Ontology

기술백서

Version 0.6.6

Date Updated： 18/04/2018

차세대 퍼블릭 멀티 체인 프로젝트 &
분산 신뢰 플랫폼

초록

역사적으로 사람들은 기술, 법의 지배, 공동체를 통해 신뢰를 형성해왔다.

근대의 기술 발전에도 불구하고, 신뢰 공급원의 파편화, 당사자 개인 역할의 부재, 부정확한 신원 확인, 잘못된 정보를 식별할 수 있는 능력의 부재 등의 요소들은 신뢰 형성에 큰 비용을 일으켰으며, 이는 신뢰를 통한 협력을 저해해왔다. 사회적 거버넌스, 경제적 협력, 금융 서비스 등의 분야에서 신뢰를 형성하는 비용은 여전히 막대하다.

탈중앙화된, 조작할 수 없는 블록체인 기술은 특정한 분야와 정해진 시나리오 내에서 신뢰를 형성하는 데 성공했다. 그러나 실제 비즈니스 시나리오와 결합하기 위해선 진보된, 통합적인 메커니즘이 필요하다.

온톨로지는 여러 데이터 공급원을 효과적으로 조정하여 신뢰 생태계를 위한 기반을 만드는 것과 분산 어플리케이션 개발을 위한 기반을 만드는 것을 목표로 한다.[1]

이 백서는 온톨로지의 기술적 프레임워크와 주요 기술적 원칙들, 핵심 프로토콜을 설명한다.

1. 온톨로지는 신뢰에 대한 다양한 협업 시나리오를 지원한다. 또한, 새로운 시나리오와 어플리케이션의 요구사항에 따라 필요한 모듈과 프로토콜들을 지속해서 늘려나갈 것이다. 이번 버전의 기술백서는 첫 번째 단계에서의 온톨로지의 구조와 프로토콜들을 설명한다. 어플리케이션 개발을 지속함에 따라 기술백서 또한 갱신할 예정이다.

목차

1. 개요(INTRODUCTION)

온톨로지(Ontology)는 다양한 산업과 지역으로 구성된 통합 다중 체인(multi-chain), 다중 시스템(multi-system) 프레임워크이며, 고유의 프로토콜들을 통해 다양한 체인과 전통적인 산업들을 서로 연결해준다. 이 때문에 온톨로지는 “온톨로지 체인그룹(Ontology Chain Group)” 혹은 "온톨로지 체인 네트워크(Ontology Chain Network)”라고 부를 수도 있다.

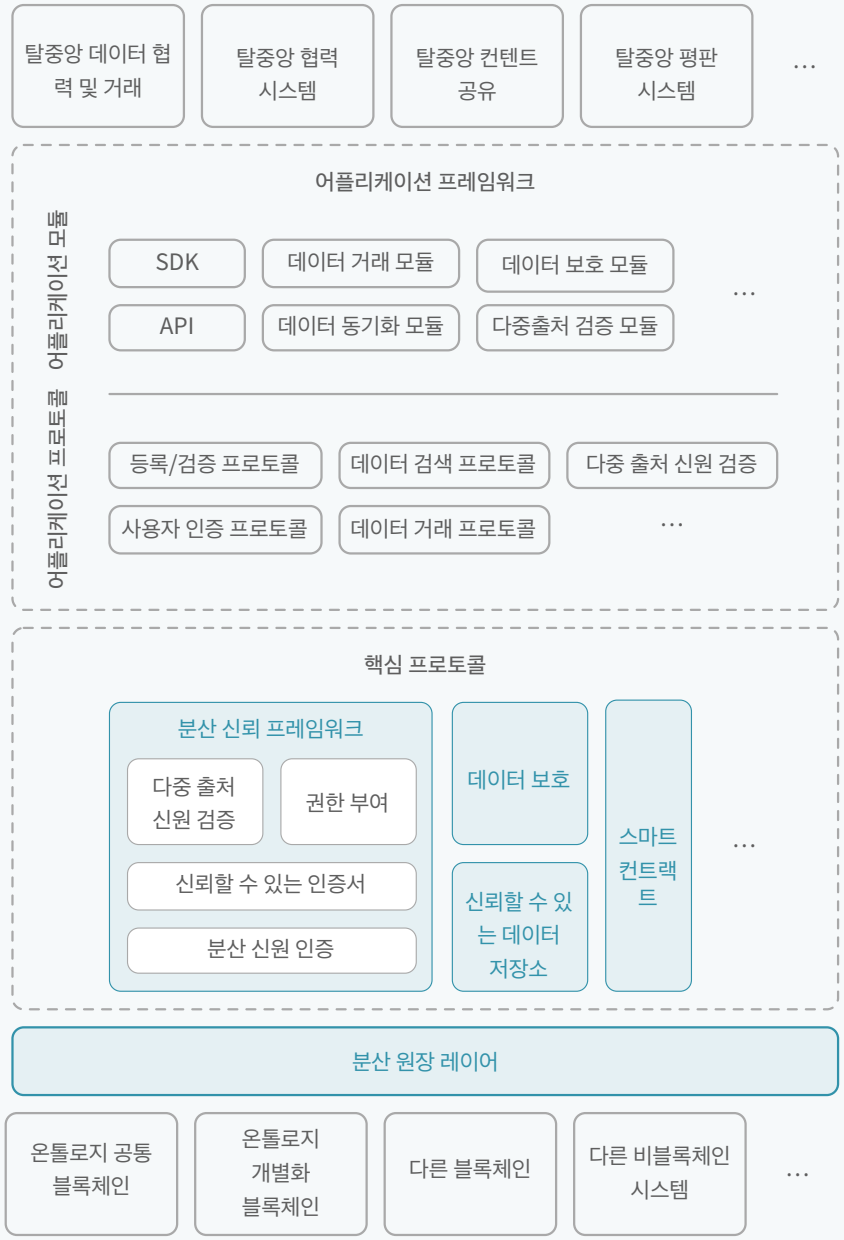


그림 1.1: 온톨로지의 기술 프레임워크(Ontology’s Technology Framework)

온톨로지는 분산 원장(distributed ledger), 스마트 컨트랙트 시스템(smart contract system), 보안 시스템(security system)을 포함하는 완전한 분산 원장 시스템을 지원한다. 분산 원장은 온톨로지의 근본적인 저장 기반이며, 분산 원장의 탈중앙화된, 상호적으로 관리되는, 조작 불가능한 특징은 온톨로지 네트워크의 참여자들 간의 신뢰를 형성하는 중요한 열쇠다. 분산 원장은 분산 신뢰 프레임워크, 합의, 저장, 스마트 컨트랙트 등을 포함하는 상위 어플리케이션 프레임워크를 지원한다. 온톨로지와 분산 원장은 분리되어 있으며, NEO, 이더리움을 비롯한 다른 블록체인을 하위 계층으로 사용할 수 있다. 장부 수준에서 데이터의 보관과 비즈니스 로직을 분리하는 고유의 공유 데이터 컨트랙트 모델을 사용한다.

분산 신뢰 프레임워크(distributed trust framework)는 온톨로지의 핵심 로직 레이어(core logic layer)로, ONT ID라는 고유한 식별자를 통해 다양한 인증 서비스를 제공하고, 사람, 통화, 재화 및 사물을 연결한다. ONT ID는 탈중앙화되었으며, 관리가 가능하고, 개인정보보호 기능이 있으며, 사용하기 쉽다. 신뢰 프레임워크는 검증가능한 내역[1][2]을 통해 분산 신뢰 모델과 분산 신뢰 전달 시스템을 형성하며, CL signature algorithm, 영지식증명(zero-knowledge proof) 프로토콜 등의 암호화 기법을 사용해 검증가능한 내역을 보호한다.

온톨로지는 신원 확인 프로토콜(identity protocol), 다중 출처 인증 프로토콜(multi-source authentication protocol), 사용자 권한 부여 프로토콜(user authorization protocol), 분산 데이터 교환 프로토콜(distributed data exchange protocol)에 프로토콜 표준을 사용한다. 예시로는 W3C가 설계했으며 국제적으로 호환되는 DID[3] 프로토콜이 있다. 또한, 암호 서명 프로토콜에는 정국 정부 표준인 RSA와 ECDSA를 지원한다. 개방과 표준화를 통한 협력과 생태계 확대를 위해 분산 데이터 거래 시스템에는 널리 사용되는 인증 프로토콜인 OAuth[4]와 UMA[5]가 호환되도록 했다.

어플리케이션 서비스 프로비져닝(application service provisioning)을 위해, 온톨로지는 어플리케이션 개발자들이 분산 시스템에 대한 지식 없이도 온톨로지 위에 분산 시스템을 만들 수 있는 기반을 제공한다. 다시 말해 온톨로지는 다양한 배경을 가진 개발자들이 어플리케이션을 개발할 수 있도록 API, SDK를 비롯한 다양한 도구를 제공한다. 이는 블록체인을 누구에게든 사용하기 쉽게 만들 것이다.

또한, 온톨로지는 암호학(cryptography), 데이터 보호 모듈(data protection module), 데이터 거래 마켓(data exchange markets), 글로벌 트랜잭션 데이터베이스(global transaction database), 하이브리드 오라클(hybrid oracle), 무제한 합의 엔진(unlimited consensus engine)과 같은 고급 기술들로 구성되어 있다. 차후 온톨로지는 생태계의 기술적 진보를 위해 개발자들과 비즈니스 파트너들 간의 커뮤니티를 만들 것이다. 또한, 지속해서 다양한 어플리케이션과 모듈을 개발할 예정이다.

2. 용어(GLOSSARY)

온톨로지 체인 그룹(Ontology Chain Group)

온톨로지 체인 네트워크(Ontology Chain Network)로도 알려져 있으며, 다양한 산업과 지역에 기반을 둔 체인들로 이루어져 있다. 각각의 체인은 별개의 분산 원장을 사용하며, 대화형 프로토콜(interactive protocol)을 통해 통신한다.

온톨로지 분산 원장(Ontology Distributed Ledger)

하나 이상의 퍼블릭 서비스 체인으로 이뤄진 온톨로지의 분산 원장/블록체인 프레임워크다. 분산 원장, 스마트 컨트랙트 시스템 외 다른 서비스들을 제공한다.

엔티티(Entity)

ONT ID로 식별할 수 있는 온톨로지 참여자이다.

ONT ID

ONT ID는 사람, 자산, 사물 등을 식별하는데 사용하는 관리하기 쉬우며, 개인정보보호가 되고, 안전하며 사용하기 쉬운 탈중앙 분산 프로토콜이다.

검증가능한 내역(Verifiable Claim)

한 엔티티가 다른 엔티티가(혹은 자신이) 가진 특정 속성에 관해 서술한 것이다. 해당 속성의 진위를 증명하기 위해 디지털 서명을 사용한다(주: Verifiable Claim은 W3C에서 비교적 최근 표준화를 시작한 개념으로, 논의되기 시작한 지 얼마 되지 않아 국제 표준 어휘임에도 불구하고 정립된 국어 표현이 존재하지 않는다. 따라서 본 백서에서는 역자가 이해한 바를 최대한 적절히 묘사할 수 있는 번역을 사용했다).

온톨로지 신뢰 프레임워크(Ontology Trust Framework)

분산 신원 검증 프로토콜(distributed identity verification protocol), 분산 신뢰 모델(distributed trust model), 분산 신뢰 전달(distributed trust transfer)을 포괄한 온톨로지의 신뢰 생태계를 구성하는 모듈을 말한다.

다중 출처 인증(Multi-source Authentication)

다출처 검증(multi-source verification)을 생성하기 위해, 한 엔티티의 서로 다른 측면들에 대한 다양한 검증 과정을 말한다.

신뢰 중재자(Trust Anchor)

검증 업무를 위탁받은 엔티티이다. 신뢰 전달 체인(trust delivery chains)의 공급자 역할을 하며, 신원 검증 서비스를 제공한다.

분산 일치 원장(Distributed Consistent Ledger)

공동 노드로 유지되는 탈중앙 P2P 네트워크의 증분 수정 데이터 저장 메커니즘(incremental modification data storage mechanism)이다. 태생적으로 투명하며 조작할 수 없다. 온톨로지의 저장기술과 스마트 컨트랙트 기술에 사용된다.

합의(Consensus)

각각의 노드는 일관성을 보장하기 위해 프로토콜에 따라 장부에 기록된 데이터를 검증한다.

스마트 컨트랙트(Smart Contract)

원장 노드 내 스마트 컨트랙트 엔진에서 작동하는 실행 가능한 코드이다. 실행의 입력값과 출력값 또한 원장에 저장된다.

온톨로지 어플리케이션 프레임워크(Ontology Application Framework)

써드 파티(third party)에 더 빠르고 저렴한 서비스를 제공하기 위한 어플리케이션 모듈, 프로토콜, SDK, API 등을 총칭하는 단어이다.

하이브리드 오라클(Hybrid Oracle)

하이브리드 오라클은 믿을 수 있는 외부의 데이터를 블록체인에 제공하는 서비스다. 이 서비스를 이용해 사용자들은 블록체인 시스템 외부의 사건에 대한 결과를 예측하고, 블록체인에 영구적으로 기록할 수 있다.

3. 체인 그룹 구조(CHAIN GROUP STRUCTURE)

온톨로지의 목표는 현실과 분산 데이터 시스템(distributed data systems)을 연결해주는 가교 역할을 하는 것이다. 따라서 현실에서의 비즈니스의 다양성, 복잡성, 전문성으로 인해, 성능, 확장성, 응용 가능성을 반드시 고려해야 한다. 하나의 퍼블릭 블록체인 혹은 제휴 체인만으로 모든 시나리오를 지원하기는 어렵다. 현실에서의 비즈니스 로직은 다양한 접근 방식과 거버넌스 모델을 가진 다양한 시나리오의 요구사항을 맞추기 위해 다양한 체인이 필요하다. 또한 많은 비즈니스 시나리오들은 독립적이지 않으며, 또 다른 시나리오들과의 상호작용을 요구한다. 그러므로 서비스 간 협력을 지원하기 위해 각각의 체인 간의 다양한 프로토콜이 필요하다.

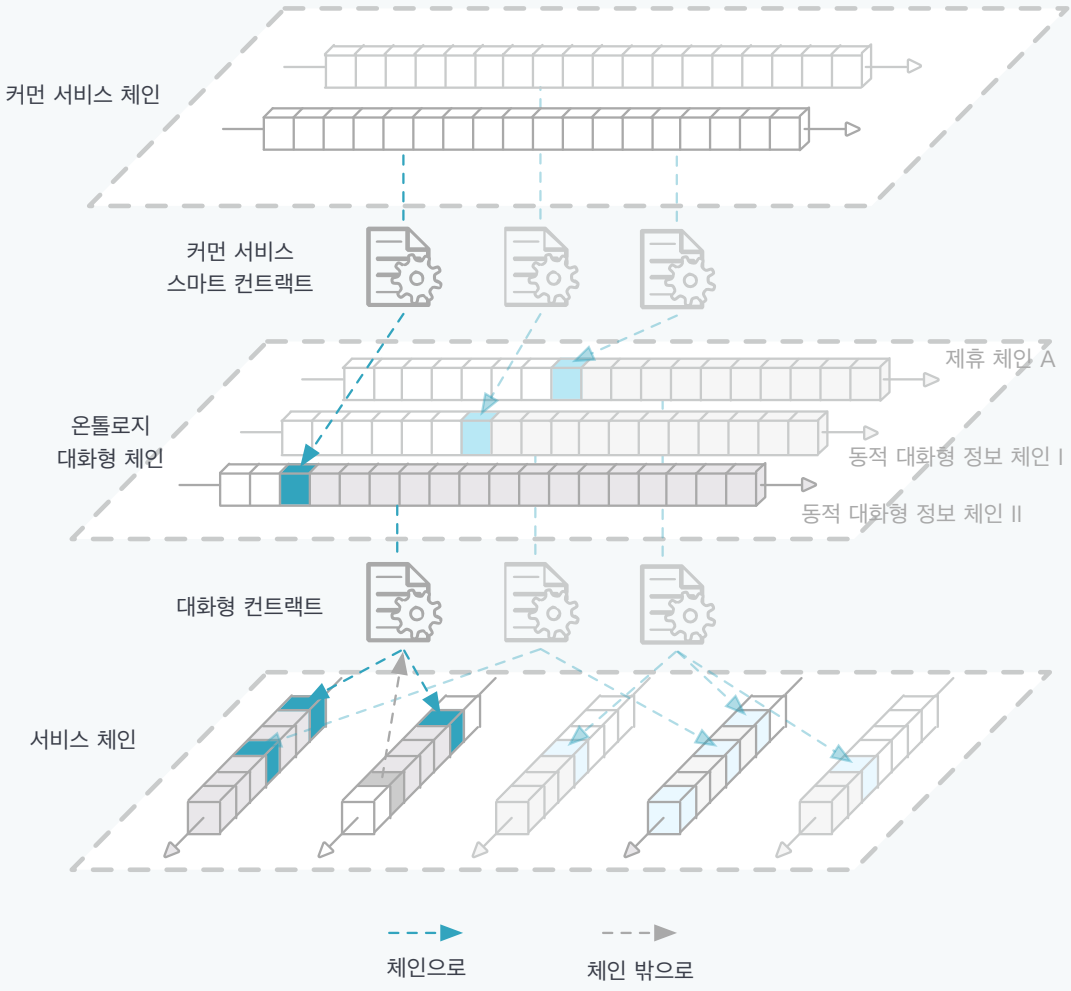


그림 3.1: 초집중 체인 그룹(Hyper-Converged Chain Group)

이러한 요구사항과 모델에 기반을 두고 있기 때문에 온톨로지는 매트릭스 그리드(matrix grid)의 형태를 하는 초집중 체인 그룹(hyper-converged chain group)을 사용한다. 체인의 수평 영역에는 엔티티 간의 연결(entity mapping), 데이터 교환을 위한 프로토콜, 스마트 컨트랙트 서비스 등을 제공하는 기본 공통 서비스(basic common services)가 있을 수 있다. 하나 이상의 퍼블릭 블록체인 위에서 다양한 산업, 지역, 비즈니스 시나리오 각각의 요구사항에 맞는 고유의 서비스 체인을 가질 수 있다. 또한 퍼블릭 체인을 사용해 엔티티 인증, 데이터 교환 프로토콜, 다른 비즈니스와의 소통과 같은 기본적인 서비스를 제공할 수 있다.

그림 3.2 체인 간 정보 교환을 위한 온톨로지 연산 플랫폼을 사용하는 서비스 체인
(Service chains using the Ontology computation platform for cross-chain information exchange)

그림 3.3 온톨로지의 연산 플랫폼을 사용해 구성된 전용 대화식 체인
(Building a dedicated interactive chain using Ontology's computation platform)

다른 퍼블릭 체인과 상호작용하는 것 외에도, 특정 산업에 특화된 체인과 연계하는 것도 가능하다. 주어진 각 협력 시나리오에 따라 관련된 서비스 체인은 각자 다를 수 있다. 따라서 특정한 비즈니스 요구사항을 만족시키기 위해 하나 혹은 여러 개의 서비스 체인과 협력하는 전용 퍼블릭/제휴 서비스 체인이 존재할 수 있다. 그러므로, 수직 영역에는 스마트 컨트랙트 서비스, 비즈니스 로직 서비스 등을 위한 특별한 체인 간 협력 지원을 포함하는 비즈니스 협력 체인들이 생길 것이다.

이러한 매트릭스 그리드 구조는 자율적이고 유연한 차세대 네트워크다. 다양한 비즈니스 시나리오는 다양한 협력 방식 중 적합한 서비스 모델을 찾아 적용할 수 있다.

온톨로지의 프로토콜은 정적이지 않다. 사용자들은 비즈니스 시나리오, 산업 요구사항, 규제 요구사항, 거버넌스 요구사항에 따라 프로토콜을 선택할 수 있다. 따라서 다양한 프로토콜의 사용성을 극대화하고, 각각의 시나리오에 따른 표준이 더 나은 호환성과 확장성을 보장할 수 있도록 프로토콜 개발은 온톨로지의 개발 과정에서 중요한 부분으로 남을 것이다.

온톨로지는 분산 원장 프레임워크를 기반으로 다양한 시나리오를 만족시키는 구성 가능한 블록체인을 구현한다. 분산 원장 프레임워크를 사용해 특정 비즈니스 시나리오를 위한 서비스 체인을 개별화하는 것도 가능하다. 또한 온톨로지는 특정 프로토콜을 사용해 다른 블록체인 시스템이나 전통적인 IT 시스템과 협력할 수도 있다.

4. 분산 신뢰 프레임워크(DISTRIBUTED TRUST FRAMEWORK)

4.1. 온톨로지 신원 인증 프로토콜(ONTOLOGY IDENTIFICATION PROTOCOL)

“엔티티(Entity)”는 현실에서의 개인, 법인(단체, 기업, 기관 등), 사물(핸드폰, 자동차, IoT 기기 등)을 지칭한다. “아이덴티티(Identity)”는 온톨로지 네트워크상에서 엔티티의 존재를 지칭한다. 온톨로지는 엔티티의 아이덴티티를 식별하고 관리하기 위해 ONT ID를 사용한다. 온톨로지에서 한 엔티티는 여러 아이덴티티를 가질 수 있다.

ONT ID는 탈중앙화 신원 확인 프로토콜이다. 각각의 ONT ID는 공개 키(public key)와 같은 속성 정보(attribute information)를 저장할 때 사용하는 ONT ID Description Object(ODD)와 호응된다. ODD는 분산 원장에 공개된 정보다. 따라서, 개인정보보호를 위해 ODD에 엔티티의 개인정보가 저장되지는 않는다.

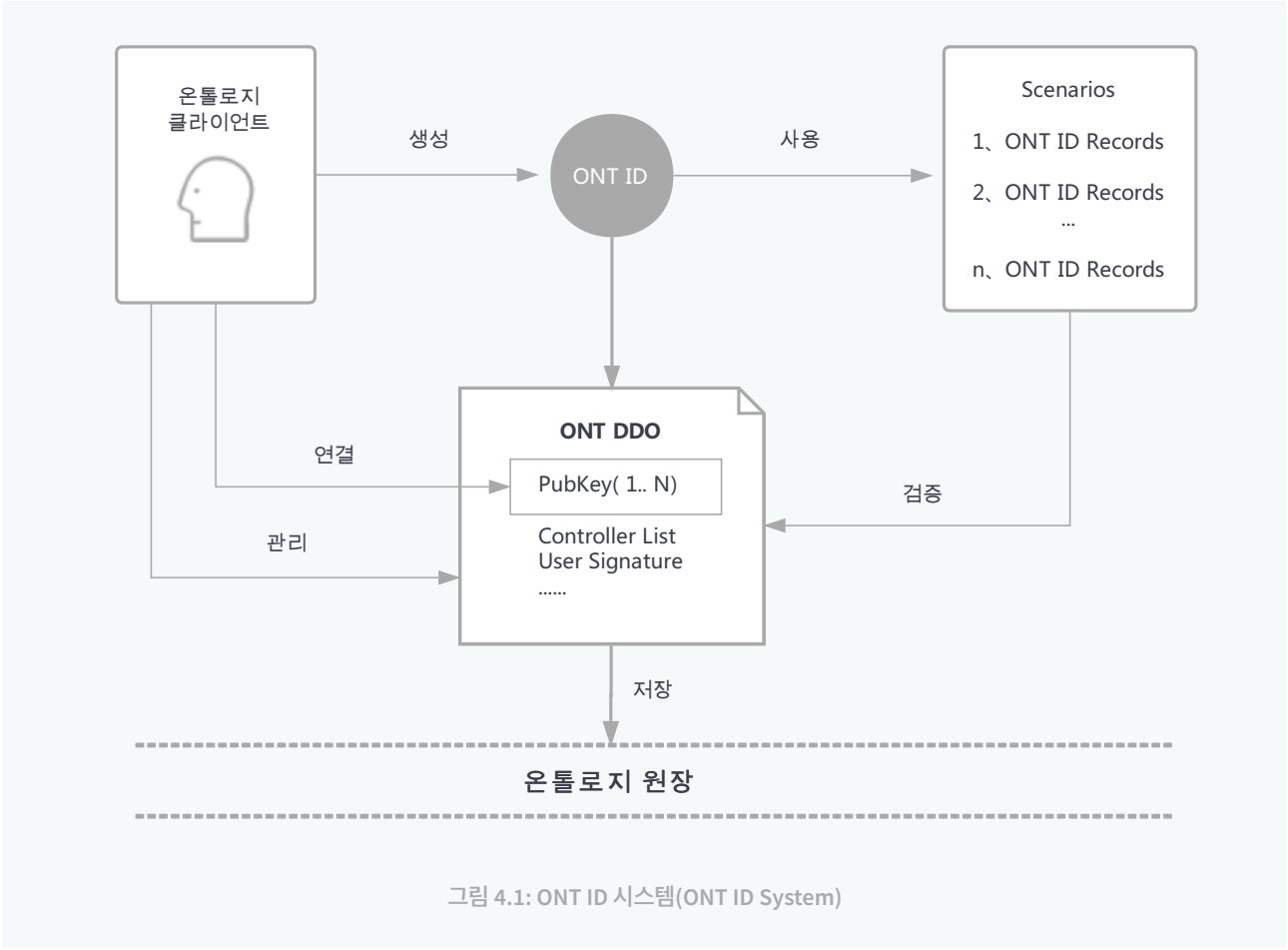


그림 4.1: ONT ID 시스템(ONT ID System)

4.1.1. ONT ID 생성(ONT ID Generation)

ONT ID는 엔티티의 URI(주: 특정 자원을 식별하는데 사용하는 고유한 문자)[6]의 일종이다. 생성 알고리즘에 의해 같은 ONT ID가 생성될 확률($\approx \frac{1}{2^{160}}$)은 매우 낮다. 또한, ONT ID가 온톨로지 네트워크에 등록될 때, 합의 노드를 해당 ID가 이미 등록되었는지 확인한다.

4.1.2. 자주권(Self-Sovereign)

온톨로지는 엔티티가 자신의 아이덴티티를 관리할 수 있도록 디지털 서명 기술을 사용한다. ONT ID는 소유권 식별을 위해 엔티티의 공개 키와 함께 등록된다. ONT ID의 사용과 속성들은 사용자에 의해 디지털 서명되어야 한다. 엔티티는 자신의 ONT ID에 대한 사용을 결정할 수 있고, 개인 키를 연결할 수 있으며, 속성들을 관리할 수 있다.

4.1.3. 다중 키 연결(Multi-Key Binding)

온톨로지는 RSA, ECDSA, SM2를 포함하는 다양한 디지털 서명 알고리즘 표준을 지원한다. 각각의 응용 시나리오의 요구사항에 따라, ONT ID에는 여러 개인 키가 연결될 수 있다. 엔티티는 ONT ID에 연결된 각각의 개인 키들에 어떤 알고리즘이 사용되었는지를 공유해야 한다.

4.1.4. 권한 관리(Authorized Control)

ONT ID의 소유자는 다른 ONT ID에게 자신의 ONT ID를 제어할 수 있는 권한을 부여할 수 있다. 예를 들어, 제어 권한을 부여받은 ONT ID는 해당 ONT ID의 속성 정보를 변경할 수 있으며, 해당 ONT ID에 최초로 부여된 개인 키가 분실되었을 경우, 새 개인 키를 연결할 수도 있다. ONT ID는 각각의 속성에 대한 세밀한 권한 관리와 “AND”, “OR”, “m of the n”과 같은 복합적인 접근 제어를 지원한다.

4.2. 신뢰 모델(TRUST MODEL)

온톨로지의 신뢰 모델은 중앙화 신뢰 모델과 탈중앙화 신뢰 모델 모두를 이용해 엔티티 간의 신뢰를 형성한다. 시나리오마다 다른 요구사항을 충족시키기 위해 다른 신뢰 모델들을 사용할 수 있다.

4.2.1. 중앙화 신뢰 모델(Centralized Trust Model)

중앙화된 신뢰 모델에서 하나 혹은 여러 엔티티가 신뢰 중재자(trust anchor)가 될 수 있으며, 이를 기반으로 엔티티 간의 신뢰 관계가 형성된다. 신뢰 중재자는 신뢰할 수 있는 엔티티를 지정할 수 있다. 지정된 엔티티들도 차례로 다른 신뢰 중재자가 될 엔티티를 지정할 수 있다. 초기에 지정된 신뢰 중재자들로부터 점차 신뢰 트리(trust tree)가 형성된다. 트리 안의 각각의 노드는 신뢰 중재자로 통하는 경로를 가진다. 이를 신뢰 체인(chain of trust)이라고 한다. 엔

티티가 트리 내의 다른 노드들과 상호작용할 때, 이 신뢰 체인을 검증하여 신뢰 중재자를 신뢰할 수 있는지를 확인할 수 있다.

PKI[7]는 가장 널리 사용되어 온 중앙화 신뢰 시스템이다. 신뢰 중재자가 되기 위해서 사용자는 디지털 서명(digital certificate)을 신청해야 한다. 신청이 승인되면, 인증 기관은 신원 정보와 공개 키를 디지털 서명으로 만든 후 저장한다. 인증 기관이 발급한 디지털 서명은 사용자의 신원과 공개 키의 결합 관계를 검증하고 신원을 확인한다. 또한, 디지털 서명을 발급받은 사용자들은 다른 사용자들을 위해 하위 서명을 발급해줄 수 있으며, 인증 기관도 여기서 발급된 서명을 보증한다. PKI 모델의 모든 참여자는 무조건 인증 기관을 신뢰해야 한다. 신뢰 관계는 인증 기관으로부터 엔티티 간의 디지털 서명을 통해 계층적으로 전송된다.

중앙화 신뢰 모델은 많은 장점이 있다. 엄격한 신뢰 전달 방식과 신뢰 여부에 대한 명확한 기준은 많은 시나리오에 대해 유용하며, 실제로 많은 문제를 해결한다. 그러나 중앙화 신뢰 모델엔 명확한 단점들도 존재한다. 중앙 노드에 대한 의존은 복잡한 신뢰 관계에 부적합할 수 있으며, 투명성과 보안이 중요한 영역에선 더욱 그렇다. 중앙 노드에 의존하는 방식은 어플리케이션의 유연성을 심각하게 제한할 수 있다.

4.2.2. 탈중앙 신뢰 모델(Decentralized Trust Model)

특정 중앙화된 개체들에 의존하여 신뢰 관계를 형성하는 방식 외에도, 엔티티들은 동등한 권한으로 서로 신뢰 관계를 형성할 수 있다. 이 모델에서 신뢰 전송은 엔티티 간의 상호 인증을 통해 수행된다. 다른 엔티티들로부터 더 많은 인증을 받은 엔티티는 더 큰 신뢰도를 가진다.

탈중앙 신뢰 모델은 불명확한(fuzzy) 신뢰 모델이다. 신뢰할 수 있는 것과 없는 것 간의 경계가 명확하지 않다. 엔티티 간의 신뢰도를 측정할 때, 여러 시나리오에 따라 여러 신뢰 평가 방식이 사용될 수 있다. 유연성이 높을수록 더 많은 현실 요구사항을 충족시킬 수 있다.

4.3. 검증가능한 내역(VERIFIABLE CLAIM)

검증가능한 내역은 엔티티의 특정 속성을 검증할 때 사용된다. 데이터 단위로 저장되고 전송될 수 있으며, 어떠한 엔티티에 의해서도 검증될 수 있다. 검증가능한 내역은 메타데이터, 내역의 내용물(어떤 데이터라도 될 수 있다), 발급자의 서명으로 이루어진다. 검증가능한 내역은 LD-Signature 표준[9]으로 서명된 JSON-LD[8] 형식으로 구성되어있다.

4.3.1. 라이프 사이클(Life Cycle)

그림 4.2: 검증가능한 내역의 라이프 사이클(Verifiable Claim Lifecycle)

검증가능한 내역과 관련된 엔티티들은 발급자(issuer), 보유자(holder), 검증자(verifier)의 세 역할 중 하나를 갖게 된다. 검증가능한 내역의 라이프 사이클은 아래 다섯 가지 작업으로 나뉜다.

- 발급*(Issurance):* 어떠한 엔티티도 다른 엔티티의 검증가능한 내역을 발급할 수 있다. 예를 들어, 한 학교가 검증가능한 내역을 발급함으로써 학생에게 성적표를 줄 수 있다. 검증가능한 내역에는 유효기간을 설정할 수 있다. 유효기간이 지나면, 내역*(claim)*은 자동으로 만료된다.
- 저장*(Storage):* 검증가능한 내역은 공개 내역*(public claim)*과 개인 내역*(private claim)*으로 나뉜다. 공개 내역은 온톨로지 원장에 저장된다. 개인 내역은 엔티티의 클라이언트에 저장되거나 엔티티가 직접 관리한다.
- 전달*(Presenting):* 검증가능한 내역의 소유자는 어떤 사람에게 내역*(claim)*을 공개할지 그리고 내역*(claim)*의 무결성에 영향을 주지 않는 선에서 어떤 정보를 공개할지를 정할 수 있다.
- 검증*(Verification):* 검증에는 해당 내역*(claim)*의 발급자가 필요 없다. 필요한 것은 온톨로지의 원장에서 공개 키를 가져올 때 사용할 발급자의 *ONT ID*뿐이다. 공개 키를 사용해 내역*(claim)*의 디지털 서명을 검증할 수 있다.
- 취소*(Cancellation):* 검증가능한 내역의 발급자는 만료 기간 전에 내역*(claim)*를 취소할 수 있다. 만료된 내역*(claim)*은 검증에 사용될 수 없다.

4.3.2. 익명 내역(Anonymous Claim)

일반적으로, 내역(claim)의 소유자는 내역을 만들때 검증자에게 내역의 모든 내용을 노출한다. 그러나 몇몇 시나리오에서 내역의 소유자는 특정 정보를 노출하고 싶지 않을 수 있다. 사용자의 개인 정보 보호를 위해, 온톨로지는 익명 내역(anonymous claim) 기술을 준비했다.

익명 내역 기술은 기존에 내역을 발급하거나 전달하는 동안 소유자의 정보를 감출 때 발생할 수 있던 문제를 해결한다. 익명 내역 프로토콜에서 엔티티는 두 검증자에게 각각 검증을 받는다. 두 검증자가 소유자의 정보를 유출하기 위해 담합한다고 하더라도, 그들이 받은 정보가 같은 엔티티로부터 왔다는 것을 검증할 수 없다. 익명 내역을 만들 때, 소유자는 검증자에게 내역의 원형을 그대로 보낼 필요가 없으며, 영지식 증명으로만 제공하면 된다. 검증자는 발급자의 공개 키, 증명서, 증명서에 있는 속성정보(예를 들면 “age > 18” AND “resident of Shanghai”와 같은)들을 이용해 검증 알고리즘을 실행한 후 내역의 진위 여부를 검증할 수 있다.

익명 내역은 공개적인 정보와 암호화 관련 정보를 담는 XML이나 JSON 형식의 데이터이다. 공개적인 정보는 익명 내역의 모든 속성을 포함하며, 각 속성은 속성의 이름, 속성의 타입, 속성의 값 세 부분으로 나뉘어져 있다. 속성은 문자, 정수, 날짜, 열거형과 같은 다양한 데이터 타입을 지원한다. 암호화 관련 정보는 소유자의 마스터 키나 발급자의 공공 정보 디지털 서명을 주로 포함한다.

익명 내역을 제출하는 과정에서, 내역의 소유자는 검증자에게 자신의 익명 내역이 발급자로부터 발급된 것임을 증명해야 한다. 소유자는 노출할 속성과 감출 속성을 선택할 수 있다. 또한, 소유자는 숨겨진 속성이 특정한 로직을 만족한다는 것을 증명할 수 있다.

익명 내역은 위에 설명된 기능들을 위해 CL 서명 알고리즘[10]과 Σ-protocol[11]을 사용한다.

4.3.2.1. CL 서명 알고리즘(CL Signature Scheme)

CL 서명 알고리즘은 키 생성 알고리즘(key generation algorithm), 서명 생성 알고리즘(signature generation algorithm), 서명 검증 알고리즘(signature verification algorithm)의 세 알고리즘으로 구성되어 있다. CL 서명 알고리즘은 강한 RSA 가정에 따라 확률적으로 안전하다.

키 생성 GEN:

1) 두 임의의 소수 (p, q)를 고르고, $n = pq$를 계산한다.

2) 임의의 $k + 2$ 개의 *quadratic residues* $(R_1, \ldots, R_k, S, Z)$를 고른다.

3) 개인 키 $sk = (p, q)$와 공개 키 $pk = (n, R_1, \ldots, R_k, S, Z)$를 출력한다.

서명 생성 $SIGN_{sk}(\{m_i\})$:

k개의 값 $\{m_1, \ldots, m_k\}$를 입력한다.

1) 다음의 오일러 함수를 계산한다. $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$

2) 임의의 충분히 큰 소수 e와 큰 정수 v를 고른다.

3) e *module* $\varphi(n)$의 역수 혹은 e^{-1}를 계산한다. 이는 다음을 만족한다.

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

4) 다음의 정수를 계산한다.

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

5) 서명, (A, e, v)를 출력한다.

서명 검증 $VERIFY_{pk}(\{m_i\}, A, e, v)$:

k개의 값 $\{m_1, \ldots, m_k\}$와 서명 (A, e, v)를 입력한다.

1) e와 v가 주어진 범위 안에 있는지와 e가 소수인지 확인한다.

2) 서명이 다음을 만족하는지 확인한다.

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

4.3.2.2. Σ-protocols

Σ-protocol은 증명자 P가 검증자 V에게 비밀 메시지 자체를 전달하지 않으면서, 자신이 비밀 메시지를 가지고 있음을 설득할 수 있는 효율적인 영지식 증명 프로토콜이다. 잘 알려진 Σ-protocol로는 Fiat-Shamir heuristic[12] and Schnorr signature scheme [13]이 있다.

P가 $y_1 = g^x,\ y_2 = h^x$에 대한 해를 안다고 가정하자. 그러면 그는 다음의 Σ-protocol을 사용할 수 있다.

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

이는 다음의 상호적인 프로토콜을 통해 증명될 수 있다.

1) P는 임의의 정수 r을 고른다.

2) $P(g^r, h^r)$를 계산하고 V에게 결과를 보낸다.

3) V는 임의의 정수 c를 고르고, 이를 P에게 보낸다.

4) P는 $s = r - c \cdot x$를 계산하고, s를 V에게 보낸다.

5) V는 s가 다음 방정식을 만족하는지 검증한다.

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

4.3.2.3. 익명 내역 발급(Anonymous Claim Issuance)

수신자 R에게 익명 내역을 발급할 때, 발급자 I와 수신자 R은 아래의 두 단계의 상호적 프로토콜을 거쳐야한다.

1) I는 임의의 정수 n_1을 고르고, R에게 보낸다.

2) R은 임의의 정수 v'를 고르고, $U = R_1^{m_0} S^{v'} \pmod{n}$를 계산한다. m_0은 *the master secret*이다. 그러면, R은 U를 I에게 보낸다.

3) I은 임의의 소수 e와 정수 v''를 고른다.

4) I는 R의 퍼블릭 속성 $\{m_i\}$에 기반해 *CL signature* (A, e, v'')를 계산하고, R에게 전송한다.

5) R은 $v = v' + v''$를 계산하고, 최종 익명 내역 (A, e, v)를 저장한다.

수신자는 이 프로토콜을 실행함으로써 발급자로부터 자신의 퍼블릭 속성값에 대한 CL 서명과 익명 내역의 암호화된 부분을 얻을 수 있다.

4.3.2.4. 전달과 검증(Presentation and Verification)

익명 내역의 보유자는 검증자에게 몇몇 속성을 골라서 공개할 수 있으며, 다른 부분들은 비밀로 할 수 있다.

공개되지 않은 속성들을 비밀로 유지하고 싶을 때, 소유자는 자신이 숨겨진 속성들을 알고 있음을 증명해야 한다. k개의 속성 $\{m_1, \ldots, m_k\}$ 안에 l개의 비밀 속성 $\mathbb{H}_l$가 있다고 가정하자. 증명의 구성은 다음과 같다.

1) 임의의 정수 r_A를 고르고, 이를 사용해 *CL signature*를 임의화한다.

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

2) 섹션 *4.3.2.2*에 서술된 프로토콜을 사용해 증명을 정리한다.

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

3) 증명 π를 출력한다.

다음을 통해 실제의 m을 노출하지 않으면서, $m \geq b$임을 증명할 수 있다.

1) $\Delta = m - b$를 계산하고, 이를 *4*개의 정수의 제곱의 합으로 분해한다.

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

2) *4*개의 임의의 수 T_{u_i}를 생성하고, 다음을 증명한다.

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

3) 임의의 수 T_Δ를 생성하고, 다음을 증명한다.

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

4) $m \geq b$를 증명한다.

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

5) (π_1, π_2, π_3)를 출력한다.

π와 (π_1, π_2, π_3)를 합치면, 완전한 증명을 구할 수 있다. 검증자는 섹션 4.3.2.2.에 있는 검증 방식을 사용해 각 부분 증명을 검증할 수 있다. 술어(predicate)는 모든 부분증명(subproof)이 유효한 경우에만 참이다.

5. 분산 원장(DISTRIBUTED LEDGER)

5.1. 온톨로지 원장(ONTOLOGY LEDGER)

5.1.1. 합의 메커니즘(Consensus Mechanism)

온톨로지의 원장은 차세대 온토랜드 합의 엔진(Ontorand Consensus Engine, OCE)를 지원한다. OCE는 dBFT 합의 프로토콜과 verifiable random function(VRF)를 기반으로 한 효율적으로 향상된 버전의 합의 엔진이다. OCE는 거의 무제한의 확장성에 도달했으며 계산 오버헤드가 적다. 또한 비교적 낮은 해시율(hashing rate)이 필요해, 네트워크 포크가 일어나기 어렵다. dBFT는 NEO의 퍼블릭체인과 다른 제휴 체인들을 통해 뛰어난 안정성과 신뢰를 증명해왔다. OCE의 블록생성 속도를 제한하는 것은 거의 인터넷 속도 뿐이며, 보통 10초 이내에 컨펌이 완료된다. OCE는 VRF를 사용해 누가 합의 검증 과정에 참여할지를 결정하며, 비잔티움 장애 허용 알고리즘(Byzantine fault tolerance algorithm)[14][15][16]을 사용해 합의한다. 동시에 시드(seed)는 검증자의 서명에 의해 생성되어, 다음 검증자를 가리킨다. OCE는 또한 장착형 검증기(pluggable verifier)와 프로토콜 복구/업그레이드 기능을 제공한다.

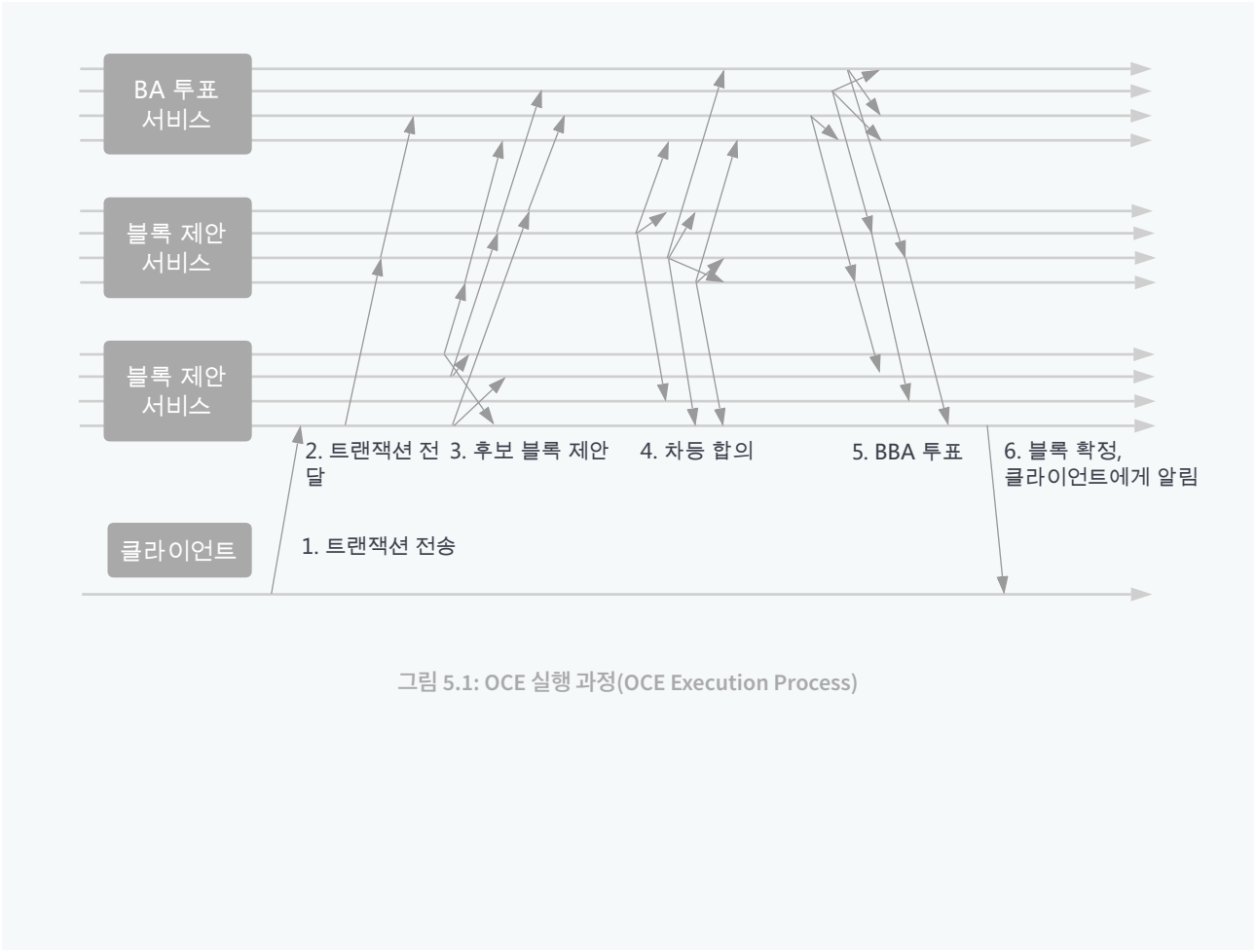


그림 5.1: OCE 실행 과정(OCE Execution Process)

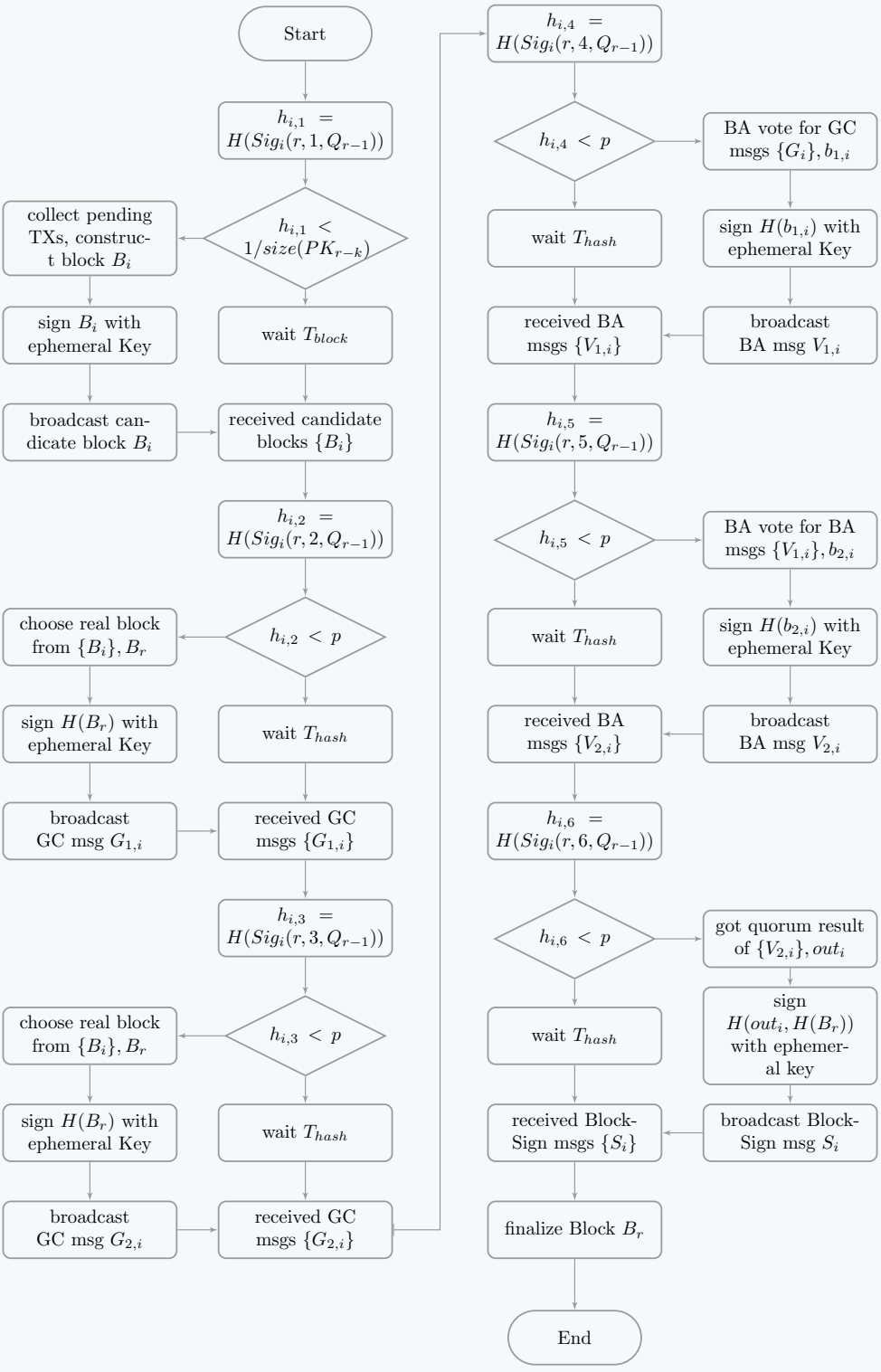


그림 5.2: OCE 순서도(OCE Flow Chart)

그림 5.1과 그림 5.2는 OCE의 구현 프로세스와 알고리즘 순서도를 설명한다.

1) 클라이언트는 *P2P* 네트워크를 통해 서명을 첨부한 데이터 트랜잭션 요청을 브로드캐스트 *(broadcast)*한다.

2) 합의 과정에 참여하는 모든 노드는 모든 트랜잭션을 모니터링하고, 이를 로컬 캐시에 저장한다.

3) 새 블록 제안*:*

a) 노드는 현재 블록 높이*(block height)*과 지난 블록의 *Q value*에 기반한 서명을 만들 수 있으며, 이에 맞는 해시값*(hash value)*를 계산할 수 있다. 계산된 해시값을 사용해 해당 노드가 현재 블록 높이의 제안자가 될 수 있는지를 결정할 수 있다.

b) 한 노드가 자신이 현재 블록 높이*(block height)*의 제안자가 될 수 있다고 평가하면*:*

i) 해당 노드는 현재 그리고 수집한 비협조적인 트랜잭션 요청*(non-consensus transaction)*을 패키징하고, 새 블록을 만든다.

ii) 해당 노드는 새 블록의 데이터와 해시값으로 서명을 만든 후, 블록과 서명을 *P2P* 네트워크로 브로드캐스트한다.

4) 모든 노드는 자신의 블록 제안과 관련한 네트워크 트래픽을 모니터링하며 T_{block}동안 기다리고, 전달받은 블록 제안을 로컬 메모리에 캐시한다.

5) 새로운 블록에 대한 합의*:*

a) T_{block} 후, 각 노드는 자신이 현재 블록 제안의 검증 노드가 될 수 있는지를 판단한다.

b) 한 노드가 자신이 현재 제안 블록의 검증 노드가 될 수 있다고 판단했다면*:*

i) 해당 노드는 모든 블록 제안의 정확성과 무결성을 검증한다.

ii) 모든 블록 제안의 *Q value* 서명의 정확성을 검증한다.

iii) 모든 블록 제안자의 신원이 정당한지 검증한다.

iv) 모든 제안들의 해시값을 계산하고, 가장 작은 해시값을 가진 블록 제안을 현재 블록에서 진짜로 선택된 블록으로 결정한다.

v) 선택된 블록 제안의 해시에 서명하고, 블록 해시 및 서명을 *P2P* 네트워크에 브로드캐스트한다.

c) T_{hash} 시간 후, 노드는 알고리즘 단계를 업데이트하고, 자신이 블록 제안에 대한 검증 과정에 참여해야할지 말지를 평가한다.

d) 노드가 자신이 검증 노드가 되어야한다고 판단했다면*:*

i) 해당 노드는 위에 설명된 것과 같은 방식으로 블록 제안을 검증한다.

ii) 선택된 블록 제안의 해시에 서명하고, 블록 해시 및 서명을 *P2P* 네트워크에 브로드캐스트한다.

6) *P2P* 네트워크에 있는 모든 노드들은 블록 검증 노드들이 브로드캐스트한 합의 메시지들을 감시한다.

a) 합의 메시지의 정확성과 무결성을 검증한다.

b) 합의 메시지를 브로드캐스트한 노드의 신원이 정당한지를 검증한다.

c) 검증 메시지를 로컬 메모리에 캐시한다.

7) 새 블록에 대한 비잔틴 투표*(Byzantine election):*

a) T_{hash} 시간 후, 노드들은 알고리즘 단계를 업데이트하고, 자신들이 새 블록에 대한 투표에 참여해야하는지 아닌지에 대해 평가한다.

b) 만약 한 노드가 자신이 투표에 참여해야한다고 결정한다면, 아래의 단계들을 실행한다*:*

i) 각각의 블록 제안에 대한 합의 검증 득표 수를 센다.

ii) 정족수에 따라 합의 결과에 투표한다.

iii) P2P 네트워크 투표 결과에 서명하고, 선택된 블록의 해시를 계산한다. 투표 결과와 서명을 *P2P* 네트워크에 브로드캐스트한다.

c) 네트워크에 있는 모든 노드는 자신이 속한 블록체인의 설정값에 따라 투표를 여러 차례 할 수도 있다. 각 횟수마다 T_{hash}의 시간을 기다려야하며, 각 투표는 독립적으로 진행된다.

8) 네트워크 상의 모든 노드는 브로드캐스트된 투표 메시지의 유효성을 지속적으로 감시하고 검증한다.

9) 새 블록의 최종 서명*:*

a) 비잔틴 투표를 완료한 후, 노드들은 알고리즘 단계를 업데이트하고, 각자가 새로운 블록의 최종 서명에 참여하는지 여부를 확인한다.

b) 노드가 최종 서명 생성에 참여해야한다고 결정한 경우, 아래의 과정을 실행한다.

i) 블록 제안이 합의를 위한 정족수에 도달되었는지를 투표 수를 토대로 판단한다.

ii) 정족수에 도달되지 않았다면, 새 블록에 대한 합의 결과가 거짓인 것으로 결정한다.

iii) 투표 메시지에 대한 결과를 패키징하고 서명한다.

iv) 최종 블록 해시와 투표 결과에 대한 서명을 *P2P* 네트워크에 브로드캐스트한다.

10) 모든 노드는 새 블록의 최종 서명 메시지를 지속적으로 감사하고, 서명 메시지의 유효성을 검증한다.

a) T_{hash} 후, 노드들은 전달받은 블록 서명 메시지에 따른 현재의 블록 높이의 블록 해시를 계산한다.

b) 계산한 블록 해시와 전달받은 블록 제안 메시지를 비교해 현재 블록 높이에 대한 최종 블록을 구한다.

한편, 블록의 Q값은 최종 합의 블록을 기반으로 계산한다. 이후 다음 블록 높이를 위해 합의 과정을 초기화한다.

온톨로지 원장은 합의 알고리즘을 쉽고 빠르게 대체할 수 있도록 모듈화된 구조를 채택한다. 따라서 다양한 시나리오에 따라 PoS, DPoS, Raft와 같은 알고리즘을 선택할 수 있다.

5.1.2. 절차 프로토콜(Procedure Protocols)

분산 절차 프로토콜은 분산 원장 기술, 엔티티 크로스체인, 크로스시스템 프라이버시, 특정 크로스체인 프로토콜을 통해 이뤄진다. 절차와 트랜잭션의 여러 단계가 다른 블록체인 또는 시스템에 분산되어 엔티티의 신원 정보를 보호하고, 전체 트랜잭션의 일관성을 보장한다.

5.1.3. 인증서 설계(Attestation Design)

분산 원장은 데이터 자체뿐 아니라 데이터가 어떻게 사용되는지도 증명한다. 더 자세히 말하자면 데이터 연결, 데이터 검색, 데이터 사용 등을 비롯한 모든 요청과 요청에 대한 과정을 모두 원장에 기록한다. 이를 통해, 데이터에 대한 안전과 신뢰성을 보장한다.

5.2. 스마트 컨트랙트(SMART CONTRACT)

Go로 구현된 온톨로지 원장의 NeoVM 가상머신은 스마트 컨트랙트 실행 환경이며, 어플리케이션 레이어 프레임워크의 지능 제어 레이어(intelligent control layer)를 구현할 수 있다. NeoVM의 튜링 완전성은 임의의 로직을 구현할 수 있게 하며, 특정 입력값에 대해 항상 같은 결과를 보장한다. 따라서 온톨로지 스마트 컨트랙트는 높은 수준의 확실성이 요구되는 시나리오에 적합하다. 또한 NeoVM은 동적 샤딩을 가능하게 하는 “deterministic call tree” 기술로 높은 확장성을 가진다. 이를 통해 온톨로지는 이론적으로 무제한의 확장성을 가진다.

이로 인해, Java, C/C#, Go, Python, Javascript 혹은 다른 언어의 개발자들은 새로운 언어를 배우지 않아도 스마트 컨트랙트를 작성할 수 있다. 플랫폼이 제공하는 쉬운 운영과 유지보수는 개발자들이 다른 파트너들(특히 정보 공유자들)과 쉽게 공유할 수 있게 하며, 스마트 컨트랙트의 편집과 접근성에 도움을 줄 것이다.

가상머신의 외부 인터페이스는 계정 정보 및 외부 데이터와 유연하게 작용할 수 있는 API를 사용해 커스터마이즈할 수 있다. 이는 스마트 컨트랙트가 실행될 때, 네이티브 코드가 높은 성능을 달성할 수 있게 해준다. 동시에 다른 블록체인을 지원하는 가상머신 메커니즘도 구현되었다.

5.3. 공유 데이터 컨트랙트 모델(SHARED DATA CONTRACT MODEL)

어플리케이션이 분산 원장에서 필요한 요구사항은 크게 데이터 구조의 정의와 저장, 비즈니스 로직 처리 및 외부 시스템과의 작용 두 가지로 나뉜다. 온톨로지는 시스템이 더 큰 확장성과 유연성을 갖도록 두 부분이 분리된 모델을 설계했다. Data Contract는 데이터 저장을 위한 부분이고, Controller Contract는 비즈니스 로직을 위한 부분이다. 우리는 이 모델을 공용 데이터 컨트랙트 모델(Shared Data Contract Model)이라 부른다.

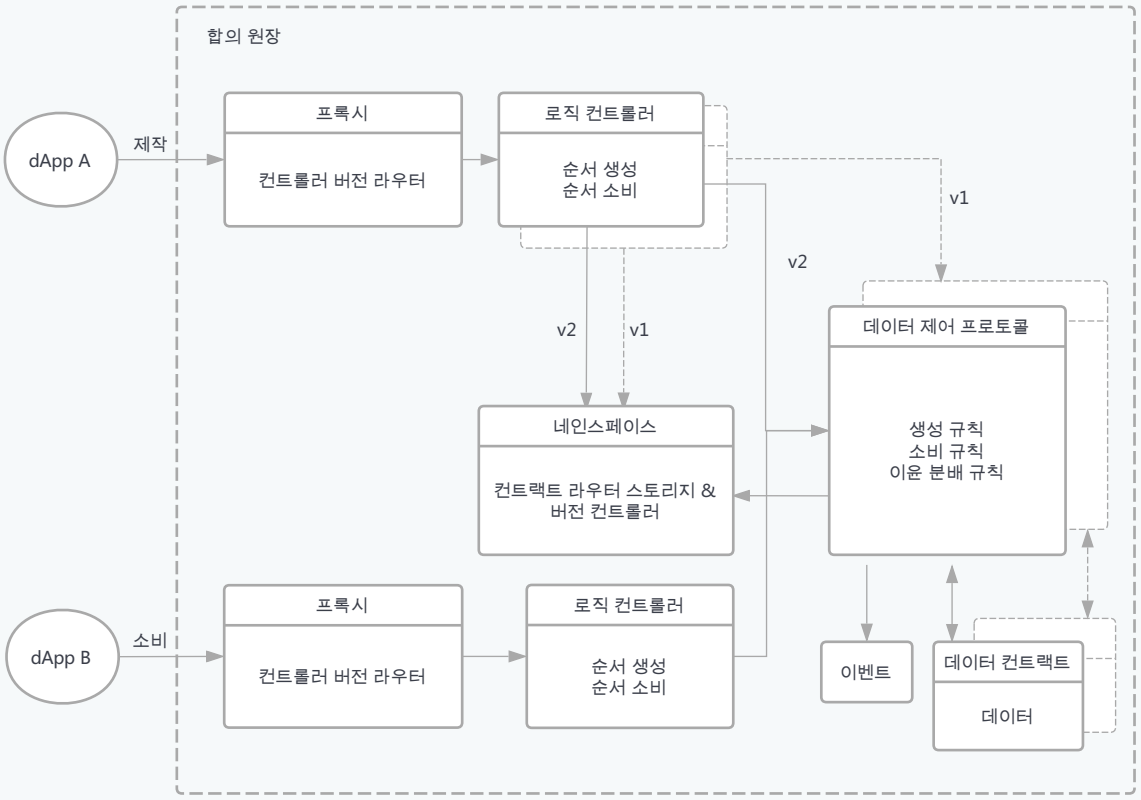


그림 5.3: 공유 데이터 컨트랙트 모델

이 모델은 다음의 설계 요소를 포함한다:

- 에이전트 컨트롤러*(Agent controller): Providing an external dApp defined contract entry for other dApps, guaranteeing that even if the contract is upgraded it will not cause inconsistencies in external transfers.*
- 로직 컨트롤러*(Logic controller):* 비즈니스 로직을 위한 컨트롤러이다.
- 네임스페이스*(Namespaces):* 각 버전의 서로 다른 *dApps* 간의 데이터 컨트랙트 주소 매핑을 지원한다. 데이터 구조를 업그레이드하더라도 비즈니스 로직에 영향이 가지 않게 해주며, 버전 간의 데이터 백트래킹을 지원한다.
- 데이터 컨트랙트*(Data contract):* 기본적인 데이터구조와 저장 인터페이스를 제공한다. *GET/SET* 메소드를 제공한다는 점에서 *DAO*와 비슷하다.
- 데이터 제어 프로토콜*(Data control protocol):* 비즈니스 당사자들이 정의하고 디지털화한 일반적인 비즈니스 규칙이다. 이 프로토콜은 다음의 요소들을 포함해야 한다.
  - 로직 컨트롤러에 대한 접근 제어
  - 외부 시스템 인터페이스와의 상호작용
  - 공통적인 서비스 로직
  - 비즈니스 핵심 원장을 위한 제어 로직

- 이벤트 알림 메커니즘

- 이벤트*(Event):* 이벤트는 스마트 컨트랙트를 실행하는 동안 콘텐츠를 밀어내는 메커니즘을 말한다. 각 당사자는 원장을 동기화하는 과정에서 분산 원장 블록을 로컬에서 재생하며 트랜잭션 정보를 얻을 수 있다. 이벤트는 비용이 낮은 저장 메커니즘이며, 컨트랙트 노드가 필요하지 않은 노드는 컨트랙트를 구현하지 않고 이벤트 정보를 받지 않도록 결정할 수 있다. 특정 컨트랙트와 관련된 노드는 해당 컨트랙트를 실행하면,관련 이벤트를 얻을 수 있다. 체인이 실제 트랜잭션의 내용을 저장하지 않기 때문에, 각 노드가 저장에 대해 가지는 압박을 크게 줄일 수 있다.

이 컨트랙트 모델은 dApp이 일반적인 프로토콜을 이용하여 다른 비즈니스들과도 자신들의 데이터를 공유할 수 있게 한다. 이는 각 영역 간의 협력을 더 수월하게 한다.

그림 5.4는 주문 등록과 트랜잭션(소비)로 이뤄진 데이터 교환 과정을 설명한다.

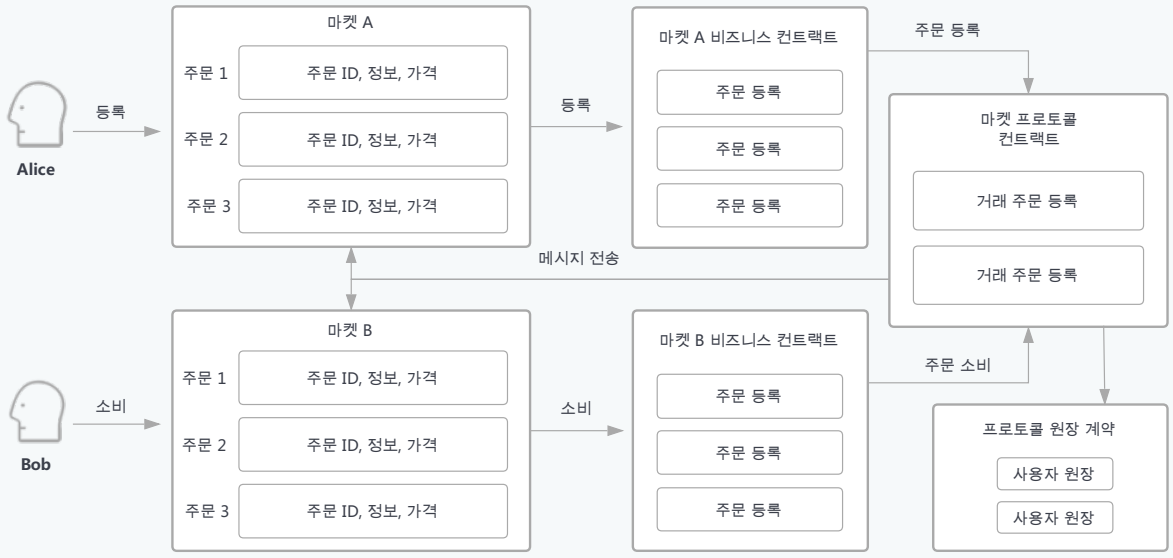


그림 5.4: 데이터 교환 과정(Data Exchange Process)

주문 등록(Registration):

1) 앨리스는 마켓 *A*에 주문 등록 요청을 개시한다.

2) 마켓 *A*는 앨리스의 주문을 확인하고, 분류 프로토콜 컨트랙트가 주문을 분류에 따라 등록한다.

3) 마켓 프로토콜 컨트랙트는 마켓 *A*의 권한과 주문 내용을 확인한다. 만약 기준이 만족되면, 컨트랙트는 주문을 원장에 등록한다.

4) 등록이 성공적으로 완료되면, 프로토콜 컨트랙트는 주문 등록 메시지를 브로드캐스트한다.

5) 마켓 *A*는 최종 결과를 앨리스에게 반환한다.

트랜잭션(Transaction):

1) 마켓 B는 블록체인과 동기화하여 모든 트랜잭션을 얻는다.
2) 마켓 B는 실행된 블록의 트랜잭션으로부터 마켓 A가 브로드캐스트한 주문 등록 메시지를 얻는다.
3) 주문 등록 메시지에 있는 주문 정보가 *Market B*에 표시된다.
4) 밥은 앨리스가 등록한 주문을 위해 마켓 B로의 트랜잭션을 개시한다.
5) 마켓 B는 밥의 주문을 확인하고, 분류 프로토콜 컨트랙트가 주문을 분류에 따라 등록한다.
6) 프로토콜 컨트랙트는 마켓 B의 권한과 트랜잭션 요청 정보를 검사한다. 요구사항이 충족되면, 마켓 B의 트랜잭션 요청을 처리하고 최종 결제를 수행한다.
7) 프로토콜은 주문의 트랜잭션 완료 정보를 브로드캐스트하고, 결과를 마켓 B에 반환한다.
8) 마켓 B는 최종 결과를 밥에게 반환한다.
9) 마켓 A는 브로드캐스트된 프로토콜 컨트랙트 완료 정보를 받고, 트랜잭션 완료 정보를 앨리스에게 보낸다.

5.4. 머클 트리 스토리지 모델(MERKLE TREE STORAGE MODEL)

비트코인이나 이더리움 같은 디지털 통화 플랫폼들의 경우, 클라이언트는 종종 자신의 계정에만 초점을 둔다. 완전한 동기화 없이 계정을 검증하기 위해, 온톨로지는 Simply Payment Verification(SPV) 기술을 고안했다. SPV는 머클 증명(Merkle proofs)을 만듦으로써 많은 양의 저장 공간을 절약하고, 네트워크 전송량을 줄일 수 있다[17]. 온톨로지에서 클라이언트는 다른 클라이언트들의 신원 정보를 검증해야 한다. 따라서 원장은 머클 증명 구축을 지원해야 한다.

5.4.1. 머클 해시 트리(Merkle Hash Tree)

이진 머클 해시 트리는 효율적인 감사 작업에 사용된다. n개의 정렬된 입력값, $D[n] = (d_0, d_1, \ldots, d_{n-1})$이 있다고 가정할 때, 머클 트리 해시(Merkle Tree Hash, MTH)는 다음과 같이 정의한다.

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

여기서 k는 n보다 작은 수 중 가장 큰 2의 제곱수이며, $D[a:b]$는 D 배열의 d_a부터 d_{b-1}까지의 부분 배열을 말한다. ||는 두 바이트 배열을 연결하는 연산자다.

5.4.2. 머클 감사 경로(Merkle Audit Path)

머클 해시 트리(Merkle Hash Tree)에 대한 머클 감사 경로(Merkle Audit Path)는 머클 해시 트리를 계산하기 위해 필요한 최단 노드 목록이다. 트리 내의 각 노드는 리프 노드거나 바로 아래의 두 자식 노드들로 계산한 결과이다. 감사 경로의 노드는 루트로 향하는 트리의 각 과정에서 지금까지 연산된 노드들과 결합된다. 다시 말해, 감사 경로는 리프에서 루트까지 다다르는데까지 필요한 노드 중 유실된 노드의 배열로 이루어진다. 만약 감사 경로를 통해 계산된 루트가 진짜 루트와 일치한다면, 감사 경로는 리프가 트리에 존재한다는 증거이다.

정렬된 n개의 입력 값으로 이뤄진 배열이 트리, $D[n] = (d_0, d_1, \ldots, d_{n-1})$의 형태로 주어졌을 때, (m+1)번째 입력 값 $d(m) : 0 \leq m < n$에 대한 머클 감사 경로, $PATH(m, D[n])$는 다음과 같이 정의된다.

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

여기서 +는 두 배열을 연결하는 연산자이다.

그림 5.5는 머클 감사 경로의 예시이다.

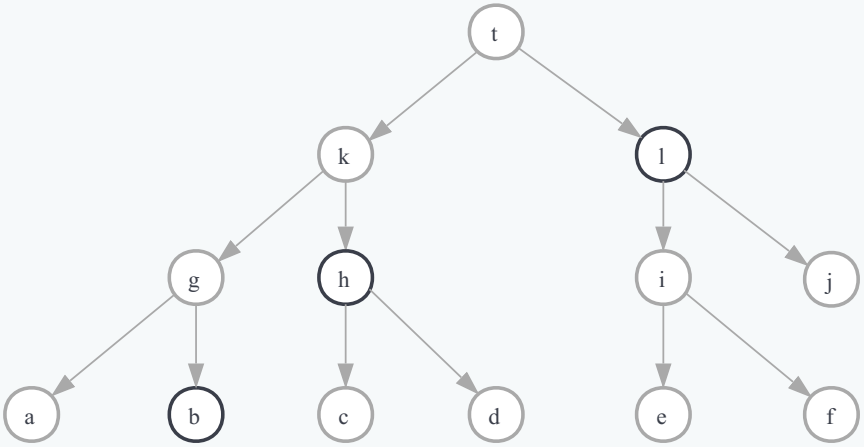


그림 5.5: a의 감사 경로는 [b, h, l]이다.

5.4.3. 머클 일관성 증명(Merkle Consistency Proofs)

머클 트리 해시 $MTH(D[n])$와 이전 해시 $MTH(D[0:m])$의 첫 m개의 리프들($m \leq n$)에 대한 머클 일관성 증명은 각 트리의 첫 m개의 입력값 D[0:m]이 서로 같다는 것을 검증할 때 필요한 노드들의 배열이다. 다음의 알고리즘을 사용해 최소한의 일관성 증명을 구성할 수 있다.

n개의 정렬된 배열로 이루어진 입력값 $D[n] = (d_0, d_1, \ldots, d_{n-1})$가 있을 때, 이전 머클 트리 해시 $MTH(D[0:m])$에 대한 머클 일관성 증명 $PROOF(m, D[n])$은 다음과 같이 정의할 수 있다.

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

그림 5.6은 머클 일관성 증명의 예시이다.

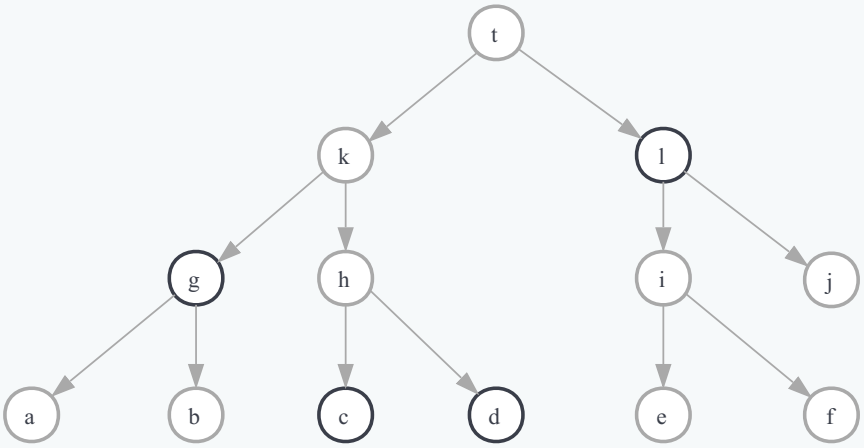


그림 5.6: $PROOF(3, D[7])$ is [c,d,g,l]

5.4.4. 머클 패트리샤 트리(Merkle Patricia Tree)

예를 들어 한 엔티티의 아이덴티티를 증명할 때와 같은 몇몇 시나리오에서는 트랜잭션의 최종 상태를 빨리 증명해야 한다. 머클 증명을 사용한다면 과거의 트랜잭션을 하나하나 증명해야 하지만 머클 패트리샤 트리(Merkle Patricia Tree, MPT)를 사용하면 더 효율적으로 요구사항을 충족할 수 있다.

MPT는 패트리샤 트리(Patricia Tree)[21]와 머클 트리의 결합이다. Key-value 매핑, 암호기술에 기반한 조작할 수 없는 자료 구조를 가지며, 결정성(determinacy), 고효율성, 보안의 특성을 갖는다.

- 결정성*(Determinacy):* 데이터를 찾을 때 같은 *key-value*는 항상 같은 결과를 반환하고, 같은 루트 해시를 가진다.
- 효율성*(Efficiency):* 새 루트를 빠르게 계산할 수 있다. 데이터가 변경될때, 삽입, 검색, 갱신, 삭제에 대한 시간 복잡도는 $O(\log_2 n)$이다.
- 보안*(Security):* 누군가 다른 이를 *DOS* 공격으로 악의적으로 공격하고 트리에 대한 제어를 얻으려는 시도를 할 수 있다. 제한된 트리의 깊이는 이러한 시도를 쓸모 없게 만들 것이다.

5.5. HYDRADAO

HydraDAO는 스마트 컨트랙트, 크로스체인, 크로스데이터 협력을 통합하는 데이터 예측 및 상호작용 모듈이다. 온톨로지의 DAO(distributed autonomous organization)와 체인 간 데이터 교환 기능을 포함한다. 온톨로지의 거버넌스 메커니즘은 민주적이며, AI에 의해 자동화된 발의, 투표, 검증 기능을 지원한다. 이 과정 중 고유한 DAO 주소와 토큰 풀이 생성되며, 이를 통해 DAO는 온톨로지에 자동으로 자금과 결과를 추가할 수 있다. 투표가 끝나면, DAO는 자동으로 조작불가능한 스마트 컨트랙트를 실행한다. 이 메커니즘은 온톨로지에서의 데이터 교환과 거버넌스가 유연하게 작동하게 해주며, 대규모의 자동화된 네트워크 작업을 위한 기술을 기원한다.

온톨로지의 오라클 스마트 컨트랙트 구현 로직은 다음과 같다.

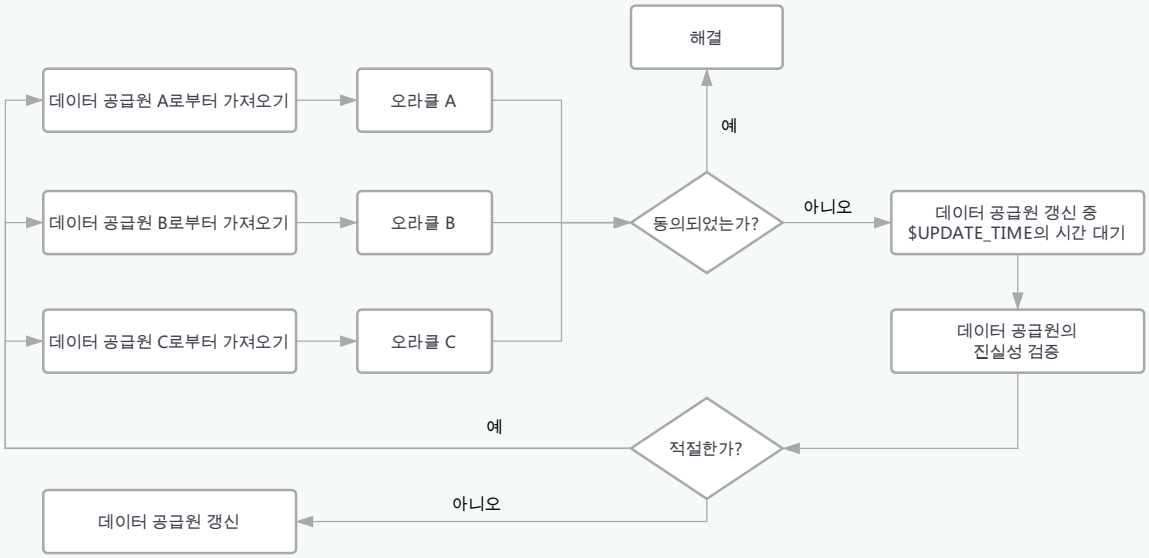


그림 5.7: HydraDAO 로직 처리 과정(Processing Logic of HydraDAO)

1) 모든 트랜잭션은 시작이나 종료 시에 오라클 스마트 컨트랙트를 사용할지 결정할 수 있다. 트랜잭션을 생성하거나 트랜잭션의 일부를 블록체인에 저장할 때, 오라클 스마트 컨트랙트와 관련 매개 변수를 설정해야만 한다. 트랜잭션이 끝나면 결과를 확인하고 유효한 데이터를 제공한 참가자들에게 토큰을 지급할 컨트랙트 로직이 자동으로 실행된다. 온톨로지는 *DDEP*의 실시간 *API*에 대한 접근을 통해 트랜잭션 결과를 분석하고 예측하여 무결성을 검증할 수 있도록 대용량 처리에 대한 컴퓨팅 파워를 지원한다. 트랜잭션에 참여하기 전, 트랜잭션 참여자는 신뢰 데이터 서비스의 일환으로 신뢰받는 데이터 공급원을 연결하거나 온톨로지 접근 서비스를 사용해 정확한 트랜잭션 상태를 검증할 수 있다.

2) 블록체인의 상태에만 의존하는 트랜잭션의 경우 오라클 스마트 컨트랙트는 권한 부여를 통해 블록체인 인터페이스에 직접 데이터를 쿼리한다.

3) 실제 세계와 관련된 대부분의 트랜잭션들의 경우, 오라클 스마트 컨트랙트는 현실 세계의 트랜잭션 결과를 얻어야한다. 오라클 스마트 컨트랙트가 실제 세계의 데이터에 접근하거나 읽으려할

때, 각각의 *hydraOracle*은 여러 신뢰 공급원 중 하나를 임의로 고르거나 데이터 중재자에 대한 임계값을 특정한다. 트랜잭션이 끝나면, 전역 상태를 검증하고, 특정된 신뢰 공급원[2]을 통해 갱신한다.

5.5.1. 내장 DAO 데이터 예측(Built-In DAO Data Prediction)

탈중앙화 시스템의 매력은 낮은 비용 및 효율성의 증진 뿐 아니라, 분산된 집단 의사 결정이기도 하다. 온톨로지에서 컨트랙트를 실행할때, HydraDAO를 연결해 "공개 테스트"를 활성화시킬 수 있는 옵션을 선택할 수 있다. 예를 들어, 사용자가 특정한 수량의 외환을 구매하길 희망하는데 대기 중인 주문이 정확한지 확신하지 못한다면, DAO 프로젝트 제안을 개시하고 특정 수량의 마진을 담보로 예치할 수 있다. 충분한 수의 승인이 이뤄지면, 제안은 예측 단계로 진입한다. 제안을 받아들이는 참여자는 사실 진술서와 서명을 첨부할 수 있다. 참여자는 자신의 주문을 언제든 수정할 수 있으나, 수정할 때는 추가 마진이 필요하다. 예측 기간이 끝나면, DAO 컨트랙트는 자동으로 마진을 잠그고, 제안의 규칙에 따라 가장 좋은 제안을 골라낸다. 가장 정확한 예측을 한 참여자는 리워드를 받을 것이다. 가장 좋은 결과에 따라 사용자가 개시한 트랜잭션 또한 자동으로 실행된다.

제안이 DAO에 제출되면, 전체 네트워크의 토큰 보유자들은 새 제안을 평가하고 결정한다. 이 기간동안 참여자들은 언제든 사실 진술서를 제출하거나 수정할 수 있다. 진술서를 수정할 때 투표와 랭킹이 필요하다. 온톨로지의 DAO 인센티브 메커니즘은 사실에 가장 근접하고 가장 많은 투표를 받는 제안서가 최대 보상을 받을 수 있게 한다.

HydraDAO는 예측 기간이 완료되면, 자동으로 조정을 시작한다. 조정이 시작되면, HydraDAO는 모든 제출을 정렬하고 계산할 것이며, DAO 컨트랙트의 자금을 리워드 수령자들의 계정으로 할당할 것이다. HydraDAO는 마지막으로 스마트 컨트랙트에 연결할 사실 상태(fact status)를 출력할 것이다.

5.5.2. 신뢰할 수 있는 외부의 데이터 공급원(External Trusted Data Source)

분쟁을 피하기 위해, 미리 정의된 설정가능한 오라클 스마트 컨트랙트는 스마트 컨트랙트의 검증 교환이 충족되기 전에 가치 교환이 이뤄지지 않게 한다. 오라클 스마트 컨트랙트는 손쉬운 정의와 개발을 위해 JSON API를 지원한다.

신뢰할 수 있는 데이터 공급원에 대한 접근 과정:

1) 오라클 스마트 컨트랙트를 생성하기 위해, 데이터의 장소, 데이터의 업데이트 빈도, 유효 기간, *API* 인증서, 개인 정보 보호 정책 등의 변수를 준비한다.

2) 다른 검증자들이 평가할 수 있도록 스마트 컨트랙트를 정의한다.

간단하고 이해하기 쉬운 용어를 사용하여, 명료하게 컨트랙트의 기능과 내용을 설명해야 한다. 컨트랙트의 레이블은 컨트랙트을 블록체인에서 쉽게 찾을 수 있게하며, 투명성을 증진시킨다. 상세하게 정리된 *"if... then..."*과 같은 문구를 사용한 묘사는 실행될 트랜잭션을 잘 설명할 수 있다.

3) 현실에서의 법적 문서와의 연계*(Association with real world legal documents)*

2. 모든 데이터의 사용은 데이터 소유자 혹은 관련 허가자의 권한을 필요로 하며, 개인 정보 보호 관련 규제를 따른다.

오라클 스마트 컨트랙트는 관련한 법적 문서를 서명하기 위해 컨트랙트 출판자의 전자 서명을 사용한다. 법적 문서를 업로드함으로써 증거를 블록체인에 추가할 수 있다.

4) 컨트랙트의 공동서명자에게 알림*(Notifying co-signers of the contract)*

계약 조건 및 이해관계자에 따라, 서명을 해야하는 참여자는 *IM*이나 *email*, 전화 혹은 다른 수단으로 통지받는다. 컨트랙트는 모든 참여자의 서명을 받기 전까지 실행되지 않는다. 데드라인 전까지 모든 참여자들이 서명하지 않을 경우, 컨트랙트는 실패한다. 참여자가 서명할 경우, 오라클 스마트 컨트랙트는 해당 참여자에게 자금 에스크로 주소를 할당한다.

5) 에스크로 스마트 컨트랙트에 대한 선불금 기탁*(Recharging tokens for the Oracle smart contract escrow address)*

외부의 데이터 공급원을 사용하는 것은 비용을 초래한다. 사용자는 스마트 컨트랙트가 관리하는 주소로 선불금을 보내야 한다.

6) 계약 완료*(Complete the contract release)*

계약이 서명되고 에스크로 주소로부터 자금 기탁이 완료되면, 컨트랙트가 실행된다. 트랜잭션이 완료되면, 에스크로 주소의 스마트 컨트랙트가 계약 사항에 따라 자금을 지불할 것이다. 트랜잭션의 요구사항에 부합하지 못하는 자금이나 모든 부정한 트랜잭션은 자동으로 에스크로 계정으로 반환될 것이다.

*7) (*선택 사항*)* 데이터 오픈 플랫폼, 오픈 데이터 접근 프로토콜*(Data open platform, open data access protocol)*

오라클 스마트 컨트랙트는 몇몇 데이터 공유 플랫폼에 대해 제공된 *API*와 데이터 사용 프로토콜에 따라 관련한 오프체인 데이터와 직접 상호작용할 수 있다. 투명한 조건에 따라, 전체 과정에 대한 조건과 법적 정보는 공개적으로 표시되며, 모든 블록체인 사용자는 관련 정보를 쿼리하고 복구할 수 있다. 이러한 데이터가 가지는 비영리적 본질로, 오라클 스마트 컨트랙트의 구성 과정을 매우 단순할 것이다. 지불 관계를 유지할 필요 없이 데이터 및 상태 업데이트만 해주면 된다.

6. 코어 프로토콜(CORE PROTOCOLS)

6.1. 다중 출처 인증 프로토콜(MULTI-SOURCE AUTHENTICATION PROTOCOL)

다중 출처 인증은 기존의 단일 요소 인증 시스템과 다르다. 온톨로지는 외부의 신원 정보와 온톨로지 내 엔티티들의 보증을 통합한 다중 출처 인증 시스템을 제공한다. 엔티티는 자신이 누구인지에 대한 정보를 제공할 수 있을 뿐만 아니라, 광범위한 신원 프토폴리오를 만들기 위해 자신이 무엇을 소유하는지, 무엇을 원하는지, 어떤 능력이 있는지 등을 제공할 수 있다.

다중 출처 인증 프로토콜은 다음의 두 방식을 제공한다.

- 외부 신뢰 인증*(External trust certification)*: 온톨로지는 검증가능한 내역을 사용해 *ONT ID*를 외부 신뢰 공급원에 연결할 수 있다. 모든 엔티티는 다른 엔티티의 *ONT ID*에 연결된 외부 신뢰 공급원을 검증해 신원을 검증할 수 있다. 한 엔티티의 인증에 관한 신뢰 여부는 *ONT ID*에 연결된 외부 신뢰 공급원의 신뢰 여부로 결정된다.
- 온톨로지 엔티티 간의 인증*(Authentication between Ontology entities):* 온톨로지의 엔티티들은 검증가능한 내역을 발급함으로써 서로 신원을 인증할 수 있다.

6.1.1. 외부 신뢰 인증(External Trust Certification)

외부 신뢰 인증은 자기 정보 제공(self-introduction)과 신뢰 중재자로부터 획득(importing through trust anchors)의 두 가지 방식을 지원한다.

6.1.1.1. 자기 정보 제공(Self-Introduction)

사용자들은 소셜 미디어, 온라인 뱅킹 등의 기존 신뢰 시스템을 사용해 신뢰를 구성할 수 있다. 원리는 매우 간단하다. 먼저 사용자는 외부 신뢰 출처의 주소를 온톨로지에 추가한다. 그리고 사용자는 다음의 형식을 따르는 신뢰할 수 있는 내역을 제공한다.

- 내역 생성 시기와 만료 기간
- 내역의 내용*(Claim content):* 내역의 종류, *ONT ID,* 소셜 미디어의 종류, 소셜 미디어의 유저 네임 등
- 서명*(Signature): ONT ID*에 포함되어 있는 공개 키

써드 파티가 사용자의 외부 신원을 검증해야할 때, 써드 파티는 먼저 온톨리지 내 사용자의 신뢰 공급원의 인증서 주소를 확인한다. 그후, 검증가능한 내역을 얻기 위해 해당 주소로 이동하고, 검증가능한 내역을 검증한다.

6.1.1.2. 신뢰 중재자로부터 획득(Importing through Trust Anchors)

신뢰 중재자는 주료 정부 기관이나 다른 공공 기관, 기업, NGO, 검증을 거친 권위를 가진 개인이다. 신뢰 중재자는 엔티티를 인증하기 위해 자신만의 인증 방식을 사용하고, 인증한 엔티티에게 검증가능한 내역을 발급한다. 내역은 인증된 엔티티의 신원 정보를 포함하지 않는다. ONT ID와 인증 서비스만이 기록된다. 인증 모델은 아래와 같다.

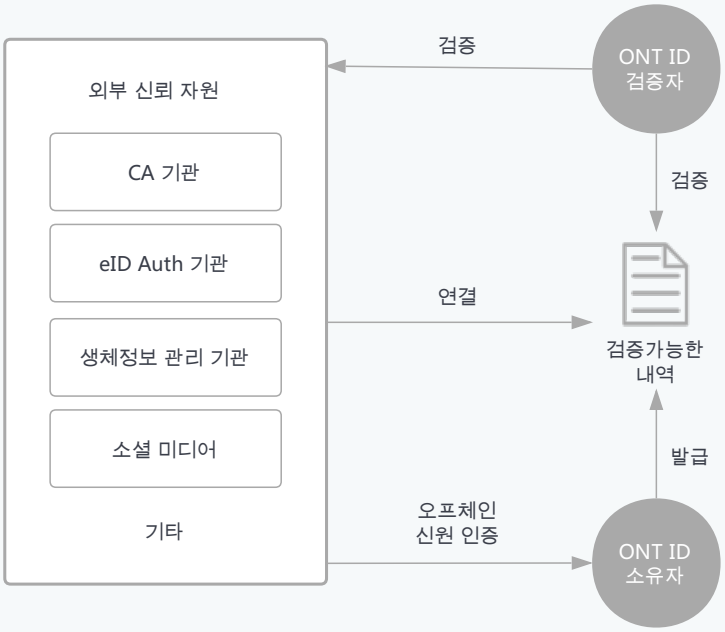


그림 6.1: 신뢰 중재자로부터 가져오기(Importing through Trust Anchors)

6.1.2. 온톨로지 엔티티 간의 신원 인증(Identity Authentication between Ontology Entities)

그림 6.2에 설명된 것과 같이, 온톨로지의 엔티티들은 외부 신뢰 공급원을 통해 신원 인증을 통과했던 다른 엔티티를 통해 신원을 인증할 수도 있다.

온톨로지의 모든 엔티티는 다른 엔티티에 관한 어떤 정보에 대해서라도 검증가능한 내역을 만들 수 있다. 이를 통해, 온톨로지는 전통적인 신원 인증 개념을 뛰어넘은 신원 인증 방식을 제공한다. 정부 기관, 학교, 병원, 은행, 산업, 가정, 친구, 파트너, 공동체 지도자, 동료, 선생님 등으로부터 내역을 모음으로써, 전통적인 시스템보다 크게 효율적인 다면적 인증 시스템을 만들어낼 수 있다.

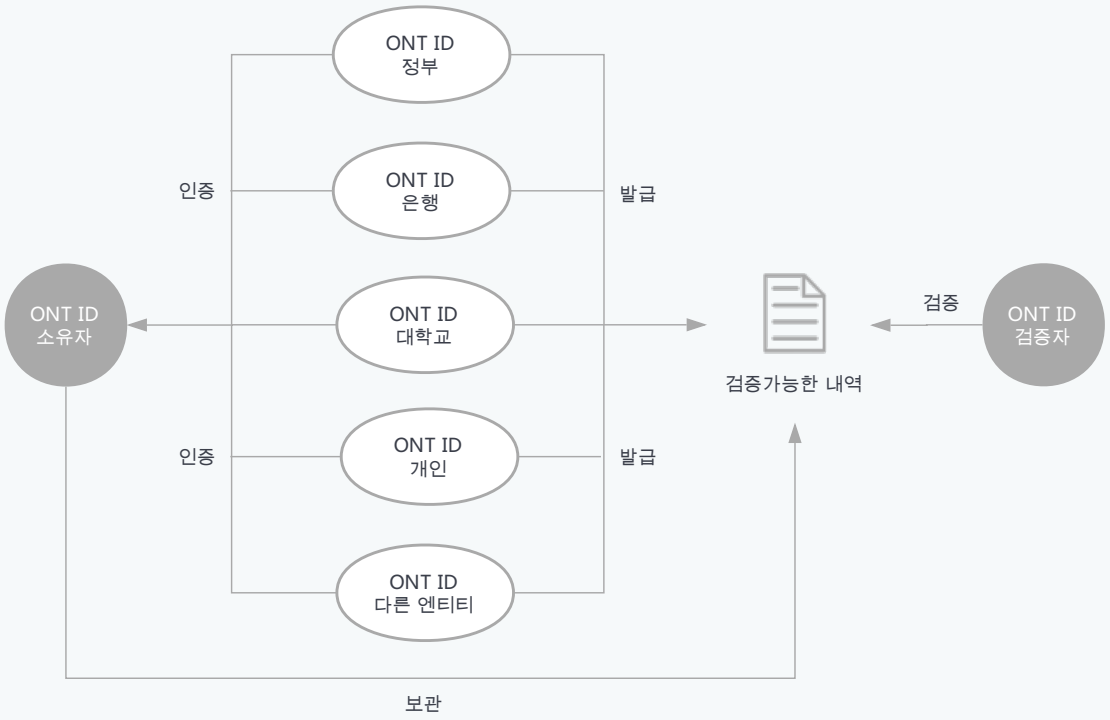


그림 6.2: 엔티티 간의 인증(Authentication between Entities)

6.2. 사용자 권한 관리 프로토콜(USER AUTHORIZATION PROTOCOL)

온톨로지에서 사용자는 자신의 데이터를 완전히 통제할 수 있다. 데이터와 관련한 모든 접근 및 트랜잭션은 사용자의 허가가 필요하다. 이에 비추어 사용자의 데이터 프라이버시를 보호하기 위해 사용자 권한 관리 프로토콜을 준비했다. 프로토콜은 검증가능한 내역을 사용해 비동기 인증을 수행하며, 인증 위임과 높은 수준의 접근 제어를 지원한다.

6.2.1. 역할(Roles)

사용자 권한 관리 프로토콜에 참여하는 주요 역할은 다음과 같다.

- 사용자*(User)*: 엔티티는 자원에 대한 소유권을 가지고, 자원에 대한 접근 권한을 부여할 수 있다.
- 자원 요청자*(Resource Requester)*: 사용자 데이터와 다른 자원을 얻어야 하는 참여자이다.
- 자원 제공자*(Resource Provider)*: 사용자 데이터와 자원을 제공하는 서비스 제공자이다.
- 인증 서버*(Authorization Server)*: 인증 요청을 받아 처리하며, 사용자를 위해 인증 위임을 제공하는 서버이다.

6.2.2. 권한 관리(Authorization)

사용자 인증 프로토콜은 아래 세 단계로 나뉜다.

1) 상호 등록*(Mutual registration):* 사용자는 자원 제공자가 특정한 자원을 인증 서버에 등록할 수 있도록 권한을 부여하고, 인증 서버는 자신의 주소를 자원 제공자에게 등록한다.

2) 접근 제어 정책 설정*(Access control policy setting):* 상호 등록이 완료되면, 사용자는 인증 서버에 접근할 수 있으며, 자원에 대한 접근 제어 전략을 설정할 수 있다.

3) 권한 부여*(Authorization):* 자원 요청자는 접근 권한 부여 요청을 개시한다. 권한 부여에 대한 조건이 충족되면, 요청자는 자원 제공자에게 데이터를 요청할 때 사용할 권한 인증서를 받는다.

6.2.3. 상호 등록(Mutual Registration)

상호 등록은 자원 제공자가 데이터 접근 요청을 처리할 때 요청자에게 인증 서버의 주소를 반환할 수 있도록, 인증 서버가 자신의 주소를 자원 제공자에게 등록하는 것을 말한다. 동시에 사용자가 자신의 자원 제어에 접근할 수 있도록, 자원 제공자는 자신의 주소를 인증 서버에 등록해야한다.

인증 작업을 수행하기 전, 사용자는 자원 제공자와 협력해야 하며, 인증 서버는 상호 등록 과정을 완료해야한다.

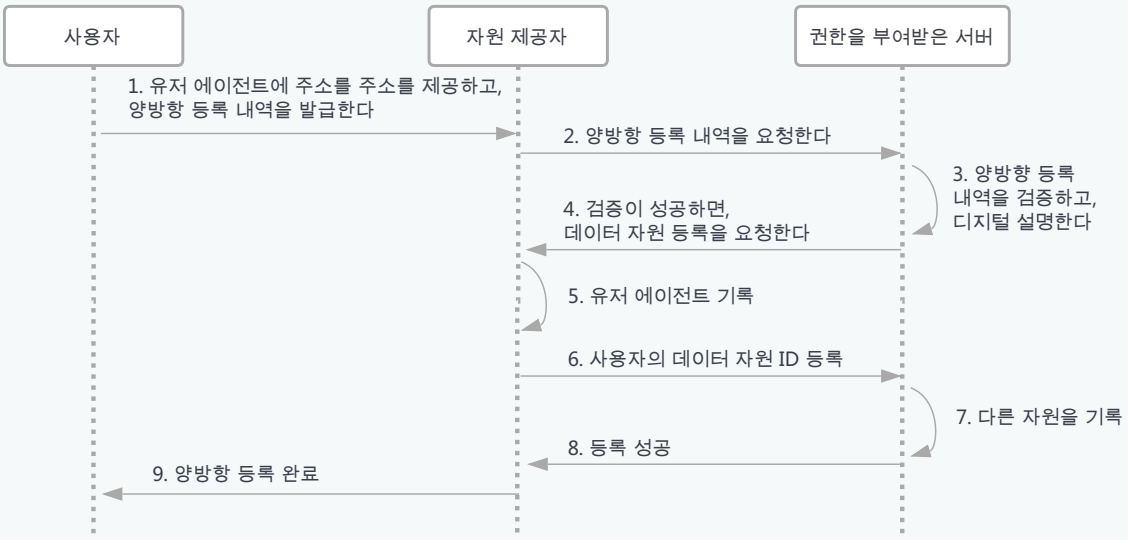


그림 6.3: 상호 등록 과정(Two-Way Registration Process)

과정은 다음과 같다.

1) 사용자가 상호 등록 과정을 개시한다. 사용자는 자신의 *ONT ID,* 자원 요청자의 주소와 인증 서버의 *ONT ID,* 주소가 담긴 상호 등록 내역에 서명한다.

2) 자원 요청자는 상호 등록 내역을 사용해 인증 서버와 핸드셰이크*(handshake)*하고, 두 관계자는 디지털 서명을 검증한다.

3) 자원 요청자는 인증 서버의 접근 주소를 기록한다.

4) 인증 서버는 자원 요청자가 제공한 자원 *ID*를 기록한다.

6.2.4. 접근 제어 전략(Access Control Strategy)

사용자는 요청자가 특정 자원에 대한 접근하는 것을 제한하는 유연한 접근 권한 전략을 설정할 수 있다. 접근 제어 전략은 $A \vee (B \wedge C)$과 같은 불 연산식으로 설명할 수 있다. 인증을 요청할 때, 요청자는 전략을 충족하는 검증가능한 내역에 있는 속성에 대한 증명을 제공해야만 한다.

6.2.5. 인증 증명서(Authorization Certificate)

인증 요청을 받으면, 인증 서버는 소유자에게 접근 제어를 할 것임을 알려야한다. 소유자는 접근 제어 전략에 충족하는 요청자들에게 증명서를 발급한다. 인증서의 유효기간 내에, 요청자는 추가적인 인증 요청 없이 데이터에 반복적으로 접근할 수 있다.

6.2.6. 권한 관리 위임(Delegated Authorization)

인증 서버는 사용자가 자신의 접근 제어 전략을 설정할 수 있게 함으로써 사용자들의 인증 대리자의 역할을 할 수 있다. 이를 통해 인증 서버는 사용자와의 소통 없이 인증 요청을 처리하고 인증서를 발급할 수 있다. 인증서가 유효함을 증명하기 위해, 사용자는 인증 서버에게 위임 내역을 발급해야 한다.

6.3. 분산 데이터 교환 프로토콜(DISTRIBUTED DATA EXCHANGE PROTOCOL)

중앙화 데이터 교환의 단점으로는 데이터 캐싱, 사용자의 허가 없이 데이터를 사용하는 것, 데이터 저작권 보호 미흡 등이 있다. 온톨로지는 엔티티 간의 데이터 트랜잭션을 위해 일련의 프로토콜 명세를 정의하는 분산 데이터 교환 프로토콜(Distributed Data Exchange Protocol, DDEP)을 제안한다.

트랜잭션과 관련한 두 관계자의 이익을 위해, 트랜잭션 합의 과정에 보증인(guarantor)의 역할을 하는 중개인이 존재한다. 중개인의 역할은 안전하고 원할한 트랜잭션 조정 과정을 보장하는 것이다. 중개인은 구매자의 자금을 보호하고, 판매자나 구매자에게 자금을 전송하는 것에 대한 책임을 진다. 중간자가 트랜잭션의 최종 조정에 대한 책임을 지기 때문에, 공정하고 안전하다. 프로토콜이 중간자의 역할을 적절히 수행할 수 있도록, 분산 원장의 퍼블릭, 탈중앙 관리 기능들을 통해 구현되었다.

6.3.1. 역할(Roles)

분산 데이터 교환 프로토콜에서의 주요 역할은 다음과 같다.

- 데이터 요청자*(Data requester):* 데이터를 사려는 에이전시/비즈니스/개인
- 데이터 제공자*(Data provider):* 가공된 혹은 가공되지 않은 데이터를 팔려는 에이전시/비즈니스/개인. 데이터는 현지 정부와 규제를 만족해야 한다.
- 사용자 에이전트*(User Agent):* 데이터 트랜잭션을 위한 사용자 권한 관리 요구사항을 충족시키기 위해 사용자와 소통하는 역할을 한다. 사용자 에이전트는 다분화될 수 있지만, 어플리케이션 프로토콜 프레임워크의 사용자 라이센스 프로토콜에 정의된대로 구현해야만 한다.
- 데이터 소유자*(Data owner):* 데이터의 소유자이며, 기관, 비즈니스, 개인이 될 수 있다.

6.3.2. 사용자 권한 관리(User Authorization)

데이터 교환 구조에서 트랜잭션 데이터는 소유자의 권한 위임이 필요하다. 따라서 권한 관리 절차는 섹션 6.2에 설명된 사용자 권한 관리 프로토콜을 완전히 준수한다.

6.3.3. 안전 트랜잭션 프로토콜(Secure Transaction)

스마트 컨트랙트를 통한 안전 트랜잭션 프로토콜은 거래 활동을 위해 중앙화된 써드 파티의 보증 서비스를 제공하고, 요청자와 제공자 모두의 이익을 보호하는 안전하고 원활한 트랜잭션 과정을 제공한다.

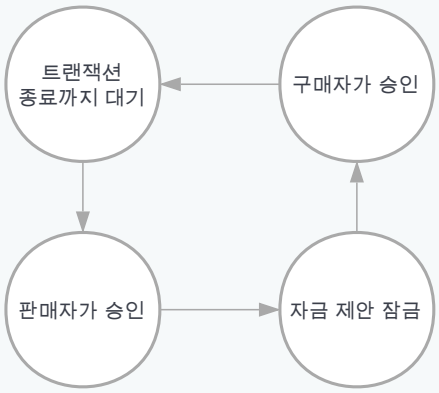


그림 6.4: 보증인 거래 과정(Guarantor Transaction Process)

프토토콜의 구현 과정은 아래와 같다.

1) 데이터 제공자가 대기중인 주문에 진입하고, 자원 *ID*, 데이터의 특징, 제공자의 계정, 가격 및 다른 특징들을 포함하는 제품 정보 작성한다. 컨트랙트는 요청자가 트랜잭션을 시작하길 기다린다.

2) 데이터 요청자는 컨트랙트에 특정한 금액을 전송하고, 컨트랙트는 전송된 금액이 판매 요구사항에 부합하는지를 확인한다.

   a) 부합하면, 컨트랙트는 자금 잠금 상태에 들어간다.

   b) 부합하지 않으면 송신자에게 에러 메시지를 반환하고, 트랜잭션은 이전 상태로 돌아간다.

3) 제공자가 데이터를 제공한 후, 컨트랙트는 승인 및 만료 기간 설정을 한다. 유효 기간이 끝날때까지 어떠한 활동도 이루어지지 않을 경우, 컨트랙트는 자동으로 조정 과정(과정 5)으로 진입한다.

4) 데이터를 수신한 후, 요청자는 컨트랙트를 승인한다.

5) 컨트랙트는 트랜잭션을 완료하기 위해 제공자의 계정에 자금을 전송하고, 다음 트랜잭션을 기다린다.

6.3.4. 데이터 교환 과정(Data Exchange Process)

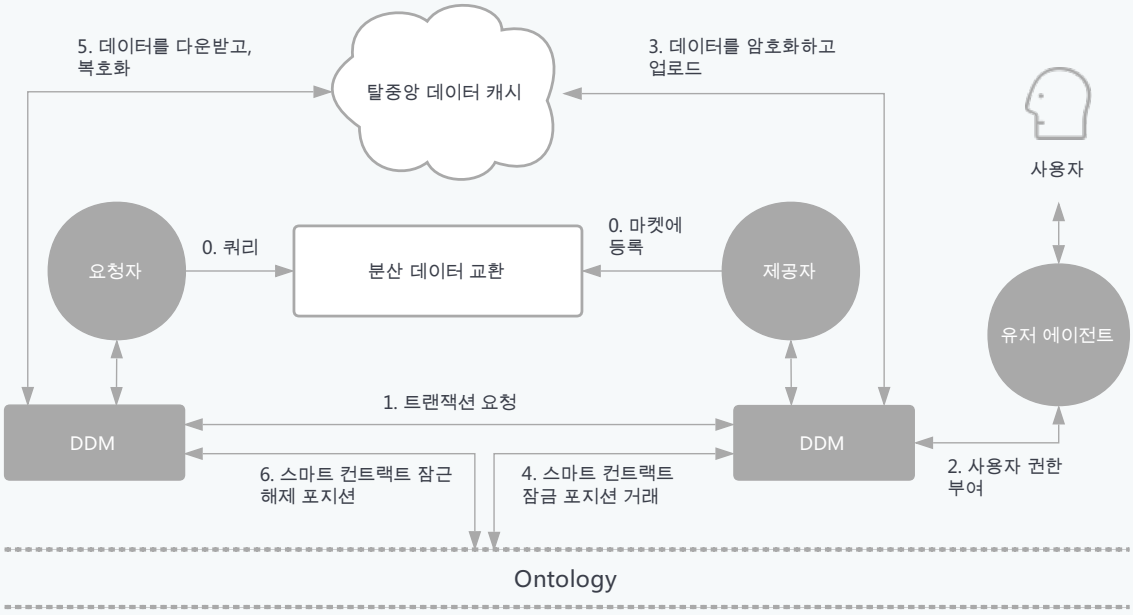


그림 6.5: 데이터 교환 과정

사전주문: 트랜잭션 준비(Pre-order: transaction preparation)

데이터 제품 공개(Data product release): 데이터 제공자는 자신의 데이터 제품 정보를 마켓플레이스에 출판한다. 요청자는 마켓플레이스를 열람하여 데이터 프로덕트(data product)를 검색하고, 구매할 데이터를 고를 수 있다. 메타데이터는 데이터 자원 소개, 키워드, 데이터 자원 해시, 컨트랙트 지불 주소 등의 정보를 포함해야한다.

상호 등록(mutual registration): 데이터 제품이 데이터 소유자의 인증을 요구할 경우, 데이터 제공자는 데이터를 풀기 전에 사용자가 지정한 유저 에이전트(uerger agent)를 통해 상호 등록을 완료해야한다. 상호 등록에 대한 자세한 내용은 위에 서술되어 있다.

1) 트랜잭션 요청(Transaction request)

구매하고 싶은 데이터를 발견하면, 요청자는 온톨로지를 통해 제공자의 신원을 검증한다. 제공자에 대한 더 자세한 정보가 필요할 경우, 다중 출처 인증 프로토콜을 참고한다. 트랜잭션 요청이 개시되기 전, 요청자는 컨트랙트 주소에 자금을 예치하고, 제공자에게 구매 데이터 요청을 보낸 후, 사용자 인증에 필요한 정보를 첨부한다. 요청은 트랜잭션 정보와 ONT ID 정보를 포함한다.

2) 권한 관리(Authorization)

요청을 받은 후, 데이터 제공자는 유저 에이전트에 접근 하여 인증 요청을 개시한다. 이 시점에서 유저 에이전트는 요구사항에 따라 온톨로지를 통해 요청자의 신원을 인증할 수 있고, 소유자가 제공한 접근 제어 정책에 따라 인증을 미리 수행할 수 있다. 소유자가 접근 제어 정책을 설정하지 않을 시, 유저 에이전트는 소유자에게 인증을 요구할 수 있다. 인증 요청이 거절될 경우, 트랜잭션은 종료되어야 한다.

3) 데이터 업로드(Uploading data)

데이터 제공자는 요청자가 제공한 대칭 키 알고리즘(symmetric-key algorithm)에 따라 일회성 세션 키를 생성하고, 이를 사용해 트랜잭션의 데이터와 데이터의 속성을 암호화한다. 그리고 암호문을 중간 저장 시스템에 전송한다.

4) 포지션 고정(Locking position)

데이터 제공자는 요청자가 예치한 자금을 확인하기 위해 스마트 컨트랙트를 호출한다. 만약 금액이 정확하면, 트랜잭션이 완료되거나 취소될때까지 포지션이 고정된다. 그 동안, 제공자는 요청자의 공개 키를 사용해 세션 키를 암호화하고, 안전한 경로를 통해 암호화된 세션 키를 요청자에게 보낸다.

5) 데이터 수신(Receiving data)

스마트 컨트랙트 이벤트(섹션 5.3에서 언급된 이벤트를 말한다)의 알림을 수신한 후, 요청자는 중간 스토리지로부터 암호문을 받고, 세션 키를 사용해 암호문을 해독한다. 해독한 평문을 계산하고 검증한 후, 성공적으로 검증을 완료하면 단계 6)으로 이동한다.

6) 트랜잭션 승인(Transaction confirmation)

데이터 거래 컨트랙트가 완료되면, 컨트랙트 자금은 데이터 제공자의 계정으로 전송된다.

예외처리 메커니즘: 다양한 비즈니스 시나리오에 따라 예외처리 메커니즘을 개별적으로 구성할 수 있다. 예를 들어, 요청자가 특정한 기간 동안 데이터를 승인하지 않는다면, 제공자는 컨트랙트가 자금을 열게 만들거나, 스마트 컨트랙트가 자동으로 자금을 열 수 있다.

6.3.5. 개인정보보호(Privacy Protection)

간혹, 참여자들은 자신의 트랜잭션을 감추고 싶을 수 있다. 이 경우, 수신자의 토큰 주소와 ONT ID를 엔티티와 연결지을 수 없게 스텔스 주소 기술(stealth address technique)을 사용할 수 있다. 요청자는 데이터 제공자만이 자신의 토큰 주소를 사용해 개인키를 복구할 수 있는 스텔스 주소를 생성한다.

데이터 제공자의 토큰 주소가 $S = s \cdot G$라고 가정하자. 요청자는 난수 r을 생성하고, $R = r \cdot G$를 계산한다. 토큰 트랜잭션의 주소로 사용될 스텔스 주소는 $E = Hash(r \cdot S) \cdot G + S$의 식으로 계산할 수 있다. 데이터 제공자만이 자신의 개인 키 s를 사용해 개인 키, $e = Hash(s \cdot R) + s$를 계산할 수 있다.

7. 온톨로지 어플리케이션 프레임워크 (ONTOLOGY APPLICATION FRAMEWORK)

7.1. 어플리케이션 프레임워크 모델(APPLICATION FRAMEWORK MODEL)

개발자들이 복잡한 분산 원장에 대한 이해 없이도 빠르고 쉽게 dApp 개발을 할 수 있게 프로토콜과 모듈을 모은 것이다. 온톨로지의 어플리케이션 프레임워크는 높은 수준의 확장성을 가지고 있으며, 시나리오의 요구사항에 따라 개선될 수 있다.

그림 7.1은 dApp이 어떻게 어플리케이션 프레임워크를 통해 온톨로지와 교류를 하며 탈중앙화된 신뢰를 형성하는지를 설명한다.

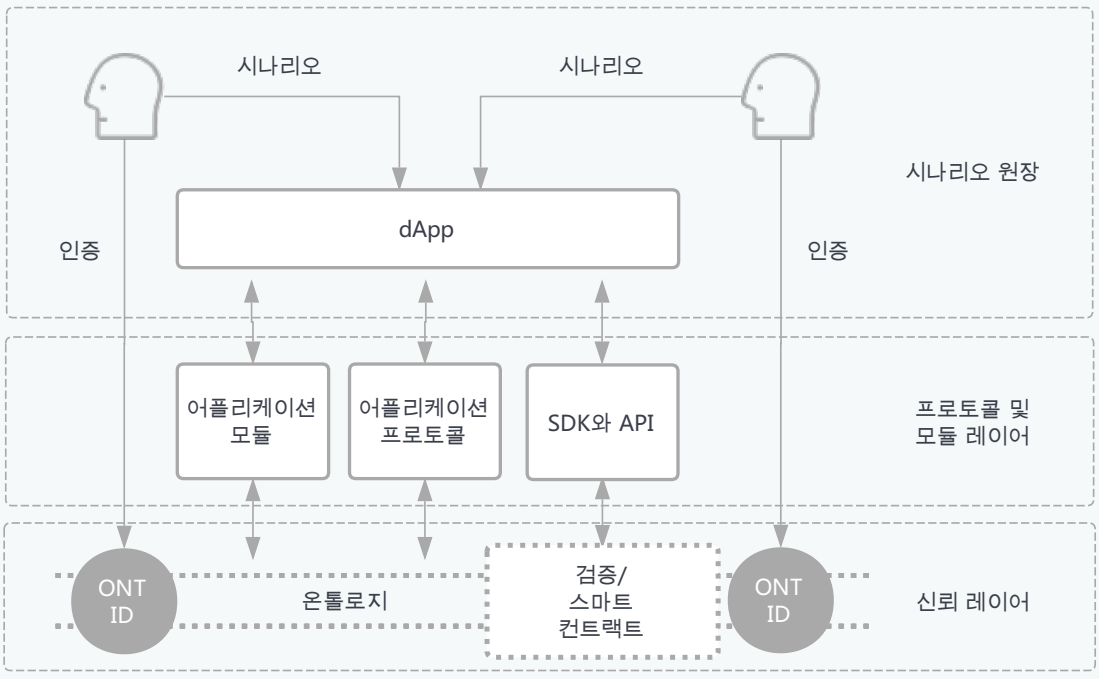


Figure 7.1: 어플리케이션 프레임워크 모델(Application Framework Model)

- 신뢰 레이어*(Trust Layer):* 온톨로지가 신원 인증, 스마트 컨트랙트 시스템 등을 통해 신뢰 시스템을 책임지는 영역이다.
- 모듈/프로토콜 레이어*(Module and protocol layer):* 상위 레이어의 시나리오들이 어플리케이션 모듈, 어플리케이션 프로토콜, *SDK, API*를 통해 온톨로지를 더 잘 사용할 수 있게 하는 영역이다.
- 어플리케이션 레이어*(Application layer):* 다양한 시나리오에 대한 *dApp*은 서비스 개발에만 집중하면 되며, 신뢰 문제는 온톨로지가 해결한다.

7.2. 데이터 마켓플레이스(DATA MARKETPLACE)

디지털화될 미래의 세상에서 데이터 교환은 데이터에 대한 소유권의 전달에 그치지 않을 것이다. 데이터에 대한 연산 역시 중요한 협력 모델이 될 것이다. 데이터 시장에서의 제품은 데이터, 데이터 예측(data forecasting), 데이터 연산 자원(data computing resources) 등을 포괄한다. 참여자로는 서로가 협업하여 생태계를 형성하는 데이터의 소비자, 공급자, 딥러닝이나 AI 기술을 통해 빅데이터 분석을 하는 서비스 제공자 등이 있다. 복잡한 데이터 협력 생태계에는 서로를 연결해주는 기반이 필요하다.

온톨로지의 분산 데이터 교환 프로토콜(Distributed Data Exchange Protocol, DDEP)과 데이터 딜러 모듈(Data Dealer Module, DDM), 일련의 암호학 모듈은 글로벌 수준의 거래량과 요구사항을 충족시키기 위해 완벽한 분산 데이터 트랜잭션과 협력 프레임워크를 구성한다. 온톨로지는 글로벌 사용자를 위한 다양한 유형의 dApp 어플리케이션을 제공한다. 이를 토대로, 데이터 거래 서비스 제공자는 모든 산업에서 다양한 범주의 데이터 거래 시장을 형성할 수 있다.

7.3. 데이터 중개 모듈(DATA DEALER MODULE)

분산 데이터 거래 프로토콜(distributed exchange protocol)에 기반한 데이터 중개 모듈(Data Dealer Module, DDM)은 온톨로지에서 중요한 어플리케이션 모듈이다. dApps 개발자와 데이터 거래 참가자는 DDM 통해 데이터 트랜잭션 어플리케이션을 신속하게 구현할 수 있다. DDM은 RESTful API, RPC, SDK 등으로 구성되며, 다양한 종류의 프로토콜을 지원한다.

DDM의 구성 요소로는 데이터 트랜잭션 서버, 단일 사용자 클라이언트, 다중 사용자 클라이언트, 경량 클라이언트가 있다. 구성 요소의 주요 기능은 ONT ID 관리, 데이터 자원 관리, 스마트 계약 트랜잭션, P2P 통신 네 가지가 있다.

DDM 데이터 트랜잭션 모듈은 다음의 장점과 기능을 가진다.

- 접근 제어 모델과의 분리*:* 모듈은 데이터 자원의 연결과 데이터 자원 소유자의 제어 서버만을 관리한다. 접근 권한에 대한 설정과 검증에는 참여하지 않는다. 이는 데이터 소유자의 개인정보를 보호할 뿐 아니라, 불필요한 분쟁을 피함으로써 데이터 제공자의 신용을 높여주기도 한다.
- 데이터 프라이버시 보호*:* 실제 데이터를 저장하지 않으며, 거래 데이터를 암호화한다.

- 데이터 검색 외에도, 데이터 요청자는 데이터 제공자에게 자신의 주문을 브로드캐스트할 수 있다.
- "단일 모듈 단일 기능"의 원칙으로 설계되어, 암호 기반의 보안 모듈과 사용자 권한 인증 모델을 사용하는 유연한 시나리오를 지원할 수 있다.

7.4. 암호 기술과 보안 모듈(CRYPTOGRAPHY AND SECURITY MODULES)

7.4.1. 안전한 다자간 계산(Secure Multi-party Computation)

전형적인 협력 연산 시나리오에서, 여러 참여자들은 사적인 데이터를 제3자에게 샤딩하고 않으면서 연산을 수행하고 싶을 수 있다. 현재의 접근은 이와 같은 시나리오에서 작동하지 않는다. 예를 들어, 어플리케이션 개발 회사 A가 강력한 머신 러닝 알고리즘을 자신의 어플리케이션에 통합하고 싶을 때, 현재의 시스템은 자신의 데이터는 머신러닝 알고리즘 제공회사 P에 그대로 넘길 것이다. 그러면 P는 받은 데이터로 알고리즘을 돌리고, A에게 결과를 반환할 것이다. P는 A의 데이터와 관련한 모든 것을 알게 될 것이다.

최초의 MPC 프로토콜은 앤드류 야오(Andrew Yao)가 야오의 백만장자 문제(Yao's Millionaires' Problem)를 풀기 위해 정리한 프로토콜이다. 야오의 백만장자 문제는 두 백만장자가 각자의 실제 재산을 서로 알려주지 않은 채로 누가 더 부자인지를 알고싶어 할 때, 이걸 어떻게 해결할 수 있을지에 대한 솔루션과 증명에 대한 문제이다. 이 문제의 일반화는 다음과 같다. n명의 참여자가 각각 x_i개의 개인 데이터를 가진다고 가정할 때, 해당 개인 데이터에 대한 알려진 함수 $f(x_1, \ldots, x_n)$를 풀어야한다.

MPC를 풀기 위한 접근에는 야오의 혼란 회로(Yao's garbled circuits)[22][24]에 기반한 접근과 비밀 공유(secret sharing)[23]에 기반한 접근 두 가지 주요 접근이 존재한다. 아래 섹션에서는 후자의 접근을 다룰 것이다.

(t,n)-threshold secret sharing scheme에서 비밀은 n개의 조각으로 나뉜다. 각 참여자는 하나의 비밀 조각을 가지고, 하나의 조각은 t개의 조각을 모았을 때만 복구할 수 있다. 비밀 공유에 기반한 MPC 프로토콜에선 중간 계산 결과를 조각낸 후 참여자들에게 분배한다. 마지막에 각 참여자들은 전달받은 중간 계산 결과를 모아서 $f(x_1, \ldots, x_n)$을 재구성한다.

7.4.2. 완전 동형 암호화(Fully Homomorphic Encryption)

일반적인 데이터 교환 시나리오에서 데이터 제공자는 요청자에게 데이터 전체를 그대로 전송하기보다는 데이터에 접근할 수 있는 권한을 주기를 원한다. 기업들에게 데이터에 대한 접근을 제한적으로 허용하면서, 데이터에 대한 보호를 유지하는 것은 매우 중요하다. 완전 동형 암호화 기술은 이에 대해 유망한 해결책이 될 수 있다[25][26][27][28]. 예를 들어, 기업은 자신의 공개 키를 사용해 데이터 m을 암호화하고 암호문에 복잡한 계산 f를 수행할 다른 관계자에게 암호문을 보낸다. 이 새 암호문은 암호화된 $f(m)$이다. 마지막으로 기업은 새 암호문을 받아 개인 키를 사용해 복호화한 후, $f(m)$을 구할 수 있다.

완전 동형 암호화 방법은 다른 공개 키 암호화 방법에도 사용되는 키 생성, 암호화, 복호화, 동형 덧셈($CAdd$), 동형 곱셈($CMul$)으로 이루어졌다. C_1과 C_2는 각각 암호문이며, M_1과 M_2는 각 암호문을 만들 때 사용한 평문이라 가정한다.

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

많은 복잡한 작업들은 위의 두 기본적인 연산으로 구성된 산술 회로로 구성할 수 있다.

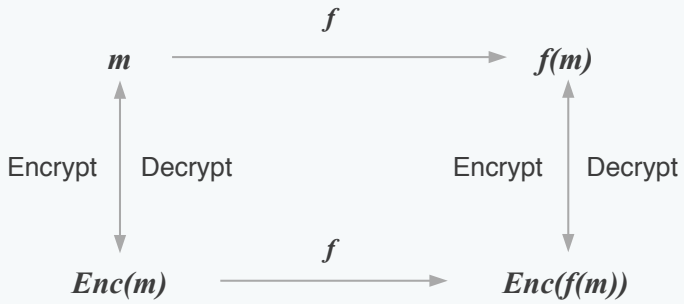


그림 7.2: 평문에 대한 동형 암호화(Homomorphic Encryption of Plain Text)

그러므로, FHE 방법을 사용해 암호문에 동형 작업f를 실행할 수 있고, 복호화 알고리즘을 사용해$f(m)$을 얻을 수 있다. 이는 바로 위 그림에 묘사되어 있다.

BGV나 FV와 같은 다양한 FHE 방법 또한 지원할 계획이다.

7.4.3. 디지털 저작권(Digital Copyright)

데이터의 특성에 따라, 온톨로지는 데이터 저장과 라이프 사이클 관리 기능을 제공한다. 첫째, 온톨로지는 라이프 사이클 추적 메커니즘을 설계했다. 온톨로지는 전체적인 등록, 요청, 권한 관리, 트랜잭션 과정을 추적하기 위해 각각의 데이터 조각에 대한 디지털 신원을 형성한다. 둘째, 온톨로지는 데이터의 저작권과 트랜잭션 정보를 분산 원장에 기록한다.

온톨로지는 디지털 워터마킹 기술을 사용해 디지털 캐리어에 식별 정보를 직접 각인한다. 이로 인해, 데이터의 가치에 영향을 주지 않으면서 공격자의 신원을 쉽게 파악할 수 있으며, 공격자가 데이터를 수정하는 것을 방지할 수 있다. 요청자는 캐리어의 숨겨진 정보를 통해, 콘텐츠 제작자를 쉽게 확인할 수 있으며 캐리어가 손상을 입었는지 식별할 수 있다. 디지털 워터마킹은 위조 방지 추적, 저작권 보호, 신원 은신, 인증, 안전한 통신 등을 구현하는데 효과적인 방법이다.

온톨로지가 제공하는 디지털 워터마킹의 주요 기능은 아래와 같다.

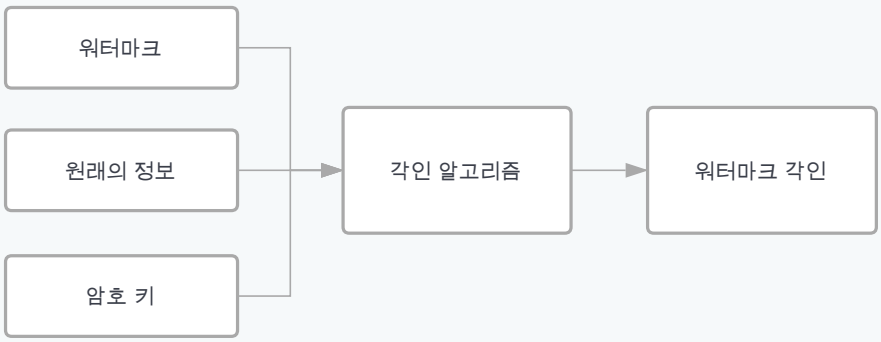
- 입증*(Vindicability):* 워터마크는 저작권의 보호를 받는 제품에 대해 완전하고 신뢰할 수 있는 증거를 제공한다. 사용자는 워터마크 알고리즘을 사용해 보호받는 대상*(*사용자 번호, 제품 로고,

문자 등)의 소유자의 정보를 식별할 수 있으며, 원할 때 추출할 수 있다. 워터마크는 대상이 보호받는지 확인하고, 데이터의 확산을 감시하고, 인증을 수행하고, 불법 복제를 저지하는데 사용할 수 있다.

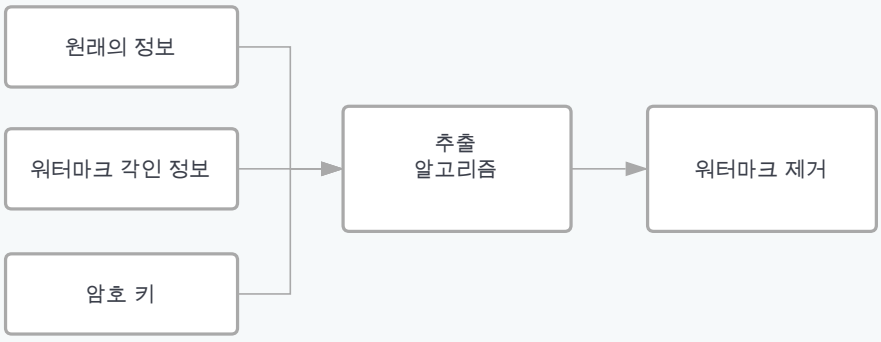
- 감지불가능함*(Imperceptibility):* 각인된 워터마크는 보이지 않는다. 이상적인 상황은 거의 모든 워터마크 알고리즘이 그렇듯 워터마크가 추가된 이미지가 원본 이미지와 시각적으로 일치하는 것이다.
- 견고성*(Robustness):* 디지털 워터마크는 다양한 의도된, 의도되지 않은 신호처리를 거친 후에도 무결성을 유지하고, 정확하게 식별된다. 견고성은 워터마크에게 중요한 특징이다. 워터마크는 의도되거나*(*악의적 공격 등*)* 의도되지 않은*(*이미지 압축, 필터링, 스캐닝, 복제, 노이즈 오염, 차원 변화 등*)* 가지각색의 물리적, 기하적 변조를 견딜 수 있어야 한다. 견고한 워터마킹 알고리즘은 워터마크가 각인된 이미지로부터 각인된 워터마크를 추출할 수 있어야 하며, 워터마크의 존재를 증명할 수 있어야 한다. 만약 특정 워터마크 각인과 감지 알고리즘이 사라진다면, 데이터 제품에 대한 저작권 보호가 어려워진다. 만약 공격자가 워터마크를 지우기로 한다면, 멀티미디어 제품은 이미지를 파기한다.

그림 7.3과 같이, 온톨로지의 데이터 워터마킹 기능은 워터마크 각인과 워터마크 제거 두 가지 모듈을 포함한다.

디지털 워터마킹은 각종 문서, 영수증, 디지털 신원 등을 위한 보안 시스템으로 사용될 것이다. 디지털 워터마킹은 문서의 신빙성을 증명하고, 불법 복제로부터 문서를 보호하는데 사용할 수 있다.



(a) 데이터 워터마크 각인 과정(Data Watermark Embedding Process)



(b) 데이터 워터 마크 제거 과정(Data Watermark Removal Process)

그림 7.3: 데이터 워터마크 모듈(Data Watermark Module)

7.5. 사용자 권한 관리 제어기(USER AUTHORIZATION CONTROLLER)

사용자 권한 관리 제어기(user authorization control, UAC) 모듈은 사용자 권한 관리 프로토콜(user authorization protocol)에 기반하며, 데이터 소유자의 자신의 데이터에 대한 권한 관리를 돕는다. 데이터와 관련한 모든 트랜잭션은 데이터 소유자가 데이터 트랜잭션 권한 관리를 수행할 수 있도록 보고된다. UAC 모듈은 관계형 데이터베이스를 지원하며, 외부 사용을 위해 RESTful API를 제공한다. UAC는 아래의 두 기능을 제공한다.

- 사용자 데이터 권한 관리 접근 정책 설정*(User Data Authorization Access Policy Settings).*
- 사용자 데이터 권한 관리 접근 제어*(User Data Authorization Access Control).*

7.5.1. 권한 관리 정책 설정(Authorization Policy Setting)

사용자는 UAC모듈이 제공하는 RESTful API를 사용해 자원 제공자가 자신의 데이터를 등록하도록 요청할 수 있다. 등록 후, 사용자는 UAC 모듈을 사용해 등록된 데이터를 검색하고 검토할 수 있으며, 데이터 접근과 제어 정책 설정을 수행할 수 있다. 자주 접근하며 보안에 대한 요구사항이 낮은 데이터에 대해 사용자는 UAC 모듈의 인증 호스팅 설정을 할 수 있다.

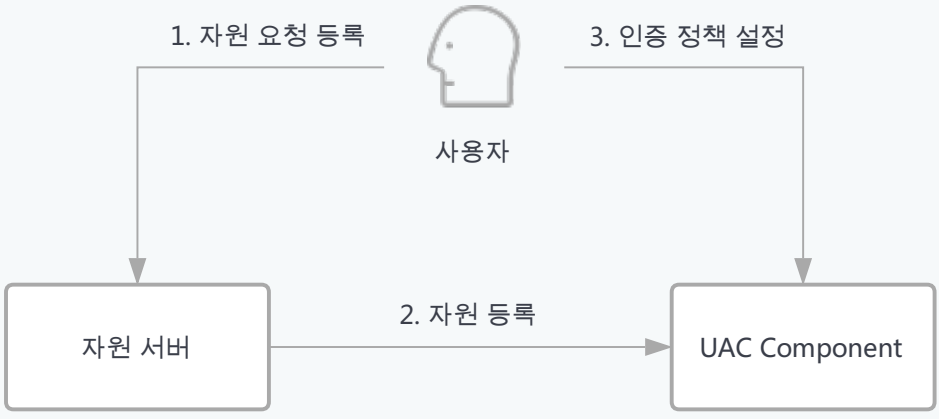


그림 7.4: 인증 접근 정책 설정(Setting Authorization Access Policy)

7.5.2. 접근 제어(Access Control)

호스트 모드가 아닐 때, UAC 모듈은 사용자 데이터를 보호하는 역할을 하며, 사용자가 요청자에게 데이터를 전송하도록 요구한다. 호스트 모드일 때, UAC 모듈은 사용자의 이익을 위한 권한 관리 결정을 하고, 사용자에게 권한 관리 영수증을 제공한다. 두 형식은 모두 사용자에게 누가 참여하는지, 언제 트랜잭션이 일어나는지 어떤 데이터가 전송되는지와 같은 데이터 트랜잭션의 세부 항목을 알려준다.

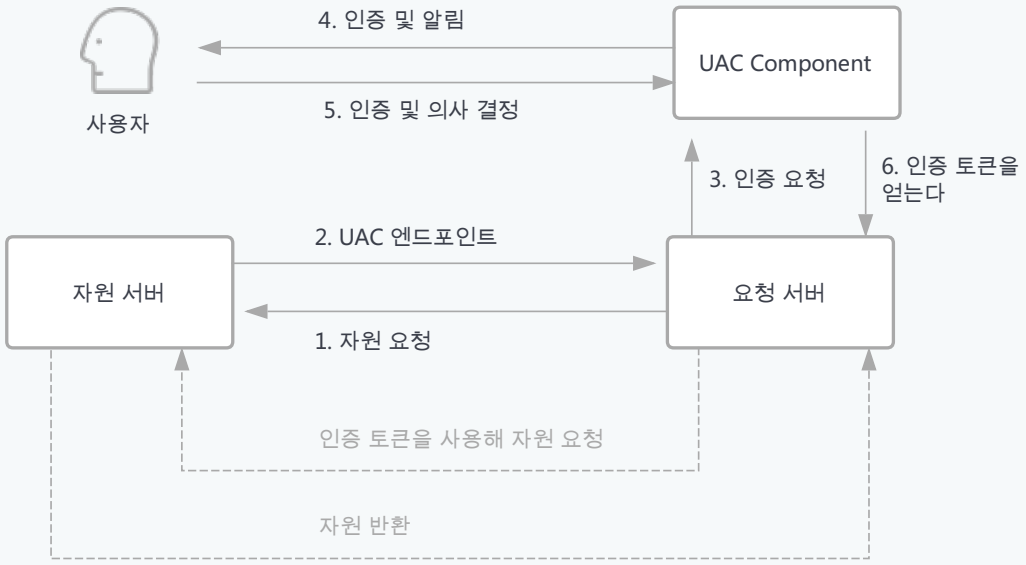


그림 7.5: 권한 관리 접근 제어(Authorization Access Control)

7.6. 내역 관리 모듈(CLAIM MANAGEMENT MODULE)

검증가능한 내역 관리 모듈은 분산 신뢰 시스템의 요구사항에 맞춰 개발되었으며 온톨로지의 신뢰 중재자와 관련되어 있는 중요하고 기본적인 모듈이다. 이 모듈은 관계형 데이터베이스(relational database) 사용을 지원하기 위해 RESTful API를 제공한다. 주요 기능은 발급, 검증, 쿼리, 신뢰 내역 취소를 포함한다.

7.7. GLOBALDB

GlobalDB는 장착형 분산 key-value 데이터베이스이다. GlobalDB의 기반 계층은 Google's Spanner/F1에 기반한 오픈소스 분산 NewSQL 데이터베이스인 TiBD이다.

GlobalDB는 블록체인/분산 원장과 IPFS에 최적화된 데이터베이스이다. GlobalDB는 SQL와의 호환, 스토리지 샤딩, 분산 트랜잭션, 수평 확장성, 에러 회복 능력을 제공한다. 블록체인과 빅데이터, 블록체인과 인공지능의 연계 어플리케이션에 응용할 수 있다.

GlobalDB는 분산 트랜잭션, 스토리지 샤딩, 로드 밸런싱, SQL on KV의 네 가지 주요 기능을 가진다.

7.7.1. 분산 트랜잭션(Distributed Transactions)

GlobalDB는 스테이트 샤딩(state sharding)과 오프체인(off-chain) 서비스를 지원하는 완전한 분산 원장을 제공한다. 트랜잭션 모델은 Google Percolator[29]를 기반으로 최적화되었다.

GlobalDB의 트랜잭션 모델은 낙관적 잠금(optimistic locking)을 차용한다. 분산 트랜잭션은 보고된 충돌만을 감지한다. 만약 전통적인 방법에서 충돌이 발생하면 다시 시도해야 한다. 이 모델은 심각한 충돌이 발생한 상황에선 비효율적이지만 대부분의 시나리오에 대해선 매우 효율적이다.

분산 트랜잭션은 두 단계의 커밋으로 수행되며, 데이터는 일관성 복제(consistent replication)를 해야만 하기 때문에 트랜잭션이 너무 크면 커밋 프로세스가 느려질 수 있다. 이를 피하기 위해 온톨로지는 다음의 제한을 두었다.

1) 하나의 *KV entry*는 *6MB*를 초과하지 않는다.

2) *KV entry*의 개수는 *30W*를 초과하지 않는다.

3) *KV entry*의 전체 크기는 *100MB*를 초과하지 않는다.

7.7.2. 스토리지 샤딩(Storage Sharding)

GlobalDB는 key의 범위에 따라 데이터를 자동으로 샤딩한다. 각 조각은 $[StartKey, EndKey)$의 간격을 유지한다. key-value 프래그먼트의 총 수가 특정 임계점을 초과하면, 자동 분할이 일어난다.

7.7.3. 로드 밸런싱(Load Balancing)

로드 밸런서(PD)는 스토리지 클러스터의 상태에 따라 클러스터에 전달되는 로드를 스케쥴링한다. 스케쥴링은 프레크멘테이션과 PD의 스케쥴링 로직에 의해 구성된 정책에 따라 이뤄진다. 전체적인 과정은 자동으로 완료된다.

7.7.4. SQL on KV

GlobalDB는 SQL구조를 KV구조로 자동으로 매핑한다. 다시 말해, GlobalDB는 다음의 두 가지 작업을 수행한다:

- 한 행의 데이터는 하나의 *KV*에 매핑된다. *TableID*의 접두사로 *TableID*가, 접미어로는 *line ID*이 쓰인다.
- 인덱스는 *KV*에 매핑된다. *key*는 *TableID+IndexID*를 접두사로, *index* 값으로 접미어를 구성한다.

한 테이블 상의 데이터나 인덱스는 동일한 접두어를 가지므로 TiKV의 key space 또한 key-value와 근접한 값을 가질 것이다. GlobalDB는 데이터 가독성을 더 높이기 위해 해당 더티 데이터 관리 전략(dirty data management strategy)을 구성할 것이다.

GlobalDB는 고도로 구성 가능하고 여러 상황에 적용할 수 있으며, 실시간 고성능 비즈니스에서도 온체인 및 오프체인을 오가며 사용할 수 있다. 이는 온톨로지의 핵심 구성 요소로서 기본 분산 원장을 위한 견고한 토대를 제공 할 것이다.[30]

8. 끝내는 말(POSTSCRIPT)

해당 백서는 온톨로지와 관련한 기술에 대한 개관을 담고 있습니다. 관련 기술이 발전함에 따라, 온톨로지 팀은 지속해서 이 백서의 내용을 갱신할 예정입니다.

온톨로지는 창조적이고, 열려있으며, 협력하는 기술 생태계를 만들 것입니다. 우리는 온톨로지에 참여하여 함께 온톨로지의 기술을 발전시켜나갈 전 세계의 개발자들을 환영합니다.

참고 문헌(REFERENCE)

[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/.

연락처(CONTACT US)

 Email: contact@ont.io

 Telegram: OntologyNetwork

 Twitter: OntologyNetwork

 Facebook: ONTnetwork

 Reddit: OntologyNetwork

 Discord：https://discord.gg/vKRdcct

 Medium: OntologyNetwork

 LinkedIn: Ontology Network