Ontology

TECHNOLOGISCHE WHITE PAPER

Versie 0.6.5

Update：2018/03/14

Een nieuw high-performance openbaar multi-keten project &
Een Distributed Trust Collaboration Platform

Abstract

Door de geschiedenis heen hebben mensen vertrouwen gevestigd door middel van technologie, de rechtsstaat en gemeenschappen.

Vertrouwen en samenwerking tussen entiteiten omvat echter meerdere bronnen en geïsoleerde systemen, wat betekent dat het kostbaar kan zijn en daarom de breedte en diepte van samenwerkingsmogelijkheden belemmert. Hoewel de technologie de laatste tijd veel vooruitgang heeft geboekt, bemoeilijken te veel factoren de samenwerking met vertrouwen. Deze omvatten fragmentatie van trustsystemen, de ontbrekende rol van het individu, onnauwkeurige identiteitsverificatie, onvermogen om valse informatie te betwisten, enz. Op gebieden zoals sociaal bestuur, economische samenwerking en financiële diensten zijn de kosten van het opbouwen van vertrouwen enorm.

De gedecentraliseerde, sabotagebestendige blockchain heeft het vertrouwen door technologie naar bepaalde industrieën gebracht, maar verdere integratieve mechanismen zijn nodig om verschillende trustsystemen en -toepassingen samen te brengen in een enkel nieuw trust-ecosysteem.

Ontology vestigt de verbindende infrastructuur voor een vertrouwensecosysteem, met effectieve coördinatie van vertrouwen en gegevensbronnen, evenals het verschaffen van de infrastructuur voor gedistribueerde applicatie-ontwikkeling.

Dit artikel concentreert zich op het technologische raamwerk van Ontology, belangrijke technologische principes en kernprotocollen

Inhoudsopgave

1. INTRODUCTIE

Ontology is een geïntegreerd multi-keten en multi-systeem raamwerk dat is samengesteld uit verschillende industrieën en regio's die door de protocollen van de Ontology gaan om het in kaart brengen mogelijk te maken tussen verschillende ketens en traditionele informatiesystemen. Om deze reden wordt Ontology ook wel "Ontologyketengroep" of "Ontologyketennetwerk" genoemd, dat wil zeggen, een verbinding tussen blockchains.

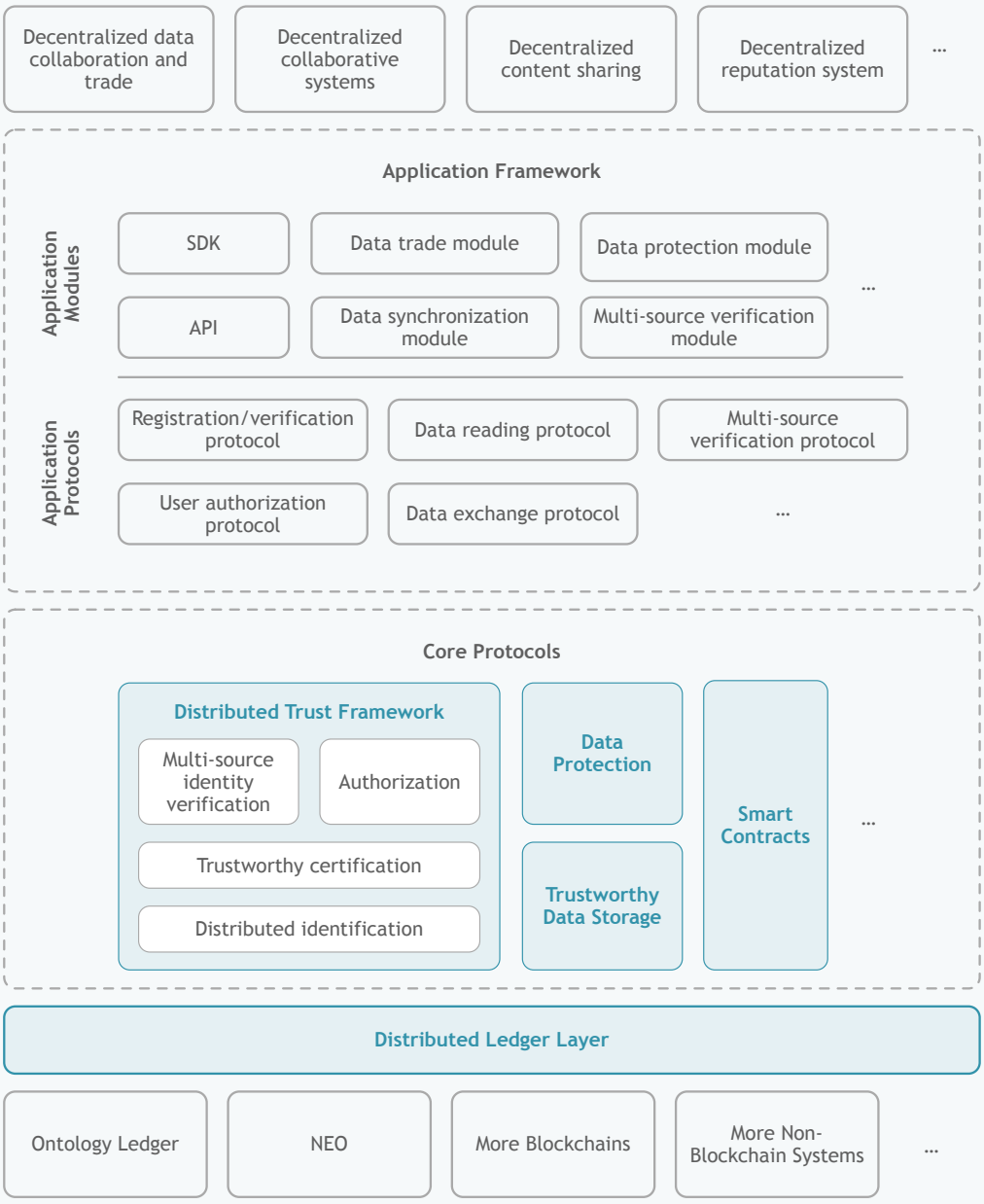

Figure 1.1: Ontology's Technology Framework

De kern van Ontology is een compleet gedistribueerd grootboeksysteem, inclusief de Core Ledger, het smart contractsysteem en het beveiligingssysteem. Het gedistribueerde grootboek is een belangrijke onderliggende opslaginfrastructuur van Ontology; de gedecentraliseerde, wederzijds beheerde en fraudebestendige kenmerken van gedistribueerde grootboektechnologie zijn essentieel voor het creëren van vertrouwen tussen entiteiten in het netwerk. Het gedistribueerde grootboek omvat een consensus en een smart contractsysteem en biedt consensus, opslag en smart contractondersteuning voor het bovenste toepassingsraamwerk. Ontology en de gedistribueerde grootboektechnologie gebruiken een ontkoppelde architectuur (wat de standaard is voor het Ontology Ledger) die NEO, Ethereum en andere blockchains als de onderliggende laag kunnen ondersteunen. Op het ledger-niveau stellen we creatief een model voor gedeelde datacontracten voor dat ontkoppelt van gegevensopslag en bedrijfslogica, en implementeert verschillende bedrijfslogica met het gebruik van slimme contracten om de schaalbaarheid en flexibiliteit van de architectuur in het algemeen te verbeteren.

Het gedistribueerde vertrouwensraamwerk is de kernlogica van Ontology. We gebruiken ONT ID om verschillende ID-authenticatiediensten aan te sluiten en een bron van vertrouwen te bieden aan mensen, geld, goederen en dingen. ONT ID is gedecentraliseerd, zelf-beheersbaar, privacy-beschermd en veilig en gemakkelijk te gebruiken. Het vertrouwenskader maakt een gedistribueerd vertrouwensmodel en verspreidt het vertrouwensbezorgsysteem via verifieerbare claims [1] [2] en gebruikt het CL-handtekeningalgoritme en het nulkennisbestendige protocol om privacybescherming van verifieerbare claims te garanderen.

Ontology gebruikt een reeks protocolstandaarden. Voorstellen omvatten een identiteitsprotocol, multi-source authenticatieprotocol, gebruikersautorisatieprotocol en gedistribueerd gegevensuitwisselingsprotocol. Er is een reeks internationale protocollen geïmplementeerd, zoals DID [3], ontworpen door W3C. Het cryptografische handtekeningprotocol ondersteunt ook cryptografie zoals de Chinese nationale cryptografiestandaard, RSA en ECDSA. Het gedistribueerde gegevensuitwisselingssysteem is compatibel met de algemeen gebruikte autorisatieprotocollen, b.v. Oauth [4] en UMA [5], om de architectuur in staat te stellen te voldoen aan het doel om open en gestandaardiseerd te zijn en om toekomstige uitgebreide ecologische samenwerking en expansie te ondersteunen.

Voor applicatiediensten biedt Ontology de infrastructuur voor applicatieontwikkelaars om direct gedistribueerde services te leveren bovenop Ontology zonder kennis te hebben van hoe gedistribueerde systemen kunnen worden ontwikkeld. Kort gezegd biedt Ontology een reeks toepassingskaders, waaronder API's, SDK's en verschillende toepassingsmodules waarmee applicatiedienstverleners met een breed scala aan technologische achtergronden hun eigen dApps kunnen ontwikkelen en dApps as a Service (DAAS) kunnen maken. Dit maakt blockchain gebruiksvriendelijk voor iedereen.

Ontology omvat ook een verscheidenheid aan geavanceerde modules: modules voor cryptografie en gegevensbescherming, markten voor gegevensuitwisseling, wereldwijde transactiedatabases, hybride orakel, onbeperkte consensusmotoren en meer. In de toekomst zal Ontology een gemeenschap van ontwikkelaars en zakelijke partners ontwikkelen, die voortdurend toepassingen

en modules verrijken om de technologische ontwikkeling van het ecosysteem van Ontology te bevorderen.

2. GLOSSARIUM

Ontology Chain Group

Dit wordt ook wel het 'Ontologyketennetwerk' genoemd en wordt gevormd door ketens van entiteiten die zijn gevestigd in ongelijksoortige industrieën en regio's die samenkomen in Ontology. Elke keten gebruikt een afzonderlijk gedistribueerd grootboek en werkt samen via interactieve protocollen.

Ontology Distributed Ledger (gedistribueerde grootboek)

Een of meer core-serviceketens maken deel uit van het gedistribueerde grootboek / blockchain-framework van Ontology. Het biedt core gedistribueerde grootboek, slimme contractsysteem en andere diensten voor alle diensten Ontology.

Entiteit

Deelnemers aan Ontology die kunnen worden geïdentificeerd aan de hand van hun ONT ID.

ONT ID

ONT ID is een gedecentraliseerd gedistribueerd identificatieprotocol dat is gebaseerd op gegevens van de identiteitsservice (s) van een entiteit, die zijn verbonden met cartografische services en andere links. Het is gedecentraliseerd, zelf-beheersbaar, privacy-beschermd en veilig en gemakkelijk te gebruiken.

Verifieerbare claim

Een verklaring om een claim van een entiteit over een andere (inclusief zichzelf) te bevestigen. De claim gaat vergezeld van een digitale handtekening die door andere entiteiten kan worden gebruikt voor authenticatie.

Ontology Trust Framework

Modules die deel uitmaken van het vertrouwensecosysteem van Ontology, waaronder het gedistribueerde protocol voor identiteitsverificatie, gedistribueerd vertrouwensmodel, gedistribueerde overdracht van vertrouwensrelaties en andere modules.

Meerdere-bronnenauthenticatie

Verwijst naar veel verschillende verificaties met betrekking tot verschillende aspecten van dezelfde entiteit om een verificatie met meerdere bronnen te maken.

Trust Anchor

Een entiteit waaraan de verificatie is toevertrouwd. Het fungeert als een bron voor distributieketens voor vertrouwensrelaties en biedt diensten voor identiteitscontrole.

Distributed Consistent Ledger

Een opslagmechanisme voor opslag van incrementele modificaties in een gedecentraliseerd P2P-netwerk dat wordt onderhouden als gemeenschappelijke knooppunten. Transparant en

fraudebestendig van aard, het biedt vertrouwde opslag en smart contractondersteuning voor Ontology.

Consensus

Elk knooppunt bevestigt gegevens die op het grootboek zijn geschreven volgens een protocol om de consistentie te waarborgen.

Smart Contract

Uitvoerbare code opgenomen in het grootboek loopt door de smart contractengine die op de knopen van de grootboek loopt. De invoer en uitvoer van elke uitvoering kunnen ook in het grootboek worden vastgelegd.

Ontology Application Framework

Een algemene term die verwijst naar toepassingsmodules, protocollen, SDK's en API's om snelle, goedkope toegang tot dApp's van derden te vergemakkelijken.

Hybrid Oracle

Hybrid Oracle is een service die geloofwaardige, externe gegevens aan de blockchain levert. Hiermee kunnen gebruikers de resultaten van gebeurtenissen buiten het blockchain-systeem voorspellen en deze permanent op de blockchain registreren.

3. KETTINGROEPSTRUCTUUR

Het doel van Ontology is om een brug te slaan tussen de echte wereld en gedistribueerde datasystemen. Vanwege de diversiteit, complexiteit en specialisatie van bedrijven in de echte wereld, moeten overwegingen met betrekking tot prestaties, schaalbaarheid en toepasbaarheid in aanmerking worden genomen, waarvan het moeilijk is om één netwerkketen / aangesloten kettingnetwerk te gebruiken om alle scenario's te ondersteunen. In de praktijk vereisen verschillende bedrijfslogica's verschillende ketens om te voldoen aan de behoeften van verschillende scenario's met verschillende toegangsmethoden en besturingsmodellen. Veel bedrijfsscenario's bestaan ook niet onafhankelijk en vereisen een gediversifieerde interactie met andere scenario's. Daarom moeten er verschillende protocollen tussen deze verschillende ketens worden aangeboden om de samenwerking tussen verschillende diensten te ondersteunen.

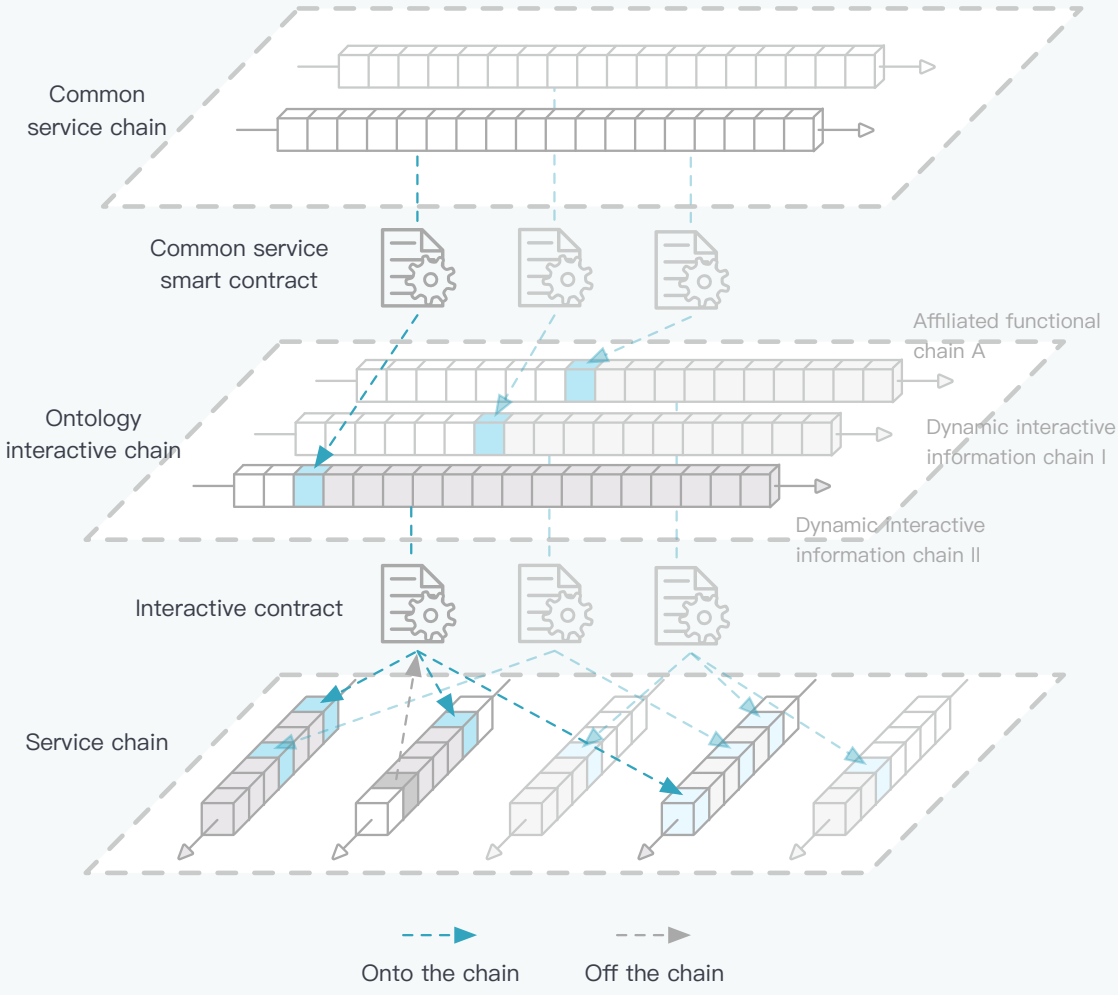

Figure 3.1: Hyper-Converged Chain Group

Op basis van deze vereisten en modellen stelt Ontology een hyper-geconvergeerde ketengroep voor, die de vorm aanneemt van een matrixraster. In een horizontaal gebied kunnen er openbare ketens zijn die elementaire gemeenschappelijke diensten bieden, zoals entity mapping, protocolondersteuning voor gegevensuitwisseling en mart contractdiensten. Op een of meer openbare blockchains kan elk bedrijfstak, geografisch gebied en bedrijfsscenario een eigen unieke serviceketen hebben die kan voldoen aan de vereisten voor toegang, compliance, governance, consensus, enz. Elk kan ook openbare ketens gebruiken om basisdiensten te leveren, zoals entiteitverificatie, protocollen voor gegevensuitwisseling, enz., evenals samenwerken met andere bedrijven.

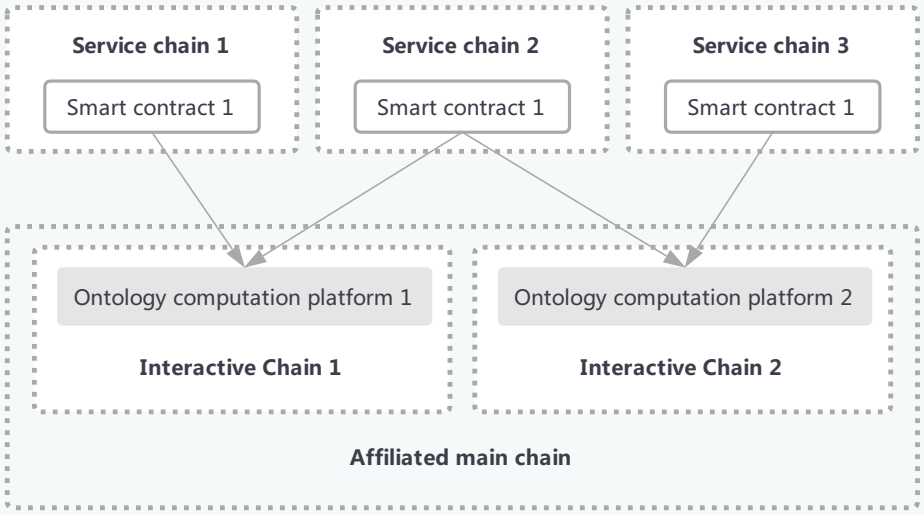

Figure 3.2 Service chains using the Ontology computation platform for cross-chain information exchange

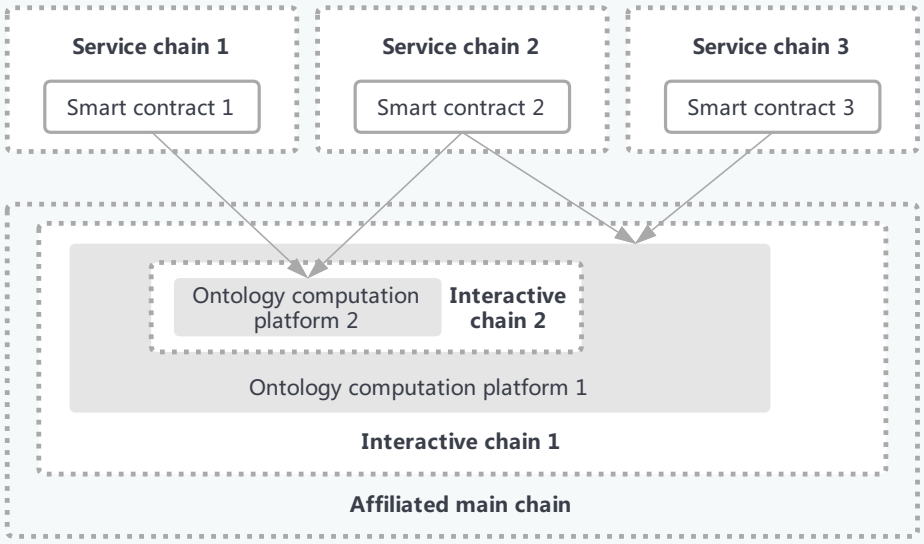

Figure 3.3 Building a dedicated interactive chain using Ontology's computation platform

Naast het gebruik van de publieke keten, zullen er ook samenwerkingsverbanden zijn met ketens die verband houden met de industrie van het specifieke bedrijf. Onder verschillende samenwerkingsverbanden kunnen de betrokken serviceketens ook variëren, dus er kunnen enkele kleine toegewijde openbare / geaffilieerde serviceketens zijn die samenwerken met een of meerdere serviceketens of servicepunten voor specifieke zakelijke vereisten. Daarom zullen er in het verticale gebied veel samenwerkingsketens voor bedrijven ontstaan die speciale gezamenlijke samenwerkingsverbanden met meerdere ketens zullen omvatten voor smart contractdiensten, logische zakelijke diensten, enz.

De matrixrasterarchitectuur kan een echt autonoom next-generation, veelzijdig netwerk vormen. Verschillende bedrijfsscenario's kunnen verschillende manieren vinden om het juiste servicemodel toe te passen via een breed scala aan samenwerkingsmethoden.

De verschillende protocollen in Ontology zijn niet statisch en gebruikers kunnen verschillende protocollen kiezen op basis van verschillende bedrijfsscenario's, industriefuncties, wettelijke vereisten en governance-vereisten. Daarom zullen protocollen in Ontology deel blijven uitmaken van het ontwikkelingsproces, hoewel het belangrijkste doel is om de bruikbaarheid te maximaliseren met verschillende protocollen en standaarden in elk scenario om de Ontology beter compatibel en schaalbaar te maken met de wereld.

Ontology gebruikt zijn gedistribueerde grootboekraamwerk om te voldoen aan verschillende scenario's met de implementatie van een of meer configureerbare blockchains. Het gedistribueerde grootboekraamwerk kan ook de serviceketen aanpassen voor specifieke bedrijfsscenario's (bijvoorbeeld verschillende mechanismen voor toegang, governance, consensus, opslag, enz.). Daarnaast kan Ontology samenwerken met andere bestaande blockchain-systemen en traditionele IT-systemen door het gebruik van protocollen.

4. DISTRIBUTED TRUST FRAMEWORK

4.1. ONTOLOGY IDENTIFICATIE PROTOCOL

"Entiteit" verwijst naar individuen, rechtspersonen (organisaties, ondernemingen, instellingen, enz.), Objecten (mobiele telefoons, auto's, IoT-apparaten, enz.) In de echte wereld, en "identiteit" verwijst naar de identiteit van de entiteit binnen het netwerk . Ontology gebruikt Ontology Identifier (ONT ID) om de identiteiten van de entiteiten in het netwerk te identificeren en te beheren. In Ontology kan een entiteit corresponderen met meerdere individuele identiteiten.

ONT ID is een gedecentraliseerd identiteitsprotocol dat verschillende authenticatiediensten van een enkele entiteit met elkaar verbindt en in kaart brengt. Het is gedecentraliseerd, zelfbeheersbaar, beschermd tegen privacy en veilig en gemakkelijk te gebruiken. Elke ONT ID komt overeen met een ONT ID Beschrijving Object, dat wordt gebruikt om de attribuutinformatie, zoals de publieke sleutel, van de ONT ID vast te leggen. Beschrijvingobjecten dienen als openbare informatie voor opslag in de distributielaag in de kern van Ontology. Om privacyredenen bevat het beschrijvingobject standaard geen informatie over de identiteit van een entiteit.

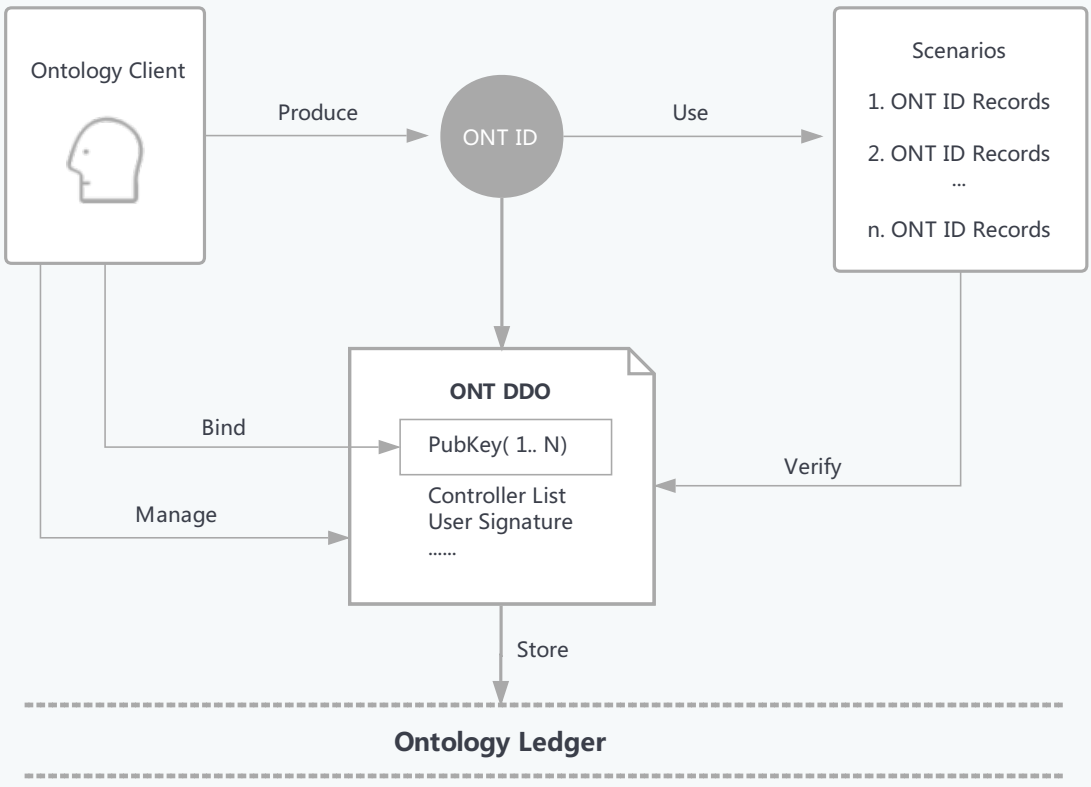

Figure 4.1: ONT ID System

4.1.1. ONT ID Generatie

ONT ID is een type URI [6], gegenereerd door elke entiteit. Het generatie-algoritme zorgt ervoor dat er een zeer kleine kans is op duplicatie van twee ONT-ID's($\approx \frac{1}{2^{160}}$). Tegelijkertijd controleert het consensusknooppunt bij het registreren op Ontology of de ID al is geregistreerd.

4.1.2. Self-Sovereign

Ontology maakt gebruik van digitale handtekeningtechnologie om ervoor te zorgen dat elke entiteit haar eigen identiteit beheert. ONT ID wordt geregistreerd bij de openbare sleutel van de entiteit om het eigendom aan te geven. Het gebruik van een ONT-ID en de bijbehorende kenmerken moeten door de eigenaar digitaal worden ondertekend. Een entiteit kan het eigen gebruik van de ONT-ID bepalen, een privésleutel binden en de kenmerken ervan beheren.

4.1.3. Multi-Key Binding

Ontology ondersteunt een verscheidenheid aan nationale en internationale algoritmestandaarden voor digitale handtekeningen, waaronder RSA, ECDSA en SM2. Privésleutels die aan een ONT-ID

zijn gekoppeld, moeten het gebruikte algoritme specificeren en tegelijkertijd kan ONT ID verschillende privésleutels binden om te voldoen aan entiteitsvereisten in verschillende toepassingsscenario's.

4.1.4. Geautoriseerde controle

De eigenaar van de ONT ID kan andere ONT ID's autoriseren om managementrechten uit te oefenen over hun ONT ID. De attribuutinformatie die overeenkomt met de ONT ID kan bijvoorbeeld worden gewijzigd, of een andere private sleutel kan worden gebonden aan een ONT ID als de originele sleutel verloren is gegaan. ONT ID ondersteunt fijnmazig permissiemanagement voor elk kenmerk en meerdere toegangscontroles zoals "AND", "OF", "m van de n".

4.2. VERTROUWENSMODEL

Het vertrouwenmodel van Ontology genereert vertrouwen tussen entiteiten die zowel gecentraliseerde als gedecentraliseerde vertrouwensmodellen gebruiken. Verschillende vertrouwensmodellen kunnen worden gebruikt volgens specifieke scenario's om aan verschillende behoeften te voldoen.

4.2.1. Gecentraliseerd vertrouwensmodel

Onder dit model fungeren een of een groep entiteiten als het vertrouwensanker en wordt de vertrouwensrelatie tussen entiteiten vastgesteld op basis van het vertrouwensanker. Het vertrouwensanker specificeert de entiteiten die het vertrouwt, en de andere entiteiten kunnen op hun beurt andere entiteiten aanwijzen die zij vertrouwen. Deze indeling, met het vertrouwensanker als de bron, maakt een vertrouwensstructuur van vertrouwenslevering wordt gemaakt. Elk knooppunt in de structuur heeft een pad (het "vertrouwenspad") naar het vertrouwensanker, de vertrouwensketen. Bij interactie met andere knooppunten in de structuur kan een entiteit die het vertrouwensanker herkent, de reeks vertrouwensrelaties verifiëren.

Als het meest volwassen en meest gebruikte vertrouwensysteem is PKI [7] een gecentraliseerd vertrouwensmodel. Om een vertrouwensanker te worden moet een gebruiker eerst een digitaal certificaat aanvragen. Na goedkeuring van de aanvraag schrijft het certificatiecentrum zijn identiteitsgegevens en openbare sleutel in het digitale certificaat en voegt vervolgens de digitale handtekening van de certificeringsinstantie toe. Het digitale certificaat dat door de certificeringsinstantie is uitgegeven, verifieert de binding tussen de identiteit van de gebruiker en de openbare sleutel. Iedereen kan de openbare sleutel van het certificatiecentrum gebruiken om de echtheid van het certificaat te verifiëren en vervolgens de openbare sleutel in het certificaat gebruiken om de handtekening van de gebruiker te verifiëren en de identiteit ervan te bevestigen. Tegelijkertijd kunnen gebruikers die digitale certificaten hebben ook digitale certificaten afgeven aan andere gebruikers als ondergeschikte certificeerders, en de geldigheid van de digitale

certificaten die door hen zijn uitgegeven, wordt gegarandeerd door het certificatiecentrum. In het PKI-model moet elke deelnemer onvoorwaardelijk de certificeringsinstantie vertrouwen en de vertrouwensrelatie wordt laag voor laag van de certificaatautoriteit tussen entiteiten doorgegeven via digitale handtekeningen.

Het gecentraliseerde vertrouwensmodel heeft veel voordelen. De strikte methode voor overdracht van vertrouwensrelaties en de duidelijke vertrouwens- en vertrouwelijkheidsgrens zijn goede functies in veel scenario's en kunnen veel problemen oplossen. Er zijn echter bepaalde nadelen aan het gecentraliseerde vertrouwensmodel. De afhankelijkheid van het centrale knooppunt kan ongeschikt zijn voor complexe vertrouwensrelaties, evenals voor gevallen waarin er hogere eisen zijn aan eerlijkheid en veiligheid. Deze methode om een beroep te doen op het centrale knooppunt kan de flexibiliteit van de toepassing ernstig beperken.

4.2.2. Gedecentraliseerd vertrouwensmodel

Naast het vertrouwen op specifieke centrale entiteiten om vertrouwensrelaties op te bouwen, kunnen entiteiten zelf even sterke vertrouwensrelaties opbouwen. Vertrouwensoverdracht wordt bereikt door wederzijdse authenticatie tussen entiteiten. Entiteiten zullen een hogere geloofwaardigheid hebben als ze meer authenticaties van andere entiteiten ontvangen - vooral als die andere entiteiten een hoge geloofwaardigheid hebben.

Het gedecentraliseerde vertrouwenmodel is een wazig vertrouwensmodel en er is geen duidelijke vertrouwenslimiet. Onder verschillende scenario's kunnen verschillende betrouwbaarheidsbeoordelingsmethoden worden gebruikt om het vertrouwen van entiteiten te evalueren. Het is vanwege deze hoge mate van flexibiliteit dat het model in het echte leven een breed scala aan toepassingen heeft.

4.3. VERIFIEERBARE CLAIM

Verifieerbare claims worden gebruikt om bepaalde kenmerken van een entiteit te bewijzen. Ze kunnen worden opgeslagen en verzonden als gegevenseenheden en worden gecontroleerd door een entiteit. Een claim bevat metagegevens, claimcontent en de handtekening van de uitgever, terwijl de inhoud gegevens kan zijn. Claims zijn geformatteerd in JSON-LD [8], met handtekeningen volgens de LD-handtekening specificatie [9].

4.3.1. Levenscyclus

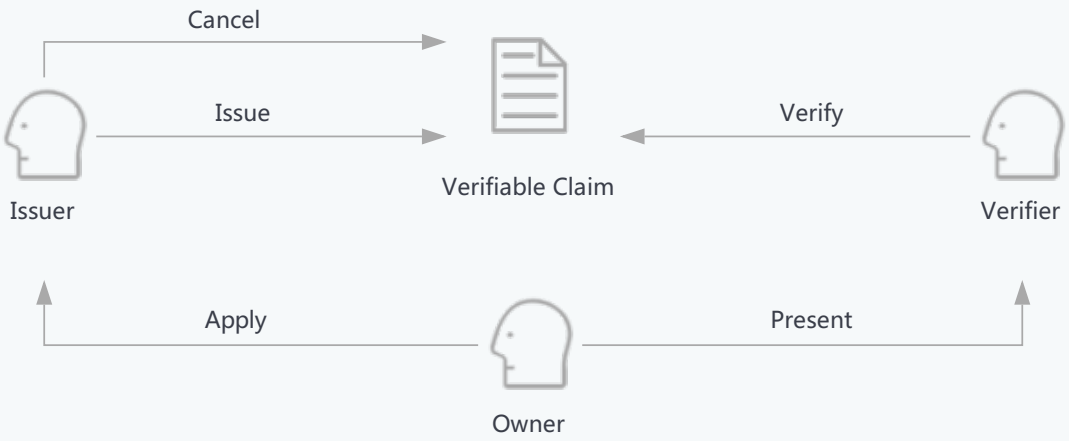

Figure 4.2: Levenscyclus van een verifieërbare claim

Entiteiten die geassocieerd zijn met een verifieerbare claim vallen in drie categorieën: uitgever, houder en certificeerder. De levenscyclus van een verifieerbare claim omvat de volgende vijf bewerkingen:

- *Uitgifte: elke entiteit kan een verifieerbare claim van een attribuut van een andere entiteit afgeven. Een school kan bijvoorbeeld een student een transcriptie geven door een verifieerbare claim uit te reiken. Wanneer een verifieerbare claim wordt gebruikt, kan een geldigheidsperiode worden ingesteld. Wanneer de geldigheidsperiode is verstreken, vervalt de claim automatisch.*
- *Opslag: Verifieerbare claims kunnen worden afgegeven als openbare claims of privéclaims. Openbare claims worden opgeslagen in een gedistribueerd grootboek in Ontology, terwijl privéclaims doorgaans worden opgeslagen op de client van de entiteit en beheerd door de entiteit zelf.Presenteren: de eigenaar van de verifieerbare claim kan kiezen aan wie de claim openbaar wordt gemaakt en welke informatie wordt getoond zonder de integriteit van de claim te beïnvloeden.*
- *Verificatie: de verificatie van de verifieerbare claim hoeft geen interactie te hebben met de verstrekker van de claim, maar hoeft alleen de ONT ID van de uitgever te gebruiken om de openbare sleutelinformatie van het gedistribueerde grootboek van Ontology te verkrijgen. Vervolgens kan het de openbare sleutel gebruiken om de digitale handtekening van de claim te verifiëren.*
- *Annulering: de uitgever van de verifieerbare claim kan zijn claim annuleren vóór de vervaldatum. De geannuleerde claim kan niet worden gevalideerd.*

4.3.2. Anonieme Claim

In normale omstandigheden onthult de claimeigenaar de volledige inhoud van de claim aan de verificateur wanneer deze een claim indient. In sommige gevallen kan het echter zijn dat de claimeigenaar bepaalde inhoud van de claim niet aan de verificateur wil laten zien. In het licht

hiervan biedt Ontology's anonieme verifieerbare claimtechnologie om de privacy van zijn gebruikers te beschermen.

De anonieme claimtechnologie lost het probleem op van het verbergen van de informatie van de houder tijdens het proces van het afgeven en presenteren van een claim. In het anonieme claimprotocol ontvangt een entiteit twee verificaties van hun claim van twee verschillende verificateurs. Zelfs als de twee verificateurs samen wilden samenspannen om de informatie die ze bevatten, te lekken, zouden ze niet kunnen verifiëren of de informatie die ze hebben ontvangen afkomstig is van dezelfde entiteit. Bij het indienen van een anonieme claim hoeft de verstrekker de oorspronkelijke claim niet aan de verificateur te verstrekken, maar deze hoeft alleen een bewijs van nulkennis te leveren. De verificateur kan de echtheid van de claim verifiëren door een validatie-algoritme uit te voeren met de openbare sleutel van de uitgever, het certificaat en een bewering van de attribuutwaarden in het certificaat, bijvoorbeeld "Leeftijd> 18" EN "inwoner van Shanghai".

Een anonieme claim is meestal een XML- of JSON-bestand dat zowel openbare als cryptografische informatie bevat. De openbare informatie omvat alle attributen van de anonieme claim, die uit drie delen bestaat: de naam van het attribuut, het type van het attribuut en de waarde van het attribuut. Attributen ondersteunen een verscheidenheid aan gegevenstypen, zoals strings, gehele getallen, datums en opsommingstypen. De cryptografische informatie omvat hoofdzakelijk de eigen master key van de eigenaar en de digitale handtekening van de uitgever.

Tijdens de presentatie van de anonieme verifieerbare claim bewijst de eigenaar aan de derde partij dat hij een anonieme claim van een emittent bezit. Ze kunnen selectief een aantal attribuutwaarden blootleggen en andere attribuutwaarden verbergen. Bovendien kunnen ze bewijzen dat sommige verborgen kenmerken voldoen aan bepaalde logische beweringen.

Anonieme claims gebruiken het CL-handtekeningschema [10] en Σ-protocol [11] om de bovengenoemde functies te bereiken.

4.3.2.1. CL Signature Scheme

Het CL-handtekeningschema bestaat uit drie algoritmen, namelijk het sleutelgeneratiealgoritme, het handtekeninggeneratiealgoritme en het handtekeningverificatiealgoritme. Dit schema is aantoonbaar veilig onder de sterke RSA-aanname.

Key generatie GEN:

1) *Kies twee willekeurige beveiligde priemgetallen* (p, q) *en bereken* $n = pq$.

2) *Kies* $k + 2$ *willekeurige kwadratische residu* $(R_1, \ldots, R_k, S, Z)$.

3) *Output de private key* $sk = (p, q)$, *en de public key* $pk = (n, R_1, \ldots, R_k, S, Z)$.

Signatuur generatie $SIGN_{sk}(\{m_i\})$:

Input k values $\{m_1, \ldots, m_k\}$.

1) *Bereken de Euler totient functie* $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$.

2) *Kies een grote prime e en een groot geheel getal v willekeurig.*

3) *Bereken de inverse van e module* $\varphi(n)$*, aangeduid als* e^{-1}*, wat voldoet*

$$e \cdot e^{-1} \equiv 1 \pmod{\varphi(n)}$$

4) *Bereken een geheel getal*

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

5) *Output signatuur* (A, e, v).

Signatuur verificatiie $VERIFY_{pk}(\{m_i\}, A, e, v)$:

Input k values $\{m_1, \ldots, m_k\}$ *en de signatuur* (A, e, v).

1) *Check ofe en v zijn in het opgegeven interval en e is een prime.*

2) *Check of de signatuur voldoet:*

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

4.3.2.2. Σ-protocollen

Σ-protocollen zijn efficiënte zero-knowledge proof-protocollen, die kunnen worden gebruikt door een prover$P$ om een verificateur te overtuigen V dat hij een geheim kent, zonder het geheim of enige andere informatie over te brengen. Er zijn enkele bekende Σ-protocollen, b.v. Heuristiek van Fiat-Shamir [12] en Schnorr-handtekening [13].

Uitgaande dat P een oplossing kent voor het volgende systeem van vergelijkingen $y_1 = g^x,\ y_2 = h^x$ dan kan hij het volgende sigma-protocol gebruiken dat in de standaardnotatie is geschreven als:

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

Het kan worden bewezen met behulp van het volgende interactieve protocol:

1) *P kiest een willekeurig geheel getal r.*

2) *P berekent* (g^r, h^r) *en stuurt die naar V.*

3) *V kiest een willekeurig geheel getal c en stuurt die naar P.*

4) *P berekent* $s = r - c \cdot x$ *en stuurt s naar V.*

5) *V verifieert of s voldoet aan de volgende vergelijkingen*

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

4.3.2.3. Anonieme Claim Issuance

Een anonieme claim indienen bij de ontvanger R, de emittent I en ontvanger R moeten een interactief twee-ronde protocol uitvoeren als volgt:

1) *I kiest een willekeurig geheel getal n_1 en stuurt dat naar R.*
2) *R kiest een willekeurig geheel getal v' en berekent $U = R_1^{m_0} S^{v'} \pmod{n}$, waar m_0 is het master secret. Dan R stuurt U naar I.*
3) *I kiest een prime e en een geheel getal v'' willekeurig.*
4) *I berekent CL signatuur (A, e, v'') op de R's publieke attributen $\{m_i\}$ en verzend het naar R.*
5) *R berekent $v = v' + v''$ en bewaart de uiteindelijke anonieme claim (A, e, v).*

De ontvanger krijgt een CL-handtekening op zijn publieke attribuutwaarden van de uitgever door dit protocol uit te voeren. De CL-handtekening vormt de cryptografische informatie van een anonieme referentie.

4.3.2.4. Presentatie en Verificatie

De houder van een anonieme claim kan selectief bepaalde attributen aan de verificateur bekendmaken en de anderen privé houden. Het anonieme, verifieerbare claimschema maakt het ook mogelijk om de juistheid van bepaalde beweringen te bewijzen.

Om de niet-geopenbaarde kenmerken privé te houden, moet de houder zijn kennis van de niet-onthulde kenmerken aantonen. Stel dat er l attributen zijn $\mathbb{H}_l$ bewaard privé onder k attributen $\{m_1, \ldots, m_k\}$. Het bewijs kan als volgt worden geconstrueerd:

1) *Kies een willekeurig geheel getal r_A en gebruik het om de CL-handtekening willekeurig te maken*

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

2) *Bouw het bewijs met behulp van het protocol beschreven in sectie 4.3.2.2:*

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

3) *Output het bewijs π.*

Om het bewijs van een predikaat te illustreren, laten we zien hoe we het ongelijkheidspredicaat van sommige privéattributen kunnen bewijzen. Een predikaat groter dan de ongelijkheid specificeert een attribuutwaarde m, en ondergrens b. Hier geven we een protocol om dat te laten zien $m \geq b$ zonder dat het daadwerkelijk lekt m.

1) *Bereken $\Delta = m - b$ en ontbind het in de som van vierkanten van voor gehele getallen:*

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

2) *Genereer vier willekeurig getallen T_{u_i} om deze vier nummers te verbergen. Construeer vervolgens de proof*

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

3) *Kies een willekeurig getal T_Δ om het verschil te verbergen Δ, en construeer de proof*

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

4) *Construeer de proof voor $m \geq b$:*

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

5) *Output* (π_1, π_2, π_3).

Combineer π en (π_1, π_2, π_3) om het volledige bewijs te verkrijgen. De verificateur kan de verificatiemethode gebruiken die wordt beschreven in sectie 4.3.2.2 om elke subproof te controleren. Het predikaat is waar en alleen als alle subproofs geldig zijn.

5. DISTRIBUTED LEDGER

5.1. ONTOLOGY LEDGER

5.1.1. Consensus Mechanism

Ontology Ledger supports the next generation Ontorand Consensus Engine (OCE). OCE is an efficient, consensus-based version of the dBFT consensus protocol and verifiable random function (VRF) consensus engine. It has reached near-infinite scalability and requires a relatively low hashing rate, making network forks highly unlikely. dBFT has demonstrated excellent stability and reliability during its operation on NEO's public chain and other affiliated blockchain projects. OCE's block-creation speed is only limited to internet speed, usually resulting in confirmations within 10 seconds. OCE selects who will participate in consensus validation using a verifiable random function and reaches consensus by using a Byzantine fault tolerance algorithm[14][15][16]. Meanwhile, the seed will be generated by signatures of validator group and direct to the next validator group. OCE also supports pluggable verifiers and online protocol recovery and upgrade.

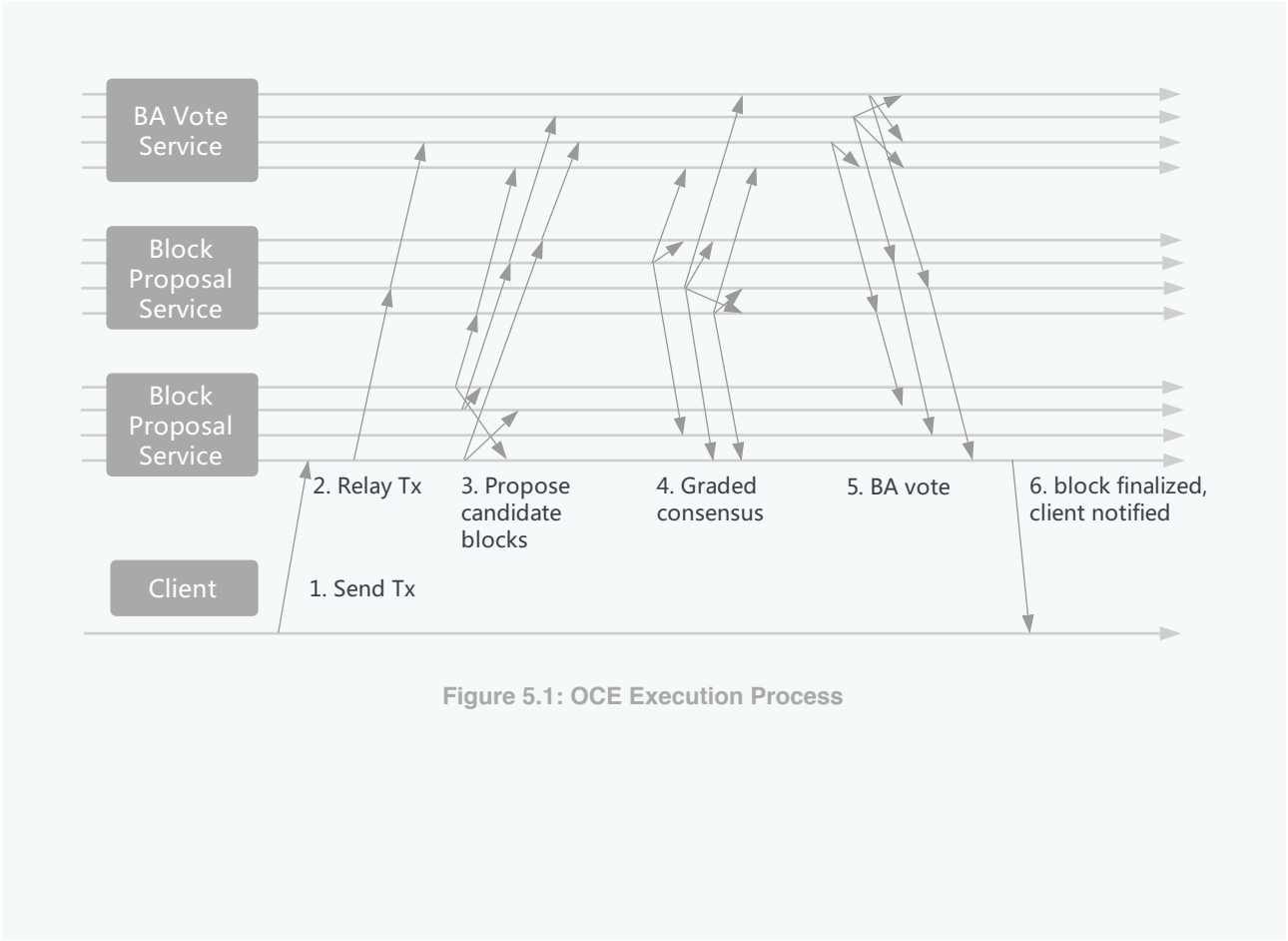

Figure 5.1: OCE Execution Process

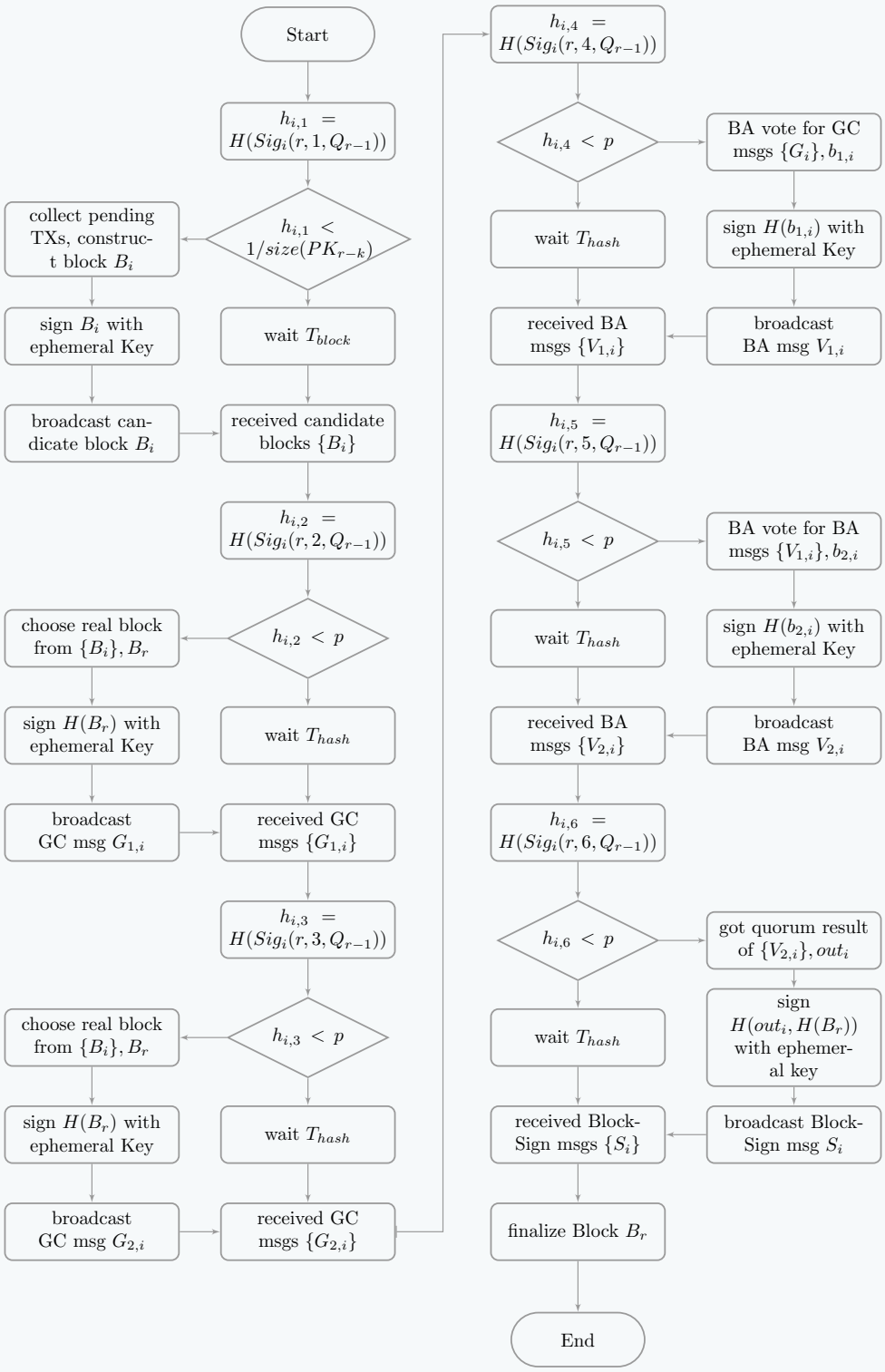

Figure 5.2: OCE Flow Chart

De verwerkingsstappen en workflow van OCE worden weergegeven in figuur 5.1 en figuur 5.2.

1) *De cliënt zendt een verzoek tot gegevensuitwisseling uit via het P2P-netwerk met de handtekening*
2) *(signatuur) van de aanvrager.*
3) *Alle knooppunten die deelnemen aan het consensusproces kunnen alle transacties in het netwerk bewaken en opslaan in een lokale cache.*
4) *Een nieuw blok voorstellen:*
    a) *Nodes kunnen een handtekening genereren op basis van de huidige blokhoogte, evenals de Q-waarde van het laatste blok, en de bijbehorende hash-waarde berekenen. De hash-waarde kan dan worden gebruikt om te bepalen of de node een potentiële aanbieder van de huidige blokhoogte kan zijn.*
    b) *Als een node heeft geëvalueerd dat het een voorstel kan zijn voor de huidige blokhoogte:*
        i) *Het verpakt huidige en verzamelde niet-consensus-transactieverzoeken en construeert een nieuw blok.*
        ii) *Het maakt een handtekening op basis van de gegevens- en hash-waarde van het nieuwe blok en verzendt het blok en de handtekening naar het volledige P2P-netwerk.*
5) *Alle nodes wachten op T_{block} bij het alleen monitoren van netwerkverkeer voor blokvoorstellen en de ontvangen blokvoorstellen tijdelijk in het lokale geheugen opslaan.*
6) *Nieuwe blok-genivelleerde consensus:*
    a) *Nadat T_{block} elke node in het netwerk evalueert of dit de verificatienode voor het huidige blokvoorstel zou kunnen zijn.*
    b) *Als een node zichzelf als een verificatienode van het huidige blokvoorstel heeft geëvalueerd, doet het het volgende:*
        i) *Controleert de juistheid en integriteit van alle blokvoorstellen.*
        ii) *Verifieert de juistheid van de Q-waardehandtekening van alle blokvoorstellen.*
        iii) *Controleert de wettigheid van de identiteit van de blokaanbieder.*
        iv) *Berekent de hash-waarde op basis van de Q-waardehandtekening en selecteert het blokvoorstel met de kleinste hash-waarde als het werkelijk gekozen blok door het huidige blok.*
        v) *Berekent de hash-waarde op basis van de Q-waardehandtekening en selecteert het blokvoorstel met de kleinste hash-waarde als het werkelijk gekozen blok door het huidige blok.*
    c) *Nadat T_{hash}, het knooppunt de stap van het algoritme heeft bijgewerkt en beoordeelt of deze zelf moet deelnemen aan het verificatieproces van het blokvoorstel tijdens deze stap in het proces.*
    d) *Als een knooppunt zichzelf heeft beoordeeld als een verificatieknooppunt voor het blokvoorstel in de huidige stap, doet het het volgende:*
        i) *Verifieert het blokvoorstel op dezelfde manier als hierboven beschreven.*
        ii) *Genereert de hash-waardehandtekening van het geselecteerde blokvoorstel en zendt de blokhashwaarde en handtekening naar het P2P-netwerk.*
7) *Alle knooppunten op het P2P-netwerk bewaken consensusberichten op zichzelf die werden uitgezonden door blokverificatieknooppunten.*

a) *Controleer de juistheid en integriteit van consensusberichten.*

b) *Controleer de geldigheid van de identiteit van het knooppunt dat het consensusbericht uitzendt.*

c) *Sla het verificatiebericht tijdelijk op in het lokale geheugen.*

8) *Byzantijnse verkiezing voor het nieuwe blok.*

a) *Nadat T_{hash}, knooppunten de algoritmestap hebben bijgewerkt en zelf evalueren of een van hen zich zou moeten bezighouden met de verkiezing van een nieuw blok.*

b) *Als een knooppunt bepaalt dat het moet deelnemen aan de verkiezing, voert het de volgende stappen uit:*

i) *Aantal ontvangen peilingen van consensusverificatie van verschillende blokvoorstellen.*

ii) *Stemmen op basis van het resultaat van consensusresultaat als een quorum is bereikt.*

iii) *Ondertekent de stem, berekent de hash van het geselecteerde blok en zendt de stemming uit evenals zijn handtekening naar het P2P-netwerk*

c) *Alle knooppunten op het netwerk zullen meerdere stemrondes uitbrengen volgens de configuratie van hun blockchain. Elke ronde wacht op de T_{hash} interval en werkt onafhankelijk.*

9) *Alle knooppunten op het netwerk controleren en controleren continu de geldigheid van stemmingsberichten die worden uitgezonden op het P2P-netwerk.*

10) *Het genereren van de definitieve handtekening van een nieuw blok:*

a) *Na de laatste ronde van byzantijnse verkiezingen, zullen knooppunten de algoritmestap bijwerken en beoordelen of een van hen moet samenwerken om de definitieve handtekening van een nieuw blok te maken.*

b) *Als een knooppunt heeft geëvalueerd dat het moet meewerken aan het maken van de definitieve handtekening, voert het de volgende stappen uit:*

i) *Volgens de telling van de stemmen, beoordeelt het of het blokvoorstel een quorum gesteunde consensus heeft bereikt.*

ii) *Als er geen door consensus gesteunde consensus wordt bereikt, wordt het consensusresultaat van het nieuwe blok als onwaar beschouwd.*

iii) *Pakt het pollbericht in op basis van de vorige beoordeling en ondertekent het.*

iv) *Verstuurt de laatste blokhash en de ondertekening van het pollingresultaat naar het P2P-netwerk.*

11) *Alle knooppunten controleren continu het laatste handtekeningbericht van nieuwe blokken in het P2P-netwerk en controleren de geldigheid van handtekeningberichten:*

a) *Nadat T_{hash}, knooppunten berekenen de blokhash van de huidige blokhoogte in overeenstemming met het ontvangen bloksignatuurbericht.*

b) *Knopen vergelijken de berekende blokhash met het eerder ontvangen blokvoorstelbericht om het laatste blok voor de huidige blokhoogte te krijgen.*

Ondertussen wordt de Q-waarde van het blok berekend op basis van het laatste blok van de consensus en initialiseert het consensusproces van het blok voor de volgende blokhoogte.

Ontology Ledger neemt een gemodulariseerde architectuur aan, die een pluggable consensusalgoritme ondersteunt dat gemakkelijk en snel kan worden vervangen door andere consensusalgoritmen, bijvoorbeeld PoS, DPoS en Raft, volgens verschillende scenario's.

5.1.2. Procedure Protocols

Protocollen met gedistribueerde procedures (gedistribueerde transacties) worden bereikt door middel van gedistribueerde grootboektechnologie, cross-chain-systemen van entiteiten en systeemoverschrijdende privacy en specifieke protocollen met verschillende ketens. De identiteits-privacy van entiteiten wordt beschermd, terwijl procedure / transactie-stappen werken op verschillende blockchains of systemen om de consistentie van de gehele transactie te waarborgen.

5.1.3. Attestation Design

Niet alleen gegevens, maar ook het gedrag van gegevens wordt aan het gedistribueerde grootboek bevestigd. Dit betekent met elke gegevensaanvraag dat gegevensvergelijking, gegevensopname en gegevensgebruik worden vastgelegd in het grootboek, waardoor een record wordt vastgelegd van het gehele gegevensproces en de gegevensbeveiliging en betrouwbaarheid worden gegarandeerd.

5.2. SMART CONTRACT

Ontology Ledger's NeoVM virtuele machine, geschreven in Go, wordt gebruikt voor het uitvoeren van slimme contracten en kan intelligente besturingslogica implementeren in het applicatielaag-framework van Ontology. De Turing-volledigheid van de NeoVM-virtuele machine maakt willekeurige logica en een hoge mate van zekerheid mogelijk, wat ervoor zorgt dat dezelfde ingangen altijd dezelfde outputs produceren, waardoor de onzekerheid van de container / docker wordt vermeden. Het is geschikt voor gebruik in scenario's waarbij zekerheid een sterke vereiste is. Bovendien beschikt NeoVM over een hoge schaalbaarheid door middel van "deterministische call tree" -technologie, waarmee dynamische scherf kan worden bereikt, wat in theorie onbeperkte uitbreidingsmogelijkheden betekent.

Bovendien kunnen met behulp van onafhankelijk ontwikkelde compilers andere compilers, waaronder Java Bytecode en C # MSIL, samenwerken met de virtuele blockchain-machines. Dit bevordert de deelname van slimme contractontwikkelaars, waardoor ze slimme contracten kunnen schrijven met behulp van Java, C / C #, Go en andere programmeertalen zonder een nieuwe taal te hoeven leren. Het gebruiksgemak en onderhoud van het platform maakt het eenvoudig deelbaar met andere partners (met name informatieproviders), en biedt hen de mogelijkheid om het contract slim te bewerken en toegankelijk te maken.

De virtuele NeoVM-machine combineert hoge-toptaalanalyse en conversie. De externe interface van de virtuele machine kan worden aangepast met API's, die flexibel kunnen communiceren met account- en externe gegevens. Dit zorgt voor hoge prestaties van native code-uitvoering wanneer slimme contracten worden uitgevoerd. Tegelijkertijd is een mechanisme voor virtuele machines geïmplementeerd voor het ondersteunen van verschillende blockchains.

5.3. SHARED DATA CONTRACT MODEL

Applicatie-eisen van het gedistribueerde grootboek omvatten hoofdzakelijk twee functies: de eerste is de definitie en opslag van datastructuren; de andere is business logic-verwerking en interactie met externe systemen. We hebben een model ontworpen dat deze twee delen scheidt om het systeem een betere schaalbaarheid en flexibiliteit te geven. Een deel is voor gegevensopslag (het gegevenscontract) en het andere deel voor de bedrijfslogica (het controllercontract). We noemen dit het "Shared Data Contract Model".

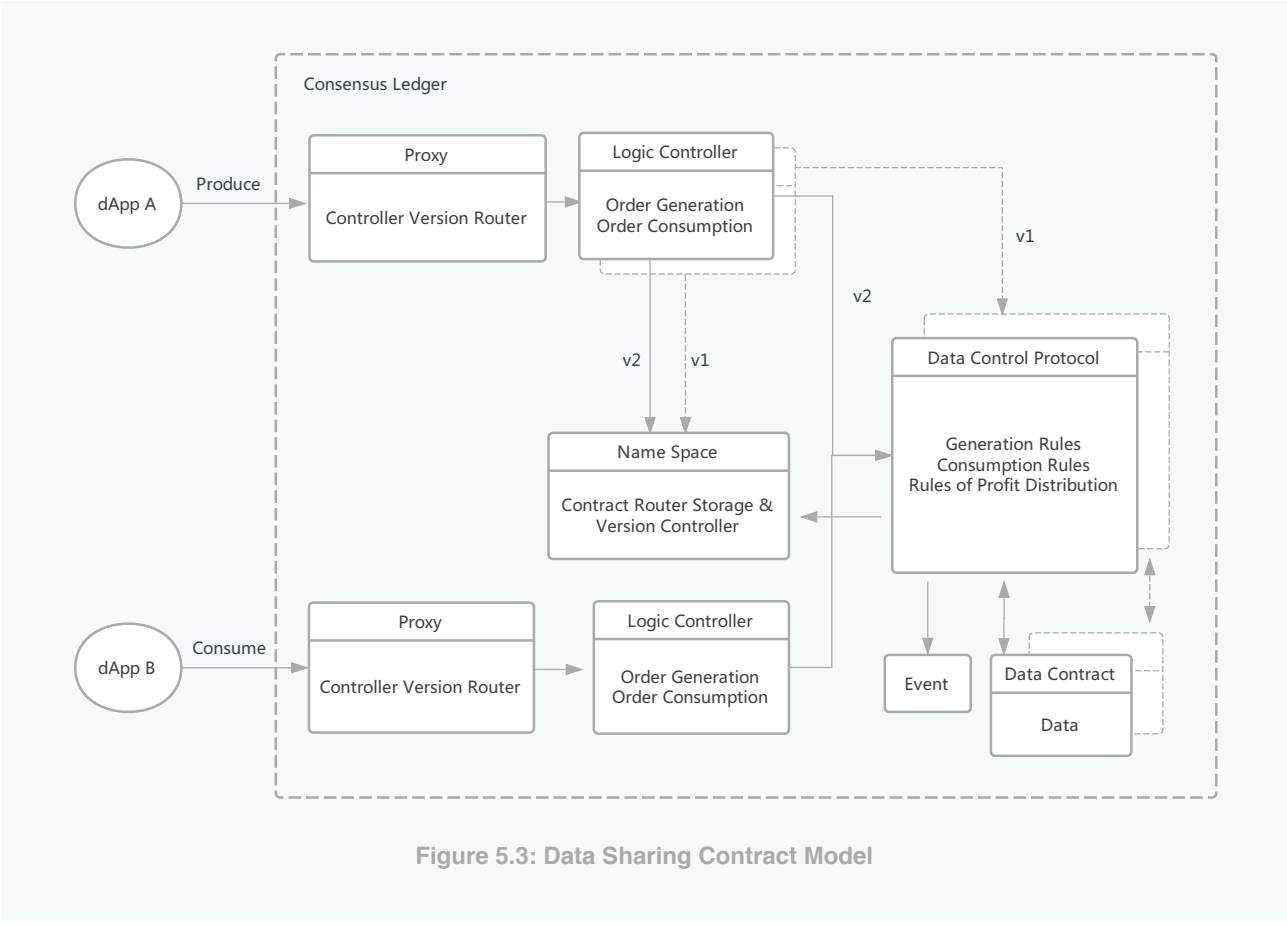

Figure 5.3: Data Sharing Contract Model

Het model bevat de volgende ontwerpcomponenten:

- *Agentcontroller: een externe door dApp gedefinieerde contractvermelding voor andere dApp's verstrekken, die garandeert dat, zelfs als het contract wordt bijgewerkt, dit geen inconsistenties in externe overdrachten zal veroorzaken.*
- *Logische controller: voor besturing van bedrijfslogica.*
- *Namespaces: toewijzing van gegevenscontractadressen voor verschillende versies van verschillende dApps. Het zorgt ervoor dat upgrades van de gegevensstructuur geen invloed hebben op bedrijfslogica en kan traceerbaarheid van gegevens tussen verschillende versies ondersteunen.*
- *Gegevenscontract: biedt een basisgegevensstructuur en opslaginterface. Vergelijkbaar met DAO, met behulp van GET / SET-methoden.*
- *Protocol voor gegevensbeheer: algemene bedrijfsregels die zijn gedefinieerd door aan bedrijven gelieerde partijen, worden gedigitaliseerd in protocolcontrollers. Dit protocol moet de volgende componenten bevatten:*
  - *Maak verbinding met de permissieregeling van de logische controller*
  - *Interactie met de interface van externe systemen*
  - *Gemeenschappelijke servicelogica*
  - *Besturingslogica voor het bedrijfsgrootboek*
  - *Event notificatie mechanisme*
- *Gebeurtenis: een gebeurtenis verwijst naar een mechanisme voor het verwijderen van inhoud tijdens de uitvoering van een slim contract. Wanneer elke partij bezig is met het synchroniseren van het grootboek, kunnen ze transactie-informatie krijgen door het gedistribueerde grootboekblok lokaal terug te spelen. De gebeurtenis is een betaalmechanisme met lage kosten en knooppunten die niet om het contractknooppunt geven, kunnen ervoor kiezen om het contract niet uit te voeren en niet om de evenementinformatie te ontvangen. Knooppunten die overeenkomen met het slimme contract kunnen de gebeurtenisinformatie ontvangen door uitvoering van het contract. Dit kan de opslagdruk op elk knooppunt aanzienlijk verminderen, omdat de ketting de daadwerkelijke transactie-inhoud niet opslaat.*

Met dit contractmodel kunnen dApps hun gegevens met elkaar delen met behulp van generieke protocollen, ondanks hun bedrijfsactiviteiten. Het maakt ook samenwerking over grenzen heen veel eenvoudiger.

Het proces van orderregistratie en -uitwisseling wordt getoond in figuur 5.4.

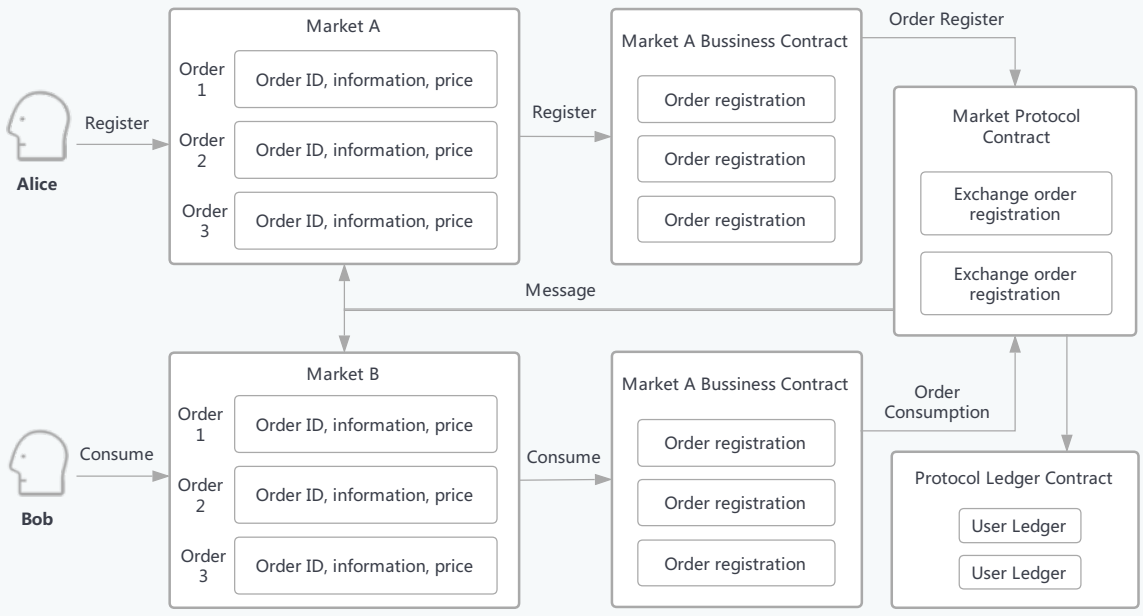

Figure 5.4: Data Exchange Process

Registratie:

1) *Alice initieert een opdrachtregistratieverzoek voor markt A;*
2) *Market A controleert de bestelling van Alice en volgens de classificatie registreert het protocol voor het classificatieprotocol de bestelling;*
3) *Het contract met het marktprotocol onderzoekt de autoriteit van markt A en de inhoud van de bestelling. Als aan de criteria is voldaan, registreert het contract de bestelling naar het kernboek van het protocol;*
4) *Nadat de registratie geslaagd is, zendt het protocolcontract het orderregistratiebericht uit, waarbij het resultaat wordt geretourneerd;*
5) *Het zakelijk protocolcontract van Markt A retourneert het eindresultaat aan Alice.*

Transactie:

1) *Market B verkrijgt alle transacties door te synchroniseren met de blockchain.*
2) *Markt B verkrijgt het orderregistratiebericht uitgezonden door markt A door de transactie in het uitgevoerde blok.*
3) *De bestelinformatie in het orderregistratiebericht wordt weergegeven in Market B.*
4) *Bob initieert de transactie naar Markt B voor de bestelling die Alice hierboven registreerde;*
5) *Markt B controleert het transactieaanvraag van Bob en volgens de classificatie start het protocol de transactie.*
6) *Het protocolcontract inspecteert de machtigingen en de transactieverzoekinformatie van de markt B. Als aan de vereisten is voldaan, worden de transactieaanvragen van B behandeld en wordt de vereffening uitgevoerd.*

7) *Het protocol verzendt de transactie-voltooiingsinformatie van de order en retourneert de resultaten naar markt B.*

8) *Markt B levert het eindresultaat aan Bob.*

9) *Markt A ontvangt de informatie over het ontvangen van protocolcontractafrondingen en verzendt vervolgens de transactievoltooiingsinformatie naar Alice.*

5.4. MERKLE TREE OPSLAG MODEL

Voor digitale valutaplatformen zoals Bitcoin en Ethereum, richt de klant zich meestal alleen op zijn eigen accountstatus. Om de accountstatus te verifiëren zonder een volledige synchronisatie van het grootboek, wordt de SPV-technologie (simple payment verification) voorgesteld, die de opslagruimte aanzienlijk kan besparen en de netwerkoverdracht kan verminderen door Merkle-bewijzen te construeren [17]. In Ontology moet de client ook de identiteitsinformatie van andere klanten verifiëren. Daarom moet het grootboek merkle-proofconstructie ondersteunen.

5.4.1. Merkle Hash Tree

Een binaire Merkle Hash Tree [18] [19] wordt gebruikt voor efficiënte auditing. Gegeven een geordende lijst met n inputs $D[n] = (d_0, d_1, \ldots, d_{n-1})$,de Merkle Tree Hash (MTH) wordt als volgt gedefinieerd:

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

Waar k is de grootste macht van 2 kleiner dan n, $D[a:b]$ is de sublijst van D from d_a to d_{b-1}, $\|$ is de aaneenschakeling van de twee byte-arrays.

5.4.2. Merkle Controlepad

Een Merkle audit pad voor een blad in een Merkle Hash Tree is de kortste lijst van extra knooppunten in de Merkle Tree die nodig is om de Merkle Tree Hash voor die tree te berekenen. Elk knooppunt in de structuur is een bladknooppunt of wordt berekend uit de twee knooppunten direct eronder (d.w.z. naar de bladeren toe). Bij elke stap van de tree (naar de root) wordt een knooppunt van het auditpad gecombineerd met het tot nu toe berekende knooppunt. Met andere woorden, het controlepad bestaat uit de lijst met ontbrekende knooppunten die nodig zijn om de knooppunten te berekenen die van een blad naar de hoofdmap van de tree leiden. Als de root die is berekend op basis van het auditpad overeenkomt met de ware root, is het controlepad het bewijs dat het leaf in de structuur bestaat.

Gegeven een geordende lijst van n inputs naar de tree, $D[n] = (d_0, d_1, \ldots, d_{n-1})$, het Merkle controlepad $PATH(m, D[n])$ voor de (m+1)th input $d(m) : 0 \leq m < n$ is als volgende gedefineerd:

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

Waar + een aaneenschakeling van lijsten is.

Figuur 5.5 toont een voorbeeld van Merkle audit-pad.

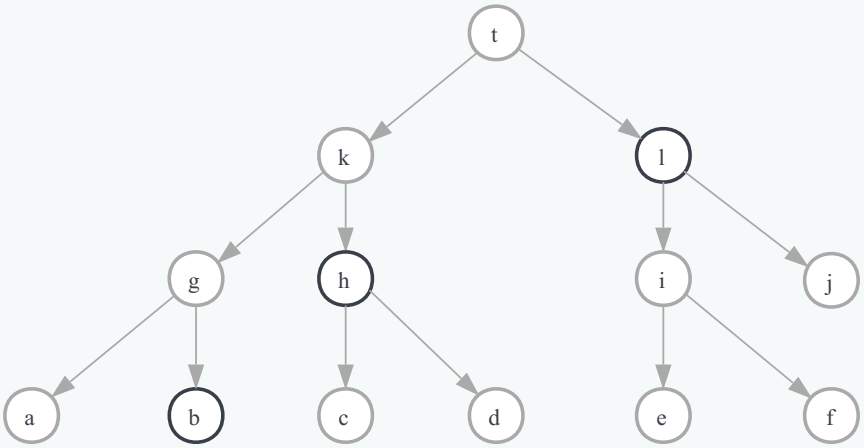

Figure 5.5: Audit Path of a is [b, h, l]

5.4.3. Merkle Consistentie Bewijzen

Merkle-consistentiebewijzen bewijzen de alleen-toevoegende eigenschap van de tree. Een Merkle-consistentiebewijs voor een Merkle Tree Hash $MTH(D[n])$ en een eerder geadverteerde hash $MTH(D[0:m])$ van de eerste m bladeren($m \leq n$), is de lijst met knooppunten in de Merkle Tree die nodig is om te verifiëren dat de eerste m-inputs D [0: m] in beide trees gelijk zijn. Het volgende algoritme kan het (unieke) minimale consistentiebewijs construeren.

Gegeven een geordende lijst van n inputs voor de tree, $D[n] = (d_0, d_1, \ldots, d_{n-1})$, wordt het consistentiebewijs van Merkle $PROOF(m, D[n])$ voor een eerdere Merkle Tree Hash $MTH(D[0:m])$ gedefinieerd als:

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

Figure 5.6 is een voorbeeld van Merkle consistentie-bewijs.

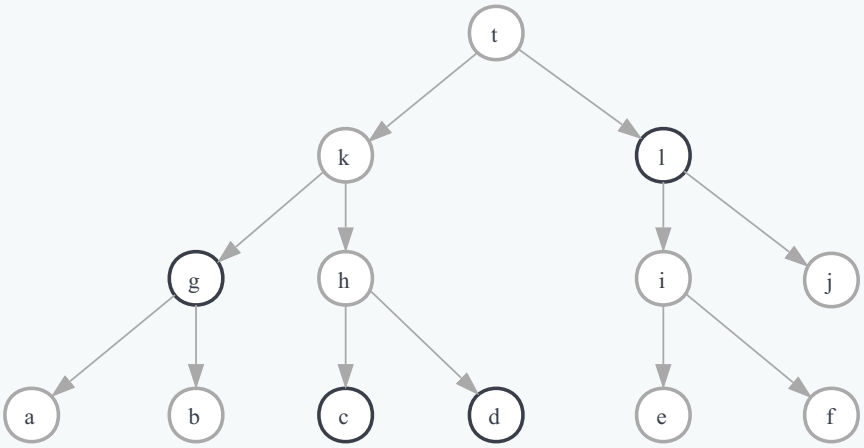

Figure 5.6: $PROOF(3, D[7])$ **is [c,d,g,l]**

5.4.4. Merkle Patricia Tree

In sommige scenario's in Ontology moeten we snel de uiteindelijke status van een transactie voor een bepaalde entiteit bewijzen, bijvoorbeeld om de identiteit van een entiteit te bewijzen. Het gebruik van een Merkle-bewijs vereist dat elke historische transactie één voor één wordt getest en dat met behulp van een Merkle Patricia Tree (MPT) [20] efficiëntie aanzienlijk kan worden verbeterd.

MPT is de combinatie van Patricia Tree [21] en Merkle Tree. Het bevat een key-value mapping-relatie, biedt een anti-tamper datastructuur op basis van cryptografie en heeft determinatie, hoge efficiëntie en beveiliging.

- *Determinatie: bij het opzoeken van gegevens zorgen dezelfde sleutelwaarden ervoor dat hetzelfde resultaat wordt gevonden en dezelfde root-hash.*
- *Efficiëntie: nieuwe wortels kunnen snel worden berekend. Wanneer gegevens worden gewijzigd, is de tijdcomplexiteit van invoegen, zoeken, bijwerken en verwijderen is* $O(\log_2 n)$.
- *Beveiliging: wanneer iemand een ander met een DOS-aanval kwaadwillig aanvalt en de controle probeert te krijgen over de diepte van de tree. Beperkte tree-diepte maakt dit soort aanvallen onbereikbaar.*

5.5. HYDRADAO

HydraDAO is een gegevensvoorspelling en interactiemodule die smart contracts, cross-chain en cross-data source-samenwerking integreert. Het bevat de DAO (distributed autonome organisatie) en de cross-chain data-interactie (big data / AI) -functies van Ontology. Het bestuursmechanisme van Ontology ondersteunt democratische en AI-geautomatiseerde proposities, stemmingen en verificaties. Tijdens het proces worden een uniek DAO-adres en polling-token gemaakt, waarmee DAO automatisch fondsen en resultaten aan Ontology kan toevoegen. Zodra het pollen is voltooid, voert DAO autonoom uit in overeenstemming met het fraudebestendige slimme contract. Het mechanisme maakt gegevensuitwisseling en governance in Ontology mogelijk met flexibiliteit en technologie voor grootschalige geautomatiseerde netwerkbewerkingen.

De Oracle Smart Contract implementatielogica van Ontology ziet er als volgt uit::

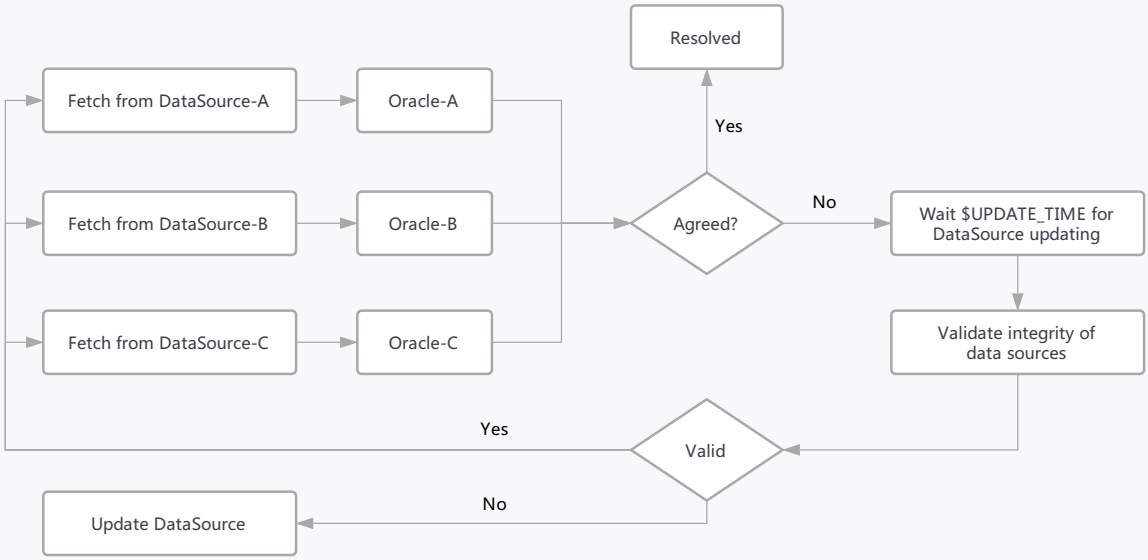

Figure 5.7: Processing Logic of HydraDAO

1) *Wanneer een transactie aan het begin of einde wordt uitgevoerd, is het mogelijk om te kiezen voor het Oracle Smart-contract. Oracle smart-contracten en hun parameters moeten worden ingesteld op het moment dat de transactie wordt gemaakt en op de blockchain wordt opgeslagen als onderdeel van de transactie. Nadat de transactie is beëindigd, wordt de contractlogica automatisch uitgevoerd om de uitkomst te bepalen en worden tokens die op het DAO-contractadres zijn gedeponeerd, betaald aan de deelnemer die geldige gegevensinformatie heeft verstrekt. Via het Distributed Data Exchange Protocol (DDEP) van Ontology wordt real-time API-toegang tot big data-rekenkracht geleverd, samen met deelname aan de gegevensmarkt in de analyse en voorspelling van transactieresultaten om de integriteit ervan te verifiëren. Als onderdeel van de vertrouwensdatadienst kunnen transactiedeelnemers vertrouwde gegevensbronnen binden of gebruikmaken van Ontology-services om nauwkeurige transactiestateldata te valideren voordat ze aan de transactie deelnemen.*

2) *Voor transacties die alleen afhankelijk zijn van de status van de blockchain, vraagt het Oracle smart-contract de blockchain-gegevens direct na het autoriseren van de blockchain-interface.*

3) *Voor de meeste transacties die verband houden met de echte wereld, vereisen Oracle smart-contracten gegevens om real-world transactieresultaten te krijgen. Wanneer een Oracle smart-contract gegevens in de echte wereld moet openen / lezen, kiest elk hydra-orakel willekeurig een van de meerdere vertrouwensbronnen of geeft het een drempelwaarde voor het gegevensanker op. Aan het einde van de transactie wordt de globale status geverifieerd en interactief bijgewerkt via de opgegeven vertrouwde gegevensbron.Voor de meeste transacties die verband houden met de echte wereld, vereisen Oracle smart-contracten gegevens om real-world transactieresultaten te krijgen. Wanneer een Oracle smart-contract gegevens in de echte wereld moet openen / lezen, kiest elk hydra-orakel willekeurig een van de meerdere vertrouwensbronnen of geeft het een drempelwaarde voor het gegevensanker op. Aan het einde van de transactie wordt de globale status geverifieerd en interactief bijgewerkt via de opgegeven vertrouwde gegevensbron.*

5.5.1. Ingebouwde DAO Data Voorspelling

De charme van gedecentraliseerde systemen is niet alleen het verlagen van de kosten en het verbeteren van de efficiëntie, maar ook het voordeel van gedistribueerde groepsbeslissingen. Bij het uitvoeren van contracten op Ontology is er de mogelijkheid om HydraDAO aan te sluiten om "openbare testen" te activeren. Als een gebruiker bijvoorbeeld een bepaald bedrag in vreemde valuta wil kopen bij een vreemde valuta, maar de gebruiker niet zeker weet of de lopende bestelling correct is, kan hij een DAO-projectvoorstel initiëren en een bepaald bedrag van de marge als hypotheek storten. Zodra voldoende goedkeuring is verkregen, gaat het voorstel in de prognoseperiode. Elke deelnemer die het voorstel accepteert, kan een feitenverklaring en zijn handtekening toevoegen. De deelnemer kan zijn bestelling op elk moment wijzigen, maar hiervoor is een extra marge vereist. Zodra de voorspellingsperiode voorbij is, zal het DAO-contract automatisch de marge in het contract blokkeren en de beste voorstellen uitfilteren volgens de regels van het voorstel. De deelnemers die de nauwkeurigste voorspellingen hebben gedaan, worden beloond. Door de gebruiker gestarte transacties worden ook automatisch uitgevoerd in overeenstemming met de beste uitvoer.

Nadat het voorstel bij de DAO is ingediend, wordt het nieuwe voorstel geëvalueerd en vastgesteld door de token-houders van het gehele netwerk. Tijdens deze periode kunnen deelnemers feiten indienen en deze op elk moment wijzigen. Veranderende feiten kunnen ook stemming en rangorde vereisen. Het DAO-stimuleringsmechanisme voor Ontology zal ervoor zorgen dat het voorstel dat de meeste goedkeuringen ontvangt en het dichtst bij het feit ligt dat het de maximale beloning ontvangt.

HydraDAO start automatisch arbitrage nadat de prognoseperiode is voltooid. Zodra arbitrage is geactiveerd, zal HydraDAO alle inzendingen berekenen en sorteren en de DAO-contractgelden toewijzen aan de account van de ontvanger van de beloning. Om het af te maken, zal HyrdraDAO een feitstatus uitvoeren om aan het smart contract te binden.

5.5.2. Externe Vertrouwde Gegevensbronnen

Om geschillen te voorkomen, zorgen vooraf gedefinieerde configureerbare Oracle Smart-contracten ervoor dat waardeoverdracht niet plaatsvindt voordat aan de slimme contractvalidatie-voorwaarden is voldaan. Oracle smart-contracten kunnen elk JSON API-type ondersteunen voor eenvoudige definitie en ontwikkeling.

Vertrouwd toegangsproces voor gegevensbronnen:

1) *Selecteer de datalocatie, de gegevensbron, de frequentie, de vervaldatum, het API-certificaat, de privacyvoorwaarden en andere invoerparameters om Oracle smart-contracten te genereren.*

2) *Definieer het smart contract voor andere te evalueren verificateurs.*

   Beschrijvende informatie moet duidelijke instructies bevatten om de functie van het contract uit te leggen. Het label van het contract maakt het gemakkelijk voor gebruikers om het te vinden op de blockchain en verbetert de transparantie. Gedetailleerde "if ... then ..." beschrijvingen kunnen de voorwaarden uitleggen voor de uit te voeren transactie.

3) *Associatie met echte juridische documenten*

4) *Oracle smart-contracten gebruiken de elektronische handtekening van de contractuitgever om het bindende juridische document te ondertekenen. Er wordt bewijsmateriaal op de blockchain geplaatst door juridische documenten te uploaden.*

5) *Mede ondertekenaars van het contract op de hoogte brengen*

6) *Volgens de voorwaarden van het contract en de betrokken partijen worden de deelnemers die moeten ondertekenen gewaarschuwd door IM / e-mail / telefoon / enz. Het contract wordt niet uitgevoerd totdat alle ondertekenaars ondertekenen (vóór de deadline). Als de ondertekenaars niet vóór de deadline tekenen, mislukt het contract. Wanneer een deelnemer het contract ondertekent, wijst het Oracle Smart-contract de deelnemer een escrow-adres toe.*

7) *Tokens opladen voor het Oracle Smart contract escrow-adres*

8) *Het gebruik van een externe gegevensbron brengt kosten met zich mee. Gebruikers moeten vooraf het adres crediteren dat is gecreëerd door het smart contract, dat wordt beheerd door het smart contract.*

9) *Voltooi de contractrelease*

   Zodra het contract is ondertekend en de aanbetaling is voltooid, loopt het contract autonoom op Ontology. Zodra een transactie is voltooid, betaalt het smart contract van het escrow-adres fondsen volgens de contractvoorwaarden. Fondsen die niet voldoen aan de vereisten van de transactie of frauduleuze transacties worden automatisch teruggestort op de geblokkeerde rekening.

10) *(Optioneel) Data open platform, open datatoegangsprotocol*

    Voor sommige open platforms voor het delen van gegevens kunnen Oracle smart-contracten rechtstreeks communiceren met gerelateerde off-chaingegevens, op basis van verstrekte API's en protocollen voor gegevensgebruik. Op basis van de open en transparante voorwaarden worden de voorwaarden en juridische informatie van het gehele proces openbaar weergegeven, kunnen alle blockchaingebruikers opvragen / ophalen en het non-profitkarakter van dergelijke gegevens vereenvoudigt het configuratieproces voor Oracle smart-contracten. Het is niet nodig om de betalingsrelatie te onderhouden, alleen om gegevens en statusupdates te behouden.

6. KERNPROTOCOLLEN

6.1. MULTI-SOURCE AUTHENTICATIE PROTOCOL

Multi-source authenticatie verschilt van bestaande single-factor authenticatiesystemen. Ontology kan entiteiten voorzien van multi-source authenticatiesystemen die de authenticatie van externe identiteit vertrouwensbronnen en de goedkeuring van entiteiten in Ontology integreren. Niet alleen kan een entiteit informatie verstrekken over wie ze zijn, ze kunnen ook bieden wat ze bezitten, wat ze willen, welke vaardigheden ze hebben, en andere informatie gerelateerd aan hun identiteit om een uitgebreid identiteitsportfolio te creëren.

Multi-source authenticatie protocol omvat de volgende twee modi:

- *Externe vertrouwenscertificering: Ontology bindt ONT ID aan een externe vertrouwensbron met een zelfondertekende, verifieerbare claim. Elke entiteit kan de identiteit van een entiteit verifiëren door de externe vertrouwensbron die aan de ONT-ID is gebonden te verifiëren. De betrouwbaarheid van de authenticatie van een entiteit wordt bepaald door de betrouwbaarheid en acceptatie van externe vertrouwensrelaties die gebonden zijn aan de ONT ID.*
- *Authenticatie tussen Ontology-entiteiten: Entiteiten in Ontology kunnen elkaar ook verifiëren door claims onderling af te geven.*

6.1.1. Externe Vertrouwenscertificering

Externe vertrouwenscertificering ondersteunt de volgende twee methoden: zelfintroductie en importeren via vertrouwensankers.

6.1.1.1. Zelfintroductie

Gebruikers binden vertrouwen via sociale media, e-bankieren, enz., Gebruikmakend van bestaande vertrouwenssystemen. Het principe is heel eenvoudig, eerst voegt de gebruiker een bewijsadres van een externe vertrouwensbron toe aan Ontology. Vervolgens geeft de gebruiker een geloofwaardige verklaring af op het bewijsadres, waarvan het formaat als volgt is:

- *Claimcreatie en vervaltijd.*
- *Claiminhoud: inclusief het claimtype, ONT-ID, type sociale media, gebruikersnaam voor sociale media, enzovoort.*
- *Handtekening: een openbare sleutel die al in de ONT-ID staat.*

Wanneer de externe partij de externe identiteit van de gebruiker moet verifiëren, leest deze eerst het certificeringsadres van de vertrouwensbron van de gebruiker in Ontology, gaat vervolgens naar het adres om een verifieerbare claim te verkrijgen en verifieert ten slotte de verifieerbare claim.

6.1.1.2. Importeren via Trust Anchors

Trust-ankers zijn meestal overheidsorganisaties en andere openbare instellingen, ondernemingen, non-profitorganisaties en personen met autoriteit die zijn geverifieerd. Trust-ankers gebruiken hun eigen verificatiemethoden om entiteiten te verifiëren en verifieerbare claims uit te geven aan de geverifieerde entiteiten. De claim hoeft niet de identiteitsinformatie van de geverifieerde entiteit te bevatten; alleen de ONT ID en de authenticatieservice worden geregistreerd en het authenticatieresultaat kan worden verstrekt. Het authenticatiemodel is als volgt:

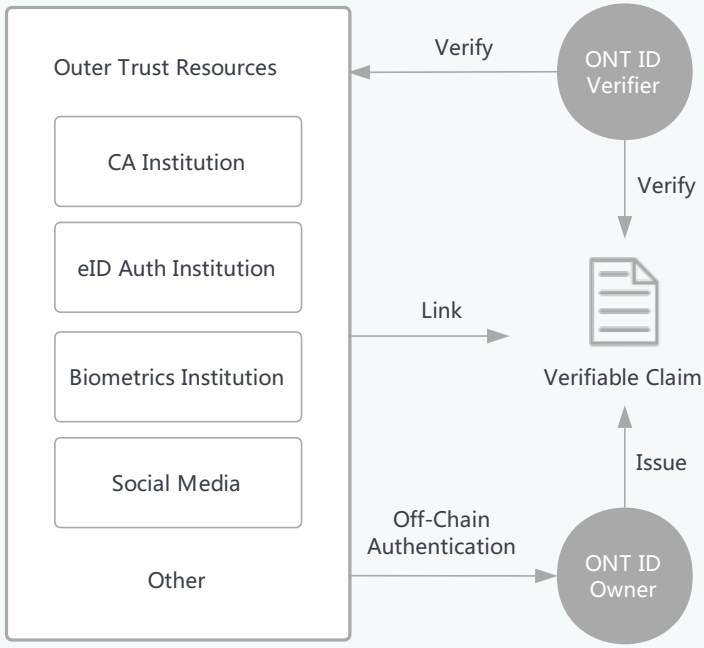

Figure 6.1: Importing through Trust Anchors

6.1.2. Identiteitsverificatie tussen Ontology-entiteiten

Entiteiten in Ontology kunnen ook identiteit verifiëren via entiteiten die identiteitsverificatie hebben doorgegeven (van een externe vertrouwensbron) in Ontology, zoals weergegeven in Afbeelding 6.2.

Vanwege de veelzijdigheid van verifieerbare claims, kan elke entiteit in Ontology een claim indienen over iets met betrekking tot een andere entiteit. Dit vereist identiteitsauthenticatie in Ontology die ver voorbij het traditionele concept van identiteitsverificatie ligt. Door claims van overheidsinstellingen, scholen, ziekenhuizen, banken, bedrijven, families, vrienden, partners, community leaders, collega's en docenten te verzamelen, kunnen we een meer veelzijdige authenticatie creëren die veel effectiever is dan traditionele systemen.

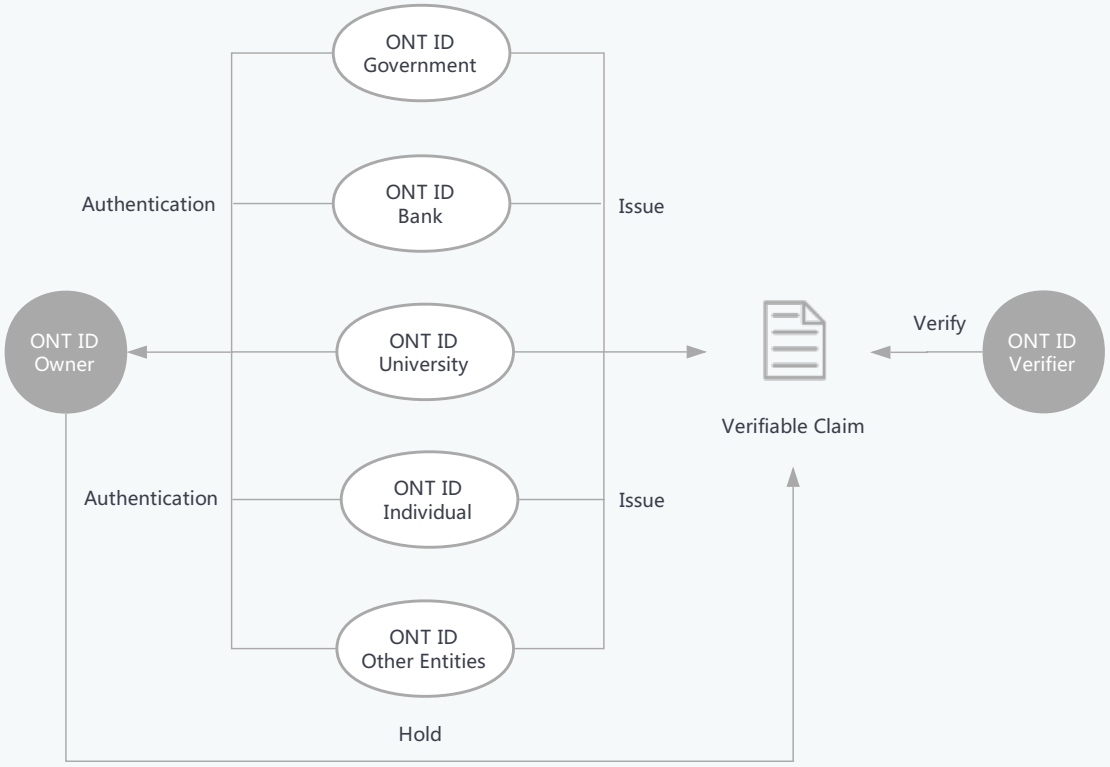

Figure 6.2: Authentication between Entities

6.2. GEBRUIKERSAUTHORISATIEPROTOCOL

In Ontology heeft de gebruiker volledige controle over zijn eigen gegevens. Elke toegang of transactie waarbij de gegevens zijn betrokken, moet door de gebruiker worden geautoriseerd. In het licht hiervan hebben we een reeks protocollen voor gebruikersautorisatie opgesteld om de gegevensprivacy van gebruikers te beschermen. Het protocol gebruikt verifieerbare claims om asynchrone en verifieerbare autorisaties uit te voeren en ondersteunt gedelegeerde autorisatie en fijnmazig toegangsbeheer.

6.2.1. Rollen

De belangrijkste rollen in de protocollen voor gebruikersautorisatie zijn:

- *Gebruiker: de entiteit die eigenaar is van een resource en die toegang tot de resource kan verlenen.*
- *Resource Requester: de aanvrager die de gebruikersgegevens of andere bronnen moet verkrijgen.*
- *Resource Provider: de serviceprovider die de gebruikersgegevens of andere bronnen verstrekt.*
- *Autorisatieserver: de server die de autorisatieverzoeken ontvangt en verwerkt, en kan een gedelegeerde autorisatiedienst voor gebruikers bieden.*

6.2.2. Authorisatie

Het gebruikersautorisatieprotocol voor toepassingsscenario's is verdeeld in drie fasen:

1) *Wederzijdse registratie: de gebruiker machtigt de bronprovider om de opgegeven resource te registreren bij de autorisatieserver en de autorisatieserver registreert zijn adres bij de bronaanbieder.*
2) *Instelling toegangsbesturingsbeleid: nadat de wederzijdse registratie is voltooid, kan de gebruiker toegang krijgen tot de machtigingsserver en de toegangscontrolestrategieën van de resource instellen.*
3) *Autorisatie: Resource requester initieert een toegangsautorisatieverzoek. Als aan de autorisatieconditie is voldaan, ontvangt de aanvrager een autorisatiecertificaat dat wordt gebruikt om gegevens aan te vragen bij de bronaanbieder.*

6.2.3. Wederzijdse registratie

Wederzijdse registratie betekent dat de autorisatieserver hun adres bij de bronaanbieder registreert, zodat wanneer de bronverschaffer het gegevenstoegangverzoek verwerkt, het het adres van de autorisatieserver aan de aanvrager retourneert. Tegelijkertijd moet de bronaanbieder zijn eigen adres registreren bij de autorisatieserver, zodat gebruikers toegang hebben tot de resourcebesturingen van de gebruiker.

Voordat autorisatiebewerkingen worden uitgevoerd, moet de gebruiker samenwerken met de bronprovider en de machtigingsserver om het wederzijdse registratieproces te voltooien.

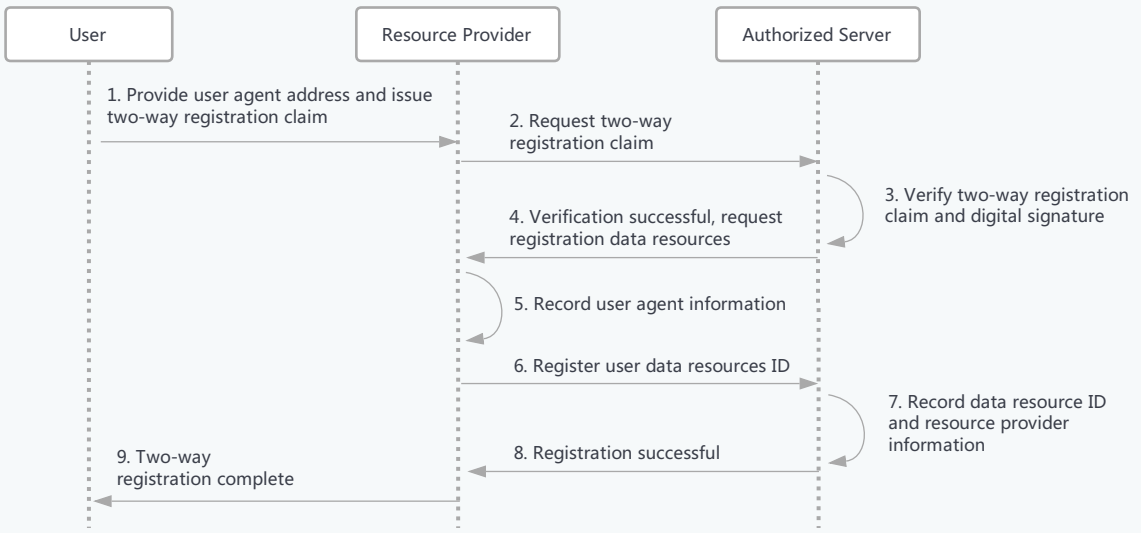

Figure 6.3: Two-Way Registration Process

Het proces wordt kort uitgelegd als volgt:

1) *Het wederzijdse registratieproces wordt geïnitieerd door de gebruiker, de gebruiker ondertekent een wederzijdse registratieclaim met hun ONT ID, het adres van de bronaanbieder en de ONT ID en het adres van de autorisatieserver.*

2) *De bronprovider handshake met de machtigingsserver die de wederzijdse registratieclaim gebruikt en beide partijen verifiëren de digitale handtekening.*

3) *De bronprovider registreert het toegangsadres van de autorisatieserver.*

4) *De machtigingsserver registreert de bron-ID die wordt verschaft door de bronverschaffer.*

6.2.4. Toegangscontrolestrategie

Met op attributen gebaseerde toegangscontrole kunnen gebruikers flexibele strategieën voor toegangsbeheer instellen die bepalen dat specifieke bronnen niet toegankelijk zijn voor aanvragers die voldoen aan specifieke attribuutvoorwaarden. Een toegangscontrolestrategie kan worden beschreven als een Booleaanse uitdrukking, zoals de strategie $A \vee (B \wedge C)$, wat een combinatie is van drie attribuutcondities. Bij het aanvragen van autorisatie moet de aanvrager het bewijs leveren van attributen in een verifieerbare claim die voldoet aan de strategie.

6.2.5. Authorisatiecertificaat

Wanneer een autorisatieverzoek wordt ontvangen, stelt de machtigingsserver de eigenaar op de hoogte om toegangscontrole uit te voeren. Voor aanvragers die voldoen aan de strategieën voor toegangsbeheer, geeft de eigenaar een certificaat van autoriteit af voor hen. Binnen de geldigheidsperiode van het certificaat heeft de aanvrager herhaaldelijk toegang tot de gegevens zonder opnieuw toestemming te vragen.

6.2.6. Gedelegeerde Autorisatie

Autorisatieservers kunnen fungeren als deelnemers voor gebruikers door hen toe te staan hun toegangscontrolestrategieën in te stellen. In dit patroon kunnen de servers autorisatieverzoeken verwerken en certificaten afgeven zonder interactie met gebruikers. Om de geldigheid van de certificaten te bewijzen, moeten gebruikers gedelegeerde claims indienen voor de machtigingsserver.

6.3. GEDISTRIBUEERDE GEGEVENSUITWISSELINGSPROTOCOL

De nadelen van gecentraliseerde gegevensuitwisseling zijn gegevenscaching, gebruik van gegevens zonder toestemming van de gebruiker en bescherming van gegevens door auteursrechten. Ontology stelt een Distributed Data Exchange Protocol (DDEP) voor, dat een set protocolspecificaties voor datatransacties tussen entiteiten definieert.

Om het eigen vermogen van beide partijen in de transactie te beschermen, wordt een tussenpersoon die optreedt als een 'garant' geïntroduceerd in het transactieproces van de

overeenkomst om ervoor te zorgen dat het afwikkelingsproces veilig en soepel wordt afgehandeld. De tussenpersoon is verantwoordelijk voor het houden van het geld van de koper en het overmaken van het geld aan de verkoper of de koper op basis van het uiteindelijke handelsresultaat. Aangezien de tussenpersoon verantwoordelijk is voor de definitieve afhandeling van de transactie, is deze eerlijk en veilig. Het werkt op basis van een gedistribueerd grootboekcontract met openbare en gedecentraliseerde beheerfuncties om ervoor te zorgen dat het geschikt de rol van tussenpersoon kan spelen.

6.3.1. Rollen

De belangrijkste rollen in het gedistribueerde gegevensuitwisselingsprotocol zijn:

- *Data requester: gegevensbureaus / bedrijven / personen die gegevens willen kopen.*
- *Gegevensprovider: gegevensbureaus / bedrijven / personen die gegevens willen verkopen, zowel onbewerkt als verwerkt. De gegevens moeten voldoen aan de lokale wet- en regelgeving.*
- *Gebruikersagent: Verantwoordelijk voor interactie met gebruikers om te voldoen aan gebruikersautorisatie-eisen voor datatransacties. Gebruikersagenten kunnen worden gediversifieerd (enterprise OA-systemen, internetplatforms of zelfs eenvoudige sms-gateways), maar ze moeten volledig worden geïmplementeerd zoals gedefinieerd in het gebruikerslicentieprotocol van het toepassingsprotocol.*
- *Gegevenshouder: de betrokkene, die instellingen / bedrijven / personen kunnen zijn.*

6.3.2. Gebruikersautorisatie

Omdat in de architectuur voor gegevensuitwisseling transactiegegevens moeten worden geautoriseerd door de eigenaar van de gegevens, voldoet het autorisatieproces volledig aan het gebruikersmachtigingsprotocol dat wordt beschreven in sectie 6.2.

6.3.3. Beveiligde transactie

Veilige transactieprotocollen door middel van smart contracten bieden gecentraliseerde externe assurancediensten voor handelsactiviteiten en zorgen voor een veilig en soepel transactieproces dat de equity van gegevensaanvragers en providers beschermt.

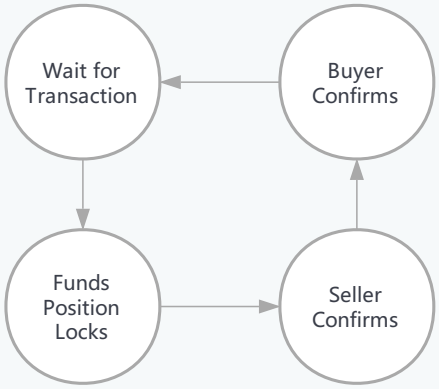

Figure 6.4: Guarantor Transaction Process

Het implementatieproces van de beveiligde slimme contracttransactie is als volgt:

1) *De gegevensprovider voert een lopende bestelling in, schrijft de productinformatie inclusief de bron-ID, gegevenskenmerken, provideraccount, prijs en andere functies en het contract wacht op een aanvrager die een transactie start.*
2) *De gegevensverzoeker draagt het opgegeven bedrag over aan het contract en het contract controleert of het overgedragen bedrag aan de verkoopvereisten voldoet.*
    a) *Als het slaagt, gaat het contract in een geblokkeerde geldstatus;*
    b) *Als de controle mislukt, wordt een foutmelding teruggestuurd naar de overnemer en keert de transactiestatus terug naar de vorige status.*
3) *Nadat de provider de gegevens heeft verstrekt, bevestigt het contract en wordt een vervaldatum bepaald. Als er aan het einde van de geldigheidsperiode geen andere actie is ondernomen, gaat het contract automatisch naar het afrekenproces (stap 5).*
4) *Na ontvangst van de gegevens bevestigt de aanvrager met het contract.*
5) *Het contract draagt het geld over naar het account van de provider om deze transactie te voltooien en te wachten op de volgende transactie.*

6.3.4. Data Exchange-proces

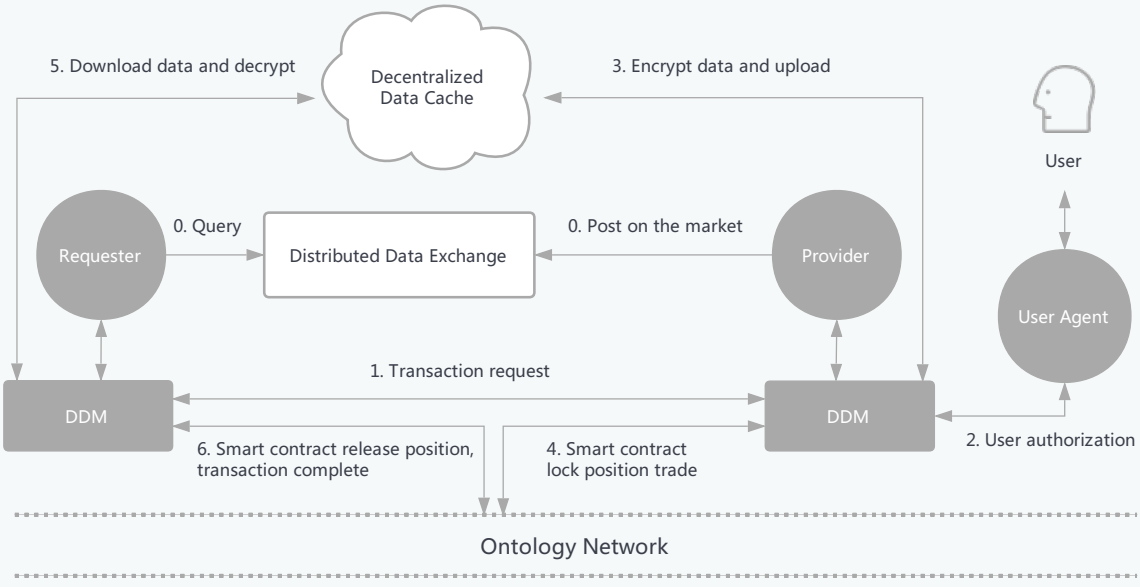

Figure 6.5: Data Exchange Process

Pre-order: transactie voorbereiden

Release voor dataproduct: de dataprovider publiceert hun dataproductinformatie op de markt. Aanvragers kunnen op de markt zoeken, zoeken naar gegevensproducten en gegevens kiezen die ze willen kopen. Meta-gegevensinformatie moet omvatten, maar is niet beperkt tot: introductie van gegevensbron, trefwoorden, hash-gegevensbron, betalingsadres van het contract en andere informatie.

Wederzijdse registratie: als het gegevensproduct de toestemming van de eigenaar van de gegevens vereist, moet de gegevensaanbieder de wederzijdse registratie voltooien met de door de gebruiker aangewezen gebruikersagent voordat de gegevens worden vrijgegeven. Raadpleeg de wederzijdse registratie hierboven voor meer informatie.

1) Transactie verzoek

Na het vinden van de gegevens die ze zouden willen kopen, verifieert de aanvrager de identiteit van de provider via Ontology en voor meer details verwijst naar het multi-source authenticatieprotocol. Voordat het transactieaanvraag wordt geïnitieerd, deponeert de aanvrager eerst geld naar het contractadres, stuurt hij een aanvraag voor de aankoop van gegevens naar de aanbieder en voegt hij de informatie toe die vereist is voor de autorisatie van de gebruiker. Het verzoek omvat maar is niet beperkt tot transactie-informatie en ONT ID-informatie.

2) Authorisatie

Na ontvangst van het verzoek van de aanvrager, opent de gegevensprovider de Gebruikersagent en initieert een autorisatieverzoek. Op dit punt kan de User Agent de identiteit van de aanvrager op verzoek verifiëren via Ontology en autorisatie uitvoeren volgens het toegangsbeleid dat vooraf door de Eigenaar is verstrekt. Als de Eigenaar geen toegangscontrolebeleid instelt, waarschuwt de user-agent de eigenaar voor autorisatie. Als het autorisatieverzoek wordt afgewezen, moet de transactie worden beëindigd.

3) Uploading data

De gegevensverschaffer genereert een eenmalige sessiesleutel volgens het symmetrische sleutelalgoritme dat wordt ondersteund door de aanvrager, gebruikt deze om de gegevens- en gegevenskarakteristiekwaarden van de transactie te coderen, en verzendt de ciphertekst (gecodeerde tekst) naar een tussenliggend opslagsysteem, b.v. IPFs.

4) Locking position

De gegevensprovider roept het slimme contract op om het geld dat door de aanvrager is gestort te controleren. Als het bedrag klopt, wordt de positie vergrendeld totdat de transactie is voltooid of geannuleerd. Ondertussen versleutelt de provider de sessiesleutel met de openbare sleutel van de aanvrager en stuurt deze naar de aanvrager via een beveiligd kanaal.

5) Receiving data

Na ontvangst van de melding van de smart contractgebeurtenis (gebeurtenis refererend aan sectie 5.3), haalt de aanvrager de *ciphertex*t uit de tussentijdse opslag, decodeert deze met de sessiesleutel, berekent en verifieert de kenmerken van de leesbare tekst, en gaat verder met stap 6) als de verificatie slaagt.Transaction confirmation

Wanneer het gegevenshandelscontract is voltooid, worden de contractgelden overgedragen naar de account van de gegevensaanbieder.

Uitzonderingsafhandelingsmechanisme: het uitzonderingsafhandelingsmechanisme kan worden aangepast aan verschillende bedrijfsscenario's. Bijvoorbeeld: als de aanvrager de gegevens nog niet binnen de opgegeven periode heeft bevestigd, kan de aanbieder het contract ontgrendelen of kan het slimme contract automatisch het geld ontgrendelen.

6.3.5. Privacy Bescherming

In sommige gegevensuitwisselingsgevallen kunnen deelnemers hun transacties verbergen. We stellen voor om de stealth-adrestechniek te gebruiken om het onhaalbaar te maken om het tokenadres van de ontvanger en ONT ID te correleren. De aanvrager genereert een verborgen adres waarvan de privésleutel alleen door de gegevensprovider kan worden hersteld op basis van het tokenadres.

SStel dat het tokenadres van de gegevensprovider $S = s \cdot G$ is. De aanvrager genereeert een willekeurig getal r,en berekent $R = r \cdot G$. Vervolgens het stealth address

$E = Hash(r \cdot S) \cdot G + S$, zal worden gebruikt als het uitvoeradres in een token-transactie. Alleen de provider de private key berekenen $e = Hash(s \cdot R) + s$ met zijn eigen private key s.

7. ONTOLOGY APPLICATIEFRAMEWORK

7.1. APPLICATIEFRAMEWORK MODEL

Het toepassingsraamwerk van Ontology biedt een uitgebreide set toepassingsprotocollen en modules waarmee externe dApp-ontwikkelaars snel decentrale applicaties kunnen bouwen zonder de complexiteit van het onderliggende gedistribueerde grootboek te hoeven begrijpen. Het toepassingskader van Ontology heeft een hoge mate van schaalbaarheid en kan worden uitgebreid volgens de behoeften van de scenario's.

Figuur 7.1 laat zien hoe dApps via het applicatieframework interactie aangaat met Ontologie om gedecentraliseerd vertrouwen te bereiken:

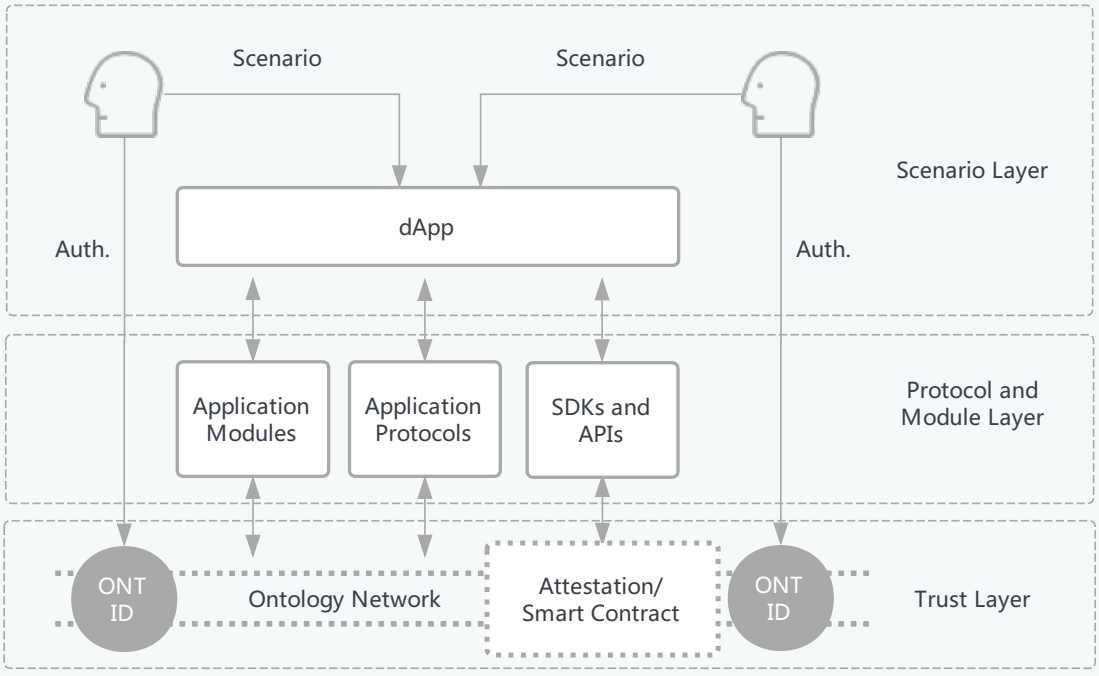

Figure 7.1: Application Framework Model

- *Trust Layer: Ontology neemt de verantwoordelijkheid van een trustsysteem op zich door zijn identiteit, attest en smart contractsystemen, etc.*
- *Module- en protocollaag: met deze laag kunnen de scenario's uit de bovenste laag de Ontology beter benutten via toepassingsmodules, toepassingsprotocollen, SDK's en API's.*

- *Toepassingslaag: dApps voor verschillende scenario's en de teams erachter kunnen zich richten op scenario-ontwikkeling, gebruikersservices, enz. Ontology kan helpen bij het oplossen van vertrouwensproblemen.*

7.2. DATA MARKTPLAATS

De toekomst is een digitale wereld en een markt voor gegevensuitwisseling die geschikt is voor de toekomst zal niet beperkt zijn tot alleen de overdracht van eigendom van gegevens. Vertrouwde collaboratieve berekening zal ook een belangrijke manier van samenwerking worden. Producten op de datamarkten omvatten data, data forecasting en datacommunicatiemiddelen, enz. Deelnemers zijn data-consumenten, providers en verwerkers zoals big data-services met behulp van deep learning en andere AI-technologie, die samen een datacommunicatie-ecosysteem zullen vormen. Dit complexe ecosysteem voor gegevenssamenwerking vereist een bijpassende infrastructuur.

Het gedistribueerde gegevensuitwisselingsprotocol (DDEP), de Data Trading Module (DDM) van de ontologie en een reeks cryptografische modules vormen samen een compleet gedistribueerd raamwerk voor gegevenshandel en samenwerking, dat wereldwijde schaalgroottes en uitwisselingsbehoeften mogelijk maakt voor ontologie dApp-gebruikers. Lagen van gegevensuitwisselingen dienstverleners bieden verschillende soorten gegevensuitwisseling tussen verschillende sectoren.

7.3. DATA DEALER MODULE

Data Dealer Module (DDM) is gebaseerd op het gedistribueerde gegevensuitwisselingsprotocol en is een belangrijke applicatiemodule in Ontologie. Met DDM kunnen zowel de dApp-ontwikkelaars als de gegevensuitwisselingsdeelnemers snel gebruikmaken van gegevensuitwisseling. De modules kunnen RESTful API, RPC, SDK, enz. Gebruiken en kunnen meerdere soorten protocollen ondersteunen.

DDM heeft verschillende typen om aan verschillende vereisten te voldoen, waaronder gegevenstransactieserver, client voor één gebruiker, client voor meerdere gebruikers en client voor lichte portemonnee. Er zijn vier belangrijke componenten voor ONT-identiteitsbeheer, databronbeheer, slimme contracttransacties en P2P-communicatie.

De DDM-gegevenstransactiemodule heeft de volgende voordelen en functies:

- *Ontkoppeling van toegangscontrolemodule. De module beheert alleen de binding van de gegevensbron en de beheerserver van de gegevensbronbezitter. Het neemt niet deel aan de configuratie en verificatie van toegangsrechten. Het onderhoudt niet alleen de bescherming van de privacy van de gegevens voor de eigenaar van de gegevens, maar verbetert ook het krediet van de gegevensaanbieder om onnodige geschillen te voorkomen.*

- *Bescherming van gegevensprivacy. Modules slaan de feitelijke gegevens niet op en met transactiegegevens versleuteld, zijn gegevensreclasseringsproblemen van leveranciers geëlimineerd.*
- *Naast het ophalen van gegevensdirectory, kan een gegevensverzoeker ook zijn opdrachten uitzenden die door gegevensverschaffer zullen worden ontvangen.*
- *Ontworpen volgens het principe "enkele module, enkele functie", ondersteunt het eenvoudig flexibele scenario's met cryptografische beveiligingsmodules en gebruikersautorisatiemodules.*

7.4. CRYPTOGRAPHIE EN BEVEILIGINGS MODULES

7.4.1. Secure Multiparty Computation

In een typisch collaborative computing-scenario willen twee of meer deelnemers de berekening uitvoeren met behulp van privégegevens zonder deze met anderen te delen. De huidige gebruikte benaderingen werken niet in dit scenario. Bijvoorbeeld, als applicatie-ontwikkeling bedrijf A krachtige algoritmen voor het leren van machines wil integreren in hun toepassing, zullen huidige systemen hun gegevens overbrengen naar ML-algoritmeprovider P. Vervolgens zal P hun algoritme op de gegevens uitvoeren en de resultaten terugsturen naar A. P zal weet alles over A's gegevens.

Voor dit soort scenario gebruiken we beveiligde multiparty-berekeningstechnieken (MPC). Het eerste beveiligde tweestarigheidsberekeningsprotocol werd geïntroduceerd door Andrew Yao, dat kan worden gebruikt om het probleem van Yao's miljonairs op te lossen: twee miljonairs die geïnteresseerd zijn in weten wie rijker is zonder de ander zijn werkelijke rijkdom te laten weten. Het algemene probleem kan worden samengevat als: n deelnemers houden elk een aantal privégegevens bij en willen de waarde van een openbare functie op die privégegevens berekenen: $f(x_1, \ldots, x_n)$. Het doel is om een protocol zodanig te ontwerpen dat alle informatie die elke partij kan leren, is wat ze kunnen leren van de uitvoer en hun eigen invoer. Er is dus geen extra gegevenslek voor elke deelnemer.

Twee belangrijke benaderingen voor MPC zijn in de loop van de jaren ontwikkeld: die gebaseerd op Yao's onleesbare circuits [22] [24] en die op basis van geheime delen [23]. In de rest van deze sectie zullen we ons concentreren op de laatste benadering.

Om het eenvoudig te zeggen, in een (t, n) -drempel geheim deelenschema wordt een geheim verdeeld in n aandelen. Elke partij krijgt maar een deel van dat geheim. Men kan het geheim herstellen als en alleen als hij tenminste t-aandelen krijgt. MPC-protocollen op basis van geheim delen partitioneren de tussenresultaten meestal in verschillende aandelen en distribueren deze naar de deelnemers. Uiteindelijk zal elke deelnemer de waarde reconstrueren $f(x_1, \ldots, x_n)$ van de aandelen die ze hebben ontvangen.

7.4.2. Volledig Homomorfe Encryptie

In een typisch gegevensuitwisselingsscenario wil de gegevensprovider de aanvrager toegang verlenen tot zijn gegevens in plaats van de volledige gegevens naar de aanvrager over te dragen. Het is van cruciaal belang dat een bedrijf in staat is om de privacy van zijn gegevens te handhaven terwijl beperkte toegang tot zijn gegevens wordt toegestaan. Recente ontwikkelingen in cryptografie, namelijk volledig homomorfe codering, bieden veelbelovende oplossingen voor dit probleem [25] [26] [27] [28]. Het bedrijf gebruikt bijvoorbeeld eerst zijn openbare sleutel om gegevens te versleutelen m en verzendt vervolgens ciphertekst naar de andere partij die ingewikkelde berekeningen gaat uitvoeren f op de ciphertekst. Deze nieuwe ciphertext is een codering van $f(m)$. Ten slotte wordt de nieuwe coderingstekst teruggestuurd naar de onderneming die de privésleutel kan decoderen en ophalen $f(m)$.

Formeel bestaat een volledig homomorfisch versleutelingsschema uit drie elementaire algoritmen die gemeenschappelijk zijn voor alle coderingsschema's voor openbare sleutels: genereren van sleutels, codering en decodering, plus homomorfe toevoeging $CAdd$ en multiplicatie $CMul$. Verondersteld dat C_1(resp. C_2) ciphertext is van bericht M_1(resp. M_2), dan:

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

Veel gecompliceerde bewerkingen kunnen worden opgebouwd als een rekenkundige schakeling die bestaat uit deze twee elementaire bewerkingen.

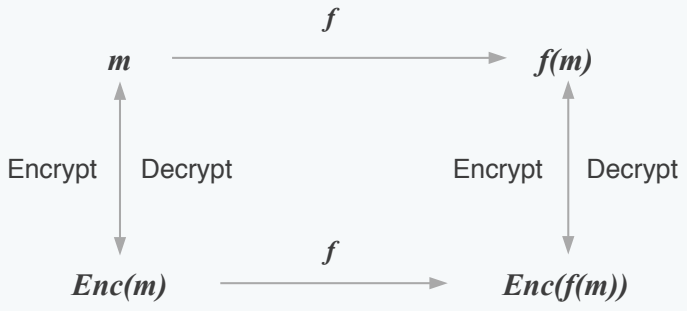

Figure 7.2: Homomorphic Encryption of Plain Text

Met behulp van het FHE-schema kunnen we daarom homomorfe activiteiten uitvoeren f op de cipherteksten en het ontcijferingsalgoritme zal ons $f(m)$ teruggeven (zoals geïllustreerd in de bovenstaande grafiek).

We zijn van plan in de toekomst verschillende FHE-schema's zoals BGV en FV te ondersteunen.

7.4.3. Digitaal Auteursrecht

Gebaseerd op de digitale kenmerken van data, biedt Ontology functies voor gegevensopslag en levenscyclusbeheer. Ten eerste ontwerpt Ontology het levenscyclus traceerbaarheidsmechanisme. Ontology vestigt een digitale identiteit voor elk stuk gegevens om het hele proces van registratie, aanvraag, autorisatie en transactie bij te houden. Ten tweede zorgt Ontology ervoor dat het auteursrecht en de transactie-informatie van gegevens allemaal in het gedistribueerde grootboek worden vastgelegd.

Ontology maakt gebruik van digitale watermerktechnologie om de identificatie-informatie direct in de digitale drager in te bedden. Zonder de gebruikswaarde te beïnvloeden, kan Ontology eenvoudig aanvallers identificeren en voorkomen dat ze wijzigingen aanbrengen. Met de verborgen informatie in de koerier kan de aanvrager de maker van de inhoud eenvoudig bevestigen en vaststellen of met de koerier is geknoeid. Digitale watermerken zijn een effectieve methode om namaaktraceerbaarheid, auteursrechtbescherming, verborgen identificatie, authenticatie en beveiligde geheime communicatie te implementeren.

De belangrijkste kenmerken van digitale watermerktechnologie die in Ontology wordt gebruikt:

- *Vindbaarheid: het watermerk biedt volledig en betrouwbaar bewijs van kenmerken van het auteursrechtelijk beschermde product. Met behulp van het watermerkalgoritme kunnen gebruikers de informatie van de eigenaar die is ingesloten in het beveiligde object (zoals het geregistreerde gebruikersnummer, productlogo, tekst, enz.) Identificeren en uitpakken wanneer dat nodig is. Watermerken kunnen worden gebruikt om te bepalen of het object is beveiligd, de verspreiding van beschermde gegevens te bewaken, authenticatie uit te voeren en illegaal kopiëren te voorkomen.*
- *Imperceptibility: het ingesloten watermerk kan niet worden gezien. De ideale situatie is dat de afbeelding met watermerk visueel identiek is aan de originele afbeelding, wat de meeste watermerkenalgoritmen kunnen bereiken.*
- *Robuustheid: Na het ondergaan van een verscheidenheid aan onopzettelijke of opzettelijke signaalverwerking, behoudt het digitale watermerk nog steeds de integriteit en kan het nauwkeurig worden geïdentificeerd. Robuustheid is uiterst belangrijk voor watermerken. Een digitaal watermerk moet bestand zijn tegen verschillende fysieke en geometrische vervormingen, inclusief opzettelijke vervormingen (bijvoorbeeld kwaadwillende aanvallen) en onbedoelde vervormingen (zoals beeldcompressie, filteren, scannen en kopiëren, geluidshinder, dimensionale veranderingen, enz.). Een robuust watermerkalgoritme moet in staat zijn om het ingesloten watermerk uit de afbeelding met watermerk te extraheren of om het bestaan van het watermerk aan te tonen. Als een specifiek watermerk-insluitings- en detectiealgoritme ontbreekt, moet het copyrightbeschermingsmerk van het gegevensproduct moeilijk te vervalsen zijn. Als de aanvaller het watermerk probeert te verwijderen, zullen multimedia-producten de afbeelding vernietigen.*

De gegevenswatermerkfunctie van Ontology bevat twee modules: een voor het insluiten van watermerken en de andere voor het verwijderen ervan, zoals weergegeven in Afbeelding 7.3.

Digitale watermerken worden gebruikt als beveiligingssysteem voor documenten, facturen en digitale identiteit. Door digitale watermerken kan de authenticiteit van een document worden bewezen en worden beschermd tegen illegaal kopiëren.

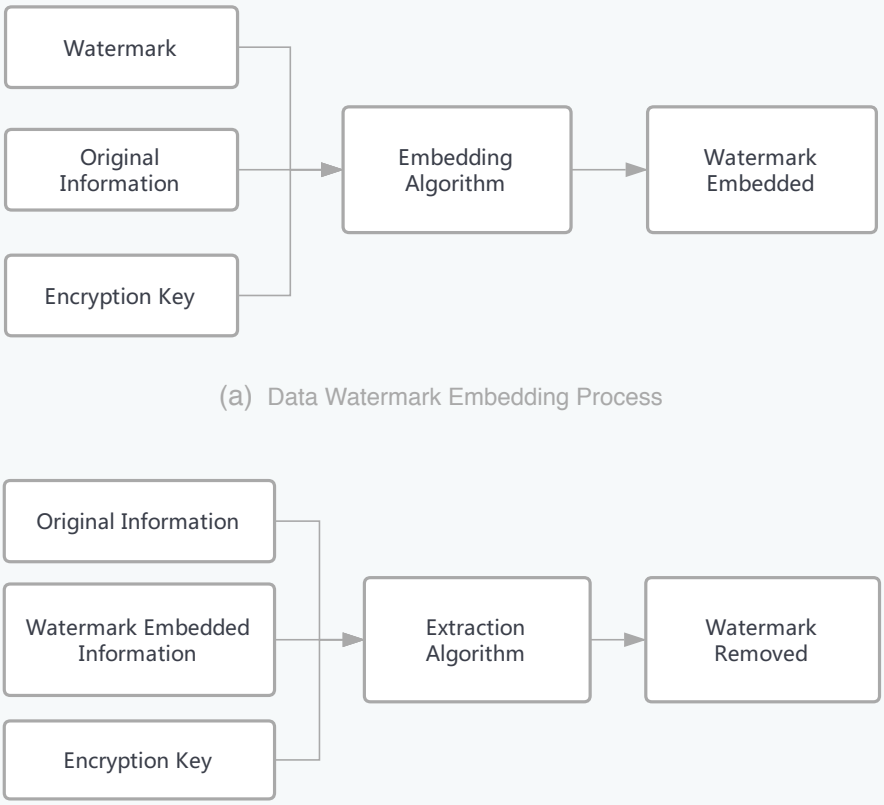

Figure 7.3: Data Watermark Module

7.5. USER AUTHORIZATION CONTROLLER

De gebruikersautorisatiecontrolemodule (UAC) is gebaseerd op het gebruikersautorisatieprotocol. Met behulp van granulaire toegangscontrole helpen UAC-modules de eigenaar van de gegevens met de autorisatie van hun eigen gegevens. Alle met gegevens verband houdende transacties brengen de gegevenseigenaar op de hoogte om een gegevenstransactieautorisatie uit te voeren. UAC-modules bieden RESTful API's voor extern gebruik en ondersteunen het gebruik van relationele databases (Mysql of Oracle). Ze hebben twee belangrijke functies:

- *User Data Authorization Access Policy Settings.*
- *User Data Authorization Access Control.*

7.5.1. Authorization Policy Setting

De gebruiker kan de bronaanbieder vragen om zijn gegevens te registreren met behulp van de RESTful API van de UAC-module. Na registratie kan de gebruiker de geregistreerde gegevens opzoeken en bekijken met behulp van de UAC-module en de instellingen voor toegang tot gegevens en controle bepalen. Voor sommige specifieke, vaak gebruikte gegevens met lage privacyvereisten kunnen gebruikers autorisatiehosting instellen op UAC-modules.

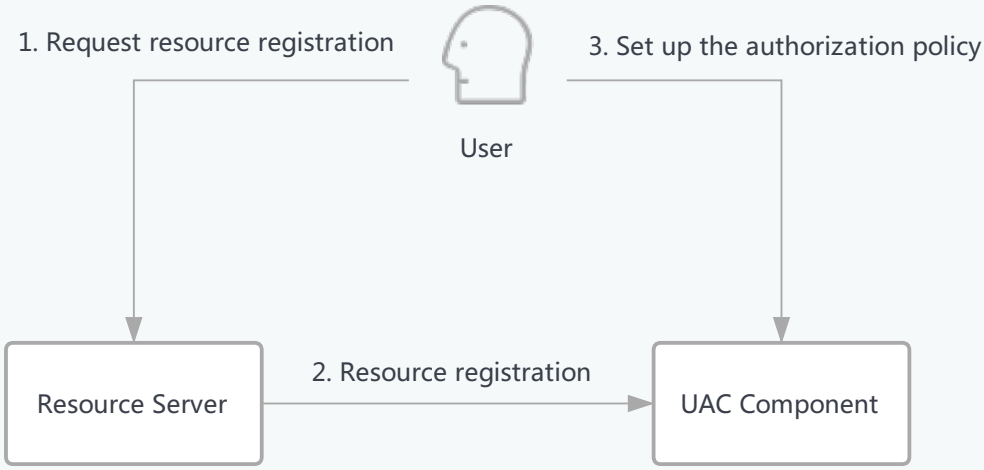

Figure 7.4: Setting Authorization Access Policy

7.5.2. Access Control

In een niet-gehoste modus fungeert de UAC-module als de beschermer van gebruikersgegevens en vereist de gebruiker om gegevensoverdracht aan een aanvrager te autoriseren. In de gehoste modus neemt de UAC-module autorisatiebeslissingen namens gebruikers en verschaft autorisatiebewijzen aan de gebruikers. Beide modi informeren gebruikers over de details van datatransacties, inclusief wie erbij betrokken is, hoe laat de transactie plaatsvindt en welke gegevens worden overgedragen.

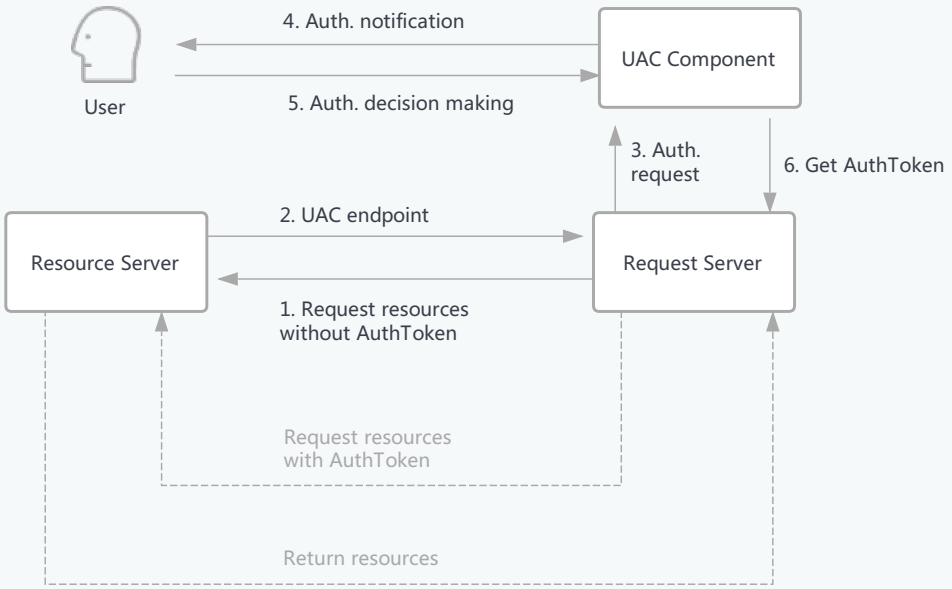

Figure 7.5: Authorization Access Control

7.6. CLAIM MANAGEMENT MODULE

Verifieerbaar claimbeheer is een belangrijke en basismodule voor het vertrouwensanker in Ontology. De claimbeheermodule is ontwikkeld in overeenstemming met de vereiste van het gedistribueerde vertrouwensysteem. Deze module biedt RESTful API's ter ondersteuning van het gebruik van relationele databases (MySQL of Oracle). De belangrijkste kenmerken van zijn modules omvatten, maar zijn niet beperkt tot: de uitgifte, verificatie, bevraging en annulering van vertrouwde claims.

7.7. GLOBALDB

GlobalDB is een pluggable gedistribueerde sleutel / waarde-database. De onderliggende laag is TiBD, een open-source gedistribueerde NewSQL-database op basis van Google's Spanner / F1.

GlobalDB is een database die is geoptimaliseerd voor blockchain / distributed ledger en IPFS. GlobalDB biedt SQL-compatibiliteit, sharding voor opslag, gedistribueerde transacties, horizontale schaalbaarheid en mogelijkheden voor foutherstel. Het zou kunnen worden toegepast in gezamenlijke toepassingen van blockchain en big data, blockchain en kunstmatige intelligentie (AI).

GlobalDB heeft vier belangrijke functies: gedistribueerde transacties, sharding van gegevens, taakverdeling en SQL op KV.

7.7.1. Gedistribueerde Transacties

GlobalDB is een pluggable gedistribueerde sleutel / waarde-database. De onderliggende laag is TiBD, een open-source gedistribueerde NewSQL-database op basis van Google's Spanner / F1.

GlobalDB is een database die is geoptimaliseerd voor blockchain / distributed ledger en IPFS. GlobalDB biedt SQL-compatibiliteit, sharding voor opslag, gedistribueerde transacties, horizontale schaalbaarheid en mogelijkheden voor foutherstel. Het zou kunnen worden toegepast in gezamenlijke toepassingen van blockchain en big data, blockchain en kunstmatige intelligentie (AI).

GlobalDB heeft vier belangrijke functies: gedistribueerde transacties, sharding van gegevens, taakverdeling en SQL op KV.

1) *Een enkele KV-invoer overschrijdt niet 6 MB.*
2) *Het totale aantal KV-vermeldingen is niet hoger dan 30W.*
3) *De totale grootte van de KV-invoer overschrijdt niet 100 MB.*

7.7.2. Storage Sharding

GlobalDB scheurt automatisch de onderliggende gegevens op basis van het bereik van de sleutel. Elke regio is een sleutelbereik $[StartKey, EndKey)$. Wanneer het totale bedrag van de gefragmenteerde sleutelwaarde een bepaalde drempel overschrijdt, wordt deze automatisch gesplitst.

7.7.3. Load Balancing

De load balancer PD berekent de belasting van het cluster op basis van de status van het opslagcluster. De planning is gebaseerd op regio, het beleid dat door de PD wordt geconfigureerd, is de planningslogica en het hele proces wordt automatisch voltooid.

7.7.4. SQL on KV

GlobalDB wijst de SQL-structuur automatisch toe aan een KV-structuur. In een notendop doet GlobalDB twee dingen:

- *Eén gegevenslijn wordt toegewezen aan een KV. De sleutel is voorafgegaan aan de tabel-ID en de regel-ID is van het achtervoegsel voorzien.*
- *Een index is toegewezen aan een KV. De sleutel maakt een voorvoegsel met TableID + IndexID en een achtervoegsel met indexwaarden.*

Gegevens of indexen in één tabel hebben hetzelfde voorvoegsel, dus in de sleutelruimte van TiKV bevindt de sleutelwaarde zich in een aangrenzende positie. GlobalDB zal de bijbehorende vuile

gegevensbeheerstrategie configureren om een beter presterende gegevensleesbaarheid te bereiken. [30]

GlobalDB is zeer configureerbaar en aanpasbaar en kan tegelijkertijd worden gebruikt in on-chain en off-chain real-time high-performance business. Het zal dienen als een kerncomponent in Ontology om een solide basis te bieden voor het onderliggende gedistribueerde grootboek.

8. POSTSCRIPTUM

Dit white paper bevat een overzicht van technologieën die bij Ontologie zijn betrokken. Naarmate de technologie echter voortschrijdt, zou het team van Ontologie de inhoud in de toekomst voortdurend bijwerken.

Ondertussen zal Ontology een open, collaboratief en creatief technologie-ecosysteem creëren. Het team van Ontology verwelkomt ontwikkelaars over de hele wereld om zich bij het gezin aan te sluiten, deel te nemen en de technologie van Ontology te bevorderen.

REFERENTIES

[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/.

CONTACT

 Email: contact@ont.io

 Telegram: OntologyNetwork

 Twitter: OntologyNetwork

 Facebook: ONTnetwork

 Reddit: OntologyNetwork

 Discord：https://discord.gg/vKRdcct

 Medium: OntologyNetwork

 LinkedIn: Ontology Network