최종연구보고서	KISA-WP-2024-0012

분산ID(DID) 직접 해석방안 연구

Research on Direct Resolution of Distributed ID(DID) Documents

수탁기관 : 전남대학교 산학협력단

2024. 11.

KISA 한국인터넷진흥원

제 출 문

한국인터넷진흥원 원장 귀하

본 보고서를 “분산ID(DID) 직접 해석방안 연구”의 최종 결과 보고서로 제출합니다.

2024년 11월 29일

수탁 기관 : 전남대학교 산학협력단
연구책임자 : 교 수 김경백 (전남대학교 인공지능융합학과)
참여연구원 : 교 수 박태준 (전남대학교 인공지능융합학과)
연 구 원 진현석 (전남대학교 인공지능융합학과)
연 구 원 임도현 (전남대학교 소프트웨어공학과)
연 구 원 김건민 (전남대학교 소프트웨어공학과)
연 구 원 조세빈 (전남대학교 인공지능학부)

요 약 문

1. 제목

2. 연구개발의 목적 및 중요성

- ㅇ **(목적)** DID의 상호운용성을 보장하고, 확장성을 높일 수 있도록 특정 해석기에 대한 의존성 없이 DID 문서를 조회할 수 있는 직접 해석 방안 마련 및 해석모델 설계를 수행하고 이를 기반으로 표준문서 초안을 마련한다
- ㅇ **(중요성)** 최근 DID는 혁신적인 탈중앙 구조와 신뢰성 있는 검증 여부로 인해 국내·외에서 많은 관심을 받고 있으며, 보다 안전하고 투명한 신원 관리 및 증명이 가능하다. 지속적으로 새로운 DID Method들이 발표되고 있으나, 동작 방식, 신뢰 기반 및 필요 인프라가 모두 상이하여 해석에 어려움이 존재하고 확장성이 제한된다는 한계점이 있다. 따라서 특정 해석기에 대한 의존성 없이 다양한 DID Method들의 DID 문서를 조회할 수 있는 기술방안에 대한 연구를 수행할 필요가 있으며, 지속적인 상호운용성을 보장하기 위해 표준화를 진행할 필요가 있다

3. 연구개발의 내용 및 범위

- ㅇ **(기술조사)** DID 식별자, 방법론 등 최신 동향 조사 및 해석기 없이 다른 DID 식별자를 클라이언트에 입력하여 DID 문서를 조회·검증할 수 있는 기술방안 도출
- ㅇ **(모델설계)** DID 직접 연계 방안의 가능성 검토를 위한 참여자별 동작절차, 기능 요구사항 도출 및 실효성 있는 해석모델 설계

○ **(표준지원)** 표준문서, 워킹그룹 현황 등 최신 표준화 동향 조사 및 분석, 국제 인터넷 표준화 기구(W3C, IETF) 인터넷 표준 초안 제출을 위한 표준문서 초안 마련

4. 연구결과

4-1. 국내 · 외 현황

○ DID는 Uniform Resource Identifier (URI) 구조를 갖는 새로운 유형의 식별자로, 탈중앙 구조 등 특유의 분산성과 신뢰성 있는 검증 여부로 인해 보다 안전하고 투명한 신원 관리 및 증명이 가능하여 다양한 산업 분야에서 적용하려는 시도가 이뤄지고 있다

○ 여러 국제단체에서 표준화를 진행하고 있으며, 국내에서도 과학기술정보통신부에서 DID 통합 해석기 제작 발표 및 상호호환/인증 논의를 본격화하고, 공공플랫폼 인증 방법에 적용하는 등 관심도가 높아지고 있다

4-2. DID 기술 개요

○ **(핵심 규격)** URI의 한 형태로, DID 주체를 인증 정보 등이 포함된 DID 문서와 연결하여 해당 주체와 신뢰할 수 있는 상호작용을 가능하게 한다

[표 1] 주요 DID 구성 요소

주요 DID 구성 요소	내용
DID 및 DID URL	· DID: 특정 Method를 따르는 URI 식별자의 한 형태 · DID URL: 기본 DID 구문을 확장하고 표준 URI 구성 요소를 통합하여 특정 리소스를 찾을 수 있도록 한다
DID 주체	· DID에 의해 식별된 엔티티이며, 무엇이든 DID 주체가 될 수 있음과 더불어 자체가 Controller가 되는 것도 가능하다
DID Controller	· 하나의 DID에 2개 이상 존재할 수 있으며, Method에 정의된 대로 DID 문서를

	변경할 수 있다
검증 가능한 저장소	· DID를 기록하고 DID 문서를 생성하는데 필요한 데이터를 반환하는 모든 시스템을 일컫는다
DID 문서	· DID 관련 정보가 포함되어 있으며, 검증 방법과 DID 주체와의 상호작용 관련 서비스를 표현하는 문서
DID Methods	· 특정 유형의 DID 및 관련 DID 문서가 생성(Create), 해석(Read), 업데이트(Update), 비활성화(Deactivate)되는 CRUD 메커니즘을 일컫는다
DID Resolvers (Resolution)	· Resolver는 DID를 입력으로 받아 해당하는 DID 문서를 출력으로 반환하는 역할을 수행한다 · 관련 방법은 DID의 유형 및 Method 사양에 따라 정의된다
DID URL Dereferencers (Dereferencing)	· Dereferencer는 DID URL을 입력으로 받아 출력으로 리소스를 생성하며, 이러한 시스템 구성 요소 및 과정을 일컫는다

○ **(DID 시스템)** Controller(통제자), Relying Parties (RP, 요청 당사자), 주체(Subject) 라는 세 개의 개별 개체가 주요 작업을 수행한다

[표 2] DID 시스템 개체별 수행 작업

DID 시스템 개체	내용
Controller	· DID를 생성 및 통제하는 역할 수행한다
RP	· DID 주체와 관련된 상호작용을 위한 식별자로, DID에 의존한다
주체	· DID에서 지칭하는 개체로, 무생물은 주체가 될 수 없고 기능적 역할을 수행하지 않는다

○ **(Use Cases)** DID 시스템은 기업, 교육 및 근로, 리테일 및 소비자, 법률 등 다양한 환경에서 안전한 인증 및 사용자의 프라이버시를 보호하는 데 사용되고 있다

4-3. DID Methods

- ㅇ **(DID Methods)** DID의 고유한 작동 방식을 정의한 규격으로, 특정 DID 체계, DID와 DID 문서에 대한 CRUD 절차와 더불어 구현/보안 및 개인정보 보호 고려사항 등이 포함된다. 블록체인, 외부 저장소, 식별자를 문서로 해석하는 방안 등으로 구현이 가능하다
- ㅇ **(블록체인 기반 DID Methods)** 블록체인 네트워크를 신뢰기반으로 하는 DID Method로, 이미 구성되어 있는 블록체인 네트워크에 DID 문서를 저장함으로써 내부에 있는 정보에 대한 신뢰기반을 구축하여 신뢰성과 보안성을 높이는 방식이다. 개인 기기의 지갑을 통해 네트워크에 접속하거나 RPC URL로 간접접근, 혹은 신뢰할 수 있는 노드들을 통해 직접 접속하여 DID 문서를 확보할 수 있으므로 상대적으로 단순한 해석이 가능하고 특정 인프라에 종속되지 않아 유연성과 확장성이 뛰어나다

[표 3] 주요 블록체인 기반 DID Methods

Method	신뢰 기반	내용
did:ethr	이더리움	· 이더리움과 호환되는 모든 블록체인에 사용가능한 DID Method이다 · 소문자로 구성된 did:ethr와 더불어 이더리움 주소를 식별자로 사용한다 · 이더리움에 작성된 Smart Contract 내부 함수를 사용하여 별도의 자원 없이 DID 해석 및 DID 문서를 획득할 수 있다
did:indy	Hyperledger Indy	· W3C Credentials Community Group에서 발표한 DID 규격을 준수하는 DID Method로, Hyperledger Indy를 기본으로 하여 해당 규칙을 준수하는 모든 DID에 대하여 해석이 가능하다 · 소문자로 구성된 did:indy와 더불어 indy ledger를 명시하며, 보조 ledger에 대한 이름을 가질 수 있다. 또한 고유한 namespace-identifier를 식별자로 사용하

		며 이는 Public Key에 해당한다 · Indy의 Ledger Instance를 찾아 연결하고 NYM을 검색하여 DID 문서를 조립하는 방식으로 DID 문서를 획득한다

○ **(Web 기반 DID Methods)** Web 환경과 DNS 인프라를 통해 분산된 신원 관리 시스템을 구축하는 Method로, 도메인 네임을 기반으로 DID를 구성하여 기존 Web 및 네트워크 인프라를 활용할 수 있다

[표 4] 주요 Web 기반 DID Methods

Method	신뢰 기반	내용
did:web	미지정	· Web 도메인을 기반으로 Domain Name System (DNS)를 통해 접근 가능한 DID Method · 소문자로 구성된 did:web과 더불어 도메인 네임을 식별자로 사용하며, 식별자 내 포트 번호 및 디렉토리 경로를 포함할 수 있다 · 식별자를 HTTPS URL로 변경하고 HTTP GET 요청을 수행하여 DID 문서를 획득한다
did:webs	KERI	· did:web에 Key Event Receipt Infrastructure (KERI)를 사용하여 보안 요소를 추가한 Method로, 중앙화된 인증 기관 대신 암호화된 이벤트 체인으로 신원 확인 · 검증이 가능하다 · 소문자로 구성된 did:webs와 더불어 도메인 네임에 KERI Autonomic Identifier (AID)가 포함된 식별자를 사용하며, did:web과 호환이 가능하다 · 식별자를 HTTPS URL로 변경하고 DID 문서 및 KERI Event Stream에 HTTP

		GET 요청을 수행, KERI Event Stream을 처리하여 DID 문서 도출 및 검증이 가능하다
did:dns	did:key	· did:web을 기반으로 개선점을 제공한 DID Method로, 기존 DNS 인프라를 활용하여 DID를 게시하고 검색하기 위해 DNS 레코드 (Resource Record, RR)에 IP 대신 did:key에 대한 정보를 담고, 해당 정보를 기반으로 DID 문서를 생성한다 · 소문자로 구성된 did:dns와 더불어 도메인 네임으로 구성된 식별자를 사용한다 · 비어있는 (Resolved) 문서를 생성하고 RR 검색 및 did:key를 해석하여 문서를 구성, did:key의 식별자를 dns의 식별자 (도메인 네임)으로 대체하여 문서를 획득한다

4-4. 관련 인증 방법 및 VC/VP 시스템

- ㅇ **(OpenID)** 사용자가 다양한 Web사이트나 서비스에서 하나의 ID로 인증할 수 있는 사용자 프로토콜로, OAuth2.0 프로토콜을 기반으로 인증 기능을 추가하여 사용자의 신원 인증 기능을 제공한다
- ㅇ **(DID Comm Messaging)** DID를 기반으로 하는 탈중앙화된 통신 프로토콜로, 사용자(DID 주체) 간의 통신을 위해 사용된다. DID를 통해 서로의 신원을 검증하고 안전한 메시지 교환을 가능하게 하며, 탈중앙화 신원 시스템의 핵심 요소 역할을 하고 Verifiable Credential (VC) 및 DID 문서의 통신에 사용된다
- ㅇ **(VC/VP 시스템 개요)** 주체(Subject), 발급자(Issuer), 소유자(Holder), 검증자(Verifier), 검증 가능한 데이터 레지스트리로 구성된다

[표 5] VC/VP 시스템 구성 요소

구성 요소	내용
주체	· 클레임(주체에 대한 진술)이 제기되는 것으로, 인간, 동물, 사물이 포함될 수 있다

발급자	· 엔티티가 하나 이상의 주체에 대한 클레임을 주장할 때, 클레임에서 VC를 발급하고, VC를 소유자에게 전송하는 역할을 수행한다
소유자	· 엔티티가 하나 이상의 VC를 소유할 때, 이를 통해 VP(Verifiable Presentation)를 생성하여 검증자에게 제출하는 역할을 수행한다 · 소유자는 자격 증명 저장소에 자격 증명을 저장한다
검증자	· 하나 이상의 VP 또는 VC를 수신하여 검증을 수행한다
검증 가능한 데이터 레지스트리	· 발급자의 Public Key, VC 등 식별자, 키 및 기타 관련 데이터의 생성과 검증을 중재할 수 있는 저장소로, 신뢰할 수 있는 데이터베이스, 분산형 데이터베이스를 포함할 수 있다

○ **(DID가 포함된 VC/VP 시스템)** VC/VP 시스템에서 DID를 사용하는 경우, VC/VP 시스템을 구현하는 소프트웨어 라이브러리는 DID를 해석해야 한다. DID 기반 URL은 주체, 발급자, 소유자, 자격 증명 상태 목록, 암호화 키 및 VC 관련 기계 판독 가능 정보와 관련된 식별자를 표현하는 데 사용된다

4-5. DID 관련 표준 및 서비스 동향

○ DID 관련 표준문서는 주로 W3C에서 관리되며, Decentralized Ientifiers v1.0 표준문서 외 다수의 Draft 문서가 존재한다

○ IETF RFC 표준문서는 DNS Resource Record (RR) 유형 등에 대해서만 언급되며, DID 관련 문서는 Internet Draft만 존재한다

○ DID 기반 서비스는 디지털 신원 관리를 중점으로 하며, 여러 국내 · 외 대기업과 기관의 주목을 받아 다양한 지원사업 및 기술개발이 이뤄지고 있다

4-6. DID 직접해석방안 제안 & 모델 설계

○ **(DID 직접해석 개념)** 특정 DID를 Web 브라우저 등에 입력했을 때, 각 DID Method 별로 요구되는 해석기 없이 DID 문서를 반환할 수

있는 해석 방법으로, 기존 방식의 확장성 제한 해결을 목표로 한다. Web Application Server (WAS), DNS, 블록체인 및 신규 DID Method 혹은 신규 Resolver 등 다양한 방식의 기술적 접근이 가능하다

- ㅇ **(WAS 활용방안)** DID Resolver의 모음인 Universal Resolver를 WAS로 활용하여 다양한 DID Method의 DID 문서를 반환할 수 있다
- ㅇ **(블록체인 활용방안)** 블록체인 네트워크를 DID Resolver의 주소를 저장하는 분산 디렉토리 서비스로 활용하여 다양한 DID Method의 DID 문서를 반환할 수 있다
- ㅇ **(DNS 활용방안)** DNS로부터 반환받는 TXT RR에 추가적인 메타데이터를 포함시켜 확장하는 방안으로, RR에 포함된 값에 여러 신뢰기반을 포함하여 상호운용성을 보장할 수 있다. did:direct라는 신규 DID Method로써 표준으로 지정할 수 있다
- ㅇ **(DID Common Resolver 활용방안)** DID Common Resolver라는 신규 Resolver를 활용하는 방안으로, 클라이언트 내부에서 DNS Resolver 등과 통신 및 문자열 Parsing을 수행하고 다양한 DID Method에게 DID 문서를 요청 및 반환받는 API 역할을 수행한다

[표 6] DID 직접해석 기술적 접근방안별 장 · 단점

직접해석 기반	장점	단점
WAS (Universal Resolver)	· 각 DID Method가 기존에 사용하던 형식의 식별자를 입력할 수 있다 · 간편하게 단일 인터페이스를 사용해 식별자를 입력받고, DID 문서를 반환할 수 있다	· 각 DID Method 해석을 위한 드라이버는 Universal Resolver에 종속되어 개발 및 유지관리 복잡성 상승한다 · 중앙 해석기에 대한 의존성 상승으로 인한 탈중앙화 원칙과의 상충이 우려될 수 있다
블록체인 (분산 디렉토리	· 탈중앙화된 방법으로 직접해석 지원 및 데	· 초기 네트워크 구축 및 Smart Contract

서비스)	이터의 무결성 · 신뢰성을 보장한다 · 새로운 DID Method 추가 및 DID Resolver 변경이 간편하여 확장성이 용이하다	비용 부담이 존재하고 DNS 대비 네트워크 성능이 낮다 · Smart Contract 오류, HTTPS 리다이렉션, 서비스제공자간 신뢰 문제에 취약하다
DNS (신규 DID Method)	· 기존 DNS 인프라를 활용할 수 있다 · 간단한 방식으로 구현이 가능하다	· 서버 운영자 등, DNS 관리 주체에 종속될 수 있다 · 스푸핑 등 DNS 보안 취약점 방지 대책을 고려해야 한다
DID Common Resolver (신규 Resolver)	· 기존 DNS 인프라를 활용할 수 있다 · 간단한 방식으로 구현이 가능하다 · 각 DID Method에 대한 Resolver를 호출하는 API 역할을 수행함으로 인해 의존성과 복잡성이 낮다	· 스푸핑 등 DNS 보안 취약점 방지 대책을 고려해야 한다 · DID 서비스를 제공하는 DID Controller의 부담이 높다

5. 기대효과

- ㅇ 국제 표준화 기구의 요구사항을 만족하여 향후 DID 표준화에 기여할 수 있다
- ㅇ 여러 DID Method를 아우르는 DID 직접해석을 통해 상호운용성을 보장하여 DID 적용 산업을 확대하고, 복잡성을 감소시켜 관련 기술 개발을 촉진시킬 수 있다

목 차

그림 목차

표 목차

제 1 장 기술개발과제 개요

제 1 절 연구개발의 필요성

1. 연구 배경

- ㅇ 최근 혁신적인 탈중앙 구조와 신뢰성 있는 검증 여부로 인해 분산 ID (Decentralized Identifier, DID)에 대한 관심도가 높아지고 있다

- ㅇ 기존 고유 식별자 대비 DID는 보다 안전하고 투명한 신원 관리 및 증명이 가능해 전화번호, 이메일, 여권, 면허, 제품 식별자 등 광범위한 분야를 대체할 수 있다

- ㅇ 다양한 방식으로 구현이 가능해 지속적으로 새로운 DID Method들이 발표되고 있으나 동작 방식, 신뢰 기반 및 필요 인프라가 상이하여 해석에 어려움이 존재하고 확장성이 제한된다는 문제가 존재한다

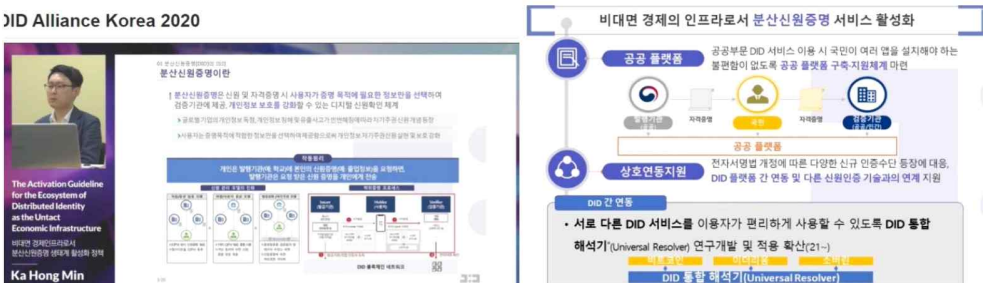
- ㅇ 따라서 기존 DID 생태계와의 호환성을 보장하면서도 특정 해석기에 대한 의존성 없이 DID 문서를 조회할 수 있는 기술방안에 대한 연구를 수행할 필요가 있으며, 지속적인 상호운용성을 보장하기 위해 표준화를 진행할 필요가 있다

2. 국내 · 외 동향

- ㅇ DID는 현재 다수의 표준화 기구에서 활발하게 다뤄지고 있다. W3C는 DID를 표준으로 승인하였으며, 2017년에 DID의 활용 및 채택을 가속화하기 위해 Decentralized Identity Foundation (DIF) 가 설립되

었다. 또한 European Blockchain Services Infrastructure (EBSI), Silicon Valley Innovation Program (SVIP) 등에서 관련 기술을 개발하는 등 상호운용성을 고려한 개발이 이루어지고 있다 [1]

ㅇ 국내에서는 2020년에 과학기술정보통신부의 DID 통합 해석기 제작 발표 및 상호호환/인증 논의 본격화 [2]가 이루어졌으며, DID의 공공기관 보급을 늘리고 민간 분야에서의 활용성을 높이기 위한 DID 통합 해석기의 연구 개발을 진행하여 조기 생태계 활성화를 위한 추진체계를 마련하고 글로벌 표준을 선도하려는 움직임이 증가하고 있다 [3]



(그림1-1) 과학기술정보통신부의 DID 통합 해석기 제작 발표

ㅇ DID 해석 관련 관심도가 높아지고 있는 상황에서 해석기가 불필요한 DID 직접해석 방안 마련 및 글로벌 표준화를 적용하면 자율성, 보안성, 확장성이 뛰어난 DID 시스템 구축에 이바지할 수 있다

제 2 절 연구개발의 목표

1. 기술조사

ㅇ 해석기 없이 다른 DID 식별자를 Web 브라우저에 입력하여 검증정보 (DID 문서)를 조회 · 검증할 수 있는 기술방안 도출
- DID Method 및 P2P 네트워크 기술 조사를 기반으로 기술방안 기획 및 제안
- 관련 플랫폼, 프레임워크, 응용 기술 등 기술조사 및 적용 가능성 검토를 통한 상호 운용성 및 신뢰성이 뛰어난 검증 방안 탐색

ㅇ DID 식별자, 방법론 등 최신 동향 조사 및 해석방안 마련

2. 모델설계

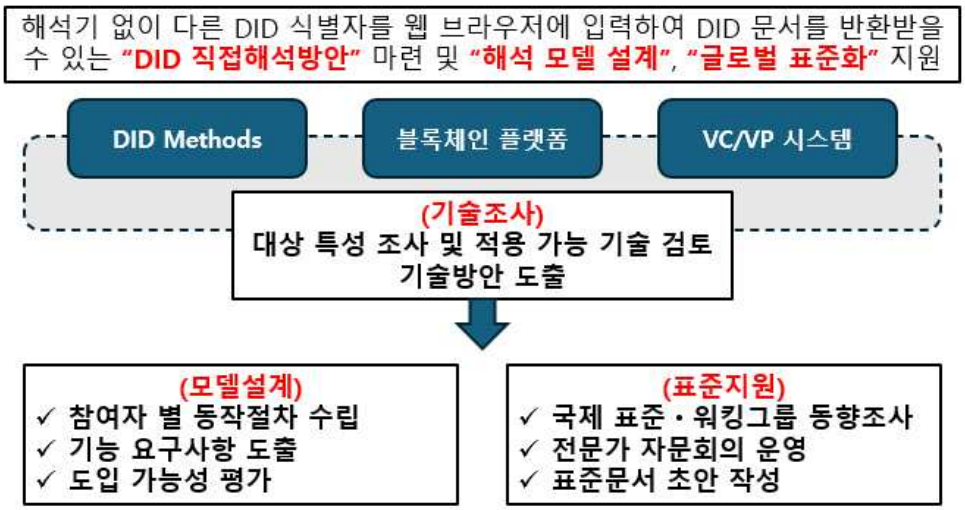
ㅇ DID 직접 연계방안의 가능성을 검토하기 위한 참여자 별 동작절차, 기능 요구사항 도출 및 해석모델 설계
- 기술조사 내용을 바탕으로 모델 설계 전략 수립 및 참여자 별 동작절차, 요구사항 도출, 장단점 및 특징 분석
- 관련 전문가로 구성된 자문회의 운영을 통해 설계 모델의 도입 가능성 (실용성/보안성/확장성 등) 평가 수행 및 고도화

3. 표준지원

ㅇ 국제 인터넷 표준화 기구 (W3C, IETF) 인터넷 표준 초안 제출을 위한 “DID 직접해석 방안” 표준문서 마련
- W3C, IETF 표준문서 분석 및 표준화 절차, 워킹그룹 등 표준화 동향 조사

- 표준문서 작성, 관련 전문가로 구성된 표준 전문가 자문회의 운영, 협력체계 구축 등 행정 지원 수행

○ 표준문서, 워킹그룹 현황 등 최신 표준화 동향 조사 및 분석, 표준화를 위한 행정 지원



(그림1-2) 최종 목표 및 연구 내용

제 2 장 DID 기술개요

제 1 절 DID 핵심규격

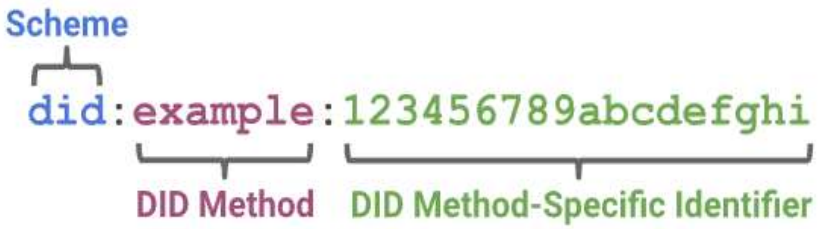
1절에서는 DID 핵심 규격에 대해서 다루며, W3C에서는 해당 부분을 DID-CORE 문서 [4]에 정의하였다

1. DID

- ㅇ 검증 가능한 분산된 디지털 ID를 가능하게 하는 새로운 유형의 식별자로, URI (Uniform Resource Identifier)의 한 형태이다

- ㅇ DID 주체를 DID 문서와 연결하여 해당 주체와 신뢰할 수 있는 상호작용을 가능하게 한다

- ㅇ URI 체계 식별자, DID Method, DID Method 별 식별자로 구성된 단순한 텍스트 문자열이며, Resolve 과정을 거쳐 연결된 DID 문서를 가져올 수 있다



(그림2-1) DID 형식 예시

- ㅇ 그림 2-2 예시와 같이 DID 문서는 키-값 형태로 표현되며, DID Controller를 암호화하여 인증하는 방법 등의 정보들이 포함된다

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

(그림2-2) DID 문서 예시

ㅇ DID는 Verifiable Credential (VC) System과 같은 더 큰 규모의 자격 증명 시스템의 구성요소이며, 설계 목표는 다음과 같다

- **(탈중앙화)** 전역 고유 식별자, Public Key 등 정보의 등록을 포함하여 식별자 관리에서 중앙화된 권한 또는 단일 실패 지점에 대한 사항들을 제거해야 한다
- **(제어)** 사람과 사물 등 모든 개체에 외부 기관 의존 없이 디지털 식별자를 직접적으로 제어할 수 있는 권한을 부여할 수 있어야 한다
- **(프라이버시)** 엔티티가 속성이나 기타 데이터를 최소한, 선택적, 점진적으로 공개하는 것을 포함하여 개인 정보 보호 기능을 제어할 수 있도록 해야 한다
- **(보안)** 요청 당사자가 필요한 수준의 보증을 위해 DID 문서에 의존할 수 있도록 충분한 보안을 제공해야 한다
- **(증명 기반)** DID Controller가 다른 엔티티와 상호작용할 때 암호화 증명을 제공할 수 있게 해야 한다
- **(발견 가능성)** 엔티티가 다른 엔티티의 DID를 검색하고 해당 엔티티에 대해 더 자세히 알아보거나 해당 엔티티와 상호작용할 수 있게 해야 한다
- **(상호운용성)** 상호운용 가능한 표준을 사용하면 DID 인프라가 상호

운용성을 위해 설계된 기존 도구와 소프트웨어 라이브러리를 활용할 수 있어야 한다

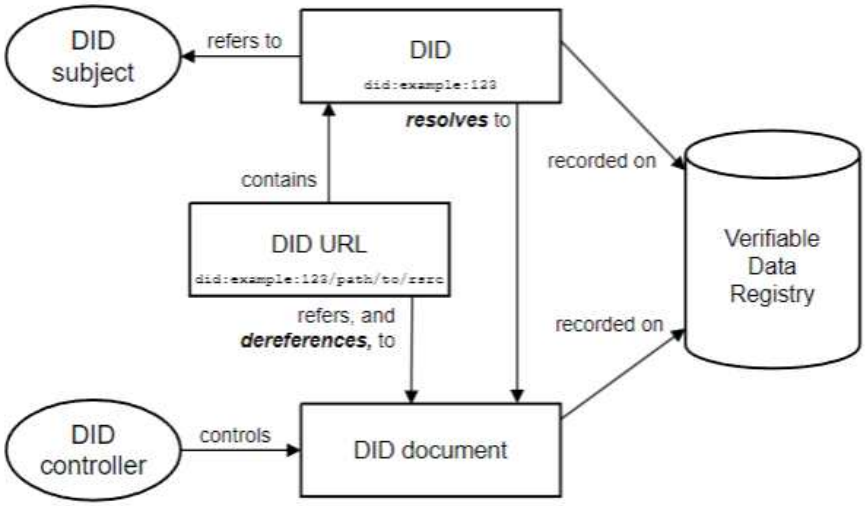
- **(이동성)** 시스템과 네트워크에 독립적이고, 기업이 DID 및 DID Method를 지원하는 모든 시스템에서 디지털 식별자를 사용할 수 있게해야 한다
- **(단순함)** 기술을 더 쉽게 이해 및 구현하고 배포할 수 있도록 간단한 기능들을 축소해야 한다
- **(확장성)** 가능하다면 상호운용성, 이동성, 단순함 3가지 목표를 크게 저해하지 않는 한에서 확장 가능성을 제공해야 한다

2. DID Architecture 주요 구성 요소

ㅇ DID architecture의 주요 구성 요소에 대한 개요는 다음과 같다

- **(DID 및 DID URL)** DID는 URI의 한 형태이다. DID URL은 기본 DID 구문을 확장하여 경로, 쿼리 등의 다른 표준 URI 구성 요소를 통합하여 DID 문서 내의 암호화 Public Key, 외부 리소스 등의 특정 리소스를 찾을 수 있다
- **(DID 주체)** DID에 의해 식별된 엔티티로서, DID 주체 자체가 DID Controller일 수도 있으며, 사람, 그룹, 조직, 사물, 개념 등 무엇이든 DID 주체가 될 수 있다
- **(DID Controller)** DID Controller는 DID Method에 정의된 대로 DID 문서를 변경할 수 있으며, DID에 2개 이상의 Controller가 있을 수도 있다
- **(검증 가능한 저장소)** DID 문서로 Resolve하기 위해 DID는 일반적으로 기본 시스템이나 어떤 종류의 네트워크에 기록된다. 특정 기술에 관계없이 DID를 기록하고 DID 문서를 생성하는 데 필요한 데이터를 반환하는 모든 시스템은 검증 가능한 레지스트리라고 표현한다. 예로는 분산 파일 시스템, 분산 원장, P2P 네트워크 및 기타 형태의 신뢰할 수 있는 데이터 저장소가 있다

- **(DID 문서)** DID 문서는 DID 관련 정보를 포함하는 문서이다. 일반적으로 암호화, Public Key와 같은 검증 방법과 DID 주체와의 상호작용 관련 서비스를 표현하며, 바이트 스트림으로 직렬화될 수 있다
- **(DID Methods)** 특정 유형의 DID 및 관련 DID 문서가 생성, resolve, 업데이트, 비활성화되는 메커니즘을 일컫는다
- **(DID Resolvers, Resolution)** resolver는 DID를 입력으로 받아 해당하는 DID 문서를 출력으로 가져오는 역할을 수행하며, 관련 방법은 DID의 유형 및 DID Method 사양에 따라 정의됨
- **(DID URL Dereferencers, Dereferencing)** Dereferencer는 DID URL을 입력으로 받아 출력으로 리소스를 생성한다



(그림2-3) DID Architecture 구성요소

3. DID Syntax

ㅇ 일반적인 DID 스키마는 RFC 3986 [5]에 따른 URI 스키마이며, 모든 DID는 그림 2-4의 ABNF Rules를 따라야 한다

The DID Syntax ABNF Rules

```
did                = "did:" method-name ":" method-specific-id
method-name        = 1*method-char
method-char        = %x61-7A / DIGIT
method-specific-id = *( *idchar ":" ) 1*idchar
idchar             = ALPHA / DIGIT / "." / "-" / "_" / pct-encoded
pct-encoded        = "%" HEXDIG HEXDIG
```

(그림2-4) DID syntax ABNF Rules

4. DID URL Syntax

○ DID URL은 특정 리소스에 대한 네트워크 위치 식별자로 그림 2-5의 규칙을 따라야 한다. DID 주체, 검증 방법, 서비스, DID 문서의 특정 부분 또는 기타 리소스의 표현 등을 검색할 수 있다

The DID URL Syntax ABNF Rules

```
did-url = did path-abempty [ "?" query ] [ "#" fragment ]
```

(그림2-5) DID URL Syntax ABNF Rules

- **(Path)** Path-abempty ABNF Rules를 따르고 일반적인 URI 경로와 동일하며, DID Method로 경로 의미를 지정하여 해당 의미를 더욱 특수화할 수 있다
- **(Query)** 일반적인 URI 쿼리와 동일하고, ABNF Rules를 따른다
- **(Fragment)** 일반적인 URI fragment과 동일하고, ABNF Rules를 따르며 DID 문서 또는 외부 리소스에 대한 독립적인 참조로 사용된다

제 2 절 DID Resolution

2절에서는 DID Resolution에 대해서 다루며, W3C에서는 해당 부분을 DID-CORE 문서 [4]에 정의하였다

1. DID Resolution

- DID Resolver는 최소한 하나의 DID Method에 대한 DID Resolution 기능을 구현해야 하며, 해당 규격에 맞는 표현으로 DID 문서를 반환할 수 있어야 한다

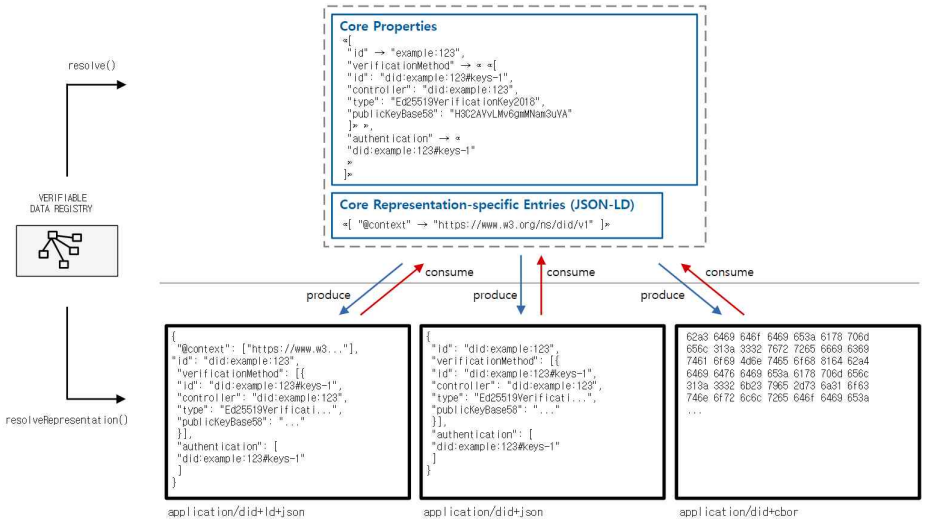
- DID Resolution 기능은 해당 DID Method의 "Read" 작업을 사용하여 DID를 DID 문서로 Resolve하는 역할을 수행한다

- resolve 함수는 DID 문서의 구조적 데이터를 추상적인 형태 (맵)로 반환하며, resolveRepresentation 함수는 특정 표현 형식으로 포맷된 DID 문서의 바이트 스트림을 반환한다

```
resolve(did, resolutionOptions) →
  « didResolutionMetadata, didDocument, didDocumentMetadata »

resolveRepresentation(did, resolutionOptions) →
  « didResolutionMetadata, didDocumentStream, didDocumentMetadata »
```

(그림2-6) resolve, resolveRepresentation 함수 1

- resolve 및 resolveRepresentation 함수의 입력 값은 아래와 같다
  - **(did)** 필수적인 입력이며, 그 값은 DID Syntax 규칙에 맞는 DID여야 한다
  - **(resolutionOptions)** 표준으로 정의된 DID 문서 표현의 미디어 타입 속성을 포함하는 메타데이터 구조이다



(그림2-7) resolve, resolveRepresentation 함수 2

ㅇ resolve 및 resolveRepresentation 함수의 반환 값은 아래와 같다

- **(didResolutionMetadata)** DID Resolve의 결과와 관련된 값을 포함하는 메타데이터 구조이며 필수적인 요소이다. resolveRepresentation 함수가 호출된 경우, didDocumentStream에서 발견된 contentType 속성을 반드시 포함해야 한다. Resolve 과정에 문제가 있을 경우, 이를 설명하는 error 속성을 반드시 포함해야 한다
- **(didDocument)** Resolve가 성공적일 때, 적합한 표현의 DID 문서로 변환될 수 있는 DID 문서 추상 데이터 모델이어야 한다. Resolve된 DID 문서의 id 값은 반드시 해석된 DID와 일치해야 하며 Resolve를 실패한 경우, 이 값은 비어 있어야 한다
- **(didDocumentStream)** Resolve가 성공적이고 resolveRepresentation 함수가 호출된 경우, 이는 적합한 표현 중 하나로 Resolve된 DID 문서의 바이트 스트림이어야 한다. 바이트 스트림은 함수 호출자에 의해 데이터 모델 형태로 Parsing될 수 있고, 이는 다시 검증, 처리될 수 있다. Resolve를 실패한 경우, 이 값은 비어 있어야 한다
- **(didDocumentMetadata)** Resolve가 성공적일 때 문서 상태 정보인 활

성, 폐기, 비활성 등 didDocument 속성에 포함된 DID 문서에 대한 메타데이터를 포함하며 Resolve를 실패한 경우, 이 출력은 반드시 비어 있어야 한다

2. DID Resolution Options/Metadata

- (accept) 선택사항으로, resolveRepresentation 함수에서만 사용되며, 그림2-7 과 같이 호출자가 선호하는 DID 문서 표현의 미디어 타입을 나타낸다 (JSON-LD, CBOR 형식 등). 이 미디어 타입은 ASCII 문자열로 표현되어야 한다

- (contentType) 반환된 didDocumentStream의 미디어 타입으로, 적합한 표현의 미디어 유형인 ASCII 문자열이어야 한다. resolveRepresentation 함수의 호출자는 이 값을 사용하여 반환된 didDocumentStream을 데이터 모델로 Parsing하고 처리해야 한다

- (error) resolve 과정에서의 에러 코드로, 프로세스상의 오류가 있을 시 필수적이다. 일반적인 오류 값은 다음과 같다
  - (invalidDid) 제공된 DID가 유효한 구문을 따르지 않음을 의미한다
  - (notFound) DID Resolver가 요청된 DID 문서를 찾을 수 없음을 의미한다
  - (representationNotSupported) 요청된 표현이 지원되지 않음을 의미한다

3. DID URL Dereferencing

- DID URL Dereferencing 기능은 DID URL을 입력으로 받아서 리소스로 Dereferencing 한다

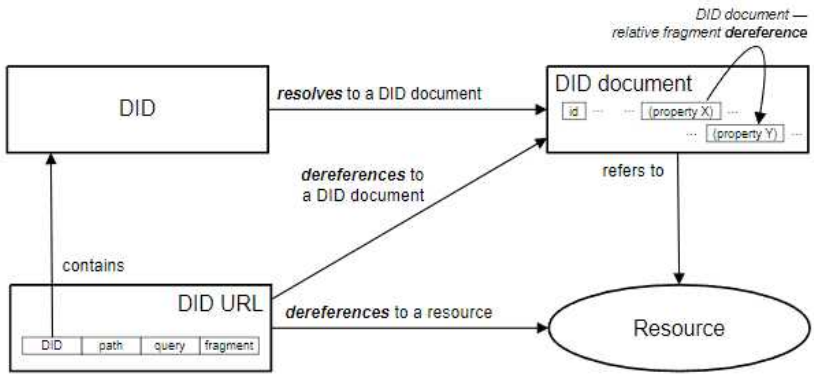
○ Dereferencing 내용은 DID URL의 구성 요소인 DID Method, Method 별 식별자, Path, Query, Fragment 등에 따라서 달라진다

○ Dereferencing 과정은 DID URL에 포함된 DID의 DID Resolution에 따라 달라진다. DID URL 역참조에는 여러 단계가 포함될 수 있으며 (역참조되는 DID URL에 fragment가 포함된 경우) 모든 단계가 완료된 후 최종 리소스를 반환하도록 함수가 정의된다

○ 모든 규격에 맞는 DID Resolver는 다음과 같이 추상적인 형식을 갖는 함수를 구현한다

```
dereference(didUrl, dereferenceOptions) →
   « dereferencingMetadata, contentStream, contentMetadata »
```

(그림2-8) dereference 함수



(그림2-9) DID URL dereferencing 개요

○ dereferencing 함수의 입력 값은 다음과 같다

- **(didUrl)** 필수적인 입력으로, 역참조하게 되는 DID URL이며 단일 문자열로 구성된다. DID Fragment를 역참조하려면 DID Fragment를 포함한 전체 DID URL을 사용해야 한다

- **(dereferenceOptions)** dereference 함수와 DID URL 자체에 대한 입력 옵션을 포함하는 메타데이터 구조이다

ㅇ dereferencing 함수의 반환 값은 다음과 같다

- **(dereferencingMetadata)** 필수적인 반환 값으로, DID URL 역참조 과정의 결과에 관한 값을 포함하는 메타데이터 구조이며 역참조에 실패한 경우, 이 구조는 오류를 설명하는 error 속성을 포함해야 한다
- **(contentStream)** 함수가 호출되고 성공하면 DID URL에 해당하는 리소스를 포함해야 하며, 반환 값은 DID 문서, 검증 방법, 서비스 등 다양한 형태의 자원일 수 있다. Dereference를 실패한 경우, 이 값은 비어 있어야 한다
- **(contentMetadata)** Dereference에 성공하면 반환하는 값으로, contentStream에 대한 메타데이터를 포함하는 구조이다. DID 문서를 반환한 경우, DID Resolution에서 설명된 didDocumentMetadata 구조여야 한다. 역참조를 실패한 경우, 비어 있어야 함

4. DID Dereferencing Options/Metadata

ㅇ (accept) 호출자가 선호하는 DID 문서 표현의 미디어 타입을 ASCII 문자열로 표현한 것을 일컫는다. DID URL 역참조 구현은 반환 값에 포함되는 표현의 contentType를 결정하기 위해 이 값을 사용해야 한다

ㅇ **(contentType)** 반환된 contentStream의 미디어 타입으로, 역참조를 성공한 경우 필수적인 요소이다. 값은 적합한 표현의 미디어 유형인 ASCII 문자열이다

ㅇ **(error)** 역참조 과정에서의 에러 코드로서, 단일 키워드 ASCII 문자열로 구성된다. 일반적인 오류 값은 다음과 같다

- **(invalidDidUrl)** DID URL이 유효한 구문을 따르지 않음을 의미한다
- **(notFound)** DID URL Dereferencer가 요청된 contentStream을 찾을 수 없음을 의미한다

제 3 절 DID Actions

3절에서는 Create, Read 등 DID의 동작에 대해서 다루며, W3C에서는 해당 부분을 DID-USE-CASES 문서 [6]에 정의하였다

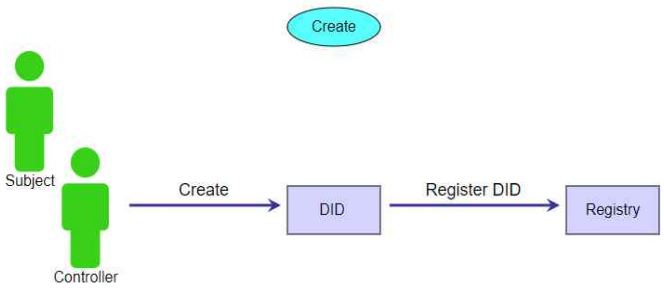
1. Create

- Controller는 DID를 생성하여 암호화 증명을 식별자와 고유하게 바인딩하며, 일반적으로 공개-비공개 키 쌍을 사용한다

- 이러한 DID는 DID 문서로 Resolve할 수 있는 방식으로 저장소에 기록될 수 있다

- DID 문서는 Resolve를 통해 동적으로 생성되거나 독립 실행형 리소스로 명시적으로 구성되고, 저장소에 저장되거나 참조될 수 있다

- 시나리오상 프로세스는 모든 저장소, 이상적으로는 분산 시스템에 액세스해야 하며 나머지 DID 작업과 마찬가지로 특정 권한과 상호작용하지 않고도 DID를 생성할 수 있어야 한다



(그림2-10) DID 생성 및 등록에 관한 기본 흐름

2. Use

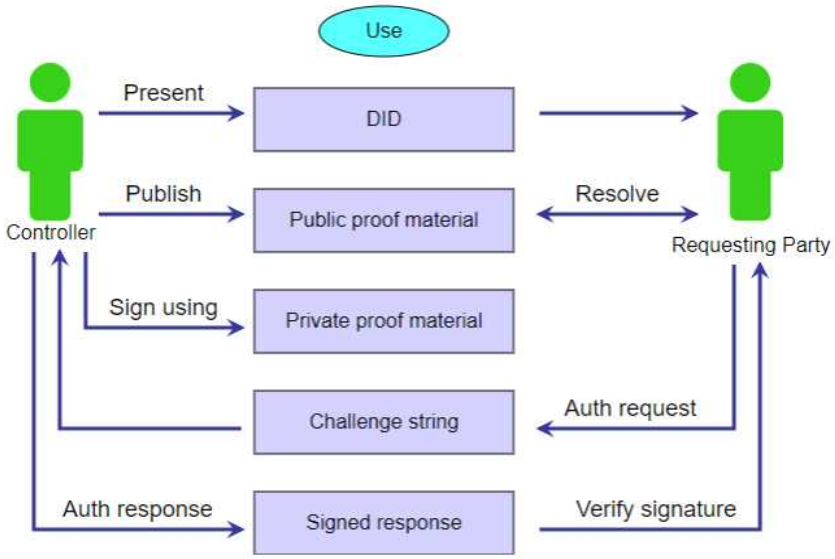
ㅇ DID는 문자열인 URI이기 때문에 해당 문자열을 전송하거나 표시하기만 하면 URI와 동일한 방식으로 표시될 수 있다

ㅇ DID가 사람이 읽을 수 있어야 한다는 요구사항은 없으므로 다양한 형식으로 표현된 길고 복잡한 숫자를 포함할 수 있다

ㅇ 사용 편의성을 위해 구현은 스마트폰과 같은 카메라 지원 장치를 사용하여 쉽게 캡처할 수 있도록 QR 코드와 같은 데이터 캐리어에 의존할 수 있다

ㅇ 요청 당사자는 DID를 제시하는 개인이 실제로 DID Controller이거나 특정 서비스 엔드포인트에 대한 Controller로 지정되었는지 증명하고자 할 수 있다

- (authenticate) 이 프로세스는 DID 문서의 암호화 자료를 사용하여 주장된 Controller가 일반적으로 일종의 Challenge-response를 통해 실제로 제어를 증명할 수 있는지 테스트해야 한다. DID 문서와 Method는 업데이트 및 삭제 작업과 별도로 다른 서비스 엔드포인트에 대한 별도의 증명을 허용할 수 있다
- (sign) DID 문서에서 발견된 것과 관련된 암호화 자료를 사용하여 DID Controller는 디지털 자산이나 문서에 서명할 수 있다. 이 서명은 나중에 자산의 진위성을 입증하기 위해 검증될 수 있다
- (verify signature) DID로 서명된 디지털 자산이 제공된 경우, 요청 당사자는 DID 문서의 암호화 자료를 사용하여 서명을 확인할 수 있다



(그림2-11) 익명 인증 목적의 DID 사용에 관한 인증 상호작용 흐름

3. Read

○ **(Resolve)** 프레젠테이션 이외의 용도로 DID를 사용하는 첫 번째 단계는 DID를 특정 DID 문서로 Resolve하여 해당 DID와 관련된 암호화 자료와 서비스 엔드포인트를 공개하는 것이며, 이는 메소드에 따라 다르게 설정된다

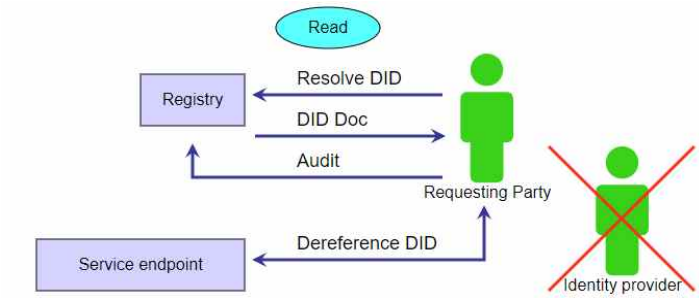
○ **(Dereference)** DID의 Dereference는 DID 문서의 자료를 사용하여 리소스를 반환한다

- 서비스 엔드포인트에 대한 참조 없이 DID를 Dereferencing하면 DID 문서 자체가 반환된다
- DID가 DID URL을 형성하는 Service 파라미터와 결합되면, Dereference는 DID를 DID 문서로 Resolve하고 엔드포인트를 조회하여 발견한 서비스 엔드포인트에서 가리키는 리소스를 반환한다
- 요청 당사자는 주어진 DID에 대한 현재 서비스 엔드포인트를 동적

으로 발견하고 상호 작용할 수 있으며. 기본 서비스 엔드포인트가 변경되어도 변경되지 않는 영구 식별자를 서비스에 지정할 수 있다

○ **(Audit)** 일부 메소드는 해당 DID의 모든 작업에 대한 명시적 감사 추적을 제공할 수 있고, 여기에는 작업이 발생한 시점의 타임스탬프도 포함된다

- 분산 원장 기반 저장소의 경우, 이 감사 추적은 원장이 거래를 기록하는 방식에 기본이 된다
- 요청 당사자는 DID가 얼마나 최근에 Rotate 되었는지, 서비스 엔드포인트가 업데이트되었는지 확인할 수 있으며, 이는 DID의 암호화 자료의 신뢰성에 관한 특정 분석에 정보를 제공할 수 있다



(그림2-12) DID 문서의 상호작용 과정

4. Update

○ **(Rotate)** Controller는 저장소에 기록된 DID 문서를 업데이트하여 DID의 암호화 자료를 회전, 즉 업데이트 할 수 있다

- DID Method마다 다르게 처리될 수 있어야 하고, 결과는 DID 및 DID 문서의 제어를 증명하는 데 필요한 핵심 암호화 증명에 대한 업데이트가 된다

○ **(서비스 엔드포인트 수정)** DID Controller는 DID에 관한 서비스 엔드포인트를 변경할 수 있어야 하며, 여기에는 주어진 엔드포인트에 대한 주체로 인증하기 위한 증명 메커니즘도 포함된다

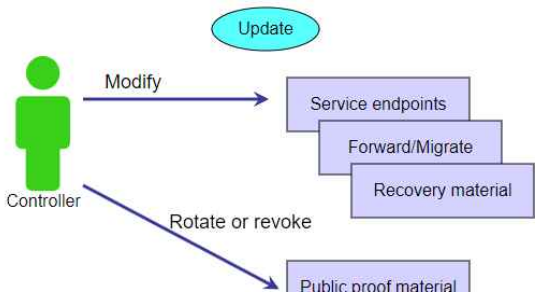
- 이를 수행하는 프로세스는 메소드에 따라 다르지만, DID Controller가 DID 자체의 제어를 위한 기본 증명 메커니즘을 반드시 변경하지 않고도 이러한 변경을 수행할 수 있도록 설계되어야 한다

○ **(Forward/Migrate)** 상호운용성을 지원하기 위해 일부 메소드는 DID Controller가 저장소에 기록하는 방법을 제공할 수 있다

- DID는 DID를 나타낼 수 있는 모든 권한을 가진 다른 DID로 리다이렉션 되어야 하며, 해당 메커니즘을 통해 DID Controller는 DID를 어떠한 하나의 Method나 저장소에서, 다른 DID Method 또는 저장소로 Migrate할 수 있다

○ **(Recover)** 일부 DID Method는 기존 개인 암호화 자료가 손실된 경우 DID의 제어를 복구하는 수단을 제공할 수 있다. 복구 수단은 메소드에 따라 다르지만, 소셜 복구, 다중 서명, Shamir 공유 또는 사전 회전 키를 포함할 수 있다

- 일반적으로 복구는 새로운 증명으로의 회전을 트리거하여 해당 새로운 증명의 DID Controller가 요청 당사자와 상호 작용하지 않고도 DID의 제어를 복구할 수 있도록 한다

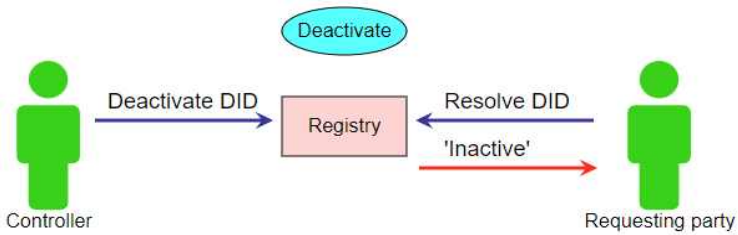


(그림2-13) DID의 유용함을 보장하고자 Controller가 수행하는 옵션

5. Delete

ㅇ **(Deactivate)** DID를 삭제하는 대신, Controller는 DID를 비활성화하여 인증 및 Dereferencing과 같은 다운스트림 프로세스가 더 이상 작동하지 않게 해야 한다

- 대부분의 분산 시스템은 레코드의 실제 삭제를 보장할 수 없으며, DID Method는 삭제와 동일한 효과를 얻기 위해 비활성화 프로세스를 정의해야 한다
- 비활성화 메커니즘은 DID Method에 따라 다르므로 “비활성” 메시지도 DID Method에 따라 다르다



(그림2-14) 비활성화할 수 있는 DIDs

제 4 절 DID Documents and Data Model

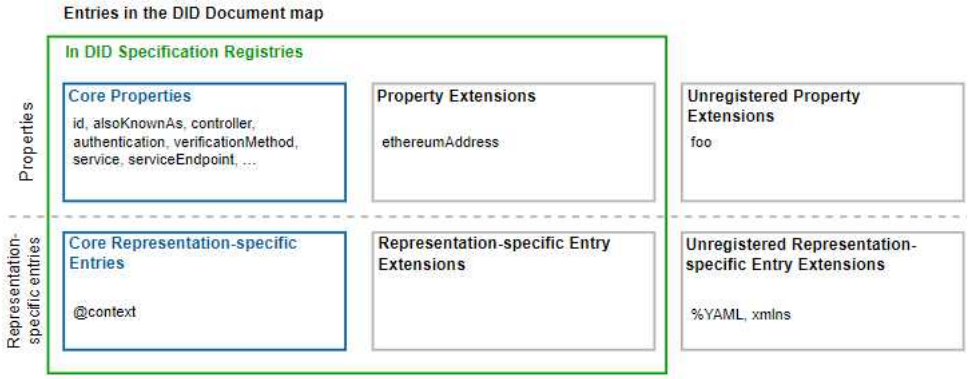
4절에서는 DID 문서와 데이터 모델에 대해서 다루며, W3C에서는 해당 부분을 DID-CORE [4] 및 DID-SPEC 문서 [7]에 정의하였다

1. Data Model

ㅇ DID 문서와 데이터 구조를 표현하는 데 사용할 수 있는 데이터 모델은 여러 구체적인 표현으로 직렬화할 수 있다

ㅇ DID 문서는 키/값 쌍으로 구성된 항목의 맵으로 구성되며, DID 문서

데이터 모델은 최소한 속성(properties), 표현별 항목(representation specific entries) 두 가지 다른 클래스의 항목을 포함해야 한다



(그림2-15) DID 문서의 엔트리

○ DID 문서 데이터 모델의 모든 항목 키는 문자열이며, 값은 표 2-1의 추상 데이터 유형 중 하나를 사용하여 표현되고, 각 표현은 데이터 유형의 구체적 직렬화 형식을 지정해야 한다

[표2-1] 데이터 모델 항목 키의 데이터 유형

데이터 유형	내용
map	키가 두 번 나타나지 않는 키/값 쌍의 유한한 순서화된 시퀀스
list	항목의 유한한 순서화된 시퀀스
set	동일한 항목을 두 번 포함하지 않는 유한한 순서가 있는 항목 시퀀스
datetime	모든 값을 손실 없이 표현할 수 있는 날짜 및 시간 값
string	인간이 읽을 수 있는 언어를 표현하는 데 자주 사용되는 일련의 코드
integer	분수 구성 요소가 없는 실수로서, 상호운용성 극대화를 위해 구현자에게 정수 관련 표준을 권장함
double	임의의 실수를 근사하는 데 자주 사용되는 값으로, 표준을 따르는 것이 좋음
boolean	참 또는 거짓인 값

null	값의 부족을 나타내는 데 사용되는 값

2. Core Properties

○ DID 문서는 데이터 모델을 사용하여 표현되며, 직렬화될 수 있다

○ DID 문서의 속성은 표 2-2에 정의된 표준 핵심 속성에 대한 정보 참조와 필수 여부를 포함하며, DID 주체와 속성값 간의 관계를 설명한다

[표2-2] DID 문서 속성

속성	필수	내용
id	Y	DID Syntax 규칙에 맞는 문자열
alsoKnownAs	N	RFC 3986 [5]규칙을 준수하는 문자열 집합
controller	N	DID Syntax 규칙에 맞는 문자열 또는 문자열 집합
verificationMethod	N	검증 방법 속성의 규칙을 따르는 검증 방법 맵 세트
authentication	N	검증 방법 속성의 규칙을 준수하는 검증 방법 맵 세트 또는 DID URL Syntax 검증 방법 속성의 규칙을 준수하는 문자열 세트
assertionMethod	N	
keyAgreement	N	
capabilityInvocation	N	
capabilityDelegation	N	
service	N	서비스 속성의 규칙을 따르는 서비스 엔드 포인트 맵 세트

[표2-3] 검증 방법 속성

속성	필수	내용
id	Y	DID URL Syntax 규칙에 맞는 문자열
controller	Y	DID Syntax 규칙에 맞는 문자열
type	Y	문자열

publicKeyJwk	N	RFC 7517 [8]에 따른 JSON Web 키를 나타내는 맵
publicKeyMultibase	N	Multibase 방식으로 인코딩된 Public Key에 맞는 문자열

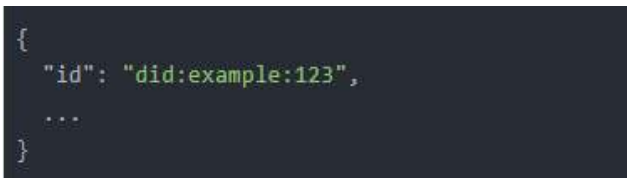
[표2-4] 서비스 속성

속성	필수	내용
id	Y	RFC 3986 [5]규칙을 준수하는 문자열
type	Y	문자열 또는 문자열 집합
serviceEndpoint	Y	RFC 3986 [5]규칙을 준수하는 문자열, 맵 또는 URI 및/또는 맵에 대한 RFC 3986 [5]을 준수하는 하나 이상의 문자열로 구성된 세트

3. 식별자 관련 속성

○ **(DID Subject)** 특정 DID 주체에 대한 DID는 DID 문서의 id 속성을 사용하여 표현된다

- **(id)** DID 문서의 최상위 맵에 존재하는 경우에만 DID 주체의 DID임을 나타내며, 이 속성값은 반드시 DID 구문 규칙을 따르는 문자열이어야 한다



(그림2-16) id 속성 예시

○ **(DID Controller)** DID 문서를 변경할 수 있는 권한이 있는 엔티티로, DID Controller를 승인하는 프로세스는 DID Method로 정의된다

- **(controller)** 선택 사항으로, 해당 속성이 존재하는 경우 값은 DID 구문 규칙을 따르는 문자열 또는 문자열 세트여야 하며, 해당 DID 문

서에는 특정 목적을 위해 특정 검증 방법의 사용을 명시적으로 허용하는 검증 관계 속성이 포함되어야 한다

(그림2-17) controller 속성 예시

ㅇ **(alsoKnownAs)** 두 개 이상의 DID (또는 다른 유형의 URI)가 동일한 DID 주체를 참조하는 경우 사용된다

- 선택 사항으로, 해당 속성이 존재하는 경우 값은 각 항목이 RFC 3986 [5]을 따르는 URI 집합이어야 하며, 이 관계는 주체가 하나 이상의 다른 식별자에 의해서도 식별될 수 있다

(그림2-18) alsoKnownAs 속성 예시

4. 검증 방법 관련 속성

ㅇ DID 문서는 DID 주체 또는 관련 당사자와의 상호작용을 인증하거나 승인하는 데 사용할 수 있는 암호화 Public Key와 같은 검증 방법을 표현할 수 있으며, 검증 방법은 많은 매개변수를 취할 수 있다

- **(verificationMethod)** 선택 사항으로, 해당 속성이 존재하는 경우 값은 검증 방법 집합이어야 하고, 각 검증 방법은 id, type, controller 및 특정 검증 자료 속성이 포함된 맵을 사용하여 표현된다

(그림2-19) verificationMethod 속성 예시

5. 검증 자료 관련 속성

ㅇ 검증 자료는 검증 방법을 적용하는 프로세스에서 사용하는 모든 정보이며 type 속성은 이러한 프로세스와의 호환성을 확인하는데 사용될 수 있다. 아래에서 설명된 검증 자료 관련 속성은 선택사항이다

- (publicKeyJwk) 해당 속성이 존재하는 경우 값은 JSON Web Key (JWK)를 나타내는 맵이어야 한다. Public Key를 나타내는 JWK를 사용하는 검증 방법은 kid 속성값을 프래그먼트 식별자로 사용하며 JWK 값은 Public Key 지문으로 설정하는 것이 권장된다

(그림2-20) publicKeyJwk를 사용하는 검증 방법 예시

- (publicKeyMultibase) 해당 속성이 존재하는 경우 값은 인코딩된 Public Key의 문자열 표현이어야 한다

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/jws-2020/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  ...
  "verificationMethod": [{
    "id": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
    "type": "JsonWebKey2020", // external (property value)
    "controller": "did:example:123",
    "publicKeyJwk": {
      "crv": "Ed25519", // external (property name)
      "x": "VCpo2LMLhn6iWku8MKvSLg2ZAoC-nlOyPVQaO3FxVeQ", // external (property name)
      "kty": "OKP", // external (property name)
      "kid": "_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A" // external (property name)
    }
  }, {
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020", // external (property value)
    "controller": "did:example:pqrstuvwxyz0987654321",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  ...
}
```

(그림2-21) publicKeyJwk와 publicKeyMultibase를 사용하는 검증 방법 예시

6. 검증 방법으로의 참조

o 검증 관계는 다양한 검증 관계와 관련된 속성에 포함되거나 참조될 수 있으며, 검증 방법을 참조하는 경우 여러 검증 관계에서 사용할 수 있다

o 검증 방법 속성값이 맵이면 검증 방법이 내장되어 있음을 의미하며 해당 속성에 직접 액세스할 수 있다

o 검증 방법 속성값이 URL 문자열인 경우, 검증 방법이 참조로 포함되어 있음을 의미하며 DID 문서의 다른 곳이나 다른 DID 문서에서 검색해야 한다. 해당 검색은 URL을 Dereferencing하고 결과 리소스

에서 URL과 값이 일치하는 id 속성이 있는 검증 방법 맵을 검색하는 방식으로 수행된다

```
...

 "authentication": [
   // this key is referenced and might be used by
   // more than one verification relationship
   "did:example:123456789abcdefghi#keys-1",
   // this key is embedded and may *only* be used for authentication
   {
     "id": "did:example:123456789abcdefghi#keys-2",
     "type": "Ed25519VerificationKey2020", // external (property value)
     "controller": "did:example:123456789abcdefghi",
     "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
   }
 ],

...
```

(그림2-22) 검증 방법 내장 및 참조 예시

7. 검증 관계 관련 속성

ㅇ DID 주체와 검증 방법 간의 관계를 표현하며 이를 통해 연관된 검증 방법을 다양한 목적으로 사용할 수 있다

ㅇ DID 문서의 적절한 검증 관계 속성에 사용된 검증 방법이 포함되어 있는지를 확인하여 검증 시도의 유효성을 확인하는 것은 검증자의 몫이 된다

ㅇ DID 주체와 검증 방법 간 검증 관계는 DID 문서에 명시되어 있어야 하고, 연관 없는 검증 방법은 해당 관계에 사용될 수 없으며 이때는 keyAgreement 속성값을 사용하여야 한다

ㅇ DID 문서는 검증 관계를 사용하여 해지된 키를 표현하지 않으며 참조된 검증 방법이 그것을 Dereferencing 하는데 사용된 최신 DID

문서에 존재하지 않는 경우, 그 검증 방법은 무효하거나 해지된 것으로 간주된다. 각 DID Method 사양은 해지가 수행되고 추적되는 방법을 자세히 설명해야 한다

ㅇ 아래 설명은 여러 가지 유용한 검증 관계를 정의하며 DID는 특정 검증 관계를 표현하기 위해 이를 포함할 수 있고 DID-SPEC 문서 [7]에 등록될 필요가 존재한다

ㅇ **(authentication)** DID 주체가 웹사이트에 로그인하거나 모든 종류의 Challenge-response 프로토콜에 참여하는 등의 목적으로 인증되는 방식을 지정하는 데 사용된다

- 인증이 수립되면 DID Method 또는 다른 애플리케이션이 해당 정보로 무엇을 수행할지 결정하며, 이때 다른 DID Method는 인증에 사용된 것과 다른 키 또는 완전히 다른 검증 방법을 제시하여 DID 문서를 업데이트하거나 삭제해야 할 수 있다
- 인증 확인 후에 수행되는 작업은 데이터 모델의 범위를 벗어나며, DID Method와 애플리케이션은 이를 직접 정의해야 한다
- 인증을 시도하는 엔티티가 실제로 유효한 인증 증명을 제시하는지 확인해야 하는 모든 검증자에게 유용하며, 인증 목적으로 만들어진 증명을 포함하는 검증자는 엔티티가 DID에 의해 식별된다고 말하는 일부 데이터를 수신하면 DID 문서에 authentication 속성의 내용으로 나열된 검증 방법을 사용하여 증명을 검증할 수 있는지 확인한다
- DID 문서의 authentication 속성에 표시된 검증 방법은 DID 주체를 인증하는 데만 사용할 수 있으며, 다른 DID Controller를 인증하려는 경우, controller 값과 연결된 엔티티는 자신의 DID 문서와 관련된 인증 검증 관계를 통해 인증받아야 한다

(그림2-23) authentication 속성 예시

○ **(assertionMethod)** 검증 가능한 자격 증명을 발급하는 목적과 같이 DID 주체가 클레임을 표현하는 방법을 지정하는 데 사용된다

- 검증자가 자격 증명을 처리하는 동안 유용하게 사용될 수 있으며, 검증하는 동안 DID 주체가 만든 증명이 포함되어 있는지 확인하기 위해 사용된 검증 방법이 해당 DID 문서의 assertionMethod 속성과 연결되어 있는지 확인한다

(그림2-24) assertionMethod 속성 예시

○ (keyAgreement) DID 주체를 대상으로 하는 기밀 정보를 전송하기 위해 엔티티가 암호화 자료를 생성하는 방법을 지정하거나 수신자와 안전한 통신 채널을 설정하려는 목적으로 사용된다

- DID 주체를 위한 메시지를 암호화할 때 유용하게 사용될 수 있으며, 상대방은 검증 방법에서 암호화 Public Key 정보를 사용하여 수신자를 위한 복호화 키를 래핑한다

(그림2-25) keyAgreement 속성 예시

○ (capabilityInvocation) DID 주체가 암호화 기능을 호출하는 데 사용할 수 있는 검증 방법을 지정하는 데 사용된다

- HTTP API를 제공하는 서버의 기능 검증자 역할을 수행하며, 호출된 기능이 참조하는 검증 방법이 DID 문서의 capabilityInvocation 속성에 있는지 확인해야 한다
- 수행 중인 작업이 유효하고 기능이 액세스 중인 리소스에 적합한지 확인하기 위해 검사가 이루어지며, 검증이 성공하면 서버는 호출자가 보호된 리소스에 액세스할 권한이 있음을 암호화하여 확인한다

○ (capabilityDelegation) DID 주체가 암호화 기능을 다른 당사자에게 위임하는 데 사용할 수 있는 메커니즘을 지정하는 데 사용되며, DID Controller가 보호된 HTTP API에 액세스하는 기능을 다른 당사자에게 위임하기로 선택할 때 유용하게 사용될 수 있다

- 위임을 위해 DID 주체는 다른 주체에게 암호화 방식으로 서명을 진행한다

(그림2-26) capabilityInvocation 속성 예시

(그림2-27) capabilityDelegation 속성 예시

ㅇ 위의 5가지 속성은 모두 선택 사항이며, 존재하는 경우 연관된 값은 하나 이상의 검증 방법 세트여야 하고 각 검증 방법은 포함되거나 참조될 수 있다

8. 서비스 관련 속성

ㅇ 서비스는 DID 문서에서 DID 주체 또는 관련 엔티티와 통신하는 방법을 표현하는 데 사용된다. 서비스는 DID 주체가 다른 곳에 알리려고 하는 모든 유형의 서비스일 수 있고 추가 발견, 인증, 권한 부여 또는 상호작용을 위한 분산형 ID 관리 서비스가 포함된다

ㅇ 개인 정보 보호 문제로 인해 소셜 미디어 계정, 개인 웹사이트, 이메일 주소와 같은 서비스를 통해 정보를 공개하는 것은 권장되지 않으며, 서비스 관련 정보는 종종 서비스별로 상이하다

ㅇ 서비스 속성 자체는 선택 사항이지만, 존재하는 경우 하위의 id, type, serviceEndpoint 속성은 반드시 존재해야 한다

- **(service)** 선택 사항으로, 존재하는 경우 연관된 값은 각 서비스가 맵으로 설명되는 서비스 세트여야 한다. 각 서비스 확장은 추가 속성을 포함할 수 있고, 확장에 관한 속성을 추가로 제한할 수 있다
- **(serviceEndpoint)** service 속성이 존재할 때 반드시 하위 항목으로 존재해야 한다. 속성값은 문자열, 맵 또는 하나 이상의 문자열 혹은 맵으로 구성된 집합이여야 하고, 모든 문자열 값은 RFC 3986 [5]을 준수해야 한다. 이 표준의 정규화 및 비교 규칙과 해당 URI 체계 사양의 모든 정규화 규칙에 따라 정규화된 유효한 URI여야 한다

(그림2-28) service 및 serviceEndpoint 속성 예시

9. Representations

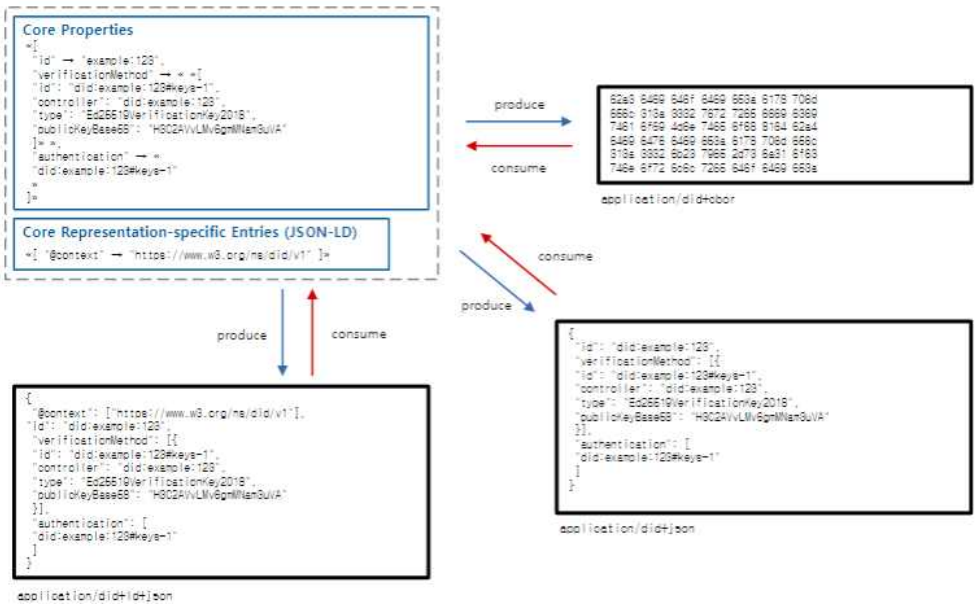
ㅇ DID 문서의 구체적 직렬화를 표현 (Representation)이라고 일컫는다. 표현은 생산 (Production) 프로세스를 통해 데이터 모델을 직렬화하여 생성된다

ㅇ 표현은 소비 (Consumption) 프로세스를 통해 데이터 모델로 변환되며, 생산 및 소비 프로세스를 통해 정보를 특정 표현에서 다른 표현으로 변환할 수 있다

ㅇ 주로 JSON, JSON-LD에 대한 표현이 사용되며, 구현자는 데이터 모델을 표현할 수 있는 XML, YAML과 같은 다른 표현을 사용할 수 있다

ㅇ 모든 표현에 대한 요구사항은 다음과 같다

- 데이터 모델 표준에서 명시하는 모든 데이터 유형에 대해서 결정론적 생산 및 소비 규칙을 정의해야 한다
- IANA에 등록된 미디어 유형과 고유하게 연결되어야 한다
- 표현은 표준의 fragment 부분에서 정의된 처리 규칙을 준수하는 미디어 유형에 대한 fragment 처리 규칙을 정의해야 한다
- 표현은 데이터 모델 데이터 유형의 어휘 표현을 사용해야 하며, 데이터 모델로의 소비 프로세스가 손실 없는 한, 다른 어휘 직렬화를 사용하여 데이터 유형을 직렬화 할 수 있도록 선택할 수 있다
- 표현은 생산 및 소비 프로세스 동안 사용하기 위해 표현 별 항목 맵을 정의할 수 있으며, 해당 항목을 통해 손실 없는 변환을 보장한다
- 상호 운용성을 극대화하기 위해 표현 사양 작성자는 DID-SPEC [8]에 자신의 표현을 등록해야 한다



(그림2-29) 표현의 생산 및 소비 프로세스

제 5 절 DID method Registry

1. Data method Registry

- DID-extensions-methods 문서 [9] 내 현재 개발 중인 DID Method 사양이 요약되어 있으며, 메소드 관련 링크는 후속 구현자 초안이 생성됨에 따라 업데이트된다

- 2024년 11월 27일 기준, 최신 DID-extensions-methods 문서에 등록되어 있는 DID Method의 개수는 약 203가지가 존재한다

DID Method	Registry	Contact
3	Ceramic Network	3Box Labs (email) (website)
abt	ABT Network	ArcBlock
aergo	Aergo	Blocko (website)
ala	Alastria	Alastria National Blockchain Ecosystem
amo	AMO blockchain mainnet	AMO Labs (website)
antelope	Antelope	Tonomy Foundation (website)
art	Artwork ID Method	Ming-lam Ng (RealMatter) (email) (web
asset	Ledger-independent generative DID method based on CAIP-19 identifiers	BOTLabs GmbH (email) (website)
bba	Ardor	Attila Aldemir (email)
bee	Ledger agnostic	mesur.io (email) (website)
bid	bif	teleinfo caict
bit	d.id	d.id (email) (website)
bluetoqueagent	Trusted Digital Web	Hyperonomy Digital Identity Lab, Parall Corporation (email) (website)
bluetoquedeed	Trusted Digital Web	Hyperonomy Digital Identity Lab, Parall Corporation (email) (website)
bluetoquenfe	Trusted Digital Web	Hyperonomy Digital Identity Lab, Parall Corporation (email) (website)
bluetoqueproc	Trusted Digital Web	Hyperonomy Digital Identity Lab, Parall Corporation (email) (website)
bnb	Binance Smart Chain	Ontology Foundation
bryk	bryk	Marcos Allende, Sandra Murcia, Flavia M Ruben Cessa
btco	BTC	Aviary Tech (email) (website)
btcr	Bitcoin	Christopher Allen, Ryan Grant, Kim Harr
candid	Quorum	Eric Lin (email) (website)
ccd	Concordium	Concordium development team (email)
ccf	Confidential Consortium Framework (CCF)	Microsoft (email) (website)
ccp	Quorum	Baidu, Inc.
celo	Celo	Ontology Foundation
cheqd	cheqd	CHEQD FOUNDATION LIMITED (email)
com	commercio.network	Commercio Consortium
content		Mizuki Sonoko (email) (website)
corda	Corda	Nitesh Solanki,Moritz Platt,Pranav Kirtar
cosmos	Cosmos application chains	Shaun Conway (email) (website)
cot	CoTChain	CoTNetwork, Inc. (website)
cr	Hyperledger Fabric	(email)
ct	Circular Trust	(email) (website)
		Beijing Zhongdun Anxin Science and Te

(그림2-30) DID method Registry의 일부

2. 등록 절차

ㅇ DID-CORE [5]사양이 Use case를 완전히 처리할 수 없기 때문에, Use case를 전 세계적으로 상호 운용할 수 있는 방식으로 달성하기 위해 DID-SPEC [7] 레지스트리에 새로운 매개변수, 속성 또는 값을 추가해야 할 수 있다

ㅇ 새로운 매개변수, 속성 또는 값을 추가하려는 경우 구현자는 DID-SPEC 레지스트리가 호스팅되는 Repository에서 Full Request로 DID-SPEC 레지스트리에 대한 수정 요청을 제출해야 하며, 아래 정책을 준수해야 한다

- DID-CORE 레지스트리에 추가되는 모든 내용은 반드시 사람이 읽을 수 있는 설명으로 명시되어야 한다
- 속성, 매개변수의 이름이나 값은 반드시 해당 기능을 나타내야 하며, “foo”와 같은 일반적인 용어를 피해야 한다
- 모든 메소드 이름에서 일반적인 용어는 피해야 한다
- 저작권, 상표 또는 지적재산권에 대한 우려가 있는 경우, 추가 및 사용은 F/RAND 라이센스에 따라 지적재산권 보유자가 서면으로 허가해야 한다
- 추가된 내용은 다른 사람에게 직접적인 피해를 줄 수 있는 불합리한 법적, 보안적, 도덕적 또는 개인정보보호 문제를 야기해서는 안된다
- DID-CORE 레지스트리에 추가되는 모든 내용은 구현자가 속성을 구현할 수 있도록 최소한 URL을 통해 정의 사양에 연결되어야 한다
- DID-CORE 레지스트리에 속성이나 값을 추가하는 경우 추가 사항에 대한 기계에서 읽을 수 있는 JSON-LD 컨텍스트를 지정해야 한다
- JSON-LD 컨텍스트는 제출물의 일부로 포함되어야 하며, 소비자 구현이 URI를 전체 컨텍스트에 일관되게 매핑할 수 있도록 JSON-LD 컨텍스트에 대한 네임스페이스 URI를 제공해야 한다
- 제공된 URI는 반드시 영구적이어야 하며, 모든 용어를 연관된 사람이 읽을 수 있는 설명에 연결해야 한다
- JSON-LD 컨텍스트는 버전이 지정되어야 하며, 날짜가 표시되어서는 안된다
- JSON-LD 컨텍스트는 범위가 지정된 용어를 사용해야 하며, 용어 충돌 가능성을 제거하는 @protected 기능을 사용해야 한다
- DID-CORE 레지스트리의 속성은 제거해서는 안되며, 더 이상 사용되지 않도록 설정해야 한다

ㅇ DID-SPEC 레지스트리 편집자는 레지스트리에 추가된 내용을 검토할 때, 위의 모든 정책을 고려해야 하고 위반하는 경우 레지스트리 항목을 거부해야 한다

제 6 절 DID Use cases and Requirements

6절에서는 DID의 사용사례와 요구사항에 대해서 다루며, W3C에서는 해당 부분을 DID-USE-CASES 문서 [6]에 정의하였다

1. Required Attributes

ㅇ DID의 요구사항은 다음과 같다
- **(분산화)** 중앙 발급 기관이 없어야 한다
- **(지속성)** 식별자는 본질적으로 지속적이어야 하며, 하위 주직의 지속적인 운영을 요구하지 않아야 한다
- **(암호학적 검증 가능)** 식별자의 제어를 암호학적으로 증명할 수 있어야 한다
- **(확인 가능)** 식별자에 대한 메타데이터를 발견할 수 있어야 한다

ㅇ 기존 식별자가 이러한 특성 중 일부를 표시할 수는 있지만, 네 가지 특성 모두를 표시하는 식별자는 존재하지 않는다

2. DID System

ㅇ DID System은 Controller (통제자), RP (요청 당사자), 주체 (subject)라는 세 개의 개별 개체가 주요 작업을 수행한다
- **(Controller)** DID를 생성하고 통제하는 역할을 수행한다
- **(RP)** DID 주체와 관련된 상호작용을 위한 식별자로, DID에 의존한다
- **(주체)** DID에서 지칭하는 개체로 사람, 조직, 기기, 위치 혹은 개념일 수 있으며, Controller이기도 하지만 보호자, 에이전트 (사람 또는 소프트웨어), 무생물은 주체가 될 수 없고 기능적 역할을 수행하지 않는다

- DID Controller와 RP는 개인이거나 상호작용하는 시스템일 수 있지만, 후술될 내용에서는 단순화를 위해 두 당사자를 모두 작업을 수행하는 개인인 것처럼 지칭된다

- DID 통제는 Controller만이 수행할 수 있지만, DID Controller를 포함하여 자신이 알고 있는 모든 DID에 대한 RP 역할을 하여 자신의 DID를 검사 및 검증하고자 하는 사람은 누구나 될 수 있다

- DID의 사용 사례를 다루는 DID-USE-CASES 문서 [6]에서는 최종 시스템의 관점에서 사례들을 다루고 있으며, “ID 공급자”가 없다는 점을 DID의 요점으로 바라보고, Controller가 DID를 관리하고 RP가 DID를 적용하는 데 사용하는 분산 시스템에 이러한 요점이 포함된다

- 분산 시스템인 DID 레지스트리는 특정 서비스 공급자와 독립적으로 작동할 수 있게 설계되었기에 주어진 플랫폼 권한과 무관하며, DID는 분산 원장 기술 (DLT)을 사용하여 등록된다

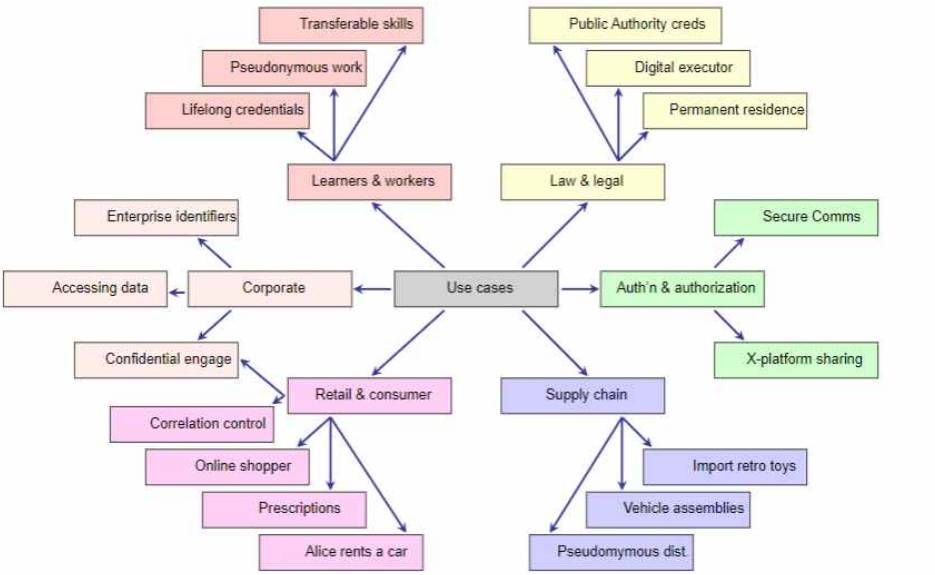
- 현재 모든 분산 시스템의 정의와 운영은 중앙 제어의 일부 요소를 유지하며 평가기준에 따라 시스템이 중앙 집중화 및 분산화 스펙트럼에 속하는 위치가 상이하다

3. Use Cases

- 그림 2-31은 Use Case를 표현한 Map이다. 사용 사례는 크게 6가지 항목으로 나눌 수 있고, 하위 항목을 통해 사례를 세분화하여 나타낸다

- 기업에서는 DID를 통한 안전한 고객 데이터 관리 및 엔티티의 마스터 데이터 접근을 가능하게 할 수 있다

- 엔티티의 권한 있는 마스터 데이터 레코드를 통해 DID 문서에 명시된 서비스 엔드 포인트를 찾게 하며, 자체 인증 또는 제3자의 Verifiable Presentation (VP)로 검증된다
- 장치의 경우, 제조업체가 사업을 중단하더라도 DID는 해당 장치의 마스터 데이터에 대한 지속적인 접근을 보장한다



(그림2-31) Use Case Map

o 교육이나 근로 분야에서는 경력에 대한, 또는 직장 내에서의 포트폴리오나 자격 증명을 관리하도록 하여 외부에서도 증명에 사용할 수 있게 한다
- Lifelong Credentials 개념을 통해 교육용 VC를 구현하여 키 회전과 자격 증명 유지를 편리하게 하여 유연하게 활용될 수 있는 방법을 제공한다

o 리테일과 소비자 환경에서는 다음과 같은 사례가 존재한다
- 온라인 쇼핑 시 DID를 통해 카드의 출처를 확인하거나 상품의 정품

여부를 확인하는 등 신뢰할 수 있는 거래를 제공할 수 있다
- 상관관계 제어 서비스를 구축하여 사용자의 프라이버시를 보호할 수 있다
- SSI 기술을 활용한 차량 렌트 서비스 등의 활용이 가능하나, 이 사례에서는 다양한 서비스 간의 상호운용성을 제공하기 위해 기술 표준과 프로토콜을 인간이 이해할 수 있는 방식으로 결합하는 것이 도전과제로 남아있다

o 법률적 환경에서는 이민자의 영주권 발급이나 공공 기관 신원 증명서 등의 행정 절차에서의 활용이 가능하며, 개인이 사망했을 때, 이후 자산 관리 권한을 제3자에게 안전하게 이전할 수 있는 '디지털 집행자' 개념 또한 제공할 수 있다
- 권한 이전의 경우에는 디지털 집행자의 변경이 지연될 때 추가로 발생할 수 있는 문제가 있다는 것을 고려해야 한다

o 차량의 유지보수 기록 관리, 가명성을 통한 민감한 정보 보호, 수입품의 제조 및 유통 관리 등의 다양한 분야에서의 DID 적용으로 공급망의 복잡성을 관리하고, 각 단계에서 데이터 보안을 강화할 수 있다

o 사용자가 서비스마다 추적 권한을 부여할 수 있게 하여 플랫폼 간의 정보 공유의 주권을 스스로 소유할 수 있다
- 탈중앙화된 메시징 아키텍처를 통해 사용자에게 더 높은 수준의 보안과 유연성을 제공할 수 있고, 중앙 기관에 의존하지 않는 안전한 커뮤니케이션을 가능하게 한다

제 3 장 DID Method

제 1 절 DID Method 개요

1. DID Method

ㅇ DID Method는 W3C DID 표준 사양에서 설명하는 기능을 어떻게 구현할 수 있을지를 정의하며, 특정 검증 가능한 데이터 저장소와 연관된다. 새로운 DID Method는 다양한 구현 간 상호운용성을 보장하기 위해 자체 사양으로 정의된다

ㅇ DID Method 사양은 특정 DID 체계를 정의하는 것 외에도 특정 유형의 검증 가능한 데이터 저장소를 사용하여 DID와 DID 문서를 생성, 확인, 업데이트 및 비활성화하기 위한 메커니즘 또한 정의한다. 더불어 DID에 관한 모든 구현 고려 사항과 보안 및 개인 정보 보호 고려 사항도 문서화한다

2. Method Syntax/Operations

ㅇ DID syntax를 정의할 때 모든 DID Method 사양에 대한 요구 사항은 다음과 같다

- DID Method 사양은 정확히 하나의 Method 마다의 DID Scheme를 정의해야 하고, 이는 DID Syntax 규칙에서 명시된 method-name 규칙에 의해 하나의 Method 이름으로 식별되어야 한다
- DID Method 사양은 DID의 method-specific-id 구성 요소를 생성하는 방법을 명시해야 한다
- DID Method 사양은 method-specific-id의 민감도와 정규화를 정의해

야 한다

- method-specific-id은 DID Method 내에서 고유해야 하며, 전역적으로 고유할 수 있다
- DID Method에 의해 생성된 모든 DID는 전역적으로 고유해야 한다
- Method name 충돌 가능성을 줄이기 위해, DID Method 사양은 DID-SPEC-REGISTRIES [7]에 등록되어 있어야 한다
- DID Method는 여러 method-specific-id 형식을 정의할 수 있다
- method-specific-id 형식은 콜론을 포함할 수 있고, 이는 ABNF Rules를 따라 정의된 문법을 준수해야 한다
- DID Method 사양은 DID Path에 대한 ABNF Rules를 지정할 수 있고, 이는 Syntax 상의 Path 규칙보다 더 제한적일 수 있다
- DID Method 사양은 DID Fragment와 DID Query에 대한 ABNF Rules를 지정할 수 있고, 이는 일반 규칙보다 더 제한적일 수 있다

ㅇ DID Method 사양에서 Method 작업을 정의할 때 모든 요구 사항은 다음과 같다

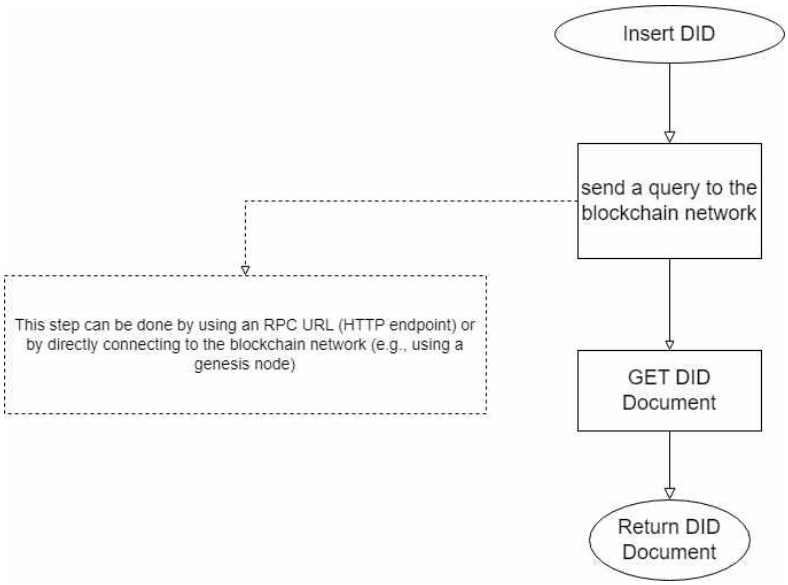
- DID Method 사양은 필요한 암호화 프로세스를 포함하여 모든 작업을 실행하기 위한 권한 부여 방법을 정의해야 한다
- DID Method 사양은 DID Controller가 DID 및 DID 문서를 생성하는 방법을 지정해야 한다
- DID Method 사양은 DID Resolver가 DID를 사용하여 DID 문서를 확인하는 방법, 특히 DID Resolver가 응답의 진위성을 확인하는 방법을 지정해야 한다
- DID Method 사양에서는 DID 문서 업데이트에 대한 구성 요소와 DID Controller가 DID 문서를 업데이트하는 방법 또는 업데이트가 불가능하다는 것을 명시해야 한다
- DID Method 사양에서는 DID Controller가 DID를 비활성화하는 방법을 지정하거나 비활성화가 불가능하다는 것을 명시해야 한다

제 2 절 블록체인 네트워크 기반 DID Method

1. 개요

○ 블록체인 네트워크를 신뢰기반으로 하는 DID Method들은 이미 구성되어 있는 블록체인 네트워크에 DID 문서를 저장함으로써 내부에 있는 정보에 대한 신뢰기반을 구축한다

○ 일반적으로 개인 기기의 지갑을 통해 네트워크에 접속하거나 RPC URL로 간접적으로 접근 혹은 해당 블록체인 네트워크에서 지정한 신뢰할 수 있는 노드들을 통해 직접 접속하여 DID 문서를 확보하는 방식을 사용한다



(그림3-1) 블록체인 네트워크 기반 DID Method의 DID 문서 획득 절차

○ 표 3-1과 같이, 신뢰 기반으로 활용되는 블록체인 네트워크의 종류에 따라 다양하게 존재한다

[표3-1] 블록체인 기반 DID Methods

Method	내용
did:ethr	이더리움과 호환되는 모든 블록체인에 사용 가능한 DID Method로, Smart Contract를 사용하여 해석을 수행한다
did:indy	Hyperledger Indy를 기반으로, 해당 규칙을 준수하는 모든 DID에 대해 해석이 가능하다
did:algo	Algorand라는 DLT를 기반으로 동작하는 DID Method로, 누구나 식별자를 생성할 수 있으며 제 3자가 다른 사람을 위해 DID를 발급할 수 있다
did:dyne	Dyne.org에서 제공하는 DID Method로 서버에 키가 없는 end-to-end 암호화를 제공하며, 파일 시스템 기반의 저장소를 사용한다
did:ebsi	European Blockchain Services Infrastructure(EBSI)를 준수하는 DID Registry를 저장소로 활용하며, 자체 api를 통해 질의하는 방식으로 활용할 수 있다
did:iota	Distributed Ledger 기술 중 Unspent Transaction Ouput (UTXO) 기반인 IOTA에 의해 생성된 DID 문서에 해석을 수행하는 DID Method이다
did:kscirc	k4security의 kschain을 기반으로 하는 DID Method. k를 접두사로 붙여 식별자로 활용한다
did:v1	Veres One 블록체인을 위한 DID Method로, ledger를 사용할 필요가 없는 Cryptonym 방식과 metadata를 ledger에 등록하는 UUID 방식, 두 가지 종류의 did 타입을 제공한다

ㅇ 본 절에서는 다양한 블록체인 네트워크 기반 DID Method 중, 대표적으로 사용되는 did:ethr 및 did:indy에 대해서 기술한다

2. did:ethr

ㅇ 이더리움과 호환되는 모든 블록체인에 사용가능한 DID Method 이다

ㅇ did:ethr의 식별자 구조는 아래와 같다

```
ethr-did = “did:ethr:” ethr-specific-identifier
ethr-specific-identifier = [ethr-network “:” ] ethreum-address / public-key -hex
ethr-network = “mainnet” / “goerli” / network-chain-id
network-chain-id = “0x” *HEXDIG
ethereum-address = “0x” 40*HEXDIG
public-key-hex = “0x” 66*HEXDIG
```

(그림3-2) did:ethr Identifier 구조(ABNF rule)

ㅇ did:ethr가 생성-해석-업데이트-비활성화되는 일련의 CRUD 절차는 아래와 같이 이루어진다

ㅇ **(Create)** 이더리움의 주소가 생성되었을 경우, 암묵적으로 did:ethr의 생성이 이루어진다. 이더리움 주소 “0xf3be...” 의 식별자 “did:ethr:0xf3be..” 의 DID 문서는 다음과 같다

```
{
  “@context“: [
    “https://www.w3.org/ns/did/v1“,
    “https://w3id.org/security/suites/secp256k1recovery-2020/v2“
  ],
  “id“: “did:ethr:0xf3be...“,
  “verificationMethod“: [
```

```
    {
      “id“: “did:ethr:0xf3be...#controller“,
      “type“: “EcdsaSecp256k1RecoveryMethod2020“,
      “controller“: “did:ethr:0xf3be...“,
      “blockchainAccountId“: “eip155:1:0xf3be...“
    }
  ],
  “authentication“: [
    “did:ethr:0xf3be...#controller“
  ],
  “assertionMethod“: [
    “did:ethr:0xf3be...#controller“
  ]
}
```

(그림3-3) did:ethr DID 문서 예시

○ **(Read)** 해석은 이더리움에 작성된 Smart Contract 내부 함수를 사용하여 이루어지며, 별도의 자원을 소모하지 않는다

- 이더리움 주소를 기반으로 한 식별자는 verificationMethod의 type 속성에 “ecdsaSecp256k1RecoveryMethod2020”, blockchainAccountId에는 CAIP10 형식에 따라 표시된 이더리움 주소가 들어가게 된다
- id는 식별자와 동일하며, controller 속성 역시 기본 구성은 식별자와 같다
- did:ethr의 DID는 Smart Contract의 identityOwner (address identity) 함수로 확인되는 controller address를 가지고 있는데, 이에 #controller를 붙여 DID 문서에 활용한다. 이 controller address는 DID 문서의 authentication과 assertionMethod 배열에 작성된다

```
"verificationMethod": [
  {
    "id": "did:ethr:0xf3be...#controller",
    "type": "EcdsaSecp256k1RecoveryMethod2020",
    "controller": "did:ethr:0xf3be...",
    "blockchainAccountId": "eip155:1:0xf3be..."
  }
],
```

(그림3-4) did:ethr verificationMethod 구성 예시

```
"authentication": [
  "did:ethr:0xf3be...#controller"
],
"assertionMethod": [
  "did:ethr:0xf3be...#controller"
]
```

(그림3-5) did:ethr controller address 활용 예시

○ **(Update)** 업데이트는 식별자를 제외한 계정 소유자 변경, 위임자 추가, 속성 (attribute) 추가가 있다. 위 업데이트 항목들은 이더리움의 Smart Contract를 통해 DID 문서에 동적으로 반영되며, 위임 키 및 추가 속성 등은 유효기간이 지나면 자동으로 만료되어 사라지게 된다

○ **(Deactivate)** 비활성화는 식별자의 소유자 주소를 "0x0"으로 설정하여 무효화시키는 방식으로 이루어진다. 블록체인 네트워크의 특성상 삭제는 불가능하다

3. did:indy

ㅇ W3C Credentials Community Group에서 발표한 DID 규격을 준수하는 DID Method로, Hyperledger Indy를 기본으로 하여 해당 규칙을 준수하는 모든 DID에 대하여 해석이 가능하다

ㅇ did:indy의 식별자 구조는 아래와 같다

```
did:indy: [namespace] : [namespace-identifier]
```

(그림3-6) did:indy Identifier 구조

- namespace는 indy ledger를 명시하며, 뒤에 “sovrin”, “sovrin:staging” 과 같이 보조 ledger에 대한 이름을 가질 수 있다
- namespace-identifier는 주어진 namespace에서 유일한 식별자로, Public Key에 해당하며 이는 indy 내부 NYM의 verkey 값을 SHA256 해시 함수를 통과시킨 후 나온 값의 가장 처음 16byte를 다시 base58로 인코딩 한 값이다

```
# Sovrin MainNet에 작성된 DID
- did:indy:sovrin:7Tgg6Bw...
# Sovrin StagingNet에 작성된 DID
- did:indy:sovrin:staging:6cgbu8Z...
# ID Union Test ledger에 작성된 DID
- did:indy:idunion:test:2MZYuP...
```

(그림3-7) did:indy 구조 예시

ㅇ did:indy가 생성-해석-업데이트-비활성화되는 일련의 CRUD 절차는 아래와 같이 이루어진다

ㅇ **(Create)** did:indy의 DID 생성은 주어진 Indy 네트워크에서 NYM ledger transaction을 실행함으로써 수행된다

- Indy NYM transaction은 식별자 dest, ED25519 검증 키 (verkey), JSON으로 이루어진 diddocContent (선택사항), NYM transaction의 버전을 나타내는 version (선택사항)으로 구성된다
- NYM은 Verinym의 준말로써, 데이터 객체에 대한 Hyperledger Indy의 특정 용어를 나타낸다. NYM은 주어진 namespace를 가진 특정 Indy 네트워크에 기록되게 된다

1) NYM의 verkey가 null이면 DID 비활성화로 간주, 조립 종료
2) 네트워크의 namespace, NYM, dest, NYM verkey를 이용해 기본 DIDDoc 템플릿 생성, diddocContent가 없으면 조립 완료
3) diddocContent가 있는 경우 id 항목이 존재해서는 안되며, 기존 DIDDoc과 중복되는 id 값 역시 존재해서는 안된다
4) verificationMethod와 authentication은 배열 형태여야 한다

(그림3-8) did:indy DID 문서 조립과정

- 생성된 DID 문서 예시는 아래와 같다

```
{
  “id“: “did:indy:<namespace>:<dest>“,
  “verificationMethod“: [{
       “id“: “did:indy:<namespace>:<dest>#verkey“,
       “type“: “Ed25519VerificationKey2018“,
       “publicKeyBase58“: “<verkey>“,
       “controller“: “did:indy:<namespace>:<dest>“
    }
  ],
  “authentication“: [
    “did:indy:<namespace>:<dest>#verkey“
```

```
  ]
}
```

(그림3-9) did:indy DID 문서 예시

○ **(Read)** DID를 보유하고 있는 indy의 Ledger Instance를 찾아 연결하고, 이 DID와 연관된 NYM을 검색 후, 반환된 NYM에 대한 상태 증명을 확인한 다음 DID 문서를 조립해야 한다. Read 과정은 다음 단계를 거쳐 수행된다

1) did:indy: [namespace] : [dest] 에서 namespace를 추출한다
2) resolver가 알 수 있는 namespace라면 연결하고, 아니라면 Indy의 instance에 대한 genesis파일을 가져와서 ledger에 연결해야 한다. 만약 이 과정 또한 원활하게 수행되지 않는 경우 Not Found를 반환하고, 클라이언트 측에서 연결하지 않기로 결정했다면 Not Authorized 상태를 반환한다
3) 연결되었다면 GET_NYM Indy 요청을 보내고 namespace에 대한 식별자를 전달한다
4) 찾은 DID 문서에 대해 DIDDoc assembly process를 적용하여 DID 문서를 조립한다

(그림3-10) did:indy Read 프로세스

- did:indy는 Genesis 파일을 참조해 초기 노드인 Genesis 노드에 질의하여 DID 및 verkey를 부여받고 이를 기반으로 네트워크에 질의하여 DID Resolution 과정을 수행한다
- 각 질의의 타입 및 번호는 Indy-node 문서 [10]에 명시되어 있으며, DID 문서를 읽을 때 수행하는 질의는 GET_NYM과 GET_ATTR 이다

○ **(Update)** Indy DID Method를 이용한 DID 갱신은 NYM의 Controller (소유자)가 동일한 식별자 (dest)를 사용하여 NYM 거래를 수행할 때

발생한다. DID 및 dest 값은 변경되지 않으며, DID 문서 내의 내용은 변경될 수 있다

- (Deactivate) NYM의 verkey 항목을 null로 바꿈으로써 비활성화가 수행된다

제 3 절 Web 기반 DID Methods

1. 개요

- 인터넷 상의 표준 Web 기술을 활용하여 DID를 구현한 방식으로, 기존의 인터넷 인프라를 활용할 수 있다는 장점이 존재한다

- 기존의 인터넷 인프라를 활용하여 높은 접근성과 호환성을 제공하며, 블록체인 네트워크에 의존하지 않고도 DID 생태계를 구축할 수 있다

- Domain Name System (DNS)를 통해 접근할 수 있어야 하며 이는 기존 Web 시스템과 호환성을 유지하는 중요한 요소이며, 대표적인 Web 기반 DID Method로 did:web, did:webs, did:dns가 존재한다

2. did:web

- Web 도메인을 사용하여 DID를 생성 및 관리하는 DID Method로, 반드시 접두사 did:web이 있어야 하며, 이 문자열은 반드시 소문자여야 한다

ㅇ 식별자는 DID 문서에 대한 경로를 가지는 TLS/SSL 인증서에 의해 보안이 유지되는 완전한 도메인 네임이며, 자세한 구조는 아래와 같다

- SL/TLS 인증서 내의 Common Name (CN) 과 일치해야 하고, IP 주소는 포함하면 안되지만 포트 포함은 가능하며, 경로의 충돌 방지를 위해 퍼센트 인코딩으로 포트를 표현해야 한다
- 디렉토리 및 하위 디렉토리 역시 포함 가능하나 “/” 대신 “:” 으로 구분된다

구조) did:web:domain-name(%3Aport-num)(:path) did:web:w3c-ccg.github.io did:web:w3c-ccg.github.io:user:alice did:web:example.com%3A3000

(그림3-11) did:web Identifier 예시

ㅇ 대부분의 Web 서버가 콘텐츠를 제공하는 방식을 따라, DID 문서는 주로 JSON 타입으로 제공된다. 자세한 핵심 요소 및 문서 관리 방법은 아래와 같다

- JSON 문서의 root에 @context가 있다면 이 문서는 JSON-LD 규칙에 따라 처리되어야 한다
- 이것이 불가능하거나 문서 처리에 실패한 경우, did:web 문서로 간주되지 않는다
- JSON 문서의 root에 @context가 있고 JSON-LD 처리를 통과하며, @context 내에 https://www.w3.org/ns/did/v1이 포함되어 있다면 DID 문서로 처리될 수 있다
- 해당 데이터의 mime type은 application/did+ld+json으로 명시되어야 한다
- @context가 없는 경우, DID 처리를 위한 일반 JSON 규칙을 통해 처

리되어야 하며, 해당 데이터의 mime type은 application/did+json 으로 명시되어야 한다

ㅇ did:web이 생성-해석-업데이트-비활성화되는 일련의 CRUD 절차는 아래와 같이 이루어진다

ㅇ **(Create)** did:web의 DID 생성과정은 아래와 같다

- (1) 도메인 네임 사용을 위해 도메인 네임 등록 기관에 이름 사용을 신청한다
- (2) Hosting Service의 위치 및 IP 주소를 DNS Lookup 서비스에 저장한다
- (3) Koblitz Curve와 같은 적절한 키 쌍을 포함한 JSON-LD 파일을 생성하고 전체 도메인을 대표하는 경우, did.json 파일을 domain-name/.well-known/ 아래 혹은 현재 도메인에서 여러 DID가 해석될 경우 지정된 경로 아래에 did.json파일을 저장한다

```
domain name: w3c-ccg.github.io
1. Creating DID
did:web:w3c-ccg.github.io
 -> https://w3c-ccg.github.io/.well-known/did.json

2. Creating DID with optional path
did:web:w3c-ccg.github.io:user:alice
 -> https://w3c-ccg.github.io/user/alice/did.json

3. Creating DID with optional path and port
did:web:example.com%3A3000:user:alice
 -> https://example.com:3000/user/alice/did.json
```

(그림3-12) did.json 접근 예시

ㅇ **(Read)** did:web에서 DID 문서를 해석하기 위해서 아래 절차들이 반드시 수행되어야 한다

- (1) did:web의 identifier 에서 “:” 를 “/”로 교체하여 정규화된 도메인 네임과 Path를 획득한다
- (2) 도메인에 port가 있다면 colon을 percent 디코딩한다
- (3) DID 문서의 예상 위치에 대한 HTTPS URL 을 생성하기 위해 https:// 를 추가한다
- (4) URL에 경로가 지정되지 않은 경우 /.well-known을 추가한다
- (5) URL의 완성을 위해 /did.json을 추가한다
- (6) 보안 요구사항을 준수하는 안전한 HTTPS 연결을 수행할 수 있는 Agent를 사용하여 해당 URL에 HTTP GET 요청을 수행한다
- (7) GET 요청 중 DNS Resolution 과정에서 클라이언트는 신원 추적 방지를 위해 DNSoverHTTPS (DoH)를 활용해야 한다

did:web:example.com%3A3000:user:alice 1. example.com%3A3000/user/alice 2. example.com:3000/user/alice 3. https://example.com:3000/user/alice 4. https://example.com:3000/user/alice/did.json

(그림3-13) Read 절차 예시

ㅇ Update

- DID 문서를 업데이트 하기 위해, did.json이 업데이트 되어야 한다
- DID 자체는 변경되지 않지만, DID 문서의 내용은 변경될 수 있다
- git과 같은 version control system 혹은 github actions와 같은 continuous integration system을 활용하여 did 문서 update를 관리하면 인

증 (authentication)과 감사 (audit) 기록에 대한 지원을 제공할 수 있으며, 해당 update 프로세스에 지정된 HTTP API는 없다

ㅇ Deactivate

- DID 문서의 삭제를 위해 did.json 파일을 제거하거나 다른 어떤 방식으로든 공개적으로 이용할 수 없게 해야 한다

```
{
  “@context“: [
    “https://www.w3.org/ns/did/v1“,
    “https://w3id.org/security/suites/jws-2020/v1“
  ],
  “id“: “did:web:example.com“,
  “verificationMethod“: [
    {
      “id“: “did:web:example.com#key-0“,
      “type“: “JsonWebKey2020“,
      “controller“: “did:web:example.com“,
      “publicKeyJwk“: {
        “kty“: “OKP“,
        “crv“: “Ed25519“,
        “x“: “0-e2i2_Ua1S5HbTYnVB0lj2Z2ytXu2-tYmDFf8f5NjU“
    }
  ],
  “authentication“: [
    “did:web:example.com#key-0“,
  ],
  “assertionMethod“: [
    “did:web:example.com#key-0“,
  ],
  “keyAgreement“: [
```

```
      "did:web:example.com#key-1",
    ]
}
```

(그림3-14) JSON Web Key를 활용한 DID 문서 예시

```
{
  "@context":
        [
         "https://www.w3.org/ns/did/v1",
         "https://w3id.org/security/suites/secp256k1recovery-2020/v2"],
  "id": "did:web:example.com",
  "verificationMethod": [{
      "id": "did:web:example.com#address-0",
      "type": "EcdsaSecp256k1RecoveryMethod2020",
      "controller": "did:web:example.com",
      "blockchainAccountId":
        "eip155:1:0x89a932207c485f85226d86f7cd486a89a24fcc12"
  }],
  "authentication": [
      "did:web:example.com#address-0"
  ]
}
```

(그림3-15) Ethereum 주소를 활용한 DID 문서 예시

2. did:webs

ㅇ ToIP에서 제안한 DID Method로, did:web + secure의 약자이며 https에 있는 s와 동일한 의미를 가진다

- did:webs는 did:web과 마찬가지로 기존의 Web 인프라를 활용하여 DID를 게시하고 검색할 수 있도록 한다 [11]

ㅇ ToIP란 Trust Over의 줄임말로 디지털 신뢰 생태계를 구축하기 위한 프레임워크이며, 인터넷에서 신뢰를 형성하고 유지하는데 필요한 표준, 프로토콜 기술 및 가이드라인을 제공한다

ㅇ DNS, webmasters, X509, 인증기관을 신뢰 기반으로 사용하는 did:web과 달리, KERI (Key Event Receipt Infrastructure, 데이터 교환과 통신을 위한 규칙 집합)를 사용하여 식별자를 제어하는 사람들에 의한 cryptographic key events secure chain을 제공한다

ㅇ 신뢰 확립을 위해 블록체인 네트워크가 필요하지 않지만, KERI를 사용함으로써 임의의 블록체인 네트워크가 선택적인 게시 메커니즘으로 참조되도록 허용하여 블록체인 생태계와의 상호 운용성을 보장한다

ㅇ 암호화 신뢰 (Cryptographic trust)가 분산되어 있고 Governance가 투명하다

ㅇ 블록체인에 의존적이지 않으므로 블록체인 관련 규제 문제에서 벗어나 있지만 배포와 검색이 Web에 의존적이고 이는 분산화와 프라이버시에 영향을 줄 수 있다

ㅇ KERI 시스템을 이해하고 사용하기 위한 시간 및 노력이 별도로 필요하며 사용자에게 책임감, 신중함, 자율성을 요구한다

ㅇ 메소드 이름은 다음과 같은 규칙을 준수하여야 한다
- 이 DID method를 식별하는 식별자는 반드시 webs여야 한다
- 이 method를 사용하는 DID는 반드시 접두사 did:webs: 로 시작해야 한다
- DID 명세에 따라, 이 문자열은 반드시 소문자여야 한다

- 접두사 이후 나머지 부분은 case-sensitive한 identifier 여야 한다

ㅇ 식별자는 다음과 같은 규칙을 준수하여야 한다
- did:webs의 method-specific-identifier는 반드시 did:web과 동일한 선택적 경로가 존재하는 Host + 경로의 마지막 구성요소로 존재하는 KERI Autonomic Identifier (AID) 의 두 부분으로 구성되어야 한다
- Host는 반드시 RFC1035 [12], RFC1123 [13], RFC2181 [14] 에서 찾을 수 있는 유효한 구문을 설명하는 공식 규칙을 준수해야 한다
- 포트는 “:” 을 포함하며 경로와의 충돌방지를 위해 반드시 Percent Encoding 되어야 한다
- 디렉토리 및 하위 디렉토리 포함이 가능하며 “/” 대신 “:” 으로 구분된다
- KERI AID는 고유한 식별자이며 KERI 식별자의 Inception Event로부터 도출되어야 한다
- did:web과의 호환성을 위해 AID는 경로요소로 취급된다
- 모든 did:webs는 did:web으로 표현될 수 있지만, 역은 성립하지 않는다

ㅇ did:web과 마찬가지로 did:webs는 DID 중 Host 부분의 DID가 Resolve되는 Web서버에서 데이터를 읽어야 한다

ㅇ did:webs의 DID는 did:web과 마찬가지로 HTTPS URL을 통해 간단한 텍스트 변환을 사용하여 DID 문서를 확인해야 한다

ㅇ 동일한 식별자를 가진 did:web과 did:webs의 DID는 id, controller, alsoKnownAs와 같은 사소한 속성을 제외하고 같은 DID 문서를 반환해야 한다

ㅇ did:web과 마찬가지로 DID 문서의 위치는 DID를 HTTPS URL로 변환

하여 결정해야 하며, 규칙은 did:web과 동일하나 path 요소가 없을 경우에 대한 변환은 존재하지 않는다

ㅇ 생성된 URL에 GET 요청을 보내면 반드시 DID 문서를 반환해야 한다

ㅇ KERI Event Stream의 위치는 URL을 아래와 같이 변환하여 결정됨
- (1) /did.json을/keri.cesr로 대체한다
- (2) URL에 대한 GET 요청은 did:webs 식별자의 AID에 대한 KERI Event Stream을 반환해야 한다
- (3) KERI event stream은 application/cesr 형식이어야 하며, KERI event은 KERI Rule을 사용하여 확인할 수 있어야 한다

ㅇ did:web 버전의 did는 동일한 did.json파일을 가리키지만, keri.cesr 파일에 대한 참조는 없어야 한다

did:webs:w3c-ccg.github.io:12124313423525
URL
ex) https://w3c-ccg.github.io/12124313423525/did.json
KERI event stream URL
ex) https://w3c-ccg.github.io/12124313423525/keri.cesr

did:webs:w3c-ccg.github.io:user:alice:12124313423525
URL
ex) https://w3c-ccg.github.io/user/alice/12124313423525/did.json
KERI event stream URL
ex) https://w3c-ccg.github.io/user/alice/12124313423525/keri.cesr

did:webs:example.com%3A3000:user:alice:12124313423525
URL
ex) https://example.com:3000/user/alice/12124313423525/did.json

KERI event stream URL ex) https://example.com:3000/user/alice/12124313423525/keri.cesr

(그림3-16) AID가 추가된 did:webs URL 예시

○ did:webs가 생성-해석-업데이트-비활성화되는 일련의 CRUD 절차는 아래와 같이 이루어진다

○ **(Create)** did:webs의 DID 생성과정은 아래와 같다

- (1) AID가 될 마지막 요소를 제외하고 DID 문서가 게시될 URL을 선택한다
- (2) KERI AID를 만들고 URL의 마지막 부분에 추가한다
- (3) verification method 및 service endpoint와 같은 DID 문서의 속성에 해당하는 적절한 KERI Event를 AID의 KERI Log에 추가한다
- (4) KERI Event Stream을 처리하여 DID 문서를 도출한다
- (5) 호환성을 보장하기 위해 위해 도출된 did:webs의 DID 문서는 일치하는 did:web문서로 변환될 수 있어야 한다
- (6) 반드시 선택한 위치에 있는 Web 서버에 AID에 해당하는 폴더를 생성하고 did:web의 문서 리소스 did.json과 KERI Event Stream 리소스 keri.cesr를 배치해야 한다

○ **(Read)** did:web에서 DID 문서를 얻기 위한 과정은 다음과 같다

- (1) did:webs URL를 HTTPS URL로 변환한다
- (2) DID 문서의 URL과 KERI Event Stream의 URL 모두에게 HTTP GET 요청을 수행한다
- (3) KERI Rules를 활용하여 KERI Event Stream을 처리한 후 DID 문서를 도출한다
- (4) 검색된 did:web 문서를 일치하는 did:webs 문서로 변환해야 한다
- (5) 파생된 did:webs의 DID 문서가 변환된 DID 문서와 일치하는지 확인한다

- (6) KERI를 인식하는 Application들은 KERI의 사용으로 활성화된 추가 기능을 활용하기 위해 KERI Event Stream을 활용할 수 있다

○ **(Update)** did:webs의 업데이트 과정은 다음과 같다

- (1) did:webs DID의 AID가 업데이트 가능한 경우, KERI Event Stream에 KERI Event를 추가하여 AID를 업데이트해야 한다
- (2) did:webs DID와 관련된 KERI Event Stream에 대한 업데이트는 가능한 한 빨리 DID 문서에 반영되어야 하며, 만약 did:webs DID 파일이 정적으로 Host되면 Web 서버에 다시 게시되어 기존 파일을 덮어쓰게 된다

○ **(Deactivate)** did:webs의 비활성화 과정은 다음과 같다

- (1) did:webs를 비활성화 하려면 Controller가 key를 null로 바꾸는 효과가 있는 KERI Event를 실행하고 DID 문서와 KERI Event Stream을 계속 게시해야 한다
- (2) 비활성화 이벤트가 적용되면, Controller는 KERI Event Stream에서 DID 문서를 재생성하고 did.json 및 keri.cesr를 Web 서버에 다시 게시하여 기존 파일을 덮어쓴다
- (3) Controller는 게시된 Web 서버로부터 DID 폴더 및 파일을 제거해서는 안 된다

○ did:webs의 DID 문서는 해당 AID의 KERI Event Stream에서 파생 혹은 생성되어야 한다

- AID의 KERI Event Stream을 처리할 때, 생성 알고리즘은 DID 문서를 생성하기 위해 AID Key Event Log (KEL) 및 연결된 Transaction Event Log (TEL)을 읽어야 한다
- did:webs의 DID 문서는 반드시 순수한 JSON이어야 하며, 문서를 요청한 사람이 원할 경우 @context를 추가하여 JSON-LD로 처리할 수 있다

- 모든 Hash, 암호화 키, 서명은 반드시 CESR 문자열로 나타내야 한다

○ **(KERI KSN)** 표 3-1 과 같은 key state를 참조로 사용하여 그림 3-17의 현재 키 상태에서 field를 식별할 수 있으며, 이는 DID 문서의 값으로 변환된다


```
{
    “v“: “KERI10JSON000274_“,
    “i“: “EeS834LMlGVEOGR8WU3rzZ9M6HUv_vtF32pSXQXKP7jg“,
    “s“: “1“,
    ...
    “dt“: “2021-11-04T12:55:14.480038+00:00“,
    “b“: [
      “BGKVzj4ve0VSd8z_AmvhLg4lqcC_9WYX90k03q-R_Ydo“,
      “BuyRFMideczFZoapylLIyCjSdhtqVb31wZkRKvPfNqkw“,
      “Bgoq68HCmYNUDgOz4Skvlu306o_NY-NrYuKAVhk3Zh9c“
    ],
    “ee“: {
      “s“: “0“,
      “d“: “ESORkffLV3qHZljOcnijzhCyRT0aXM2XHGVoyd5ST-Iw“,
    },
    “di“: ““
}
```


(그림3-17) KERI KSN 예시

[표3-1] key state

Key	내용	비고
v	Version String	
i	Identifier Prefix	

s	Sequence Number	
t	Message Type	
te	Last received Event Message Type in Key State Notice	
d	Event Digest (Seal or Receipt or Key State Notice)	
p	Prior Event Digest	
kt	Keys Signing Threshold	
k	List of Signing Keys (ordered key set)	
n	Next Key Set Commitment	
wt	Witnessing Threshold	
w	List of Witnesses (ordered witness set)	
wr	List of Witnesses to Remove (ordered witness set)	
wa	List of Witnesses to Add (ordered witness set)	
c	List of Configuration Traits/Modes	
a	List of Anchors (seals)	
da	Delegator Anchor Seal in Delegated Event (Location seal)	
di	Delegator Identifier Prefix in Key State	
rd	Merkle Tree Root Digest	
ee	Last Establishment Event Map in Key State	
vn	Version Number (“major.minor”)	exact scheme tbd
dt	time stamp	
et	event type	
br	list of branches	
ba	list of witnesses to Add	

ㅇ did:webs의 식별자 속성은 아래 내용과 같다

ㅇ DID Subject

- DID 문서의 id 값은 생성 혹은 resolution 된 did:webs의 DID이다
- key state의 i 값은 did:webs의 마지막 “:” 이후의 값이어야 한다

ㅇ DID Controller
- DID 문서의 controller 값은 id와 같은 값이어야 한다

ㅇ Also Known As
- DID 문서의 Root에 있는 alsoKnownAs 속성은 동일한 AID를 가지는 임의의 DID를 가지고 있을 수 있다
- did:webs의 DID는 did:webs와 함께 그것과 일치하는 did:web의 식별자를 alsoKnownAs로 제공해야 한다
- did:webs의 DID는 해당하는 did:keri를 alsoKnownAs로 제공해야 한다
- 동일한 AID는 여러 개의 did:webs의 DID와 연관 되어 있을 수 있으며, 각각의 host, path 등은 다를 수 있다
- did:webs의 DID는 AID의 designated aliases 증명 (attestation)에 나열되어야 한다

```
{
  “alsoKnownAs“: [

“did:web:example.com:ENro7uf0ePmiK3jdTo2YCdXLqW7z7xoP6qhhBou6gBLe“,
    “did:web:foo.com:ENro7uf0ePmiK3jdTo2YCdXLqW7z7xoP6qhhBou6gBLe“,
    “did:webs:foo.com:ENro7uf0ePmiK3jdTo2YCdXLqW7z7xoP6qhhBou6gBLe“
  ]
}
```

(그림3-18) alsoKnownAs 배열 예시

ㅇ Verification Methods
- KSN의 k필드의 배열 값에 나열된 각각의 키에 대하여 상응하는 ver

ification method가 DID 문서에 생성되어야 한다
- 각 Public Key의 verification method의 type 속성은 반드시 Public Key를 생성한 알고리즘에 의해 결정되어야 한다
- verification method의 타입은 DID Specification Registries와 이specification에 등록해야 한다
- verification method의 id 속성은 연관된 DID URL 이어야 하며 KERI key의 CESR 값을 fragment 구성 요소의 값으로 사용해야 한다
- verification method의 controller 속성은 반드시 DID 문서의 id 값이어야 한다

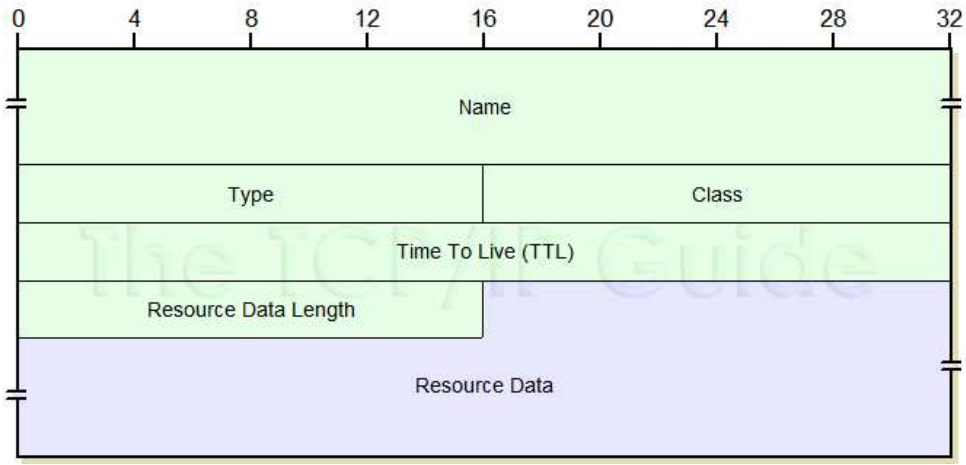
○ **(did:webs에서 web DID 문서로의 변환)** Web 서버의 리소스로 존재하는 DID 문서는 did:web과 호환되므로 id, controller, alsoKnownAS 속성이 did:webs와 달라야 한다
- DID 문서의 id 및 controller 속성 값에서 did:webs 접두사 문자열을 did:web으로 교체해야 한다
- alsoKnownAs 속성의 값에서 id 속성의 새 값 (did:web을 사용한 값)으로 이전 id 속성 값 (did:webs를 사용한 값)을 교체해야 한다
- 동작은 Create 과정에서 수행되며, DID 문서의 다른 모든 내용은 수정하지 않아야 한다

○ **(did:web에서 webs DID 문서로의 변환)** did:web의 DID 문서에 대하여 did:webs의 DID 문서는 다음 차이점이 있어야 한다
- DID 문서의 최상위 레벨 id와 controller 속성 값에서 did:web 접두사를 did:webs로 교체해야 한다
- alsoKnownAs 속성은 반드시 새 값 (did:webs를 사용한 값)으로 이전 id 속성 값 (did:web을 사용한 값)을 교체해야 한다
- 문서의 다른 모든 내용은 수정하지 않아야 한다

3. did:dns

ㅇ did:web에서 몇 가지 개선점을 제공한 버전으로, did:key와 같은 일시적인 목적으로 사용할 did를 사람이 읽을 수 있는 형태로 검색할 수 있도록 설계되었다

ㅇ did:dns는 DNS 레코드에 ip 대신 did:key에 대한 정보를 담고 해당 did:key를 기반으로 did 문서를 생성한다 [15]



(그림3-19) DNS resource record (RR) 구조

[표3-2] RR 요소

Label	내용
NAME	RR의 주체, domain name을 나타냄
TYPE	RR의 유형
CLASS	요청된 RR의 클래스, 보통 인터넷(IN)에 대한1 값을 가짐
TTL	time to live, record가 캐시에 저장될 수 있는 시간
data length	데이터의 길이
RDATA	실제 데이터 부분

ㅇ DID의 식별자인 도메인 네임이 속한 authoritative DNS 서버에 의해 제어되는 DNS Zone File의 Set (or Chain)을 타겟으로 한다

ㅇ Method Name
- 해당 방식을 식별할 메소드 이름은 “dns” 이다. 해당 메소드를 사용하는 DID는 반드시 “did:dns”로 시작해야 하며 소문자여야 한다

ㅇ did:dns가 생성-해석-업데이트-비활성화되는 일련의 CRUD 절차는 아래와 같이 이루어진다

ㅇ Create
- 도메인 네임 혹은 서브도메인을 domain name register 등을 활용하여 등록해야 한다
- Resource Records(RR) 내 did:key를 포함하도록 도메인 네임의 Authoritative DNS Zone File을 구성해야 한다

ㅇ Resolve
- (1) resolved DID document라 불리는 비어있는 DID 문서를 생성한다
- (2) resolved DID document의 id를 현재 resolve 중인 DID로 설정한다
- (3) 도메인 네임과 연관된 RR을 검색하기 위해 DID의 식별자에서 도메인 네임을 확인해야 한다. 이 과정은 DNSSEC와 다른 보안 메커니즘을 사용하여 수행하는 것이 권장된다
- (4) 만약 도메인 이름이 존재하지 않는다면, DID는 더 이상 존재하지 않는 것으로 간주되고 notFound 에러를 리턴해야 한다
- (5) did:key를 Resolve하여 DID 문서를 구성하고, did:key의 식별자를 did:dns의 식별자로 대체하여 문서를 구성한다

ㅇ Update
- DID 문서를 업데이트 하기 위해, 도메인 네임의 Authoritative DNS

Zone File을 통해 RR을 업데이트 된다

ㅇ Deactivate

- DID 문서를 비활성화 하기 위해 도메인이 삭제되어야 한다

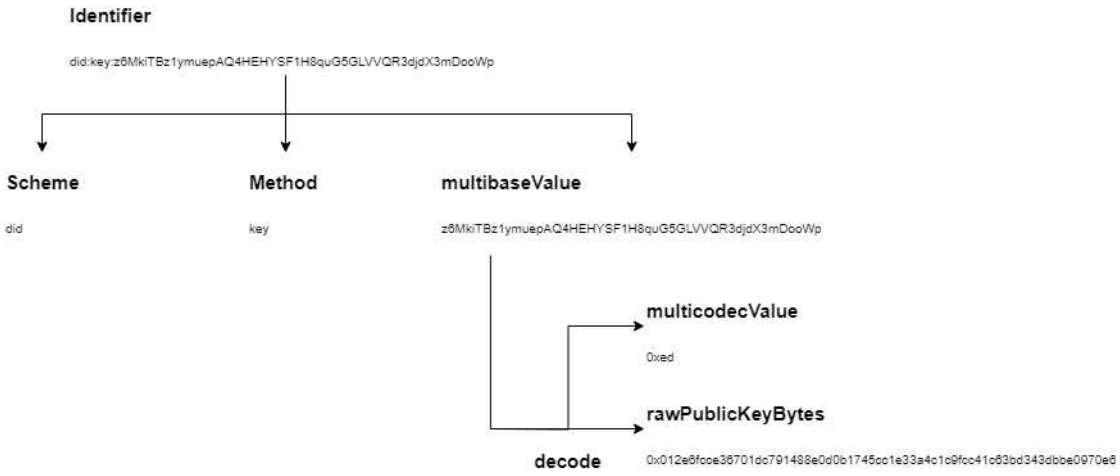
ㅇ **(did:key)** 일시적으로 사용하는 데 초점이 맞춰져 있는 DID Method로, 오랜 기간 사용하려는 경우 안전한 하드웨어 보안 방식과 함께 사용하는 것이 권장된다 [16]

- did:key의 식별자에 대한 ABNF Rules는 그림 3-20과 같다

```
did=key=format := did:key:<mb-value>
mb-balue       := z[a-km-zA-HJ-NP-Z1-9]+
```

(그림3-20) did:key의 ABNF rules

- 공개 키 생성 알고리즘으로 생성된 Public Key를 multicodec 형식으로 키 생성 알고리즘을 명시하고 이를 base58-btc 형식으로 인코딩 함으로써 식별자를 생성한다. 그림3-11과 같이 did 문서는 이 식별자를 메소드를 통해 확장하여 생성하게 된다
- CRUD중 Update와 Deactivate는 지원하지 않으며, Read의 경우 단순히 DID 값을 DID 문서로 확장하는 작업이므로 Create와 동일하다
- DID 문서 생성은 식별자 값을 요구하는 특정 알고리즘을 통해 이루어진다
- 식별자의 경우 scheme, method, multibaseValue의 세 부분으로 나누어지며, multibaseValue는 디코딩시 multicodecValue와 rawPublicKey Bytes로 나뉜다



(그림3-21) did:key Identifier 구조

```
{
  “@context“: [
    “https://www.w3.org/ns/did/v1“,
    “https://w3id.org/security/suites/ed25519-2020/v1“,
    “https://w3id.org/security/suites/x25519-2020/v1“
  ],
  “id“: “did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK“,
  “verificationMethod“: [{
    “id“:
        “did:key:z6Mkha...“,
    “type“: “Ed25519VerificationKey2020“,
    “controller“: “did:key:z6Mkha...“,
    “publicKeyMultibase“: “z6Mkha...“
  }],
  “authentication“: [
    “did:key:z6Mkha...“
  ],
  “assertionMethod“: [
```

```
    "did:key:z6Mkha..."
  ],
  "capabilityDelegation": [
    "did:key:z6Mkha..."
  ],
  "capabilityInvocation": [
    "did:key:z6Mkha..."
  ],
  "keyAgreement": [{
    "id":
        "did:key:z6Mkha...",
    "type": "X25519KeyAgreementKey2020",
    "controller": "did:key:z6Mkha...",
    "publicKeyMultibase": "z6LSj72tK8brWgZja8NLRwPigth2T9QRi..."
  }]
}
```


(그림3-22) did:key의 DID 문서 예시

제 4 절 기타 DID Method

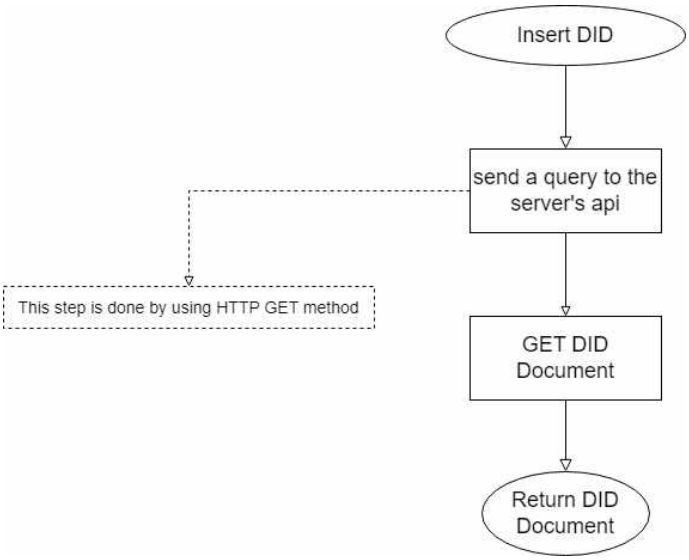
1. Web 서버, 호스트 등 외부 저장소를 사용하는 Methods

ㅇ 해당 DID Method들은 중앙 저장소를 관리하는 서버 자체가 신뢰 기반이자 DID의 발급자로서 동작한다. 혹은 DID 문서만 저장하고 신뢰 기반은 별도로 명시할 수 있다

ㅇ did:content

- 이미지, 비디오, 음악과 같은 컨텐츠를 식별하는 DID Method로, 컨텐츠와 그 권리를 분산시켜 관리하는 것이 목적이다

- resolved DID document의 id를 현재 Resolve 중인 DID로 설정해야 한다



(그림3-23) 외부 저장소 기반 Method의 DID 문서 획득 절차

did:content: “did:content” + content-id content-id: Base58 (RIPEMD160 (sha3-256 (DID Document withour id)))

(그림3-24) did:content의 ABNF rules

- content-id는 utf-8로 인코딩한 DID 문서를 sha3-256 해시 함수를 통과시킨 후 다시 이를 RIPEMD160 해시 함수를 통과시킨 뒤 Base58로 인코딩한다

ㅇ did:mydata

- Data Agreement와 함께 사용하기 적합한 DID를 생성하고 편집하는 방법을 명시한 DID Method이다
- Data Agreement는 데이터 제공자와 사용자 간의 명시적인 계약 혹은 합의를 나타낸다

```
mydata-did = “did:mydata” ( “:” did-type-integer) “:” mb-value
did-type-integer = 0 / 1 / 2 / 3 / 4
mb-value = z[a-km-zA-HJ-NP-Z1-9]
```

(그림3-25) did:mydata의 ABNF rules

- mb-value는 ed25519의 키를 Multicodec을 통해 키 타입을 명시하고 base58-btc로 인코딩되어 있는 형태이다
- did-type-integer는 선택사항이며, DID에 의해 식별된 DID 주체의 유형을 나타낸다

[표3-3] did-type-integer값 별 DID 주체유형

did-type-integer value	내용
0	개인 데이터를 수집하는 조직인 Data Source
1	데이터 주체 혹은 개인
2	서비스를 제공하기 위해 하나 이상의 Data Source에서 개인 데이터를 처리하는 조직인 Data Using Service
3	Assessor(평가자, 데이터 보호와 개인정보 관리의 감시자)가 조직의 관행을 검토하고 Data Protection Impact Assessment (DPIA)를 수행하며, 제 3자를 위한 Data Agreement 및 회사간 계약 초안 작성
4	Auditor(감사원)가 Data Agreement를 검토하고 Data breaches 혹은 정기 검사 시 Data Agreement가 제대로 이루어졌는지 확인하기 위해 요청

- CRUD 작업은 DID Comm을 통해 수행되고, Automated Data Agreement(ADA) 서비스 자체가 DID를 할당받는다
- DID-SPEC 문서 [7]에 명시된 바와 같이 well-known DID Configurati

on 엔드포인트에서 접근이 가능하다

2. 식별자가 DID 문서로 해석되는 Methods

ㅇ 대부분 일시적인 용도로 사용하는데 그 목적이 있으며, 공개 키 자체를 확장하여 DID 문서의 구조에 맞게 작성하게 된다

ㅇ did:oyd

- Own Your Decentralized Identifier(OYDID)에 맞춰 동작하는 DID Method
- OYDID는 DID 및 DID 문서를 공공 원장에 저장하지 않고, 하나 이상의 공개 가능한 로컬 스토리지를 사용하는 접근 방식을 취한다
- OYDID는 식별자와 DID 문서를 암호학적으로 연결한다
- **(Create)** DID 문서 생성에는 JSON으로 이루어진 payload와 암호화 키 쌍이 필요하다
- **(Read)** did:oyd의 문서를 읽는 것은 식별자를 인코딩하여 직접 연결하거나 로그를 통해 제공된 식별자를 최신 버전의 DID 문서에 연결할 수 있는 DID 문서를 검색하는 것이다. 식별자는 내부 DID 문서의 해시 값을 인코딩한 값이며, 해시 알고리즘은 MULTIHASH를 사용해 식별자에 인코딩되고 기본값은 SHA2-256이다. 인코딩 알고리즘도 MULTIBASE를 사용해 식별자에 인코딩되며, 기본값은 base58-btc 이다
- **(Update)** 업데이트 과정은 소유권 증명을 위해 취소 문서를 게시하고, 원래 키로 서명함과 동시에 키의 전환을 위한 새로운 DID 문서와 키를 제공하는 것이다
- **(Deactivate)** OYDID를 비활성화하는 것은 업데이트를 제공하지 않고, private revocation entry를 게시함으로써 이루어진다. 식별자를 DID 문서와 암호학적으로 연관 짓는다는 점에서 did:key와 비슷하나, Log File을 별도로 저장하여 최신 버전을 무조건 보장하는 기능이 추가되었다

제 4 장 DID Related Identification Methods

제 1 절 OpenID

1. 개요

ㅇ OpenID (OpenID Connect, OIDC)는 OAuth2.0을 기반으로 사용자가 다양한 Web사이트나 서비스에서 하나의 ID로 인증 (Single-Sign On, SSO)할 수 있는 사용자 인증 프로토콜이다

ㅇ OAuth2.0은 자원 접근을 위한 권한 부여에 중점을 둔 프로토콜로, OpenID Connect는 OAuth2.0에 인증 기능을 추가하여 사용자의 신원 인증 기능을 제공한다

ㅇ OpenID Connect는 사용자 인증을 신뢰할 수 있는 제공자 (Identity Provider, IdP)에게 맡기고, 서비스 제공자 (Relying Party, RP)에게 필요한 정보를 제공하는 방식을 이용한다

ㅇ 사용자가 하나의 인증 서버에 로그인하면, 해당 인증 정보와 토큰이 여러 RP에서 재사용될 수 있으며, 이러한 원리로 OpenID는 SSO 기능을 제공한다

2. OpenID Connect 작동 방식

ㅇ OpenID Connect 작동 방식은 다음과 같다

- (1) 사용자는 브라우저를 통해 Web 사이트나 Web 애플리케이션으로 이동하여 OpenID와 비밀번호를 입력한다

- (2) RP는 사용자를 인증하기 위해 IdP에 요청을 보낸다
- (3) IdP에서 권한 승인이 이루어지면, 인증 서버는 인증 코드를 발급한다
- (4) RP는 수신한 인증 코드를 사용하여 토큰 엔드포인트에 Access Token과 ID Token을 요청한다
- (5) 토큰 엔드포인트는 RP에 Access Token, Refresh Token, ID Token을 발급한다. Access Token으로 보호된 자원에 접근할 수 있고, ID Token으로 사용자 인증 정보를 확인할 수 있다

3. OpenID와 DID

ㅇ OpenID와 DID는 디지털 신원 관리 및 인증 시스템의 두 가지 중요한 패러다임으로, 사용자 인증과 신원 확인이라는 공통된 목표를 가지고 있으나, 그 접근 방식과 구현에는 차이점이 존재한다

ㅇ 사용자는 특정 DID를 통해 신원을 증명할 수 있는 다양한 증명을 발급받고, 이를 필요할 때마다 제시할 수 있으나, OpenID에서는 IdP가 RP에 인증 토큰을 제공하는 방법으로 신원 인증을 진행한다

ㅇ DID는 분산 원장을 사용하여 신원 정보를 저장하여 사용자가 자신의 신원 정보를 직접 소유하고 통제할 수 있는 반면, OpenID는 사용자의 신원 정보가 IdP에 저장된다. IdP에 사용자의 신원 정보를 저장하기 때문에 프라이버시 문제가 발생할 가능성이 존재한다

ㅇ OpenID Connect와 같은 기존 인증 시스템과 DID의 통합을 통해 중앙화된 IdP의 역할을 줄여 사용자의 신원을 좀 더 안전하고 분산된 방식으로 관리하고 신원의 자율성을 높이려는 시도가 진행되고 있다

제 2 절 DIDComm Messaging

1. 개요

- DIDComm Messaging은 DID를 기반으로 하는 탈중앙화된 통신 프로토콜로, 사용자 (DID 주체) 간의 통신을 위해 사용된다

- DIDComm Messaging은 DID를 통해 서로의 신원을 검증하고 안전한 메시지 교환을 가능하게 하여 중앙화된 신뢰기간에 대한 의존성을 없애는 것을 목표로 한다

- 탈중앙화신원 (Self-Sovereign Identity, SSI) 시스템의 핵심 요소로, VC 및 DID Document의 통신에 사용된다

- DIDComm Messaging은 다양한 DID 네트워크 간의 메시지 교환을 지원한다. 또한, 전송 채널에 독립적이어서 다양한 프로토콜을 통해 전송될 수 있으며, 통신 수단에 구애를 받지 않는다

- DIDComm Message에는 일반 텍스트 메시지, 암호화된 메시지 (Encrypted Message), 서명된 메시지 (Signed Message) 3가지 유형이 존재한다

- 암호화된 메시지는 암호화된 JWM으로, 권한이 있는 수신자 외에는 모든 수신자에게 콘텐츠를 숨기고, 해당 수신자에게만 정확히 공개하고 증명하여 무결성을 보장한다. 이는 DIDComm Messaging 데이터를 저장하기에 가장 안전한 형식이며, 암호화된 메시지의 미디어 유형은 application/didcomm-encrypted+json 이어야 한다

ㅇ 서명된 메시지는 거부 불가능한 서명을 평문 메시지에 연관시킨 서명된 JWM으로, 일반 텍스트의 출처를 제 3자에게 증명해야 하는 경우 또는 브로드캐스트 시나리오 등에 사용된다. 서명된 메시지의 미디어 유형은 application/didcomm-signed+json이어야 한다

2. DIDComm Messaging의 기술적 구성 요소

ㅇ **(DID)** 각 참가자는 고유의 DID를 가지고 있다

ㅇ **(DID 문서)** DID와 연관된 공개 키, 서비스 엔드포인트, 인증 방법 등의 정보를 담고 있는 문서이다

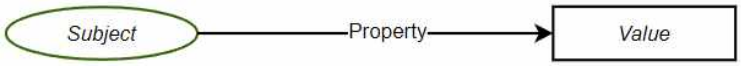
ㅇ **(DIDComm Protocol)** DID를 기반으로 하는 메시징 프로토콜로, 안전한 메시지 교환을 가능하게 한다. 이를 통해 디지털 인증서, 서명, 암호화된 메시지 등의 기능을 활용할 수 있다. 상대방의 DID 문서를 조회하여 상대방의 DID를 확인하고, Public Key와 통신 엔드포인트를 파악하며 통신이 이루어진다

제 3 절 VC/VP System

3절의 전반적인 내용은 VC-DATA-MODEL 문서 [17] 를 기반으로 한다

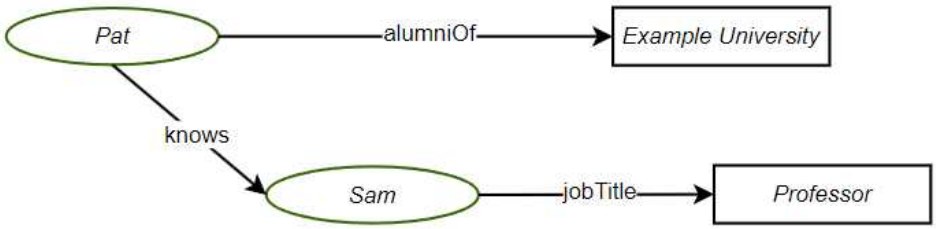
1. 개요

ㅇ 클레임(claims, 주장)
- 클레임은 주체에 대한 진술이며, 주체는 클레임을 할 수 있는 사물이다
- 클레임은 다양한 진술을 표현하는 데 사용될 수 있다



(그림4-1) 클레임 기본 구조

- 개별적인 클레임은 한 주체에 대한 정보 그래프를 표현하기 위해 병합될 수 있으며, 주장된 클레임을 신뢰할 수 있으려면 더 많은 정보가 추가되어야 한다



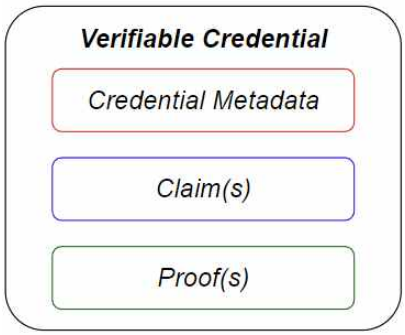
(그림4-2) 확장된 클레임 예시

ㅇ 크리덴셜 (Credentials, 자격 증명)

- 크리덴셜은 동일한 주체가 만든 하나 이상의 클레임 집합이다
- 크리덴셜에는 발급자 정보, 유효 날짜 및 기간, 검증 자료, 상태 정보 등과 같은 자격 증명의 속성을 설명하는 식별자와 메타데이터도 포함될 수 있다

ㅇ VC

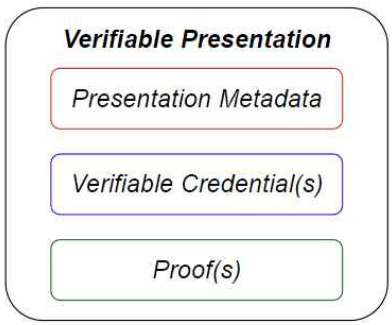
- 검증 가능한 자격 증명 (VC)은 누가 발급했는지 암호화하여 증명하는 변조 방지 클레임과 메타데이터의 집합이다
- VC의 예로는 디지털 직원 신분증, 디지털 운전면허증, 디지털 증명서 등이 있고 활용 방면은 무수히 많다



(그림4-3) VC data model

ㅇ VP

- VC/VP 시스템은 프라이버시 강화를 중점으로 설계되었고, 시스템에서의 주체가 되는 엔티티는 주어진 상황에서 원하는 정보만을 증명에 사용할 수 있음. 프레젠테이션은 크리덴셜에 포함된 정보의 일부를 하위 집합으로 표현한 것이다
- VP는 하나 이상의 발급자가 발급한 하나 이상의 VC에서 파생된 데이터이다
- VP는 소유자가 생성하고, 여러 VC를 포함하여 데이터를 표현할 수 있으며, 임의의 추가 데이터를 포함할 수 있다
- 소유자가 검증자에게 클레임을 제시할 때 VP가 사용되고, 이때 VC를 직접 제시하는 것도 가능하다



(그림4-4) VP data model

2. VC/VP System 개요

ㅇ 주체

- 클레임이 제기되는 것으로, 인간, 동물, 사물이 포함될 수 있다

ㅇ 발급자(Issuer)

- 엔티티가 하나 이상의 주체에 대한 클레임을 주장하고, 클레임에서 VC를 발급하고, VC를 소유자에게 전송한다

ㅇ 소유자(Holder)

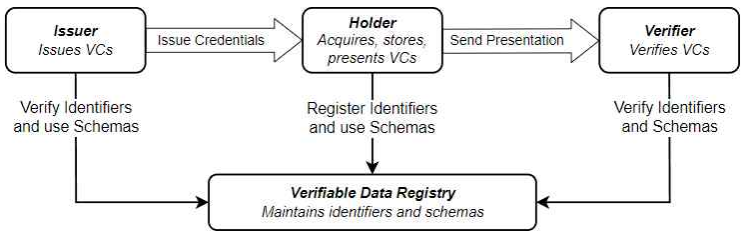
- 엔티티가 하나 이상의 VC를 소유하고, 이를 통해 VP를 생성하여 검증자에게 제출한다
- 소유자는 자격 증명 저장소에 자격 증명을 저장한다

ㅇ 검증자(Verifier)

- 하나 이상의 VP 또는 VC를 수신하여 검증을 수행한다

ㅇ 검증 가능한 데이터 레지스트리

- 발급자의 Public Key, VC 등 식별자, 키 및 기타 관련 데이터의 생성과 검증을 중재할 수 있는 저장소이다
- 검증 가능한 데이터 레지스트리는 신뢰할 수 있는 데이터베이스, 분산 데이터베이스를 포함할 수 있다



(그림4-5) VC/VP system overview

3. VC/VP System Architecture with DID

- VC/VP에는 시스템 참여자들에 대한 식별자를 표현하는 속성들이 존재하는데, 해당 속성값에 DID를 넣을 수 있다

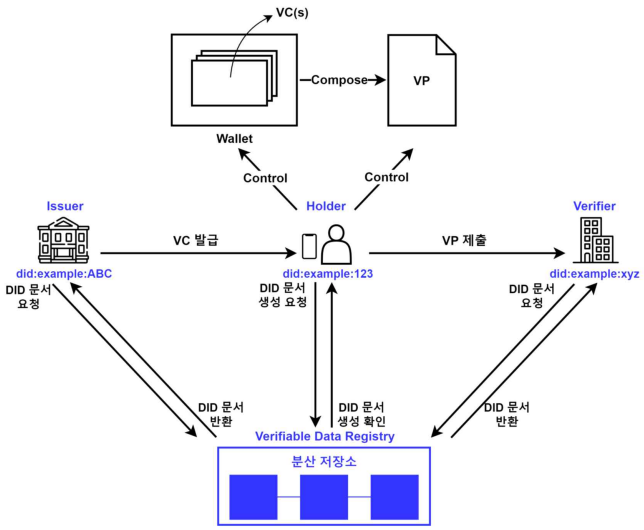
- DID와 VC/VP가 서로 의존하는 관계는 아니며, DID는 VC가 유용하기 위한 목적으로 반드시 필요한 식별자 유형이 아니다

- 그러나 많은 VC/VP에서 DID를 사용하고, VC/VP 시스템을 구현하는 소프트웨어 라이브러리는 DID를 Resolve 해야 한다

- DID 기반 URL은 주체, 발급자, 소유자, 자격 증명 상태 목록, 암호화 키 및 VC와 관련된 기타 기계 판독 가능 정보와 관련된 식별자를 표현하는 데 사용된다

4. DID를 활용한 VC/VP System Scenario

- VC 발급
  - 발행인 (did:example:ABC)은 사용자 (did:example:123)에게 사용자의 ID 속성을 증명할 수 있는 신원 증명을 발급해 준다. 해당 VC에는 크리덴셜 메타데이터, claim, proof가 포함되어야 한다
  - 크리덴셜 메타데이터에는 VC의 발행인, VC가 명시하고 있는 credentialSubject 등의 요소가 포함된다
  - VC의 proof 하위로는 발행인이 해당 VC를 발급할 때 사용한 서명에 대한 속성값들이 포함된다. 예시에서 proof는 발행인의 DID 문서 내에 클레임을 주장할 때 사용하겠다고 명시한 assertionMethod 검증 관계에 대한 키를 사용했음을 보여주고 있다



(그림4-6) VC/VP system architecture


```
{
    ...
    "type" : ["VerifiableCredential", "AlumniCredential"],
    "issuer" : "https://example.edu/issuers/565049",

    "credentialSubject": {
        "id": "did:example:123",
        "alumniOf": { ... }
    },
    "proof": {
        "type": "Ed25519VerificationKey2018",
        "created": "2017-06-18T21:19:10Z",
        "proofPurpose": "assertionMethod",
        "verificationMethod": "https://example.edu/issuers/565049#keys-2",
        "jws": "eyJh...BBPM"
    }
}
```

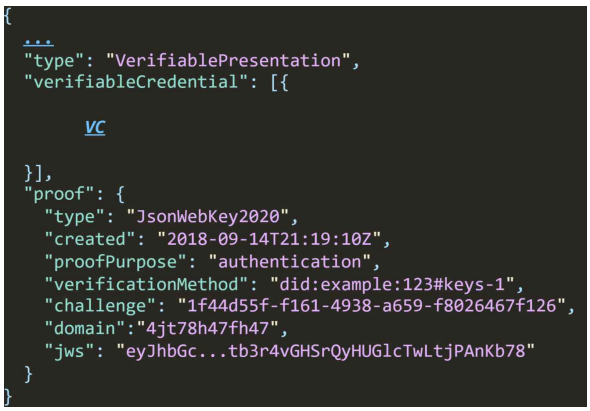

(그림4-7) 사용자가 발행인으로부터 발행받은 VC

ㅇ VP 생성

- 사용자는 발급받은 여러 개의 VC 중에서 공개하고 싶은 VC만을 선택하여 VP를 생성한다
- 해당 VP에는 프레젠테이션 메타데이터, VC, proof가 포함된다
- 프레젠테이션 메타데이터에는 해당 데이터가 VP라는 것을 명시한 type 등 VP 검증에 참고할 수 있는 데이터를 포함한다
- VC의 proof 하위에는 발행인의 서명이 포함된다면, VP의 proof 하위

에는 사용자의 서명이 포함된다

- VP의 proof 하위로는 사용자가 해당 VP를 발급할 때 사용한 서명에 대한 속성값들이 포함된다. 예시에서 proof는 사용자의 DID 문서 내에 인증을 수행할 때 사용하겠다고 명시한 authentication 검증 관계에 대한 키를 사용했음을 보여주고 있다



(그림4-8) 사용자가 VC를 일부 선택하여 생성한 VP

ㅇ VC와 VP에서의 DID 문서 검증 관계 속성 요구 사항

- VC를 생성할 때는 발행인이 assertionMethod 속성으로 자격 증명의 진위성에 서명한다. 이 속성은 발행인이 VC에 대한 책임을 증명하고, 자신이 해당 자격 증명을 발행했음을 보증하는 데 사용된다
- VP를 생성할 때는 사용자가 authentication 속성으로 프레젠테이션의 진위성에 서명한다. 이 속성은 사용자가 본인의 신원을 증명하기 위한 목적으로 사용된다
- 두 가지 검증 관계 속성에서 사용하는 키는 반드시 관련 DID 문서의 verificationMethod 속성에 등록되어 있어야 한다
- 검증 관계 속성에서 verificationMethod에 등록되지 않은 키를 직접 정의하는 것은 불가능하다
- DID 문서의 verificationMethod 하위에 키를 여러 개로 정의해두고,

검증 관계 속성값의 정의 없이 상황에 따라 동적으로 키를 지정해서 VC/VP에 서명하는 것은 DID 문서 표준에 부합하지 않는다. 검증 관계 속성 없이 키를 사용하면 어떤 목적에서의 사용인지 목적 판단이 모호해지고, 각 키의 용도를 이해할 수 없기 때문에 검증 프로세스에서 혼란을 초래할 수 있다. 따라서 속성값에 대한 관계 정의를 지키지 않는 것은 권장되지 않는다

ㅇ 검증 프로세스 기본 사항 - VP 검증

- 사용자는 VP를 검증자에게 제출하고, 검증자는 검증을 수행한다
- 검증 프로세스에서는 다음 3가지 항목을 검증한다: 1) VP에 포함된 VC의 proof, 2) VP에 포함된 VC credentialSubject에 대한 DID Auth (신원 인증), 3) VP의 proof
- 3가지의 검증 항목은 필수 사항은 아니며, 제출자의 신분이 중요하지 않거나, Bearer Credential의 경우는 추가적인 인증 과정 없이도 소유자가 자격을 증명할 수 있다. 이런 경우에서는 VC, VP의 모든 항목을 검토할 필요 없이 VP와 연결된 기본 정보에 대해 소유자가 증명된 것으로 간주할 수 있다
- VP에 포함된 VC의 proof 검증은 VC를 발급한 발행인이 올바른 발행인이며 해당 VC가 위/변조되지 않았는지를 확인하기 위해 수행된다
- 제시된 자격 증명의 주체에 대한 DID Auth는 VC가 가리키는 객체가 사용자가 맞는지 확인하기 위해 수행된다
- VP의 proof 검증은 해당 VP의 제출자가 사용자 본인이 맞는지, 그리고 사용자로부터 제출된 VP가 위/변조되지 않았는지 확인하기 위해 수행된다

ㅇ 검증 프로세스 세부 사항 - DID 문서를 통한 검증 수행

- DID 기반 VC/VP 시스템에서는 앞서 서술한 설명에서의 검증을 수행하기 위해 발행인과 사용자(제출자)의 DID 문서에 접근한 후 VC, V

P에서 명시된 키를 그대로 활용하여 검증을 수행하게 된다

- VP의 proof 검증은 사용자가 VP를 생성할 때 서명한 proof에 사용된 Public Key가 사용자 DID 문서의 authentication 키에 포함되어 있는지 확인하고, 해당 Public Key로 VP 서명의 지위를 검증하여 VP를 제시한 사람이 DID의 소유자인지 확인한다
- VP 내의 VC proof 검증은 VC의 proof 필드의 verificationMethod에 명시된 키를 통해 발행인의 서명이 유효한지 검증해야 한다. 추가적으로 VC의 만료 여부와 VC가 검증자가 기대한 자격 증명의 내용을 포함하는지 확인해야 한다
- credentialSubject DID Auth는 VP에 포함된 VC가 사용자 본인과 관련된 자격 증명인지 확인한다. VC의 credentialSubject의 id 속성값이 VP 내 사용자 DID와 일치하는지 확인해야 한다

(그림4-9) 사용자의 DID 문서

(그림4-10) 발행인의 DID 문서

제 5 장 DID 관련 표준 및 서비스 동향

제 1 절 DID 관련 표준문서 및 워킹그룹 동향

1. DID-CORE (W3C Decentralized Identifiers v1.0) [4]

- ㅇ 2022년 7월 19일에 W3C에서 발표한 기술 사양으로 DID를 정의한다

- ㅇ W3C Recommendation (공식 권고안)으로, 광범위한 합의 과정을 거쳐 표준으로 채택된 문서이며, 이 문서는 특히 Web 상에서 중앙 권한 없이 신뢰할 수 있는 디지털 신원을 관리하는 방법을 다룬다

- ㅇ W3C의 Decentralized Identifier Working Group에서 작성되었으며, DID의 핵심 구조, 관련 문법, 데이터 모델, 표현 방식, 핵심 속성 및 검증 방법과 요구 사항 등 DID에 대한 표준을 제공한다

- ㅇ DID 문서의 각 속성값과 JSON-LD 등의 표현 방식에 관해서 서술하고 있으며, DID Resolution의 옵션과 메타데이터에 대한 상세한 설명도 제공한다

2. DID-SPEC (W3C DID Specification Registries) [7]

- ㅇ 2024년 8월 31일에 W3C에서 발표된 Group Note로, DID 생태계를 위한 알려진 확장 기능들을 정의한다

- ㅇ W3C의 Group Note로서, W3C의 공식 권고안이 아니며, 이 레지스트리는 현재 개발 중이다

- Decentralized Identifier Working Group에 의해 작성되었으며, DID 생태계에서 사용되는 모든 글로벌 매개변수, 속성 및 값들을 위한 공식 레지스트리이다

3. DID-USE-CASES (W3C Use Cases and Requirements for DID) [6]

- 2021년 3월 17일에 작성된 W3C의 문서로, DID에 관한 사용 사례와 요구 사항을 정의한다

- 공식 표준이 아닌 워킹그룹 노트 (초안 문서)로서, 언제든지 업데이트될 수 있는 작업 중인 문서이다

- Decentralized Identifier Working Group에 의해 작성되었으며, 탈중앙화(decentralized), 지속성 (persistent), 암호학적 검증 가능성 (cryptographically verifiable), 해석 가능성 (resolvable)의 필수적인 특성을 가진 새로운 유형의 식별자인 DID에 관한 사용 사례와 요구 사항을 설명한다

4. DID-IMP-GUIDE (W3C DID Implementation Guide v1.0) [18]

- 2023년 10월 12일에 작성된 W3C의 문서로, DID 개발자를 위한 DID 이해 및 구현 가이드를 제시한다

- 공식 표준이 아닌 워킹그룹 노트 (초안 문서)로서, 언제든지 업데이트될 수 있는 작업 중인 문서이다

- Decentralized Identifier Working Group에 의해 작성되었으며, 그룹 전체의 합의를 반영하는 것은 아니지만 일부 멤버가 기여한 지침을

제공하는 문서이다

- ㅇ DID Method 개발자들에게 직렬화, 암호화 선택, 거버넌스 및 상호운용성에 관한 지침을 제공한다

- ㅇ DID Method 구현을 위한 기술적 설명을 제공하고, 개발자가 고려해야 할 다양한 주제에 대한 가이드를 포함한다

5. Decentralized Identifier Working Group

- ㅇ DID 사양 및 관련 작업 그룹 노트를 유지 관리한다

- ㅇ 공통 요구 사항, 알고리즘. 아키텍처 옵션을 통해 DID Resolution 및 DID URL Dereferencing 프로세스의 상호운용성을 효과적으로 달성하는 방법에 대한 합의를 찾는다

- ㅇ 공개된 GitHub repositories에는 did-core, did-wg, did-rubric, did- extensions, did-test-suite 등이 있다

6. did:web Method Specification [19]

- ㅇ did:web 메소드 관련 내용을 포함하며, 2024년 7월 31일에 작성된 Unofficial Draft로 공식 문서 채택을 받지 않은 잠재적 사양의 초안이다

- ㅇ Web 도메인의 기존 평판을 활용하여 새로운 DID 메소드를 제안한다

7. did:dns Method Specification [15]

- ㅇ did:dns 메소드 관련 내용을 포함하며, 2022년 1월 4일에 작성된 Unoffici

al Draft로 공식 문서 채택을 받지 않은 잠재적 사양의 초안이다

- 기존 DNS 및 Web 서버 인프라를 기반으로 하는 did:web과 유사하지만 여러 가지 개선 사항을 제공하는 did:dns를 정의한다

8. did:key Method v0.7 [16]

- 2022년 9월 2일에 작성된 did:key 메소드 관련 내용을 포함하는 Unofficial Draft로, 공식 문서 채택을 받지 않은 잠재적 사양의 초안이다

- 암호화 공개 키를 DID 문서로 확장하는 것에 기반한 non-registry 기반 DID method를 설명한다

9. RFC 7553(The Uniform Resource Identifier(URI) DNS Resource Record) [20]

- DNS 리소스 레코드 유형 중 URI RR에 대해 설명한다. 2015년 6월에 작성된 Informational 라벨링 문서로서, 인터넷 표준 트랙 사양이 아니며 정보 제공 목적으로만 출판되었다

- 호스트 이름에서 URI로의 매핑을 게시하는 데 사용되는 URI RR에 대한 포맷과 사용 예시에 관한 설명을 포함한다

10. RFC 1464(Using the DNS To Store Arbitrary String Attributes) [21]

- 1993년 5월에 작성되었으며, DNS RR 유형 중 TXT RR에 관해 설명하는, 추후 발전되거나 폐기될 가능성이 있는 Experimental RFC 문서이다

- ASCII 텍스트 정보를 DNS에서 정의하지 않은 속성과 연결하는 간단

한 방법을 설명하고, DNS TXT RR을 통해 이러한 정보를 저장하도록 하는 내용을 포함한다

- ㅇ 기존 DNS 서버에 대한 변경을 필요치 않게 하는 이점을 가져오는 TXT RR 관련 접근 방식을 소개한다

11. VC-DATA-MODEL(W3C Verifiable Credentials Data Model v2.0) [17]

- ㅇ 2024년 10월 5일에 작성된 W3C Candidate Recommendation Draft 문서로, VC/VP 시스템의 구성 요소와 세부적인 개념을 설명한다

- ㅇ Verifiable Credentials Working Group에 의해 작성되었다

12. 분산 ID 기술 및 표준화 포럼

- ㅇ 우리나라의 DID 분야에서 정부의 시험사업 연계를 통해 DID 기능 요구사항 제정 및 글로벌 표준에 대응하기 위해 설립되었다

- ㅇ DID 분야의 국제표준에 대한 기술정보를 수집 및 제공하고, 표준 규격 개발 및 국내 산업체의 사실표준화 활동, 관련 제도 및 정책 방안 마련 등을 통해 국내 DID 분야 산업체의 국가 경쟁력을 제고하고 산업을 활성화하고자 하는 활동 목표가 있다

- ㅇ 주 연구 분야는 DID에 관한 용어 정의, 서비스 간 상호운용성 표준, 참조 아키텍쳐, 서비스 및 보안 요구사항, 활용 사례 등이 포함된다

- ㅇ DID 활성화를 위한 정책마련을 목표로, 보증 수준별 신원 증명 서비스를 발굴하고 DID 기반 신원관리 서비스 평가기준과 DID 보관 및 연계서비스 모델 요구사항을 마련하는 활동을 추진한다

- DID 상호 연동을 위한 기술마련을 목표로, DID 서비스 간 상호 연동 표준, DID 기반 증명서 규격 표준 및 DID 서비스 보안 요구사항을 연구 중이다

제 2 절 DID 플랫폼 동향

DID 플랫폼은 디지털 ID 관리에 있어 효율적이고 안전한 접근 방식으로, 2022년 6억 5천만 달러의 시장 규모에서 2025년까지 35억 달러에 이를 것으로 예상된다

1. DID 플랫폼의 필요성

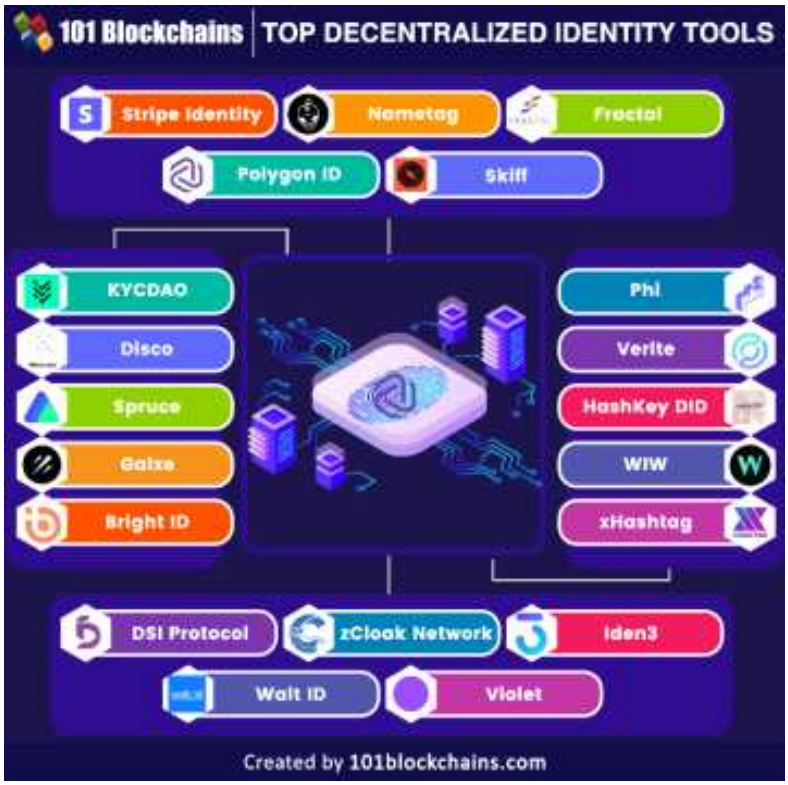
- 금융, 교육, 거버넌스, 의료와 같은 다양한 산업에 적용되고 있으며, 대부분의 DID 플랫폼은 블록체인 기술을 사용하여 데이터를 저장하고 관리하기 위한 보안을 용이하게 해준다

- 사용자가 DID 플랫폼을 통해 디지털 지갑에 관련된 자격증명, 식별자 및 데이터 보관 등 개인 데이터와 그 사용방법에 대한 포괄적인 제어권을 행사할 수 있다

- 개인 정보 보호를 함과 동시에 자신의 신원을 제어할 수 있으며, DID는 특히 블록체인 기술을 사용하여 데이터 해킹 및 잠재적 위협을 줄일 수 있다. 특히 Central DB와 같이 Single Point of Failure이 없는 것이 가장 큰 장점이다

2. 주요 DID 플랫폼

- 101 blockchains에서는 가장 유망한 상위 20가지 DID 플랫폼을 제시하였다



(그림5-1) 상위 20가지 DID 플랫폼(blockchain 101)

○ (Stripe ID entity) 다양한 ID 문서의 진위성을 확인하는데 도움이 되는 DID 도구로, 33개국 이상에서 사용되고 있다

- 사용자를 사기로부터 보호하기 위해 사용자 ID를 프로그래밍 방식으로 확인할 수 있다
- 개발자가 Advance Machine Learning을 사용하여 KYC rule을 지원하고, fake ID를 감지할 수 있게 하며, 고객의 마찰을 줄이는 데에 도움을 준다

○ (Name Tag) 분산형 소셜 네트워크에 적합한 DID 플랫폼 중 하나

- Web3 & Web2 service accounts에 연결하는 데 도움을 주어 사용자를 위한 분산형 소셜 미디어 프로필 역할을 수행한다

- Name Tag Profile은 다른 여러 chain에 거쳐 wallet들을 연결하는데 도움을 주며, 본인의 gallery를 생성할 수 있다

○ (Fractal) crypto-native identity provider로 사용될 수 있는 DID 플랫폼 중 하나
- Tool을 사용하는 160개 이상의 APP과 약 100만개의 재사용 가능한 검증된 ID를 보유하고 있다
- KYC Checks, KYB Verifications와 같은 다양한 규정 준수 관련 사용 사례에 대한 온보딩 기능을 제공한다
- 암호화폐 게임, 암호화폐 에어드랍 참여 등 특정 이벤트에 대한 사용자의 적격성 확인에 도움을 준다

○ (Polygon ID) Web3와 신뢰할 수 있는 커뮤니케이션을 만드는 데 도움을 주는 DID 플랫폼
- privacy에 이점을 갖춘 DID 플랫폼을 제공하여 사용자가 분산되고 privacy 중심의 다양한 Web3 서비스와 상호작용할 수 있게 한다
- 암호화 보호 기능을 용이하게 하기 위해 zk-SNARK 기술을 사용한다

○ (Skiff) End-to-End 분산형 작업 공간을 제공하는 DID 플랫폼
- 효과적인 협업과 커뮤니케이션을 위한 암호화의 이점을 갖추었으며, 메일, 드라이브 앱, 캘린더와 같은 필수적인 workspace 기능이 포함되어 있다
- 개인 정보 보호에 중점을 둔 wallet 기반 workspace의 편의성에서 장점을 보인다

○ (KYCDAO) Web3 native compliance service 도입으로 명성을 얻은 DID 플랫폼
- 재사용 가능한 on-chain KYC 검증을 발행하는 멀티체인 플랫폼으로 작동한다

- 다양한 사례에서 신원 검증을 위한 신뢰할 수 있는 접근 방식을 제공하며, 여러 wallet에 KYC NFT 주조 기능을 제공한다

ㅇ (Disco) 가명 식별 서비스를 위한 최고의 DID 플랫폼
- 원하는 곳 어디든 자격 증명을 가져갈 수 있는 유연성이 있으며, chain에 독립적인 DID 플랫폼을 제공한다
- 데이터 소유권을 위해 검증 가능한 자격 증명과 함께 분산형 식별자를 활용한다
- 개인화된 데이터를 그룹화하여 사용하며, 사용자는 검증자와 공유 여부를 선택할 수 있다

ㅇ (Spruce) 사용자가 신원을 제어하는데 도움이 되는 오픈소스 도구 모음을 제공하는 DID 플랫폼
- 소셜 미디어용 암호화 자격 증명과 자격 증명 기반 지갑을 제공한다. 이더리움 지갑 주소와 같은 분산형 식별자를 통해 사용자가 데이터를 제어할 수 있도록 하는 것을 목표로 한다

ㅇ (Galxe) Web3 개발자에게 접근 가능한 자격 증명 인프라를 제공하는 DID 플랫폼
- 온체인과 오프체인 커뮤니티의 요구사항을 해결할 수 있는 자격 증명 데이터 네트워크를 만든다
- 특정 네트워크에 디지털 자격증명을 큐레이팅하고 기여하는데 도움을 주며, 온체인 자격 증명의 경우 Galxe는 정적 스냅샷 또는 하위 그래프 쿼리를 사용한다
- Github, Snapshot, Twitter와 같은 다양한 조직의 데이터 소스를 활용한다

ㅇ (Bright ID) 소셜 미디어의 신원 확인에 대한 새로운 접근 방식을 시도하는 소셜 신원 네트워크
- 비침습적, 오픈 소스 분산 기술을 활용하며, 소셜 그래프를 개발하

고 분석하여 고유한 신원 증명을 만든다

ㅇ **(Phi)** 사용자의 온체인 활동을 나타낼 수 있는 virtual land를 생성하여 작동하는 DID 플랫폼
- 지갑 활동 및 ENS 도메인과 같은 범용 Web3 구성 요소에 따라 virtual land parcel을 만들 수 있는 메타버스 virtual land 시스템을 제공한다

ㅇ **(Verite)** 민감한 개인 정보를 공개하지 않고 Web3 ID 클레임을 검증하는 데 도움을 주는 DID 플랫폼
- Smart Contract, Web 및 모바일 앱을 위한 포괄적인 ID verification 도구를 제공하며, ID 속성 공유에 있어 접근 방식, 위치 및 타이밍을 제어하는 데에도 도움을 준다

ㅇ **(HashKey DID)** 크로스 플랫폼 신원 인증 검증 도구
- 약 840,000명의 사용자와 800만 명 이상의 커뮤니티 멤버가 존재한다
- HashKey는 여러 브랜드와의 안전한 협업을 제공하여, KYC 통합을 더 쉽도록 만들어준다

ㅇ **(xHashtag)** 온체인 자격 증명을 쌓고 획득하기 위한 효과적인 플랫폼
- Web3 세계에서 매우 중요한 자격 증명 기반 증명이 필요한 온체인 reputation을 지원한다
- “.soul” 도메인 이름은 soul-bound NFT를 통해 사용자의 온체인 활동 증명을 나타내도록 한다

ㅇ **(WIW)** 영지식 증명(ZKP) 기술을 활용하는 DID 플랫폼
- WIW의 ZKP 기술은 사용자의 온체인 평판을 관리하는 것과 함께 사용자의 ID를 보호하는데 도움을 준다
- IPFS와 같은 분산형 파일 저장 시스템에 사용자 데이터를 저장하며,

모든 사용자의 프로필 태그에 대한 세부 정보를 찾을 수 있는 검색 포털을 제공하여 온체인 평판을 확인할 수 있도록 한다

ㅇ **(DSI Protocol)** 데이터 관리 및 소유권을 위한 포괄적인 분산형 프로토콜
- 온체인 및 오프체인 데이터에 액세스하기 위한 완전한 소유권과 권한을 얻기 위한 풍부한 선택을 제공한다
- 사용자가 Web3 자격증명과 DeFi 포트폴리오를 쉽게 저장하고 관리하는데 도움을 준다

ㅇ **(zCloak Network)** 사용자가 특정 서비스에 로그인하기 위해 명확하고 빠르며, 개인적인 매커니즘을 달성하도록 돕는 zkID 로그인 도구를 제공하는 플랫폼
- 다중체인 호환성의 이점을 누릴 수 있으며, 현재 zkID 로그인은 브라우저 확장 프로그램 형태로 액세스할 수 있다
- 사용자는 전용 지갑에 검증 가능한 자격 증명을 저장할 수 있다

ㅇ **(Iden3)** 이더리움을 기반으로 개인정보 보호 및 익명의 온체인 활동을 지원하는 플랫폼
- zk-SNARK 기술을 활용하여 개인정보 보호를 수행하며, layer-2 솔루션으로 확장할 수 있는 유연성 또한 제공한다

ㅇ **(Walt ID)** NFT 및 지갑 인프라와 함께 DID 서비스를 제공하는 플랫폼
- Walt ID의 SSI 기트는 Self-Sovereign Identity(SSI)를 활용하기 위한 유연하고 편리한 접근 방식을 제공한다

ㅇ **(Violet)** ID 관리 및 규정 준수를 위한 포괄적인 플랫폼
- Violet은 규정 준수 및 ID 관리에서 더 나은 효율성을 보장하는 주요 Application 중 하나로, 사용자의 프라이버시를 보장하기 위해 영지

식 증명의 기술적 측면을 활용한다
- 원활한 오프체인 데이터 저장 시스템은 Violet 사용자의 데이터 안전을 보장한다

제 3 절 DID 기반 서비스 동향

DID 기반 서비스는 주로 디지털 신원 관리를 중점으로 국내외 여러 대기업과 기관의 주목을 받고 있다. 삼성, LG, SK텔레콤 등 주요 기업들은 블록체인과 DID 기술을 활용하여 개인이 직접 통제할 수 있는 신원 인증 시스템을 구축하고자 여러 연구를 시도 중이며, 이와 더불어 정부 기관 주도의 DID 표준화 작업과 함께 DID를 적용한 디지털 신분증의 사례가 점차 확대되고 있다

1. DID 기반 모바일 운전면허증 TTA 표준화

- 행정안전부/과학기술정보통신부 주도하에 DID 기술 및 표준화포럼, 한국인터넷진흥원, 라온시큐어 등 전문가들의 협력을 통해 표준을 제안하였다

- 한국정보통신기술협회 (TTA)는 온라인에서도 신분증으로 이용할 수 있고, 사용자가 원하는 정보만 선택적으로 제공할 수 있는 DID 기반 모바일 운전면허증 표준화를 진행하였다

(그림5-2) DID 기반 모바일 운전면허증(한국정보통신기술협회)

2. Orange Protocol

ㅇ 2024년 5월에 열린 연례 MicroStrategy World Conference에서 Micro Strategy는 open-source MicroStrategy Orange DID protocol을 새롭게 공개하였다

ㅇ Orange Protocol을 통해서 사용자는 이메일을 사용하여 비트코인 주소와 연결된 공개 및 개인 키로 신원을 검증할 수 있고, 학위 인증과 같은 정보도 안전하게 기록된다

ㅇ 함께 공개된 비공식 초안 문서에 따르면 이 DID 프로토콜은 Ordinals 방식을 수정하여 적용하지만, DID 관련 데이터만 저장한다. 이를 통해 비트코인의 SegWit 기능을 활용함과 동시에, 내용에 대한 큰 제한 없이 문서를 생성하고 업데이트할 수 있다

3. 디지털 인증

ㅇ 라온시큐어에서는 '옴니원 디지털아이디'라는 플랫폼을 제공한다

- 이는 블록체인 기술과 FIDO 기반의 편의성을 결합한 최초의 DID 공개 플랫폼으로, 블록체인 기술을 활용하여 안전하고 신속한 디지털 신분증 서비스를 제공하며, 한국의 모바일 신분증 구축에서도 대표적인 사례로 평가받고 있다

- 이 기술은 블록체인 기술로 디지털 신분증을 분산하여 저장함과 동시에 참여자 간 Public Key 기술을 활용한 상호 신뢰를 형성하여 안전한 데이터 보호를 가능하게 한다

- 라온시큐어는 2024년 후반기와 2025년에 DID 기술을 오픈소스화하여 K-DID 오픈소스 개발자 생태계를 확보하는 데 집중하고 있고, 해외 정부와 기타 기관 등과도 협력할 계획임을 발표하였다

4. 디지털 제품 여권 (DPP, Digital Product Passport)

- 디지털 제품 여권(DPP)은 EU가 추진 중인 유통 제도로, 유통되는 모든 제품의 생애주기 정보를 디지털로 수집, 저장, 공유하는 제도이다

- DPP의 핵심 기술인 DID는 제품에 대한 고유한 디지털 식별자 역할도 가능하여, 암호화 기술을 통해 검증 가능하고 지속적인 추적이 가능하다

- EU에서 모든 물리적 제품에 DPP를 적용 의무화하여 빠르면 2026년부터 EU가 정한 우선 순위 산업 분야에 적용을 시작하기로 발표되었으며, 배터리의 경우 이미 디지털 여권 제도 시행이 확정되었다

- 2024년 7월, 국내 기업 케이포시큐리티는 DID 기술을 통해 디지털 신원

을 넘어 제품의 진위성까지 검증하는 기술을 개발했다고 밝혔다

- 2024년 8월에는 케이포시큐리티에서 다뉴브테크(Danube Tech GmbH)와 협력하여 DPP 전용 제품 신뢰 서비스(PTS) 베타 버전을 선보였다. PTS는 제품의 고유 식별자를 생성하고, 이를 통해 제품의 진위를 확인하는 동시에 DPP에 연결된 제품 및 공급자 정보를 신속하고 안정하게 검증할 수 있는 기능을 제공한다

- 또한, 이러한 서비스 개발과 적용을 통해 규제 대상인 제품들에 대해 상호운용성과 글로벌 표준화 기술이 보장된 제품 신뢰 서비스를 제공할 것이라고 밝혔다

5. 국내 기업체 동향

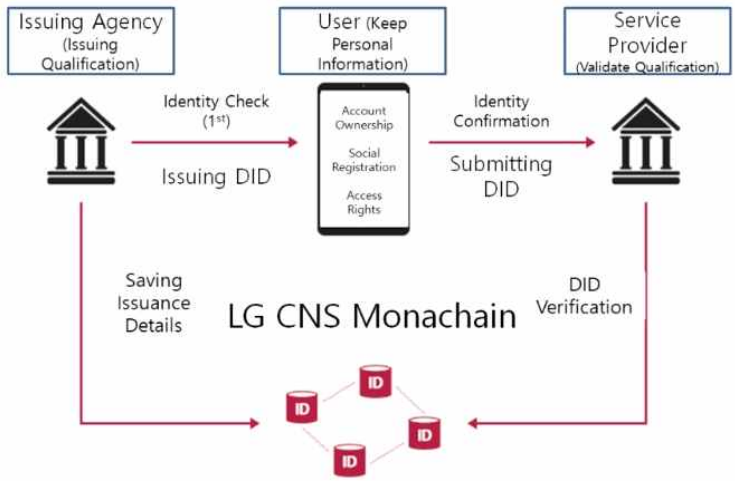
- 2019, 2020년부터 LG, SK텔레콤, 삼성 등 국내 대기업은 블록체인 기반 신원 서비스에 관심을 갖고 SSI 앱을 구축하였다

- 2020년 기준, 최초로 한국 내에 대학 위주로 신원 서비스가 도입되어 취업 신청 시 지원자가 SSI 서비스를 이용하여 졸업 증빙을 할 수 있도록 하였다

- LG U+는 분산형 손실 보험 서비스를, KT는 SSI 프로젝트를 블록체인 제품에 통합하는 등의 체계 구축을 통해 DID 활용 기술 개발을 시도하였다

- 삼성은 자체 탑재 보안 계층인 Knox를 통해 DID를 관리하는 시스템을 사용하고 있으며, 스마트폰에서 블록체인 기반 키를 보호하는 데에 사용되고 있다

○ LG CNS는 2020년에 블록체인 기반 DID 기술 서비스 전문 기업인 Evemym과 DID 글로벌 표준을 확립하기 위한 MOU를 체결하였다



(그림5-3) LG CNS의 글로벌 인증 관련 DID 개념도

6. 국내 공공기관 동향

○ 2020년 데이터 3법 발의로 인해 개인의 동의 하에 여러 개인정보 내역을 통합 관리할 수 있는 마이데이터 산업이 가능해짐과 더불어, 공공기관의 DID에 대한 관심도가 높아지고 있다

○ 과학기술정보통신부는 DID 얼라이언스 코리아 2020에서 분산신원증명 생태계 활성화 정책을 주제로 발표를 진행하였으며, DID 통합 해석기 연구 개발 및 DID 상호호환 논의를 본격화 하였다

○ 한국인터넷진흥원은 2018년부터 블록체인 지원사업을 진행하고 있으며, 2020년 DID 기반 디지털 공공서비스 플랫폼 구축 시범사업 진행 이후 확산사업을 진행하여 2023년 기준 약 369억원의 생산유발효과를 달성하였다

(그림5-4) 블록체인 지원사업 현황(한국인터넷진흥원)

ㅇ 최근 국내 기업인 파라메타와 핑거랩스는 과학기술정보통신부와 한국인터넷진흥원이 추진하고 있는 2024년 블록체인 민간분야 확산사업의 일환인 공공용 블록체인 공동 인프라 구축 사업을 수주하였으며, 2025년 도입을 목표로 DID 신원인증 플랫폼 마이아이디 및 NFT 기반 온·오프라인 고객관리 솔루션 페이버렛을 토대로 DID 및 NFT 공동 플랫폼 서비스를 개발하고 있다

제 6 장 DID 직접해석

제 1 절 DID 직접해석 방안 개요

1절에서는 DID 직접해석에 대한 전반적인 개요를 설명한다

1. DID 직접해석 방안 개념

ㅇ 특정 DID를 Web 브라우저 등에 입력했을 때, 각 DID Method 별로 요구되는 해석기 없이 DID 문서를 반환할 수 있는 해석 방법을 의미한다

ㅇ 기존 DID 해석 방식은 아래와 같은 한계점이 존재한다
- DID Method 별로 동작 방식이 상이하여 신뢰 기반, 필요 인프라 등이 다르고, DID Method 간 상호운용성이 저하될 수 있다
- 특정 해석기, 플랫폼에 대한 의존성 증가로 인한 탈중앙화 원칙 위배의 가능성이 존재한다
- 새로운 DID Method가 추가될 때마다, 새로운 모듈 개발 및 관리가 필요하여 복잡성이 증가된다

ㅇ DID 직접해석은 기존 해석 방식의 한계점 해결을 목표로 기존 DID 해석 모델을 보완하고 보다 자유롭고 신뢰성 있는 DID 생태계를 구축하기 위해 설계된 해석 방법이다. 단일 인터페이스를 통해 서로 다른 DID Method가 입력되었을 때, 각 DID Method 별로 요구되는 해석기 없이도 DID 문서를 반환할 수 있도록 하여 상호운용성을 보장하면서도, 사용자나 시스템이 특정 Method의 인프라에 종속되지 않게끔 한다

- DID 직접해석은 Web Application Server(WAS), DNS, 블록체인 및 신규 DID Method 혹은 신규 Resolver 등 다양한 방식의 기술적 접근이 가능하다

- 현 단계의 DID 직접해석은 기술적 접근 방안으로써, DID를 입력했을 때, DID 문서를 반환받을 수 있는 과정에 대해 설명한다

2. 기대 효과

- **(표준화 기여)** W3C, IETF 등 국제 인터넷 표준화 기구에서 요구하는 탈중앙화를 고려하고 상호운용성을 보장함으로써 DID 표준화에 기여할 수 있다

- **(적용 산업 확대)** 공공, 금융, 의료 등 DID가 적용될 수 있는 산업 분야에서 상호운용성을 보장할 수 있으며, 다양한 산업 환경에서 DID 기술 활용이 가능하다

- **(기술 개발 촉진)** 특정 해석기, 플랫폼에 대한 의존성 없이 다양한 방법으로 DID Method를 개발하고 호환성이 보장됨으로써, 기술 제약 없이 기술 개발 및 사용 사례를 촉진할 수 있다

제 2 절 WAS 활용 직접해석 방안

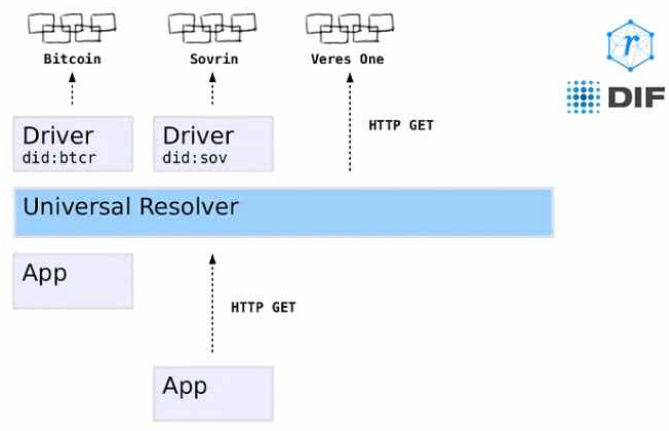
2절에서는 DID 직접해석의 접근 방법 중 하나로 기존에 제안되었던 Universal Resolver [22]를 Web Application Server (WAS)로 활용하는 직접해석모델에 대해 설명한다

1. Universal Resolver

- ㅇ Universal Resolver는 다양한 DID Method에 대해 DID 문서를 해석할 수 있는 통합 해석기 역할을 수행한다

- ㅇ 다양한 DID Method를 통합 관리하는 시스템 구성 요소로써, 서로 다른 검증 가능한 데이터 저장소로부터 각각 정의되는 DID Resolver의 모음으로 정의된다

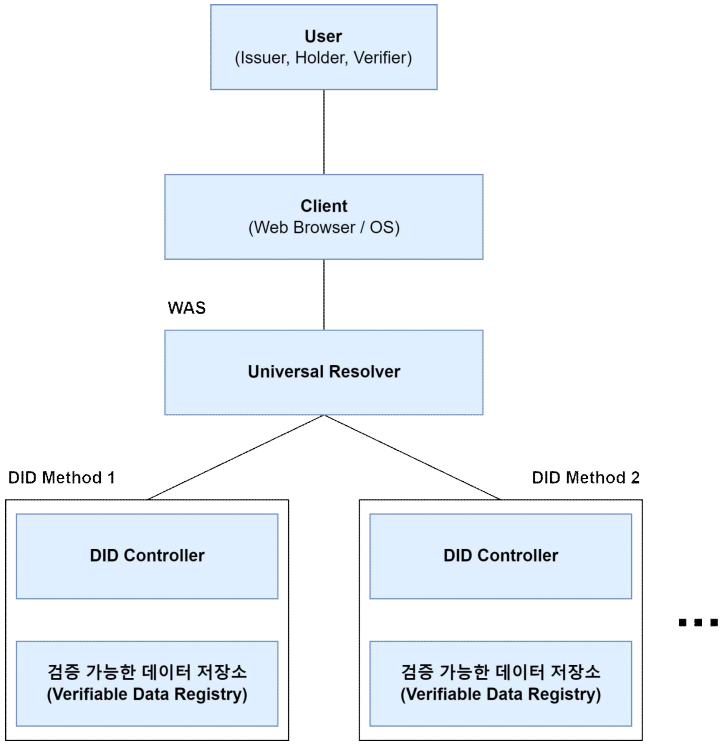
- ㅇ 여러 DID Method에 대한 해석 방법을 드라이버 모듈로 통합하여 단일 인터페이스를 통해 다양한 방식으로 구현된 DID Method들을 해석하고 DID 문서를 반환하는 기능을 제공하는 방식으로 DID Method 간 상호운용성을 제공한다



(그림6-1) Universal Resolver 구조

2. WAS 방식 DID 직접해석 모델

ㅇ Universal Resolver를 WAS로 활용하는 DID 직접해석 모델은 그림 6-2와 같다



(그림6-2) Universal Resolver 기반 DID 직접해석 모델 구조

ㅇ 해당 직접해석 모델에서 Universal Resolver의 동작 프로세스는 아래와 같다

- (1) 사용자는 특정 DID를 Web 브라우저를 통해 Universal Resolver에 전달한다
- (2) Universal Resolver는 DID Method를 분석하여 적합한 DID driver를 선택한다
- (3) 선택된 Driver는 블록체인 노드, 중앙 서버 등 해당 DID의 정보

를 조회하고 Verifiable Data Registry와 통신하여 DID 문서를 생성하고 사용자에게 반환한다

ㅇ Universal Resolver를 WAS로 활용하는 방안은 각 DID Method가 기존에 사용하던 형식의 식별자를 그대로 단일 인터페이스를 통해 입력받고, DID 문서를 반환함으로써 편리한 통합 해석 방안을 제공할 수 있다는 장점이 있다

ㅇ 반면에, 각 DID Method를 위한 driver는 Universal Resolver에 종속되어 드라이버 개발 및 유지 관리의 복잡성이 상승하고, Universal Resolver에 대한 의존성이 높아지게 된다. 이는 DID 생태계의 탈중앙화 원칙과 상충될 수 있다

제 3 절 블록체인 활용 직접해석 방안

3절에서는 블록체인을 분산 디렉토리 서비스로 활용하는 직접해석모델에 대해 설명한다

1. 개요

ㅇ 블록체인을 활용한 직접해석 방안은 별도의 DID Resolver 네트워크를 블록체인 기반으로 구축하여 분산 디렉토리 서비스 역할을 수행하도록 하고, 직접해석을 탈중앙화된 방식으로 처리하는 것을 목적으로 설계되었다

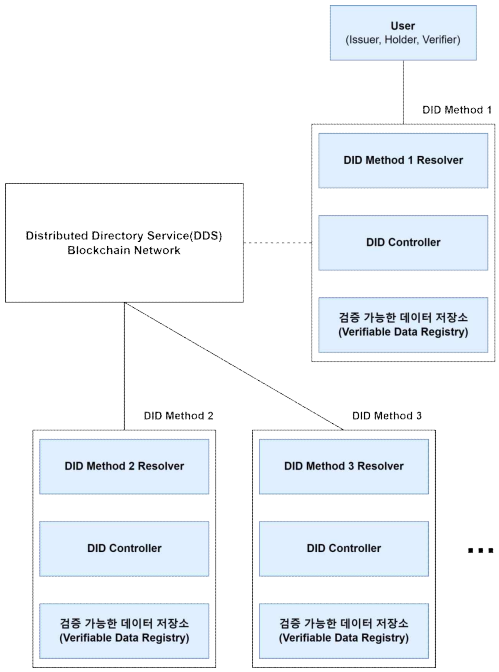
ㅇ DNS 대신 블록체인 네트워크를 활용하여 DID Method 별 DID Resolver 주소를 탈중앙화된 방식으로 등록 및 관리할 수 있다

ㅇ 각 DID Method에 대한 Controller가 해당 네트워크의 노드로 참여 및 Smart Contract를 통해 DID Method와 DID Resolver 간의 매핑 정보를 저장할 수 있으며, 이를 통해 데이터의 무결성과 신뢰성을 보장할 수 있다

ㅇ 사용자의 DID를 입력받은 DID Resolver에서 DID의 해석이 가능할 경우, 직접 DID 문서를 해석하여 반환하며, 해석이 불가능할 경우 블록체인 네트워크에 질의하고 Resolver 주소에 HTTPS 리다이렉션을 통해 DID 문서를 해석하고 반환한다

ㅇ 새로운 DID Method를 추가하고자 할 때는 해당 정보를 블록체인 네트워크에 등록하여 즉각적인 상호운용성을 제공할 수 있다

2. 블록체인 방식 DID 직접해석 모델



(그림6-3) 블록체인 기반 DID 직접해석 모델 구조

ㅇ 블록체인 네트워크를 분산 디렉토리 서비스로 활용하는 DID 직접해석 모델은 그림 6-3과 같다

ㅇ 해당 직접해석 모델에서 DID 질의 및 해석 프로세스는 아래와 같다

- (1) 사용자는 DID를 특정 블록체인 네트워크를 기반으로하는 DID Resolver 에 질의한다. 해당 DID Resolver는 입력받은 식별자 구조를 기반으로, 해석 가능 여부를 확인한다
- (2) 질의받은 DID 식별자가 해석이 가능할 경우, 해당 블록체인 네트워크를 활용하여 직접 DID 문서를 반환한다
- (3) 해석이 불가능할 경우, 분산 디렉토리 서비스 역할을 수행하는 블록체인 네트워크에 DID 식별자를 질의하여 적합한 Resolver 주소를 조회한다
- (4) 조회된 DID Resolver 주소로 HTTPS 요청을 리다이렉트하여 DID를 전달하고, 해당 DID Resolver는 DID를 해석하고 DID 문서를 획득하여 반환한다

ㅇ 블록체인 네트워크를 기반으로 DID Resolver 주소를 관리하는 분산 디렉토리 서비스 방식 직접해석은 중앙 서버 없이도 데이터의 무결성과 신뢰성을 보장하며, 새로운 DID Method 추가 및 DID Resolver 변경 등 확장성이 뛰어나다는 장점이 있다

ㅇ 하지만 초기 네트워크 구축 및 Smart Contract로 인해 발생하는 비용 부담이 높고, 참여자가 많아질수록 블록체인 노드와 트랜잭션 부하가 증가하여 DNS 대비 네트워크 성능이 낮다는 단점이 존재한다. 또한 Smart Contract 오류 및 취약점에 민감하고, HTTPS 리다이렉션 및 서비스 제공자 간 신뢰 문제가 필수적으로 고려되어야 한다

제 4 절 DNS 활용 직접해석 방안

4절에서는 DNS 기반 신규 DID Method를 활용하는 직접해석 모델에 대해 설명한다

1. 개요

ㅇ 해당 DID 직접해석 방식은 DNS로부터 반환받는 TXT RR에 추가적인 메타데이터를 포함시켜 확장 및 다양한 신뢰 기반을 아우를 수 있는 방식으로 동작한다

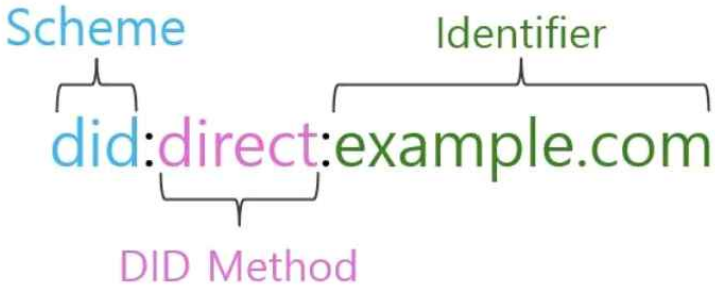
ㅇ 기존 DNS 인프라를 활용할 수 있고, 기존 인프라와 손쉬운 통합 및 간단한 구현이 가능함과 더불어 새로운 DID Method를 제안할 수 있다

ㅇ **(Resource Record)** IETF RFC 표준에 정의된 바를 따르며, 해당 방법을 통해 DID 문서를 저장하고 공개한다

- **(URI RR)** 도메인 이름에 DID 문서의 URI를 저장하는 방식으로 도메인 이름과 DID를 연결한다
- **(TXT RR)** 텍스트 정보를 저장하고 리소스에 연결할 수 있는 DNS 레코드 유형으로서, DID 문서 텍스트를 TXT 레코드에 포함시키는 방식으로 활용할 수 있다. TXT RR 구조를 확장하기 위해 DID Controller는 DNS Zonefile에 추가적인 정보를 입력해야 한다. DNS 서버는 복잡한 DID 문서를 담도록 하기 위해 충분한 크기의 DNS 레코드를 지원할 수 있어야 한다

2. DNS 방식 DID 직접해석 모델

ㅇ **(식별자)** 전반적인 구문은 W3C 표준문서 내 식별자 구문을 따르며, 식별자 구조는 “did:direct: [Domain Name]” 을 사용하며 예시는 그림 6-4와 같다



(그림6-4) did:direct 식별자 예시

- 소문자로 작성된 “did:direct” 와 함께 그 사용 목적을 밝히는 도메인을 뒤에 붙여 작성해야 하며, 도메인 네임은 Fully Qualified Domain Name (FQDN) 형식으로 구성되어야 한다

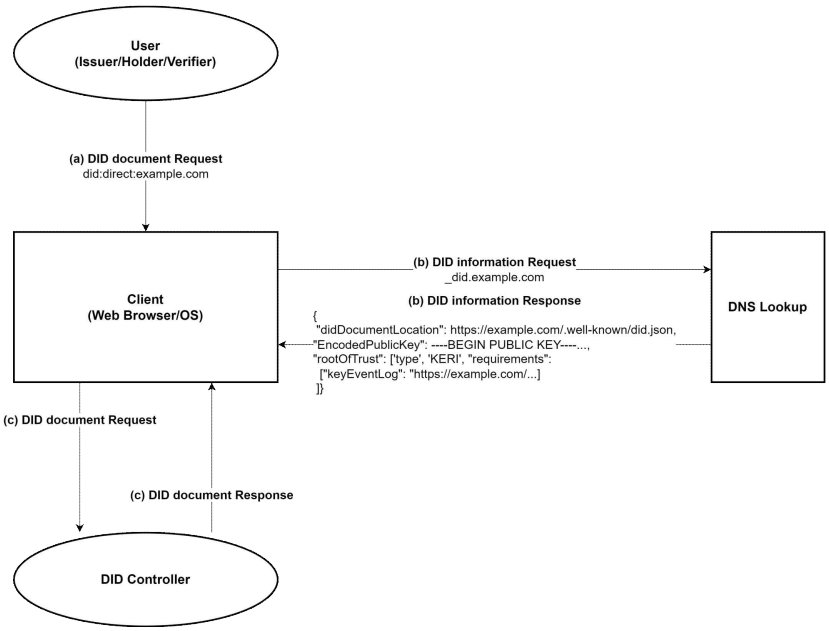
```
did-direct = “did:direct:” domain

domain = *(ALPHA/DIGIT/ “-” / “.” );
```

(그림6-5) did:direct 식별자 ABNF rules

ㅇ 해당 직접해석 모델에서 DID 문서를 반환하는 프로세스는 아래와 같다

- (1) DID를 입력하면 해당 DID의 “did:direct” 를 제외한 도메인 네임을 DNS Lookup 서비스에 질의하기 위한 형태로 변경해야 한다. 일반적이지 않은 도메인 네임임을 명시하기 위해 도메인 네임의 앞에 “_did” 를 붙여 질의해야 한다
- (2) Parsing된 데이터는 DNS Lookup 서비스를 통해 등록된 도메인을 찾아갈 수 있으며, 이를 위해 DID Controller는 TXT 형식의 DNS Zonefile을 등록해야 한다. DNS Zonefile이 등록된 서버는 255 b

yte 이상의 TXT 타입 RR을 위해 Extended Mechanism for Doma in Name System (EDNS)를 지원해야 한다



(그림6-6) did:direct 직접해석 절차

did:direct:example.com -> _did.example.com

(그림6-7) did:direct 식별자 parsing 예시

- (3) 클라이언트는 반환된 URI를 사용하여 DID 문서를 가져오고, 문서에 포함된 정보를 parsing 및 “rootOfTrust” 배열에 명시된 신뢰 기반을 사용하여 무결한 DID 문서를 검증할 수 있다. 기존 Web 기반 DID Method와 구분되는 구성요소는 서명된 DID 문서의 위치를 나타내는 “didDocumentLocation”, Base64로 인코딩된 Public Key를 나타내는 “EncodedPublicKey”, 신뢰기반을 명

시하는 “rootOfTrust”가 있다

```
$ORIGIN example.com
$TTL 86400
@       IN      SOA     dns1.example.com.  hostmaster.example.com. (
                                200162501 ; serial
                                21600   ; refresh after 6 hours
                                3600    ; retry after 1 hour
                                604800 ; expire after 1 week
                                86400)  ; minimum TTL of 1 day
_did  IN   TXT   “---user defined information---”
```

(그림6-8) DNS Zonfile 예시

- (4) RR을 기반으로 DID 문서가 저장된 주소 및 Public Key, 신뢰기반 정보를 추출하여 무결한 문서를 검증하고 반환할 수 있다

```
{
   “didDocumentLocation” : “https://example.com/.well-known/did.json
   “EncodedPublicKey” : “-----BEGIN PUBLIC KEY-----
                          .-----END PUBLIC KEY-----”
   “rootOfTrust” : [
     “type” : “KERI” ,
     “requirements” : {

 “keyEventLog” : “https://example.com/.well-known/1234/keri.cesr”
    },
     “type” : “certificate” ,
     “requirements” : {
```

```
        “certificate” : “https://example.com/.well-known/cert.pem”
    },
     “type” : “blockchain”,
     “requirements” : {

 “rpcUrl” : “https://example-RPC-URL-for-blockchain-network”,
        “queryString” : “exampleQueryString”
    },
  ]
}
```

(그림6-9) 반환받는 RR 예시

ㅇ 해당 DID 직접해석 방안의 다양한 신뢰기반에 따른 인증 고려사항은 다음과 같다

- **(KERI 신뢰기반)** requirements 항목에 keri.cesr에 대한 위치 정보인 “keyEventLog”가 반드시 존재해야 한다. 처리된 keri.cesr을 통해 현재 Public Key가 유효함을 검증할 수 있으며, 서명한 DID 문서 역시 유효함을 확인할 수 있다
- **(인증서 신뢰기반)** 사용자가 발급받은 인증서의 위치 정보인 certificate가 존재해야 한다. 인증서에는 Certificate Authority (CA)가 서명한 Public Key 정보가 담겨있어야 한다. 사용자에게 존재하는 CA의 정보로 서명된 Public Key를 복호화하여 RR에서 추출된 정보와 대조하는 방식으로 DID 문서의 유효성을 확인할 수 있다
- **(블록체인 신뢰기반)** requirement 항목에 질의를 위한 RPC URL 정보 및 네트워크에 질의할 완전한 형태의 쿼리가 제공되어야 한다. 해당 질의를 통해 저장된 DID 문서를 획득하고, publicKey를 사용해 복호화한 문서와 비교하여 유효성을 확인할 수 있다

ㅇ DNS를 기반으로 RR 구조를 확장하여 여러 신뢰기반에 대한 호환성을 제공하는 직접해석 방안은 기존 DNS 인프라를 활용할 수 있고,

DNS Zonefile 업데이트 및 RR 확장이라는 간단한 방식으로 구현이 가능하다는 장점이 있지만, DNS를 사용함으로 인해 서버 운영자 등 관리 주체에 종속될 수 있으며 스푸핑 등 보안 취약점을 고려해야 한다는 단점이 존재한다

제 5 절 DID Common Resolver 활용 직접해석 방안

5절에서는 새로운 Resolver인 DID Common Resolver를 기반으로 하는 직접해석모델에 대해 설명한다

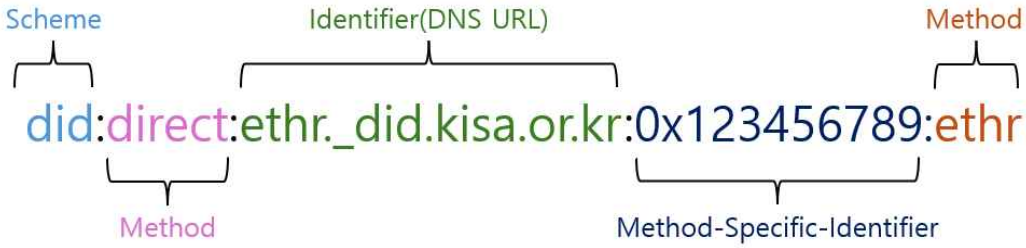
1. 개요

- 해당 DID 직접해석 방식은 새로운 클라이언트에 DID Common Resolver를 배치하는 방식으로, 중앙화된 해석기에 대한 의존성을 낮추고 탈중앙화 원칙과 상충되지 않도록 설계되었다

- DID Resolver의 집합으로써 작동하는 기존 방법 대비, DID Common Resolver는 사용자가 Web 브라우저에 DID를 입력하면, 문자열 parsing 및 HTTPS URL 변환을 통해 DID Controller에게 DID 문서 해석을 요청하는 API 역할을 수행하여, 해석기 없이 DID 직접 해석이 가능하도록 설계되었다

2. DID Common Resolver 방식 DID 직접해석 모델

- **(식별자)** 구조는 그림 6-10과 같다. “did:direct: [DNS-URL] : [Method - Specific-Identifier] : [DID Method/Blockchain 정보] 로 구성된다



(그림6-10) 직접해석 Identifier 구조

- **(did:direct)** W3C에서 정의한 DID 표준을 준수하여 소문자로 구성된 Scheme인 did와 Method 이름인 direct가 필수로 포함되어야 한다
- **(DNS-URL)** DNS 관련 표준에서 정의된 형식의 DNS URL 구조로, 일반적이지 않은 도메인 네임임을 명시하기 위해 ” _did “가 포함되어야 한다
- **(Method-Specific-Identifier)** 특정 DID Method로 구성된 DID 문서를 해석 요청하기 위해 포함되어야 하며, W3C에서 정의한 DID 표준을 준수하는 Method에 대한 식별자를 나타낸다
- **(DID Method/Blockchain 정보)** 선택사항으로, 특정 DID Method 이름을 표기해야 하며, 블록체인 네트워크를 신뢰기반으로 사용하는 경우 해당 네트워크 이름을 표기해야 한다

ㅇ 식별자 구조에 대한 ABNF rules는 그림 6-11과 같다

```
did-direct = “did:direct:” DNS-URL “:” Method-Specific-Identifier“:” DID Method
DNS-URL = 1*(ALPHA/DIGIT/“.”/“-”) “_did”;
Method-Specific-Identifier = “0x”1*HEXDIG;
DID Method = 1*ALPHA;
```

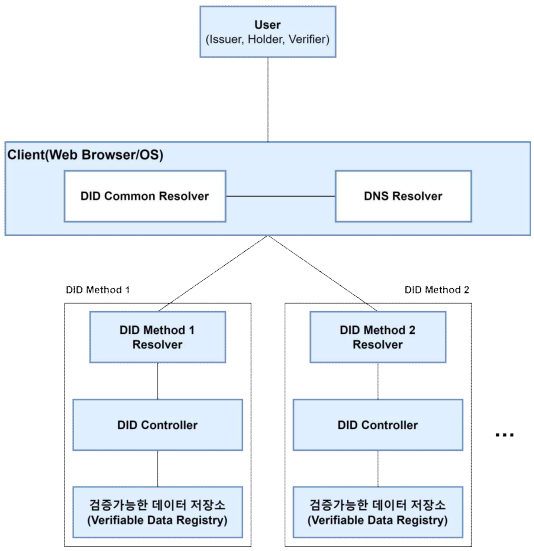
(그림6-11) 직접해석 Identifier ABNF rules

ㅇ DID Common Resolver를 활용한 DID 직접 해석 모델은 기존 대비 식별자를 Web 브라우저, OS 등 클라이언트에 입력하고 식별자를 해석 및 DID 문서를 반환받는 경로를 변경하는 방식으로 재설계되었다

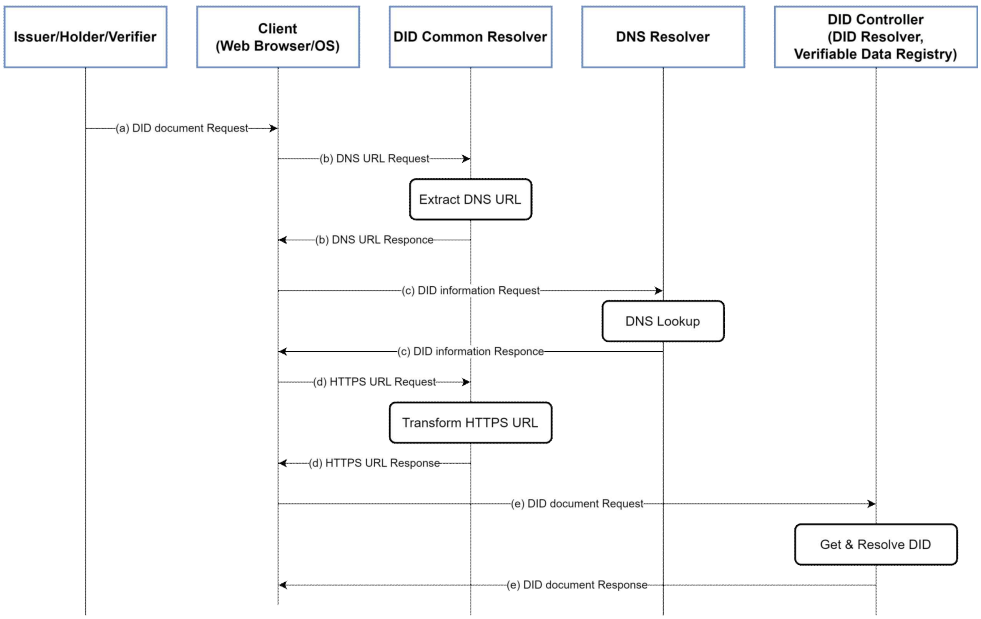
ㅇ 사용자 (발급자, 소유자, 검증자)는 각 지갑이 속하는 블록체인의 주소맞다 고유한 DID를 발급받을 수 있으며, 사용자는 DID Method에 구애받지 않고 DID 식별자를 클라이언트에 입력하여 DID 문서를 요청할 수 있다

ㅇ 클라이언트는 DID Common Resolver를 사용하여 DID에 포함된 DNS URL을 추출하고, 이를 기반으로 직접 DNS Resolver와 통신하여 정보를 요청할 수 있다

ㅇ 이후 DID Common Resolver는 HTTPS URL을 생성하고 DID Controller에게 DID 문서를 요청하는 방식으로 식별자를 해석하고 DID 문서를 요청할 수 있다



(그림6-12) DID Common Resolver 직접 해석모델 구조



(그림6-13) DID Common Resolver 직접 해석 절차

ㅇ 해당 직접해석 모델에서 Universal Resolver의 동작 프로세스는 아래와 같다

(1) 클라이언트에 입력된 값 처리

- 클라이언트는 기존 시스템과의 호환성을 유지하기 위해 사용자가 “did: [method name]:[method-specific-identifier]” 혹은 “did:direct:[DNS-URL]:[Method-Specific-Identifier]:(DID Methjod)” 형식의 식별자를 입력하도록 허용한다
- 일반적인 DID 식별자가 입력된 경우, 기존 방식(Universal Resolver, did:web 등)으로 동작하며, 후자인 “did:direct:...” 이 입력된 경우, 해당 값을 DID Common Resolver가 수신한다

(2) DNS-URL 추출(식별자 문자열 Parsing)

- 예시로 “did:direct:_did.example.com:0x123456789:ethr” 가 입력된 경우, 아래 표와 같이 Parsing을 진행한다

[표 6-1] 식별자에 대한 문자열 parsing 결과

구성 요소	Parsing 결과 값
DNS-URL	_did.example.com
Method-Specific-Identifier	0x123456789
DID Method	ethr

- DID Common Resolver는 DNS-URL이 유효한지 확인한 후, DNS-URL을 클라이언트에게 반환하여 DNS Resolver와 통신할 수 있도록 한다

(3) HTTPS URL 생성(DID 정보 및 식별자 문자열 parsing)

- 클라이언트는 TXT 타입의 Resource Record(RR)을 요청하여 didResolution Location과 availableNetwork를 반환받는다. DID Common Resolver는 이를 수신하고 식별자 정보와 함께 HTTPS URL로 변환한다
- JSON 형식의 정보를 수신하며, 수신 정보 및 parsing 예시는 표 6-2와 같다
- HTTPS URL은 식별자의 Method-Specific-Identifier, availableNetwork 정보를 결합하여 생성되며 정보는 HTTPS URL내 “id”, “&network” 에 배치되며 “https://example.com/?id=0x123456789&network=ethr” 와 같이 생성된다

[표 6-2] TXT RR parsing 결과

구성 요소	내용

수신 정보(JSON)	{
	“didResolverLocation” :
	‘https://example.com’,
	“availableNetworks” :
	[‘ethr’, ‘indy:sovrin’, ...]
	}

didResolverLocation	https://example.com
availableNetworks	ethr

- DID Common Resolver는 해당 HTTPS URL을 반환하여 클라이언트가 DID Controller와 통신하여 질의하고 DID 문서를 반환받을 수 있도록 한다

ㅇ 해당 DID 직접해석 방안의 기능 요구사항은 다음과 같다

- **(사용자)** DID 직접 해석 방식으로 DID 문서를 요청하기 위해 1절에서 명시한 형식의 DID 식별자를 클라이언트에 입력해야 한다
- **(클라이언트)** 추출된 DNS URL을 기반으로 DNS Resolver에게 DID 정보를 요청하고, 반환된 정보를 기반으로 입력된 DID와 대조하여 해석 가능 여부를 판별할 수 있는 기능이 포함되어야 한다
- **(DID Common Resolver)** DNS URL 추출 및 HTTPS URL 생성을 위해 문자열 Parsing 기능을 지원해야 한다
- **(DID Controller)** 특정 DID Method로 구성된 DID 서비스의 Controller는 W3C에서 정의한 DID 사양을 따르는 블록체인 기반 DID Method를 사용하여야 하며, DID Controller는 DNS Resolver에서 해석기의 위치, 사용할 수 있는 블록체인 네트워크 목록 등 DNS URL에 해당하는 정보를 제공할 수 있도록 TXT형식의 DNS Zonefile을 등록하여야 함과 더불어, DID 서비스에 포함된 DID Resolver를 기반으로 DID 문서를 해석할 수 있도록 HTTPS URL에 포함된 Method 이름, Method-Specific-Identifier을 문자열 Parsing 등의 과정을 통해 DID 식별자로 변환할 수 있어야 한다
- **(DNS Resolver)** 255byte 이상의 TXT 타입 Resource Record(RR)을 제공할 수 있어야 하며, 이를 위해 RFC 6891을 준수하는 Extended Mechanism for Domain Name System(EDNS)를 지원하여야 한다

ㅇ did:direct와 마찬가지로, 기존 DNS 인프라를 활용할 수 있고, DNS Zone file 업데이트 및 RR 확장이라는 간단한 방식으로 구현이 가능함과 더불어, 각 DID Method의 Resolver를 호출하는 API 역할을 수행함으로써 중앙화된 해석기에 대한 의존성과 복잡성이 낮다는 장점이 있다

- 반면에, DNS를 사용함으로 인한 보안 취약점 고려 및 DID 서비스를 제공하는 Controller의 부담이 높아진다는 단점이 존재한다

제 7 장 결과 및 제언

제 1 절 DID Method 분석 결과

ㅇ DID는 URI의 한 형태로, DID 주체를 인증 정보 등이 포함된 DID 문서와 연결하여 신뢰할 수 있는 상호작용을 가능하게 하며, 국내·외 기관 및 산업체에서 표준화, 지원 사업 및 기술 도입 시도가 증가하는 등 관심도가 높아지고 있다

ㅇ 본 연구에서는 DID Method 간 상호운용성을 보장할 수 있는 직접해석 방안을 마련하기 위해 DID Method, DID가 포함된 VC/VP 시스템, 관련 표준 및 서비스 동향에 대해 기술조사 및 분석을 수행하였다

ㅇ DID Method는 DID의 고유한 작동 방식을 정의한 규격으로, 특정 DID 체계, DID와 DID 문서에 대해 생성(Create), 해석(Read), 업데이트(Update), 비활성화(Deactivate) 되는 일련의 CRUD 절차와 더불어 구현, 보안 및 개인정보 보호 고려사항 등이 포함된다

ㅇ DID Method는 블록체인 네트워크, Web 등 다양한 방식을 신뢰 기반으로 사용하여 구현될 수 있으며, 신뢰 기반을 기준으로 표 7-1과 같이 분류할 수 있다

ㅇ VC/VP 시스템에서 DID가 사용되는 경우, 시스템을 구현하는 소프트웨어 라이브러리는 DID를 해석할 수 있는 기능이 포함되어야 한다. DID 기반 URL은 주체, 발급자, 소유자, 자격 증명 상태 목록, 암호화 키 및 VC 관련 기계 판독 가능 정보와 관련된 식별자를 표현하는 데 사용된다

○ DID 관련 표준문서는 주로 W3C에서 관리되며, DID 표준문서 외에도, 워킹그룹 및 다수의 Internet Draft 문서를 지속적으로 유지 · 관리하고 있다

[표 7-1] DID Method 분류

신뢰기반	내용
블록체인 네트워크	블록체인 네트워크를 신뢰기반으로, 이미 구성되어 있는 블록체인 네트워크에 DID 문서를 저장함으로써 내부 정보에 대한 신뢰성과 보안성을 높이는 방식이다. 개인기기의 지갑, RPC URL, 신뢰할 수 있는 노드 등을 통해 직접 접속하여 DID 문서를 확보할 수 있다 상대적으로 단순한 해석이 가능하고 특정 인프라에 종속되지 않아 유연성과 확장성이 뛰어나다
Web	Web 환경과 DNS 인프라를 통해 분산된 신원 관리 시스템을 구축하며, 도메인 네임을 기반으로 DID를 구성하여 기존 Web 및 네트워크 인프라를 활용할 수 있다 주요 기반이 중앙화된 인프라에서 관리된다는 한계가 존재하며, 핵심 요소인 DNS 자체를 신뢰할 수 있는 방안에 대한 제시가 필요하다
기타	Web 서버, 호스트 등 외부 저장소를 사용하거나 식별자를 DID 문서로 해석하는 DID Method 외부 저장소를 사용하는 경우, 신뢰 기반은 중앙 저장소를 관리하는 서버 자체 혹은 별도로 명시할 수 있다 식별자를 DID 문서로 해석하는 경우, 대부분 일시적인 용도로 사용하는데 초점이 맞춰져 있으며, 공개 키 자체를 확장하여 DID 문서의 구조에 맞게 작성하게 된다

○ IETF에서는 Web 기반 DID와 간접적으로 관련성이 있는 DNS, RR 유형 등을 RFC 표준문서로써 관리하고 있으며, DID와 직접적으로 관련이 있는 내용들은 Internet Draft로만 관리하고 있다

제 2 절 DID 직접해석 방안 제언

○ DID Method는 다양한 방식으로 구현될 수 있어 필요 인프라가 Method 별로 상이하며, 상호운용성에 한계가 존재하여 확장성 측면에서 한계를 야기하게 된다. 따라서 DID에 대한 관심도가 높아지고 있는 현 상황에서 DID 기술 확산을 위해 다양한 DID Method를 아우를 수 있는 DID 직접해석 방안에 대한 연구가 필요하다

○ DID 직접해석은 특정 DID를 Web 브라우저 등에 입력했을 때, 각 DID Method 별로 요구되는 해석기 없이 DID 문서를 반환할 수 있는 해석 방법으로, 기존 방식의 확장성 한계를 극복하는 것을 목표로 한다

○ DID 서비스 구조의 여러 계층에서 DID 직접해석을 고려할 수 있으며, 본 연구에서는 4가지 직접해석 방안을 도출하였다

○ **(WAS 활용방안)** Universal Resolver를 WAS로 활용하여 DID 직접해석을 수행하는 방안

- 사용자는 클라이언트를 통해 Universal Resolver에 DID를 입력한다
- Universal Resolver는 입력받은 DID를 분석하여 해당 DID Method에 적합한 Driver를 선택하고, 이를 기반으로 해석을 수행하여 DID 문서를 반환한다
- **(장점)** 기존에 사용하던 식별자 구조를 변경하지 않고, 단일 인터페이스를 통해 DID를 입력받고 DID 문서를 반환할 수 있다

- **(단점)** DID Method 해석을 위한 Driver는 Universal Resolver에 종속되어 개발 및 유지관리의 복잡성이 상승하고, 중앙 해석기에 대한 의존성이 높다

○ **(블록체인 네트워크 활용방안)** 블록체인 네트워크를 기반으로 DID Resolver 네트워크를 구축하여 분산 디렉토리 서비스 역할을 수행하는 방안

- 사용자는 DID를 특정 DID Resolver에 질의한다. DID Resolver는 입력받은 DID를 분석하여 해석 가능 여부를 확인한다
- 해석이 가능한 경우, 해당 DID Resolver를 활용하여 DID 문서를 반환하며, 해석이 불가능한 경우, 분산 디렉토리 서비스 역할을 수행하는 블록체인 네트워크에 추가 질의를 수행하여 적합한 DID Resolver 주소를 조회한다
- 조회된 적합한 DID Resolver 주소로 HTTPS 요청을 리다이렉트하여 DID를 전달 및 해석 요청을 수행하고, DID 문서를 반환받는다
- **(장점)** 탈중앙화된 방법으로 DID 직접해석 지원 및 데이터의 무결성·신뢰성을 보장할 수 있으며, 새로운 DID Method 추가 및 DID Resolver 변경이 간편하여 확장성이 용이하다
- **(단점)** 초기 네트워크 구축 및 Smart Contract 비용 부담이 존재하고, DNS 대비 네트워크 성능이 낮다. Smart Contract 오류, HTTPS 리다이렉션, 서비스 제공자 간 신뢰 문제에 취약하다

○ **(DNS 활용방안)** DNS로부터 반환받는 RR에 여러 신뢰기반 정보가 포함된 추가적인 데이터를 포함시켜 did:dns를 확장하는 방안으로, did:direct와 같이 신규 DID Method로 제공될 수 있다

- 식별자 구조는 기존 did:dns와 동일하며, 사용자는 클라이언트에 DID를 입력한다. 식별자에 포함된 도메인 네임을 변경하여 DNS lookup 서비스를 수행한다
- DID Controller는 DNS Zonefile을 등록해야 하며, DNS lookup 서비스

를 통해 DID 문서 위치, Public Key, 신뢰 기반 정보가 포함된 RR을 반환받으며, 이를 기반으로 DID 문서를 검증하고 반환받을 수 있다

- **(장점)** 기존 DNS 인프라를 활용할 수 있으며, 간단한 방식으로 구현이 가능하다
- **(단점)** 서버 운영자 등, DNS 관리 주체에 종속될 수 있으며, DNS 보안 취약점 방지 대책을 고려해야 한다

○ **(DID Common Resolver 활용방안)** 클라이언트에 문자열 Parsing을 수행하는 새로운 Resolver인 DID Common Resolver를 배치하는 방식으로, DID Resolver를 호출하는 API 역할을 수행하도록 하는 방안이다. DNS 활용방안과 마찬가지로 did:direct와 같이 신규 DID Method로 제공될 수 있다

- 식별자에 did:direct와 더불어 DNS URL과 DID Method 해석을 위한 식별자가 포함되어야 하며, 사용자는 클라이언트에 DID를 입력한다
- DID Common Resolver는 DNS URL을 추출하여 클라이언트가 DNS Resolver와 통신할 수 있도록 하며, DNS Resolver로부터 DID Resolver 위치와 해석에 사용될 수 있는 Method 정보가 포함된 RR을 반환받는다
- DID Common Resolver는 반환받은 RR과 입력받은 DID를 활용하여 HTTPS URL을 생성하고, 클라이언트는 이를 기반으로 DID Resolver와 통신하여 DID 문서를 반환받는다
- **(장점)** 기존 DNS 인프라를 활용할 수 있으며, 간단한 방식으로 구현이 가능하다. API 역할을 수행함으로 인해 의존성과 복잡성이 낮다
- **(단점)** DNS 보안 취약점 방지 대책을 고려해야 하며, DID 서비스를 제공하는 DID Controller에 대한 요구사항이 많아 부담이 높다

○ 본 연구를 통해 4가지 DID 직접해석 방안을 도출하였으며, 직접해석 방안에 활용되는 기술 분석을 통해 각 방안별 장단점이 있음을 확인하였다

- 따라서 향후 다각도적인 관점으로 추가적인 연구를 수행할 필요가 있다. 해당 연구를 통해 국제 표준화 기구의 요구사항을 만족하는 방법론을 도출하여 DID 표준화에 기여하거나, DID 적용 산업을 확대하고 관련 기술 개발을 촉진시킬 수 있을 것으로 기대된다

참고문헌

[1] 김현일, 박경철, and 서창호. “과기부 ‘Web 3.0 기반 분산 ID 표준화 동향”, TTA 저널 208 (2023): 99-105.

[2] 블록미디어, “과기부 ‘DID 통합 해석기’ 만든다…DID 상호호환 논의 본격화”, 2020, https://www.blockmedia.co.kr/archives/159622 Accessed Aug., 02. 2024.

[3] 딜사이트, “과기부 ‘DID 통합 해석기 연구 개발한다’”, 2020, https://dealsite.co.kr/articles/66045. Accessed Aug, 02, 2024

[4] Manu Sporny, Amy Guy, Markus Sabadello and Drummond Reed. “Decentralized Identifiers (DIDs) v1.0”. W3C Recommendation. 19 July 2022. URL: https://www.w3.org/TR/did-core/ Accessed Nov, 03, 2024.

[5] Berners-Lee T, Fielding RT and Masinter LM. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, 2005. DOI:10.17487/RFC3986. URL: https://www.rfc-editor.org/info/rfc3986. Accessed Aug, 02, 2024.

[6] Joe Andrieu, Phil Archer, Kim Duffy, Ryan Grant and Adrian Gropper. “Use Cases and Requirements for Decentralized Identifiers”. W3C Working Group Note. 17 March 2021. URL: https://www.w3.org/TR/did-use-cases/. Accessed Aug, 02, 2024

[7] Orie Steele, Manu Sporny and Michael Prorock. “DID Specification Registries.” W3C Working Group Note. 28 June 2022. URL: https://www.w3.org/TR/did-spec-registries/. Accessed Aug, 02, 2024

[8] Michael B. Jones. JSON Web Key (JWK). RFC 7517, 2015. DOI:10.17487/RFC7517. URL https://www.rfc-editor.org/info/rfc7517 Accessed Nov, 29, 2024.

[9] Manu Sporny, et al. “DID Methods”, W3C Group Note. 19 Novembe

r 2024. URL: https://www.w3.org/TR/did-extensions-methods/. Accessed Nov, 27, 2024

[10] Hyperledger. “Hyperledger Indy Node documentation”, (2024), URL https://hyperledger-indy.readthedocs.io/projects/node/en/latest/. Accessed Nov, 29, 2024.

[11] Phil Feairheller, et al. “ToIP did:webs Method Specification v0.9.15”. Implementors Draft. URL: https://trustoverip.github.io/tswg-did-method-webs-specification/. Accessed Nov, 03, 2024.

[12] Domain names - implementation and specification. RFC 1035, 1987. DOI:10.17487/RFC1035. URL https://www.rfc-editor.org/info/rfc1035. Accessed Nov, 03, 2024.

[13] Braden RT. Requirements for Internet Hosts - Application and Support. RFC 1123, 1989. DOI:10.17487/RFC1123. URL https://www.rfc-editor.org/info/rfc1123. Accessed Nov, 03, 2024.

[14] Elz R and Bush R. Clarifications to the DNS Specification. RFC 2181, 1997. DOI:10.17487/RFC2181. URL https://www.rfc-editor.org/info/rfc2181. Accessed Nov, 03, 2024.

[15] Markus Sabadello. “did:dns Method Specification”, Unofficial Draft, 04 January 2022. URL: https://danubetech.github.io/did-method-dns/. Accessed Nov, 03, 2024.

[16] Manu Sporny, et al. “The did:key Method v0.7” Unofficial Draft, 02 September 2022. URL : https://w3c-ccg.github.io/did-method-key/. Accessed Nov, 03, 2024.

[17] Manu Sporny, et al. “Verifiable Credentials Data Model v2.0”, W3C Candidate Recommendation Draft, 19 October 2024. URL: https://www.w3.org/TR/vc-data-model-2.0/. Accessed Nov, 03, 2024.

[18] Orie Steele, et al. “DID Implementation Guide v1.0”, W3C Working Group Note, 12 October 2021. URL: https://www.w3.org/TR/did-imp-guide/. Accessed Nov, 03, 2024.

[19] Michael Prorock, et al. “did:web Method Specification”, Unofficial Draft, 31 July 2024. URL: https://w3c-ccg.github.io/did-method-web/. Accessed Nov, 03, 2024.

[20] P. Faltstrom and O.Kolkman. The Uniform Resource Identifier (URI) DNS Resource Record. RFC 7553, 2015. DOI:10.17487/RFC7553. URL https://datatracker.ietf.org/doc/html/rfc7553. Accessed Nov, 03, 2024.

[21] R. Rosenbaum. Using the Domain Name System To Store Arbitrary String Attributes. RFC 1464, 1993. DOI:10.17487/RFC1464. URL https://datatracker.ietf.org/doc/html/rfc1464. Accessed Nov, 03, 2024.

[22] DIF Identifiers&Discovery Working Group. “universal-resolver”, 2024, https://github.com/decentralized-identity/universal-resolver. Accessed on Nov 03, 2024.

부록 1. 표준문서 초안

ㅇ 본 표준은 정식 표준문서가 아닌, 향후 표준화 추진시 가이드라인 역할을할 수 있도록 구성된 표준문서 초안이며, 국문으로 작성되었다

ㅇ 본 표준의 구성은 IETF의 RFC 양식을 기반으로 작성되었으며, 6장 4절 DID Common Resolver 활용 직접해석 방안을 기반으로 작성되었다

Network working Group
Internet-Draft
Intended status: Standard Track
Expires: December 31, 2025

J. Hyeonseok
CNU

November 16, 2024

DID Common Resolver의 분산 ID(DID) 직접 해석 및 클라이언트와의 협업을 위한
메시지 전달절차 및 포맷

Direct Resolution of Decentralized Identifier(DID) by DID Common Resolver and Message Passing Procedure and Format for Collaboration with Clients

Status of This Memo

이 문서는 표준 등록 절차를 준비하기 위해 작성된 초안이다. 분산 ID(DID)를 클라이언트에 입력했을 때, 다른 DID 서비스의 식별자를 인식하고 DID 문서를 읽어올 수 있는 DID Common Resolver의 직접 해석 방법, 메시지 전달 절차와 포맷을 제안한다.

Abstract

이 표준의 목적은 특정 해석기에 대한 의존성을 줄이고, 다양한 DID Method 간 상호운용성을 보장하고 DID 확산에 기여하는데 있다. 이 표준은 DID 직접 해석을 위한 DID Common Resolver를 활용한 식별

자 인식 및 해석 방법, 기능 요구사항, DID 문서를 반환받기 위한 메시지 전달 절차와 포맷을 정의한다. 특정 신뢰 기반이나 암호화 요구사항, 무결한 DID 문서를 검증하는 방법 등은 이 표준에서 다루지 않는다.

Table of Contents

1. Introduction

1.1 분산 ID(DID) 직접 해석

non-normative

DID는 Uniform Resource Identifier(URI) 구조를 갖는 새로운 유형의 식별자이다. 최근 DID는 탈중앙 구조 등 특유의 분산성과 Distributed File Systems(DFS), Data Distributed Service(DDS), Peer-To-Peer(P2P) 등과의 상호운용성으로 대표되는 신뢰성 있는 검증 여부로 인해 많은 관심을 받고 있으며, 이러한 특성을 기반으로 기존 고유 식별자가 사용되던 전화번호, 이메일, 여권, 면허, 제품 식별자 등 광범위한 분야를 대체할 수 있다.

높아지는 관심과 함께 지속적으로 새로운 DID Method들이 발표되고 있으나, 분산원장 혹은 기타 분산 인프라 외에도 Domain Name System(DNS) 및 기타 Web 서버 인프라 등 다양한 방식으로 구현이 가능하기 때문에 동작 방식, 신뢰 기반 및 필요한 인프라가 모두 상이하고, 이는 서로 다른 DID Method간 상호운용성을 어렵게 한다. 일부 DID Method는 기존 인프라를 활용하여 단순한 문자열 Parsing만으로 동작하는 반면, 블록체인을 사용하는 DID Method와 같은 경우는 네트워크 별로 Remote Procedure Call Uniform Resource Locator(RPC URL)의 사용 여부, 블록체인에 접근할 수 있는 Application의 필요성 등 많은 부분이 달라 해석에 어려움이 존재한다.

이 표준은 이러한 한계를 극복하기 위해 DID를 Web 브라우저에 입력했을 때, DID Common Resolver를 활용하여 다른 DID 서비스의 식별자를 인식하고 DID 문서를 읽어올 수 있는 직접 해석방법과 메시지 전달 절차, 포맷을 제안한다.

제안된 DID Common Resolver 기반 DID 직접 해석은 사용자가 클라이언트에 DID를 입력하면, 클라이언트는 DID Common Resolver를 사용하여 DID에 포함된 DNS URL를 추출하고, DNS Resolver와 통신하여 DID 정보를 반환받을 수 있고, 해석 가능 여부를 확인할 수 있다. DID Common Resolver는 이를 기반으로 HTTPS URL을 구성하고, 이

를 통해 특정 DID Method를 기반으로 구현된 DID 서비스의 DID Controller에게 DID 문서 해석을 요청하고 해당 DID 서비스의 DID Resolver를 통해 해석된 DID 문서를 반환받음으로써, 다양한 DID Method와의 상호운용성을 보장하는 형태로 DID 문서를 반환받을 수 있다.

1.2 문서 구성

non-normative

이 표준은 DID 직접 해석을 위한 DID Common Resolver의 해석 모델 및 방법, 기능 요구사항 및 클라이언트와의 협업을 위한 기본적인 메시지 전달 절차 및 포맷을 정의한다. 특정 신뢰 기반이나 암호화 요구사항, 무결한 DID 문서를 검증하는 방법 등은 이 표준에서 다루지 않는다.

메시지 전달 절차 및 포맷은 DID 문서를 해석 및 반환받는 경로인 사용자-클라이언트-DID Controller간의 통신 과정에 대해서만 다루며, 사용자(발급자, 소유자, 검증자) 간의 통신과 같이 인터페이스에 해당하는 부분에 대해서는 다루지 않는다.

다양한 DID Method 및 DNS 통신과 관련된 표준을 참고하여 통합하지만, 다음과 같은 DID 및 DNS 표준을 변경하거나 새로운 기능을 추가하지 않는다:

o Decentralized Identifiers (DIDs) v1.0 from W3C [1],

o DOMAIN NAMES from RFC 1035 [2]

o Requirements of Internet Hosts from RFC 1123 [3]

o Clarifications to the DNS Specification from RFC 2181 [4]

o Extension Mechanisms for DNS(EDNS) from RFC 6891 [5]

이 표준의 나머지 구성은 다음과 같다:

- **2. DID 직접 해석 모델** 섹션에서는 DID 직접 해석 모델을 도식화하고 표준에서 변경 및 제안하고자 하는 범위와 직접 해석 방법에 대해서 정의한다.

- **3.1 메시지 전달 절차** 섹션에서는 직접 해석 모델에서 DID 문서를 반환받기 위한 메시지 전달 절차를 정의하며, **3.1.1 기능 요구사항** 및 **3.1.2 구현 및 보안 고려사항**에서 직접 해석을 위한 기능 요구사항 및 고려사항을 다룬다.

- **3.2 메시지 포맷**에서는 DID 식별자 구조 정의가 포함된 직접 해석을 위한 메시지 포맷에 대해서 정의한다.

1.3 문서 규칙

이 표준문서 내 non-normative로 표기된 섹션은 표준으로 정의하고자 하는 부분이 아니며, 그 외 다른 부분은 모두 표준으로 정의하고자 하는 부분을 나타낸다.

이 표준문서에 포함된 예시 등에는 DNS URL과 더불어, 블록체인 기반 DID Method, JSON-LD 형식의 DID 문서 등이 포함되어 있다. DNS URL 형식은 RFC 1035 [2], RFC 1123 [3] RFC 2181 [4]에 명시된 요구사항을 준수하며, 해당 표준에 명시된 바와 같이 해석되어야 한다. DID 식별자, DID Controller, DID 문서 등은 DIDs v1.0 [1]에 명시된 요구사항을 준수하고 해당 표준에 명시된 바와 같이 해석되어야 한다.

2. DID 직접 해석 모델

이 섹션에서는 특정 해석기에 대한 의존성을 줄이고, DID Method 간 상호운용성 보장 및 확산을 위한 직접 해석 모델을 정의한다.

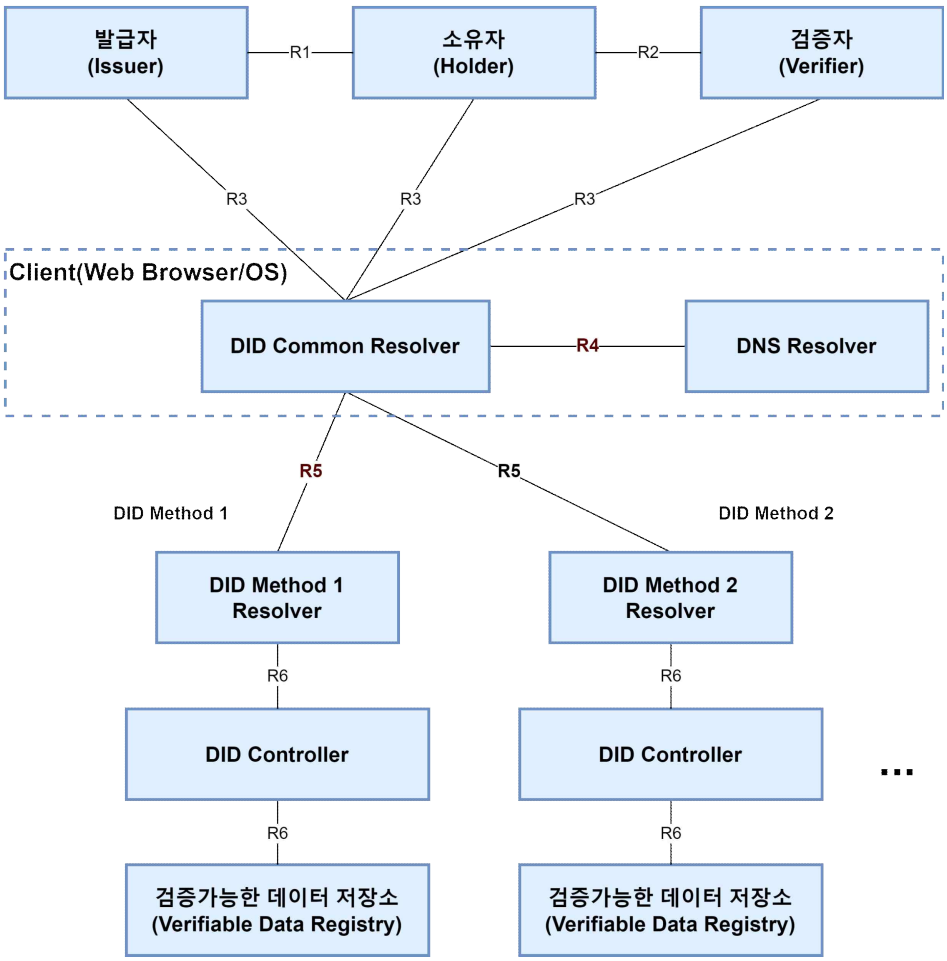


그림 1. DID Common Resolver 해석 모델 아키텍쳐

이 표준의 범위는 식별자를 Web 브라우저, OS 등 클라이언트에 입력하고 식별자를 해석 및 DID 문서를 반환받는 경로에 해당하며, 그림 1과 같이 클라이언트 내에 DNS Resolver와 통신할 수 있는 R4를 추가하여 R5를 통해 DID Resolver와 통신하는 것이다. 특정 신뢰 기반이나 암호화 요구사항, 무결한 DID 문서를 검증하는 방법과 더불어, 인터페이스에 해당하는 경로인 R1, R2는 다루지 않는다.

DID 직접 해석 모델에서 사용자(발급자, 소유자, 검증자)는 각 지갑이 속하는 블록체인의 주소마다 고유한 DID를 발급받을 수 있다. 사용자는 DID Method에 구애받지 않고, DID 식별자를 클라이언트에 입력하여 DID 문서를 요청할 수 있다. 클라이언트는 DID Common Resolver

를 사용하여 DID에 포함된 DNS URL을 추출하고, 이를 기반으로 직접 DNS Resolver와 통신하여 정보를 요청할 수 있다. 이후 클라이언트 내 DID Common Resolver는 HTTPS URL로 변환하고 DID Controller에게 DID 문서를 요청하는 방식으로 식별자를 해석할 수 있다. DNS Resolver로부터 해석기 위치, DID Method 및 식별자 정보가 포함된 정보를 반환받으며, DID Common Resolver는 이를 HTTPS URL로 변환하여 특정 DID Method로 구성된 DID 서비스에게 식별자 해석 요청을 수행하고 해당 DID Method에 적합한 DID Resolver가 포함된 DID 서비스로부터 DID 문서를 반환받아 표현할 수 있다.

입력한 식별자를 해석하기 위해 클라이언트는 DNS Resolver와 직접 통신하여 DID Controller에게 해석 요청을 보냄으로써, Universal Resolver를 사용하지 않고도 다양한 DID Method와의 상호운용성을 보장하는 형태로 DID 문서를 반환받을 수 있으며, 특정 DID Method들은 Universal Resolver에 드라이버를 업데이트 할 필요 가 없으므로 Universal Resolver에 종속되지 않고 유연한 확장이 가능하다. DID 직접 해석 모델은 아래 그림과 같이 표현될 수 있다:

2.1 DID Common Resolver의 역할 및 동작 절차

DID Common Resolver의 역할 및 동작 절차는 아래와 같이 정의된다:

(1) 클라이언트에 입력된 값 처리

- 클라이언트는 기존 시스템과의 호환성을 유지하기 위해 사용자가 did:[method]:[method-specific-identifier] 혹은 **did:direct:[DNS- URL]:[method-specific-identifier]:(DID Method)** 형식의 식별자를 입력하도록 허용한다.
- 일반적인 DID 식별자가 입력된 경우, 기존 방식(Universal Resolver, did:web 등)으로 동작하며, 후자인 did:direct:... 이 입력된 경우, 해당 값을 DID Common Resolver가 수신한다.

(2) DNS-URL 추출(식별자 문자열 parsing)

- 예시로 did:direct:_did.example.com:0x1234567:ethr가 입력된 경우, **DNS-URL**은 “_did.example.com”, **Method-Specific-Identifier**는 “0x 1234567”, **DID Method**

는 “ethr”로 parsing 된다.

- DNS-URL에는 “_did”가 포함되어야 하며, RFC 1035 및 관련 DNS 표준을 기반으로 DNS-URL이 유효한지 확인한다. 이후 DNS-URL을 반환하여 클라이언트가 DNS Resolver와 통신할 수 있도록 한다.

(3) HTTPS URL 생성(DID 정보 및 식별자 문자열 parsing)

- 클라이언트는 TXT 타입의 Resource Record(RR)을 요청하여 didResolverLocation과 availableNetwork를 반환받는다. DID Com- mon Resolver는 이를 수신하고, 식별자 정보와 함께 HTTPS URL로 변환한다.
- JSON 형식의 정보를 수신하며, 예시로 didResolverLocation으로 “http s://example.com/”, availableNetworks로 배열의 첫 번째 값인“ethr”가 parsing된다.
- HTTPS URL은 didResolverLocation과 식별자의 Method-Specific -Identifier, availableNetwork 정보를 결합하여 생성되며 위 두가지 예시와 같이 정보를 parsing한 경우, “id”, “&” 에 parsing 정보를 배치하여 https://example.com/?id=0x1234567&network=ethr로 변환된다. 해당 URL을 반환하여 클라이언트가 DID Controller와 통신하여 질의하고 DID 문서를 반환받을 수 있도록 한다.

3. 메시지 전달 절차 및 포맷

이 섹션에서는 DID 직접 해석을 위한 메시지 전달 절차 및 포맷을 정의한다.

3.1 메시지 전달 절차

이 표준에서 정의하는 직접 해석 모델에서 DID 문서를 반환받기 위한 메시지 전달 절차는 아래와 같은 그림으로 표현될 수 있다:

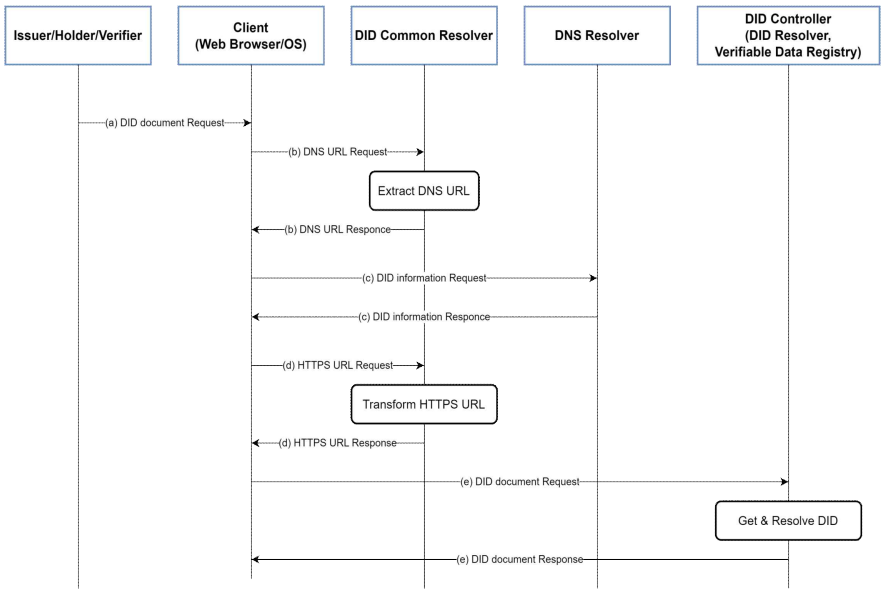


그림 2. DID Common Resolver 메시지 전달 절차

초기에 클라이언트는 DID Common Resolver를 초기화하고 요청된 식별자를 처리하기 위해 아래 절차대로 메시지를 전달한다:

(a) 사용자는 클라이언트를 통해 DID 문서를 요청한다. 요청 메시지 내에는 이 표준에서 정의하는 형식의 DID 식별자가 입력되어야 하며, 사용자는 발급자, 소유자, 혹은 검증자일 수 있다.

(b) 클라이언트는 DID Common Resolver에 DNS URL 정보를 요청한다. DID Common Resolver는 DNS Resolver와 통신할 수 있도록 문자열 Parsing을 통해 식별자에서 DNS URL을 추출하고 클라이언트에 반환한다.

(c) 클라이언트는 DNS URL을 기반으로 DNS Resolver와 통신을 통해 DID 정보를 요청 및 반환받고, 이를 사용자의 식별자와 대조하여 해석 가능 여부를 확인한다.

(d) 해석 가능 여부 확인 이후, 클라이언트는 DID Common Resolver에 HTTPS URL 정보를 요청한다. DID Common Resolver는 반환받은 DID 정보와 식별자, 문자열 Parsing을 활용하여 HTTPS URL을 생성하고 클라이언트에 반환한다.

(e) 클라이언트는 HTTPS URL을 통해 특정 DID Method로 구성된 DID 서비스의 DID Controller에게 DID 문서를 요청한다. DID Controller는 HTTPS URL을 기반으로 문자열 Parsing을 통해 DID를 얻고,

DID 서비스에 포함된 DID Resolver를 이용하여 해석을 수행하여 DID 문서를 반환한다.

3.1.1 기능 요구사항

이 표준에서 제안하는 DID 직접 해석 방법의 기능 요구사항은 아래와 같다:

- o 사용자는 DID 직접 해석 방식으로 DID 문서를 요청하기 위해 이 표준에서 정의한 형식의 DID 식별자를 클라이언트에 입력해야 한다.

- o 클라이언트는 추출된 DNS URL을 기반으로 DNS Resolver에게 DID 정보를 요청하고, 반환된 정보를 기반으로 입력된 DID와 대조하여 해석 가능 여부를 판별할 수 있는 기능이 포함되어야 한다.

- o DID Common Resolver는 DNS URL 추출 및 HTTPS URL 생성을 위해 문자열 Parsing 기능을 지원해야 한다.

- o 기존 DID Method와의 호환성을 고려하여 Method 명이 포함된 DID 식별자가 입력된 경우, Universal Resolver 등 기존 방식을 사용하여 해석을 수행하고, 본 문서에서 정의한 형식의 식별자가 입력된 경우 DID Common Resolver 및 DNS Resolver로 자동으로 인식하여 직접 해석 방식으로 DID 문서를 요청할 수 있어야 한다.

- o 특정 DID Method로 구성된 DID 서비스는 W3C에서 정의한 DID 사양을 따르는 블록체인 기반 DID Method를 사용하여야 하며, DID Controller는 DNS Resolver에서 해석기의 위치, 사용할 수 있는 블록체인 네트워크 목록 등 DNS URL에 해당하는 정보를 제공할 수 있도록 TXT형식의 DNS Zonefile을 등록하여야 함과 더불어, DID 서비스에 포함된 DID Resolver를 기반으로 DID 문서를 해석할 수 있도록 HTTPS URL에 포함된 Method 이름, Method-Specific-Identifier을 문자열 Parsing 등의 과정을 통해 DID 식별자로 변환할 수 있어야 한다.

- o DNS Resolver는 255byte 이상의 TXT 타입 Resource Record(RR)을 제공할 수 있어야 하며, 이를 위해 RFC 6891을 준수하는

Extended Mechanism for Domain Name System(EDNS)를 지원하여야 한다.

3.1.2 구현 및 보안 고려사항

이 표준에서 제안하는 DID 직접 해석 방법의 구현 및 보안 고려사항은 아래와 같다:

- 구현자는 클라이언트의 해석 가능 여부 판별, DID Common Resolver의 DNS URL 추출 및 HTTPS URL 형식 변환을 위해 Web Extension 혹은 별도의 Application을 개발하여 사용할 수 있으며, Web Extension을 사용하는 경우, Web 브라우저간 상호운용성을 고려하여야 한다.

- 구현자는 DNS Resolver를 DNS 시스템에 통합하기 위해 구현자의 DNS 서버에 추가적인 기능을 설치하거나, OS에 기본적으로 설치된 DNS Resolver에 배포 혹은 특정 기관에서 운영하는 DNS Server에 매핑하는 방식으로 구현할 수 있다.

- 구현자는 HTTPS URL 형식으로 DID Controller에게 DID 문서를 요청할 때, GET, POST 등의 방식을 사용할 수 있으며, 보안을 위해 각 엔티티간 인증 절차를 추가할 수 있다. 본 표준의 예시와 같이 API를 사용하는 경우 인증 토큰 또는 OAuth2.0 등을 활용하여 API 액세스를 제어하는 등의 절차를 추가할 수 있다.

- 구현자는 DNS 보안 취약점을 인지하고 이를 고려하여야 하며, 하이재킹, 스푸핑 등 방지를 위해 RFC 4033 [6], RFC 4034 [7], RFC 4035 [8] 등을 준수하여 DNS Security Extension(SEC) 등을 지원하는 등 신뢰도를 보장하기 위한 방안을 추가적으로 마련해야 한다.

- 본 표준의 메시지 포맷에 명시된 요청 및 응답 메시지 예시는 REST API를 기준으로 명시되었으며, 구현자는 dig와 같이 DNS 시스템에서 사용되는 CLI 명령어등을 사용할 수 있다.

- 구현자는 본 표준에서 정의한 Common Resolver를 구현할 때, 에러 처리, 메시지 포맷 및 API 설계 확장을 통해 확장성을 제공할 수 있다.

3.2 메시지 포맷

사용자는 이 표준에서 정의한 DID 직접 해석방식으로 DID 문서를 요청하기 위해 아래와 같이 정의된 DNS URL과 특정 DID Method의 식별자, 그리고 DID Controller가 하나 이상의 신뢰 기반을 제공하는 경우를 고려하여 선택사항으로 특정 DID Method 이름 혹은 블록체인 네트워크 이름을 포함하는 식별자를 클라이언트에 입력해야 한다.

이 표준에서 정의하는 직접 해석을 위한 DID 식별자 구조 및 메시지 포맷이 포함된 DID 문서 요청 및 반환은 아래 그림과 같이 표현될 수 있다:

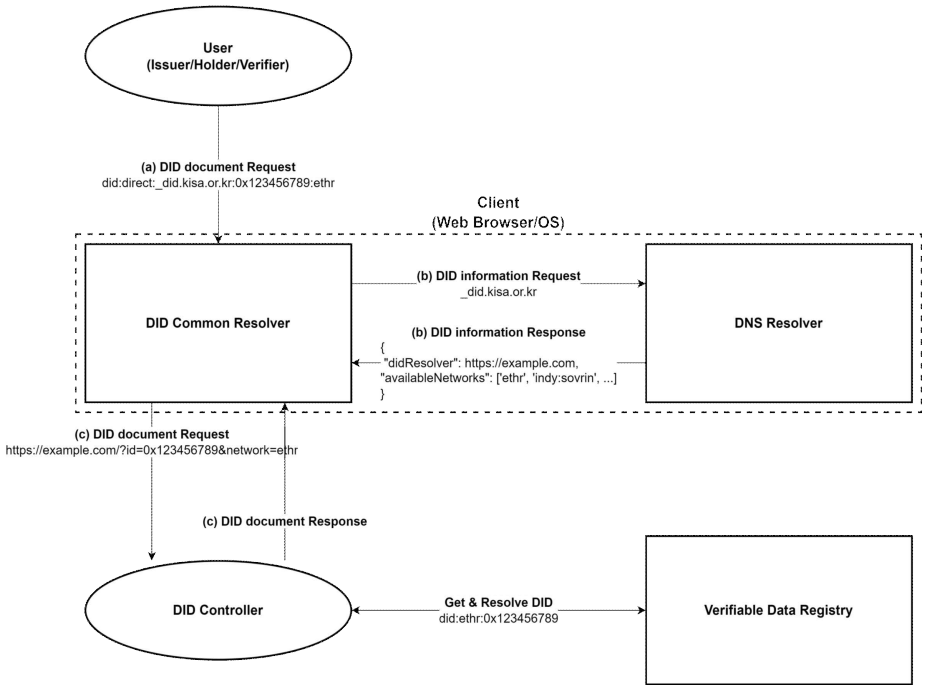


그림 3. 메시지 포맷이 포함된 DID 문서 요청 및 반환

(a) Syntax:

did:direct: [DNS-URL] : [Method-Specific-Identifier] : (DID Method)

ex) did:_did.kisa.or.kr:0x1234567:ethr

위의 직접 해석을 위한 **(a) DID Document Request 메시지인 식별자 구조**에 대한 정의는 아래와 같다:

DNS-URL	= DNS관련 표준에서 정의된 형식의 DNS URL 구조로, 일반적이지 않는 도메인 네임임을 명시하기 위해 "_did"가 포함되어야 함
Method-Specific-Identifier	= W3C에서 정의한 DID 사양을 준수하는 특정 DID Method에 대한 식별자
DID Method	= 선택사항, 블록체인을 사용하는 경우 블록체인 네트워크 이름을 표기해야 하며, 그렇지 않은 경우 DID Method 이름을 표기해야 함

해당 정보는 TXT 형식의 RR이어야 하며, DID Controller는 DNS Zone file을 작성하여 등록해야 한다. DNS Zonefile에는 특정 DID Method의 해석기 위치와 더불어 사용할 수 있는 DID Method 혹은 블록체인 네트워크 목록 등을 표현할 수 있어야 하고, 클라이언트는 이를 기반으로 해석 가능 여부를 판별할 수 있어야 한다. **(b) DNS Zonefile** 예시는 아래와 같다:

```
$ORIGIN         example.com
$TTL    86400
@ IN SOA        dns1.example.com. hostmaster.example.com. (
                200162501       ; serial
                21600           ; refresh after 6 hours
                3600            ; retry after 1 hour
                604800          ; expire after 1 week
                86400)          ; minimum TTL of 1 day
ethr._did IN TXT "{
        'didResolverLocation': 'https://example.com/',
        'availableNetworks': [
                'ethr', 'ethr:testnet', 'indy:sovrin', ...
        ]}"
```

사용자가 식별자에 선택사항을 포함하지 않은 경우, DNS Resolver로부터 반환받은 'availableNetworks' 배열의 첫 번째 인자를 HTTPS URL 변환에 사용하여야 한다. DID Controller가 하나 이상의 신뢰 기반을 제공하는 경우, 선택사항을 포함한 식별자를 입력하는 것이 권장된다.

(b) DID Information Request 메시지 예시는 아래와 같이 표현될 수 있다:

GET /resolve?dns=_did.example.com HTTPS/1.1
HOST: dnsresolver.example.com
(Endpoint): /resolve
(Method): GET
(Params): dns(string): DNS-URL (_did.example.com)

해당 TXT RR에 담긴 정보는 JSON 형태로 제공되어야 한다. 'didResolver'의 값은 DID Controller가 작성한 HTTPS URL이고 String 형식이어야 한다. 'availableNetworks'의 값은 해당 URL의 DID Resolver가 해석 가능한 블록체인 네트워크 혹은 DID Method 이름이 포함된 배열이어야 하며, String 형식의 JSON 배열이어야 한다. **(b) DID Information Response의 결과인 RR** 정보는 아래와 같이 표현될 수 있다:

```
{
        'didResolver': 'https://example.com',
        'availableNetworks': [
                'ethr',
                'ehtr:testnet',
                'indy:sovrin', ...
        ]
}
```

클라이언트는 식별자의 DID Method 혹은 블록체인 네트워크와 'availableNetwork' 키 값의 정보를 대조하여 해석 가능 여부를 판별해야 한다. 식별자에서 선택사항인 DID Method 혹은 블록체인 네트워크

정보가 생략되었을 경우, ‘availableNetwork’ 배열의 첫 번째 인자를 사용해야 한다.

해석 가능 여부 판별 이후, Web 브라우저는 아래와 같은 형식으로 표현된 HTTPS URL 형식으로 변환 및 이를 기반으로 DID Controller에게 DID 문서를 요청하여야 한다. **(c) DID Document Request 메시지인 HTTPS URL 구조**에 대한 정의는 아래와 같다:

(c) Syntax:
https:// [didResolver key value] /?id= [Method-Specific-Identifier] &network= [DID Method]
ex) https://example.com/?id=0x123456789&network=ethr

위의 **(c) HTTPS URL**의 각 항목에 대한 정의는 아래와 같다:

didResolver key value	= DNS Resolver를 통해 획득한 JSON 형식의 RR 값 중 ‘didResolver’ 키의 값. DID Controller의 DID Resolver 위치를 의미.
Method-Specific-Identifier	= 입력받은 식별자 중 ‘Method-Specific-Identifier’에 해당하는 값
DID Method	= 입력받은 식별자 중 ‘DID Method’에 해당하는 값

(c) DID Document Request 메시지 예시는 아래와 같이 표현될 수 있다:

```
GET /resolve?id=0x123456789&network=ethr HTTPS/1.1
HOST: example.com

POST /resolve HTTP/1.1
HOST: example.com
Content-Type: application/json
# POST Body
```

{
  “id”: “0x123456789”,
  “network”: “ethr”
}
(Endpoint): /resolve
(Method): GET / POST
(Params): id(string): Method-Specific-Identifier (0x1234...)
network(string): DID-Method (ethr, ...)

DID Controller는 문자열 Parsing을 통해 HTTPS URL 값 중 ‘id’와 ‘network’에 대한 처리를 수행할 수 있어야 한다. 이를 통해 아래와 같이 W3C에서 정의된 규격을 준수하는 **(c) DID Document Response 메시지로 JSON 형식의 DID 문서**를 반환한다.

(c) DID 문서 예시:

```
{
  “@context”: [
    ‘https://www.w3.org/ns/did/v1’,
    ‘https://w3id/org/security/suites/ed25519-2020/v1’
  ],
  “id” : ‘did.example.123’,
  “authentication”: [{
    ‘id’: ‘did:example:123#z...’,
    ‘type’: ‘Ed25519VericationKey2020’,
    ‘controller’: ‘did:example:123’,
    ‘publicKeyMultibase’: ‘zAKJp.,...’
  }],
  ‘capabilityinovocation’: [...], ...
}
```

4. References

[1] W3C, “Decentralized Identifiers (DIDs) v1.0“, W3C Recommendation, 19 July 2022

[2] Mockapetris, P., “DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION“, RFC 1035, November 1987.

[3] Braden, R., “Requirements for Internet Hosts — Application and Support“, RFC 1123, October 1989.

[4] Elz, R., and Bush, R., “Clarifications to the DNS Specification“, RFC 2181, July 1997.

[5] Damas, J., and Vixie, P., “Extension Mechanisms for DNS (EDNS(0))“, RFC 6891, April 2013.

[6] Arends, R. and Austein, R. et al. ”DNS Security Introduction and Requirements“, RFC 4033, March 2005

[7] Arends, R. and Austein, R. et al. ”Resource Records for the DNS Security Extensions“, RFC 4034, March 2005

[8] Arends, R. and Austein, R. et al. ”Protocol Modifications for the DNS Security Extensions“, RFC 4035, March 2005

분산 ID(DID) 직접 해석방안 연구

인 쇄 : 2024 년 11 월
발 행 : 2024 년 11 월

발행인 : 이 상 중
발행처 : 한국인터넷진흥원(KISA, Korea Internet & Security Agency)
전라남도 나주시 진흥길 9
Tel: 1544-5118
인쇄처 : 시티기획
Tel: (062) 266-4884

<비매품>