- From the book *"Self-Sovereign Identity"*

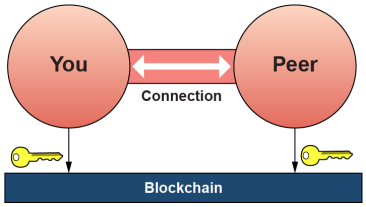
디지털 신원인증 모델

인터넷 신원인증 모델 발전 단계

- 중앙형 모델
  - You $\xrightarrow[\text{계정}]{}$ Org
  - 웹사이트는 사용자의 신원을 계정 가입을 통해 관리 - Centralized
  - 중앙 통제(사이트 관리자)에 의한 신원 관리

- 연합형 모델
  - You $\xrightarrow[\text{계정}]{}$ IDP $\longrightarrow$ Org
  - IDP(Identity Provider) : 서비스, 신원 제공자 (ex. Google, Facebook, Kakao)
  - 하나의 IDP에 계정 가입하여 여러사이트 이용 가능

- 탈중앙형 모델 - *Decentralized*
  - 위의 두가지 모델 모두 중앙 컨트롤 기관에서 신원을 확인하는 한계 해결
  - 공개/개인키 암호화 기법 기반 블록체인 사용
  - 블록체인 기술을 암호화폐가 아닌, DPKI (Decentralized PKI)에 적용
    - DID는 블록체인을 기반으로 한 신분증이라고 할 수 있습니다.
    - 이용자 스스로 자신의 신원정보를 관리하고 통제할 수 있도록 하는 디지털화된 신원관리 체계
    - 공개키 → 블록체인에 저장하여 디지털 신원 자격(VC) 서명(signature) 증명하는데 사용
    - 개인키 → 디지털 지갑에 저장
    - VC → 개인 기기에 저장
      - Verifiable Credentials: 실생활에서 신원증명 제공을 위해 교환 가능한 자격 증명 (디지털화 된 신분증) 입니다



- 공개키를 블록체인을 통해 직접 교환 하여 안전한 Peer-to-Peer 연결 생성하기 때문에 Decentralized라고 할 수 있습니다

암호화 기법

DID는 기존의 PKI에 사용되는 암호화 기법을 그대로 사용합니다 :

- 비대칭 키 기법 Asymmetric-key cryptography
  - 개인키(private)는 디지털지갑에 저장
  - 공개키(public)는 블록체인에 저장
- 디지털 서명
  - 공개키 암호화 기법 기반 : 메시지를 개인키로 암호화하여 디지털 서명 생성, 공개키로 서명 검증

DID 배경

PKI의 필요성

- Ip 주소 (ex. 172.20.10.11) 자체는 해당 ip를 소유한 신원 대상에 대한 어떠한 정보도 제공하지 않기 때문에,

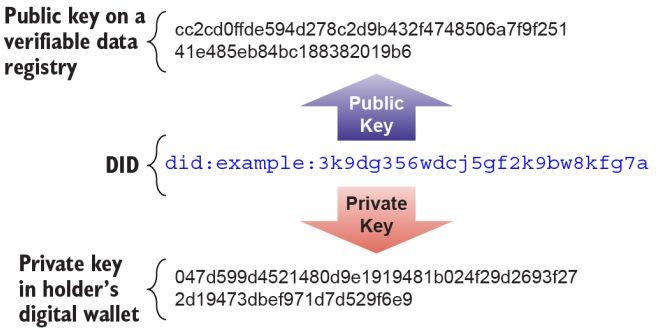
- 신원 주체자는 본인 임을 증명 할 수 있는, 디지털 Proof를 제공 해야합니다
  - 공개/개인키 암호화기법 으로 증명가능 한 디지털 Proof 제공
  - 개인키 로 메시지 서명(sign), 공개키 로 검증(verify)

PKI 도입

공개 키 기반 인프라 구조

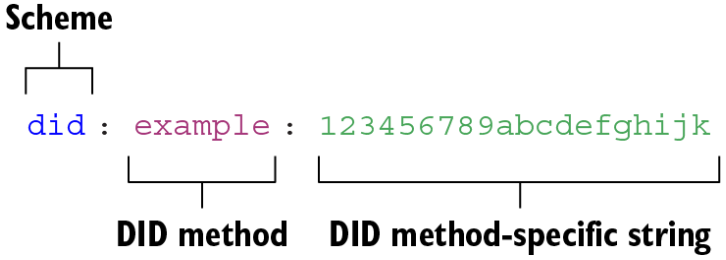
- PKI를 통해 공개키의 소유권 을 검증할 수 있게 되었습니다 - 신원 대상자의 공개키가 실제로 소유자의 것인지 여부 확인 할 수 있게 됨
- 검증된 CA기관들로 부터 공개키 인증서 발행하는 중앙집권형 시스템이므로
- 개인들이 여러개의 암호화 키페어를 가지고 있는 환경(Decentalized Identity model)에 한계가 있음

- 탈중앙 인증 식별자 DIDs으로 위의 한계를 해결 할 수 있으며, 다음 네가지 특성을 가집니다 :
  - 영속성 - Permanent (블록체인에 기록된 데이터는 tampered 될 수 없습니다)
  - 분해성 - DID Resolver를 통해 DID를 DID document로 분해(resolve)
  - 암호화기법으로 검증 가능 - Cryptographically verifiable
    - 신원 소유자가 암호화 기법으로 개인키 검증가능
    - 암호화기법으로 DID 생성
    - DID는 1개의 공개/개인키와 연결되므로 개인키 소유자(controller)가 DID 소유자(controller)임 증명 가능
  - 탈중앙 - Decentralized
    - 암호화기법을 사용하여 중앙 신원인증 기관들(CAs)의 통제 없이, 블록체인 등 탈중앙화 네트워크에 기반함
    - 공개/개인키 생성하는 암호화 알고리즘은 프라임넘버, 랜덤숫자생성기, 타원곡선 암호학에 기반하여
    - globally 고유한 식별자를 만들기 때문에 중앙기관 없이 Uniqueness 검증가능



DID 정의

- 새로운 타입의 고유 식별자 *globally unique identifier*
- DIDs 는 VC(신원자격 증명)를 암호화한 형태라고 볼 수 있습니다.(Cryptographic counterpart to Verifiable Credentials)
- DID는 블록체인에 공개키 주소로서 역할을 하며, DID subject의 agent를 찾는데도 사용 됩니다
- DID 메소드를 통해 블록체인, DLT(Distributed Ledger Technology) 등을 이용할 수 있도록 설계됨
- 소프트웨어를 통해 누구나 DID 메소드(sov,btcr,ethr, ...)를 사용하여 중앙기관 통제 없이 DID 발행 및 사용가능
- DID를 생성하는 것은 비트코인이나 이더리움 블록체인에 공개 지갑 주소를 생성하는 것과 동일한 프로세스 - DID 탈중앙화 핵심

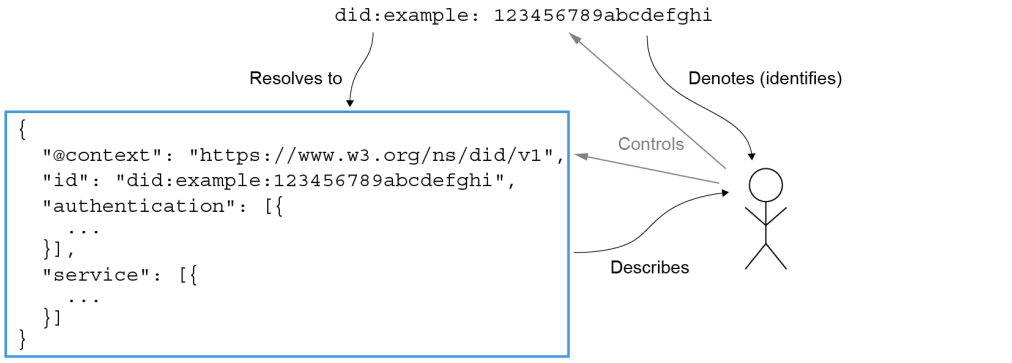


- DID 예시 데모

- 링크: https://www.youtube.com/watch?v=r28GeXkxn0w
- 예시 1
  - 개인키/공개키 페어 생성
  - 개인키 → 디지털 지갑 앱
  - 공개키 → 블록체인 (sovrin, bitcoin, ethereum, ...) 트랜젝션을 통해 암호화하여 저장
  - 블록체인은 응답으로 DID를 생성하여 반환
    - 은행 로그인 시 DID를 개인키로 서명하여 요청
    - 은행은 블록체인에서 DID와 연관된 트렌젝션 조회 & 공개키 조회
    - 공개키로 서명 검증 및 로그인 완료처리
- 예시 2
  - 학생정보 입력하여 학교 웹사이트 로그인
  - 대시보드에서 고유 디지털 ID 스캔 및 인증 → DID 고유 식별자 생성
    (개인키 생성 및 블록체인에 공개키 저장하여 DID 생성)
  - 온라인서적 사이트에서 DID 로그인

DID documents

- DID $\xrightarrow{\text{DID resolver(software/hardware)}}$ DID document
  - 디지털 신원인증 앱, 디지털 지갑, 또는 에이전트 등에서 인증을 위한 기초 빌딩블록 으로 사용
  - DID $\longleftrightarrow$ DID document (1-to-1 대응)
- DID document는 퍼블릭이므로 개인정보와 관련된 데이터는 넣지 않습니다
- DID document는 표준화된 규격 구조(json)를 가지고 있으며 다음 데이터 등을 포함 합니다 :
  - 공개키: 1개 이상의 DID subject의 공개키; 거래시 공개키로 subject 검증 - essence of DPKI, SSI
  - 서비스: 프로토콜을 통한 거래 시에 사용 할 DID subject 관련 서비스들
  - 메타데이터: 타임스탬프, 디지털서명, 암호학적proof, deleation 및 인증 관련 메타데이터



```
// 1개의 공개키와 1개의 서비스를 가진 DID document 구조
{
  // JSON-LD context statement,
  // required in JSON-LD documents (but not in other DID document representations).
  "@context": "https://www.w3.org/ns/did/v1",

  // DID subject (필수)
  "id": "did:example:123456789abcdefghi",

  // DID controller (옵션): DID document 변경 권한을 가짐
  "controller": "did:example:bcehfew7h32f32h7af3",

  // public key for authenticating the DID subject. (옵션)
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58" : "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  // 서비스 service endpoint(옵션) for exchanging verifiable credentials.
  // DID subject와 communication하는 방법
  "service": [{
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

DID methods

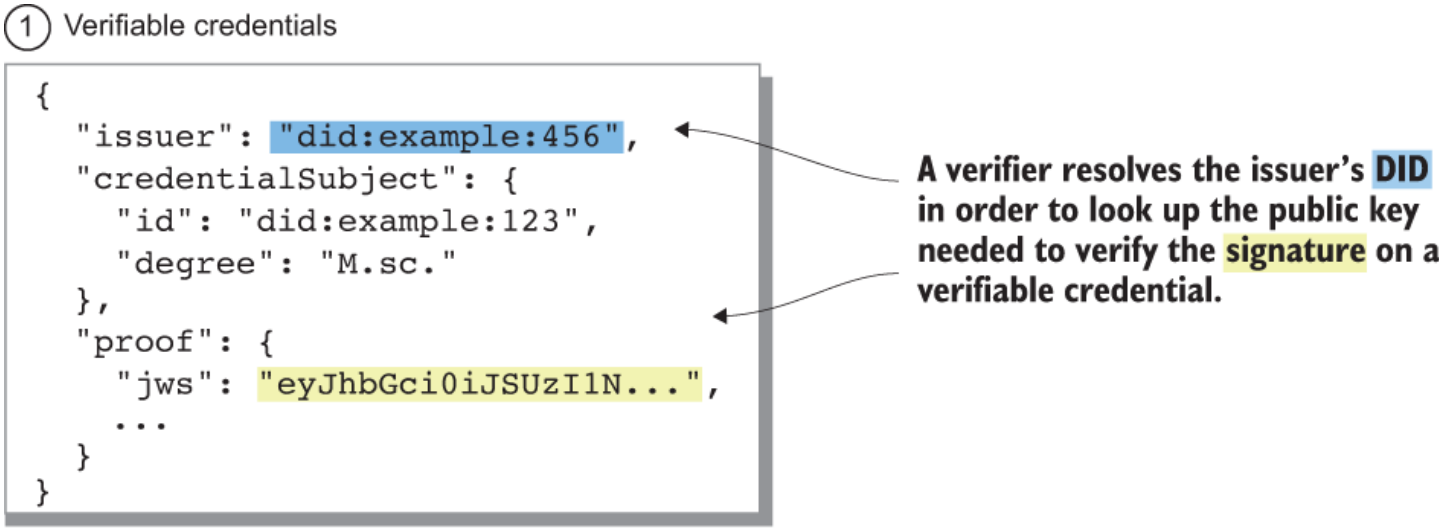
- 메소드별 상세스팩 : https://www.w3.org/TR/did-spec-registries/#did-methods
- 각 DID 메소드는 다음과 같은 기술적 정의가 요구됨:
  - 메소드 고유 식별 (예: sov,btcr,v1,ethr,jolo,...)
  - DID에 대한 CRUD 4가지 operation 수행 가능
    - 블록체인이나, 분산 ledger시스템에 기반한 DID메소드의 경우 create/update시 ledger에 트랜젝션 기록
  - 메소드에 따른 보안 및 개인정보보호 장치

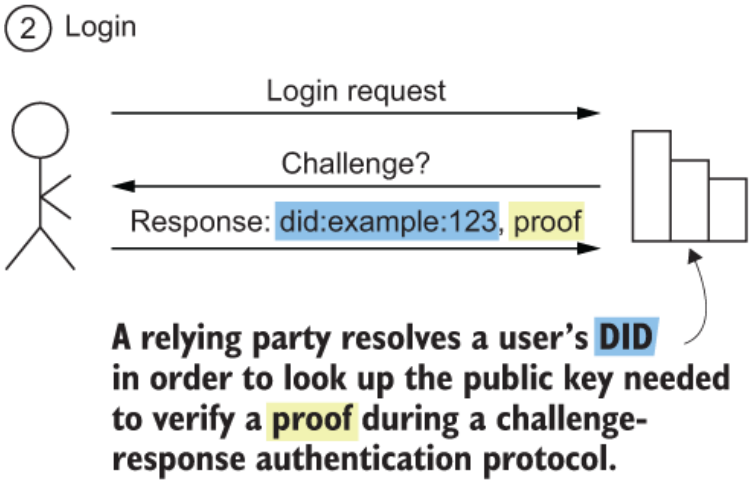
```
did:sov:WRfXPg8dantKVubE3HX8pw
did:btcr:xz35-jzv2-qqs2-9wjt
did:v1:test:nym:3AEJTDMSxDDQpyUftjuoeZ2Bazp4Bswj1ce7FJGybCUu
did:ethr:0xE6Fe788d8ca214A080b0f6aC7F48480b2AEfa9a6
did:jolo:1fb352353ff51248c5104b407f9c04c3666627fcf5a167d693c9fc84b75964e2
```

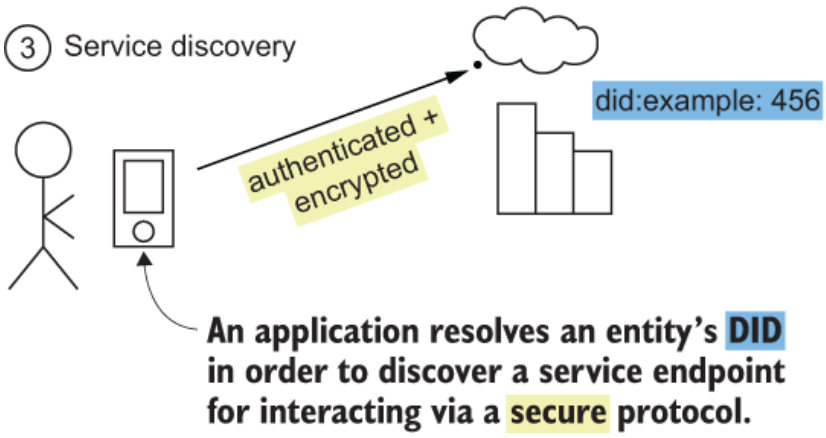
- did-rubric
  - 특정 DID 메소드가 얼마나 사용 커뮤니티 필요에 부합하는지 평가/측정 할 수 있는 문서

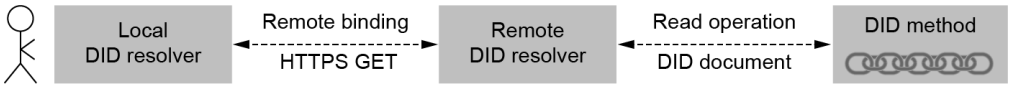
DID resolution

- DID로부터 DID method의 "Read" 오퍼레이션을 통해 DID document를 얻는 과정
- DID관련 앱이나 서비스가 DID document에서 DID subject 관련 메타데이터를 얻어 추가 상호작용 가능:
  - VC 발행자로 부터의 디지털 서명을 검증할 공개키 조회
    - VC의 "issuer"가 DID subject이고 proof를 DID subject의 공개키로 검증하는 예시
  - DID 컨트롤러가 웹사이트나 앱에 로그인해야할 때 검증 진행
    - 로그인 요청자의 DID resolve한 document에 있는 공개키로, 요청자의 proof 검증
  - 웹사이트, 소셜 네트워크 또는 라이선스 기관과 같은 DID 컨트롤러와 관련된 잘 알려진 서비스를 검색하고 액세스
  - DID 컨트롤러로 DID-to-DID 연결을 요청

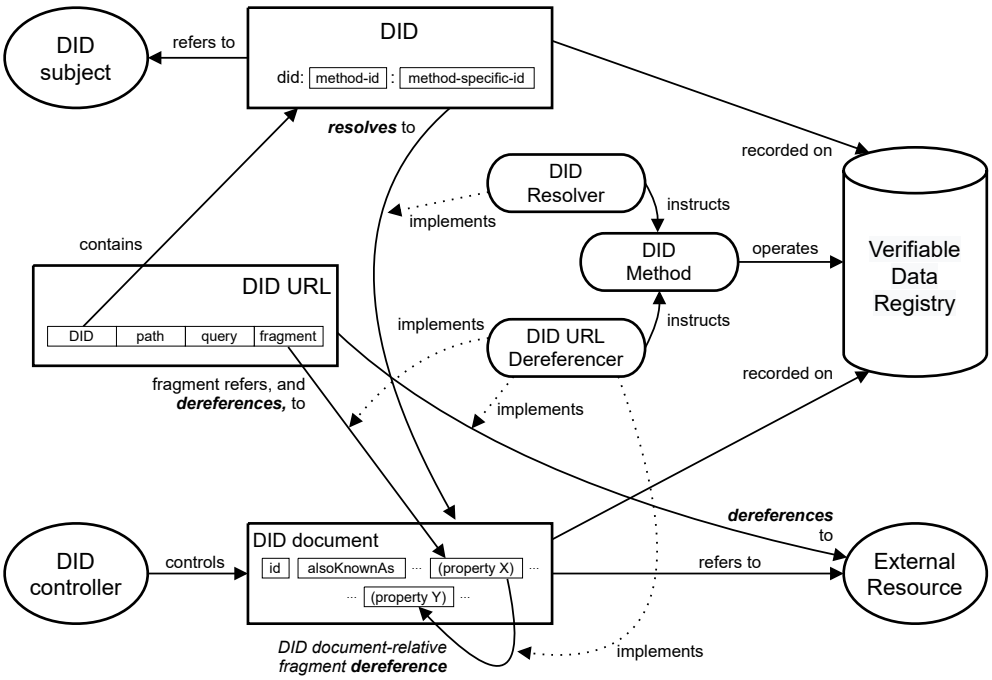

1 Verifiable credentials
{
"issuer": "did:example:456",
"credentialSubject": {
"id": "did:example:123",
"degree": "M.sc."
},
"proof": {
"jws": "eyJhbGci0iJSUzI1N...",
...
}
}
A verifier resolves the issuer's DID in order to look up the public key needed to verify the signature on a verifiable credential.


2 Login
Login request
Challenge?
Response: did:example:123, proof
A relying party resolves a user's DID in order to look up the public key needed to verify a proof during a challenge-response authentication protocol.


3 Service discovery
authenticated + encrypted
did:example: 456
An application resolves an entity's DID in order to discover a service endpoint for interacting via a secure protocol.


Local DID resolver
Remote binding
HTTPS GET
Remote DID resolver
Read operation
DID document
DID method

아키텍쳐 뷰



DID URLs

웹사이트 URL과 같이 DID도 쿼리 스트링을 통한 추가 파라미터 사용 가능

Table 8.1 Example DID URLs and their meanings

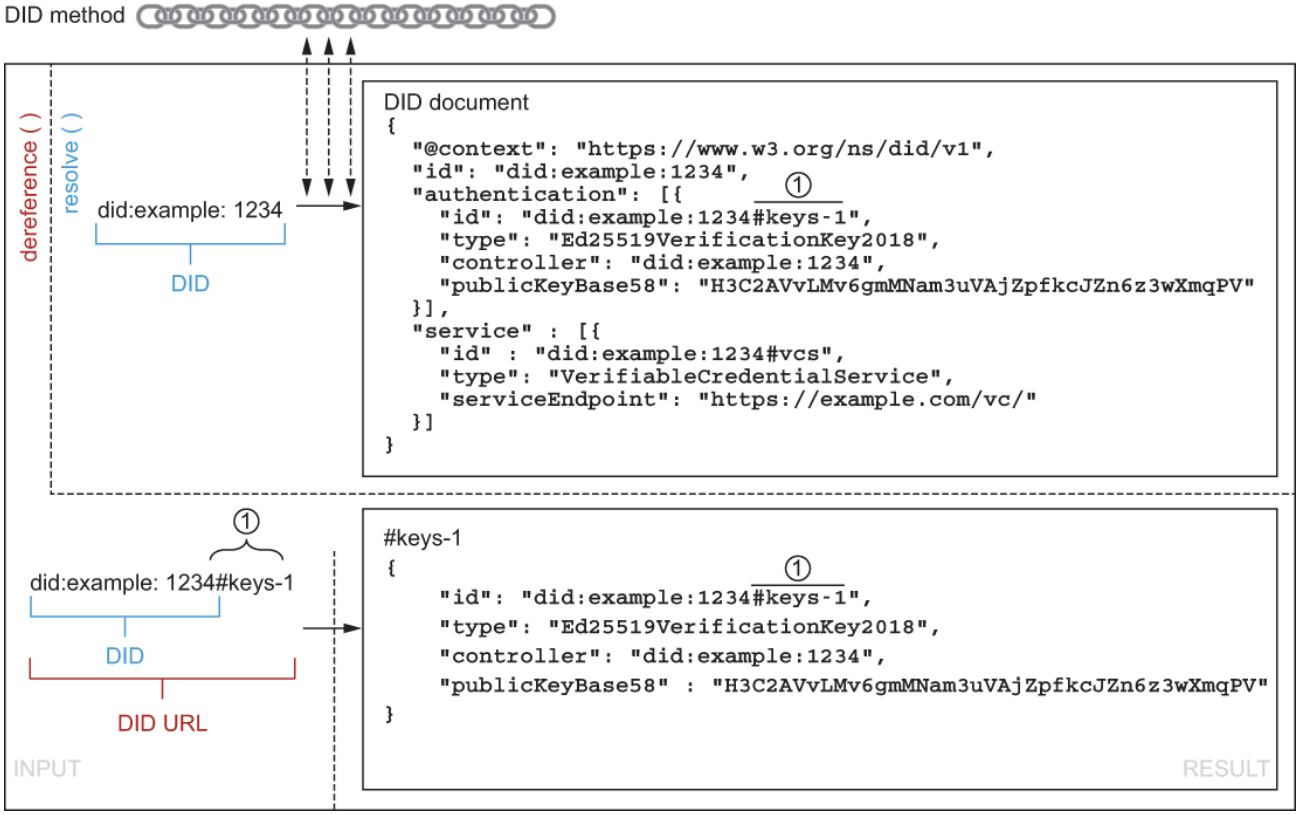
did:example:1234#keys-1
DID URL with a fragment. Identifies a specific public key inside the DID's associated DID document. This is similar to how an http: or https: URL with a fragment can point to a specific bookmark within an HTML web page.

did:example:1234;version-id=4
DID URL with a DID parameter (`version-id`). Identifies a previous version of the DID's associated DID document, as opposed to the latest version. This is useful when the contents of a DID document have been updated but a stable reference to a specific version is needed.

did:example:1234;version-id=4#keys-1
This DID URL is a combination of the previous two examples. It identifies a specific public key inside a specific previous version of the DID's associated DID document.

did:example:1234/my/path?query#fragment
DID URL with a path, query string, and fragment. The meaning and processing rules of this DID URL are not defined by the core DID standard but are dependent on the DID method and/or the application that consumes the DID URL.

did:example:1234;service=hub/my/path?query#fragment
DID URL with a DID parameter (`service`). Identifies a specific service inside the DID's associated DID document (in this case, the `hub` service). The meaning and processing rules of the remaining syntactic components (path, query string, fragment) are not defined by the core DID standard but are specific to the `hub` service.


DID method
dereference ( )
resolve ( )
did:example: 1234
DID
DID document
{
"@context": "https://www.w3.org/ns/did/v1",
"id": "did:example:1234",
"authentication": [{
①
"id": "did:example:1234#keys-1",
"type": "Ed25519VerificationKey2018",
"controller": "did:example:1234",
"publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJZn6z3wXmqPV"
}],
"service" : [{
"id" : "did:example:1234#vcs",
"type": "VerifiableCredentialService",
"serviceEndpoint": "https://example.com/vc/"
}]
}
①
did:example: 1234#keys-1
DID
DID URL
#keys-1
{
①
"id": "did:example:1234#keys-1",
"type": "Ed25519VerificationKey2018",
"controller": "did:example:1234",
"publicKeyBase58" : "H3C2AVvLMv6gmMNam3uVAjZpfkcJZn6z3wXmqPV"
}
INPUT
RESULT

DID 타입

Table 8.4 The broad categories of DID methods developed as of August 2020

Category	Description and examples

Ledger-based DIDs	The original category of DID methods involves a blockchain or other distributed ledger technology (DLT), which serves the purpose of a registry that is not controlled by a single authority. This registry is typically public and globally accessible. A DID is created/updated/ deactivated by writing a transaction to the ledger, which is signed with the DID controller's private key:
	`did:sov:WRfXPg8dantKVubE3HX8pw`
	`did:btcr:xz35-jzv2-qqs2-9wjt`
	`did:ethr:0xE6Fe788d8ca214A080b0f6aC7F48480b2AEfa9a6`
	`did:v1:test:nym:3AEJTDMSxDDQpyUftjuoeZ2Bazp4Bswj1ce7FJGybCUu`

Ledger middleware (Layer 2) DIDs	An improvement to classic ledger-based DID methods, this category adds an additional storage layer such as a distributed hash table (DHT) or traditional replicated database system on top of the base layer blockchain. DIDs can be created/updated/deactivated at this second layer without requiring a base layer ledger transaction every time. Instead, multiple DID operations are batched into a single ledger transaction, increasing performance and decreasing cost:
	`did:ion:test:EiDk2RpPVuC4wNANUTn_4YXJczjzi10zLG1XE4AjkcGOLA`
	`did:elem:EiB9htZdL3stukrklAnJ0hrWuCdXwR27TNDO7Fh9HGWDGg`

Peer DIDs	This special category of DID method does not require a globally shared registration layer such as a blockchain. Instead, a DID is created and subsequently shared with only one other peer (or a relatively small group of peers). The DIDs that are part of the relationship are exchanged via a peer-to-peer protocol, resulting in private connections between the participants (see https://identity.foundation/peer-did-method-spec/index.html): `did:peer:1zQmZMygzYqNwU6Uhmewx5Xepf2VLp5S4HLSwwgf2aiKZuwa`
Static DIDs	There is a category of DID methods that are "static", i.e. they enable a DID to be created and resolved, but not updated or deactivated. Such DID methods tend to not require complex protocols or storage infrastructure. For example, a DID may simply be a "wrapped" public key, from which an entire DID document can be resolved algorithmically, without requiring any data other than the DID itself: `did:key:z6Mkfriq1MqLBoPWecGoDLjguo1sB9brj6wT3qZ5BxkKpuP6`

Alternative DIDs	A number of other innovative DID methods have been developed that do not fall into any of the previous categories. They demonstrate that DID identification architecture is flexible enough to be layered on top of existing internet protocols, such as Git, the Interplanetary File System (IPFS), or even the web itself:
	`did:git:625557b5a9cdf399205820a2a716da897e2f9657`
	`did:ipid:QmYA7p467t4BGgBL4NmyHtsXMoPrYH9b3kSG6dbgFYskJm`
	`did:web:uport.me`

DID가 작동하는 이유 (아키텍쳐 관점)

- identity보다 cryptography의 관점에서 DID가 왜 작동하는지
- Public Key Infrastructure (PKI)의 문제점
  - 해결책 1: 전통적 PKI 모델
  - 해결책 2: web-of-trust 모델
  - 해결책 3: 공개키 기반 식별자 (Public key-based identifiers)
  - 해결책 4: DIDs and DID documents
    - DID controller 공개/개인키 기반 식별자 (DID) 생성
    - DID controller 공개/개인키 변경 시
      - 새로운 DID document를 이전 document의 개인키로 서명 (chain of trust)
      - DID document는 CA없이도 공개키에 대한 digital certificate 역할 수행
- DIDs의 4가지 장점 (that go beyond PKI)
  - 1: Guardianship and controllership

    - 2: Service endpoint discovery
    - 3: DID-to-DID connections
    - 4: Privacy by design at scale

DID의 의미

- 주소는 자체적으로 존재하지 않으며, 그것들을 사용하는 네트워크의 컨텍스트에서 존재

Origin	Address type	Network
1994	Persistent address (URN)	World Wide Web (machine-friendly)
1994	Web address (URL)	World Wide Web (human-friendly)
2003	Social network address	Social network
2009	Blockchain address	Blockchain or distributed ledger network
2016	DID	DID network

- 기타 리소스
    - 2020-02-SSI 웨비나 자료
    - COOV DID 구현
        - infra 메소드 스펙