Ontology

テクニカルホワイトペーパー

バージョン 0.6.6

更新日：2018/04/13

新高性能パブリックマルチチェーンプロジェクトと分散型信用共同プラットフォーム

要旨

これまでの歴史において、人々は技術、法律、コミュニティなどを通じて信頼関係を築いてきました。

しかし、このようなエンティティ間の信頼と協力は、複数ソースと独立したシステムに関わるもので、非常に高いコストがかかる可能性があるため、協力に関する可能性のその深まりと広がりを阻害しています。近年、インターネット技術が目覚ましく進歩していますが、あまりにも多くの要因が依然として信頼関係を妨げています。これには、信用システムの断片化、個人の役割の欠如、不正確な身元確認、誤った情報によるとトラブルなどが含まれます。社会ガバナンス、経済協力、金融サービスなどの分野では、日々信頼関係を確立するコストは膨大です。

分散型の改ざん防止ブロックチェーンは、特定の領域でテクノロジーによって信頼関係をもたらしました。しかし、多様な信頼システムとアプリケーションを単一の新しい信頼エコシステムに結合するために、さらに統合的なメカニズムが必要です。

オントロジーは分散型アプリケーション開発にインフラストラクチャを提供するだけでなく、信用とデータソースの効果的な調整を行うことで、信用エコシステムにおける連携インフラストラクチャを確立します。[1]

本書では、オントロジーの技術的フレームワーク、主要な技術原則、主要プロトコルについて重点を置きます。

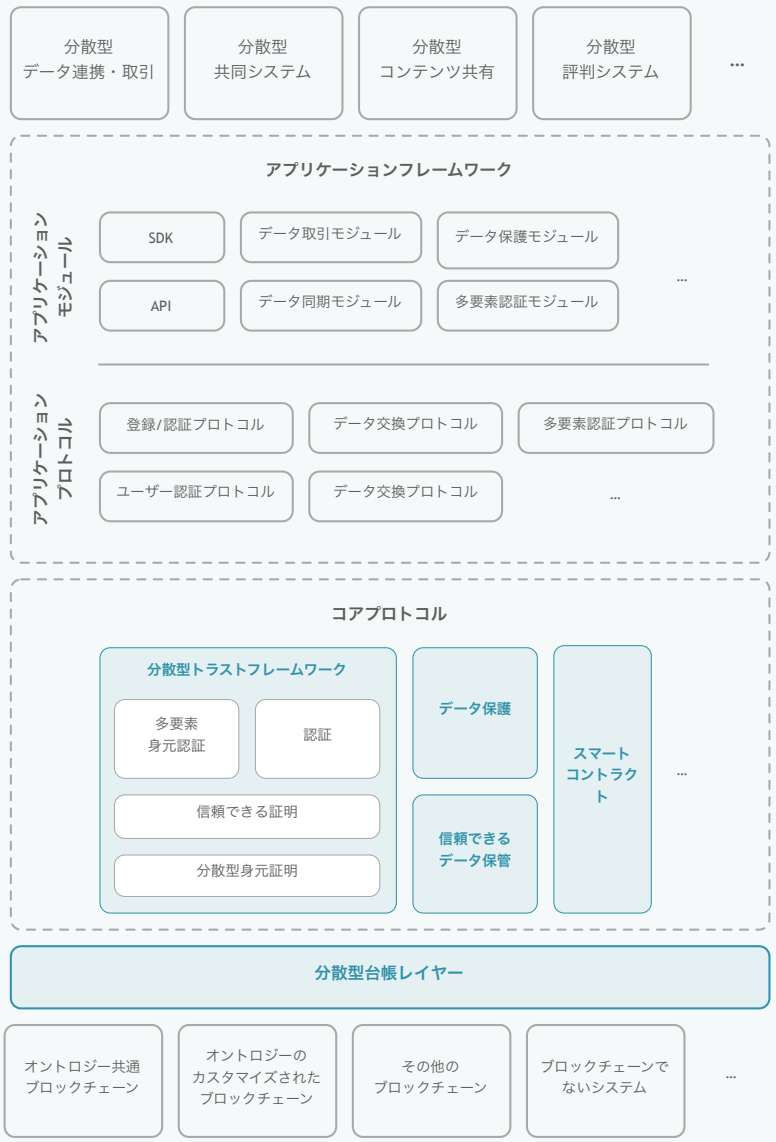
---

1. オントロジーは、様々な共同トラストシナリオをサポートします。シナリオやアプリケーションの必要性に対応し、モジュールとプロトコルを継続的に拡大させます。本テクニカルホワイトペーパーでは、第一段階におけるオントロジーのアーキテクチャとプロトコルを説明しています。アプリケーション開発に従い、テクニカルホワイトペーパーは引き続き更新されます。

目次

1. 概要

オントロジーは、異なるチェーンや伝統的な情報システム間のマッピングを可能にするために、オントロジーのプロトコルを通過するさまざまな産業と地域で構成され、統合されたマルチチェーンとマルチシステムのフレームワークです。この理由から、オントロジーはブロックチェーンの間のコネクターである「オントロジーチェーングループ」や「オントロジーチェーンネットワーク」と呼ばれています。



図表1.1: オントロジーテクノロジーフレームワーク

オントロジーの中核は分散型元帳、スマートコントラクトシステム、セキュリティシステムを含む、完全な分散型元帳システムです。分散型元帳は、オントロジーの重要かつ基礎的なストレージインフラです。分散型元帳技術における分散的、共同運営的、改ざん防止機能という特徴は、ネットワークにおけるエンティティ間の信頼関係を作り上げるための鍵となります。分散型元帳には、コンセンサスやスマートコントラクトシステムを含んでおり、上層アプリケーションフレームワークのためにコンセンサス、ストレージ、スマートコントラクトのサポートを提供しています。オントロジーとその分散型元帳技術は、分離型構造をしており（オントロジー元帳のデフォルト）、下位層としてNEO、イーサリアムなど、他のブロックチェーンをサポートすることができます。元帳レベルでは、私たちはストレージとビジネスロジックを分離させる共有データコントラクトモデルを創造的に作り上げ、全体的なアーキテクチャの拡張性や柔軟性を向上させるために、スマートコントラクトを利用することで異なるビジネスロジックを実装します。

分散型トラストフレームワークは、オントロジーの中核的なロジック層です。私たちはONT IDを使用して、異なるID認証サービスと接続し、人々、お金、モノ、その他に対する信頼の源を提供します。ONT IDは分散型であり、自主管理、プライバシー保護、安全の面で利用しやすい特徴があります。トラストフレームワークは検証可能な宣言[1][2]を用いて、分散トラストモデルと分散トラストデリバリシステムを確立し、検証可能な宣言のプライバシー保護を保証するために、CL署名アルゴリズムとゼロ知識証明プロトコルを使用します。

オントロジーは、さまざまなプロトコル基準を使用します。アイデンティティプロトコル、マルチソース認証プロトコル、ユーザー認証プロトコル、分散型データ交換プロトコルなどが含まれます。W3CによるデザインのDID[3]のように国際的に互換性のあるプロトコルの範囲において実行されます。暗号化署名プロトコルもまた、中国の暗号化基準であるRSAやECDSAなどの暗号化をサポートします。分散型データ交換システムは目標であるオープン性やスタンダード化の条件を満たすために、また将来的な更なるエコシステムな協力や拡大をサポートするためにOauth[4] やUMA[5]のような世界中で認証プロトコルとして使用されているものと互換性があります。

アプリケーションサービスの提供のために、オントロジーは、アプリケーション開発者が分散システムを開発する方法を知らずとも、オントロジーの上に直接分散型サービスを提供するためのインフラストラクチャを提供します。簡単に言うと、オントロジーはAPI、SDK、さまざまなアプリケーションモジュールを含む一連のアプリケーションフレームワークを提供しており、さまざまな技術的背景を持つアプリケーションサービス提供者が独自のdAppを開発し、dAppをサービス（DAAS）として使用できます。これにより、ブロックチェーンが、すべての人に使いやすくなります

オントロジーには、暗号化およびデータ保護モジュール、データ交換市場、グローバルトランザクションデータベース、ハイブリッドオラクル、無制限コンセンサスエンジンなど、さまざまな高度なモジュールも含まれています。将来、オントロジーは、オントロジーのエコシステムの技術開発を促進するために、開発者とビジネスパートナーのコミュニティを開発し、アプリケーションとモジュールを継続的に充実させます。

2. 用語

オントロジーチェーングループ

「オントロジーチエーンネットワーク」としても呼ばれ、異なる業界や地域に拠点を置くエンティティのチエーンにより、オントロジー内で組み合わされたチエーンです。各チエーンは、それぞれの分散型元帳を使い、インタラクティブなプロトコルを通じて連携し、動作します。

オントロジー分散型元帳

オントロジーの分散型元帳/ブロックチエーンフレームワークの一つとして、パブリックサービスチエーンの中核です。オントロジーすべてのサービスに対して、コアな分散型元帳、スマートコントラクト、その他サービス提供します。

エンティティ

ONT IDを使用することで、身元確認ができるオントロジーの参加者です。

ONT ID

ONT IDは、エンティティのIDサービスからのデータに基づいて、マッピングサービスおよび他のリンクに接続された分散型分散識別プロトコルです。分散型、自主管理、プライバシー保護、安全で使いやすいという特徴を持っています。

検証可能な要求

あるエンティティがもう一つのエンティティ（自身を含む）に対して行われた要求を確認するための宣言です。この要求には、認証のために他のエンティティによって使用されるデジタル署名が付いています。

オントロジートラストフレームワーク

分散型身元認証プロトコル、分散型トラストモデル、分散型トラスト伝達、その他モジュールを含む、オントロジーのトラストエコシステムを構成するモジュールのことです。

多要素認証

同じエンティティのさまざまな側面をカバーするさまざまな検証を参照して、多要素による検証を作成します。

トラストアンカー

検証を行うことを委任されたエンティティのことです。これは、信頼性の高い配信チェーンの提供元となり、身元確認サービスを提供します。

分散型の一貫性した元帳

共同ノードとして維持されている分散型P2Pネットワークにおける増幅変更データストレージメカニズムです。透明性と不正防止機能は、オントロジーをサポートする信頼されたストレージとスマートコントラクトを提供します。

コンセンサス

各ノードは一貫性を確保するために、プロトコルに従って元帳へ書き込むデータを確認します。

スマートコントラクト

元帳に記録された実行可能なコードは、元帳ノードにおけるスマートコントラクトエンジンを通じて実行されます。各実行の入力と出力は、元帳に記録することもできます。

オントロジーアプリケーションフレームワーク

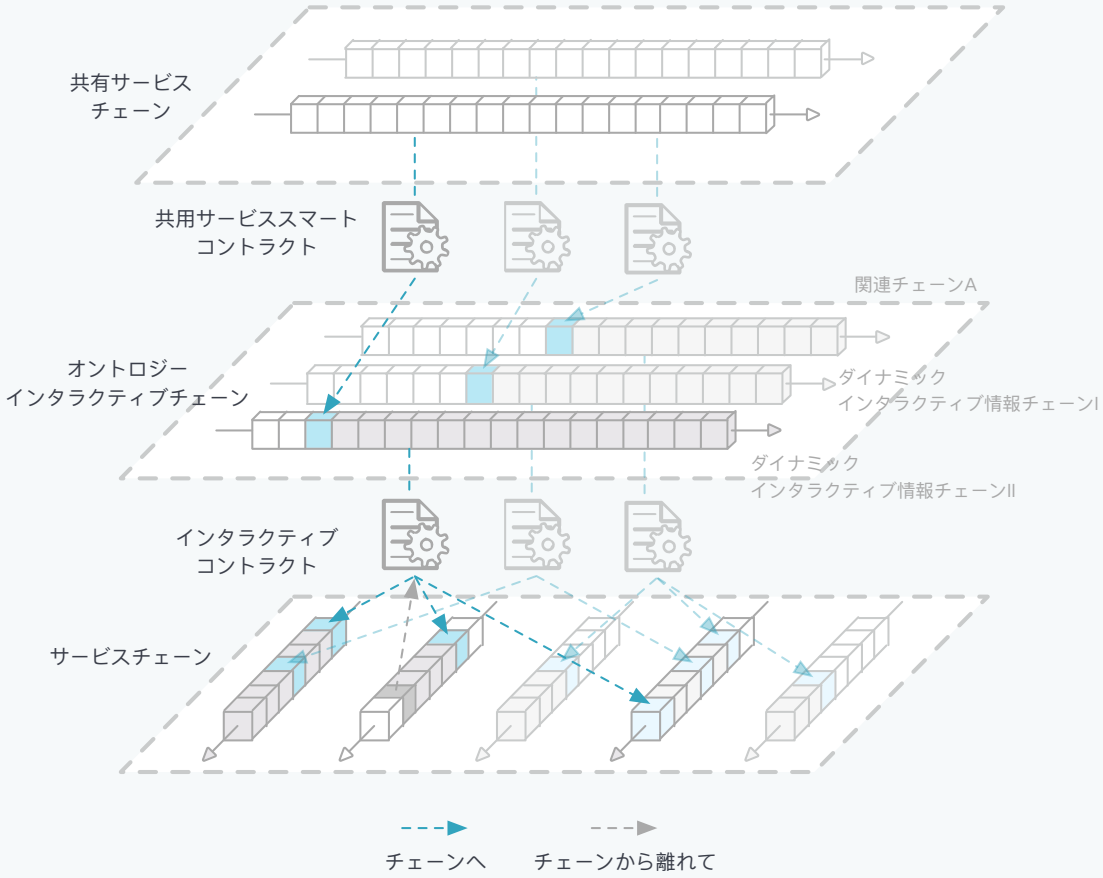
アプリケーションモジュール、プロトコル、SDK、およびAPIを指す一般的な用語です。サードパーティが作成したdAppへの高速、かつ低コストのアクセスを容易にします。

ハイブリットオラクル

ハイブリットオラクルはブロックチェーンに信頼可能な外部データを提供するサービスです。ユーザーはハイブリットオラクルを利用し、ブロックチェーンシステムの外部における出来事を予測し、これらをブロックチェーンに永久的に記録することができます。

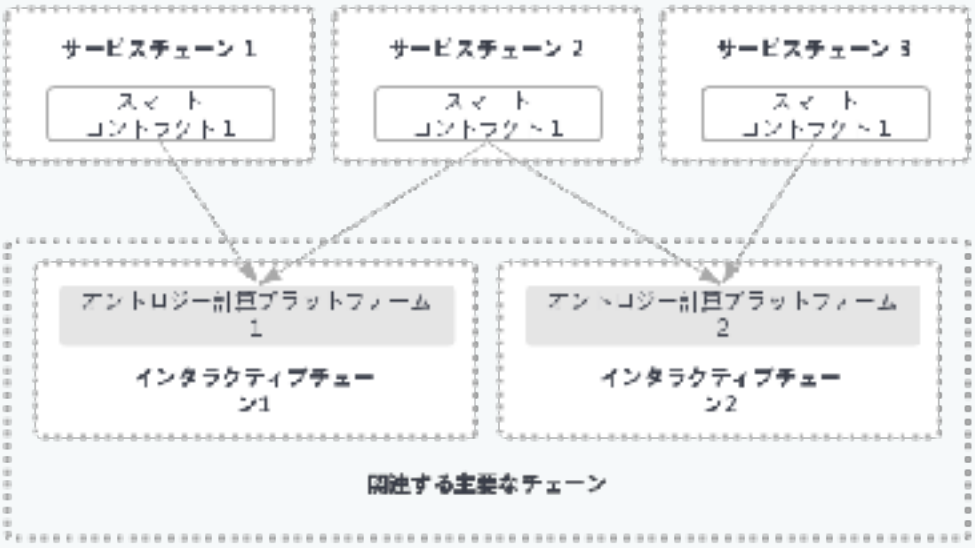
3. チェーングループストラクチャー

オントロジーの目的は、現実世界と分散型データシステムをつなぐ架け橋となることです。現実世界におけるビジネスの多様性、複雑性、特殊性のために、パフォーマンス、スケーラビリティ、適応性を考慮に入れなければなりません。そのため一つのパブリックチェーン、もしくは関係するチェーンネットワークだけですべてのシナリオに対応することが困難です。実際には様々なビジネスロジックは、異なるアクセス方法やガバナンスモデルとともに、異なるシナリオのニーズに応える複数のチェーンを必要とします。また多くのビジネスシナリオは、それぞれ独立しているのではなく、他のシナリオと多様な交流を行うことを必要とします。したがって、サービス間でんおプロセスにおいて協力をサポートするために、これらの異なるチェーン間に異なるプロトコルを提供する必要があります。

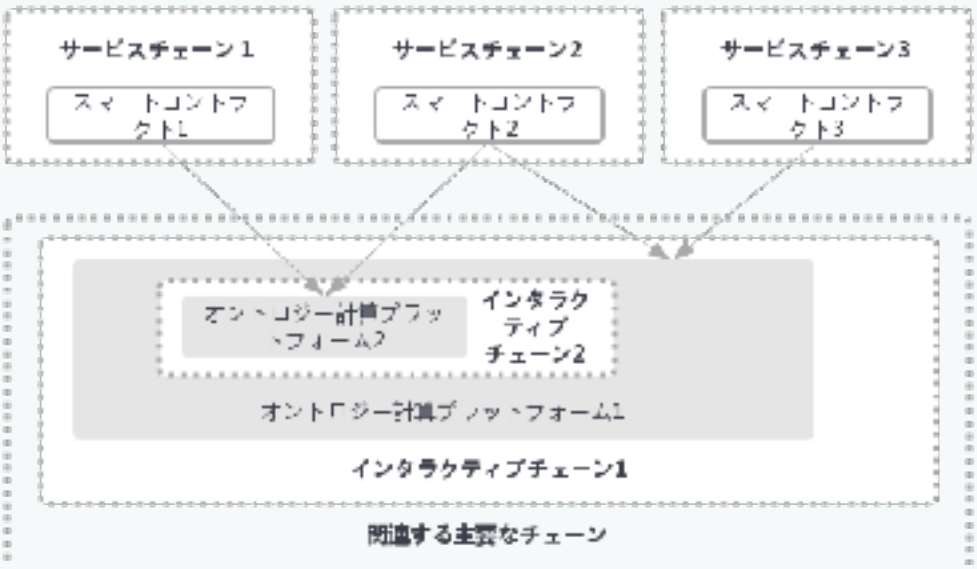


図表3.1：ハイパーカバレッジチェーングループ

オントロジーは要件とモデルに基づいて、マトリックスグリッドの形をとるハイパーカバレッジチェーングループを提案します。水平領域ではエンティティマッピング、データ交換のプロトコルサポート、そしてスマートコントラクトサービスのような基本的な共通サービスを提供するパブリックチェーンがあります。一つあるいは、それ以上のパブリックなブロックチェーンにおいて、各業界、地域、そしてビジネスシナリオにアクセス、コンプライアンス、ガバナンス、コンセンサス等の要件を満たす独自のサービスチェーンを設定することができます。それぞれはエンティティの認証、データ交換プロトコル、他のビジネスと共同で作業することができます。



図表3.2クロスチェーン情報交換に対して、オントロジー計算プラットフォームを使ったサービスチェーン



図表3.3： オントロジー計算プラットフォームを使用した専用インタラクティブチェーンの構築

パブリックチェーンの使用に加えて、特定ビジネスの業界に関連するチェーンとの協力の可能性もあります。多くの共同において、関係するサービスチェーンも異なる可能性があるため、特定のビジネス要件のために一つまたは複数のサービスチェーンまたはサービスポイントと協力する、専用の公的/アフィリエイトサービスチェーンがいくつかあります。したがって垂直的な分野では、スマートコントラクトサービス、ビジネスロジックサービスなどのための特別なクロスチェーン共同サポートを含む多くのビジネスコラボレーションチェーンが出現するでしょう。

このようなマトリックスグリッドアーキテクチャは、真に自立した次世代の多目的なネットワークを形成することができます。異なるビジネスシナリオは幅広い協力的な方法を通じて、適切なサービスモデルを適用するさまざまな方法を見つけることができます。

オントロジーにおける様々なプロトコルは静的ではなく、ユーザーはさまざまなビジネスシナリオ、業界の機能、規制要件、ガバナンス要件などに応じて、複数のプロトコルを選択することができます。そのためオントロジーでは、プロトコルは開発プロセスの一部であり続けますが、主な目的は異なるプロトコルや基準の中でユーザビリティを最大化させ、オントロジーはより良い互換性やセケーラビリティを作ります。

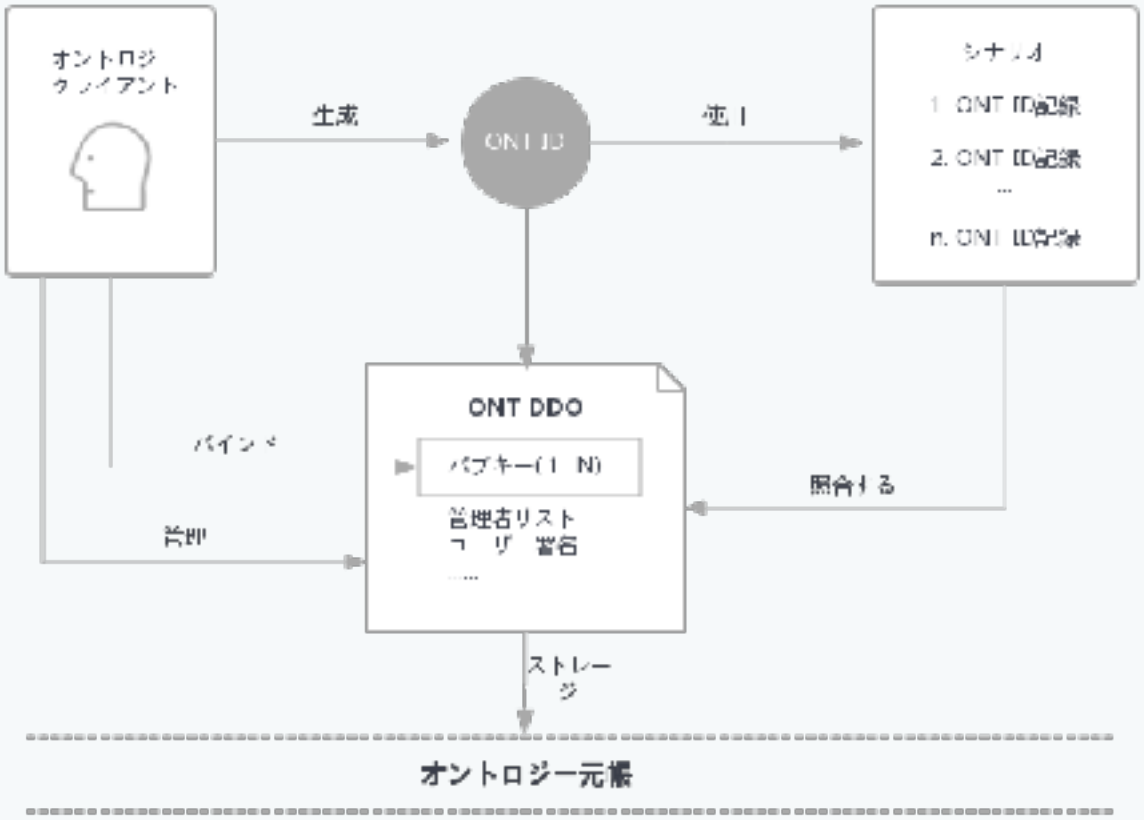
オントロジーは、一つまたは複数の設定可能なブロックチェーンの実装でさまざまなシナリオを満たすために、分散型元帳フレームワークを使用します。同時にオントロジーの分散型元帳フレームワークは、特定のビジネスシナリオのためのサービスチェーンをカスタマイズすることもできます。（例えばアクセス、ガバナンス、コンセンサス、ストレージなどのための複数のメカニズム）さらに、オントロジーはプロトコルを通じて、他の多くの現存するブロックチェーンや従来のITシステムと連携することもできます。

4. 分散型トラストフレームワーク

4.1. オントロジー認識プロトコル

「エンティティ」とは、現実世界における個人、合法的な組織（組織、会社、機関など）、モノ（携帯電話、自動車、IOT設備など）のことを示し、「アイデンティティ」とはエンティティのネットワークにおける身元のことを示します。オントロジーはオントロジーの識別子（ONT ID）を使い、エンティティのネットワークにおける身元を特定し、管理します。オントロジーにおいて、一つのエンティティは複数の個人の身元に対応することができます。

ONT IDは分散型認証プロトコルであり、個別エンティティの異なる認証サービスをマッピングし、結びつけます。これは分散型、自主管理、プライバシー保護、安全で使いやすいなどの特徴を持ちます。各ONT IDは、ONT ID記述オブジェクトに対応し、ONT IDの公開鍵などの属性情報を記録するために使用されます。ディスクリプションオブジェクトはオントロジーの中核における、分散型レイヤーにおいてストレージのための公開情報としての役割を果たします。プライバシー保護のため、デフォルトとしてのディスクリプションオブジェクトは、どのようなエンティティの識別情報も含んでいません。



図表4.1：ONT IDシステム

4.1.1. ONT ID生成

ONT IDはURI[6]の一種であり、各エンティティによって生成されます。生成アルゴリズムにより2つのONT IDの重複（$\approx \frac{1}{2^{160}}$）の可能性は極めて低く、またオントロジーに登録するときも、コンセンサスノードがそのIDが登録済みかどうか検証します。

4.1.2. 自主管理

オントロジーは、それぞれのエンティティが自身のアイデンティティを管理していることを確実にするために、デジタル署名技術を使用します。ONT IDはその所有を表すためにエンティティの公開鍵を登録します。ONT IDの使用及びその属性はオーナーによってデジタル署名されることが必要となっています。エンティティはONT IDの使用範囲を決めること、秘密鍵と結びつけること、そしてその属性を管理することができます。

4.1.3. マルチキーバインディング

オントロジーは世界中の多くのデジタル署名アルゴリズムをサポートしています。例えばRSA、ECDSA、SM2が挙げられます。ONT IDにバインドする秘密鍵は使用されているアルゴリズムを特定される必要があります。そして、ONT IDは複数の秘密鍵とバインドすることができ、これにより異なるアプリケーションシナリオにおいての多くのエンティティ要件に対応することができます。

4.1.4. 認定制御

ONT IDの所有者は他のONT IDを認定することができ、自分ONT IDの範囲を超えて管理する権限を行使することができます。例えばONT IDの属性情報は改訂されたり、オリジナルの秘密鍵が紛失した際、ONT IDに別の秘密鍵が結びつけられることがあります。ONT IDは「AND」「OR」「m of the n」などの複数のアクセスコントロールとそれぞれの属性に対して、きめ細かい管理許可をサポートします。

4.2. トラストモデル

トラストモデルは集中型トラストモデルと分散型トラストモデルを使用し、エンティティ間の信頼を築きます。異なる必要条件を満たすために、特定のシナリオによって異なるトラストモデルが使用されます。

4.2.1. 集中型トラストモデル

このモデルにおいては、一つあるいは複数のエンティティをトラストアンカーとします。このトラストアンカーを基本に、エンティティ間の信頼関係を築きます。トラストアンカーは信頼したエンティティを指定し、そして他のエンティティは同様に信頼しているその他のエンティティを指定します。ソースとしてトラストアンカーを使用し、このフォーマットは信頼関係を伝達するトラストツリーを作り上げることができます。ツリーにおける各ノードは、すべてトラストアンカーへの通路（トラストパス）を持っており、すなわちトラストチェーンを持っています。ツリーにおける他のノードと情報を交換する際に、トラストアンカーを認識したエンティティはそのトラストチェーンを検証することができます。

トラストシステムにおいて現在最も成熟し、広く使用されている集中型トラストモデルはPKI[7]です。トラストアンカーになるために、ユーザーはまずデジタル証明書を申請しなければなりません。申請が承認されてから、認証センターはアイデンティティ情報と公開鍵をデジタル証明書に書き込み、認証センターのデジタル署名を付与します。認証センターによって発行されたデジタル証明書は、アイデンティティと公開鍵とのバインド関係を証明します。すべての人がユーザーの証明書の信頼性を確認するために認証センターの公開鍵を使用することができます。そしてユーザー署名を検証し、アイデンティティを確認するために証明書の公開鍵を使用することができます。同時にデジタル証明書を持つユーザーは下位証明者として他のユーザーにデジタル証明書を発行することもでき、そうして発行されたそのデジタル証明書の効力は、認証センターにより保証されています。PKIモデルにおいて、参加者たちは無条件に認証センターを信頼し、そして信頼関係はエンティティ間におけるデジタル署名を通じ、認証センターから次々と伝達していきます。

集中型トラストモデルには、多くの利点があります。その厳密なトラスト伝達方式と明確なトラスト範囲は、多くのシナリオにおいて優れた機能であり、さまざまな問題を解決することができます。しかし集中型トラストモデルにも短所はあります。セントラルノードへの依存度は、公正さやセキュリティに対する高い要求があるケースだけでなく、複雑なトラスト関係においても不適切となるかもしれません。セントラルノードに頼るこの方法は、アプリケーションの柔軟性をひどく制限する可能性があります。

4.2.2. 分散型トラストモデル

信頼関係を築くために特定の集中型エンティティに依存することに加えて、エンティティは自発的に均等な強い信頼関係を築くこともできます。信頼の移転はエンティティ間の相互認証によって実現されます。エンティティは他の多くのエンティティによってより認証されると、その信頼性が高くなり、特に他のエンティティが高い信頼性を持っている場合には尚更です。

分散型トラストモデルは曖昧なトラストモデルであり、明確なトラストバウンダリーはありません。異なるシナリオにおいて、エンティティのトラストを評価するために異なるトラスト評価方法が使用されます。それは、そのモデルが現実世界においは幅広く使用されるという高い柔軟性のためです。

4.3. 検証可能な要求

検証可能な要求は、あるエンティティの特定の属性を証明するために使用されます。これらは保存され、データユニットとして移転され、エンティティによって証明されます。内容についてはどのようなデータでも可能なのですが、要求にはメタデータ、要求の内容、発行者の署名が含まれています。要求はLD署名仕様[9]に従った署名とともにJSON-LD[8]にフォーマットされます。

4.3.1. ライフサイクル

図表4.2：検証可能な要求ライフサイクル

検証可能な要求に関連するエンティティは3つに分類されます。それは発行者、保有者、検証者です。検証可能な要求のライフサイクルは、以下の5つのオペレーションを含んでいます。

- 発行：いかなるエンティティも他のエンティティの属性についての検証可能な要求を発行することができます。例えば、学校は検証可能な要求を発行することで、学生に成績表を提供することができます。検証可能な要求が使用された時、効力のある期間が設定されます。効力のある期間が過ぎた時は、その要求は失効します。
- ストレージ：検証可能な要求は、公開要求とプライバシー要求のどちらの方法でも発行されます。公開要求がオントロジーの分散型元帳に記録することができます。プライバシー要求が通常エンティティのクライアントに保存され、エンティティ自身によって管理されています。
- 提示：検証可能な宣言の所有者はそれが誰に公開され、どの情報が宣言の完全性に影響を与えずに表示されるかを選択できます。

- 検証：検証可能な要求の検証は、要求の発行者と対話する必要はなく、発行者の*ONT ID*を使用して、オントロジーの分散型元帳から公開鍵情報を取得する必要があります。その後、公開鍵を使用して要求のデジタル署名を検証することができます。
- キャンセル：検証可能な要求の発行者は、要求の有効期限前に撤回することができ、撤回された要求は有効とはなりません。

4.3.2. 匿名要求

l通常の状況下では要求が作成された時、要求の所有者は要求内容全体を検証者に公開します。しかし要求の所有者は、要求の具体的な内容を検証者に公開したくない場合もあります。この場合にはオントロジーは、ユーザーのプライバシーを保護するために、匿名の要求技術を使用します

匿名要求の技術は、要求の発行と提示の過程で、所有者の情報を隠す問題を解決します。匿名で要求プロトコルを発行する場合、一つのエンティティは要求について証明書二つを検証者二人から受け取ります。検証者二人が、彼らの持つ情報を共謀して漏らしても、彼らは同じエンティティから受け取った情報かどうかを検証することができません。匿名要求を行う際に、発行者はオリジナルの要求を検証者に提出する必要はなく、ゼロ知識証明を提供するだけでいいのです。検証者は発行者の公開鍵、証明書、証明書に含まれている属性値のアサーション（例えば、「年齢> 十八」かつ「上海住民」など）と併せて、検証アルゴリズムを実行することにより、要求の信憑性を検証することができます。

匿名要求は、通常一つのXMLやJSONファイルとなっており、公開情報と暗号化情報が含まれます。公開情報は匿名要求のすべての属性を含んでおり、それは属性の名前、属性の種類、属性の値です。属性は文字列、整数、日付、列挙型などのさまざまなデータタイプをサポートします。暗号情報は主に、所有者自身のマスターキーや発行者のデジタル署名がある公開情報が含まれています。

検証可能な匿名要求を提示するプロセスにおいて、所有者は第三者の検証者に対して発行者による匿名要求を持っていることを証明します。そして選択的に属性値の一部を公開したり、他の属性値を非表示にすることができます。また隠れた属性の中には、特定のロジックのアサーションを満たすことを証明することもあります。

匿名要求は、CL 署名スキーム[10]とΣプロトコル[11]を使用し、上記の特性を達成します。

4.3.2.1. CL署名スキーム

CL署名スキームは三つのアルゴリズムから構成されており、それはキー生成アルゴリズム、署名作成アルゴリズム、そして署名検証アルゴリズムです。このスキームは強力なRSA仮定の下で安全です。

キー生成 GEN:

1) ランダムに*2*つの安全素数*!*（p、q）を選択し、n *!=* pqを計算します。

2) ランダムにk *!+ 2* と二次余剰 *(R1*、*...*、*Rk, S*、*Z)*を選択します。

3) 秘密鍵*!sk=*（p、q）と公開鍵*p!k = [*n、*R1...Rk,*、*S*、*Z*）を出力します。

サイン生成 $SIGN_{sk}(\{m_i\})$:

*k!*を入力すると、*{m1, ..., mk}*の値となります。

1) オイラーのファイ関数φ *(n) =* φ *(p)*φ *(q) = (p −1)(q −1)*を計算します。

2）ランダムに十分な素数 *!e*と整数*v!*を選択します。

3）*e! module !*φ*(n)*の逆数を計算し、*!e−1*として示され、それは*! e·e−1≡1 (mod*φ*(n))*を満たします。

4) 以下、整数を計算します。

$$A \equiv (A^e)^{e^{-1}} \equiv \left(\frac{Z}{S^v \cdot \prod_i R_i^{m_i}} \right)^{e^{-1}} \pmod{n}$$

5) 署名 (A, e, v)を出力します。

署名検証 $VERIFY_{pk}(\{m_i\}, A, e, v)$:

!{m1, ..., mk} に*k*値と署名*(A*、*e*、*v)*を入力します。

1) *e*、*v*が与えられた範囲内かどうか、そして*e*は素数かどうかを確認します。

2) 署名が満たされているかどうかを確認します。

$$A^e \equiv \frac{Z}{S^v \cdot \prod_i R_i^{m_i}} \pmod{n}$$

4.3.2.2. Σプロトコル

Σプロトコルとは効率的なゼロ知識証明のプロトコルであり、そのようなプロトコルを使って、証明者Pは検証者Vに秘密内容を明らかにしない条件下で、秘密を知っていることを検証者に証明で

きます。よく知られているΣプロトコルがFiat-Shamirヒューリスティック[12]とSchnorr署名スキーム[13]です。

Assume 仮に証明者Pが方程式系 $y_1 = g^x,\ y_2 = h^x$ の解を知っていると、次のように表記されるシグマプロトコルを使用することができます。

$$SPK\{x : g^x = y_1,\ h^x = y_2\}$$

この証明は、次のインタラクティブプロトコルを使って証明することができます。

1) 証明者Pは乱数rを選択します。
2) 証明者Pは(gr, hr)を計算し、検証者Vに送信します。
3) 検証者Vは乱数cを作成してから証明者Pに送信します。
4) 証明者Pが$s = r - c \cdot x$を計算してから、sを検証者Vに送信します。
5) 検証者Vは以下の式が成り立つかどうかを確認します。

$$g^s \cdot y_1^c = g^r, \quad h^s \cdot y_2^c = h^r$$

4.3.2.3. 匿名要求の発行

受信者Rに匿名の要求を発行するには、発行者Iと受信者Rは、次のように2回のインタラクティブプロトコルを実行する必要があります。

1) 発行者Iが乱数n_1を作成してから、受信者Rに送信します。
2) 受信者Rが乱数v'を作成し、マスターキーm_0に対して$U = R_1^{m_0} S^{v'} \pmod n$を計算してから、発行者$I$に送信します。
3) 発行者Iはランダムな素数eと整数v''を作成します。
4) 発行者Iは属性$\{m_i\}$と(A, e, v'')に対してのCL署名を計算し、受信者Rに送信します。
5) 受信者Rは$v = v' + v''$を計算し、最終的な匿名要求(A, e, v)を得ます。

受信者は、このプロトコルを実行することによって、発行者からの公開属性値にCL署名を取得します。CL署名は、匿名認証情報の暗号化情報を構成しています。

4.3.2.4. 提示と検証

匿名要求の所有者は、いくつかの属性を検証者に選択的に開示し、他のものを非公開として保つことができます。 匿名の検証可能な請求スキームは、特定のアサーションの正当性を証明することも可能にします。

属性を非公開とし続けるために、所有者は公開されていない属性の知識を証明する必要があります。k個の属性{m1, ..., mk}の中に*l*属性があると仮定して、証明する方法は以下です。

1) 乱数r_Aを生成し、要求の中の*CL*署名をランダム化します。

$$A' = A \cdot S^{r_A}, v' = v - e \cdot r_A$$

2) *4.3.2.2.*項に記載されているプロトコルに従ってゼロ知識証明を構築します。

$$\pi = SPK\{r_A, e, v', \{m_i : m_i \in \mathbb{H}_l\} : VERIFY_{pk}(\{m_i\}, A', e, v') = TRUE\}$$

3) 証明πを出力します。

プレディケートの説明を証明するために、いくつかの私的属性の不等式述語を証明する方法を示します。不等号より大きい述語は、属性値mと下限bを指定します。 ここでは実際のmを開示することなく、m≥bであることを示すプロトコルを与えます。

1) $\Delta = m - b$を計算し、それを 4 つの整数の二乗和

$$\Delta = u_1^2 + u_2^2 + u_3^2 + u_4^2$$

2) *4*つの乱数T_{u_i}を作成し、これらの*4*つの乱数を隠し、以下の証明を構築します。

$$\pi_1 = SPK\{(u_i, r_i) : T_i = S^{u_i} Z^{r_i}\}$$

3) 乱数T_Δを作成し、この差Δ*!*を隠し、以下の証明を構築します。

$$\pi_2 = SPK\{(u_1, u_2, u_3, u_4, a) : T_\Delta = \prod_i T_i^{u_i} Z^\alpha\}$$

4) 属性値が*b*以上であることを証明します

$$\pi_3 = SPK\{(m, r_\Delta : S^b T_\Delta = S^m Z^{r\Delta}\}$$

5) 証明書(π_1, π_2, π_3)の出力をします。

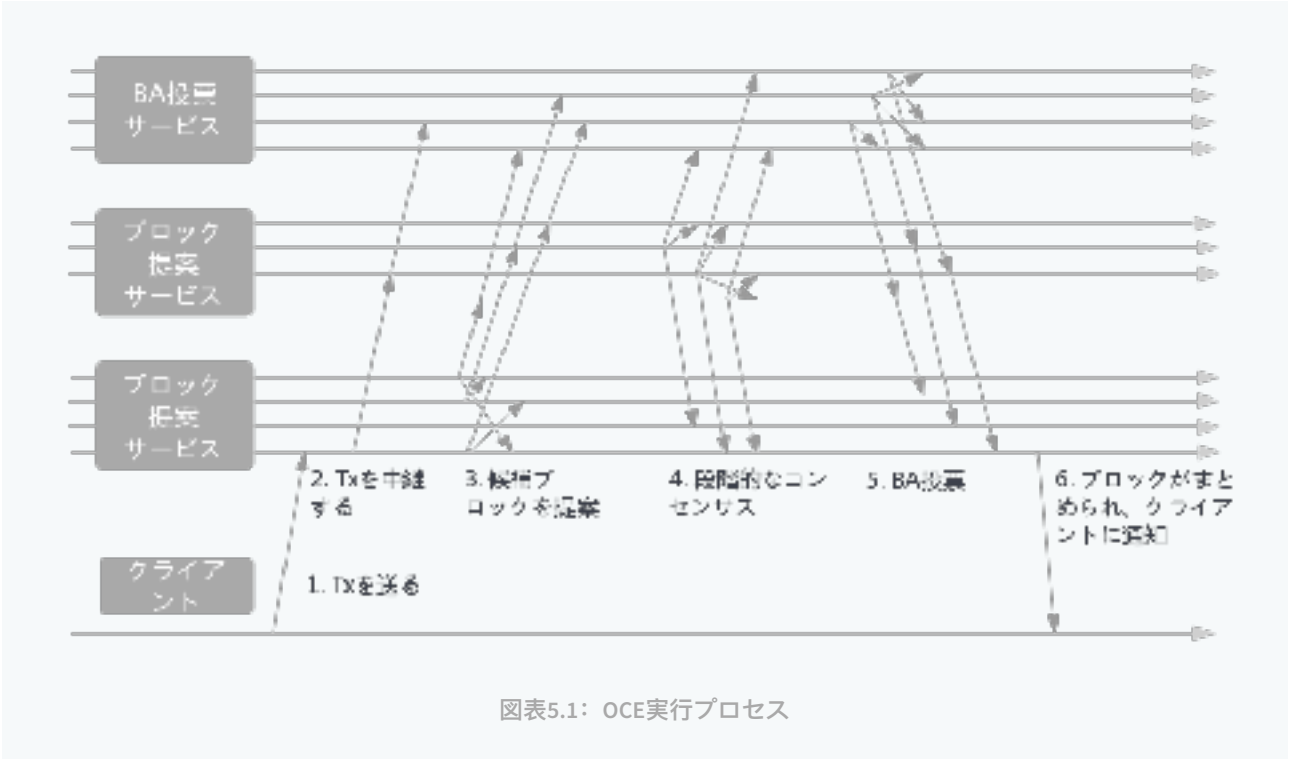
πと(π_1, π_2, π_3)を組み合わせると、完全な証明が得られます。 検証者は、セクション4.3.2.2.で説明した検証方法を使用して、各サブプルーフをチェックすることができます。述部は、すべてのサブプルーフが有効である場合にのみ正しくなります。

5. 分散型元帳

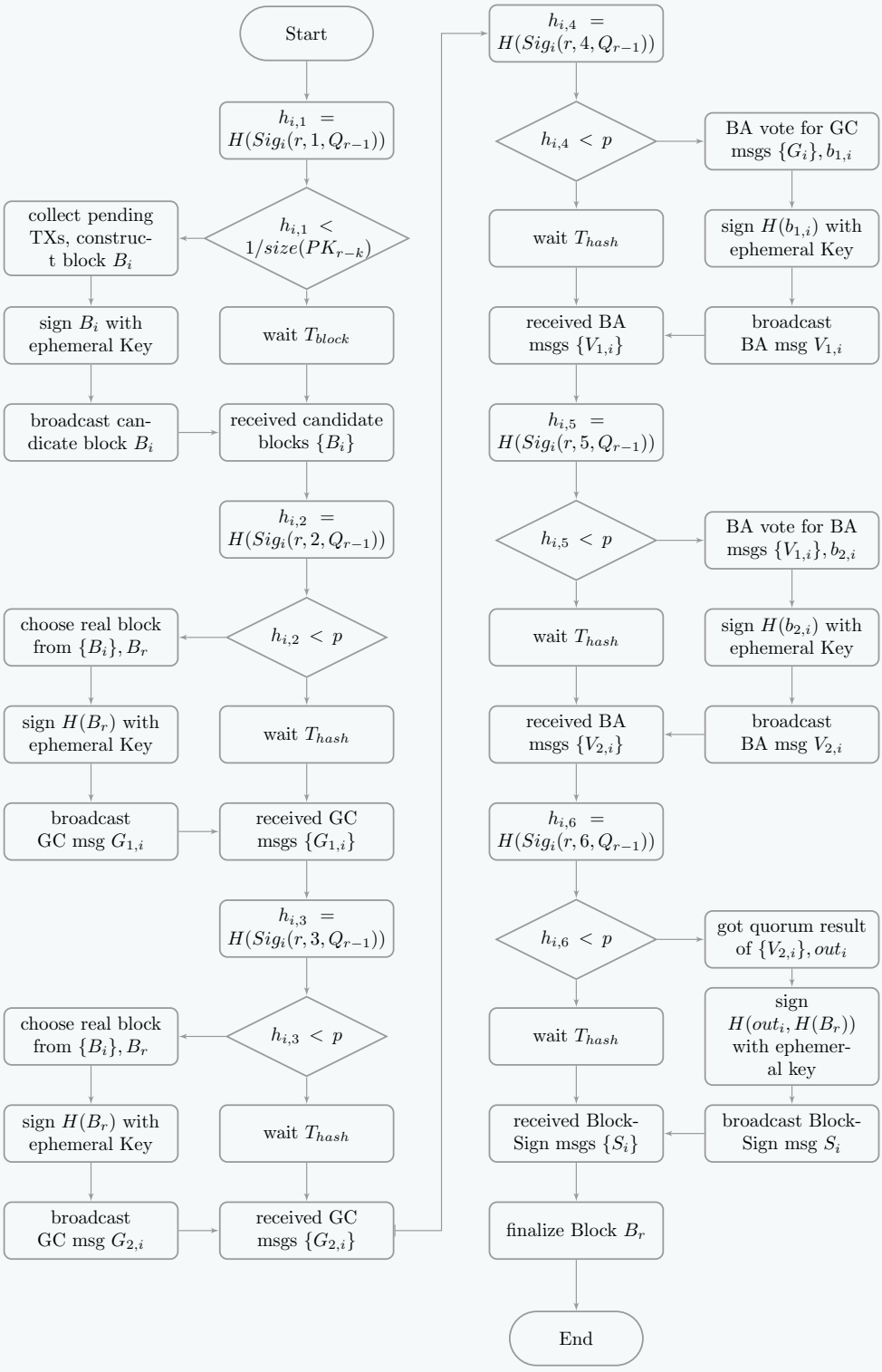
5.1. オントロジー元帳

5.1.1. コンセンサスメカニズム

オントロジー元帳は、次世代のOndorand Consensus Engine（OCE）をサポートしています。OCEは効率的で、dBFTコンセンサスプルトコルと検証可能なランダム関数(VRF)コンセンサスエンジンのコンセンサスベースのバージョンであり、事実上無制限のスケーラビリティを実現し、ネットワークフォークはほとんどありません。dBFTはNEOパブリックチェーンや他の関連するブロックチェーンプロジェクトにおいて、長期間にわたって優れた安定性と信頼性を示しています。OCEのブロック生成速度はインターネットの速度のみに制限されており、通常は10秒以内に確認が完了します。OCE は検証可能なランダム関数を使用して、コンセンサス検証に参加する対象者を選択し、ビザンチンフォールトトレランスアルゴリズム[14][15][16]を使用することによって、コンセンサスに達します。同時に、検証者グループのグループ署名によってオントロジーのシードが生成され、次の検証者グループに直接送られます。OCEは、接続可能な検証者とオンラインプロトコルの修復とアップグレードをサポートします。



図表5.1：OCE実行プロセス



図表5.2：OCEフローチャート

OCEのワークフローとそのプロセスステップを図表5.1と5.2に示します。

1) クライアントは*P2P*ネットワークを通じて、データ交換要求を広め、また要求当事者の署名を添付します。

2) コンセンサスプロセスに参加するノードは、ネットワーク内のすべてトランザクションを監視し、ローカルキャッシュにそれらを保存することができます。

3) 新しいブロックの提案

   a) ノードは、現在のブロックの高さと前のブロックの*Q*値をベースに署名を作成し、ハッシュ値を計算することができます。ハッシュ値は、ノードが現在のブロックの高さの提案者となり得るのかを決定するために使用されます。

   b) ノードが、現在のブロックの高さを提案する可能性があると評価した場合

      i) 現在および収集された非コンセンサス・トランザクション要求をパッケージ化し、新しいブロックを構築します。

      ii) 新しいブロックのデータおよびハッシュ値に基づいて署名を作成し、ブロックおよび署名を*P2P*ネットワーク全体に送信します。

4) ブロックプロポーザルのネットワークトラフィックを単独で監視し、受信したブロックプロポーザルをローカルメモリに一時的に格納すると、すべてのノードが*T*ブロックを待機します。

5) 新しいブロックレベルのコンセンサス

   a) *T*ブロックの後、ネットワーク内の各ノードをブロックすると、現在のブロック提案の検証ノードであるかどうかが評価されます。

   b) ノードが、現在のブロック提案の検証ノードとして評価された場合

      i) すべてのブロック提案の正当性と完全性を検証します。

      ii) すべてのブロック提案の*Q*値署名の正当性を検証します。

      iii) ブロック提案者のアイデンティティの正当性を検証します。

      iv) *Q*値署名に基づいてハッシュ値を計算し、現在のブロックによる真の選択されたブロックとして最も小さいハッシュ値とともにブロック提案を選択します。

      v) 選択されたブロック提案のハッシュ値の署名を作成し、ブロックハッシュと署名を*P2P*ネットワークに広めます。

   c) *T*ハッシュ後、ノードはアルゴリズムステップを更新し、プロセスのこのステップにおいてブロック提案の検証へ参加すべきかどうかを評価します。

   d) ノードが現在のステップにおいてブロック提案の検証ノードとしてそれを評価する場合

i) 上記と同じようにブロック提案を検証します。.

ii) 選択されたブロック提案のハッシュ値署名を生成し、ブロックハッシュと署名を*P2P*ネットワークに広めます。

6) *P2P*ネットワーク内のすべてのノードは彼ら自身によってコンセンサスメッセージを監視し、ブロック検証ノードによって広められます。

a) コンセンサスメッセージの正確性と完全性を検証します。

b) コンセンサスメッセージを広めるノードのアイデンティティの有効性を検証します。

c) 一時的にローカルメモリにその検証を保管します。

7) 新しいブロックのビザンチン投票

a) *T*ハッシュの後、ノードはアルゴリズムステップを更新し、これらのいずれかが新しいブロックの投票に参加すべきかどうかを評価します。

b) ノードが投票に参加すべきであると判断した場合、以下のステップが実施されます。

i) 異なるブロック提案のコンセンサス検証の投票を受け入れます。

ii) もし定数に達すれば、コンセンサスの結果をベースに採決します。

iii) 投票結果に署名し、選択されたブロックのハッシュを計算し、*P2P*ネットワークへ署名だけでなくその結果も広めます

c) ネットワーク内のすべてのノードはブロックチエーンの構成に従い、ブロックの投票に対して複数回の投票を行います。各回の投票は*T*ハッシュのインターバルを待ち、独立して動作します。

8) ネットワーク内のすべてのノードは*P2P*ネットワーク内の投票メッセージの正当性を継続的に監視し、検証します。

9) 新しいブロックの最終署名の生成

a) ビザンチン投票の最終回が完了した後、ノードはアルゴリズムステップを更新し、これらのいずれかが新しいブロックの最終署名の作成に協力すべきかどうかを評価します。

b) *I*ノードが最終署名の作成に協力すべきであると判断した場合、以下のステップが実施されます。

i) 投票数に基づいて、ブロック提案が定位数に支えられたコンセンサスに達したかどうかを判断します。

ii) 定数に達していないと判断された場合、この新しいブロックコンセンサス結果を失敗（*false*）と設定します。

iii) 前回の評価をベースにした投票メッセージをパッケージ化し、署名します。

iv) 最終ブロックハッシュと投票結果の署名を*P2P*ネットワークに広めます。

10) すべてのノードは*P2P*ネットワーク内の新しいブロックの最終署名のメッセージを継続的に監視し、署名メッセージの有効性を検証します。

a) *T*ハッシュの後、ノードは受け取ったブロック署名のメッセージに従い、現在のブロックの高さのブロックハッシュを計算します。

b) ノードは計算されたブロックハッシュを前に受け取ったブロック提案メッセージと比較し、現在のブロックの高さのための最終ブロックを取得します。

一方、ブロックQ値がコンセンサスの最終ブロックをベースに計算され、次のブロックの高さについてのブロックのコンセンサスプロセスを初期化します。

オントロジー元帳は、モジュール式アーキテクチャを採用しており、プラグ可能なコンセンサスアルゴリズムをサポートし、異なるシナリオに応じて例えばPoS、DPoS、Raftなどの他のコンセンサスアルゴリズムに簡単かつ迅速に切り替えることができます。

5.1.2. プロシージャプロトコル

分散型プロシージャプロトコル（分散型トランザクション）は分散型元帳技術、エンティティクロスチェーン、クロスシステムプライバシー、そして特定のクロスチェーンプロトコルを通じて実現されます。プロシージャもしくはトランザクションステップは全体のトランザクションの一貫性を確保するために、異なるブロックチェーンもしくはシステムにおいて実行されていますが、エンティティの身元プライバシーは保護されています。

5.1.3. 認証設計

データだけでなくデータの動作も分散型元帳へ署名されます。これは毎回のデータ要求の度にデータマッチング、データ検索、データ使用が元帳へ記録されることを意味し、データセキュリティと信頼性を確保しながら、データの全体プロセスの記録を行っています。

5.2. スマートコントラクト

オントロジー元帳のNeoVM仮想マシンは、Goで書かれており、スマートコントラクトを実行するために使用され、インテリジェントな制御ロジックをオントロジーのアプリケーションレイヤフレームワークに実装することができます。NeoVM仮想マシンのチューリング完了により、任意のロジックと高い確信性を実現でき、それにより同じ入力は常に同じ出力結果となることを常に保

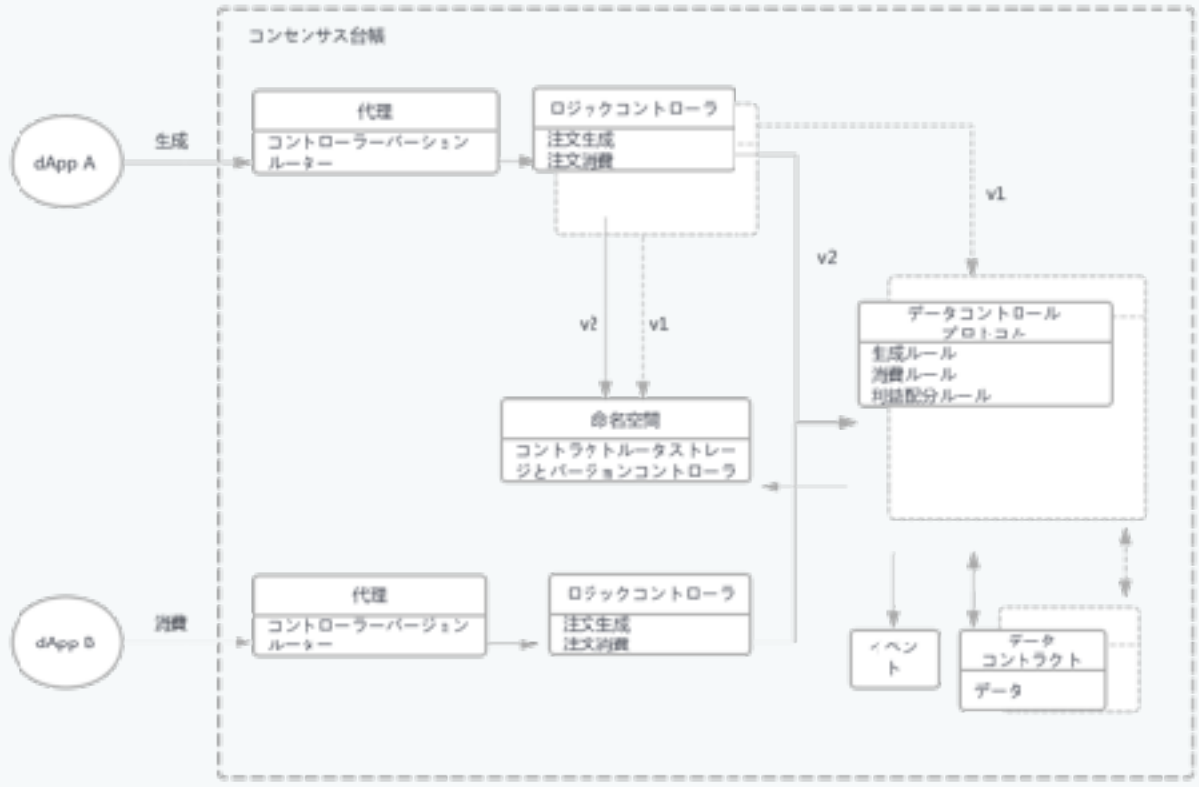
証し、コンテナやドッカーの不確実性を避けることができます。それは確実性の要求が高いシナリオにおいて適しています。さらに、NeoVMは、理論的には無制限の拡張機能を意味するダイナミックシャーディングを実現する「決定論的コールツリー（deterministic call tree）」テクノロジにーよって高いスケーラビリティを実現します。

さらに、独立して開発されたコンパイラを使用すると、Java BytecodeやC# MSILを含む他のコンパイラもブロックチェーン仮想マシンと連携することができます。これにより、スマートコントラクト開発者の間での参加が促進され、Java、C#、Goなどのプログラミング言語を使用して、新しい言語を習得することなくスマートコントラクトを作成できます。プラットフォームの容易な操作と保守は、他のパートナー（特に情報提供者）と簡単に共有でき、スマートコントラクトの編集機能やアクセシビリティを提供します。

NeoVM仮想マシンは、上位言語解析と変換を組み合わせます。仮想マシンの外部インターフェースはAPIでカスタマイズされ、アカウントや外部データと柔軟に対応することができます。これにより、スマートコントラクトが実行される際には、ネイティブコード実行の高いパフォーマンスを実現させることができます。同時に、異なるブロックチェーンをサポートする仮想マシンメカニズムも実装されています。

5.3. 共有データコントラクトモデル

分散型元帳からのアプリケーション要求には、主に二つの機能が含まれます。まず、データ構造の定義付けとストレージです。次に、ビジネスロジック処理と外部システムとの相互作用です。これら2つの部分を分離したモデルを設計し、システムのスケーラビリティと柔軟性を向上させました。1つはデータストレージ（データコントラクト）用、もう一つはビジネスロジック（コントローラコントラクト）用です。 これを「共有データコントラクトモデル」と呼びます。



図表 5.3: 共有データコントラクトモデル

このモデルには、以下の設計要素が含まれます。

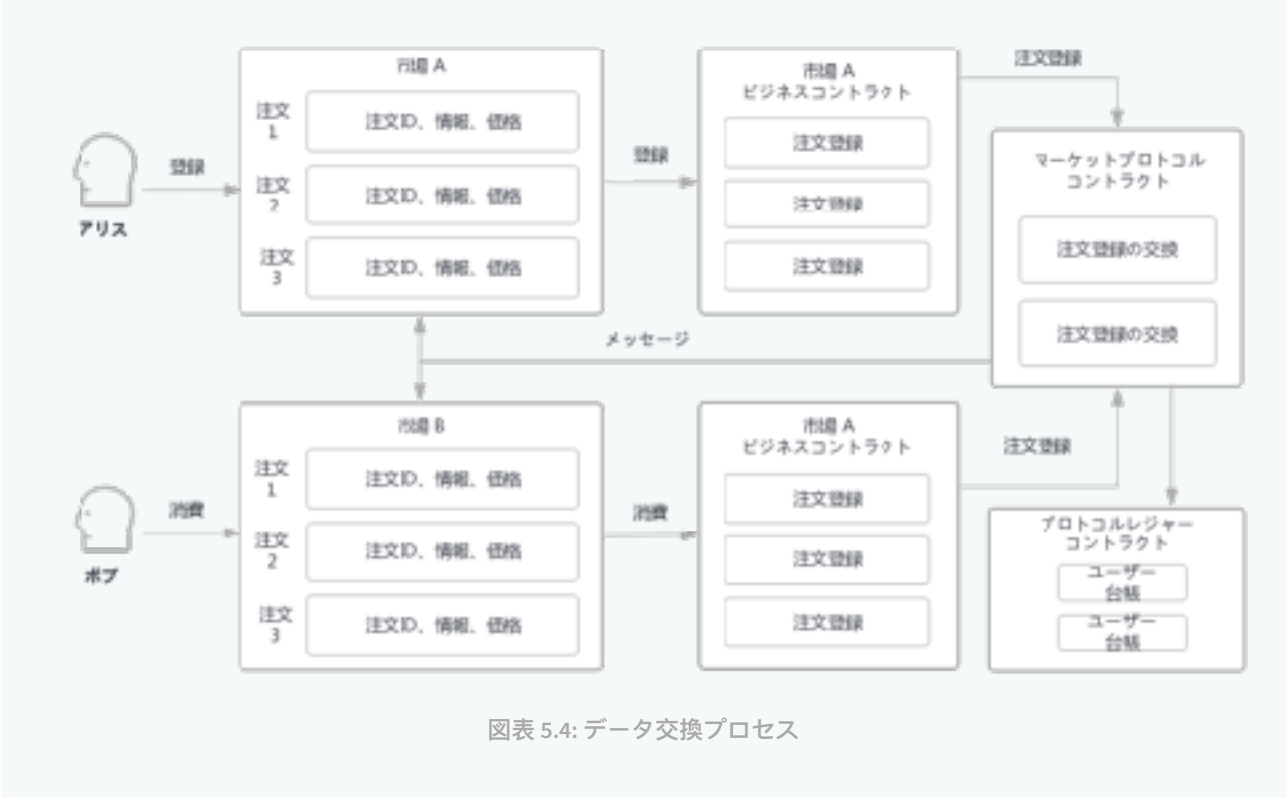
- エージェントコントローラー: *dApp*に対して、外部*dApp* の確定コントラクトエントリーを提供します。コントラクトがアップグレードしても、外部呼び出すに一致しないことが生じないことを保証します。
- ロジックコントローラー: ビジネスロジックコントロールのためのもの。
- 命名空間: 異なる*dApp*の異なるバージョンに、データコントラクトアドレスのマッピングをサポートします。データ構造のアップグレードがビジネスロジックに影響を与えずに、異なるバージョン間のデータバックトラッキングをサポートします。
- データコントラクト: 基本的なデータ構造とストレージインターフェイスを提供します。*DAO* と類似して*GET / SET* メソッドを提供します。
- データコントロールプロトコル: 各ビジネス関連者によって開発された共通のビジネスルールをデジタル化して、プロトコルコントローラーになります。このプロトコルコントローラーには以下のセクションがあります。
  - ロジックコントローラのアクセス権限コントロールに接続

- 外部システムとのインターフェースとの相互作用
- 共通サービスロジック
- ビジネスコア元帳のコントロールロジック
- イベント通知メカニズム

- イベント：イベントとは、スマートコントラクトを実行するプロセスにおいて、コンテンツを排除するメカニズムを参照します。各当事者は元帳を同期させるプロセスで、分散型元帳ブロックのローカルで再生して、現在のコントラクトの取引の内容を得ることができます。イベントは実際に分散型元帳の低コストのストレージメカニズムであり、スマート契約に対応するノードは、コントラクトを実行することによってイベント情報を受信することができます。これにより、チエーンが実際のトランザクション内容を保存しないため、各ノードの格納圧力を大幅に低減することができます。

このコントラクトモデルにより、ビジネスとは関係なしに、dAppは汎用プロトコルを使用してデータをお互いに共有することができます。また、境界を越えた協力がより簡単になります

注文の登録と交換のプロセスは、図表5.4に示されています。



図表 5.4: データ交換プロセス

登録:

1) アリスは、マーケット *A*へ注文登録要求を送信します。

2) マーケット*A* がアリスの注文をチェックした後、分類に従って分類されたプロトコルコントラクトに注文を登録します。

3) マーケットプロトコルコントラクトは、マーケット*A*の権限とそれによって提出された注文の内容を検査し、ルールを守ると注文書を契約書のコアレジャーに登録します。

4) 登録が成功した後、プロトコルコントラクトは注文登録情報を送信し、結果を返送します。

5) マーケット*A*ビジネスコントラクトは最終結果をアリスに返送します。

取引状況:

1) マーケット*B* は同期ブロックを通じてすべての取引を得られます。

2) マーケット*B* は、ブロック内の取引を実行するによってマーケット*A* がブロードキャストされた注文登録メッセージが得られます。

3) マーケット*B*は、注文登録メッセージの注文情報を通じてマーケット*B*に提示します。.

4) ボブはマーケット*B* との取引を開始し、対象が上のアリスで登録された注文です。

5) マーケット*B* は、ボブの要求をチェックして、分類に従ってプロトコルコントラクトへの取引を開始します。

6) プロトコルコントラクトはマーケット*B*の権限と取引要求情報を検査し、ルールを守った*B*の取引要求を処理し、最終的な決済をします。

7) プロトコルは、注文取引完了情報を送信し、結果を市場*B*に返送します。

8) マーケット*B*は最終結果をボブに返送します。

9) マーケット*A*は、プロトコルコントラクトの取引完了の送信情報を監視し、取引完了メッセージをアリスに送信します。

5.4. マークルツリーストレージモデル

ビットコイン、イーサリアムなどのデジタル通貨プラットフォームに対して、クライアントは通常自分のアカウントの情報のみ焦点を合わせます。元帳を完全に同期させることなく口座の状態を検証するために、マークルプルーフ[17]を構築することによってストレージスペースを大幅に節約し、ネットワーク伝送を削減できるSPV（簡易決済検証）技術が提案されています。オントロジーでは、クライアントは他のクライアントの識別情報も検証する必要があります。したがって、元帳はマークル証明の構築をサポートする必要があります。

5.4.1. マークルハッシュツリー

バイナリーマークルハッシュツリー[18][19] は、効率的な監査証明を構築するために使用されます。n個の入力$D[n] = (d_0, d_1, \ldots, d_{n-1})$の順序付きリストが与えられると、マークルツリーハッシュ（MTH）は以下のように定義されます。

$$\begin{aligned} MTH() &= sha() \\ MTH(\{d_0\}) &= sha(0\text{x}00 \| d_0) \\ MTH(D[n]) &= sha(0\text{x}01 \| MTH(D[0:k]) \| MTH(D[k:n])), \quad k < n \leq 2k \end{aligned}$$

Kはnより小さい最大二つのパワーです。$D[a:b]$はリストDのa番目からb-1までの要素からなるサブリストを表し、ll は前後二つのビット列を連結することを表します。

5.4.2. マークル審査方針

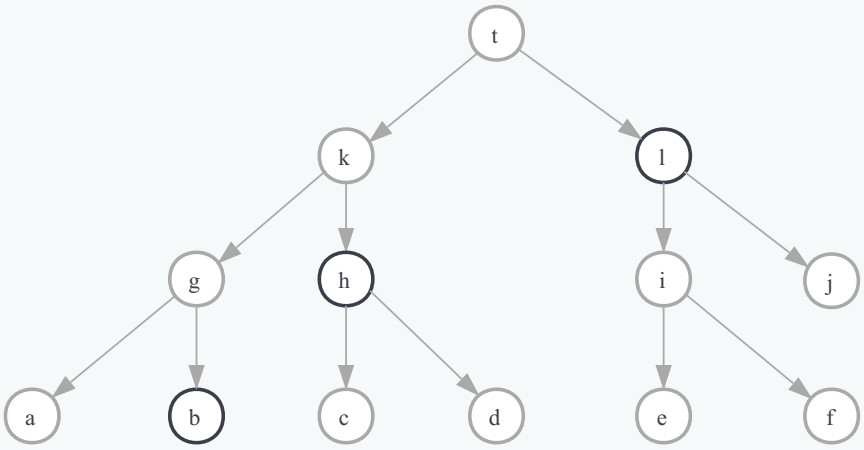
マークルハッシュツリーのリーフのためのマークル監査パスは、そのツリーのマークルツリーハッシュを計算するために必要な、マークルツリー内の追加ノードの最短リストです。ツリー内の各ノードはリーフノードであるか、またはリーフノードの直下の二つのノードから（すなわち、リーフに向かって）計算されます。ツリーを下に（ルートに向かって）降ろす度に、監査パスからのノードがこれまでに計算されたノードと結合されます。言い換えると、監査パスはリーフからツリーのルートに至るノードを計算するために必要な欠落ノードのリストで構成されます。監査パスから計算されたルートが真のルートと一致する場合、監査パスはリーフがツリー内に存在することを証明します。

ツリーへのn個の入力の順序付けられたリスト$D[n] = (d_0, d_1, \ldots, d_{n-1})$、（m + 1）個のノードに対するマークル監査パス$PATH(m, D[n])$入力$d(m) : 0 \leq m < n$は、以下のように定義されます。

$$\begin{aligned} PATH(d, \{d_0\}) &= \{\} \\ PATH(m, D[n]) &= \begin{cases} PATH(m, D[0:k]) + MTH(D[k:n]) & m < k \\ PATH(m-k, D[k:n]) + MTH(D[0:k]) & m \geq k \end{cases} \end{aligned}$$

+は前後の二つのリストを接続することです。

図 5.5 は、マークル審査通路の一例です。



図表5.5: aのノードの審査通路は[b、h、l]

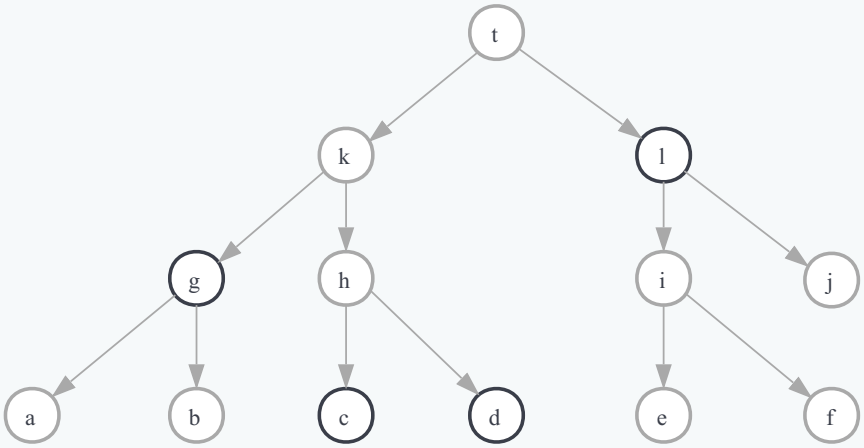
5.4.3. マークル一貫性証明

マークル一貫性証明は、ツリーの追加専用プロパティを証明します。マークルツリーハッシュ$MTH(D[n])$とそのツリーにあるm個のリーフノードに宣言されたハッシュ$MTH(D[0:m])$のマークル一貫性証明は、マークルツリー内のノードのリストです。最初のm個の入力D [0：m]が両方のツリーにおいて、等しいことを検証するために必要とされます。以下のアルゴリズムは、（唯一の）最小一貫性証明を構築することができます。

ツリーへのn個の入力の順序付けられたリストが与えられると、$D[n] = (d_0, d_1, \ldots, d_{n-1})$、以前のマークルツリーハッシュMTHのマークル一貫性証明$PROOF(m, D[n])\ (D[0:m])$は次のように定義されます。

$$
\begin{aligned}
PROOF(m, D[n]) &= SUBPROOF(m, D[n], true) \\
SUBPROOF(m, D[m], true) &= \{\} \\
SUBPROOF(m, D[m], false) &= \{MTH(D[m])\} \\
SUBPROOF(m, D[n], b) &= \begin{cases} SUBPROOF(m, D[0:k], b) + MTH(D[k:n]) & m \le k \\ SUBPROOF(m-k, D[k:n], false) + MTH(D[0:k]) & m > k \end{cases}
\end{aligned}
$$

図表5.6 は、マークル一貫性証明の例です。



図表5.6: PROOF(3,D[7]) は、[c,d,g,l]となります

5.4.4. マークルパトリシアツリー

オントロジーのいくつかのシナリオでは、エンティティの身元を証明するなど、特定のエンティティの取引の最終状態を、迅速に証明する必要があります。マークル証明を使用するには、各履歴トランザクションを一つずつ証明する必要があり、マークルパトリシアツリー（MPT）[20]を使用することで、効率を大幅に向上させることができます。

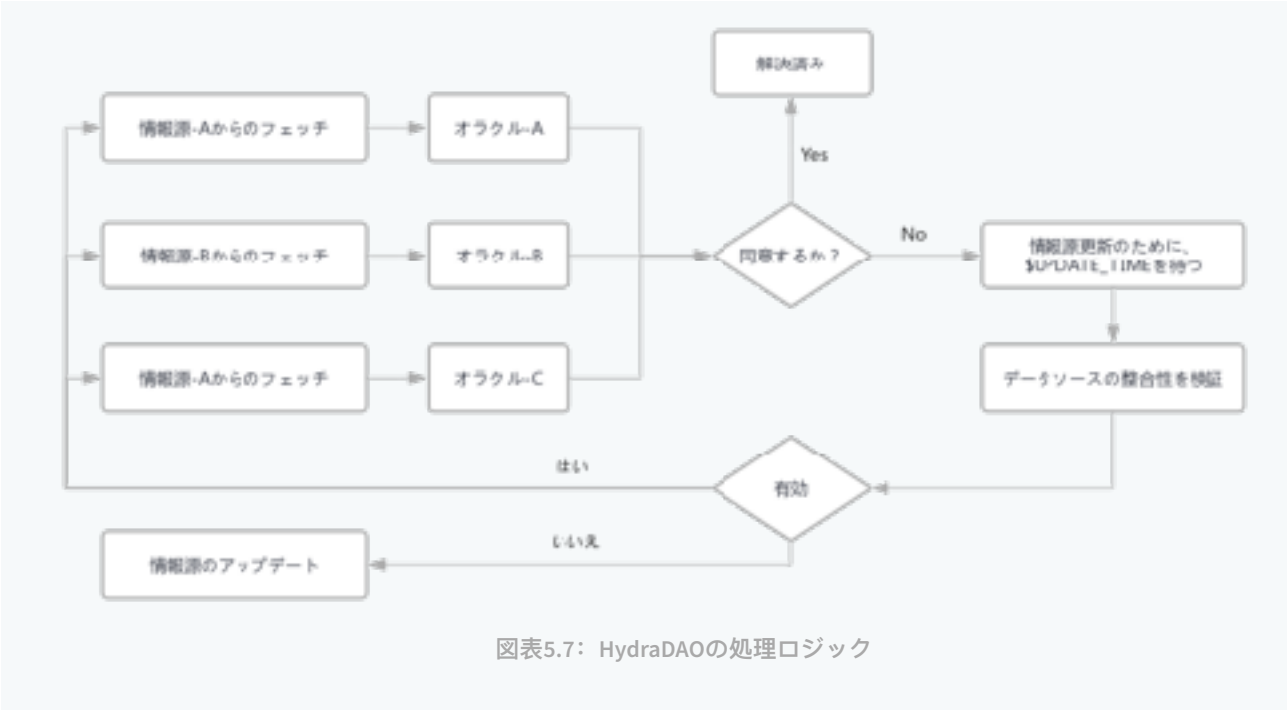
MPTはパトリシアツリー[21]とマークルツリーの組み合わせです。これには、キーと値のマッピング関係が含まれ、暗号化に基づいた不正侵入防止データ構造が提供され、決定性、高効率性、およびセキュリティがあります。

- 決定性：データを検索するとき、同じキーバリューが同じ結果を確実に見つけ、同じルートハッシュを持つことになります。
- 効率性：新しいルートを素早く計算できます。データが変更されると、挿入、検索、更新、および削除の時間の複雑さは$O(\log_2 n)$となります。
- 安全性：誰かが*DOS*攻撃で他人を悪意を持って攻撃し、ツリーの深さを制御しようとするとき、限定されたツリーの深さで、攻撃を阻止することができます。

5.5. HYRADAO

HydraDAOは、スマートコントラクト、クロスチェーン、クロスデータソースのコラボレーションを統合したデータ予測およびインタラクションモジュールです。これには、オントロジーのDAO（分散自律組織）とクロスチェーンデータ相互作用（ビッグ・データ/ AI）機能が含まれています。オントロジーのガバナンスメカニズムは、民主的およびAI自動化の提案、投票、および検証をサポートします。独自のDAOアドレスとポーリングトークンがプロセス中に作成され、DAOが自動的に資金と計算結果をオントロジーに追加できるようにします。ポーリングが完了すると、DAOは改ざん防止スマートコントラクトに従って自律的に実行します。このメカニズムにより、オントロジーのデータ交換とガバナンスは柔軟に機能し、大規模な自動ネットワーク操作のための技術をサポートできます。

オントロジのオラクルスマート・コントラクト実装のロジックは次のとおりです。



図表5.7：HydraDAOの処理ロジック

1) トランザクションが最初または最後に実行されると、オラクルスマートコントラクトの使用を選択できます。オラクルスマートコントラクトとそのパラメータは、トランザクションが作成され、トランザクションの一部としてブロックチェーンに格納されるときに設定する必要があります。トランザクションが終了すると、コントラクトのロジックが自動的に実行され、*DAO*コントラクトアドレスに保管された結果とトークンが有効なデータ情報を提供した参加者に支払われます。オントロジーの分散型データ交換プロトコル（*DDEP*）を介して、大きなデータコンピューティングパワーへのリアルタイム*API*アクセスを提供し、トランザク

ション結果の分析と予測におけるデータ市場の参加がその適合性を検証します。信頼可能なデータサービスの一部として、トランザクション参加者は信頼できるデータソースをバインドするか、オントロジーとつなぐサービスを利用して、トランザクションに参加する前に正確なトランザクションステータスデータを検証できます。

2) ブロックチェーンのステータスのみに依存するトランザクションの場合、オラクルスマートコントラクトは、認可時にブロックチェーンインタフェースからブロックチェーンデータを直接問い合せます。

3) 現実世界に関連するほとんどのトランザクションでは、オラクルスマートコントラクトは、実際の取引結果を得るためにデータが必要です。オラクルスマートコントラクトが現実世界のデータにアクセス/読取りをする際、各ヒドラオラクルはランダムに複数のトラストソースの1つを選択するか、データアンカーのしきい値を指定します。トランザクションの終了時に、グローバル状態が確認され、指定された信頼できるデータソースを[2]介して対話形式で更新されます。

5.5.1. 組み込みDAOデータ予測

分散型システムの魅力は、コストの削減と効率の向上だけでなく、分散したグループによる意思決定が利点としてあげられます。オントロジーで契約を行う場合、HydraDAOを接続して「公開テスト」を有効にするオプションがあります。例えば、ある外貨両替所から一定量の外貨を購入したいものの、保留中の注文が正しいかどうかがわからない場合、彼はDAOプロジェクト提案を開始し、ある程度のマージンを担保として預けることができます。十分な量の承認が得られると、その提案は予測期間に入ります。提案を受け入れる参加者は、ファクトステータスとその署名を添付することができます。参加者はいつでも注文を修正することができますが、これには追加マージンが必要です。予測期間が終了すると、DAOコントラクトは自動的にコントラクトのマージンをロックし、提案のルールに従って最適な提案を除外します。最も正確な予測をした参加者には報酬が与えられます。ユーザーが開始したトランザクションも、最良の出力に従って自動的に実行されます。

提案はDAOに提出されると、新しい提案は全体のネットワークのトークン保有者により評価し決められます。この間、参加者は事実を提出し、また、いつでも事実を修正することができます。事実を変更するには、投票やランキングが必要な場合もあります。オントロジーのDAOインセンティブメカニズムは、大部分の承認を受け取り、その事実に最も近い提案が、最大の報酬を受け取るものであることを保証します。

---

2. すべてのデータの使用は、データ所有者または関連する認証機関によって承認され、プライバシー規定に従う必要があります。

予測期間が終了すると、HydraDAOは自動的に仲裁を開始します。 仲裁が開始されると、HydraDAOはすべての提出物を計算してソートし、DAO契約資金を報酬の受領者の口座に割り当てます。HydraDAOを終了すると、スマートコントラクトにバインドするためのファクトステータスが出力されます。

5.5.2. 外部の信頼できるデータソース

不和を避けるために、事前定義された構成可能なオラクルスマートコントラクトは、スマートコントラクトの検証条件が満たされるまで、値転送が行われないようにします。オラクルスマートコントラクトは、すべてのJSON APIタイプをサポートし、簡単な定義と開発が可能です。

信頼できるデータソースアクセスプロセスは、以下の通りです。

1) データの場所、データ・ソースの取得頻度、有効期限、*API*証明書、プライバシー条件、およびその他の入力パラメータを選択して、オラクルスマートコントラクトを生成します。

2) 他の検証者が評価するためのスマートコントラクトを定義します。

記述的情報には、コントラクトの機能を説明する明確な指示が含まれていなければいけません。コントラクトのラベルにより、ユーザーはブロックチェーン上で簡単にそのラベルを見つけることができ、透明性が向上します。詳細な「*if ... then ...*」の説明は、実行されるトランザクションの条件を説明することができます。

3) 実世界における法的文書との関連

オラクルスマートコントラクトは、コントラクト発行者のデジタル署名を使用して拘束力のある法的文書に署名します。法的文書をアップロードすることによって、ブロックチェーンに証拠が置かれます。

4) 契約書の共同署名者への通知

コントラクトの条件と関係者によると、署名が必要な参加者は*IM*・電子メール・電話・その他によって通知されます。すべての署名者が署名するまで、締め切り前にコントラクトは実行されません。署名者が期限前に署名しない場合、契約は無効となります。参加者がコントラクトに署名すると、オラクルスマートコントラクトは参加者に資金エスクローのアドレスを割り当てます。

5) オラクルスマートコントラクトエスクローアドレスのためのトークンの再登録

外部データソースを使用するとコストが発生します。ユーザーは、スマートコントラクトによって管理される、スマートコントラクトによって作成された住所をあらかじめ記入する必要があります。

6) コントラクト発行を完了する

コントラクトに署名して入金が完了すると、コントラクトはオントロジー上で自動的に実行されます。取引が完了すると、エスクローアドレスからのスマートコントラクトは、契約条件に従って資金を払います。取引の要件を満たすことができない資金や不正な取引は、自動的にエスクロー口座に返還されます。

7) (（オプション）データオープンプラットフォーム、オープンデータアクセスプロトコル

オープンなデータ共有プラットフォームでは、オラクルスマートコントラクトは、提供された*API*およびデータ使用プロトコルに基づいて、関連するオフチェーンデータと直接つながることができます。 オープンで透明性のある条件に基づいて、プロセス全体の用語と法的情報が公開され、ブロックチェーンのすべてのユーザーが問合せ/取得することができ、そのようなデータの非営利性によりオラクルスマートコントラクトの構成プロセスが簡素化されます。データとステータスの更新を維持するだけで、支払関係を維持する必要はありません。

6. コアプロトコル

6.1. 多要素認証プロトコル

多要素認証は、従来の単一のアイデンティティ認証体系と異なります。オントロジーは、外部のアイデンティティ信頼情報源の認証とオントロジーのエンティティの承認を統合する多要素認証システムをエンティティに提供することができます。企業は情報を提供するだけでなく、自分が所有しているもの、必要なもの、スキルを持っていること、アイデンティティに関連するその他の情報を提供して、包括的なアイデンティティ・ポートフォリオを作成することもできます。

多要素認証プロトコルには、次の二つのモードがあります。

- 外部信用証明：オントロジーは、自己署名された検証可能な宣言という形で、外部信頼ソールに*ONT ID*をバインドします。すべてのエンティティは、*ONT ID*にバインドされた外部信頼ソースを検証することによってエンティティの身元を検証できます。エンティティの身分証明の信頼性は、*ONT ID*にバインドされた外部トラストソースの認証によって決定されます。
- オントロジーエンティティ間の認証：オントロジー内のエンティティは、それらの間で署名する宣言を発行することによって、お互いのアイデンティティを認証することもできます。

6.1.1. 外部信用認証

外部信用認証は、自身による導入とトラストアンカー導入という二つの導入方式をサポートします。

6.1.1.1. 自身による導入

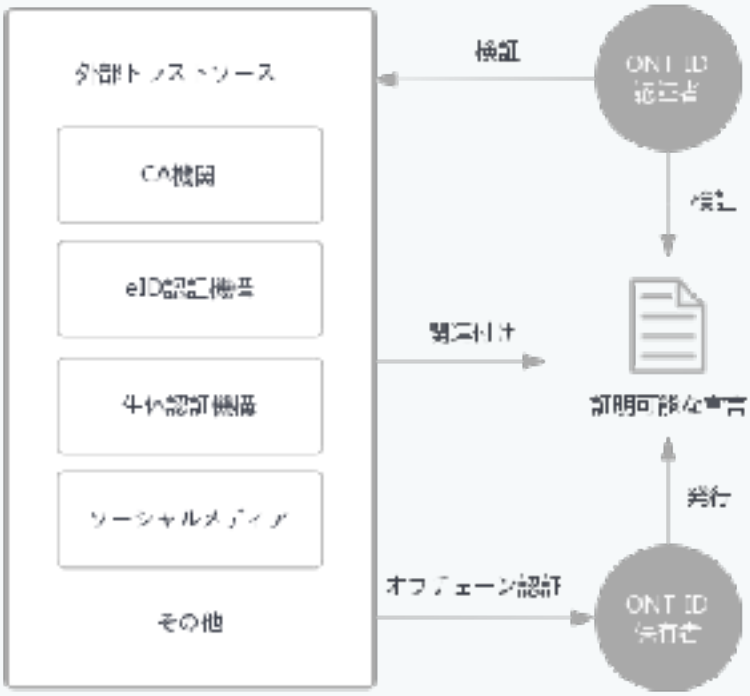
ユーザーは、既存の信頼システムを利用して、ソーシャルメディア、電子バンキングなどを通じて信頼関係を結びつけます。原則は非常に簡単です。最初にユーザは、外部信頼源の証明アドレスをオントロジーに追加します。次にユーザーは、そのプルーフアドレスに検証可能な宣言を提供します。そのフォーマットは次のとおりです。

- 宣言の作成と有効期限。
- 宣言の内容：宣言の属性、*ONT ID*、ソーシャルメディアの種類、ソーシャルメディアのユーザー名など。
- 署名：既に*ONT ID*に含まれている公開鍵を指定します。

第三者がユーザーの外部IDを確認する必要がある場合、まずオントロジーでユーザーの信頼元の証明書アドレスを読み取り、アドレスにアクセスして検証可能なクレームを取得し、最後に検証可能なクレームを検証します。

6.1.1.2. トラストアンカー導入

トラストアンカーは、通常、政府機関などの公的機関、企業、非営利団体、および認証された権限を持つ個人です。トラストアンカーは、独自の認証方法を使用してエンティティを認証し、認証されたエンティティに対して検証可能な宣言を発行します。 宣言は、認証されたエンティティの識別情報を含む必要はありません。 ONT IDと認証サービスのみが記録され、認証結果を提供することができます。 認証モデルは次のとおりです。

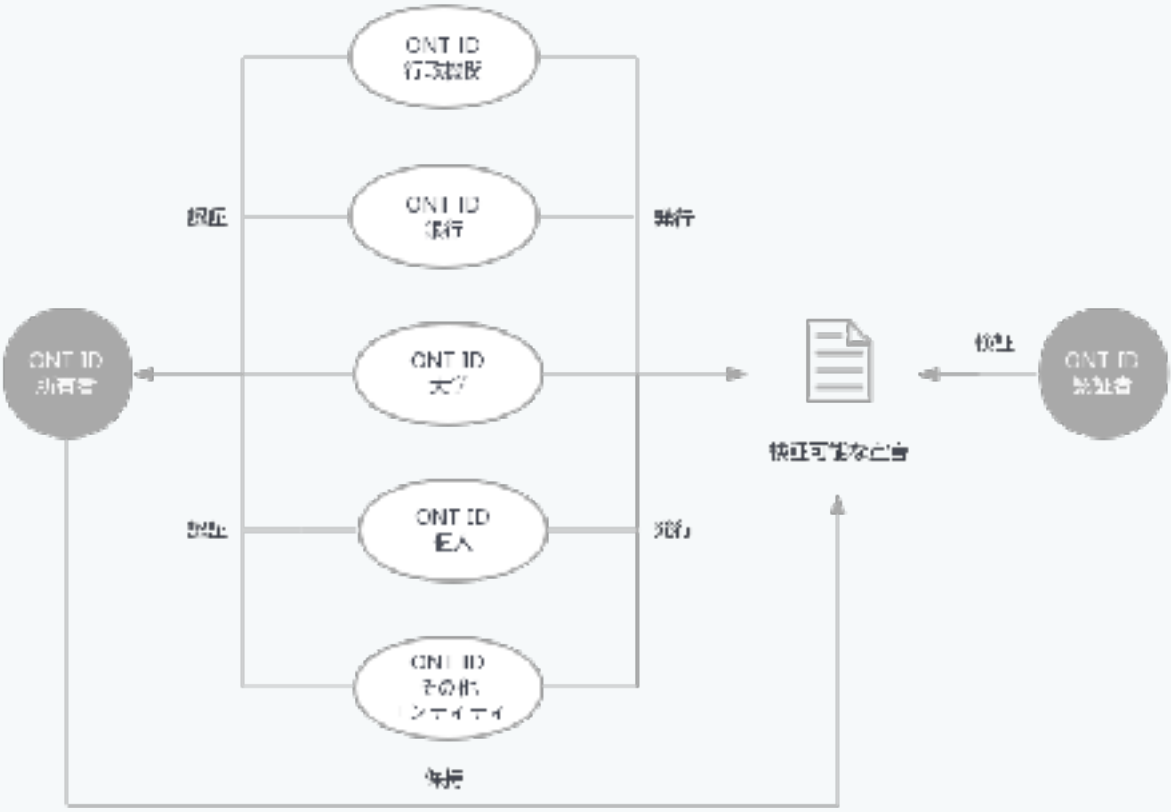


図表6.1: トラストアンカー導入

6.1.2. オントロジーエンティティ間の身元認証

オントロジーのエンティティは、図表6.2に示すように、オントロジーの（外部の信頼情報源からの）身元認証を通過したエンティティを介してアイデンティティを認証することもできます。

検証可能な主張の多様性のために、オントロジーのどのエンティティも、別のエンティティに関する何かについて宣言することができます。これは、従来の身元認証の概念をはるかに超えたオントロジーの身元認証を必要とします。政府機関、学校、病院、銀行、企業、家族、友人、パートナー、コミュニティリーダー、同僚、教師からの制限を収集することにより、従来のシステムよりもはるかに効果的な多面認証を作成できます。



図表6.2: エンティティ間の認証

6.2. ユーザー認証プロトコル

データに関わるアクセスやトランザクションはすべて、ユーザーが許可する必要があります。この点を考慮して、ユーザーのデータプライバシーを保護する一連のユーザー認証プロトコルを考案しました。このプロトコルでは、検証可能な要求を使用して非同期および検証可能な認証が行われ、委任された承認や細かいアクセス制御がサポートされています。

6.2.1. ロール

ユーザー認証プロトコルの主な役割は次のとおりです。

- *U*ユーザー：リソースの所有権を持ち、リソースへのアクセス権を付与できるエンティティ。
- リソース依頼者：ユーザーデータやその他のリソースを取得する必要がある依頼者。
- リソース提供者：ユーザーデータまたはその他のリソースを提供するサービス提供者。.
- 認可サーバー：認可リクエストを受信して処理し、ユーザーに対して委任された認可サービスを提供できる供給者。

6.2.2. 権限付与

アプリケーションシナリオのユーザー承認プロトコルは、三つのフェーズに分かれています。

1) 相互登録：ユーザーは、指定されたリソースを許可サーバーに登録するようにリソース提供者を許可し、許可サーバーはそのアドレスをリソース提供者に登録します。

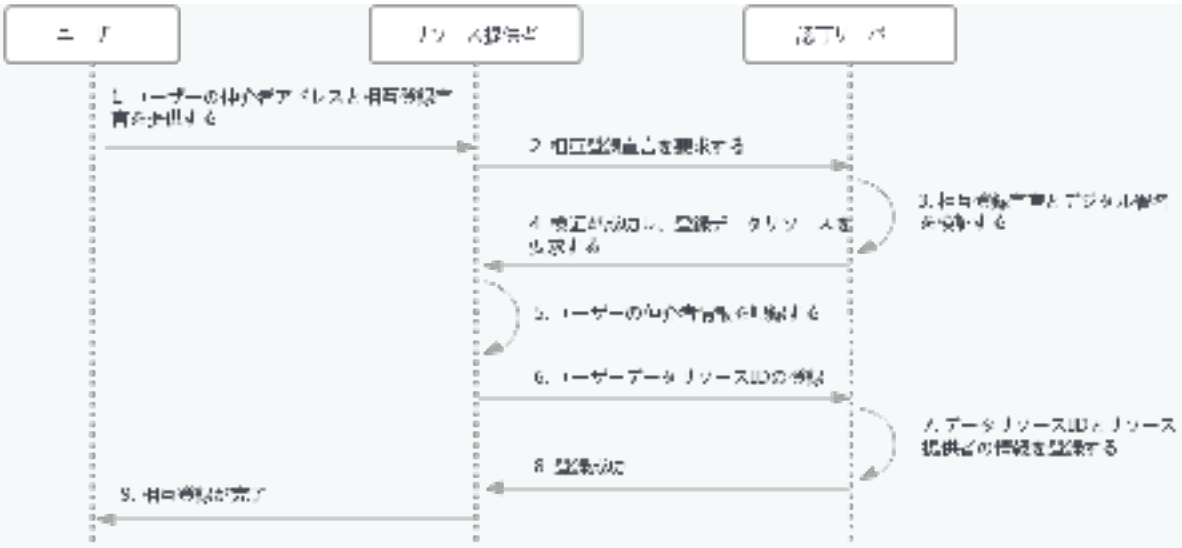
2) アクセス制御ポリシー設定：相互登録が完了した後、ユーザーは許可サーバーにアクセスし、リソースのアクセス制御ストラテジーを設定できます。

3) 認可：リソース依頼者がアクセス許可要求を開始します。認可条件が満たされている場合、リソース依頼者はリソース提供者からデータを要求するために使用される認可証明書を受け取ります。

6.2.3. 相互登録

相互登録とは、許可サーバーがそのアドレスをリソース提供者に登録することで、リソース提供者がデータアクセス要求を処理するときに、許可サーバーのアドレスを依頼者に返すことを意味します。同時にリソース提供者は、ユーザーがユーザーのリソースコントロールにアクセスできるように、独自のアドレスを認可サーバーに登録する必要もあります。

認可操作を実行する前に、ユーザーはリソース提供者と認可サーバーと協力して、相互登録プロセスを完了する必要があります。



図表6.3: 相互登録プロセス

このプロセスの簡単な説明は以下の通りです。

1) 相互登録プロセスはユーザによって開始され、ユーザは*ONT ID*、リソース提供者のアドレス、および認証サーバの*ONT ID*およびアドレスを含む相互登録クレームに自己署名します。

2) リソース提供者は、相互登録宣言と承認サーバーを通じて、両当事者はデジタル署名を検証します。

3) リソース提供者は、許可サーバーのアクセスアドレスを記録します。

4) 許可サーバーは、リソース提供者によって与えられたリソース*ID*を記録します。

6.2.4. アクセス制御ストラテジー

属性ベースのアクセス制御を使用すると、特定の属性条件を満たす依頼者が特定のリソースにアクセスするのを制限する柔軟なアクセス制御ストラテジーを設定できます。アクセス制御ストラテジーは、三つの属性条件の組み合わせであるストラテジー$A \vee (B \wedge C)$などのブール式として記述できます。認可を要求するとき、依頼者はストラテジーを満たす検証可能な宣言の属性の証明を提供しなければなりません。

6.2.5. 許可証

許可要求を受信すると、許可サーバーは所有者に許可を実行するように通知します。アクセス制御ストラテジーを満たしている供給者の場合、所有者はそれらのための許可証を発行します。許

可証の有効期間内に、依頼者は再度認証を確認せずに、データに繰り返しアクセスすることができます。

6.2.6. 委任認証

認可サーバーは、アクセス制御ストラテジーを設定できるようにすることで、ユーザーの代理人として機能することができます。この状況では、サーバーは許可要求を処理し、ユーザーと対話せずに許可証を発行できます。許可証の妥当性を証明するには、ユーザーが認可サーバーに対して承認結果の有効性を証明する必要があります。

6.3. 分散型データ交換プロトコル

集中型データ交換の欠点には、データキャッシング、ユーザー認証なしのデータの使用、データの著作権保護などがあります。オントロジーは、エンティティ間のデータトランザクションのための一連のプロトコル仕様を定義する分散型データ交換プロトコル（DDEP）を提案しています。

取引における両当事者の持分を保護するために、「保証人」として働く仲買人が契約の取引プロセスに導入され、決済プロセスが確実かつスムーズに行われるようにします。仲介者は、最終的な取引結果に基づいて、買い手の資金を保管し、売り手または買い手に資金を移転する責任があります。仲介 業者は取引の最終決済を担当しているため、十分に公平かつ安全でなければなりません。それは公的および分散管理機能を備えた分散型元帳元帳契約で機能し、仲介の役割を適切に果たすことができるようにします。

6.3.1. ロール

分散型データ交換プロトコルの主な役割は次の通りです。

- データ依頼者：データを購入する希望のある代理店/事業者/個人。
- データ提供者：オリジナルデータと処理データの両方を販売するデータ代理店/ビジネス/個人。データは、地方自治体の法律や規制を満たす必要があります。
- ユーザーの仲介者：データトランザクションのユーザー承認要件を満たすためにユーザーと双方に対話する責任があります。ユーザーの仲介者の組織形態は多種多様です（企業*OA*システム、インターネットプラットフォーム、さらには単純な*SMS*ゲートウェイ）。しかし、アプリケーションプロトコルフレームワークのユーザー許可プロトコルで定義されているように、完全に実装する必要があります。

- データ所有者：データ主体。すなわち、機関/企業/個人になります。

6.3.2. ユーザー認証

データ交換アーキテクチャでは、トランザクションデータがデータ所有者によって認可される必要があるため、認可プロセスは、セクション6.2で説明したユーザ認可プロトコルに完全に準拠しています。

6.3.3. セキュアトランザクション

スマートコントラクトによるセキュアトランザクションプロトコルは、取引活動のための一元化された第三者保証サービスを提供し、データ依頼者とプロバイダの公平性を保護する安全でスムーズなトランザクションプロセスを可能にします。

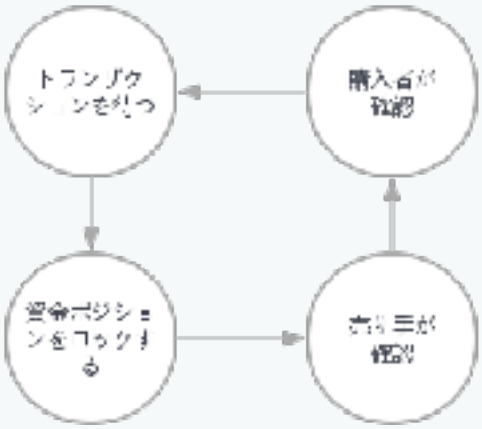


図 6.4: 保証人取引プロセス

セキュアスマートコントラクトトランザクションの実装プロセスは次のとおりです。

1) データ供給者は注文を入力し、リソースID、データ特性、売り手のアカウント、価格、およびその他の機能を含む製品情報を書き込み、契約は依頼者がトランザクションを開始するのを待ちます。

2) データ依頼者は指定された金額を契約に入金し、移転された金額が販売要件を満たしているかどうかをコントラクトがチェックします。

   a) 契約が合格すると、資金ロック状態に入ります。

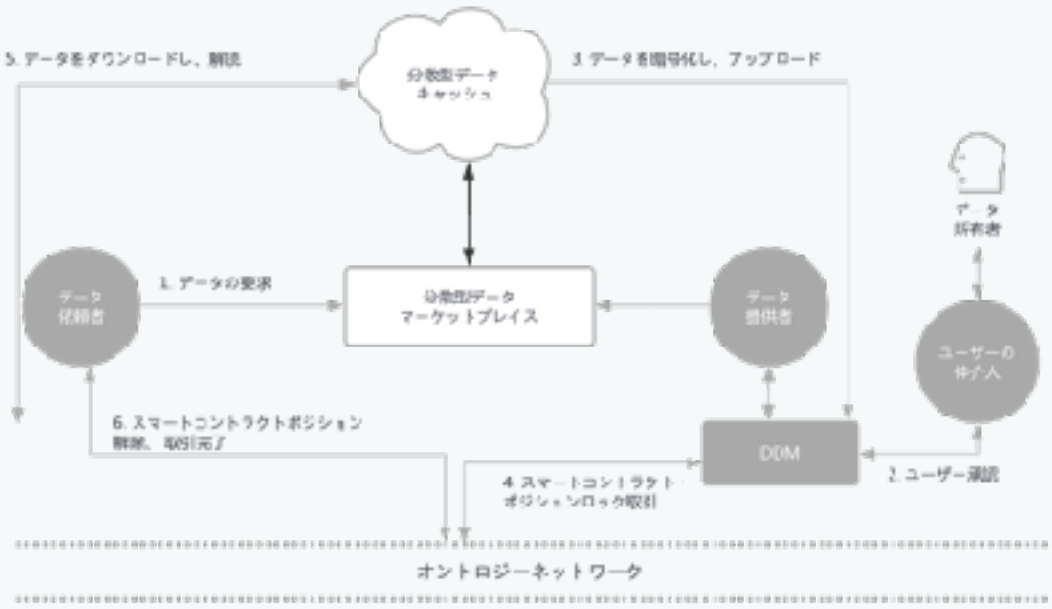
   b) チェックが失敗した場合は、エラーメッセージが譲受人に返され、トランザクションステータスが返されます 以前の状態に戻します。

3) 供給者がデータを提供した後、契約は有効期限を確認し設定します。有効期間の終了時に他のアクションが実行されなかった場合、契約は自動的に決済プロセスに入ります（ステップ5）。

4) データを受信した後、依頼者はコントラクトに確認します。

5) コントラクトは資金を供給者の口座に振り替えて、この取引を完了し、取引状態を待つ状態に戻ります。

6.3.4. データ交換プロセス



図表6.5: データ交換プロセス

先行予約：取引準備

データ製品発表：データ提供者はデータ製品情報を市場に公開します。依頼者は市場をみて、データ製品を検索し、購入したいデータを選択することができます。メタデータ情報にはデータリソースの導入、キーワード、データリソースハッシュ、契約支払いアドレス、その他の情報が含まれますが、これらに限定されません。

相互登録：データ製品がデータ所有者の承認を必要とする場合、データ提供者はデータを公開する前に、ユーザー指定のユーザーの仲介者との相互登録を完了する必要があります。詳細は上記の相互登録を参照してください。

1) 取引依頼

購入したいデータを見つけた後、依頼者はオントロジーを通してデータ提供者の身元を確認します。詳細については、多要素認証プロトコルを参照します。トランザクション要求が開始される前に、依頼者はまずコントラクトアドレスに資金を入金し、購入データ要求を提供者に送信し、ユーザーの認可に必要な情報を添付します。この要求には、取引情報とONT ID情報が含まれますが、これに限定されません。

2) 認可

依頼者から要求を受信した後、データ提供者はユーザーの仲介者にアクセスし、許可要求を開始します。この時点で、ユーザーの仲介者は、オントロジーを介してオンデマンドで依頼者の身元確認を行いし、所有者によって事前に提供されるアクセス制御ポリシーに従って認可を実行することができます。所有者がアクセス制御ポリシーを設定しない場合、ユーザー仲介者は所有者に許可を通知します。認可要求が拒否された場合、トランザクションは終了します。

3) データのアップロード

データ提供者は、依頼者によってサポートされている暗号化アルゴリズムに従ってワンタイムセッション鍵を生成し、これを使用してトランザクションのデータおよびデータ特性値を暗号化し、暗号文を中間ストレージシステム（例えばIPFS）に送信します。

4) ポジションのロック

データ提供者はスマートコントラクトを呼び出して、依頼者が入金した資金を確認します。金額が正しければ、取引が完了またはキャンセルされるまで、ポジションがロックされます。一方、提供者は、要求者の公開鍵でセッション鍵を暗号化し、それを安全なチャネルを介して依頼者側に送信します。

5) データの受信

スマートコントラクトイベントの通知（セクション5.3のイベントメカニズム）を受信した後、依頼者は中間ストレージからデータを取得し、それをセッションキーで解読し、平文固有値を計算して検証し、検証が成功した場合、ステップ6を実行します。

6) 取引確認

データ依頼者側は取引コントラクトに取引完了を確認し、コントラクト資金がデータ提供者の口座に転送されます。

例外処理メカニズム：例外処理メカニズムは、多様なビジネスシナリオに従ってカスタマイズできます。例えば、データが依頼者によって指定期間内に確認されていない場合、提供者はコント

ラクトで資金のロックを解除することができます。スマートコントラクトによって、自動的に資金がロック解除されます。

6.3.5. プライバシー保護

いくつかのデータ交換の場合、参加者はトランザクションの非表示を希望するかもしれません。ステルスアドレス技術を使用して、受信者のトークンアドレスとONT IDを相関させることが実行不可能になるように提案します。依頼者は不可視のアドレスを生成し、秘密鍵は自分のトークンアドレスに基づいて、データ提供者によってのみ把握できます。

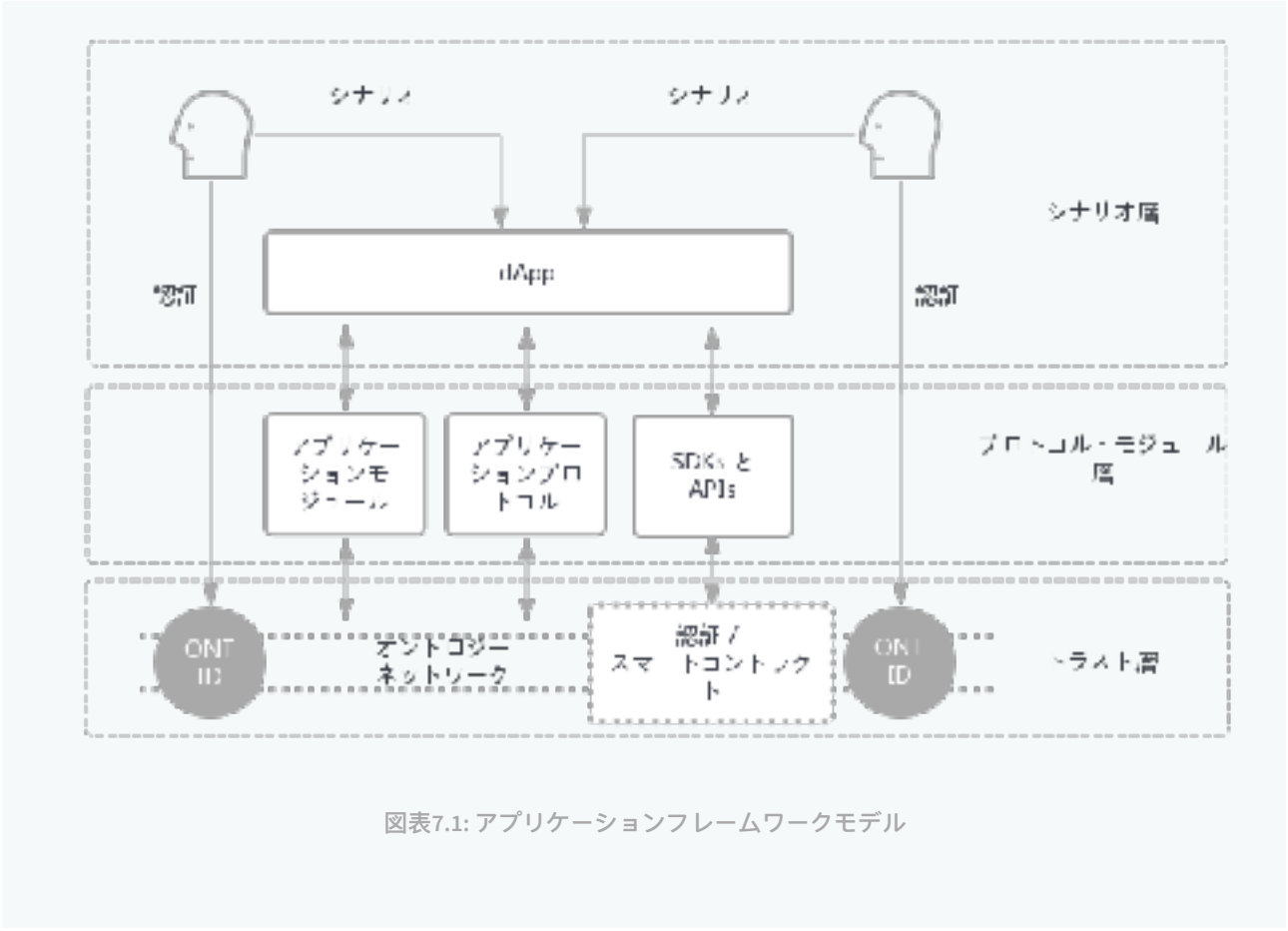
データプロバイダのトークンアドレスを$S = s \cdot G$とすします。依頼者は、乱数rを生成し、$R = r \cdot G$を計算します。次に、ステルスアドレス$E = Hash(r \cdot S) \cdot G + S$がトークントランザクションの出力アドレスとして使用されます。提供者だけが、自分の秘密鍵で秘密鍵=ハッシュ（s・R）+sを計算することができます。

7. オントロジーアプリケーションフレームワーク

7.1. アプリケーションフレームワークモデル

オントロジーのアプリケーションフレームワークは、サードパーティのdApp開発者が、分散型元帳の複雑さを理解しなくても、基盤となる分散型アプリケーションを迅速に構築できるように豊富な一連のアプリケーションプロトコルとモジュールを提供します。オントロジーのアプリケーションフレームワークはスケーラビリティが高く、シナリオの要求に応じて拡張することができます。

図表7.1は、dAppがアプリケーションフレームワークを使用してどのようにオントロジーと双方で作用し、分散型の信頼を実現するかを示しています。



図表7.1: アプリケーションフレームワークモデル

- トラスト層：オントロジーは、その身元認証、認証、スマートコントラクトシステムなどを通じて、信頼システムの責任を引き継ぎます。
- モジュールプロトコル層：この層は、上位層のシナリオが、アプリケーションモジュール、アプリケーションプロトコル、*SDK*、および*API*を通じてオントロジーをよりよく利用できるようにします。
- アプリケーション層：さまざまなシナリオのための*dApps*であり、それらの背景にあるチームは、シナリオ開発、ユーザーサービスなどに重点を置くことができます。オントロジーは、信頼の問題を解決するのに役立ちます。

7.2. データ市場

将来はデジタル世界であり、将来のデータ交換市場はデータ所有権の移転だけに限定されません。信頼できる共同計算も重要な協調の手段になります。データマーケットプレイスの商品には、データ、データ予測、データコン計算リソースなどが含まれます。参加者には、データ共同エコシステムを形成するディープラーニングやその他のAIテクノロジを使用するビッグデータサービスなどのデータコンシューマ、プロバイダ、プロセッサが含まれます。この複雑なデータコラボレーションエコシステムには、対応するインフラストラクチャが必要です。

オントロジーの分散型データ交換プロトコル（DDEP）、データトレーディングモジュール（DDM）、および一連の暗号化モジュールは、完全な分散型データ取引およびコラボレーションフレームワークを形成し、オントロジーdAppユーザーのために業界全体で大規模な取引規模および交換ニーズを可能にします。データ交換サービス提供者のレイヤーは、業界間でさまざまなデータ交換の分類を可能にします。

7.3. データ取引モジュール

データ取引モジュール（DDM）は、分散型データ交換プロトコルを基礎としたオントロジーの重要な基礎モジュールです。dAPP開発者やデータ取引の参加者はこのモジュールを通じて、速やかにデータの取引をできるようになります。このモジュールは RESTful API、RPC、SDK などを提供し、さまざまなプロトコルにもサポートします。

データ取引モジュールには場面に応じて、たくさんの種類があります。例えば、データ取引サーバー、単一ユーザー用のクライアント、複数ユーザー用のクライアント、ライトウォレットクラ

イアントなど。このモジュールには、ONT身元認証管理、データリソース管理、スマートコントラクト取引、P2Pコミュニケーションという四つの機能があります。

DDMデータトランザクションモジュールには、次のような利点と特長があります。

- アクセスコントロールモジュールとの分離：データ取引モジュールが抑制するのはデータリソースとデータ所有者のコントロールサーバーのバインドのみの関係だけで、アクセス権限の配置と検証には参加しません。これでデータ所有者の資産プライバシーを保護するだけでなく、売り手の信用を高め不必要な争いを回避できます。
- データのプライバシーの保護：データ取引モジュールは実際のデータを格納せず、取引データも暗号化できます。これで売り手がデータ降下を心配する必要はなくなります。
- 依頼者はデータの検索だけでなく、通信を通じてデータ提供者に知らせることもできます。
- 「単一モジュール、単一機能」という原則にしたがって設計されます。データ取引モジュールは暗号化セキュリティモジュールやユーザー認証モジュールと合わせて、多種多様な場面に容易に対応できるようになります。

7.4. 暗号化とセキュリティモジュール

7.4.1. セキュアマルチ計算

典型的な共同計算シナリオでは、2人以上の参加者は、他者と共有せずに自身がデータを使用して計算を行うことを望むかもしれません。このシナリオでは、現在使用されているアプローチは機能しません。たとえば、アプリケーション開発会社Aが強力な機械学習アルゴリズムをアプリケーションに統合したい場合、従来のシステムはデータをMLアルゴリズムプロバイダPに転送します。次に、Pはアルゴリズムでデータを実行し、結果をAに返します。そうするとPは、Aの情報に関するすべてがわかるでしょう。

上記のようなシナリオでは、セキュアマルチ計算（MPC）で処理します。最初の安全な2者計算プロトコルは、「億万長者の問題」を解決するために活用できるアンドリューヤオによって初めて導入されました。二人の億万長者が、自分たちの実際の富を公開せずに、誰がよりお金持ちであるかを知ることに興味があります。この一般的な問題は、n人の参加者のそれぞれがいくつかのプライベートデータx_iを保持し、その非公開データに関する公開関数の値$f(x_1, \ldots, x_n)$を計算したいとまとめることができます。この目標は、各当事者が知ることができるすべての情報が出力と自分の入力からわかることができるようなプロトコルを設計することです。したがって、すべての参加者に追加のデータ漏洩はありません。

MPCに対して、2つの主要なアプローチが長年に渡って開発されてきました。それは、ヤオ氏による不明瞭な回路[22][24]と秘密共有に基づくもの[23]です。このセクションの残りでは、後者のアプローチに焦点を当てます。

簡単に言うと、（t、n）閾値秘密分散法では、秘密はnの部分に分割されます。参加者は自分のシェアだけもらえます。秘密が公開されるのは参加者が t 人以上の場合に限られています。秘密のシェアに基づくSMCプロトコルについて、通常のやり方は秘密シェアアルゴリズムで、実行の段階的な結果を異なる参加に配ります。参加者たちは受け取った各自のシェアから、最後の$f(x_1, \ldots, x_n)$を再現します。

7.4.2. 完全準拠型暗号化

多くのデータ交換のシナリオでは、データ提供者は、データ全体を依頼者に転送するのではなく、依頼者に自分のデータへのアクセス許可を与える必要があります。企業がデータへのアクセスを制限しながらデータのプライバシーを維持できることが重要です。近年の暗号の進歩、すなわち完全準同型暗号化は、この問題[25][26][27][28]に対する非常に有望な解決法を提供します。たとえば、会社は公開鍵を使用してデータmを暗号化し、暗号文を複雑な計算fを行う相手に送信します。この新しい暗号文は、$f(m)$の暗号化であります。最後に、新しい暗号文が企業に送り返され、秘密鍵を使用して復号化して$f(m)$を得ることができます。

完全準拠型暗号化のスキームは、すべての公開鍵暗号方式に共通する3つの基本アルゴリズム（鍵の生成、暗号化、復号、準同型加算CAddと乗算$CMul$で構成されています。 例えば、C1はM1の暗号文で、C2はM2の暗号文だと仮定すると、この二つの計算や最適化の方法を使ってより複雑な暗号文計算を構築できます。

$$Decrypt(CAdd(C_1, C_2)) = M_1 + M_2$$
$$Decrypt(CMul(C_1, C_2)) = M_1 \times M_2$$

これら二つの基本演算からなる演算回路として、多くの複雑な演算を組み込むことができます。

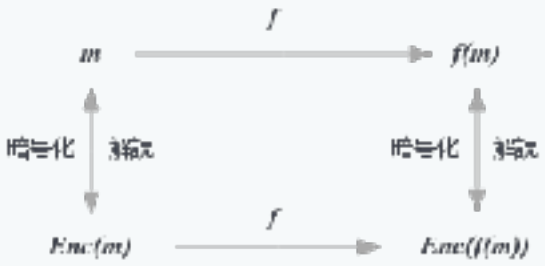


図 7.2: 暗号文の同型計算

したがって、FHE方式を用いることにより、暗号文に対して準同型演算fを実行することができ、解読アルゴリズムはf（m）を（上記のグラフに示すように）戻します。将来、BGVやFVなどの様々なFHE方式をサポートする予定です。

7.4.3. データ著作権

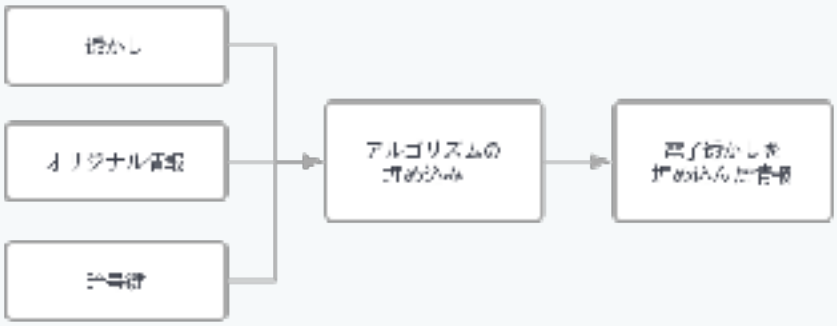
オントロジーは、データのデジタル特性に基づいて、データストレージとライフサイクル管理機能を提供します。まず、オントロジーはライフサイクルの追跡可能なメカニズムを設計します。オントロジーは、登録、要求、承認、およびトランザクションのプロセス全体を追跡するために、各データのデジタルIDを確立します。そして、オントロジーは、データの著作権およびトランザクション情報がすべて分散型元帳に記録されることを確かにします。オントロジーは、デジタル透かし技術を使用して、識別情報をデジタルキャリアに直接埋め込みます。使用価値に影響を与えずに、オントロジーは攻撃者を容易に識別し、改ざんを防ぐことができます。キャリア内の隠された情報によって、依頼者はコンテンツ作成者を容易に確認し、キャリアが改ざんされているかどうかを判断することができます。デジタル透かしは、偽造防止の追跡、著作権保護、隠れた識別、認証、秘密の通信を実装するための効果的な方法です。

オントロジーで使用される電子透かし技術の主な特徴は以下の通りです。

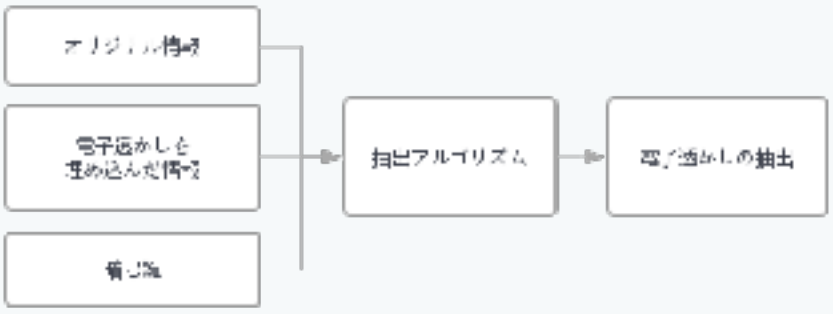
- 正当性の証明：透かしは、著作権で保護されたプロダクトの属性の完全で、信頼できる証拠を提供します。電子透かしアルゴリズムを使用すると、ユーザーは、保護されたオブジェクトに埋め込まれた所有者の情報（登録ユーザーナンバー、商品ロゴ、テキストなど）を特定し、必要なときに抽出することができます。透かしを使用して、オブジェクトが保護されているかどうかを判断し、保護されたデータの拡散を監視し、認証を行い、不正なコピーを防止することができます。

- 知覚不能：埋め込み透かしは、見えることがありません。理想的な環境は、透かし入り画像が元の画像と視覚的に同一であることであり、これはほとんどの透かしアルゴリズムが達成できるものです。
- 安定性：意図しないまたは意図的ではないさまざまな信号処理が行われた後も、電子透かしは完全性を維持し、正確に識別できます。透かしには確実性が非常に重要です。電子透かしは、意図的なひずみ（例えば、悪意のある攻撃）および、意図しないひずみ（画像圧縮、フィルタリング、スキャンおよびコピー、ノイズによる影響、サイズ変化など）を含む、異なる物理的および幾何学的ひずみに耐えることができるべきです。揺るぎない透かしアルゴリズムは、透かし入り画像から埋め込み透かしを抽出すること、または透かしの存在を証明することができなければなりません。特定の透かし埋め込みおよび検出アルゴリズムが欠けている場合、データ製品の著作権保護マークは偽造するのが難しいはずです。攻撃者が透かしを削除しようとすると、マルチメディア商品がデータを破損させます。

図表7.3にあるように、オントロジーのデータ透かしには、2つの機能があります。透かしを埋め込むためのものと取り除くためのものです。電子透かしは、文書、請求書、またデジタル認証のセキュリティシステムとして使用されます。電子透かしによって文書の正当性が証明され、また違法コピーから保護されます。



(a) 電子透かしの埋め込みプロセス



(b) 電子透かしの抽出プロセス

Figure 7.3: 電子透かしモジュール

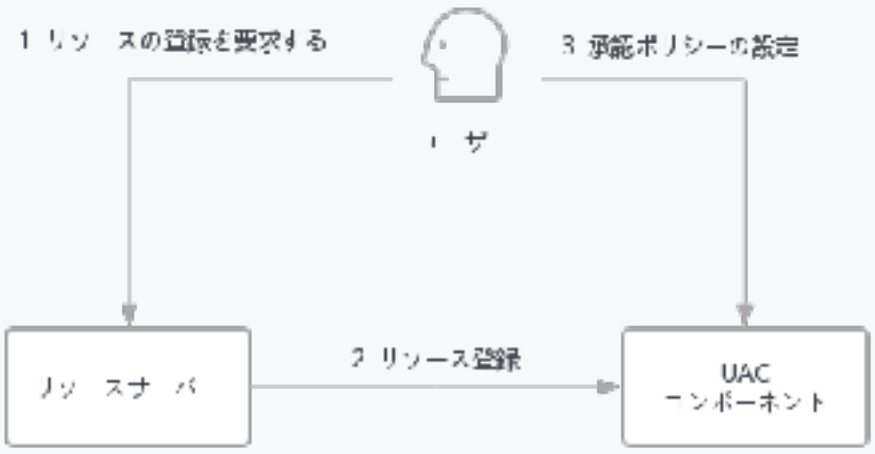
7.5. ユーザー認証コントローラー

ユーザ認証制御（UAC）モジュールは、ユーザ認証プロトコルに基づいています。 きめ細かなアクセス制御を使用して、UACモジュールはデータ所有者が自らのデータの承認を得られるように支援します。データに関連するトランザクションは、データ所有者にデータトランザクションの承認を行うことを通知します。UACモジュールは、リレーショナルデータベース（Mysqlまたはオラクル）の使用をサポートする外部使用のためのRESTful APIを提供します。 それらには二つの主要な機能を持っています:

- ユーザーデータ承認アクセスポリシー設定。
- ユーザーデータ承認アクセス制御。

7.5.1. 承認ポリシーの設定

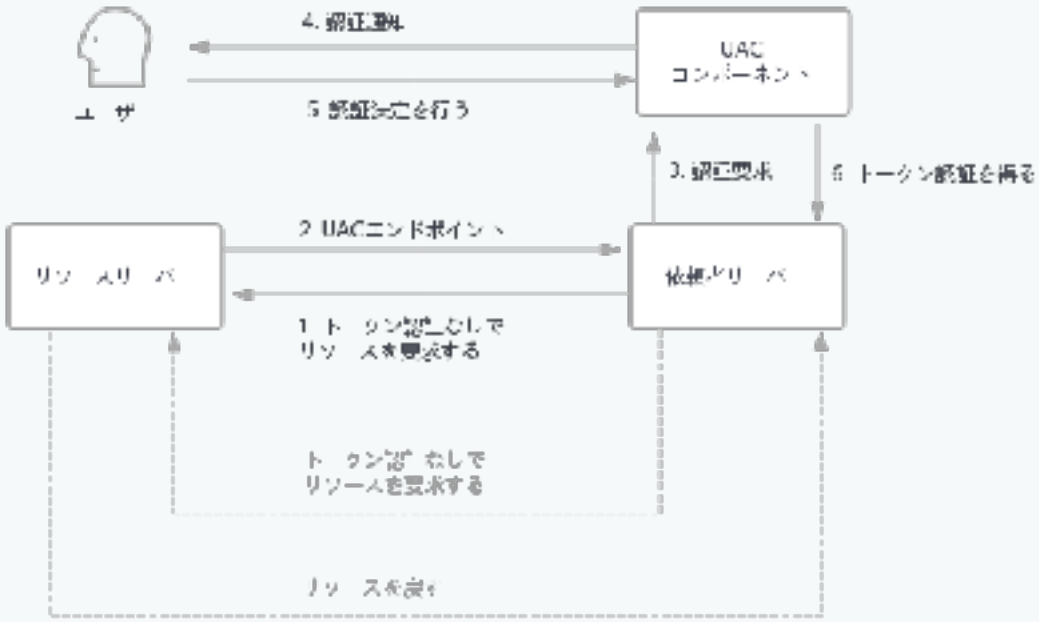
ユーザは、UACモジュールによって提供されるRESTful APIを使用して、データ提供者を登録するようにリソース提供者に要求することができます。 登録後、ユーザはUACモジュールを使用して、登録データを検索および再検討し、データアクセスおよび制御ポリシー設定を行うことができます。頻繁にアクセスされる、特定のプライバシー要件の低いデータの場合、ユーザーはUACモジュールで認可ホスティングを設定できます。



図表7.4: 承認アクセスポリシーの設定

7.5.2. アクセス制御

データ取引において、ホスティングしていない場合、ユーザーのデータを保護するために、ユーザー認証制御（UAC）モジュールはユーザーに許可を求める通知を送り、依頼者がデータにアクセスできるかどうかをユーザー自身で決めます。ホスティングしている場合、ユーザーの代わりにユーザー認証制御（UAC）モジュールが認証を決めますが、ユーザーに承認のレシートを送ります。どちらの場合においても、誰がいつ、何のデータをどのように操作するかをユーザーがわかるようになります。



図表7.5: 承認アクセス制御

7.6. 宣言管理モジュール

オントロジーのトラストアンカーにとって、検証可能な宣言管理モジュールは重要な構成要素です。このモジュールは分散型トラスト体系の要求に従って開発され、RESTful API を提供し、関係データベース（Mysql や Oracle）もサポートします。主要機能は検証可能な宣言管理の発行、検証、調査、取り消しなどです。

7.7. GLOBALDB

GlobalDBとは接続可能な、分散型キーバリューデータベースです。GlobalDBの下位層がGoogle's Spanner/F1に基づくオープンソースのNewSQLデータベース TiDBです。

GlobalDBがブロックチェーン/分散型元帳と IPFS のために最適化したデータベースモジュールです。GlobalDBがSQLに互換でき、ストレージ共有や分散型業務、水平なスケーラビリティ、エラー修復能力を提供します。GlobalDBがブロックチェーンとビッグデータ、ブロックチェーンと AI などの計算場面に応用できます。

GlobalDB が分散型業務、ストレージ共有、ロードバランシング、SQL on KV という四つの特徴を持っています。

7.7.1. 分散型トランザクション

GlobalDBが完全な分散型業務を提供し、ステートシャーディングとオフチェーン業務をサポートします。この業務モデルが Google Percolator [29]に基づいて最適化されました。

GlobalDBのトランザクションモデルはオプティミスティックロックを採用しています。 分散トランザクションは、送信された競合のみを検出します。伝統的な方法が矛盾する場合、それらは再試行される必要があります。このモデルは、重大な競合の場合は非効率的であり、オプティミスティックロックモデルはほとんどのシナリオで効率が高くなります。

分散トランザクションは二つのフェーズで実行する必要があり、基礎となるデータは一貫性のあるレプリケーションを行う必要があるため、トランザクションが非常に大きいと、コミットプロセスが遅くなり、一貫性のあるレプリケーションプロセスが邪魔になります。それを避けるために、オントロジーには以下のような制限があります。

1) 単一の*KV*エントリは*6MB*を超えないこと。

2) *KV*エントリの総数は*30W*を超えないこと。

3) *KV*エントリーの合計サイズは*100MB*を超えないこと。

7.7.2. ストレージシャーディング

GlobalDB が自動的に Key の範囲にしたがって下位層データを分割します。分割されたシャーディングがすなわち$[StartKey, EndKey)$という範囲です。シャーディングにおけるキーバリューが一定の閾値を超えた時、自動的に分割します。

7.7.3. ロードバランス

ロードバランサーPが、ストレージクラスターの状態に基づいてクラスターのロードを調整します。調整はシャーディングを単位に、PD設定をロジックにして自動的に行われます。

7.7.4. SQL on KV

GlobalDB automatically maps the SQL structure to a KV structure. In a nutshell GlobalDB does two things:

- 一行のデータが*KV*にマップされます。 キーには*TableID*の接頭辞が付き、行*ID*に接尾辞が付きます。
- インデックスは*KV*にマップされます。 キーは、*TableID* + *IndexID*のプレフィックスとインデックス値の接尾辞を作成します。

一つのテーブル内のデータまたはインデックスは同じプレフィックスを持つため、TiKVのキースペースでは、キー値は隣接する位置になります。GlobalDBは、より高いデータ可読性を達成するために、対応するダーティデータ管理戦略を構成します。[30]

GlobalDBは高度に構成可能で適応性があり、オンチェーンおよびオフチェーンのリアルタイム高性能ビジネスで同時に利用できます。オントロジーの中心的な役割を果たし、基礎となる分散型元帳の基盤を提供します。

8. あとがき

このホワイトペーパーはオントロジーに関する技術の概要を掲載しています。しかし、技術そのものが進歩するにつれて、オントロジーチームは将来的にそのコンテンツを絶えず更新することになります。

一方でオントロジーは、オープンで協力的で創造的なテクノロジーエコシステム構築します。オントロジーのチームは、世界中の開発者がオントロジーの技術に参加するために、オントロジーファミリーに加わることを歓迎します。

参考文献

[1] Burnett, Daniel C. et al. "Verifiable Claims Data Model" Verifiable Claims Working Group, W3C Editor's Draft, 2017, https://w3c.github.io/vc-data-model/.

[2] McCarron, Shane et al. "Verifiable Claims Use Cases" Verifiable Claims Working Group, W3C Working Group Note, October 2017, https://w3c.github.io/vc-use-cases/.

[3] Reed, Drummond et al. "Decentralized Identifiers (DIDs)" W3C, Credentials Community Group, 2017, https://w3c-ccg.github.io/did-spec/.

[4] Hardt , D.. "The OAuth 2.0 Authorization Framework" RFC6749, October 2012.

[5] Machulak, Maciej and Machulak Richer. "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization" User-Managed Access Work Group, Draft Recommendation, 2017, https://docs.kantarainitiative.org/uma/wg/oauth-uma-grant-2.0-09.html.

[6] Jones, M., et al. "Uniform Resource Identifier (URI): Generic Syntax" RFC3986, January 2005.

[7] Adams, Carlisle, and Steve Lloyd. "Understanding PKI : concepts, standards, and deployment considerations." Boston: Addison-Wesley, 2003.

[8] Sporny, Manu et al. "JSON-LD 1.0" W3C, W3C Recommendation, January 2014, http://www.w3.org/TR/json-ld/.

[9] Longley, Dave, et al. "Linked Data Signatures". W3C Digital Verification Community Group, Draft Community Group Report. 2017, https://w3c-dvcg.github.io/ld-signatures/.

[10] Camenisch, Jan, and Anna Lysyanskaya. "A signature scheme with efficient protocols." international workshop on security (2002): 268-289.

[11] R., Cramer. "Modular design of secure yet practical cryptographic protocols." Ph. D thesis, Universiteit van Amsterdam, Netherlands, 1997.

[12] Fiat, Amos, and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems." international cryptology conference(1987): 186-194.

[13] Schnorr, Clauspeter. "Efficient signature generation by smart cards." Journal of Cryptology 4.3 (1991): 161-174.

[14] Abdelmalek, Michael, et al. "Fault-scalable Byzantine fault-tolerant services." symposium on operating systems principles 39.5 (2005): 59-74.

[15] Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance." operating systems design and implementation (1999): 173-186.

[16] Borran, Fatemeh, and Andre Schiper. "Brief announcement: a leader-free byzantine consensus algorithm." international symposium on distributed computing (2009): 479-480.

[17] Laurie, B., et al. "Certificate Transparency" RFC6962, June 2013.

[18] Merkle, Ralph C.. "A digital signature based on a conventional encryption function." Lecture Notes in Computer Science (1989).

[19] US patent 4309569, Ralph C. Merkle, "Method of providing digital signatures", published Jan 5, 1982, assigned to The Board Of Trustees Of The Leland Stanford Junior University.

[20] Matthew, S.. "Merkle Patricia Trie Specification" Ethereum, October 2017, https://github.com/ethereum/wiki/wiki/Patricia-Tree.

[21] Morrison, Donald R.. "PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric." Journal of the ACM 15.4 (1968): 514-534.

[22] Yao, Andrew C. "Protocols for secure computations." Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on. IEEE, 1982.

[23] Shamir, Adi. "How to share a secret." Communications of the ACM 22.11 (1979): 612-613.

[24] Damgård, Ivan, et al. "Practical covertly secure MPC for dishonest majority–or: breaking the SPDZ limits." European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, 2013.

[25] Gentry, Craig. "A Fully Homomorphic Encryption Scheme." Stanford University, 2009.

[26] Brakerski, Zvika, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping." ACM Transactions on Computation Theory (TOCT) 6.3 (2014): 13.

[27] Halevi, Shai, and Victor Shoup. "Algorithms in helib." International Cryptology Conference. Springer, Berlin, Heidelberg, 2014.

[28] Fan, Junfeng, and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." IACR Cryptology ePrint Archive 2012 (2012): 144.

[29] Peng, Daniel, and Frank Dabek. "Large-scale Incremental Processing Using Distributed Transactions and Notifications" Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, 2010.

[30] Shen, Li. "Understand TiDB technology insider" PingCAP, 2017, https://pingcap.com/blog-cn/tidb-internal-2/.

お問い合わせ

 Email: contact@ont.io

 Telegram: OntologyNetwork

 Twitter: OntologyNetwork

 Facebook: ONTnetwork

 Reddit: OntologyNetwork

 Discord：https://discord.gg/vKRdcct

 Medium: OntologyNetwork

 LinkedIn: Ontology Network