ以上の異なるDLTに適用されることが理解されよう。さらに、上記では、検証セットおよびトランザクションの「ハッシュ」が説明されているが、検証セットの内容に少なくとも部分的に基づいて決定される任意の適切な検証セット識別子、およびトランザクションの内容に少なくとも部分的に基づく任意の適切なトランザクション識別子を代わりに使用し

10

20

30

40

50

てもよい。

【0126】

さらに、上記のプロセスでは検証セットのハッシュが両方のDLTに追加されるが、別の方法ではハッシュはDLTのうちの1つのみに追加されてもよい。すなわち、プロセスが動作している2つ以上のDLTに検証セットのハッシュを含めることが望ましい場合があり、これは、フォークが発生した場合でも、直前の検証セットの少なくとも1つの記録が常に存在するはずであるためである（2つのDLTが同時に独立して分岐し、両方が直前の検証セットのハッシュを失う原因になることは極めてあり得ない）。これはデータの整合性、すなわちセキュリティの理由で有用であり得る。

【0127】

さらに、関連するトランザクションのハッシュを検証セットに保存することに加えて、またはその代わりに、関連するトランザクションの完全なコピーを保存してもよい。ただし、データストレージの効率を高めるために、関連するトランザクションのハッシュを保存の方が望ましい場合がある。一実装例では、関連する各トランザクションのハッシュを保存してもよく、かつ最近の関連するトランザクションの完全なコピーを保存してもよく（例えば、直前の1つ、2つ、3つ、4つの検証セットのトランザクション）、これは欠落しているトランザクションを再度行うのに役立ち得る。最近の関連するトランザクションのみを保存することにより、データストレージ要件をなおも最小限に抑えることができる。

【0128】

さらに、欠落しているトランザクションの識別と再実行はオプションであるが、DLTが可能な限り最新であり、DLT上のトランザクションの順序が、トランザクションの記録840で識別されるように実際のトランザクションの順序に可能な限り密接に対応していることを保証するために、これらの手順を実行することが望ましい可能性がある。

【0129】

クロスレジャートランザクションの態様

上述したトランザクションの順序付けの態様の重要性をさらに理解するのを助けるため、次に、トランザクションの順序付けの態様を利用する例示的なクロスレジャートランザクション(XLT)プロセスについて説明する。

【0130】

この例では、第1のDLT12のトランザクションに現れる情報を第2のDLT14に転送することが望まれる。その情報は、例えば、暗号通貨の量、ハッシュメッセージ（上記の「メッセージングの態様」で説明したとおり）、または任意の他のデータなど、何でもよい。トランザクションは、第1の所有者から第2の所有者に情報を転送する場合もあれば、所有権がまったく移転されない場合もある（例えば、エンティティは単に別のDLTに情報を保存して、以前のDLTから情報を効率的に削除したい場合がある）。

【0131】

図10は、2つのDLTの視覚化の例と、XLTが実行される、オフチェーンに保存されたトランザクションの記録1040を構成する対応する検証セットを示す。トランザクションS1012は、第1のDLT12から第2のDLT14に転送される情報を含む。検証セット $n-1$ 1010は、トランザクションSのハッシュ1012'を含む。XLTは、2フェーズコミットプロセスを使用して実行され得る。2フェーズコミットプロセスは当業者にはよく理解されているであろうが、それでも完全を期するため以下に簡単に説明されている。トランザクションSの情報をDLT12からDLT14に転送するための提案が最初にDLT12に追加される（図10の「プロポーズド(proposed)XLT(S)1022」で識別される）。これによりトランザクションS1012が効果的にロックされ、他のいずれのトランザクションでも使用できなくなる。その後、意図された受信者（状況によっては、先に説明したように、転送される情報の現在の所有者と同じエンティティであり得る）は、DLT14で転送を受け入れる準備ができていることを示してもよい（「レディ(ready)XLT(S)1026」で識別される）。別の方法では

10

20

30

40

50

、いくつかのDLTプロトコルでは第2のDLT14で「プロポーズドXLT(S)1022」を再度行うことによって準備ができていないことを示してもよい。準備ができていないことが第2のDLT14に示されると、DLTの両方(図10では「コミット(commit)XLT(S)1032」および「コミットXLT(S)1034」で識別される)に追加されるコミットトランザクションによって、クロスレジャートランザクションが実行され得る。当業者によってよく理解されるように、このコミットメントは、取引されたアイテムを第1のDLT12から効果的に削除し、それを第2のDLT14に置く。

【0132】

不正やシステム障害のリスクなしにXLTを安全かつ確実に実行するために、XLTを実行できる前に、コミットトランザクションが2つのDLTに置かれる前に第2のDLT14に準備ができていないことを示す表示が現れることを確認する必要がある。さらに、準備ができていないことの表示が常に提案の後に続き、2つを適切に照合して認証できるようにすることが望ましい可能性がある。このようにして、準備ができていないことの表示が、何らかの他の類似しているが異なるトランザクション(例えば、同じパーティ間のXLT提案に関連しているが、DLT12上の異なる以前のトランザクションの内容に関するもの)に実際には関連していないことを保証することができる。また、準備ができていないことの表示によって、XLTの詳細の一部が不正に変更されないことを保証することもできる。

【0133】

従来、前に説明したように、特に1つ以上のDLTでフォークが発生した場合、異なるDLT上でトランザクションの正しい順序を判断することは非常に困難であった。その結果、安全で簡単かつ確実な方法でXLTを実行することは困難であった。しかし、上記のトランザクションの順序付けの態様を利用することにより、1つ以上のDLTでフォークが発生した場合でも、関連するトランザクションの正しい順序が記録され得る。

【0134】

例えば、トランザクションの順序付けの態様を使用して、例えば、プロポーズドXLT(S)1022がレディXLT(S)1026に対応していること(およびオプションで両方ともそれぞれ第1のDLT12と第2のDLT14上に依然として現れていること)、かつ(フォークが発生した場合でも)プロポーズドXLT(S)1022がレディXLT(S)に先行していることの両方など、XLT基準が満たされていることが保証され得る。この点で、コミットXLT(S)1032トランザクションは、XLT基準が満たされた後にのみ、DLTに追加され得る。必要に応じて、XLT基準は、プロポーズドXLT(S)1022とレディXLT(S)1026トランザクションを識別する検証セットのハッシュが一方または両方のDLTに存在するという基準を含んでもよく、これにより、プロポーズドXLT(S)1022およびレディXLT(S)1026トランザクションの両方がそれぞれのDLTに存在することを保証し得る。この例では、プロポーズドXLT(S)のハッシュ1022'とレディXLT(S)のハッシュ1026'は同じ検証セットに現れるが、場合によっては別の検証セットに現れ得ることが理解される。プロポーズドXLT(S)のハッシュ1022'が、レディXLT(S)のハッシュ1026'が現れる検証セットに先行する検証セットに現れる場合、正しい順序を確認できる。プロポーズドXLT(S)のハッシュ1022'が、レディXLT(S)のハッシュ1026'が現れる検証セットの後に続く検証セットに現れる場合、トランザクションが間違った順序であることが確認できるため、XLTは発生しない。

【0135】

図10に見られるように、検証セットn1020は、プロポーズドXLT(S)のハッシュ1022'を含み、その後にレディXLT(S)のハッシュ1026'が続く(検証セットn1020は、関連すると見なされる他のトランザクションのハッシュも含むが、それらはここで説明しているXLTとは関係がない。実際には、検証セットを構築するエンティティは、DLTで多くの関連するトランザクションを識別し、検証セットでそれらを識別する可能性がある。それらの関連するトランザクションの一部は、特定のXLTに関

10

20

30

40

50

連するトランザクションなど、互いに関連している可能性があり、他のものは、例えば異なるXLTや、エンティティが関連性があると見なす任意の他の種類のDLTトランザクションに関連しているなど、互いにどのようにも関連していない)。検証セット_nのハッシュ1020'は両方のDLTに現れ、その後、第1のDLT12のコミットXLT(S)1032と、第2のDLT14のコミットXLT(S)1034が続く(これらはその後、次の検証セット_{n+1}1030で識別される)。

【0136】

この種類のXLTは、様々な他の方法で実装され得る。図11は、サービスオフリング開発者310を含む1つの特定の実装例を示し、サービスオフリング開発者は、この特定の例では、XLTを実行するためのサービスアプリケーション(またはソフトウェアパッケージ/モジュール)1110を開発し、また、クライアントアプリケーション開発者320にBPIを提供し、クライアントアプリケーション開発者が特定のXLTクライアントアプリケーションを開発できるようにする(代替として、サービスオフリング開発者310自身がクライアントアプリケーションを作成してもよく、その場合にはBPIはいかなる外部開発者にも提供されない)。第1のクライアントアプリケーション1120および第2のクライアントアプリケーション1130は、第1のエンティティ1140および第2のエンティティ1150がクライアントアプリケーション開発者320からそれぞれ入手したXLTクライアントアプリケーションの複製である。この例のクライアントアプリケーションは、アプリケーション機能層30を効果的に実装することができ、ソフトウェアパッケージまたはモジュール1110は、オーバーレジャー通信層20を効果的に実装することができる。

【0137】

図12は、図11に示されているシステム例によってXLTが実行され得る例示的なステップを示す。

【0138】

ステップS1210において、サービスアプリケーション1110は、XLTを実行して第1のDLT12に現れている情報のアイテムを第2のDLT14に転送して所有権を第2のエンティティ1150に転送するために、第1のクライアントアプリケーション1120から要求を受信する。理解されるように、要求は、BPIで定義されているように、XLTを実行するために必要な任意の情報を含み得る。

【0139】

ステップS1220において、サービスアプリケーション1110は、第1のDLT12に追加するために、第1のDLTネットワーク240に対してプロポーズドXLTトランザクションをブロードキャストすることができる。

【0140】

次に、ステップS1230において、サービスアプリケーション1110は、第2のDLT14に追加するために、第2のDLTネットワーク250にレディXLTトランザクションをブロードキャストすることができる。オプションで、サービスアプリケーション1110は、XLTを実行する意図を第2のクライアントアプリケーション1130に伝えることができ、次に、トランザクションが行われるべきであるという第2のアプリケーション1130からの確認を受け取った後にのみステップS1220またはS1230を実行する。

【0141】

その間、サービスアプリケーション1110は、上記のトランザクションの順序付けの態様を実行し、関連するトランザクションの順序を示す関連するトランザクションの記録をデータストア1115に保存している。

【0142】

ステップS1240において、サービスアプリケーション1110は、プロポーズドXLTトランザクションが、対応するレディXLTトランザクションの前に行われたものとしてトランザクションの保存された記録内において識別されることと、オプションとして

、関連する検証セットのハッシュが一方または両方のDLT内に現れたことも確認できる。
【0143】

この確認に合格すると、ステップS1250において、サービスアプリケーション1110は、第1および第2のDLTに追加するために、第1のDLTネットワーク240および第2のDLTネットワーク250にコミットXLTトランザクションをブロードキャストする。オプションで、サービスアプリケーション1110は、第1のクライアントアプリケーション1120および／または第2のクライアントアプリケーション1130に、成功したXLTを通知してもよい（これは両方のDLTにコミットXLTトランザクションが現れた後、またはコミットXLTトランザクションが識別された検証セットのハッシュがDLTの一方または両方に含まれた後に実行され得る）。確認に合格しない場合、一定期間の後にXLTプロセスはタイムアウトしてもよく、第1のクライアントアプリケーション1120および／または第2のクライアントアプリケーション1130に障害が通知され得る。

10

【0144】

したがって、上記の順序付けの態様を実行することによって、そして次に検証セットでプロポーズドXLTトランザクションおよび対応するレディXLTトランザクションが正しい順序で行われていることが検証された後にコミットトランザクションをDLTにポストするだけで、サービスアプリケーション1110は、XLTが安全にかつ安定して実行されることを保証し得る。

【0145】

これは単に1つの特定の非限定的な例であることが理解されるであろう。代替案では、サービスオフリング開発者310によって提供される特定のBPIおよび／またはクライアントアプリケーション開発者320によって提供されるクライアントアプリケーションの実装に応じて、第1のクライアントアプリケーション1120自体が第1のDLTネットワーク240にプロポーズドXLTトランザクションをブロードキャストしてもよく、および／または第2のクライアントアプリケーション1130自体が第2のDLTネットワーク250にレディXLTトランザクションをブロードキャストしてもよい。この場合、サービスアプリケーション1110は上述のトランザクションの順序付けの態様を単に実行し、次に、関連するDLTにおいてプロポーズドXLTトランザクションおよび対応するレディXLTトランザクションを識別したときにステップS1140およびS1150を実行してもよい。さらに、XLTは1つのエンティティのみを含む場合があり、これはエンティティが、所有権の任意の変更を伴って、あるDLTから別のDLTに情報を単に移動する場合があるためである。

20

30

【0146】

オーバーレジャーSDKは、トランザクションの順序付けの態様およびオプションでXLTの態様も実行するためのルールおよび方法を含み得ることが理解されよう。このように、オーバーレジャーSDKは、オーバーレジャーSDKでサポートされている任意の選択したDLTでトランザクションの順序付けの態様を（およびオプションでXLTの態様も）利用する任意のサービスアプリケーション／モジュールをサービスオフリング開発者310が実装できるツールを提供する。次に、選択したものの対応するBPIを作成して、サービスオフリングの任意の適切な機能をクライアントアプリケーション開発者320が利用できるようにして、順序付けの態様を実行する際のDLTとの相互作用がオーバーレジャー通信層20によって実行され、クライアントアプリケーション上の任意のアプリケーション機能層30の動作から切り離され得るようにする。結果として、クライアントアプリケーションは、DLTとの直接の相互作用から切り離されたまま、トランザクションの順序付けの態様によって得られるセキュリティと安定性の利点を利用できる。

40

【0147】

その結果、トランザクションの順序付けの態様を実行するためのルールおよび方法を含むオーバーレジャーSDKを提供することにより、ブロックチェーンの相互作用をアプリケーション機能層30からなおも切り離しつつ、トランザクションの順序付けの態様の技

50

術的利点を幅広い様々なエンティティが様々な方法で利用する大きな柔軟性と機会が存在することがわかる。

【0148】

当業者は、本開示の範囲から逸脱することなく、本開示の上記の態様に対して様々な変更または修正を行うことができることを容易に理解するであろう。

【0149】

図2、図3、図5、および図11に矢印で示されている全てのインターフェースは、異なるエンティティとモジュールの各々の間の直接接続を示しているが、例えば通信ルータなど、これらのインターフェースの一部として任意の数の中間エンティティまたはモジュールが存在し得ることが理解されよう。さらに、各インターフェースは、任意の適切な通信アーキテクチャおよびプロトコルに従って、エンティティおよびモジュールの各々の間でデータ／通信を伝達することがある。

【0150】

上記の全てで説明した本開示の態様は、ソフトウェア、ハードウェア、またはソフトウェアとハードウェアの組合せによって実装され得る。例えば、トランザクションの順序付けの態様および／またはXLTの態様および／またはオーバーレジャーメッセージング層は、コンピュータ可読コードを含むソフトウェアによって実装されてもよく、ソフトウェアは、任意の電子デバイス（例えば、サーバ、デスクトップコンピュータ、またはラップトップコンピュータなどの任意の標準的なコンピューティングデバイス）の1つ以上のプロセッサ（1つ以上のマイクロプロセッサ、またはプログラム可能なロジックなど）で実行されると上記の機能を実行する。ソフトウェアは、任意の適切なコンピュータ可読媒体、例えば、読み取り専用メモリ、ランダムアクセスメモリ、CD-ROM、DVD、ブルーレイ、磁気テープ、ハードディスクドライブ、ソリッドステートドライブおよび光学ドライブなどの非一時的コンピュータ可読媒体に保存されてもよい。コンピュータ可読媒体は、コンピュータ可読命令が分散された方法で保存および実行されるように、ネットワーク結合されたコンピュータシステムにわたって分散されてもよい。

10

20

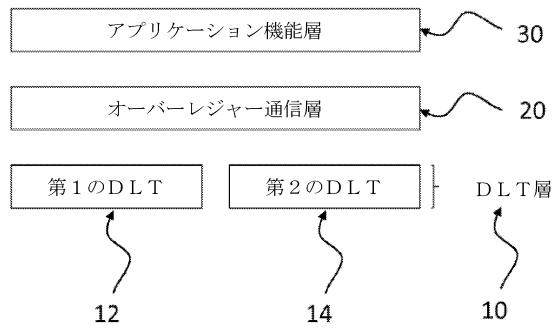
30

40

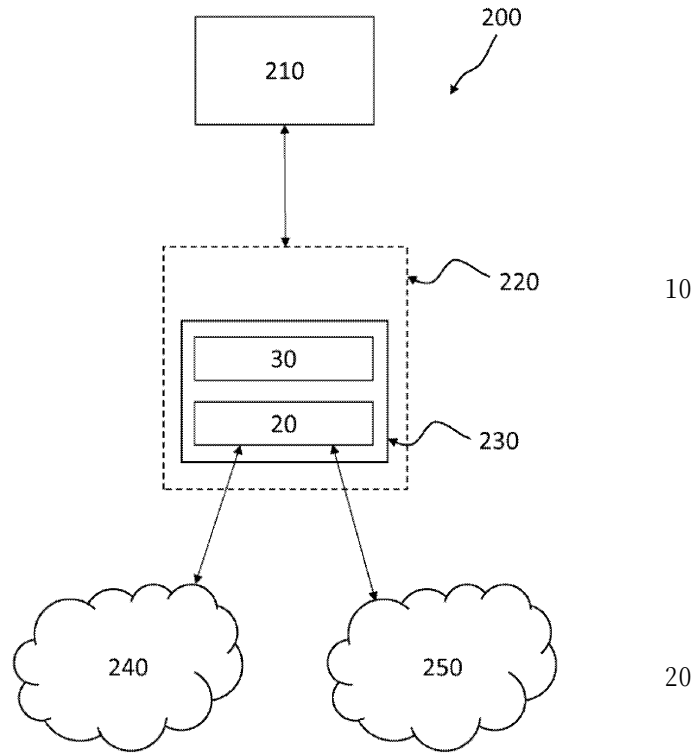
50

【図面】

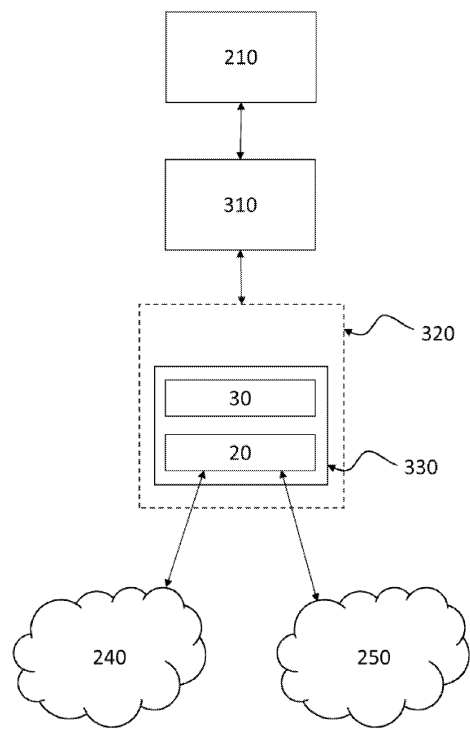
【図 1】



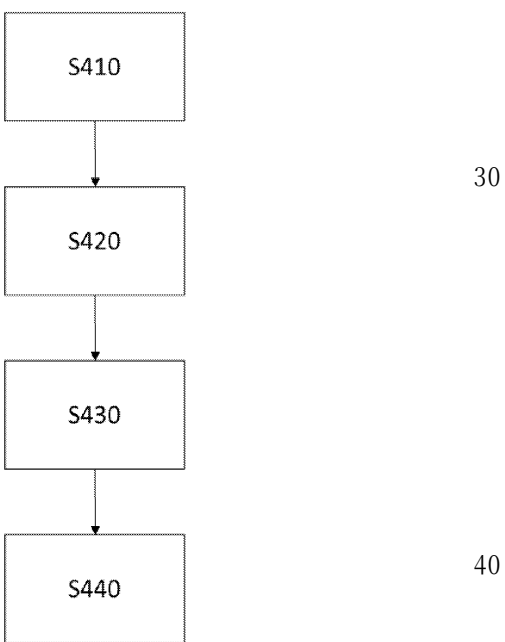
【図 2】



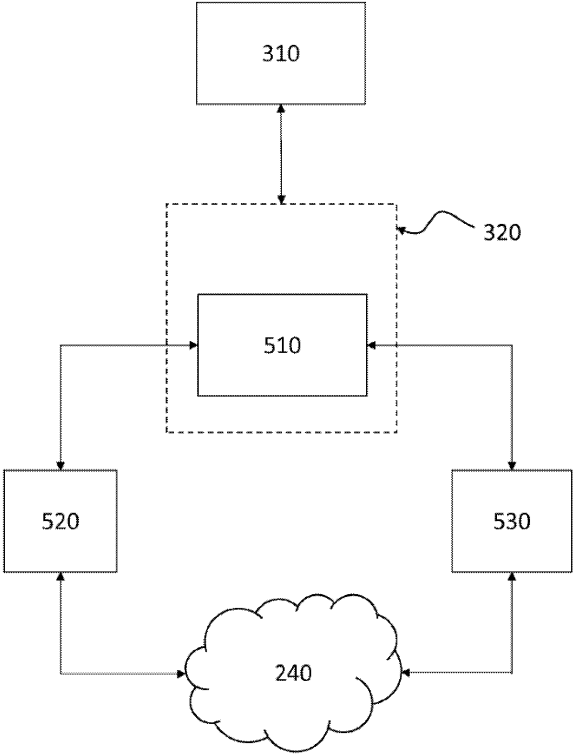
【図 3】



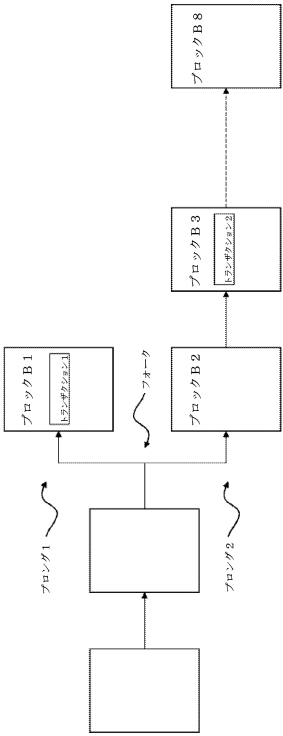
【図 4】



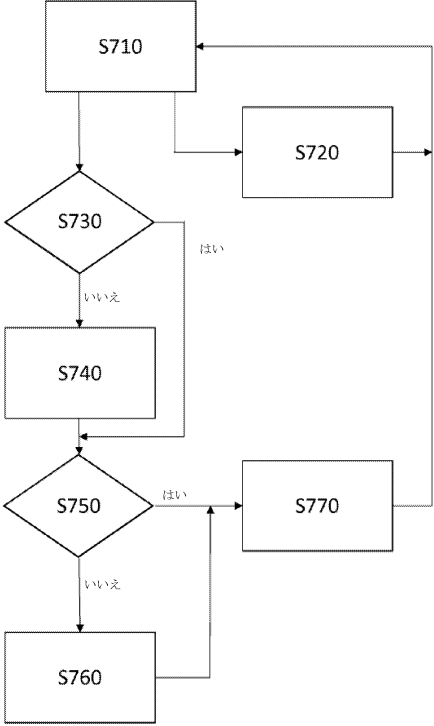
【図 5】



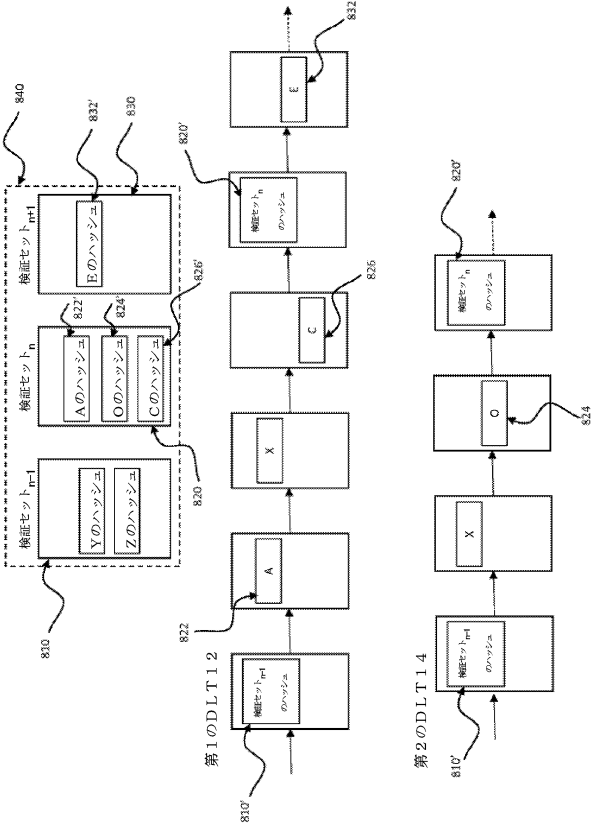
【図 6】



【図 7】



【図 8】



10

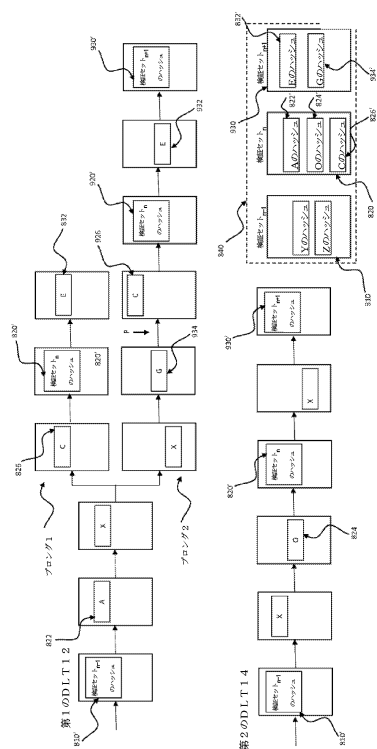
20

30

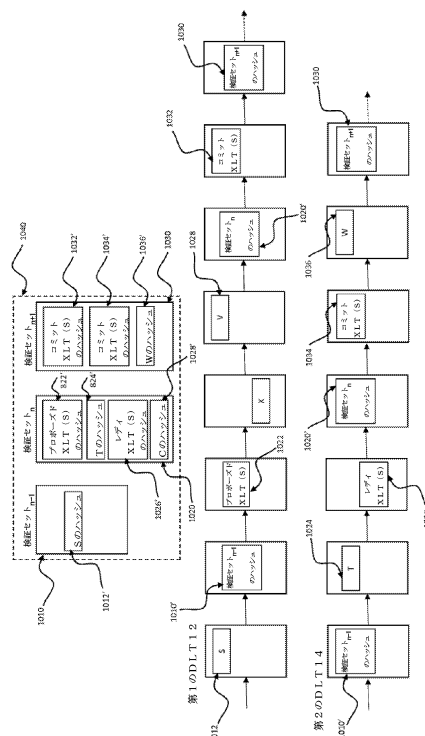
40

50

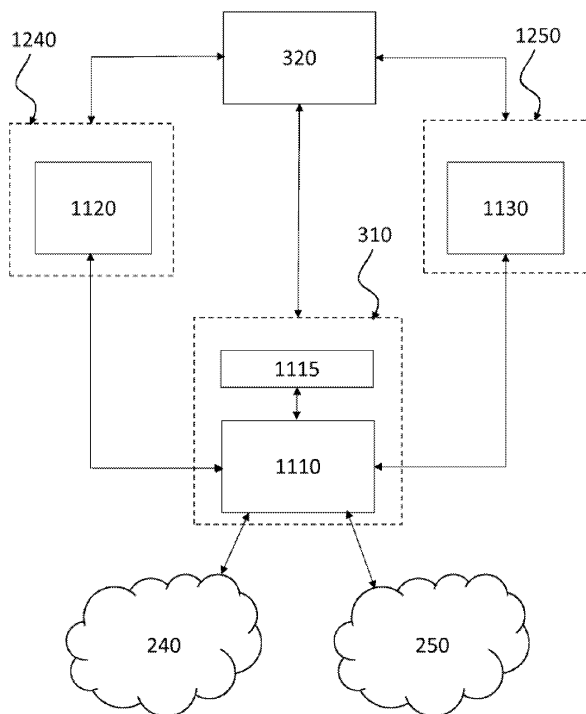
【図 9】



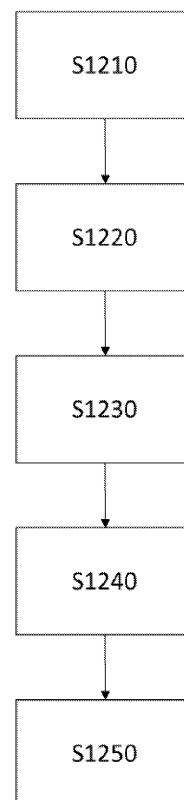
【図 10】



【図 11】



【図 12】



フロントページの続き

(33)優先権主張国・地域又は機関

イタリア(IT)

- 22 クアント ネットワーク リミテッド内

(72)発明者 モンデッリ, ガエターノ

イギリス グレーター ロンドン ロンドン N1 7GU ウェンロック ロード 20-22 クアント ネットワーク リミテッド内

(72)発明者 タスカ, パオロ

イギリス グレーター ロンドン ロンドン N1 7GU ウェンロック ロード 20-22 クアント ネットワーク リミテッド内

審査官 行田 悦資

(56)参考文献 米国特許出願公開第2017/0213209 (US, A1)

国際公開第2017/163220 (WO, A1)

(58)調査した分野 (Int.Cl., DB名)

H04L 9/32