Decentralised IDentifiers and Friends

Vurucu ve Akillica Alt Başlık

Abdulhamit Kumru

Blokzincir Laboratuvarı

2020

License Attribution-ShareAlike 4.0 International

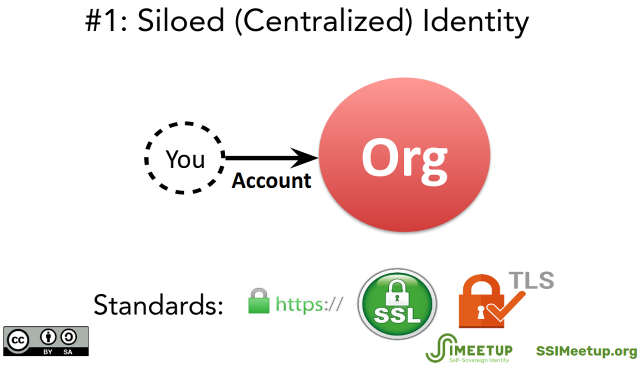
Nelerden bahsedecegiz

- DID Core
  - DID-common-java
- DID Auth
  - rwot 2018 did auth paper
  - DID TLS
  - guncel
- DID Auth + Current Auth Protocols
  - Proposed Applications
    - DID SIOP *identity.foundation/did-siop/*
  - Similar Applicable Scenarios
    - DID SAML (IdP)
    - DID CAS (IdP)
    - DID PAM
- DID Communication ?
- Peer DID ?
- V1 (Aries), V2 (DIF DIDcomm messaging) ?

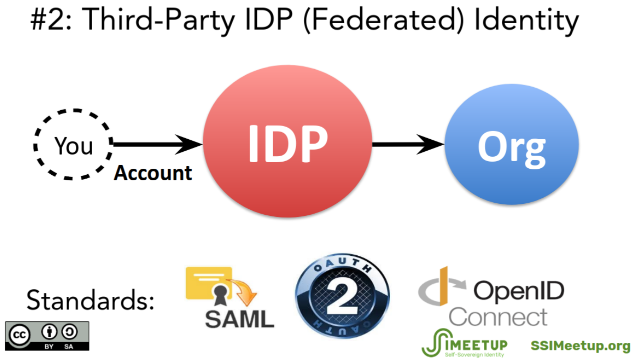
Daha Soyut Bir giris

!!! gereginden fazla degindigim yerler olabilir !!! giris slayti
!!! ikna edici bir giris hazirla
!!! did in argumanlarini daha belirgin yap
!!! aktif gelistirilen did methodlarindan bahset !!! oncesi 15 daki !!! Authentication a kadar hizli gec !!! json ld yi iyi anla, sunumda bahset !!! kimlik yontemlerini ozetleyen gorselleri ekle !!! didlerin kendini ispat mekanizmalari

Centralised ID



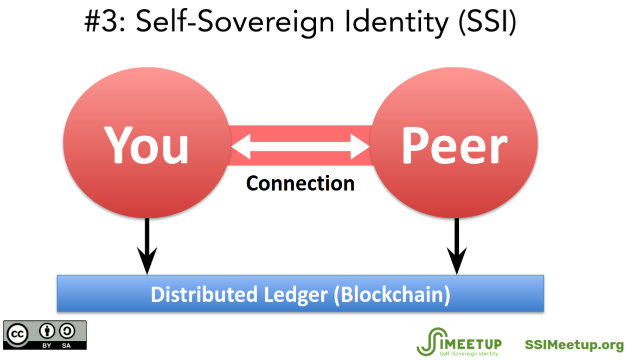
Federated ID


#2: Third-Party IDP (Federated) Identity
You
Account
IDP
Org
Standards:
SAML
OAUTH 2
OpenID Connect
SIMEETUP Self-Sovereign Identity
SSIMeetup.org
CC BY SA

Self-Sovereign Identity (SSI)

!!! not: ssi did baglantisi notlari al

!!! kisaca SSI ya degin

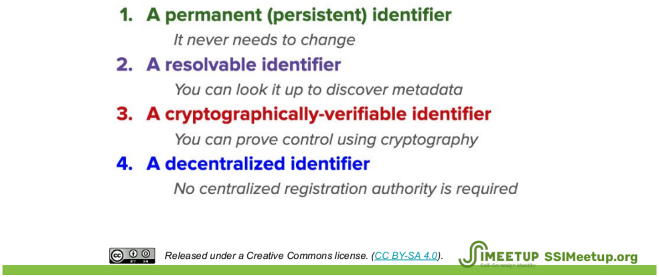


DID Core

!!! bu spec hakkinda genel bilgiler

Four Core Properities of DID

!!! notlar ve bura uzerinde biraz dur

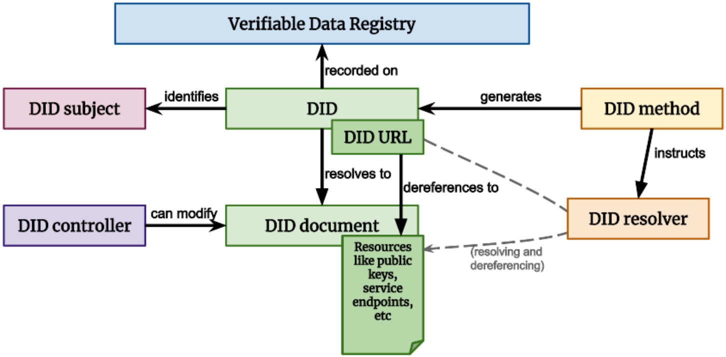


1. **A permanent (persistent) identifier**
   It never needs to change
2. **A resolvable identifier**
   You can look it up to discover metadata
3. **A cryptographically-verifiable identifier**
   You can prove control using cryptography
4. **A decentralized identifier**
   No centralized registration authority is required

Released under a Creative Commons license. (CC BY-SA 4.0). SSIMEETUP SSIMeetup.org

Architecture Overview

!!! not al



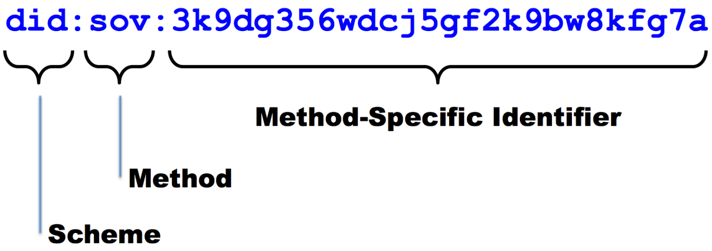
DIDs and DID URLs

!!! not: query ye detayli deginecegiz

A DID, or Decentralized Identifier, is a URI composed of three parts: ***the scheme*** “did:”, a ***method identifier***, and a unique, ***method-specific identifier*** generated by the DID method.

DIDs are resolvable to DID documents. A DID URL extends the syntax of a basic DID to incorporate other standard URI components (path, query, fragment) in order to locate a particular resource.

DID Format


did:sov:3k9dg356wdcj5gf2k9bw8kfg7a
Method-Specific Identifier
Method
Scheme

DID Subjects

!!! gorsel ekle ?

The subject of a DID is, by definition, the entity identified by the DID. The DID subject may also be the DID controller. Anything can be the subject of a DID: person, group, organization, physical thing, logical thing, etc.

DID Controllers

!!! kisalt
!!! gorsel ekle ?

The controller of a DID is the entity (person, organization, or autonomous software) that has the capability—as defined by a DID method—to make changes to a DID document. This capability is typically asserted by the control of a set of cryptographic keys used by software acting on behalf of the controller, though it may also be asserted via other mechanisms. Note that a DID may have more than one controller, and the DID subject can be the DID controller, or one of them.

Verifiable Data Registries

!!! kisalt
!!! gorsel ekle ?

In order to be resolvable to DID documents, DIDs are typically recorded on an underlying system or network of some kind. Regardless of the specific technology used, any such system that supports recording DIDs and returning data necessary to produce DID documents is called a verifiable data registry. Examples include distributed ledgers, decentralized file systems, databases of any kind, peer-to-peer networks, and other forms of trusted data storage.

DID documents

DID documents contain metadata associated with a DID. They typically express verification methods (such as public keys) and services relevant to interactions with the DID subject.

Minimal Self-managed DID document Example

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCw..."
  }],
  "service": [{
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

DID Methods

DID methods are the mechanism by which a particular type of DID and its associated DID document are created, resolved, updated, and deactivated using a particular verifiable data registry. DID methods are defined using separate DID method specifications.

DID resolvers and DID resolution

!!! not: detayli spec linkte, burda did res. in detayina girmeyecegiz

A DID resolver is a software and/or hardware component that takes a DID (and associated input metadata) as input and produces a conforming DID document (and associated metadata) as output. This process is called DID resolution.

detailed spec w3c-ccg.github.io/did-resolution/

DID Resolve Example

!!! gecis

```
did:example:1234;version-id=4#keys-1 # resolves to

{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi#keys-1",
  "type": "RsaVerificationKey2018",
  "publicKeyPem": "-----BEGIN PUB...0101010..END PUB -----\r\n"
}
```

Identifier

!!! identifier giris slayti !!! cevir

did ve did urllerinin syntaxini inceleyecegiz, generic terimi burda tanimlanan syntaxin diger did methodlarinda tanimlanabilecek syntaxlardan ayirtd edilmek amaciyla kullanildi

This section describes the formal syntax for DIDs and DID URLs. The term “generic” is used to differentiate the syntax defined here from syntax defined by specific DID methods in their respective specifications.

DID Syntax

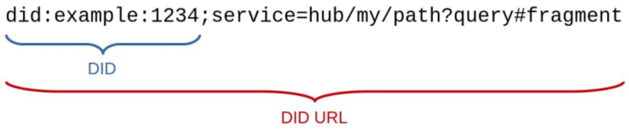
- The generic DID scheme is a URI scheme conformant with [RFC3988].
- The DID scheme name **MUST** be an ASCII lowercase string.
- The DID method name **MUST** be an ASCII lowercase string.

```
# Ethr-DID
did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a
```

A DID is expected to be persistent and immutable. That is, a DID is bound exclusively and permanently to its one and only subject. Even after a DID is deactivated, it is intended that it never be repurposed.

- aciklama nota alinabilir

DID URL Syntax



```
did:example:1234;service=hub/my/path?query#fragment
```

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

DID Parameters

The DID URL syntax supports a simple format for parameters based on the query component. Adding a DID parameter to a DID URL means that the parameter becomes part of the identifier for a resource.

DID Parameters

Relative Reference

A relative URI reference according to RFC3986 Section 4.2 that identifies a resource at a service endpoint, which is selected from a DID document by using the service parameter. Support for this parameter is **REQUIERED**

Relative Reference Example

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:1234",
  "verificationMethod": [{
    "id": "did:example:1234#key-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:1234",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwn..."
  }, ...],
  "authentication": [
    // relative DID URL to ‘did:example:1234#key-1
    "#key-1"
  ]
}
```

DID Parameters

service

Identifies a service from the DID document by service ID. Support for this parameter is **REQUIRED**

```
did:foo:21tDAKCERh95uGgKbJNHYp?service=agent
```

DID Parameters

version-id

Identifies a specific version of a DID document to be resolved (the version ID could be sequential, or a UUID, or method-specific). Support for this parameter is **OPTIONAL**

DID Parameters

version-time

Identifies a certain version timestamp of a DID document to be resolved. Support for this parameter is **OPTIONAL**

```
did:foo:21tDAKCERh95uGgKbJNHYp?version-time=2002-10-10T17:00:00Z
```

DID Parameters

hl

!!! not: iyi bir ozellik ama mekanizmasini tam olarak anlayamadim

A resource hash of the DID document to add integrity protection, as specified in Hashlink RFC. This parameter is *non-normative*

```
# url encoded hash link
hl:zm9YZpCjPLPJ4Epc:z3TSgXTuaHxY2ts...7DYuQ9QTPQyLHy
```

DID URL Syntax

Path

A DID path is identical to a generic URI path

```
did:example:123456/path
```

Query

A DID query is derived from a generic URI query and **MUST** conform to DID URL Syntax rules.
If a DID query is present, it **MUST** be used with DID Parameters.

```
did:example:123456?query=true
```

DID URL Syntax

Fragment

A DID fragment is used as method-independent reference into a DID document or external resource. DID fragment syntax and semantics are identical to a generic URI fragment and **MUST** conform to RFC 3986

```
did:example:123#agent # service endpoint
did:example:123#public-key-0 # verification method
```

Relative DID URLs

!!! buraya biraz daha bak !!! ornegine deginmistik
A relative DID URL is any URL value in a DID document that does not start with `did:<method-name>:<method-specific-id>`.

```
// ... relative DID URL to ‘did:example:1234#key-1‘
"authentication": [ "#key-1" ]
// ...
```

Example DID URLs

!!! gecis !!! not: did url ye degin

```
# A DID URL with a 'service' DID parameter
did:foo:21tDAKCERh95uGgKbJNHYp?service=agent
# A DID URL with a 'version-time' DID parameter
did:foo:21tD...gKbJNHYp?version-time=2002-10-10T17:00:00Z

did:example:1234/
did:example:1234#keys-1
did:example:1234;version-id=4#keys-1
did:example:1234/my/path?query#fragment
did:example:1234;service=hub/my/path?query#fragment
```

Core Properties

!!! core prop giris slayti
!!! hepsini anlatmaya gerek yok

- id
- authentication
- controller
- service
- verificationMethod
- assertionMethod
- keyAgreement
- capabilityDelegation ?
- capabilityInvocation ?

id

The DID subject is denoted with the ***id*** property at the top level of a DID document.

- The DID subject is the entity that the DID document is about
- DID documents **MUST** include the id property at the top level.

```
{
  "id": "did:example:21tDAKCERh95uGgKbJNHYp"
}
```

alsoKnownAs

- A DID subject can have *multiple identifiers* for different purposes, or at different times.
- The assertion that two or more DIDs (or other types of URI) identify the same DID subject can be made using the ***alsoKnownAs*** property.

Control

!!! not: did doc may have controller, illa controller olacak diye birsey yok
!!! not: no longer has access to their keys, or key compromise, where the DID controller's trusted third parties need to override malicious activity by an attacker. bunu anla

Authorization is the mechanism used to state how operations are performed on ***behalf*** of the DID subject. ***A DID controller is authorized*** to make changes to the respective DID document.

Note: Authorization vs Authentication !

DID Document With a Controller Property

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "controller": "did:example:bcehfew7h32f32h7af3",
  "service": [{

    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Verification Methods

!!! not: did controller vs verification method anla not al !!! not: A DID document MAY include a verificationMethod property. !!! not: cok detayli kisa gec

A DID document can express verification methods, such as cryptographic keys, which can be used to authenticate or authorize interactions with the DID subject or associated parties

- The information expressed often includes globally unambiguous identifiers and public key material, which can be used to verify digital signatures.
- In order to maximize interoperability, support for public keys as verification methods is restricted

verificationMethod

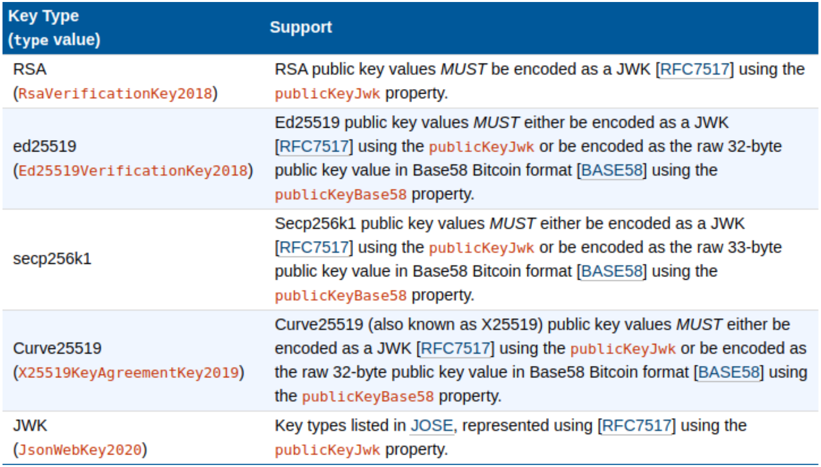
If a DID document includes a verificationMethod property, the value of the property **MUST** be an ordered set of verification methods

- The properties **MUST** include the ***id, type, controller, and specific verification method properties*** , and MAY include additional properties.
- The value of the ***id*** property for a verification method **MUST be a URI**.

Example Verification Methods

```
{
  "@context": ["https://www.w3.org/ns/did/v1", "https://w3id.org
  "id": "did:example:123456789abcdefghi",
  ...
  "verificationMethod": [{
    "id": ...,
    "type": ...,
    "controller": ...,
    ...
  }]
}
```

Key types and formats



Key Type (type value)	Support
RSA (`RsaVerificationKey2018`)	RSA public key values *MUST* be encoded as a JWK [RFC7517] using the `publicKeyJwk` property.
ed25519 (`Ed25519VerificationKey2018`)	Ed25519 public key values *MUST* either be encoded as a JWK [RFC7517] using the `publicKeyJwk` or be encoded as the raw 32-byte public key value in Base58 Bitcoin format [BASE58] using the `publicKeyBase58` property.
secp256k1	Secp256k1 public key values *MUST* either be encoded as a JWK [RFC7517] using the `publicKeyJwk` or be encoded as the raw 33-byte public key value in Base58 Bitcoin format [BASE58] using the `publicKeyBase58` property.
Curve25519 (`X25519KeyAgreementKey2019`)	Curve25519 (also known as X25519) public key values *MUST* either be encoded as a JWK [RFC7517] using the `publicKeyJwk` or be encoded as the raw 32-byte public key value in Base58 Bitcoin format [BASE58] using the `publicKeyBase58` property.
JWK (`JsonWebKey2020`)	Key types listed in JOSE, represented using [RFC7517] using the `publicKeyJwk` property.

Software/Tools

!!! did core sonu ekle

did-common-java

Sources

- https://www.w3.org/TR/did-core/
- https://w3c-ccg.github.io/did-resolution/
- https://github.com/WebOfTrustInfo/rwot5-boston/blob/master/topics-and-advance-readings/did-primer.md
- https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/
- https://docs.google.com/presentation/d/1Iv98aPWuZmRwiF01VtRjYgFHrm