DID-based Auth Protocols

Vurucu ve Akillica Alt Başlık

Abdulhamit Kumru

Blokzincir Laboratuvarı

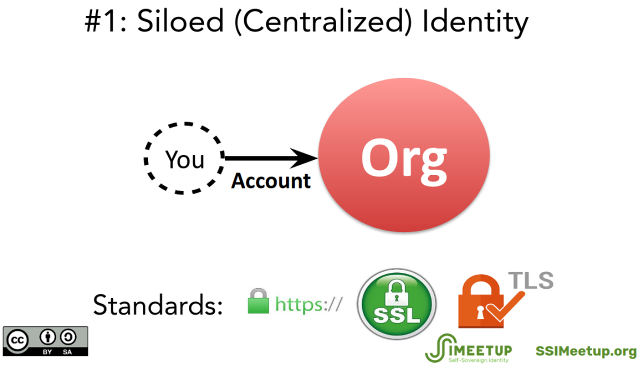
2020

Nelerden bahsedecegiz

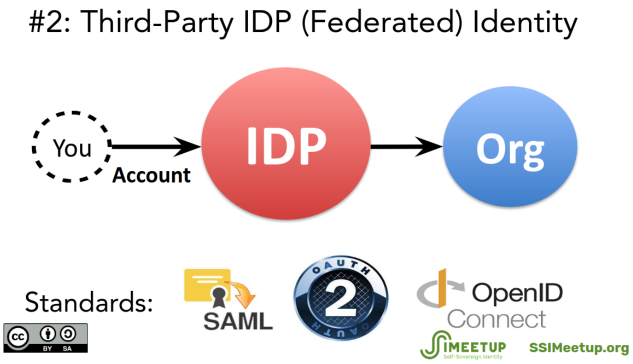
DID Core

- w3c DID Spec

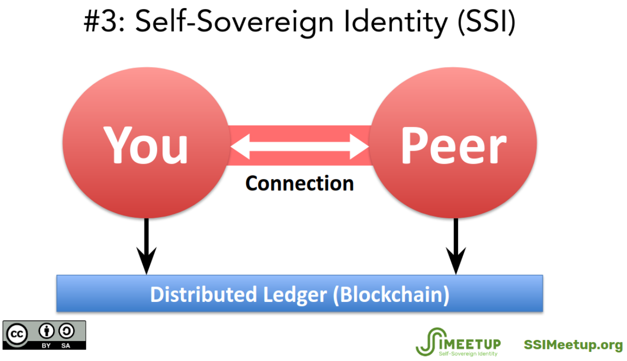
Centralised ID


#1: Siloed (Centralized) Identity
You
Account
Org
Standards:
https://
SSL
TLS
SIMEETUP Self-Sovereign Identity
SSIMeetup.org
BY SA

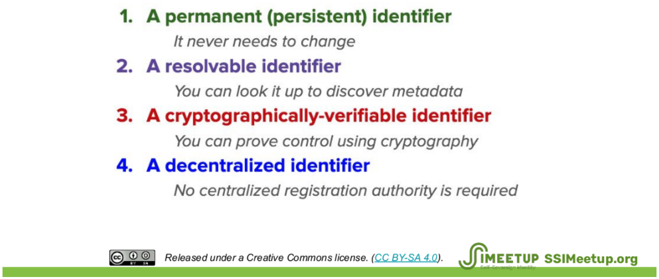
Federated ID


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

Self-Sovereign Identity (SSI)

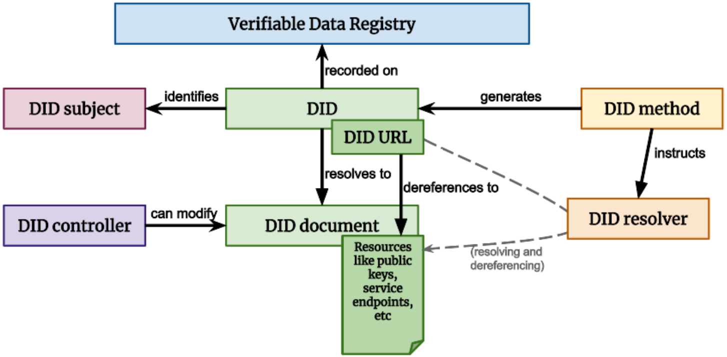


Four Core Properities of DID



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

Architecture Overview



DIDs and DID URLs

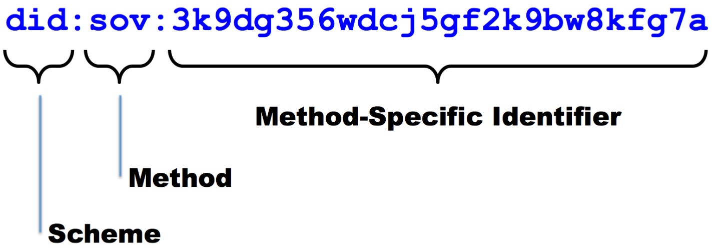
Architecture Overwiew

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

Architecture Overwiew


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

DID Subjects

Architecture Overwiew

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

Architecture Overwiew

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

Architecture Overwiew

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

Architecture Overwiew

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

Architecture Overwiew

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

Architecture Overwiew

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

```
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

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.

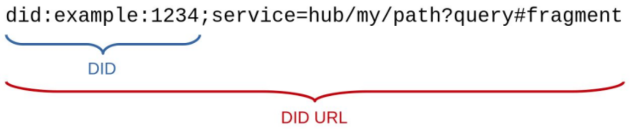
DID Syntax

- The generic DID scheme is a URI scheme conformant with [RFC3988].
- The DID scheme and 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.

DID URL Syntax



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

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

DID Parameters

DID URL Syntax

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

DID URL Syntax

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

DID URL Syntax

```
{
  "@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

DID URL Syntax

service parameter

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

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

DID Parameters

DID URL Syntax

version-id parameter

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

DID URL Syntax

version-time parameter

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

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

DID Parameters

DID URL Syntax

hl patameter

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

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

```
# 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

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

id Property

DID Subject

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

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

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. A DID document MAY include a verificationMethod property.

- The information expressed often includes globally unambiguous identifiers and public key material, which can be used to verify digital signatures.
- Verification methods might take many parameters. An example of this is a set of five cryptographic keys from which any three are required to contribute to a threshold signature.
- In order to maximize interoperability, support for public keys as verification methods is restricted

verificationMethod Property

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**.

Note: Verification method controller(s) and DID controller(s)

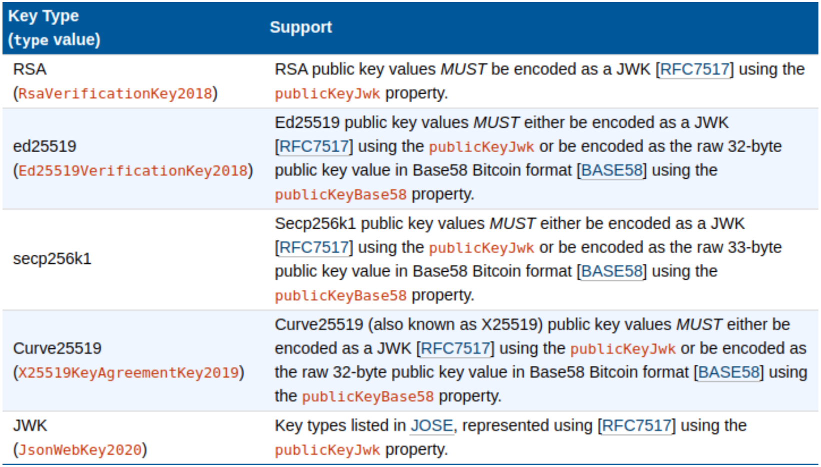
As well as the ***verificationMethod*** property, verification methods can be embedded in or referenced from properties associated with various verification relationships

Embedding and referencing verification methods

```
{ ...  "authentication": [
  // this key is referenced
  it may be used with 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": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AV...z3wXmqPV"
  }
], ... }
```

Key types and formats

Verification Methods



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.

Verification Relationships

A verification relationship expresses the relationship between the DID subject and a verification method.

Different verification relationships enable the associated verification methods to be used for different purposes

Verification Relationships

Authentication

The ***authentication*** verification relationship is used to specify how the DID subject is expected to be authenticated, such as for the purposes of logging into a website

Assertion

The ***assertionMethod*** verification relationship is used to specify how the DID subject is expected to express claims, such as for the purposes of issuing a Verifiable Credential

Key Agreement

The ***keyAgreement*** verification relationship is used to specify how to encrypt information to the DID subject, such as for the purposes of establishing a secure communication channel with the recipient

Verification Relationships

Capacity Invocation

The ***capabilityInvocation*** verification relationship is used to specify a mechanism that might be used by the DID subject to invoke a cryptographic capability, such as the authorization to access an HTTP API

Capacity Delegation

The ***capabilityDelegation*** verification relationship is used to specify a mechanism that might be used by the DID subject to delegate a cryptographic capability to another party, such as delegating the authority to access a specific HTTP API to a subordinate

Service Endpoints

Service endpoints are used in DID documents to express ways of communicating with the DID subject or associated entities. Services listed in the DID document can contain information about privacy preserving messaging services, or more public information, such as social media accounts, personal websites, and email addresses although this is discouraged

One of the primary purposes of a DID document is to enable discovery of service endpoints. A service endpoint can be any type of service the DID subject wants to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction

Various service endpoints

```
{ // ...
  "service": [{
    "id": "did:example:123456789abcdefghi#openid",
    "type": "OpenIdConnectVersion1.0Service",
    "serviceEndpoint": "https://openid.example.com/"
  }, {
    "id": "did:example:123456789abcdefghi#vcr",
    "type": "CredentialRepositoryService",
    "serviceEndpoint": "https://repository.example.com/service/8
  }, {
    "id": "did:example:123456789abcdefghi#xdi",
    "type": "XdiService",
    "serviceEndpoint": "https://xdi.example.com/8377464"
  }]
// ... }
```

Core Representations

All concrete representations of a DID document are serialized using a deterministic mapping that is able to be unambiguously parsed into the data model defined in this specification

Producers MUST indicate which representation of a document has been used via a media type in the document's metadata. Consumers MUST determine the representation of a DID document via the content-type DID resolver metadata field (see § 8.1 DID Resolution), not through the content of the DID document alone.

- JSON
- JSON-LD
- Concise Binary Object Representation (CBOR)

Methods

DID methods provide the means to implement did core specification on different *verifiable data registries*.

- The DID method specification **MUST** specify how to generate the *method-specific-id* component of a DID.
- The *method-specific-id* value **MUST** be able to be generated without the use of a centralized registry service.
- Each DID method **MUST** define how authorization is implemented, including any necessary cryptographic operations.

Note: Unique DID method names

Method Operations

Create

The DID method specification **MUST** specify how a DID controller creates a DID and its associated DID document on the verifiable data registry, including all cryptographic operations necessary to establish proof of control.

Read/Verify

The DID method specification **MUST** specify how a DID resolver uses a DID to request a DID document from the verifiable data registry, including how the DID resolver can verify the authenticity of the response.

Method Operations

Update

The DID method specification **MUST** specify how a DID controller can update a DID document on the verifiable data registry, including all cryptographic operations necessary to establish proof of control, or state that updates are not possible

Deactivate

The DID method specification **MUST** specify how a DID controller can deactivate a DID on the verifiable data registry, including all cryptographic operations necessary to establish proof of deactivation, or state that deactivation is not possible.
Note: Check Out Method Security & Privacy Requirements

DID Resolution

The DID resolution functions resolve a DID into a DID document by using the “Read” operation of the applicable DID method.

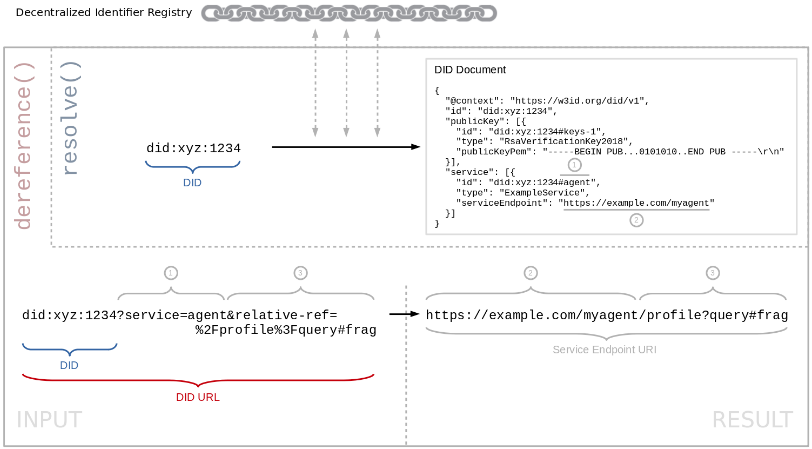
```
resolve ( did, did-resolution-input-metadata )
     -> ( did-resolution-metadata, did-document,
     did-document-metadata )
```

DID URL Dereferencing

The DID URL dereferencing function dereferences a DID URL into a resource with contents depending on the DID URL's components, including the DID method, method-specific identifier, path, query, and fragment

```
dereference ( did-url, did-url-dereferencing-input-metadata )
         -> ( did-url-dereferencing-metadata,
         content-stream, content-metadata )
```

DID URL Dereferencing


Decentralized Identifier Registry
dereference()
resolve()
did:xyz:1234
DID
DID Document
1
2
3
did:xyz:1234?service=agent&relative-ref=%2Fprofile%3Fquery#frag
DID
DID URL
https://example.com/myagent/profile?query#frag
Service Endpoint URI
INPUT
RESULT

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:xyz:1234",
  "publicKey": [{
    "id": "did:xyz:1234#keys-1",
    "type": "RsaVerificationKey2018",
    "publicKeyPem": "-----BEGIN PUB...0101010..END PUB -----\r\n"
  }],
  "service": [{
    "id": "did:xyz:1234#agent",
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/myagent"
  }]
}
```

Software / Repos

- dif/did-common-java
- w3c/did-use-cases