

# DID-based Auth Protocols

Vurucu ve Akillica Alt Başlık

Abdulhamit Kumru

Blokzincir Laboratuvarı

2020



License Attribution-ShareAlike 4.0 International

# Nelerden bahsedecegiz

## DID Core

- ▶ w3c DID Spec
- ▶ DID-common-java

## DID Auth

- ▶ DID TLS (Agust 2017) *unresolved*
- ▶ RWOT 2018 DID Auth (March 2018)
- ▶ indt\_auth
- ▶ did-auth-jose (October 2018, ietf, dif) *archived sept 22*
  - ▶ Javascript Object Signing and Encryption (jwt, jws, jwe, jwa)
- ▶ DID SIOP *identity.foundation/did-siop*
  - ▶ Javascript Object Signing and Encryption (jwt, jws, jwe, jwa)
  - ▶ Current Focus

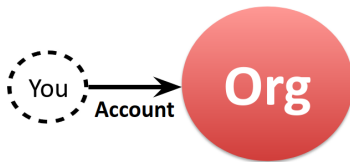
# Nelerden bahsedecegiz


## Current Auth Protocols & DID Auth

- ▶ web auth kısa
- ▶ single sign on
- ▶ single sign on ile basic auth farki
- ▶ Hali Hazirdaki Protokoller Yapilar
  - ▶ SAML
  - ▶ CAS
  - ▶ OpenID Connect
- ▶ karsilasitirilmalari
- ▶ Self-Issued OpenID Connect Provider DID Profile (did-siop, DIF)
- ▶ DID SAML ?
- ▶ DID CAS ?
- ▶ DID Authentication in PAM ?



## #1: Siloed (Centralized) Identity



Standards:  <https://>



https://



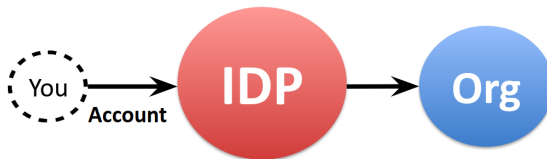
## TLS



**SSIMeetup.org**

# Federated ID

## #2: Third-Party IDP (Federated) Identity



Standards:

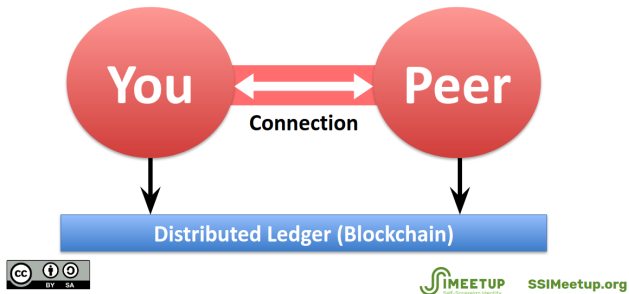


SSIMeetup.org

# Self-Sovereign Identity (SSI)

!!! not: ssi did baglantisi acikla, notlarini al !!! kisaca SSI ya degin

## #3: Self-Sovereign Identity (SSI)





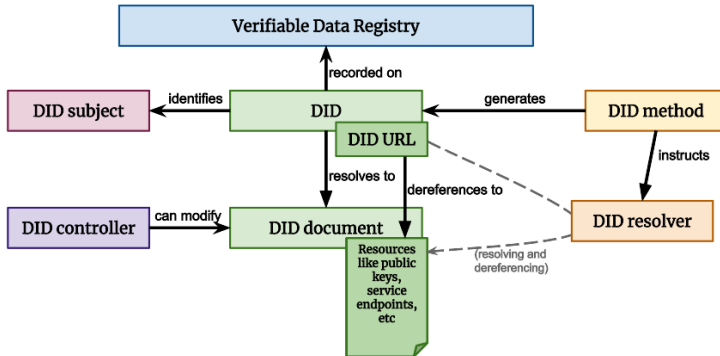
!!! DID Core giris slayti !!! bu spec hakkında genel bilgiler



Released under a Creative Commons license. ([CC BY-SA 4.0](#)).



© 2011 Blackwell Publishing Ltd *Journal of Internal Medicine* 270: 321–329



!!! not: query ye detayli degineceyiz

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.

The diagram illustrates the structure of the **Method-Specific Identifier**. It is composed of three parts: **Scheme**, **Method**, and **Method-Specific Identifier** (which is the full name of the identifier). The **Scheme** and **Method** parts are separated by a colon, and the **Method-Specific Identifier** part is separated by a hyphen.

# DID Subjects

## Architecture Overview

!!! 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.

## Architecture Overview

!!! 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

## Architecture Overview

!!! klsalt

!!! gorsel ekle ?

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



# DID Methods

## Architecture Overview

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 Overview

!!! 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/](https://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 giris slayti !!! ceviri

did ve did urllerinin syntaxini inceleyecegiz, generic terimi burda tanımlanan syntaxin diger did methodlarında tanımlanabilecek syntaxlardan ayırtılmalı edilmek amacıyla kullanıldı

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.

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

did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a

DID

DID URL

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





## 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 **REQUIRED**

# 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

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

### hl parameter

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

A DID path is identical to a generic URI path  
did:example:123456/path

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 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

!!! 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: çok detaylı, her detaya gerek yok

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

!!! not: çok detaylı, her detaya gerek olmayabilir !!! not: notu not al,

- ▶ 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

!!! not: burada birçok issue var onları not al

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

# Verification Relationships

!!! not: detaylarini not al

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

!!! not: note u al, alt basliklarin detaylarini not al

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

!!! not: detaylari not al

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

!!! kısa gec

!!! yeniden duzenle

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

!!! not: Because there is no central authority for allocating or approving DID method names, there is no way to know for certain if a specific DID method name is unique

!!! not: The authors of a new DID method specification **SHOULD** use a method name that is unique among all DID method names known to them at the time of publication.

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.

# Method Operations

!!! not: not al

## 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.



# Resolution

!!! giris !!! not: kesin implementasyon did core specinin disnda bundan dolayi cok detaya girmeyecegim !!! gorsel ekle

# DID Resolution

!!! not: detaylari not al 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

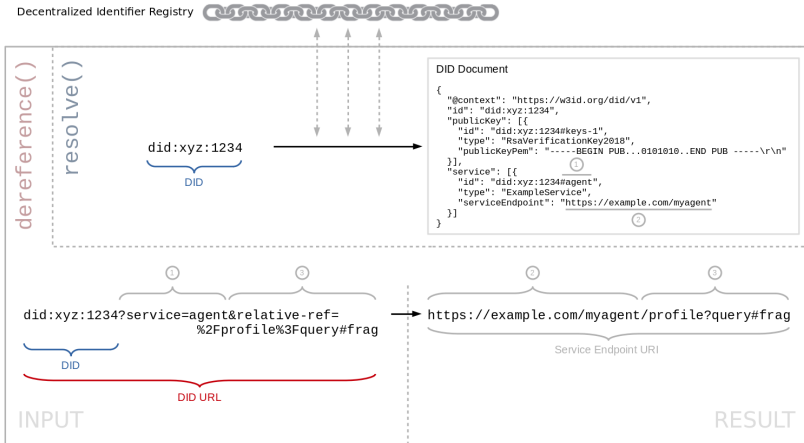
!!! not: detaylari not al

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



# Software / Repos

!!! repolari ekle

- ▶ dif/did-common-java
- ▶ w3c/did-use-cases
- ▶ peacekeeper/blockchain-identity
- ▶ peacekeeper/did-imp-guide **initial phase**



# DID Auth Development

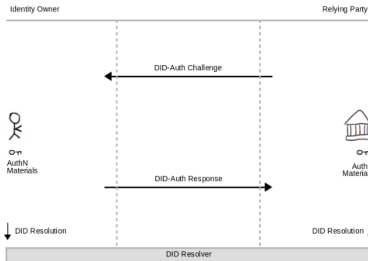
- ▶ DID TLS (Agust 2017) *unresolved*
  - ▶ agentlar arasi guvenli kanal olusturma
  - ▶ tls in cok riskli olduguna karar verildi
- ▶ RWOT 6 DID Auth (March 2018)
  - ▶ Introduction to DID Auth

# DID Auth Development

- ▶ did-auth-jose (October 2018, ietf, dif) *archived sept 22*
  - ▶ Javascript Object Signing and Encryption
  - ▶ jwt, jws, jwe
- ▶ DID SIOP (Sep 12, 2019, dif) *Current focus*
- ▶ DID Authentication WG (Jan 14 2020)

**Core idea: providing control of a did**

## DID Auth: High-Level Overview



## Challenge example, uPort

```
{ "typ": "JWT", "alg": "ES256K" }
{
  "iss": "2oeXufHGDpU51bfKBSZDdu7Je9weJ3r7sVG",
  "iat": 1525865398,
  "requested": [
    "name", // ...
  ],
  "permissions": [ "notifications" ],
  "callback": "https://.../api/v1/topic/Go...Bft7PZ9",
  "exp": 1525865998,
}
```

# DID Authn Challenge

Challenge response example

```
{  
  "header": {  
    "typ": "JWT",  
    "alg": "ES256"  
  },  
  "payload": {  
    "iss": "did:example:123456789abcdefg",  
    "sub": "did:example:123456789abcdefg",  
    "iat": 1479850830,  
    "exp": 1511305200,  
  },  
  "signature": "..."  
}
```

## DID TLS (Agust 2017, Hyperledger) *unresolved*

!!! not: detaylari not al - Using chain anchored keys to facilitate mutual authentication via TLS. - Extends the TLS SNI Specification and provides additional methods for certificate validation that does not rely on established Root Certificate Authorities.

[illegible]

100

- ▶ Setting custom SNI hint from client
- ▶ Inability to disable client certificate validation
- ▶ Lack of certificate validation callbacks

## Abstract

- ▶ OpenSSL: Allow non-validated client certificates. (removes signer tls sni hint requirement.)
- ▶ OpenSSL: Add support for newer elliptic curve certificates.
- ▶ SNI Spec: Updated to use different server\_type than HostName
- ▶ HTTP Libraries: Easier methods to specify SNI hint.

## DID TLS (Agust 2017, Hyperledger) *unresolved*

!!! not: suanki auth calismalarinin yonunden bahset

- ▶ Current Agent2Agent communication in Aries Cloud Agent is Inbound and Outbound TCP ports with *DIDcomm*
- ▶ DIF Auth WG Open is currently focused on developing DID OpenID Connect Provider (did-siop)



# RWOT 2018 DID Auth (March 2018)

## did authn and verifiable credentials

!!! yerini ayarla

!!! not: detaylari not al, did auth paperinda

!!! did auth verifiable credentials nuanslarina degin

- ▶ DID Auth and Verifiable Credentials exchange are separate.
- ▶ Verifiable Credentials exchange is an extension to (or part of) DID Auth.
- ▶ DID Auth is a certain kind of Verifiable Credential. bunu tamamla

# RWOT 2018 DID Auth (March 2018)

# RWOT 2018 DID Auth (March 2018)

# RWOT 2018 DID Auth (March 2018)

# RWOT 2018 DID Auth (March 2018)

# indy\_auth

!!! not: stajdayken yaptigimiz proje

Internship Project, Application of DID Auth scenarios with DIDs using  
did:sov / Hyperledger Indy.

Gitlab Repo

[bag.org.tr/proje/abdulhamit.kumru/indy\\_tls](https://bag.org.tr/proje/abdulhamit.kumru/indy_tls)

# Presentations

[https://bag.org.tr/proje/abdulhamit.kumru/indy\\_tls/tree/master/presentations](https://bag.org.tr/proje/abdulhamit.kumru/indy_tls/tree/master/presentations)

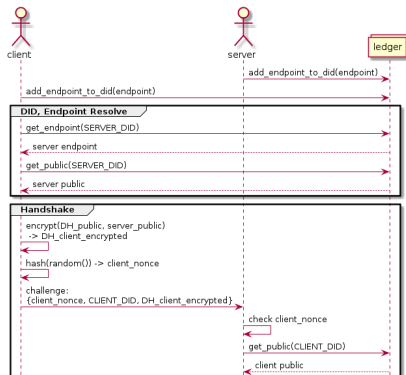
# indy\_auth

## sub projcets

- ▶ http\_signatures
- ▶ auth\_encryption
- ▶ did\_tls



did\_tls  
indy\_auth



# did\_tls

## indy\_auth



# http\_signatures

framesubtitle{indy\_auth} !!! kısaca

# auth encryption

framesubtitle{indy\_auth} !!! kısaca

did-auth-jose (October 2018, ietf, dif) *archived sept 22*

did-auth-jose (October 2018, ietf, dif) *archived sept 22*

DID SIOP *identity.foundation/did-siop*

!!! detaylara sonra girecegiz

# DIF Authentication Working Group

!!! burada bahsetmek uygun olmaya bilir

## DID Authentication Profile for SIOP

!!! not: browser extentiondan bahset

This specification defines the SIOP DID AuthN flavor to use OpenID Connect (OIDC) together with the strong decentralization, privacy and security guarantees of DID for everyone who wants to have a generic way to integrate SSI wallets into their web applications.

- ▶ Status: DRAFT, WIP
- ▶ Use Case: Use your identity wallet to authenticate against a Web Application



# DIF Authentication Working Group

## Encrypted Envelope

This concept is borrowed from the HL Aries project to create a standardized means of authenticated general message passing between DID controllers. DIF provides an implementation of pack/unpack that intends to meet the requirements of the DIF community.

- ▶ Status: PROPOSAL
- ▶ Use Case: Secure communication between DID controllers.

## decentralized-identity/DIDComm-js

!!! pack unpack details

JS implementation of pack and unpack

# Auth Protocols & DID Auth

!!! giris slayti

# Auth Protocols & DID Auth

- ▶ SAML
- ▶ CAS
- ▶ OpenID Connect
- ▶ Self-Issued OpenID Connect Provider DID Profile (did-siop, DIF)
- ▶ DID SAML ?
- ▶ DID CAS ?
- ▶ DID Authentication on PAM

# SSO

# Understanding main SSO protocols: CAS, SAML and OpenID Connect

- ▶ CAS: <https://apereo.github.io/cas>
- ▶ SAML: <https://www.oasis-open.org/standards>
- ▶ OpenID Connect : <http://openid.net/developers/specs>

# SAML

!!! giris slayti

# CAS

!!! giris slayti

# OpenID Connect

!!! giris slayti



# Protocol Comparison

- ▶ CAS: simple protocol, no strong security, fits internal usage
- ▶ SAML: complex protocol, very used for SaaS authentication, good security, well established
- ▶ OpenID Connect: easy adoption with new technologies (JSON/REST/OAuth2), mobile ready, good security, still not wide spread

# DID PAM

!!! giris slayti

did authn Pluggable Authentication Modullerde kullanilabilir mi ?

# Pluggable Authentication Module (PAM)

!!! giris slayti

# Repositories

- ▶ [decentralized-identity/did-auth-jose](#)
- ▶ [decentralized-identity/did-common-java](#)

# Sources

[https://github.com/mrkaurelius/did\\_and\\_friends/blob/master/README.md](https://github.com/mrkaurelius/did_and_friends/blob/master/README.md)