Ontology

技术白皮书

Version 0.6.5

更新日期：2018/03/14

新一代多链式公有链平台 &
分布式信任生态

摘要

长期以来，人们通过“技术”、“法制”、“社群”等不同维度和方法来建立信任，但这样多来源、多系统、多方法的单点式信任协作也带来了非常高的协作成本，阻碍了信任协作的深度和广度。虽然互联网技术日新月异，但关于信任的很多痛点至今依然存在，如信任源分散化、数据零散化、个体角色缺失、身份认证不准确、虚假信息难判断等。在社会治理、经济协作、金融服务等各种协作过程中，每天为“信任”产生着大量的成本。

去中心化、不可篡改的区块链从一定机制上建立起了特定场景下的技术信任，但要和现实世界的业务场景结合起来需要更多的融合机制，如何构造一个结合多样性信任和一体化应用的信任机制，成为对新一代“信任”基础体系的追求。

本体致力于建立一个体系化的、流程化的、一体化的信任生态，本体将作为信任生态体系的基础设施和连接器，为信任源的有效协同、为数据源的互联互通、为各类分布式应用服务提供完整的底层技术基础设施。[1]

本文重点针对本体的技术架构、关键技术原理及特定场景的技术协议进行阐述。

1. 本体是一个支持众多信任协作场景的基础性体系，根据场景和应用范围的应用会持续地进行各类模块与协议的扩展。本技术白皮书仅描述本体在第一阶段规划的主要基础架构和协议，后续会根据应用扩展对技术白皮书进行持续更新。

目录

1. 概述 1
2. 术语说明 3
3. 链网结构 5
4. 分布式信任框架 8
4.1. 本体身份标识协议 8
4.1.1. 生成ONT ID 9
4.1.2. 自主管理 9
4.1.3. 多密钥绑定 9
4.1.4. 授权控制 9
4.2. 信任模型 9
4.2.1. 中心化的信任模型 9
4.2.2. 去中心化的信任模型 10
4.3. 可验证声明 10
4.3.1. 生命周期 11
4.3.2. 匿名声明 12
5. 分布式账本技术 15
5.1. 分布式账本 15
5.1.1. 共识算法 15
5.1.2. 流程协同 19
5.1.3. 存证设计 19
5.2. 智能合约 19
5.3. 共享数据合约模型 20
5.4. 默克尔树存储模型 22
5.4.1. 默克尔哈希树 23
5.4.2. 默克尔审计路径 23
5.4.3. 默克尔一致性证明 24

© 2017 ONT Team

5.4.4. 默克尔-帕特里夏树 25
5.5. 混合预言机 25
5.5.1. 内置的数据预测DAO 27
5.5.2. 外部可信数据源 27
6. 核心协议标准 29
6.1. 多源认证协议 29
6.1.1. 外部信任源认证 29
6.1.2. 本体实体间的身份认证 30
6.2. 用户授权协议 31
6.2.1. 角色定义 31
6.2.2. 授权流程 32
6.2.3. 双向注册 32
6.2.4. 访问控制策略 33
6.2.5. 授权证明 33
6.2.6. 授权托管 33
6.3. 分布式数据交换协议 33
6.3.1. 角色定义 34
6.3.2. 用户授权机制 34
6.3.3. 担保交易模式 34
6.3.4. 数据交换流程 35
6.3.5. 对交易者的隐私保护技术 37
7. 本体应用框架 38
7.1. 应用框架模型 38
7.2. 数据交易市场 39
7.3. 数据交易组件 39
7.4. 密码学及安全组件 41
7.4.1. 安全多方计算 41
7.4.2. 全同态加密 41
7.4.3. 数据的版权保护 42

© 2017 ONT Team

7.5. 用户授权控制组件 44
7.5.1. 授权访问策略设置 44
7.5.2. 授权访问控制 44
7.6. 声明管理组件 45
7.7. 全局事务数据库 45
7.7.1. 分布式事务 46
7.7.2. 存储分片 46
7.7.3. 负载均衡 46
7.7.4. SQL on KV 46
8. 后记 48
参考文献 49
联系方式 51

© 2017 ONT Team

1. 概述

本体（Ontology）是一个多链、多系统融合的链群结构，除了本体本身的分布式账本框架可以支持实现不同治理模式下的区块链体系，也可与来自不同业务领域、不同地区的不同链，通过本体的各类协议进行协作，形成各类异构区块链和传统信息系统的跨链、跨系统交互映射。因此，本体又被称为本体链群或本体链网，即区块链之间的互联网。

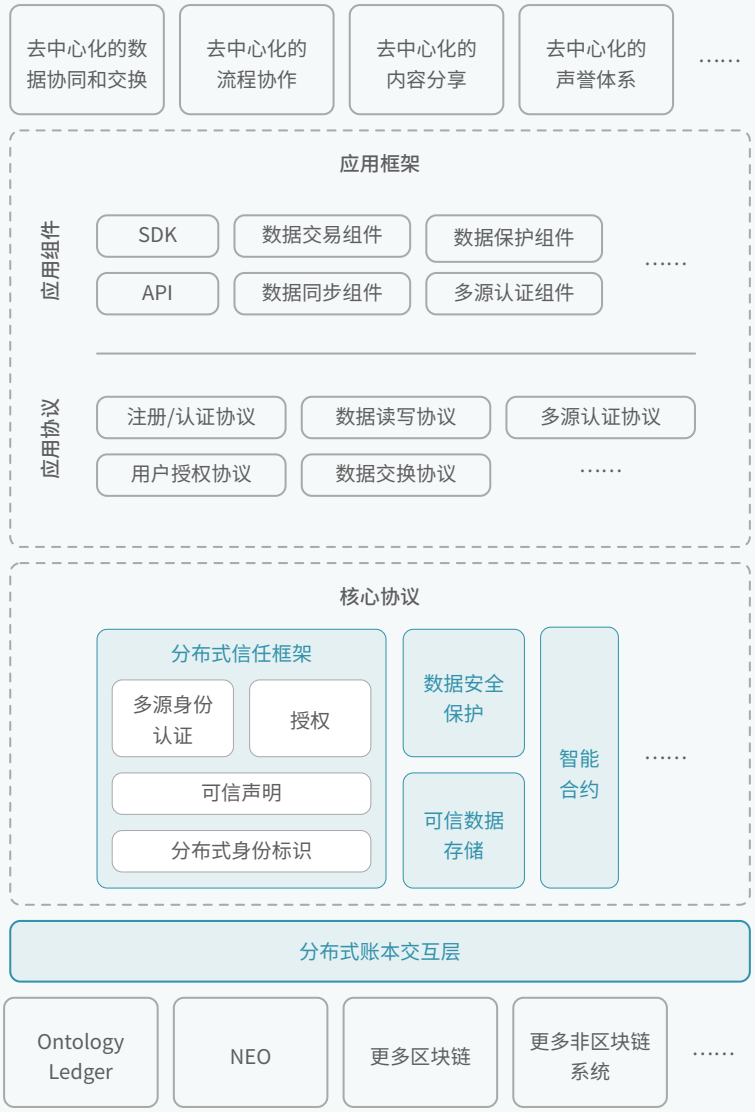


图 1.1 本体技术架构

© 2017 ONT Team

本体底层支持完整的分布式账本体系，包括核心分布式账本、智能合约体系、安全体系。分布式账本是本体底层存储的重要基础设施。分布式账本技术的去中心化、共同维护、不可篡改等特性是本体实现分布式多方信任的关键。分布式账本包括共识、智能合约体系在内的实现，并为分布式信任框架、上层应用提供共识、存储和智能合约支持。本体和分布式账本技术采用解耦设计，默认使用核心账本，也可以支持NEO、以太坊等其他区块链作为底层。在账本层，我们创造性提出共享数据合约模型，将数据存储和业务逻辑解藕，由不同的智能合约来实现不同的业务逻辑，使整个架构具备更好的伸缩性及灵活性。

分布式信任框架是本体实现分布式信任的核心逻辑层，我们通过分布式身份标识ONT ID 来连接人、财、物、事，ONT ID具有去中心化、自主管理、隐私保护、安全易用等特点。信任框架通过可验证声明[1][2]建立分布式信任模型和分布式信任传递体系，同时引入C-L签名算法、西格玛协议等密码学技术来实现可验证声明的隐私保护。

在本体中，提出包括身份标识协议、多维实体认证协议、用户授权协议、分布式数据交换协议等一系列的协议标准。各类协议的实现都兼容了国内外主要的协议标准和体系，如身份标识协议全面兼容W3C的DID[3]方案；数字签名协议同时支持国密标准、RSA、ECDSA等算法；在分布式数据交换体系中，兼容通用授权协议OAuth[4]、UMA[5]等，既使得架构满足开放性和标准性，亦可支持后续更广泛的生态合作与拓展。

对于应用服务提供来说，本体会做好“最后一公里”的支持，应用开发者无需具备底层的分布式系统开发能力，就可以直接基于本体提供分布式服务。简而言之，本体提供一系列应用框架，包括API、SDK以及各种应用功能组件，让各行各业的应用服务提供方开发自己的dApp，做到dApp as a Service(DAAS)，让区块链真正的好用起来。

除此以外，本体还包括各种先进的功能组件：密码学技术和数据保护组件、数据交易市场、全局事务数据库、混合预言机、可扩展的共识引擎……未来，本体还将联合开发者社区、业务合作伙伴，不断丰富功能组件，推动本体生态的技术进步。

© 2017 ONT Team

2. 术语说明

本体链群

也称为本体链网，是由多个不同领域、不同地区的链形成链网结构，共同构成整个本体。每条链使用独立的分布式账本，通过交互协议进行协作。

本体分布式账本

由本体的分布式账本/区块链框架构建的一个或多个核心公共服务基础链，为本体中的各项服务提供基础性的分布式账本、智能合约体系等服务。

实体

参与进行交互行为的个体，在本体中以ONT ID作为身份标识。

本体身份标识

ONT ID是一个去中心化的分布式标识协议，用于本体上对人、财、物、事的身份关联，具有去中心化、自主管理、隐私保护、安全易用等特点。

可验证声明

一个实体对另一个实体（包括自己）的某些属性作出的描述性声明，并附加自己的数字签名，用以证明这些属性的真实性，可被其他实体验证。

分布式信任框架

本体实现分布式信任的核心逻辑层，主要包含分布式身份标识协议、分布式信任模型及分布式信任传递体系等部分。

多源认证

指多个不同的认证方从不同的角度、不同的方面对同一实体进行多维认证。

信任锚

被一定的实体群体所信任的实体，作为一些信任传递链的源头，为本体提供基础身份认证服务。

分布式一致性账本

一种增量修改式的数据存储结构，由去中心化的点对点网络中的节点共同维护，具有数据公开且历史数据难以篡改的特点，为本体提供可信存储及智能合约支持。

© 2017 ONT Team

共识

账本节点按照特定的协议确认写入账本的数据，以保证账本的一致性。

智能合约

记录在账本中的可执行代码，通过账本节点上运行的智能合约引擎执行，每次执行的输入输出可记录在账本中。

本体应用框架

本体应用框架是应用组件、协议、SDK和API的统称，方便第三方应用dApp进行快速、低成本的接入。

预言机

是一种向区块链提供可信外部数据的服务。借助预言机，用户可以预测区块链系统以外事件的结果并永久记录在区块链上成为事实。

© 2017 ONT Team

3. 链网结构

本体的目标是建立起现实社会和分布式数字体系的连接桥梁，由于现实社会中业务的多样性、复杂性和特殊性，出于对性能、可扩展性和业务适用性的考量，仅使用一条公有链/联盟链难以支撑所有的应用场景。实际应用中是由不同的链来运行不同的业务逻辑，以不同的准入方式以及不同的治理模式，来满足不同场景的需求。同时，很多应用在现实当中并不是独立存在的，而是需要与其他应用进行多样化的交互协作。因此在这些不同的链之间，又需要具备多种交互协议，以支持应用间的流程协作。

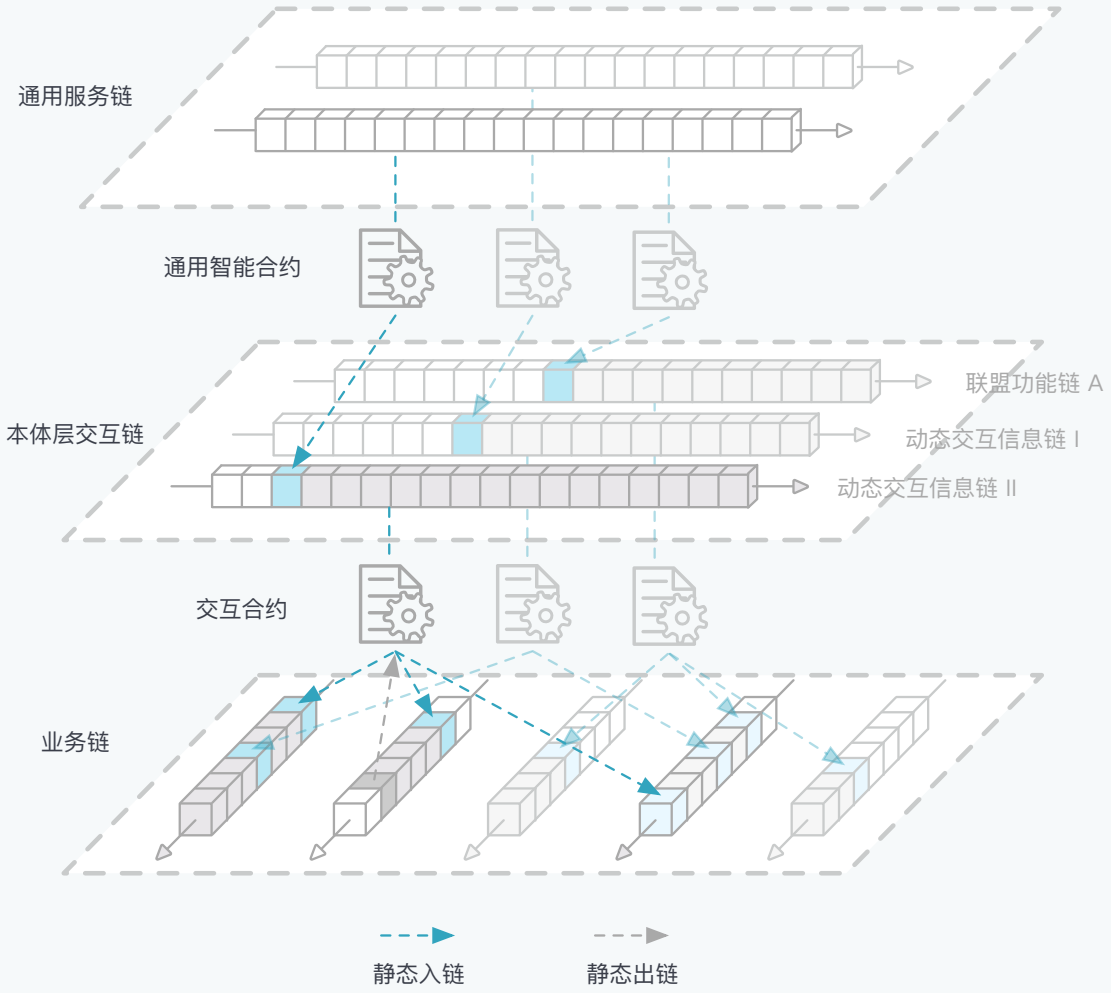


图 3.1 超融合链网结构

© 2017 ONT Team

基于以上需求和模式，本体提出了一个矩阵式立体网格架构——超融合链网结构。在横向的领域，可以有一条或多条提供基础性通用服务的公有链，如实体映射，或进行数据交换通用协议支持，以及提供通用性智能合约服务体系的多个公有性服务链。在一条或多条公用服务链的基础上，各个行业、地域和不同的业务场景，可以有自己独有的业务链，以满足不同场景下的准入要求、合规要求、治理要求及共识要求等。这些业务链，又可以使用公有服务链提供的基础性服务，如实体认证、数据交换协议等，也可以通过公有链在一些行业共性流程上进行协作。

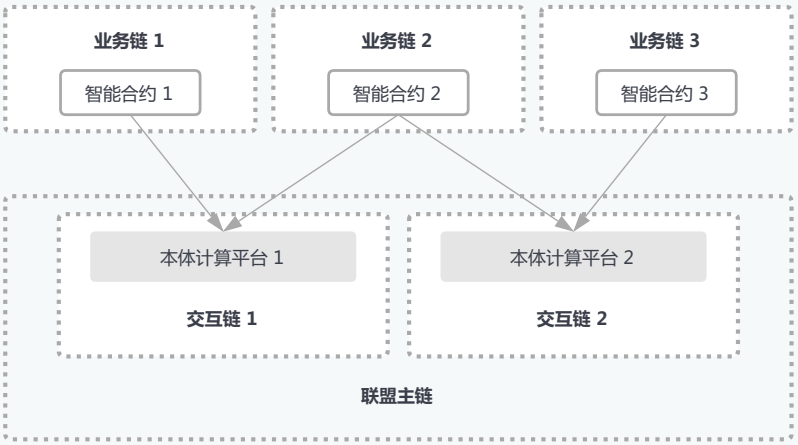


图 3.2 业务链通过本体计算平台进行跨链信息交互

除了与通用性的公有服务链交互，业务链还会与很多行业或业务性质相关的链之间进行协作。不同的协作场景会涉及不同的业务链，或者业务链上的不同业务点。因此可以有一些小型的专用公有/联盟业务服务链，按照特定的业务规则，协同一条或多条业务链或业务点。这样在纵向的领域，存在多条业务协同链，对某些特定跨链业务的智能合约、业务逻辑服务等功能提供专项协同支持。

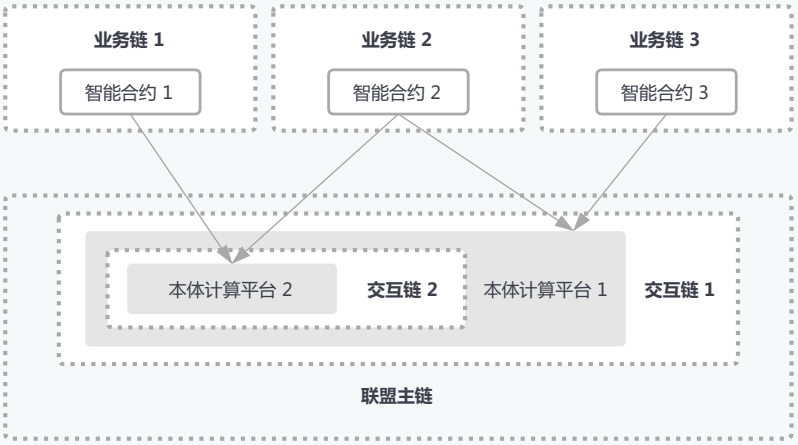


图 3.3 在本体计算平台基础上构建专用交互链，满足特定的协作需求

© 2017 ONT Team

这样的矩阵式网格架构，可以形成一张具有弹性的网，成为一个真正自治运作的下一代互联网。不同的业务场景都将以各自的展现方式在其中寻找到合适的服务模式，并且可以很方便地进行大范围、跨领域的协作。

本体中的各类协议，并不是一成不变的，会根据不同的业务场景、行业特点、监管要求及治理要求，选择和发展不同的交互协议。因此在本体当中，协议是一个不断发展的过程，但一个大的原则是：尽最大可能兼容和采用现有的各类协议与标准，在同一场景下，也尽可能多地支持更多不同协议，以使本体具有更好的兼容性和扩展性。

本体会基于本体分布式账本框架，实现一条或多条满足不同场景的可配置区块链。同时，本体的分布式账本框架还可以用于定制特定业务场景下的业务链（如不同的准入机制、治理机制、共识机制、存储模式等）。此外，本体也可以通过交互协议与现有的其他区块链体系进行协同。传统的IT系统在支持了特定协议后也可与本体对接。

© 2017 ONT Team

4. 分布式信任框架

4.1. 本体身份标识协议

实体是指现实世界中的个人、组织（组织机构、企事业单位等）、物品（手机、汽车、IOT设备等）、内容（文章、版权等），而身份是指实体在网络上的对应标识。本体使用本体身份标识（ONT ID）来标识和管理实体的网络身份。在本体上，一个实体可以对应到多个身份标识，且多个身份标识之间没有任何关联。

ONT ID是一个去中心化的身份标识协议，ONT ID具有去中心化、自主管理、隐私保护、安全易用等特点。每一个ONT ID都会对应到一个ONT ID描述对象（ONT DDO），用于记录ONT ID的控制人公钥等属性信息。描述对象作为公开信息以储存于本体底层的分布式账本中。出于隐私保护的考虑，描述对象默认不包含任何实体真实身份相关的信息。

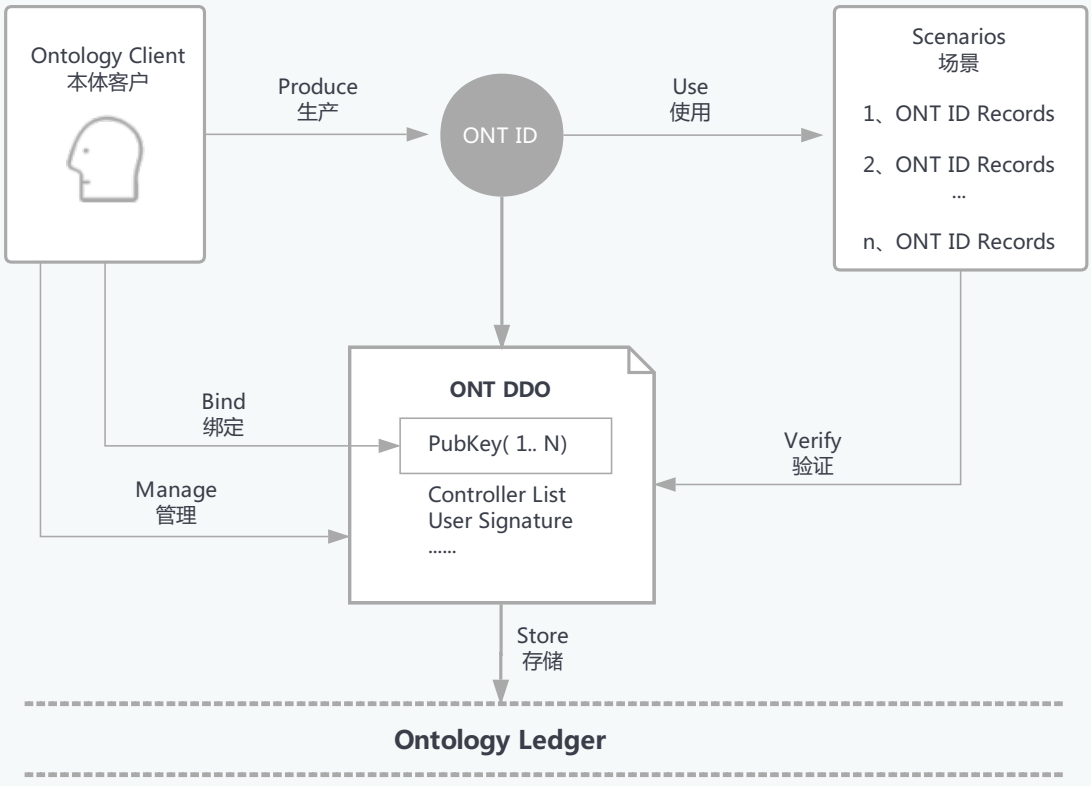


图 4.1 ONT身份标识体系

© 2017 ONT Team

4.1.1. 生成ONT ID

ONT ID是一种URI[6]，由每个实体自己生成。其生成算法保证了两个ONT ID重复的概率极小（$\approx \frac{1}{2^{160}}$），同时在向本体注册时，共识节点会检查该ID是否已被注册。

4.1.2. 自主管理

本体利用数字签名技术保障实体对自己身份标识的管理权。ONT ID在注册时即与实体的公钥绑定，从而表明其所有权。对ONT ID的使用及其属性的修改需要提供所有者的数字签名。实体可以自主决定ONT ID的使用范围，设置ONT ID绑定的密钥，以及管理ONT ID的属性。

4.1.3. 多密钥绑定

本体支持多种国内、国际标准化的数字签名算法，如RSA、ECDSA、SM2等。ONT ID绑定的密钥需指定所使用的算法，同时一个ONT ID可以绑定多个不同的密钥，以满足实体在不同的应用场景的使用需求。

4.1.4. 授权控制

ONT ID 的所有者可以授权其他ONT ID代替本人行使对 ONT ID 的管理权，如修改 ONT ID 对应的属性信息，在密钥丢失时授权其它ONT ID重设密钥等。ONT ID支持针对每条属性项的细粒度的权限管理，以及“与”、“或”、“m/n”等多种的访问控制策略。

4.2. 信任模型

信任模型是实体之间产生信任的机制，主要分为中心化的信任模型和去中心化的信任模型。本体同时支持这两种信任模型，可以根据具体场景选用不同的信任模型适应不同的需要。

4.2.1. 中心化的信任模型

此模型下，由一个或一群特定的实体作为信任锚，实体间的信任关系基于信任锚建立。信任锚指定其所信任的实体，被指定的实体又可指定其信任的其他实体。如此，以一个信任锚为源头，构

© 2017 ONT Team

建起一个信任传递的关系树。树上的每个节点有一条通向信任锚的路径，即为其信任链。任何承认该信任锚的实体在与树中的节点交互时，通过验证其信任链即可判断其是否可信。

目前最成熟、应用最广的PKI体系[7]就是一种中心化的信任模型。用户首先向作为信任锚的认证中心申请一个数字证书。认证中心对申请者审查通过后，将其身份信息和公钥写入数字证书，并附加认证中心的数字签名。通过认证中心签发的数字证书即认证了用户的身份和公钥的绑定关系，任何人都可以使用认证中心的公钥来验证该证书的真伪，进而使用证书中的公钥验证用户的签名，确认其身份。同时拥有数字证书的用户还可以申请作为下级认证者对其它用户颁发数字证书，其颁发的数字证书的效力最终由认证中心保证。在PKI模型中，任何参与方都必须无条件的信任认证中心，信任关系通过实体间的数字签名从认证中心逐层传递下去。

中心化的信任模型有很多优点，其严谨的信任传递方式、清晰明确的信任与非信任边界在很多场景下都是很好的特性，解决了现实中的很多问题。然而中心化的信任模式也存在一定的缺陷，除了需要无条件信任锚，对其诚实性及安全性由较高的要求以外，对于现实世界的复杂信任关系，这种依赖于中心节点的模式会严重限制应用的灵活性。

4.2.2. 去中心化的信任模型

除了依赖特定的中心实体构建信任关系以外，实体之间还能够自发、对等地产生信任关系。信任的传递由实体间的相互认证实现。一个实体被数量越多的实体认证，其可信度就越高；被可信度越高的实体认证的实体，亦将获得更高的可信度。

去中心化的信任模型是一种模糊的信任模型，并没有统一、清新的信任边界，在不同的场景下可以根据不同的信任评估方法评估实体的信任度。正是由于这种高度的灵活性，在现实生活中具有广泛的用途。

4.3. 可验证声明

可验证声明用来证明实体的某些属性。声明以JSON-LD[8]形式格式化，并按照LD-Signature[9]规范添加数字签名。它可以作为一个数据单元进行存储和传输，并可被任何实体验证。可验证声明中主要包括元数据、声明内容及声明者的签名，其中声明内容可以是任意数据。

© 2017 ONT Team

4.3.1. 生命周期

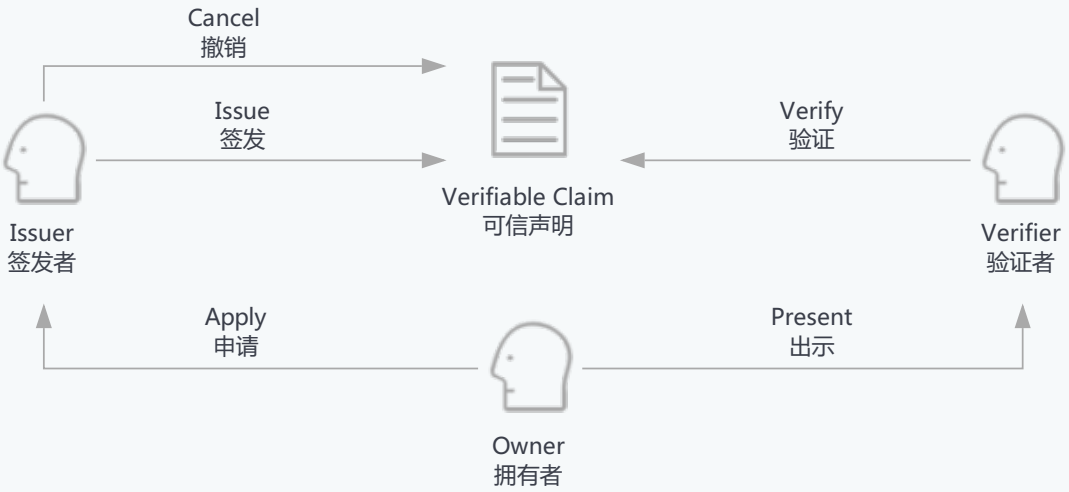


图 4.2 可验证声明的生命周期

与可验证声明相关的实体分为三种角色：签发者、持有者以及验证者。可验证声明的生命周期包括以下五种操作：

- 签发：任何实体都可以对其他实体的任何属性签发可验证声明，如学校可以给学生就一份成绩单签发一张可验证声明；银行可以就客户的资产状况签发一张可验证声明等。签发可验证声明时，可以设定声明的有效期，到期后声明自动作废。
- 存储：可验证声明可以分为公开声明和隐私声明。公开声明可以存储于本体的分布式账本中；隐私声明通常存储于实体的客户端中，由实体自己管理。
- 出示：可验证声明所有者可以自主决定向谁公开声明，并且声明所有者可以选择出示部分声明内容或者合并出示多张声明的部分内容而不影响对出示声明的验证。
- 验证：对可验证声明的验证不需要与声明的签发方交互，只需要根据证书签发者的ONT ID从本体的分布式账本中获取其公钥信息，进而使用该公钥验证声明中的数字签名，即可以认证该声明的有效性。
- 撤销：可验证声明的签发者可以在声明到达有效期之前撤销声明，被撤销的声明将无法通过验证方的验证。

© 2017 ONT Team

4.3.2. 匿名声明

通常情况下，声明所有者出示声明的时候会向验证者暴露声明的全部内容。在有些场景下，声明所有者希望既不向验证者暴露具体的声明内容，而又希望通过验证者的验证。针对于这种情况，本体使用匿名声明技术来保护用户的隐私。

匿名声明技术解决的问题，就是如何在签发及出示声明的过程中隐藏持有者的信息。利用匿名签发声明协议，同一个实体从两个不同签发者获取两个声明，即使这两个签发者共谋也无法确认是否是将声明给了同一个实体。在匿名声明展示阶段，持有者不需要提供原始声明给验证者，仅需要提供一个零知识证明。验证者结合签发者的公钥和该证明，以及对证书中所包含属性值的一个断言（如 “年龄>18” 且 “籍贯为上海”），通过运行一个验证算法，可以验证声明的真伪。

一个匿名声明通常是一个XML或JSON对象，包含公开信息和密码学信息。公开信息包括匿名声明所涵括的所有属性，每个属性包括三个部分：属性名、属性类型、属性值。属性支持多种数据类型，如字符串、整数、日期、枚举类型。密码学信息主要包括持有者自己的主密钥，签发者对公开信息的数字签名。

在出示匿名声明的过程中，持有者向第三方的验证者证明他拥有一张由某签发者签发的匿名声明，并且可以选择性地公开部分属性值，隐藏其他属性值。除此之外，还可以证明某些隐藏属性满足某些逻辑断言。

匿名声明使用C-L签名算法[10]和Σ协议[11]实现以上特性。

4.3.2.1. C-L签名算法

C-L签名算法包括三个部分，分别是签名者的密钥生成算法，签名生成算法，签名验证算法。该算法的安全性是基于强RSA假设的。

密钥生成GEN:

1) 随机两个安全素数(p,q)，并且$n=pq$;
2) 随机$k+2$个二次剩余$(R_1,\ldots,R_k,S,Z)$;
3) 输出私钥$sk=(p,q)$， 公钥$pk=(n,R_1,\ldots,R_k,S,Z)$。

签名生成$SIGN_{sk}(\{m_i\})$：输入k个值$\{m_1,\ldots,m_k\}$

1) 计算n的欧拉函数$\varphi(n)=\varphi(p)\varphi(q)=(p-1)(q-1)$;

© 2017 ONT Team

2) 随机足够大的素数e和整数v;

3) 计算e关于$\varphi(n)$的模反元素e^{-1}，即有$e \cdot e^{-1} \equiv 1 \pmod{\varphi(n)}$;

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

5) 输出签名(A, e, v)。

签名验证$VERIFY_{pk}(\{m_i\}, A, e, v)$：输入 k 个值$\{m_1, \ldots, m_k\}$，签名(A, e, v)

1) 验证e, v是否足够大，且e是一个素数;

2) 验证签名是否满足

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

4.3.2.2. Σ协议

Σ协议(Σ-protocols)是一种高效的零知识证明协议。利用此类协议，证明者可以在不向验证者暴露秘密内容的条件下向其证明他知道该秘密。著名的Σ协议有Fiat-Shamir启发式[12]和Schnorr签名算法[13]等。

假设证明者知道方程组$y_1 = g^x,\ y_2 = h^x$的解，对其的证明可以表示为

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

该证明可用如下交互式协议完成：

1) 证明者生成随机数r;

2) 证明者计算(g^r, h^r)并发送给验证者;

3) 验证者生成随机数c并发送给证明者;

4) 证明者计算$s = r - c \cdot x$，将s发送给验证者;

5) 验证者检查以下等式是否成立

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

4.3.2.3. 签发匿名声明

声明签发者与接收者进行两次交互完成匿名声明的签发：

1) 签发者生成随机数n_1，并发送给接收者;

© 2017 ONT Team

2) 接收者生成随机数v'，对主密钥m_0计算$U = R_1^{m_0} S^{v'} \pmod n$，并将$U$发送给签发者；

3) 签发者生成随机素数e和整数v''；

4) 签发者对属性$\{m_i\}$和U计算C-L签名(A, e, v'')，发送给接收者；

5) 接收者计算$v = v' + v''$，得到匿名证书(A, e, v)。

通过这个协议，接收者可以获取到签发者对公开属性的C-L签名，再加上自己的主密钥m_0，即匿名声明的密码学信息部分。

4.3.2.4. 出示与验证匿名声明

匿名声明的持有者向验证者公开一个或多个声明的部分属性，而隐藏其他属性，并以此证明一些断言的正确性。

当持有者使用匿名声明构造证明时，首先需要证明其知道声明中的隐藏属性。假设共k个属性$\{m_1, \ldots, m_k\}$中隐藏了l个属性$\mathbb{H}_l$，证明的构造方法如下:

1) 生成随机数r_A，将声明中的C-L签名随机化：$A' = A \cdot S^{r_A}, v' = v - e \cdot r_A$

2) 按照3.3.2.2节所述协议构造零知识证明：

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

3) 输出证明π。

为了证明某断言，需对匿名声明中的属性构造不等式$m \geq b$的证明，方法如下：

1) 计算差值$\Delta = m - b$，并将其表示成四个整数的平方和：

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

2) 生成随机数，将这四个整数分别隐藏在T_{u_i}中，并构造如下证明：

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

3) 生成随机数，将差值 Δ 隐藏在T_Δ，并构造如下证明：

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

4) 证明属性值确实是大于等于b：

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

5) 输出证明(π_1, π_2, π_3)。

证明π及若干三元组证明(π_1, π_2, π_3)即构成对该断言的完整证明。验证者按照4.3.2.2节所述协议验证每个子证明，若全部通过即可确认该断言为真。

© 2017 ONT Team

5. 分布式账本技术

5.1. 分布式账本

5.1.1. 共识算法

核心账本支持新一代共识引擎——Ontorand Consensus Engine (OCE)。OCE 是一个高效的、基于 dBFT 共识协议和可验证随机函数VRF的增强版本共识引擎，实现了近乎无限的可扩展性，它只需要很少的计算量开销，生成几乎不会分叉的区块链网络。dBFT 已经在NEO公用链和很多联盟链项目中经过长时间的运行表现出优秀的稳定性与可靠性。而 OCE 作为支持全球规模的高性能共识协议，其生成区块的速度仅受制于网络速度，常规区块的确认时间小于10秒。OCE 基于可验证随机函数VRF的随机抽签机制来选择验证者集合，每个验证者集合通过 dBFT 投票选取记账人，并结合拜占庭容错算法[14][15][16]的方式来达成共识。同时由验证者集合的群签名来创造本体的种子seed，指向下一个验证者集合。OCE 支持可插拔验证者、在线协议修复/升级，是未来本体的核心共识机制之一。

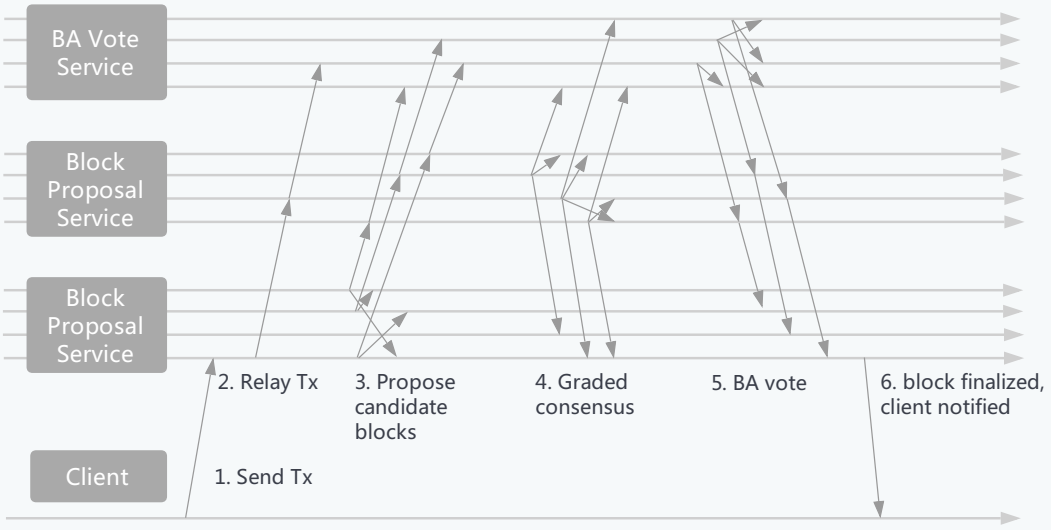


图 5.1 OCE执行过程

© 2017 ONT Team

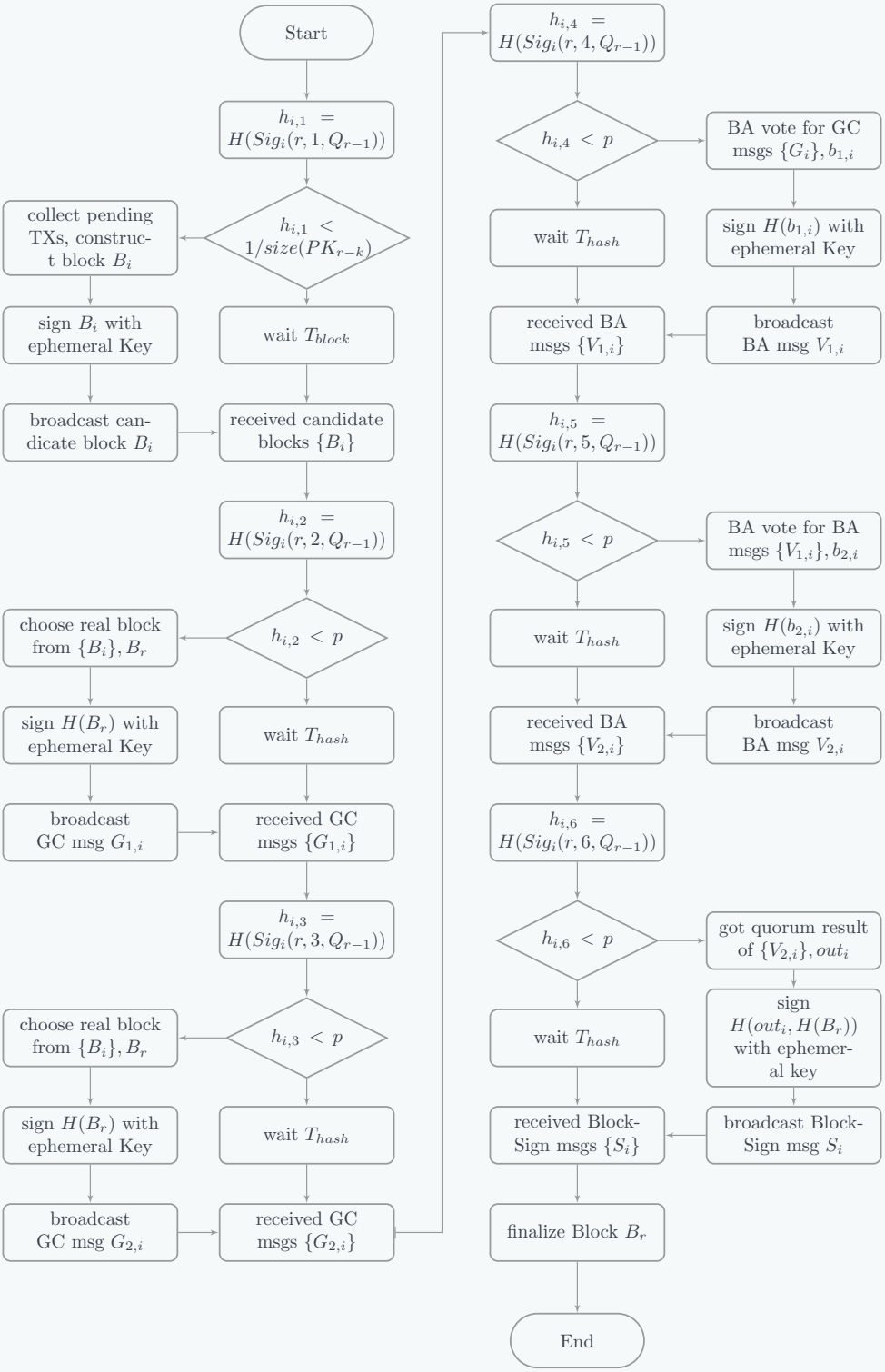


图 5.2 OCE主流程

© 2017 ONT Team

OCE的执行过程及其基本的算法流程如图5.1和图5.2所示。

1) 客户端通过P2P网络广播交易数据请求，并附上交易发起者的签名；
2) 网络中所有参与共识节点在P2P网络中独立监听全网交易，并缓存在本地内存中；
3) 提议新的区块：
   a) 节点对当前区块高度和上一区块中的Q值做签名，然后计算哈希，根据哈希值独立判断自身能否成为当前高度区块的潜在proposer；
   b) 如果节点判断自己为当前区块高度的proposer：
      i) 打包当前以及收集到的未共识的交易请求，构建新的区块；
      ii) 并对此区块数据和区块哈希进行签名，将区块和签名向P2P网络中广播；
4) 所有节点将等待T_{block}时间，独立监听网络中的区块proposal，并将收到的区块proposal缓存在本地内存中；
5) 新区块的分级共识：
   a) T_{block}后，网络中节点独立判断自身是否为当前高度各个区块proposal的验证节点；
   b) 如果节点判断自己为当前区块proposal的验证节点：
      i) 验证所有区块proposal的正确性和完整性；
      ii) 验证所有区块proposal中的Q值签名的正确性；
      iii) 验证区块proposer的身份的合法性；
      iv) 对所有proposal的Q值签名计算哈希，选取哈希最小者的区块proposal作为当前节点所选择的真正区块；
      v) 对所选择的区块proposal的哈希进行签名，将区块哈希和签名向P2P网络中广播；
   c) T_{hash}后，节点更新算法step，独立判断自身是否应该在此步骤中参与对区块proposal的验证；
   d) 如果节点判断自己在当前step为区块proposal的验证节点：
      i) 同样执行上述的区块proposal验证工作；
      ii) 同样对所选block proposal的哈希进行签名，将区块哈希和签名向P2P网络中广播；
6) 网络中所有节点都将在P2P网络中独立监听区块验证节点所广播的共识消息：

© 2017 ONT Team

a) 验证共识消息的有效性和完整性；

b) 验证发出共识消息的节点身份的合法性；

c) 将验证消息缓存在本地内存中；

7) 对新区块的拜占庭投票：

a) T_{hash}后，节点更新算法step，独立判断自身是否在此步骤中参与对新区块的投票；

b) 如果节点判断自己应当参与此轮投票：

i) 计算不同block proposal所收到的共识验证的票数；

ii) 按照quorum对共识的结果进行投票；

iii) 对投票结果进行签名，并将所投票的区块哈希，投票结果及其签名向P2P网络中广播；

c) 网络中的所有节点将按照区块链的配置，对区块的投票结果再进行多轮投票，每轮投票都将在T_{hash}后，网络中所有节点独立完成；

8) 网络中所有节点也在P2P网络中持续监听广播的投票消息，并对投票消息进行有效验证；

9) 对新区块的最终签名：

a) 完成最后一轮拜占庭投票后，节点更新算法step，判断自身是否参与对新区块的最终签名；

b) 如果节点判断自己应当参与区块签名：

i) 根据收集到的投票结果，按照quorum判断所投票的block proposal是否共识；

ii) 如果判断没有达到quorum，将此轮新的区块共识结果设置为空块；

iii) 将自身做出判断的投票消息进行打包，并进行签名；

iv) 将最终的区块哈希和自身对结果的签名向P2P网络中广播；

10) 网络中所有节点在P2P网络中持续监听区块最终签名消息，并对区块签名消息进行有效验证。

a) T_{hash}后，将根据所收到的区块签名消息，得到当前高度的区块哈希；

b) 根据区块哈希，和之前在网络中收集到的区块proposal消息相比较，得到当前高度共识的最终区块。

同时，根据共识的最终区块计算出区块的Q值，从而开始下一个高度的区块的共识。

© 2017 ONT Team

核心账本采用模块化设计，支持可插拔共识算法，根据具体的应用场景/类型可以方便、快速地切换成其它的共识算法，包括PoS、DPoS、Raft算法。

5.1.2. 流程协同

运用分布式账本技术，实体跨链及跨系统隐私，以及特定的跨链协议，实现流程协同（分布式事务）。即流程/事务的多个步骤分散在不同的区块链或系统上执行，保障不同实体在不同系统与区块链中的身份隐私，且保证整个事务的一致性。

5.1.3. 存证设计

通过分布式账本的运用，将不止提供数据存证，还可对行为存证进行支持。即每一次的数据请求、数据匹配、数据调取与数据使用等均在账本进行记录，形成了一份数据全流程的记录，保障数据的安全、可靠、且不被泄露。

5.2. 智能合约

核心账本使用go语言版NeoVM虚拟机作为智能合约的执行环境，可以为本体应用层框架实现智能控制逻辑。NeoVM虚拟机具备图灵完备性，可以实现任意逻辑，同时具有高度的确定性，可保障在技术上实现“输入一样，输出一样”，避免了container/docker的不确定性。非常适合用于对确定性要求很高的场景中。此外NeoVM还具备高扩展性，通过“确定性调用树”技术，可以实现动态分片，从而达到理论上的无限扩展能力。

此外，NeoVM将Java Bytecode、C# MSIL等中间语言编译为区块链虚拟机的指令。从而使智能合约的开发者不需要学习新的语言，就能使用熟悉的Java、C#、Python、Javascript等编程语言编写智能合约，快速覆盖并融入全球百万级的开发者社区。共享平台各参与方（特别是各信息提供者）将容易上手操作和维护，为其提供进行智能合约编辑的可操作性，降低了其接入门槛。

NeoVM虚拟机与上层高级语言解析转换相结合，灵活支持虚拟机的基础应用。通过定制化的API操作实现虚拟机的外置接口，可以灵活地与账本数据以及外部数据进行交互。这一机制实现了智能合约运行时达到原生代码执行的高性能。同时也实现了支持不同区块链的通用虚拟机机制。

© 2017 ONT Team

5.3. 共享数据合约模型

上层应用对于分布式账本的需求主要包括以下两个功能：数据结构的定义及存储、业务逻辑的处理及与外部系统的交互。为了通过解耦使之具备更好的伸缩性及灵活性，我们设计了一种模型将这两部分分开，由不同的智能合约来实现，一部分负责数据存储（数据合约）；另一部分负责处理业务逻辑（控制器合约）。通过使用多个不同的控制器合约来共享同一份数据合约的架构我们称之为共享数据合约模型。

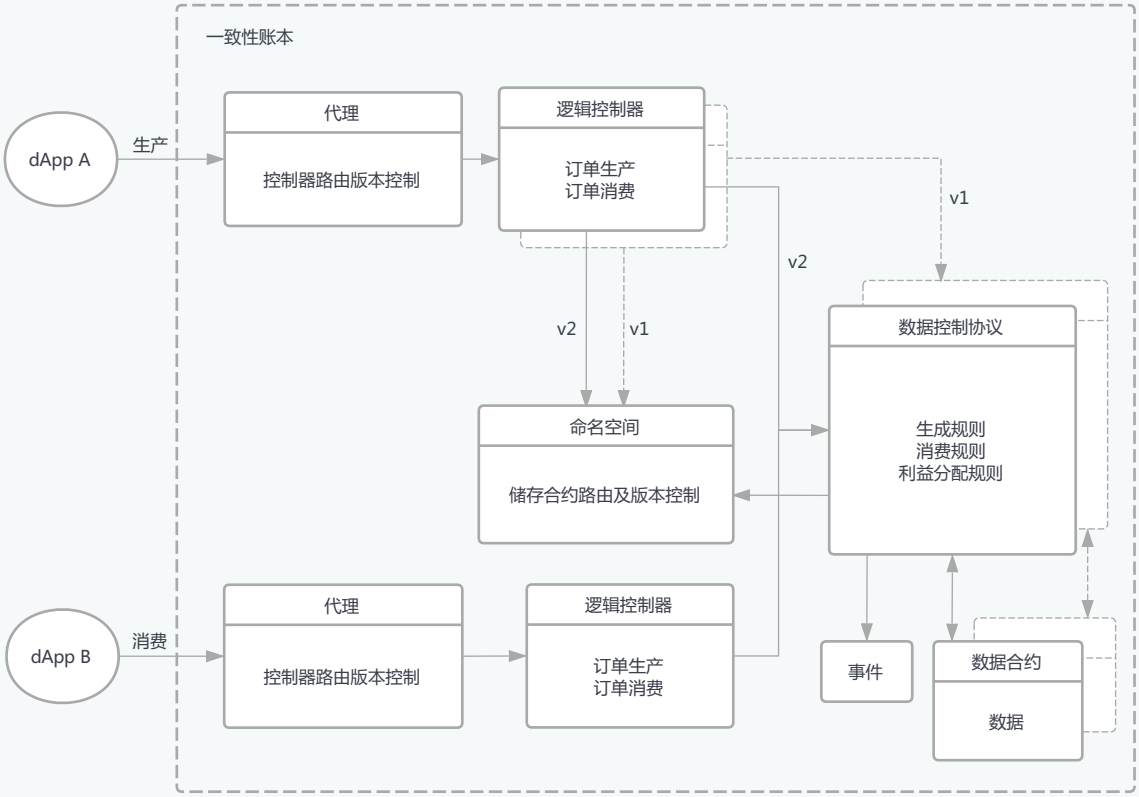


图 5.3 共享数据合约模型

该模型包括以下设计功能组件：

- 代理控制器：面向dApp，提供外部dApp确定的合约入口，保证即使合约升级，也不会引起外部调用的不一致性；
- 逻辑控制器：业务逻辑控制；

© 2017 ONT Team

- 命名空间：支持在不同dApp不同版本下，数据合约地址的映射。确保数据结构的升级不影响业务逻辑，并且能支持不同版本间的数据回溯；
- 数据合约： 提供基本的数据结构及存储接口。类似DAO，提供GET/SET方法。
- 数据控制协议：由各业务关联方制定的通用业务规则，数字化为协议控制器。这个协议控制器包括以下部分：
  - 连接该逻辑控制器的权限控制；
  - 与外部系统的交互接口；
  - 通用的业务逻辑；
  - 通用业务核心账本的控制逻辑；
  - 事件通知机制；
- 消息事件：是在智能合约的执行过程中，把一些需要广播的内容推送出来的一种机制，各方在同步账本的过程中，均能通过本地重放分布式账本区块的方式获得当前合约的交易消息。消息实际上也是分布式账本的一种廉价存储机制，不关心这个合约的节点可以选择不执行这个合约，当然也就不会收到这个消息事件了。而关心这个合约的节点可以通过合约的执行，获得跟此合约相关的事件。因为链上不存储实际的交易内容，这样就可以极大地减少各节点的存储压力。

这种设计模式意味着即使上层业务不同，只要基于通用的业务协议，就可以实现数据共享，优势互补，让跨界协作也变得更加简单。

以数据交易所dApp为例。如图5.4所示，业务逻辑包括订单登记和消费（交易）两个流程。

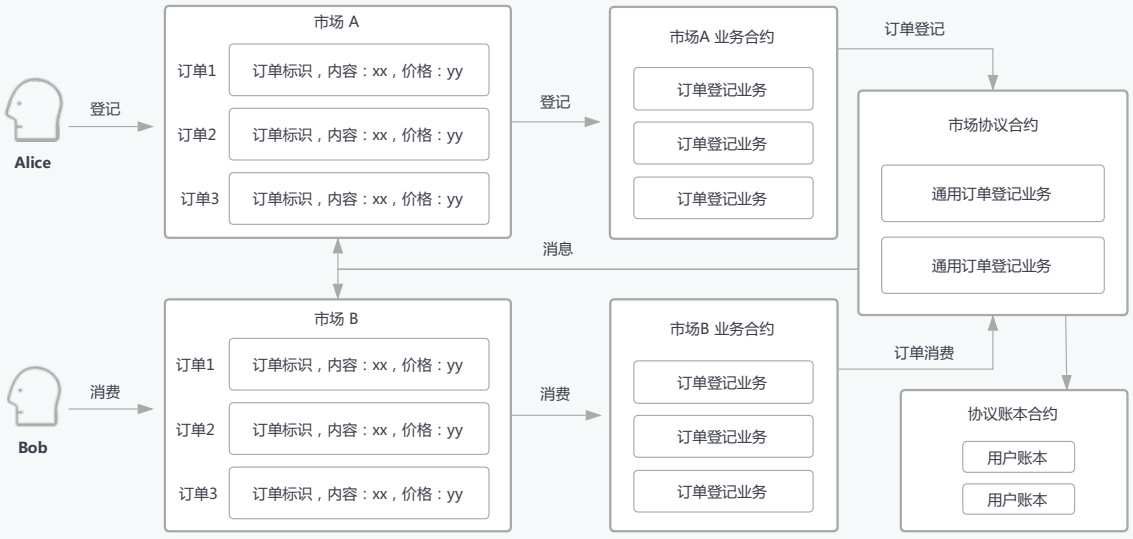


图5.4 数据交易所业务流程

© 2017 ONT Team

登记流程：

1) Alice向市场A发起订单登记请求；
2) 市场A检查完Alice的订单之后，按照分类，往该分类协议合约登记订单；
3) 市场协议合约检查市场A的权限及其提交的订单内容，满足规则则将订单登记到该协议核心账本；
4) 登记成功以后，协议合约广播订单登记消息，并返回结果；
5) 市场A业务合约向Alice返回最终结果。

交易流程：

1) 市场B通过同步区块获得所有交易；
2) 市场B通过执行区块中的交易获得市场A广播的订单登记消息；
3) 市场B通过订单登记消息中的订单信息在市场B中展示；
4) Bob向市场B发起交易行为，对象为上面Alice所登记的订单；
5) 市场B检查完Bob的交易请求后，按照分类，向该分类协议合约发起交易；
6) 协议合约检查市场B的权限及交易请求信息，满足规则则处理B的交易请求并做最终结算；
7) 协议市场广播订单交易完成信息，并向市场B返回处理结果；
8) 市场B向Bob返回最终结果；
9) 市场A监测到协议合约的交易完成广播信息，并向Alice推送交易完成信息。

5.4. 默克尔树存储模型

对于比特币，以太坊等数字资产平台，客户端通常只需要关注自己账户的信息，如果完整地同步所有的账本信息会造成效率低下。因而提出了SPV(Simple Payment Verification)的验证技术，通过构造默克尔证明[17]，客户端只需要同步区块的头信息，就可以达到验证的目的，极大地节省了存储空间，减轻终端用户和网络传输的负担。在本体中，客户端除了需要关注自己账户的信息，还需要大量地对其它ONT ID进行身份信息的获取和认证。因而账本的设计需要支持默克尔证明的构造。

© 2017 ONT Team

5.4.1. 默克尔哈希树

二叉默克尔哈希树[18][19]用于构造高效的审计证明，它的输入是一个数据项列表，这些数据项通过哈希运算得到的哈希值作为默克尔树的叶子节点。它的输出是树根节点的哈希值。 给定一个有n个输入的有序列表：$D[n] = (d_0, d_1, \ldots, d_{n-1})$，其对应的默克尔树哈希（MTH）定义如下：

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

其中 k 是小于 n 的最大2的幂 ，$D[a:b]$表示列表D的第 a 到 $b-1$ 个元素构成的子列表，||表示连接前后两个比特串。

5.4.2. 默克尔审计路径

一个叶子节点的默克尔审计路径是指默克尔树中长度最短的一个节点列表，通过这个列表可以算出这颗树的根哈希。由于树中的每个节点的值要么是叶子节点的哈希值，要么是该节点的两个子节点计算出来的哈希值。也就是说，审计路径是由从叶子节点计算到达根节点中缺少的节点构成的列表。如果通过该列表算出的哈希和根哈希相等，也就是证明了该叶子节点确实存在于该树中。

给定一个有n个输入的有序列表$D[n] = (d_0, d_1, \ldots, d_{n-1})$，对第$m+1$个输入$d(m) : 0 \leq m < n$，其对应的默克尔审计路径$PATH(m, D[n])$定义如下：

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

其中+表示连接前后两个列表。

图5.5是一个默克尔审计路径的示例：

© 2017 ONT Team

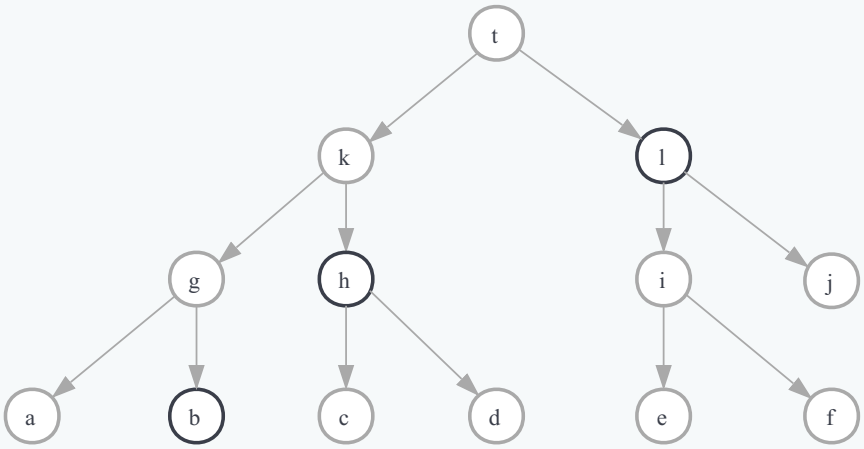


图5.5 a节点的审计路径为[b, h, l]

5.4.3. 默克尔一致性证明

在数据同步的过程中，往往需要验证对方节点的数据确实是由自己的数据的基础上附加得到的。构造默克尔一致性证明可以达到这个目的。对于默克尔树$MTH(D[n])$和该树的前m个叶节点构成的默克尔树的$MTH(D[0:m])$，其一致性证明是构造一个节点列表，证明两棵树的前m个叶子节点相同。下面的算法可以构造出唯一的最小节点数的一致性证明：

给定一个有n个输入的有序列表$D[n] = (d_0, d_1, \ldots, d_{n-1})$，对前$m$个叶节点的默克尔一致性证明$PROOF(m, D[n])$定义如下：

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

图5.6是一个默克尔一致性证明的示例：

© 2017 ONT Team

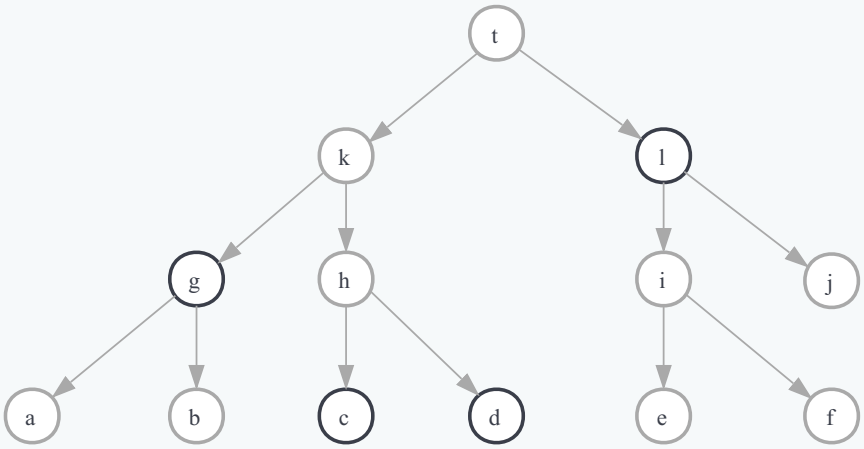


图5.6 $PROOF(3,D[7])$的证明为[c,d,g,l]

5.4.4. 默克尔-帕特里夏树

在本体的一些场景中，我们需要快速对某一主体在多个交易产生后的最终结果进行证明，比如证明某个实体的身份状态，如果使用默克尔证明，将需要对每个历史交易逐一进行证明，而使用默克尔-帕特里夏树(Merkle Patricia Tree，MPT)[20]，能够大大提升效率。

MPT是帕特里夏树[21]和默克尔树的结合，包含了键值的映射关系，提供了一个基于密码学的，自校验防篡改的数据结构，具有确定性、高效性和安全性的特点：

- 确定性：查找数据时，相同的键值，将查找到同样的结果，并且有相同的根哈希；
- 高效性：当数据发生改变时，能快速的计算出新的树根，无需重新计算整棵树，对数据的插入、查找和删除的时间复杂度控制在$O(\log_2 n)$；
- 安全性：当攻击者恶意制造大量交易，发起DOS攻击，试图操纵树的深度时，限定的树深将使攻击无法实现。

5.5. 混合预言机

混合预言机（HydraDAO）是融合智能合约、跨链、跨数据源协同的数据预测和数据交互组件。它包含本体的分布式自治组织（DAO）功能和链外的数据交互功能（大数据、人工智能）。本体的治理机制将支持民主提案和人工智能自动化提案/验证，通过算法自动提交的提案形成一个完整的HydraDAO作业，并自动接受有效的验证/投票。此过程将生成一个独特的DAO地址并创建一个代

© 2017 ONT Team

币投票池。它使DAO能够将资金和结果直接收入到本体中。一旦投票期完成，DAO将根据其智能合约中定义的不可篡改的规则自治执行。这种混合机制，为本体的数据交换和治理提供了极高的灵活性，为未来网络大规模自动化运行提供了技术上的支撑。

本体预言机智能合约的执行逻辑如图5.7所示：

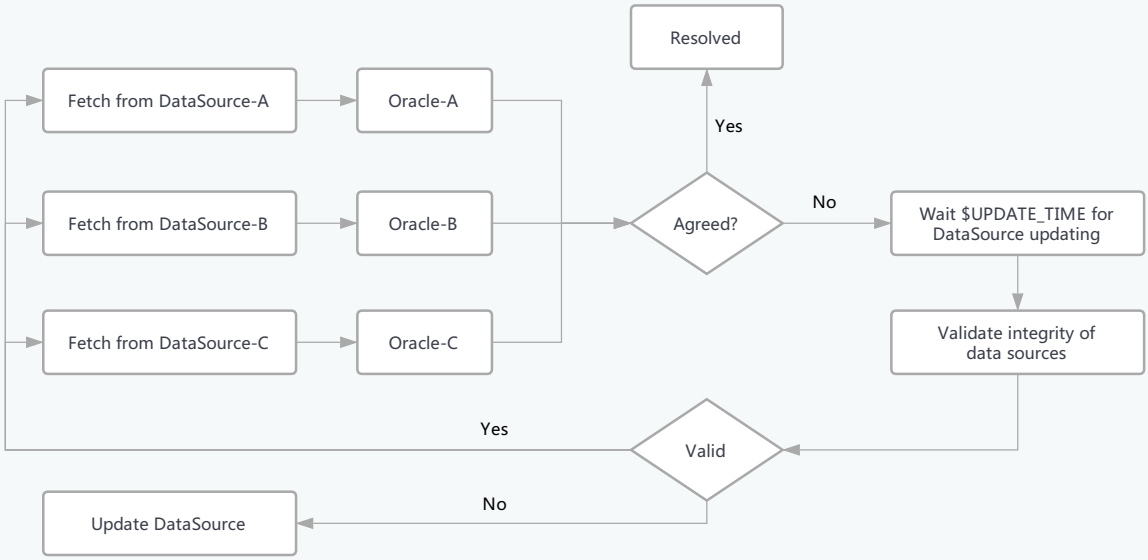


图5.7 混合预言机执行逻辑

1) 当任意交易执行开始/结束时，可以选择调用与交易相关的预言机智能合约，预言机智能合约及其参数必须在创建交易时设置，并作为交易的一部分存储在区块链上；交易结束后，自动执行合约逻辑确定哪些结果为最终事实，将存放于DAO合约地址的代币支付给提供了有效数据信息的参与者。本体通过分布式数据交易协议（Distributed Data Exchange Protocol，DDEP）面向外部提供API接入大数据实时计算能力与数据市场参与对交易结果的分析和预测，以验证其完整性。作为可信数据服务的一部分，交易参与者在参与交易之前可以绑定可信的数据源或者利用本体接入的服务来验证有效、准确的交易状态数据。

2) 对于一些只依赖区块链链上状态的交易，预言机智能合约获得授权后直接从区块链接口查询链上数据。

3) 对于大多数与现实社会关联的交易来说，预言机智能合约需要获得真实世界中交易结果的数据。当预言机智能合约需要访问/读取现实世界中的有关数据时，则必须遵循以下模式：创建交易时，每个预言机在多个可信数据源中随机选择1个或者指定阈值

© 2017 ONT Team

数量个的数据锚点；在交易结束时，经由指定的可信数据源验证并交互更新最终的全局状态[2]。

5.5.1. 内置的数据预测DAO

去中心化的魅力不仅是成本的降低、效率的提升，它更大的优势是“众智”，即分布式的群体决策。本体上执行的交易可以选择在合约执行时挂钩HydraDAO来激活“众智”。例如，用户希望在某个外汇价格窗口买入一定量外汇，但用户并不清楚何种挂单价格合适，他可发起DAO项目提案并抵押一定量的保证金，一旦提案得到足够数量的认可则进入预测期。任何接受提案的参与者都可附加一个事实和他的签名。当然，参与者可以随时修订他的事实，但是这需要支付额外的保证金。一旦预测期结束，DAO合约将自动锁定合约内的保证金，并根据提案的规则条款筛选出最优事实，预测最准确的参与者将获得奖励，用户发起的交易也会参照这个最优事实的输出结果自动执行。

提案一旦提交给DAO，新的提案将由全体网络的代币持有者进行评估和决策。在此期间，参与者可以提交事实并随时修改事实进行修正。修改事实可能也需要被投票和排名。本体的DAO激励机制将确保收到最多赞成票并最接近事实的提案获得最大奖励。

在预测期完成后，HydraDAO自动启动仲裁。一旦触发仲裁，HydraDAO将计算并排序所有的提交结果，将DAO合约的资金分配到奖励者的账户中。最后进入关闭阶段，HydraDAO将输出一个事实状态输出，提供给绑定的智能合约使用。

5.5.2. 外部可信数据源

通过预定义的可配置Oracle智能合约可以确保只有在智能合约验证条件满足后才会真正发生价值转移。同时，根据约定的合约规则履行情况可以避免纠纷。预言机智能合约可以支持任何的JSON API类型，便于条件定义和开发。

可信数据源的接入过程：

1) 选择数据源的位置，获取数据更新的频度、有效期、API证书、隐私条款等生成预言机智能合约的输入参数。
2) 定义智能合约的描述，便于其他验证签名者评估

2. 所有数据的使用需经由数据所有者或相关授权方的授权，并遵循有关隐私保护条款。

© 2017 ONT Team

描述信息可以使用简单易懂的术语来说明这个合约是做什么的。作为该合约的标签，便于区块链上的用户检索查看，提高透明度。详细的描述信息可以将具体的IF / THEN等触发条件以及最终完成交易的条件加以说明。

3) 关联现实世界的法律文件

预言机智能合约会使用合约发布者的电子签名来签署绑定的法律文件。通过声明并上传法律文本将有关证据信息固化在区块链上。

4) 通知合约的联合签署人

根据合约的条款和权益方，使用IM/电子邮件/电话等方式通知需要签名的参与者，合约只有在全部签名者在签名截止日期前都完成签名后才会生效。如果到了截止时间未完成签名流程，该合约将会失效。当参与者签署合约后，预言机智能合约会为该参与者分配资金托管地址。

5) 向预言机智能合约资金托管地址充值代币

用外部数据源会产生费用，需要将资金预先存入智能合约创建的地址，由合约来完全控制付款行为。

6) 完成合约发布

一旦合约签署和存款完成，它将在本体上自治运行。一旦某个相关的交易完成，智能合约会操作托管地址的资金支付到合约条款定义的账户，当然，未能满足条款的交易或者欺诈交易全部锁定的资金会自动退回托管账户，不会被恶意挪用。

7) （可选）数据开放平台，公开数据接入协议

对于一部分数据开放共享的平台，根据已经提供的API和数据使用协议，预言机智能合约可以直接与相关的off-chain数据进行交互。根据开放、透明的条件，全部流程中的条款和法律文本将会公开展示，所有的区块链用户都可查询/检索，并且该类数据的公益性，将会简化预言机智能合约的配置流程，无需维护支付关系，仅需维护数据和状态的更新即可。

© 2017 ONT Team

6. 核心协议标准

6.1. 多源认证协议

多源认证不同于以往的单一身份认证体系，本体可以为实体提供融合了外部身份信任源认证及本体中实体间背书的多源认证体系，不但可以为实体提供“我是谁”的基础身份认证，还可以为进一步为实体提供诸如我拥有什么、我喜欢什么、我经历过什么等多维度的身份信息，进而形成Who am I / What is it的多源认证package。

多源认证协议包括以下两种模式：

- 外部信任源认证：本体以自签可验证声明的形式给ONT ID 绑定外部信任源，任何实体都可以通过验证ONT ID绑定的外部信任源来验证实体的身份。实体身份认证的可信任程度由ONT ID绑定的外部信任源公信力和认可度决定。
- 本体实体间的身份认证：本体中的实体还可以通过本体中其他实体签发声明的方式实现身份认证。

6.1.1. 外部信任源认证

外部信任源认证支持自我导入和信任锚TrustAnchor导入两种方式。

6.1.1.1. 自我导入

用户通过社交媒体、银行UKEY签名等方法来绑定现实信任，该模式利用了现实世界已有的信任（比如微信、Facebook、银行等等）。原理实际上很简单，首先用户在ONT上添加一个外部信任源的证明地址，用户接着在该证明地址上提供一个可验证声明，包含如下内容：

- 声明创建与过期时间；
- 声明内容：包含声明的类型，ONT ID，社交媒体类型，社交媒体的用户名等；
- 签名：需要指定使用的公钥，必须是ONT ID描述信息中已包含的公钥列表中的一个。

当第三方需要验证用户的外部身份时，首先在本体中读取到用户信任源的证明地址，然后到这个地址去获取可验证声明，最后验证该可验证声明即可。正常来说，用户的社交媒体账户基本只能由其本人方可以进行管理，也就是说只有其本人才能够发表一个可验证声明。

© 2017 ONT Team

6.1.1.2. 信任锚导入

信任锚通常是那些经过了实名认证，且在社会中由一定公信力或声望的政府单位、企事业单位、非盈利组织以及社会名人等。成为信任锚需要遵守本体链委员会发布的一系列标准。信任锚使用自有的认证方式对被认证实体进行认证后，对被认证实体签发一个可验证声明。声明中不需要包含被认证实体的真实身份信息，仅记录实体及认证服务的ONT ID，作为该服务的认证结果证明即可。其认证模型如下：

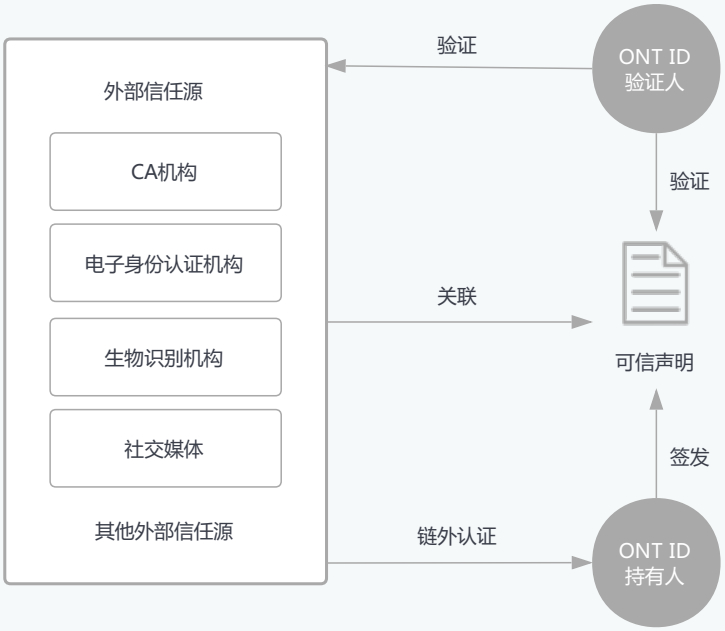


图 6.1 外部信任锚认证

6.1.2. 本体实体间的身份认证

本体中的实体还可以通过本体中已通过身份认证（外部信任源）的实体来认证身份，如图6.2所示，用户不仅可以通过学校、银行等实体做身份认证，还可以通过个人等其他方式获得身份认证。

正是由于可验证声明的开放性，使得本体中任何实体可以对任何另一实体就任何事情发布声明，这使得本体中的身份认证远远超过传统的身份认证概念。通过收集来自政府机构、学校、医院、银行、企业以及家人、朋友、领导、同事、老师、合作伙伴等的声明，可以证明“我们是谁”、“我们拥有什么”、“我们经历过什么”、“我们掌握什么技能”，甚至是“我们有什么兴趣爱好”等。这种多角度、多方面的认证相比传统的单一认证更准确、更全面。

© 2017 ONT Team

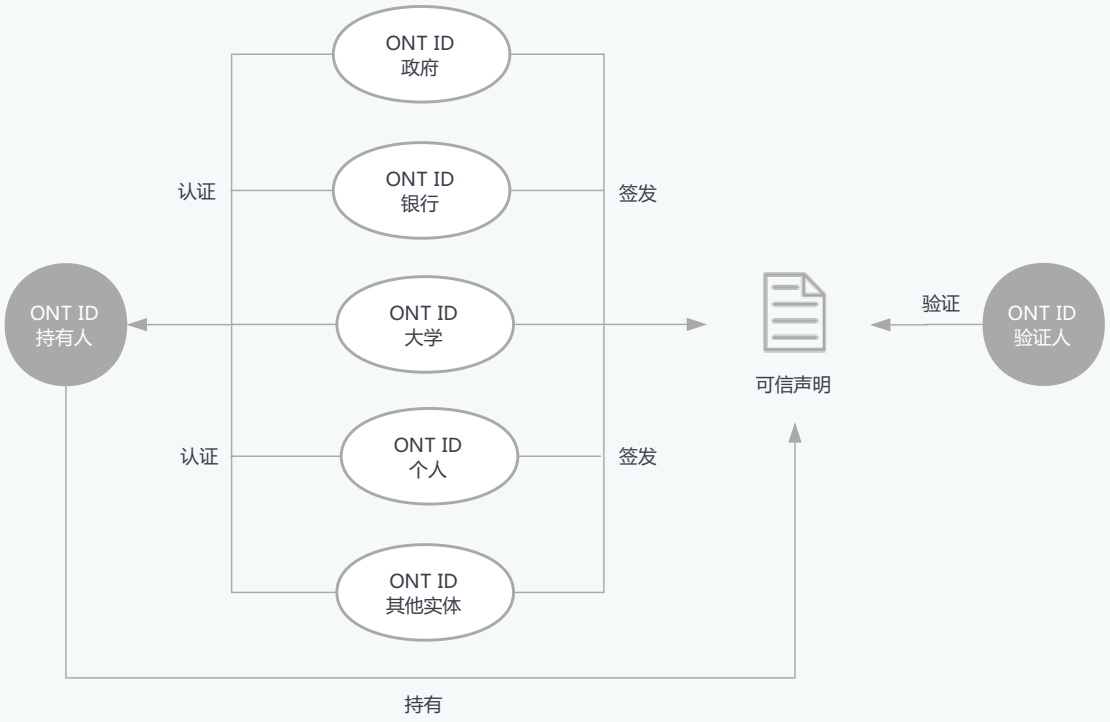


图 6.2 实体间身份认证

6.2. 用户授权协议

在本体中，用户对自己的数据有绝对的掌控权。任何涉及到用户主体的相关数据访问、交易都需要得到所有者的授权。为此，我们设计了一套用户授权协议来保护用户的数据隐私。协议利用可验证声明技术完成异步、可验证的授权，同时支持授权托管以及细粒度的访问控制策略制定。

6.2.1. 角色定义

在用户授权协议中涉及的主要角色有：

- 用户：对资源拥有所有权的实体，能够对资源作访问控制；
- 资源需求方：需要获取数据等资源的请求方；
- 资源提供方：提供资源的服务方；
- 授权服务：提供资源授权服务，接收用户对资源的访问控制策略及对资源请求进行权限认证。

© 2017 ONT Team

6.2.2. 授权流程

用户授权协议根据应用场景, 主要分为三个阶段:

1) 双向注册：用户授权资源提供方将指定资源注册到授权服务，同时授权服务向资源提供方注册自己的地址；
2) 设置访问控制策略：双向注册完成后, 用户可以访问授权服务设置资源的访问控制策略；
3) 授权：需求方向授权服务发起访问授权申请，若满足授权条件则会收到授权证明，用于向资源提供方请求数据。

6.2.3. 双向注册

双向注册是指授权服务向资源提供方注册自己的地址，以便资源提供方在处理资源访问请求时，给资源需求方返回授权服务的地址；同时，资源提供方也需要向授权服务注册自己的地址，以便用户对数据进行访问控制。

在进行授权操作之前，用户需要协同资源提供方和授权服务，完成双向注册流程。双向注册流程如图 6.3 所示：

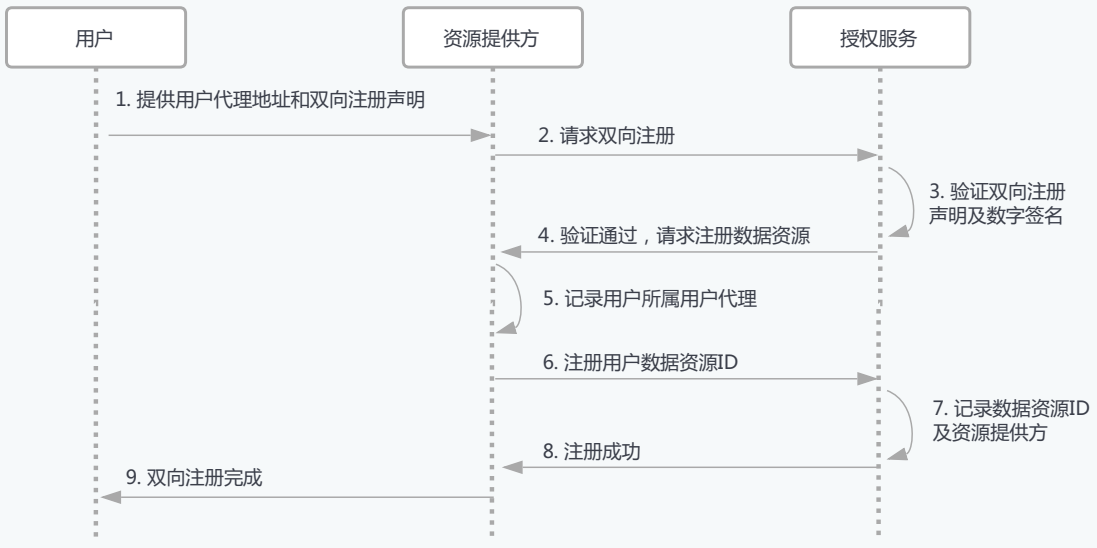


图 6.3 双向注册流程

1) 双向授权过程由用户启动，用户自签发一个双向注册声明，包含资源提供方的ONT ID和地址，及授权服务的ONT ID和地址；

© 2017 ONT Team

2) 资源提供方使用双向注册声明与授权服务进行握手，双方验证数字签名；

3) 资源提供方记录授权服务的访问地址；

4) 授权服务记录资源提供方提供的用户数据资源ID。

6.2.4. 访问控制策略

利用基于属性的访问控制，所有者能够设定灵活的访问控制策略，限制特定的资源能够由满足特定属性条件的请求者访问。访问控制策略可以描述为一个布尔表达式，如三个属性条件组合成的策略$A \vee (B \wedge C)$。请求者在申请授权时需以可验证声明的形式提供满足策略要求的属性证明。

6.2.5. 授权证明

授权服务接收到授权请求后，通知用户进行授权操作。对于满足授权条件的资源访问者，用户为其签发授权证明。在证明的有效期内，资源需求方可以重复访问数据而无需再次申请授权。

6.2.6. 授权托管

授权服务可以为用户托管授权操作。用户在授权服务中设置好自己的访问控制策略，后续的授权请求全部由授权服务处理并签发授权证明。用户需要为授权服务签发授权托管声明，以证明其授权结果的有效性。

6.3. 分布式数据交换协议

针对目前中心化数据交易所的痛点如：数据缓存、隐私数据未经用户授权、数据版权无法保护等问题，本体提出分布式数据交换协议DDEP，该协议对实体之间的数据交易行为定义了一整套协议规范。

为了保证交易双方的权益，在协议的交易流程中引入一个作为“担保人”的中间方，保证“一手交钱，一手交货”的结算过程。该中间方负责保管买方的资金，并根据最终交易结果将该资金转给卖方或退回给买方。因为中间方负责交易的最终结算，必须具备足够的公正性与安全性。依托于分布式账本运作的智能合约，具有公开且去中心化管理的特点，十分适合承担中间方的角色。

© 2017 ONT Team

6.3.1. 角色定义

在分布式数据交换协议中主要的角色有：

- 数据需求方：需要采购数据的机构/企业/个人；
- 数据提供方：提供数据的机构/企业/个人，数据可以是源数据，也可以是加工数据，数据提供需要完全满足当地政府的法律法规；
- 用户代理机构：负责和用户交互，以满足数据交易环节中需要用户的授权，用户代理机构形式可以多样（可以是企业OA系统、互联网平台甚至仅是简单的短信网关），但需要完整实现应用协议框架中定义的用户授权协议；
- 数据所有者：即数据主体，可以是机构/企业/个人。

6.3.2. 用户授权机制

数据交换体系中，由于交易的数据需要通过数据所有者的授权，授权流程完全遵守用户授权协议，其协议定义参考用户授权协议。

6.3.3. 担保交易模式

通过智能合约实现的担保交易协议，为交易行为提供了去中心化的第三方保证服务，最大程度保证“一手交钱，一手交货“的交易过程，保护买卖双方的权益。

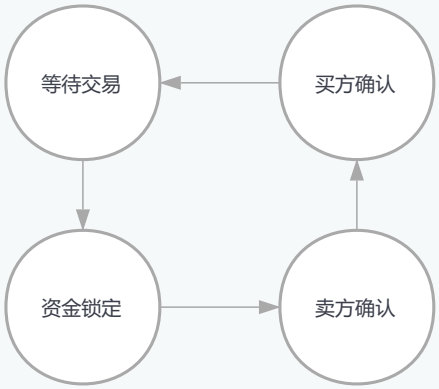


图 6.4 担保合约状态转换过程

担保交易智能合约的执行过程如下：

© 2017 ONT Team

1) 数据提供方在交易所挂单，写入售卖商品信息，包括资源ID、特征值、卖方账户、金额等参数，合约等待需求方发起交易；

2) 数据需求方向合约中转入指定的金额，合约检查转入金额是否满足售卖要求，

   a) 若检查通过，则记录该事件，合约进入资金锁定状态；

   b) 若检查未通过，则向转入者反馈错误，返回等待交易状态；

3) 数据提供方提供数据后，向合约进行确认，并设置有效期。若有效期结束时仍未执行其他操作，则合约自动进入结算过程（步骤5）；

4) 数据需求方收取数据完成后，向合约进行确认；

5) 合约将资金转到数据提供方账户，返回等待交易状态。

6.3.4. 数据交换流程

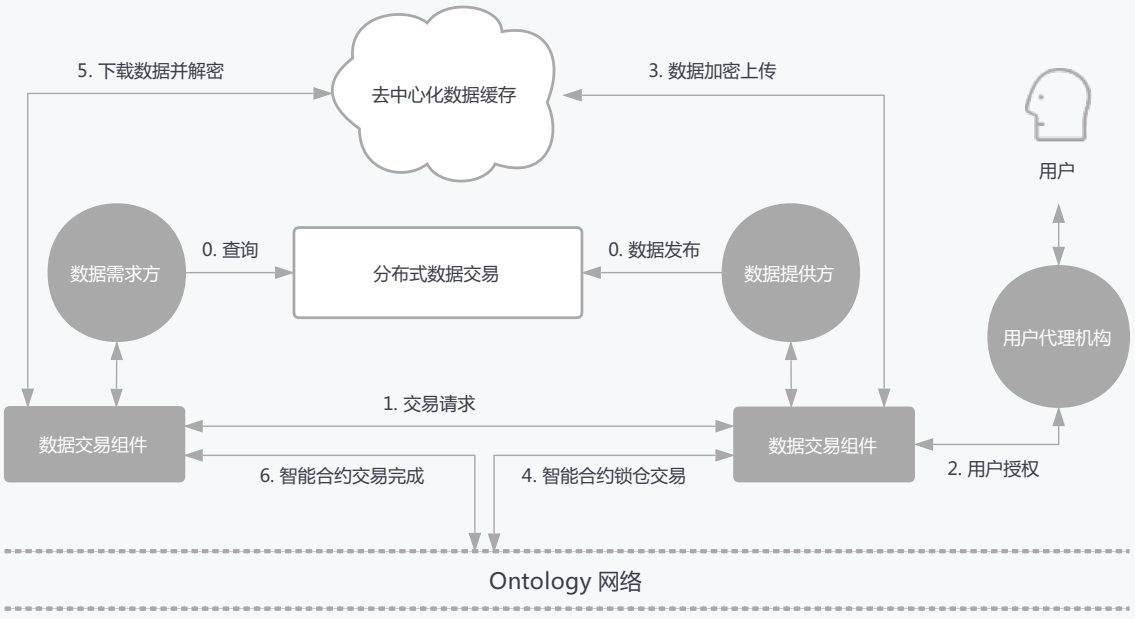


图 6.5 数据交换流程

前置步骤 交易前准备

数据产品发布：数据提供方将数据元信息在交易市场中发布，数据需求方可在交易市场中浏览、搜索数据信息，选择自己需求的数据发起交易。元信息中应包括但不限于：数据资源介绍、关键字、数据资源哈希、合约收款地址等信息。

© 2017 ONT Team

授权双向注册：数据产品如果需要数据所有者的授权，数据提供方在发布数据之前，首先需要和用户指定的用户代理机构之间进行双向注册，详见双向注册部分。

1) 交易请求

需求方在上架的数据产品中查看到想要购买的数据后，通过本体验证数据提供方的身份，具体参考多源认证协议。在交易请求发起之前，需求方首先向合约地址存入一笔资金，向提供方发送购买数据请求，并附带用户授权所需要的信息。该请求包括不限于：交易信息、ONT身份信息等。

2) 申请授权

数据提供方收到需求方的请求之后，访问用户代理，发起授权申请。此时，用户代理可以通过本体认证需求方的身份，并根据数据所有者事先提供的访问策略进行授权处理。如果所有者没有设置访问策略，用户代理通知其进行授权操作。如果未能获得的授权，则交易终止。

3) 上传数据

数据提供方根据请求方支持的对称加密算法，生成一次性会话密钥，使用会话密钥加密交易的数据和数据特征值，将密文上传到中间存储服务（如IPFS）。

4) 智能合约锁仓

数据提供方通过智能合约锁仓，合约在检查资金数额正确后，锁定数据需求方账户，直到交易完成或取消。同时使用需求方的公钥加密会话密钥，通过安全信道发送给需求方。

5) 收取数据

数据需求方接收到智能合约事件通知后（事件机制见5.3节），从中间存储服务收取数据，并使用会话密钥解密数据密文，然后计算明文的特征值进行比对验证，确认无误后，执行第六步。

6) 交易确认

数据需求方向交易合约确认交易完成，合约中的资金转到数据提供方账户。

异常处理机制：异常处理机制可以根据业务场景进行定制实现，比如可以实现：如果超过一定时间，数据需求方还没有确认的话，数据提供方可以调用合约解锁资金，或者智能合约自动触发解锁资金。

© 2017 ONT Team

6.3.5. 对交易者的隐私保护技术

在部分交易场景中，交易者需要隐藏自己真实的交易记录信息，我们使用隐形地址技术来模糊交易收款与真实身份ONT ID之间的关联。具体方法是数据需求方根据数据登记中的收款地址生成一个隐形地址，该隐形地址的私钥只有数据提供方才能够掌握。

假设数据提供方的私钥是s，收款地址是$S = s \cdot G$。需求方随机生成一个随机数r，计算$R = r \cdot G$，并计算$E = Hash(r \cdot S) \cdot G + S$。需求方进而向地址$E$转账，并附加信息$R$。仅数据提供方利用$s$可计算出隐形地址的私钥$e = Hash(s \cdot R) + s$。

© 2017 ONT Team

7. 本体应用框架

7.1. 应用框架模型

本体应用框架提供了一系列丰富的应用层协议和组件，帮助应用开发者快速构建出去中心化应用，使其不用花精力关注底层分布式账本交互的复杂性。本体应用框架具有高度的可扩展性，可以根据实际场景的需要，不断进行扩展。

图7.1为应用框架模型示意。dApp通过应用框架与本体交互，实现去中心化信任。

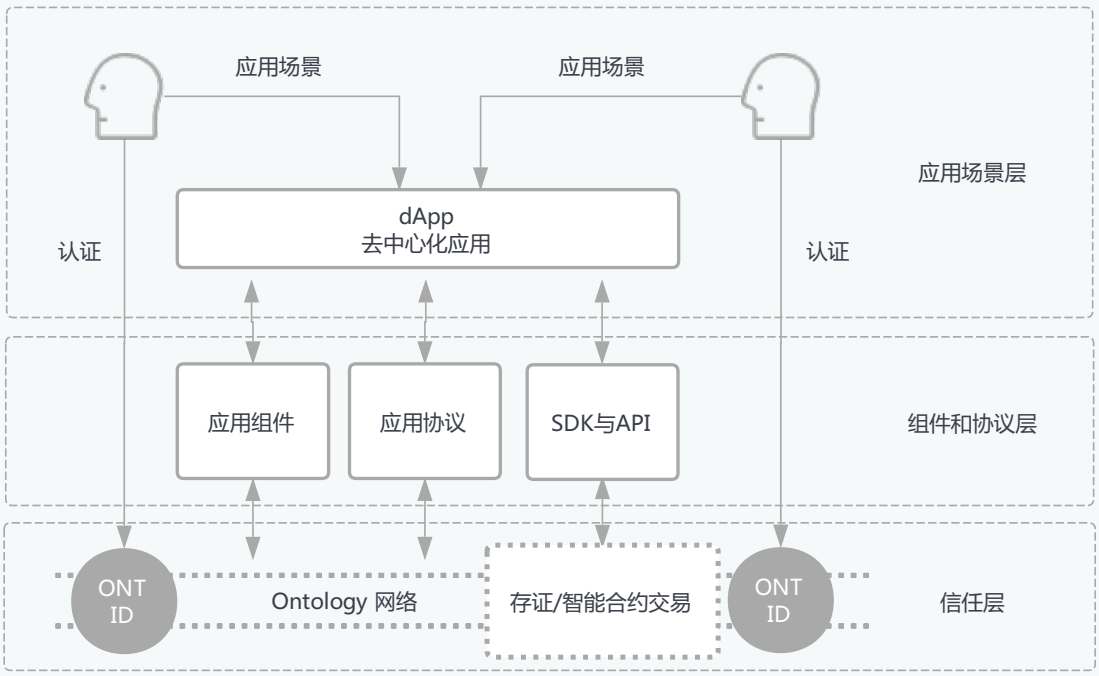


图 7.1 应用框架模型示意图

- 信任层：本体通过去中心化的身份体系、存证、智能合约交易等实现了分布式信任；
- 组件和协议层：通过应用组件、应用协议、SDK和API帮助上层场景更好的使用本体；
- 应用场景层：各种场景dApp重点关注场景开发、用户服务等，信任问题交给本体来解决。

© 2017 ONT Team

7.2. 数据交易市场

未来是一个数字化的世界，未来的数据交易市场也将不仅限于对数据所有权的转移，数据的可信协同计算也将成为重要的协作模式。数据交易市场中的商品包括数据产品、数据预测、数据计算资源等，参与角色包括数据的使用方、提供方，还包括数据的加工方，比如利用深度学习等AI技术处理大数据的服务商，这些角色共同构成了数据协作生态。复杂的数据协作生态需要与之匹配的基础设施提供支撑。

本体提出的分布式交换协议DDEP、数据交易组件DDM及一系列密码学组件共同构成了完备的分布式数据交易和协作框架，确保满足面向全球级的交易规模以及跨交易市场的交易需求，支持基于本体的服务全球用户的各类dApp应用。上层数据交易服务商可以在此基础上实现各类别、各领域的数据交易市场。

7.3. 数据交易组件

数据交易组件（Data Dealer Module，DDM）基于分布式数据交换协议实现，是本体重要的基础应用组件。无论是dApp开发者还是数据交易参与方都可以通过组件快速实现基于本体的数据交易应用。该组件提供RESTful接口、RPC、SDK等，可支持不同类型协议。

数据交易组件包括多种类型：数据交易服务端、单用户客户端、多用户客户端、轻钱包客户端。不同类型的组件适合不同的应用场景。组件的功能主要分四大模块：ONT身份管理、数据资源管理、智能合约交易、点对点通讯。

数据交易组件的设计具有以下主要优势和特点：

- 组件与访问控制模块解耦：组件只管理数据资源与数据资源所有者权限控制服务器的绑定关系，不参与访问权限配置与验证，既维护数据拥有者的隐私保护权益，也增强卖方信用度，避免不必要的纠纷。
- 保护数据隐私：组件不存储实际数据，交易数据可加密，消除卖方担心数据沉淀等顾虑。
- 需求方除了检索数据，还可通过广播需求订单通知数据提供方。
- 按照“单一模块单一功能”原则设计，和密码学安全组件、用户授权组件配合，将很容易支持灵活多变的场景需求。

© 2017 ONT Team

© 2017 ONT Team

7.4. 密码学及安全组件

7.4.1. 安全多方计算

在数据协作场景中，协作的两方甚至多方，它们既希望能够完成协作任务，又希望保留源数据所有权和控制权，而仅仅向对方开放有限的数据使用权。目前传统做法无法满足需求，比如，要使用机器学习算法的应用开发商 A，传统的做法是，提供数据给算法提供商 P，由 P在 A 提供的数据上运行其算法，得到的结果再返回给 A，但这种传统方式已将 A 的数据泄露给 P。

对于这种典型场景，我们采用多方安全计算技术（Multiparty Secure Computation, MSC）来处理。最早的 MPC 技术是由姚期智提出的，其方法可以解决著名的“百万富翁问题”：两个百万富翁希望比较谁更有钱，但是又不希望对方知道自己的真实财产是多少。在 n 个协作方的场景中，我们假设他们各自持有数据 x_i，协作的目标是计算关于这 n 个数据的函数值 $f(x_1, \ldots, x_n)$，目标是所有人最终只知道该函数值，任意一方数据的其他信息都没有泄露。

MPC技术主要分为两种，一种是基于姚期智提出的混淆电路[22][24]，另一种是基于秘密分享[23]。这两种技术路线各有优缺点，会根据场景进行选择。

在本小节，我们说明基于秘密分享的方法。简单来说，在 t- 门限 n 方秘密分享方案中，秘密被划分为 n 份，分发给每一方，只需要有超过 t 个人参与，就可以恢复原始的秘密 s。在基于秘密分享的 MPC 协议中，通常是将协议执行的中间结果利用秘密分享算法，分配到不同参与者中，最后参与者根据自己在协议执行阶段所得到的份额，进行重组，获得最后的结果。

7.4.2. 全同态加密

在许多数据交易场景中，数据提供商仅仅希望提供数据的使用权，而不是将数据直接就转让给买方。提供对企业数据的隐私保护是极为迫切的需求。全同态加密[25][26][27][28]技术提供了一个很好的解决方案。企业首先利用自己的公钥，使用全同态加密算法对数据 m 加密，将密文 C 提供给算法提供商，对 C 进行加法或者乘法运算（以及由加法和乘法合成的复杂运算），经过这些运算之后，得到密文 C' 并返回给企业，企业使用其私钥恢复出明文结果 $f(m)$。

一个全同态加密算法除了包括基本的加解密算法，还有密文相加 $CAdd$ 和相乘 $CMul$ 算法。假设 C_1 和 C_2 分别是明文 M_1 和 M_2 的密文，那么

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

© 2017 ONT Team

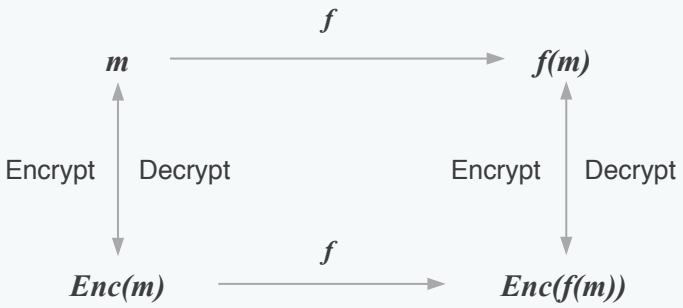


图 7.2 在密文上作同态运算

基于这两个最基本的运算，以及一些优化方法，可以构造相对复杂的密文运算，如图7.2所示，在密文上运行复杂运算f之后，得到的密文可以被解密，且恰好是等于$f(m)$。

在未来应用框架会支持更多的全同态算法，如BGV算法、FV算法。

7.4.3. 数据的版权保护

针对数据的数字化特性，本体提供数据存证与生命周期管理功能，设计了相关数据的生命周期溯源机制。实现为每一份数据建立数字身份，以对其登记、请求、授权、交易等全流程进行有主体的追踪；其次，数据的版权保护，数据的交易均在分布式账本进行记录。

本体采用数字水印技术将标识信息直接嵌入数字载体当中，在不影响原载体的使用价值前提下，可防范攻击者修改，并被生产方轻易识别和辨认。通过这些隐藏在载体中的信息，可以达到确认内容创建者、购买者判断载体是否被篡改等目的。数字水印是实现防伪溯源、版权保护、隐藏标识、认证和安全隐蔽通信等应用的有效办法。

本体引入的数字水印技术具备几大特性：

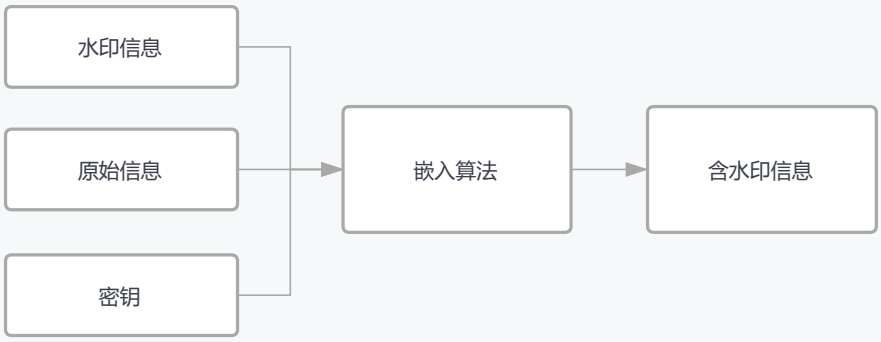
- 可证明性：水印应能为受到版权保护的信息产品的归属提供完全和可靠的证据。利用水印算法我们可以识别被嵌入到保护对象中的所有者的有关信息(如注册的用户号码、产品标志或有意义的文字等)并能在需要的时候将其提取出来。水印可以用来判别对象是否受到保护，并能够监视被保护数据的传播、真伪鉴别以及非法拷贝控制等。
- 不可感知性：不可感知是指视觉上的不可感知性(对听觉也是同样的要求)，即因嵌入水印导致图像的变化对观察者的视觉系统来讲应该是不可察觉的，最理想的情况是水印图像与原始图像在视觉上一模一样，这是绝大多数水印算法所应达到的要求。
- 鲁棒性：指在经历多种无意或有意的信号处理过程后，数字水印仍能保持完整性或仍能被准确鉴别。鲁棒性问题对水印而言极为重要。一个数字水印应该能够承受大量的、

© 2017 ONT Team

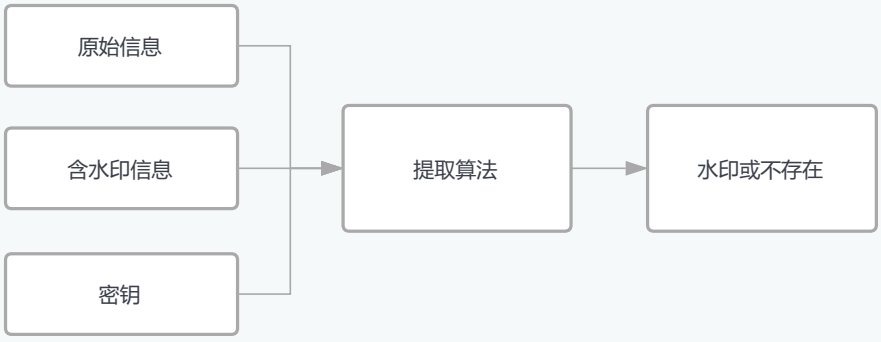
> 不同的物理和几何失真，包括有意的(如恶意攻击)或无意的(如图像压缩、滤波、扫描与复印、噪声污染、尺寸变化等等)。显然在经过这些操作后，鲁棒的水印算法应仍能从水印图像中提取出嵌入的水印或证明水印的存在。如果不掌握具体的水印嵌入和检测算法，数据产品的版权保护标志应该很难被伪造，若攻击者试图删除水印则将导致多媒体产品的彻底破坏。

本体的数字水印功能包括两个基本模块：一个是水印嵌入模块，另一个是水印提取模块，如图7.3所示。

数字水印会作为防伪的支撑服务被应用于证件、票据防伪，与数字身份实现关联。作为证件来讲，每个人需要不只一个证件，证明个人身份的有：身份证、护照、驾驶证、出入证等；证明某种能力的有：各种学历证书、资格证书等。通过数字水印技术可以确认该证件的真伪，使得该证件无法仿制和复制。



(a) 数字水印嵌入过程



(b) 数字水印提取过程

图 7.3 数字水印功能模块

© 2017 ONT Team

7.5. 用户授权控制组件

用户授权控制组件UAC基于用户授权协议实现。UAC组件实现授权主体对自己数据的多维度颗粒状的授权访问细化控制，任何涉及到授权主体相关数据的交易，都会通知授权主体，得到授权后才可进行数据交易。UAC组件对外提供RESTful API，支持使用关系型数据库(Mysql或Oracle)。主要实现了两大块功能：

- 用户数据授权访问策略设置；
- 用户数据授权访问控制。

7.5.1. 授权访问策略设置

用户要求资源提供方通过UAC组件提供的RESTful API对用户的数据进行资源注册，注册成功后用户通过UAC组件便可查看、搜索自己的已注册数据，并对自己的数据进行细化的访问控制策略设置。且针对一些频繁访问的隐私级别不太高的数据，用户可在UAC组件设置授权托管。

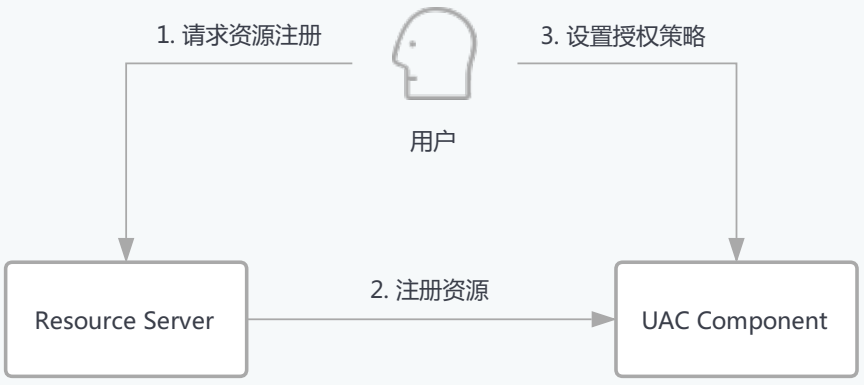


图 7.4 设置访问控制策略

7.5.2. 授权访问控制

作为用户数据的保护方，在数据交易流程中，当涉及到用户数据时，非托管模式下UAC组件会通知用户进行授权决策，由用户自行决定是否允许请求方访问自己的数据。托管模式下，UAC组件代用户进行授权决策，并提供授权回执给用户。两种模式下用户都能清楚的知道是谁，在什么时候，对自己的哪些数据，要做什么操作。

© 2017 ONT Team

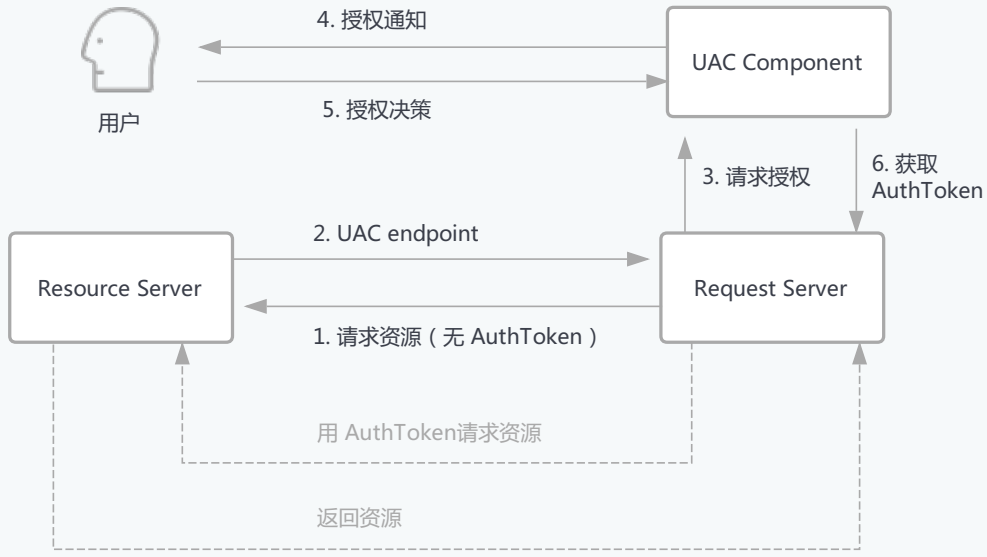


图 7.5 授权访问控制

7.6. 声明管理组件

对于本体的信任锚来说，可验证声明管理是重要的功能组件。声明管理组件按照分布式信任体系要求开发。组件对外提供RESTful API，支持使用关系型数据库（Mysql或Oracle）。其组件主要功能包括但不限于：可验证声明签发、验证、查询和注销。

7.7. 全局事务数据库

全局事务数据库GlobalDB是一个可插拔的Key-Value分布式数据库界面。它的底层是基于Google Spanner / F1的设计实现的开源分布式NewSQL 数据库TiDB。

GlobalDB是为区块链/分布式账本以及IPFS高度优化的数据库组件。GlobalDB提供了SQL兼容、存储分片、分布式事务、水平线性扩展、故障自动恢复的能力，可应用在区块链与大数据、区块链与人工智能等计算相关的场景。

GlobalDB具备分布式事务、存储分片、负载均衡以及SQL on KV四大特性。

© 2017 ONT Team

7.7.1. 分布式事务

GlobalDB可以提供完整的分布式事务，为状态分片和 off-chain 业务提供支撑，事务模型基于Google Percolator[29]的基础上做了一些优化。

GlobalDB 的事务模型采用乐观锁，分布式事务只有在真正提交的时候，才会做冲突检测。传统的方式如果有冲突，则需要重试，这种模型在冲突严重的场景下，会比较低效，而乐观锁模型在大部分场景下具备较高的效率。

由于分布式事务要做两阶段提交，并且底层还需要做一致性复制，如果一个事务非常大，会使得提交过程非常慢，并且会卡住下面的一致性复制流程。为了避免系统出现被卡住的情况，我们对事务的大小做了限制：

1) 单条KV记录不超过6MB；
2) KV记录的总条数不超过300,000；
3) KV记录的总大小不超过100MB。

7.7.2. 存储分片

GlobalDB 自动将底层数据按照Key的范围进行分片，每个分片是一个$[StartKey, EndKey)$区间。分片中的 Key-Value 总量超过一定阈值，就会自动分裂。

7.7.3. 负载均衡

负载均衡器（PD）会根据存储集群的状态，对集群的负载进行调度。调度是以分片为单位，以PD配置的策略为调度逻辑，全部过程自动完成。

7.7.4. SQL ON KV

GlobalDB 自动将 SQL 结构映射为KV结构。简单来说，GlobalDB 做了两件事：

- 一行数据映射为一个KV，Key以TableID构造前缀，以行ID为后缀；
- 一条索引映射为一个KV，Key以TableID+IndexID构造前缀，以索引值构造后缀。

可以看到，对于一个表中的数据或者索引，会具有相同的前缀，这样在 TiKV 的 Key 空间内，这些 Key-Value 会在相邻的位置，GlobalDB 会配置相应的脏数据管理策略，使得数据读取达到较高的性能。[30]

© 2017 ONT Team

GlobalDB 支持高度可配置，可同时适配 on-chain 业务与 off-chain 实时高性能业务。它将作为本体的核心组件为底层账本平台提供坚实支撑。

© 2017 ONT Team

8. 后记

本白皮书仅是对本体涉及的相关技术的部分概览，技术无止境，本体的技术白皮书将随着更多应用的拓展而持续更新。

同时，本体将创造一个开放、协作、创新的技术生态，本体团队也欢迎全球开发者加入本体家族，共同参与和推进本体技术的进步。

© 2017 ONT Team

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.

© 2017 ONT Team

[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/.

© 2017 ONT Team

联系方式

电子邮件：contact@ont.io

Telegram：OntologyNetworkCN

Twitter：OntologyNetwork

Facebook：ONTnetwork

Reddit：OntologyNetwork

Discord：https://discord.gg/vKRdcct

Medium：OntologyNetwork

LinkedIn：Ontology Network

微信公众号： **ONT本体网络**

微信客服： **ONT小秘书**

© 2017 ONT Team