DID-based Auth Protocols

Vurucu ve Akillica Alt Başlık

Abdulhamit Kumru

Blokzincir Laboratuvarı

2020

DID Auth Development

- DID TLS (Agust 2017) *unresolved*
  - TLS mutual Auth with DIDs
- RWOT 6 DID Auth (March 2018)
  - Introduction to DID Auth

DID Auth Development

- DID Authentication WG (Jan 14 2020)
  - did-auth-jose (October 2018, DIF) *archived sept 22*
  - DID SIOP (Sep 12, 2019, DIF) *Current focus*
    - successor of did-auth-jose

DID TLS (Agust 2017, Hyperledger) *unresolved*

!!! not: detaylari not al
DID TLS feature proposed as indy SDK feature.

- 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.

Links

- github.com/TelegramSam/DID-TLS
- jira.hyperledger.org/browse/IS-268

DID TLS (Agust 2017, Hyperledger) *unresolved*

Issues

!!! not: dokumandan detaylari not al

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

Future Work

- 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)

Rebooting Web-of-Trust (RWoT) 6 DID Auth

Abstract

The term DID Auth has been used in different ways and is currently not well-defined.
We define DID Auth as a ceremony where an identity owner, with the help of various components such as web browsers, mobile devices, and other agents, proves to a relying party that they are in control of a DID
Core idea: proving control of a DID to relying party

Owner vs. Controller

RWoT 6 DID Auth

This paper heavily uses the term identity owner. This helps to emphasize clearly how DID Auth is fundamentally different from earlier authentication protocols, which have traditionally revolved around "identity providers".

DID Authn and Verifiable Credentials

RWoT 6 DID Auth

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

- 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

Authentication of a DID

RWoT 6 DID Auth

Authentication of a DID

Similar to other authentication methods, DID Auth relies on a challenge-response cycle in which a relying party authenticates the DID of an identity owner.

Challenge

RWoT 6 DID Auth

The way an identity owner or their agent encounters an authentication challenge, as well as the format of the challenge, will vary depending on the situation. For example, they can come across a “Sign in with DID Auth” button or a QR code on a website

Response

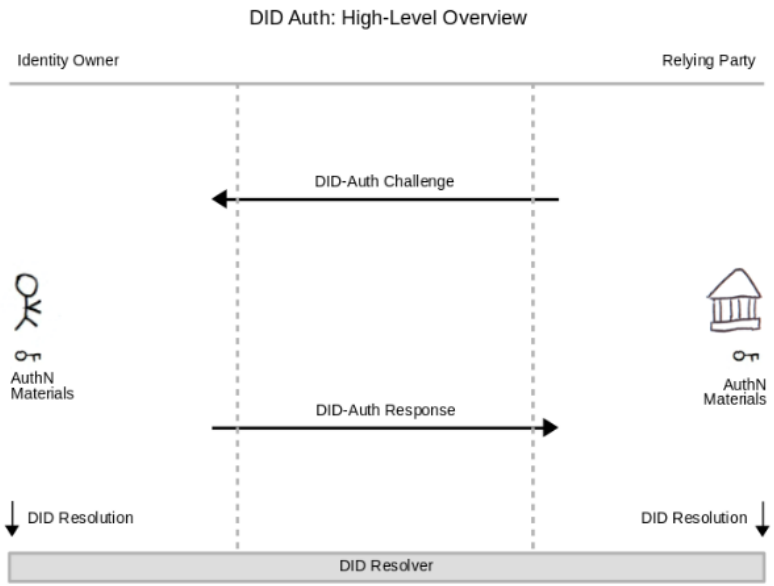
RWoT 6 DID Auth

Based on the challenge, the identity owner then constructs a response that proves control of their DID. This often involves a cryptographic signature, but can include other proof mechanisms. (As mentioned earlier, the response may also contain Verifiable Credentials that the relying party asked for in the challenge.)

Generic DID Auth Architecture

!!! not: bu auth arch bircok farkli yerde uygulanabilir, ornek olarak . . .

Introduction to DID Auth



DANUBE TECH GMBH

SSIMEETUP Self-Sovereign Identity SSIMeetup.org

Javascript Object Signing and Encryption (JOSE)

!!! not: jose den kisaca bahsetmek lazim diduth ve sonraki yerlerde siklikla kullaniliyor, bahsedecegiz.

- Used in DID Auth and DID SIOP
- Adapted in many similar area

JSON Web Algorithms (JWA)

JWA specification registers cryptographic algorithms and identifiers to be used with the JSON Web Signature (JWS), JSON Web Encryption (JWE), and JSON Web Key (JWK) specifications

JSON Web Signature (JWS)

JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures

Javascript Object Signing and Encryption (JOSE)

JSON Web Encryption (JWE)

JSON Web Encryption (JWE) represents encrypted content using JSON-based data structures

JSON Web Key (JWK)

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key

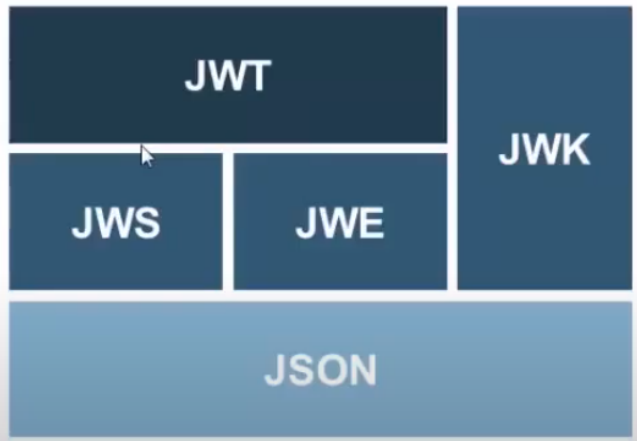
JSON Web Token (JWT)

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties.

Javascript Object Signing and Encryption (JOSE)

JWT + JOSE Overview

- JavaScript Object Signing and Encryption (JOSE)
  - **J**SON **W**eb **S**ignature (JWS)
    - A way of representing content secured with a digital signature or MAC using JSON data structures and base64url encoding
  - **J**SON **W**eb **E**ncryption (JWE)
    - Like JWS but for encrypting content
  - JSON Web Key (JWK)
    - JSON data structures representing cryptographic keys
  - JSON Web Algorithms
    - Defines the use cryptographic algorithms and identifiers for JWS, JWE and JWK
- JSON Web Token (JWT)
  - A compact URL-safe means of representing claims/attributes to be transferred between two parties
  - A JWT is a JWS and/or a JWE with JSON claims as the



DID Authn Challenge

RWoT 6 DID Auth

JWT challenge example, uPort

```
{ "typ": "JWT", "alg": "ES256K" }
{
  "iss": "2oeXufHGDpU51bfKBsZDdu7Je9weJ3r7sVG",
  "iat": 1525865398,
  "requested": [
    "name", // ...
  ],
  "permissions": [ "notifications" ],
  "callback": "https://.../api/v1/topic/Go...Bft7PZ9",
  "exp": 1525865998,
}
```

Challenge Transports

RWoT 6 DID Auth

!!! not: detaylari not al

A DID Auth challenge may be delivered by a relying party to an identity owner in different ways. DID Auth defines a few common ways that this can be done.

- DID Auth Service Endpoint
- Custom Protocol Handler
- Mobile Deep Link
- Custom Protocol Handler
- Invoke User Agent's JavaScript API
- Form Redirect
- Device-to-device Communication

DID Authn Challenge

RWoT 6 DID Auth

!!! not: challange illa jwt olmak zorunda degil, baska yontemler de olabilir (json ld vc)

JWT challenge resoponse example

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

Response Transports

RWoT 6 DID Auth

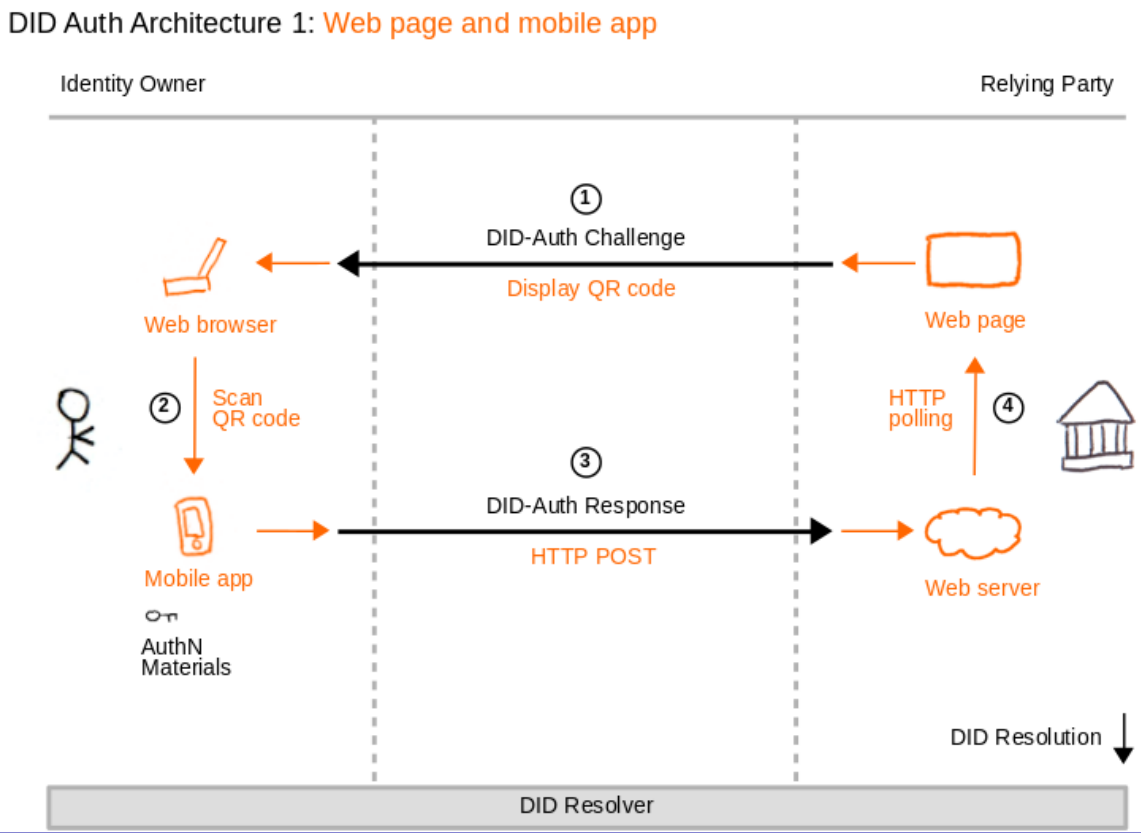
!!! not: detaylari not al

- HTTP POST to Callback URL
- Scan QR Code from Mobile App
- Fulfill JavaScript Promise
- Device-to-device Communication

Auth Architecture Web page and mobile app

RWoT 6 DID Auth

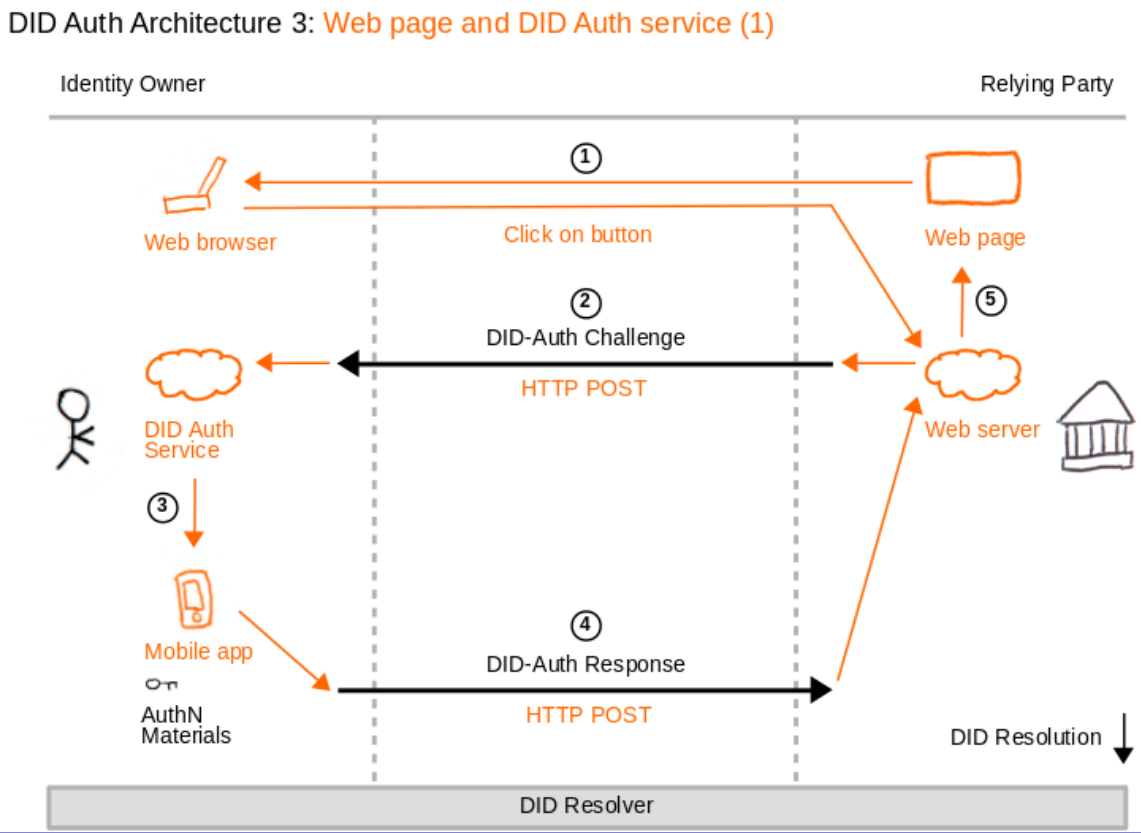
!!! Auth architecture gorselleri ekle !!! not: detaylari not al



Auth Architecture Web page and DID Auth service

RWoT 6 DID Auth

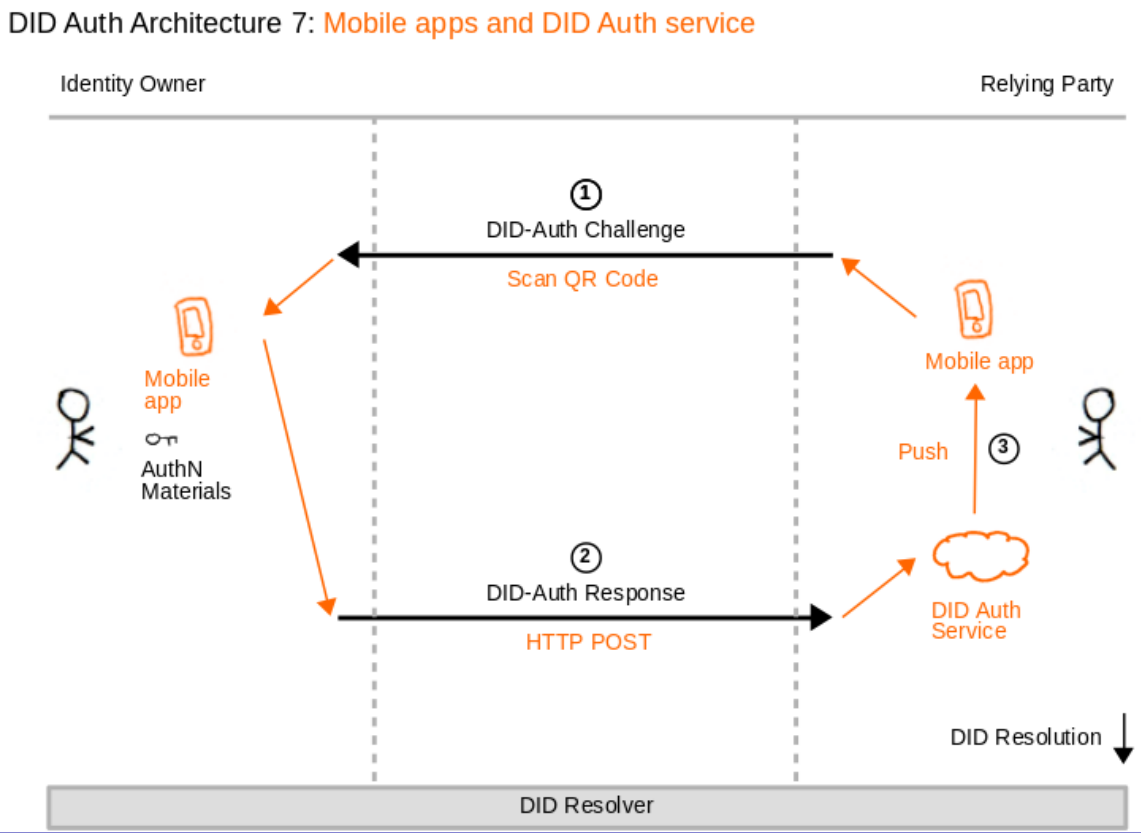
!!! Auth architecture gorselleri ekle !!! not: detaylari not al



Auth Architecture Mobile apps and DID Auth services

RWoT 6 DID Auth

!!! Auth architecture gorselleri ekle !!! not: detaylari not al



Relation to Other Technologies

RWoT 6 DID Auth

!!! not: detaylara notta degin, diger teknolijer ile baglantisi, uygulanabilirligi

- Other Public Key Infrastructure (PGP, SSH, etc.)
- WebAuthn (FIDO Authentication)

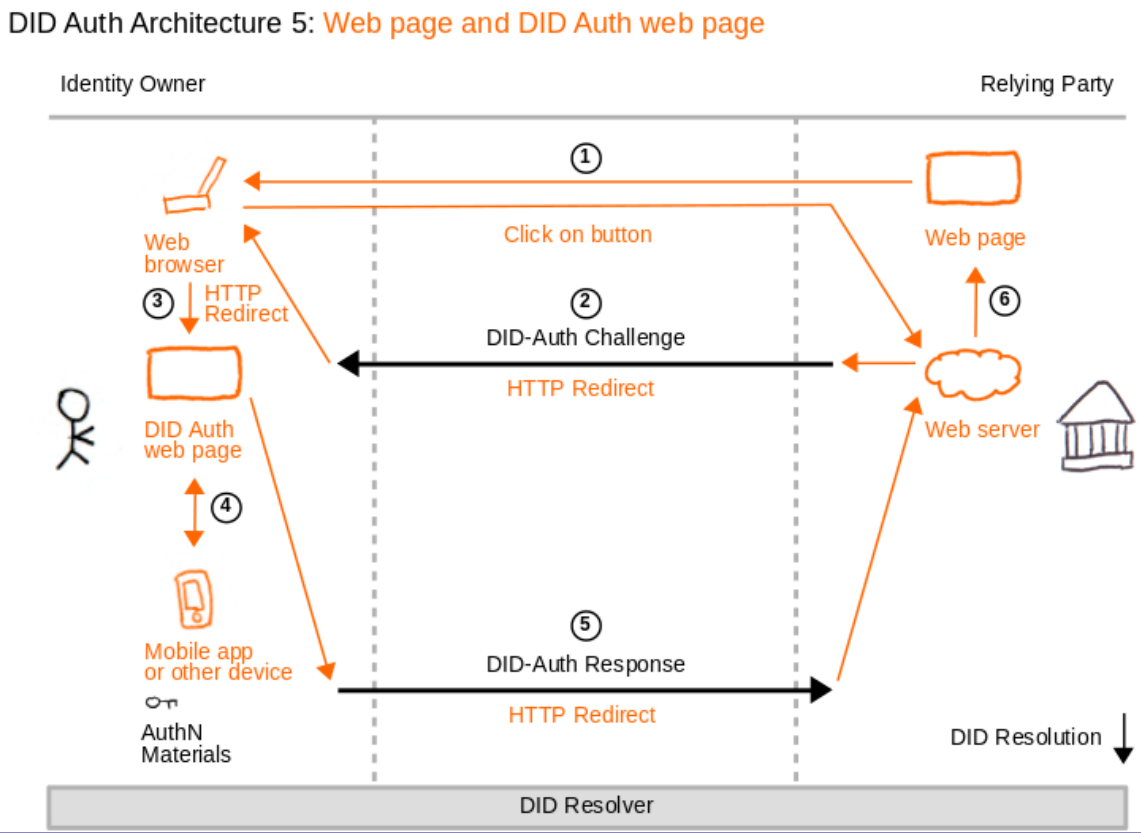
OpenID Connect *(DIF Auth Current Focus)*

!!! not: burayi iyi anla not al
OpenID Connect (OIDC) is an authentication protocol built on the OAuth 2.0 protocol. In its most common web-based form, an end-user's user agent is redirected by a relying party (OAuth 2.0 client) to an OpenID Provider (OAuth 2.0 authorization server), which authenticates the end-user and redirects them back to the relying party.

Relation to Other Technologies

RWoT 6 DID Auth

!!! not: aciklamayi not al



Next Steps

RWoT 6 DID Auth

- Agree on scope of DID Auth.
- Agree on supported formats for challenges and responses.
- Agree on supported protocols and flows.

indy_auth

!!! not: stajdayken yaptigimiz proje
!!! basettigimiz kavramlari teknolojileri anlamak icin gelistirdigimiz projemiz.

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

Gitlab Repo

bag.org.tr/proje/abdulhamit.kumru/indy_tls

Presentations

https://bag.org.tr/proje/abdulhamit.kumru/indy_tls/tree/master/presentations

indy_auth

indy_auth

!!! not: icerdigi alt projeler, proof of concept nitelidinde calismalar.

!!! indy kullanarak did auth calismalari yapildi

did_tls

- Generic DID Auth over TCP using generic JSON Object
- DH Session key generated
- TLS Socket created.

Tools

- Python ssl
- indy-sdk

Other Works

- http signatures
- auth encryption

indy-sdk

indy_auth

!!! not: not ekle

Everything needed to build applications that interact with an Indy distributed identity ledger.

libindy

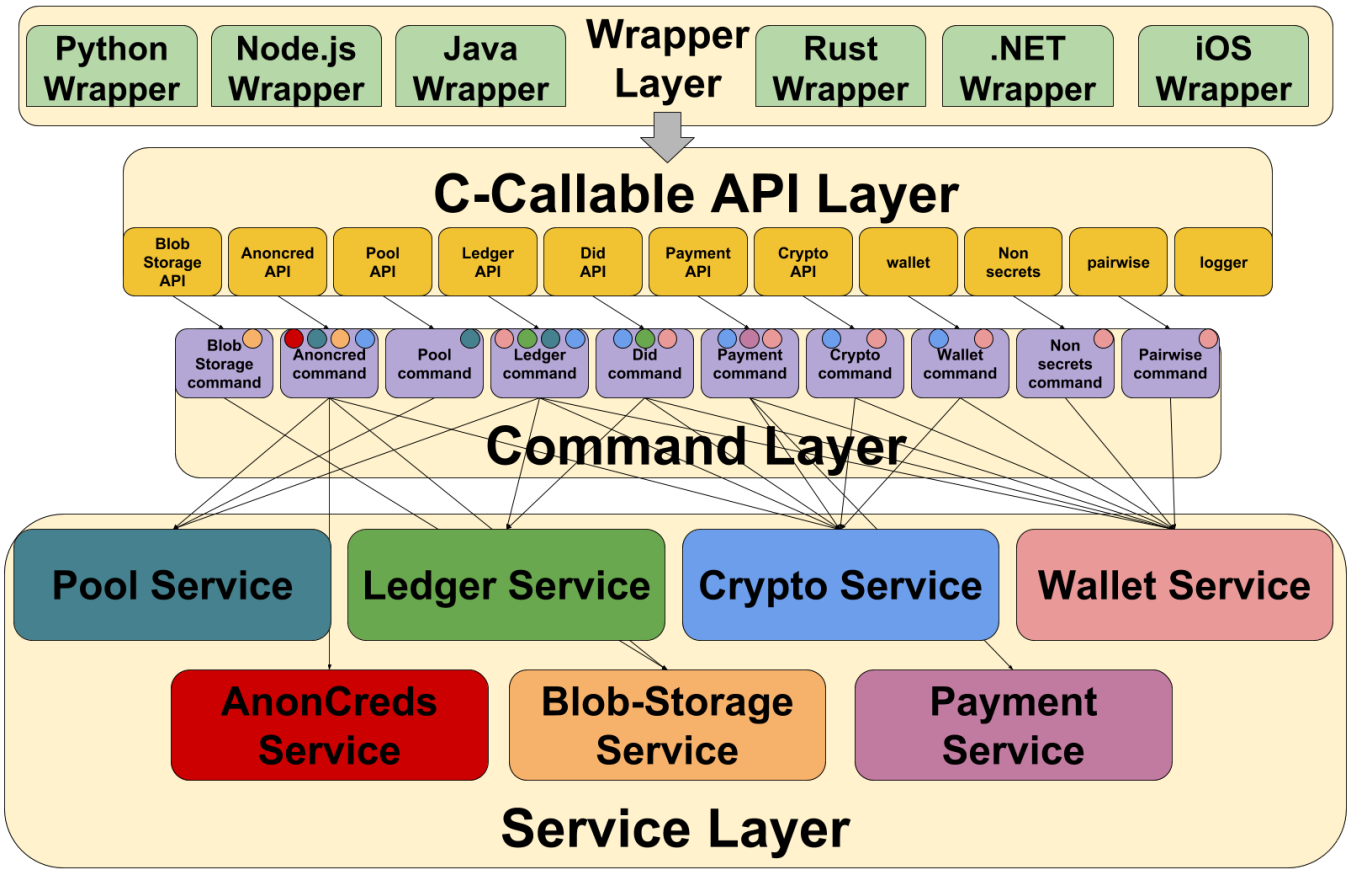
The major artifact of the SDK is a C-callable library that provides the basic building blocks for the creation of applications on the top of Hyperledger Indy

Libindy wrappers

A set of libindy wrappers for developing Indy-based applications in your favorite programming language

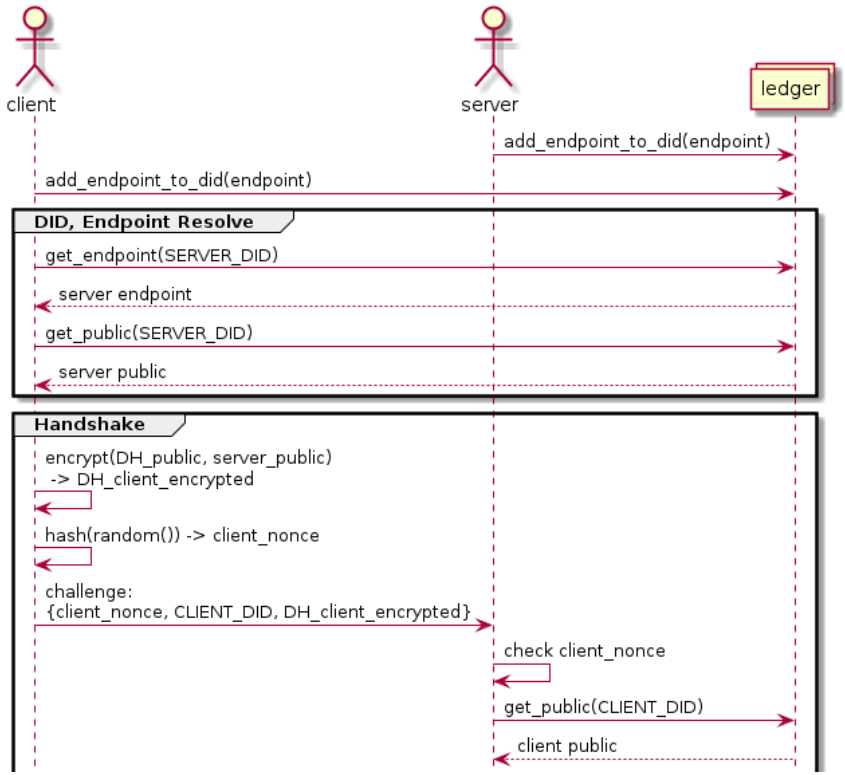
indy-sdk overview

indy_auth



did_tls

indy_auth



did_tls

indy_auth

Decentralized Identity Foundation (DIF)

Provide a neutral and inclusive place for the decentralized identity community to collaborate

Working Groups

- Authentication
- Claims and Credentials
- DID Communication
- ...

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

did-auth-jose is a library that provides JOSE (Javascript Object Signing and Encryption) encryption, decryption, signing, and verifying capabilities through a key and algorithm extensible model, as well as two authentication flows for use with decentralized identities (DIDs).

evolved to did-siop

Authentication Flow

did-auth-jose

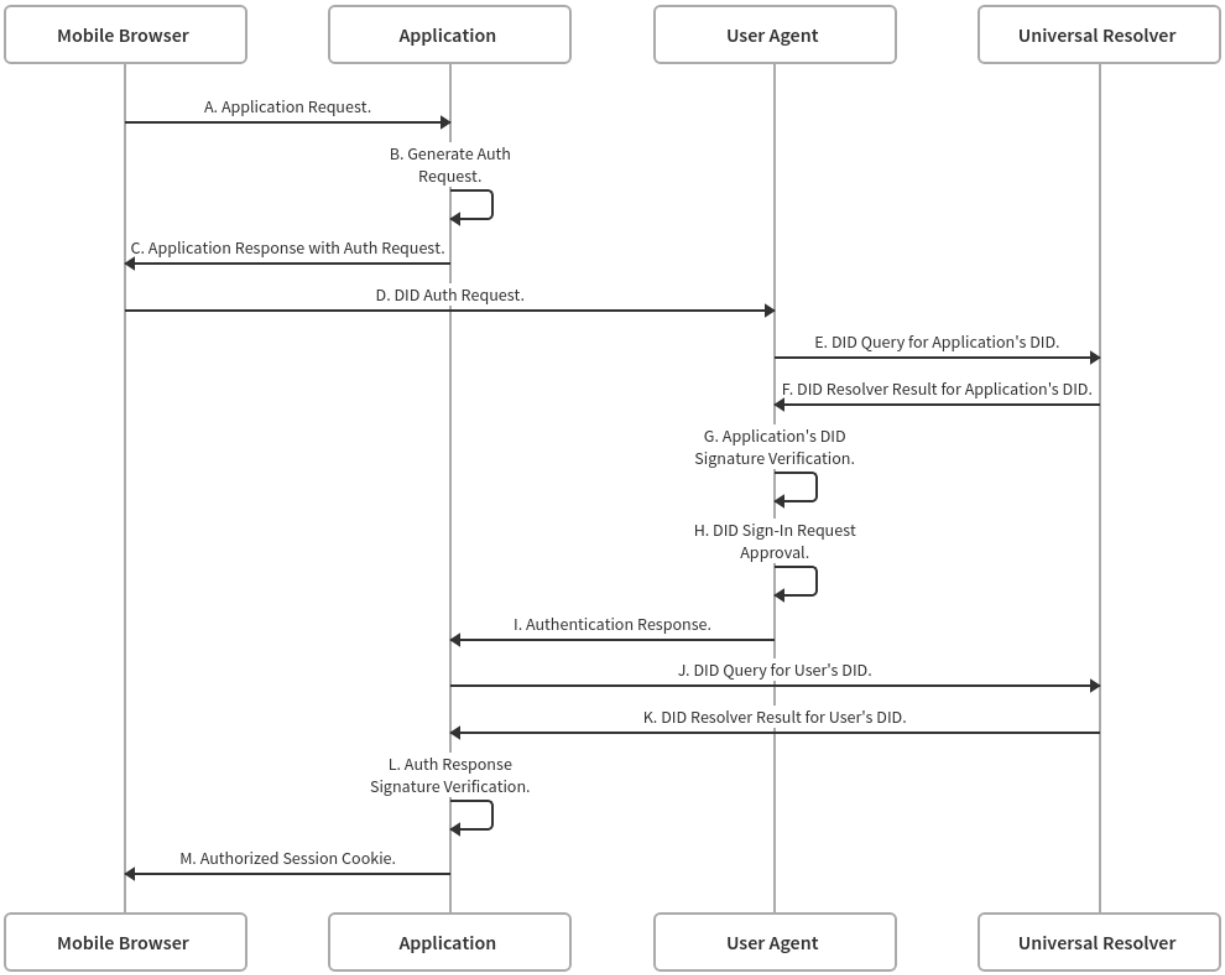
DID Authentication uses two to three JSON Web Tokens (JWT) per request.

The first is an outer JSON Web Encryption (JWE), and the second is an inner JSON Web Signature (JWS), both utilizing the public private key pair of each DID retrieved from their DID Document.

An optional third JWT access token may be included in a JWS header. This format ensures the content is encrypted end to end and independently verifiable. Each JWS contains a nonce header to associate requests with responses

Authentication Flow

did-auth-jose



DIF Authentication Working Group (Jan 14 2020)

DID Authentication Profile for SIOP

!!! not: browser extentiondan bahset
!!! not: not al !!! detaylara bir sonraki bolumde girecegiz
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

SIOP DID is an unapproved DIF working group draft specification being developed within the Decentralized Identity Foundation (DIF).

DIF Authentication Working Group (Jan 14 2020)

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

Javascript version of the cryptographic envelope of DIDComm (pack, unpack).