24. OAuth 2.0: Explained with API Request and Response Sample | High Level System...

CRYPTOGRAPHY
SYMMETRIC → With AES explanation
V/s
ASYMMETRIC → With Diffie-Hellman & Digital Signature example
51:03

25. Symmetric & Asymmetric Encryption with Explanation of AES, Diffie-Hellman an...

What is JWT (JSON Web Token)

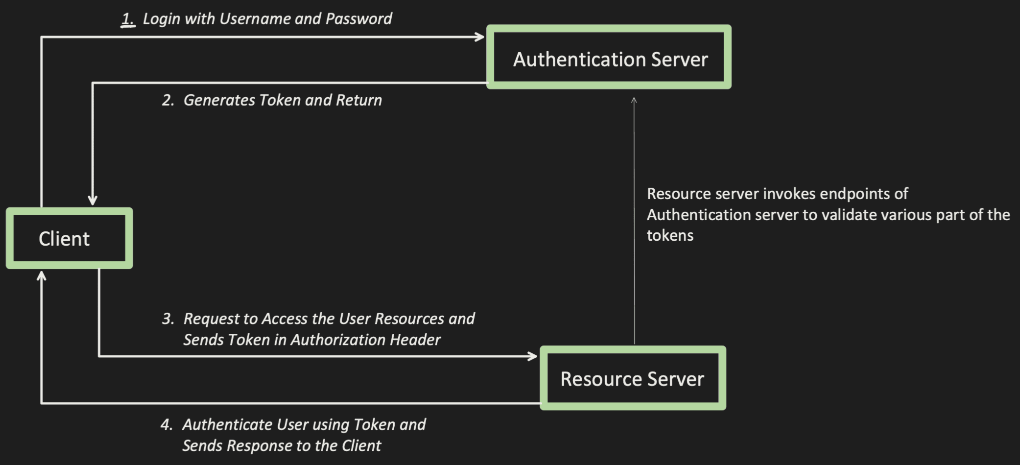
- It's provides a secure way of transmitting information between parties as a JSON object.
- This information can be verified because its digitally signed using RSA (public/private key pair) etc.

Advantages:

- **Compact**: Because of its size, it can be sent inside an HTTP header itself. And, due to its size its transmission is fast.
- **Self Contained / Stateless**: The payload contains all the required information about the user, thus it avoid querying the database.
- Can be signed using Symmetric (HMAC) or Asymmetric (RSA).
- Built in expiry mechanism.
- Custom claim (additional data) can be added in the JWT.

Where to use JWT:

- Used for AUTHENTICATING (confirming the user identity)
- Used for AUTHORIZATION (checks the user permission before providing access to resources)
- Used for SSO (Single Sign On) i.e. Authenticate once and access multiple applications.

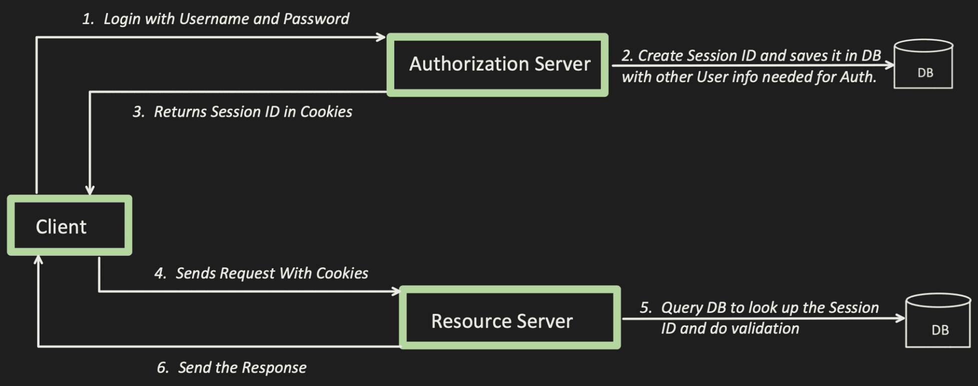


1. *Login with Username and Password*
2. *Generates Token and Return*
3. *Request to Access the User Resources and Sends Token in Authorization Header*
4. *Authenticate User using Token and Sends Response to the Client*

Resource server invokes endpoints of Authentication server to validate various part of the tokens

Before we understand more about JWT, lets first understand, what was popular before JWT and what are the problems with it?

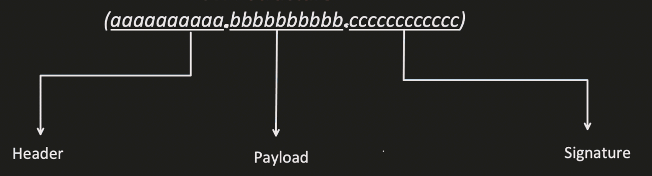
Session ID (or JSessionID) :


1. Login with Username and Password
Authorization Server
2. Create Session ID and saves it in DB with other User info needed for Auth.
DB
3. Returns Session ID in Cookies
Client
4. Sends Request With Cookies
Resource Server
5. Query DB to look up the Session ID and do validation
DB
6. Send the Response

Disadvantage:

- Stateful: it rely on server side state management, it cause problem in distributed systems.
- Its just a unique random string, when server get this id, it has to perform DB query to fetch the details.

JWT Structure



Header

```
{
   "typ": "JWT",
   "alg": "RSA",
   "kid": "sfdsf234234"
}
```

Payload

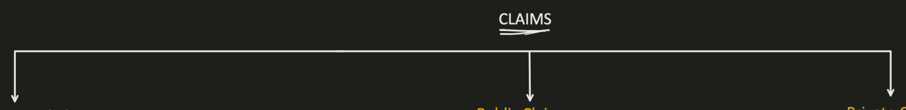
```
{
   "iss": "https://example.com/auth",
   "sub": "1234567890",
   "exp": 1711945909,
   "email": "conceptcoding@xyz.com,
   "jti": "unique_id_12312312"
}
```

Signature

```
RSASHA256(
Base64UrlEncode(Header) + "."  + Base64UrlEncode(Payload) ,
Key
)
```

Header:
- Contains metadata information of the token.
- typ: Type of the token, generally JWT always we add here.
- alg: Signing algorithm used like RSA or HMAC etc..

Payload:
- Contains Claims (or in simple terms, User information or any additional information is kept here)

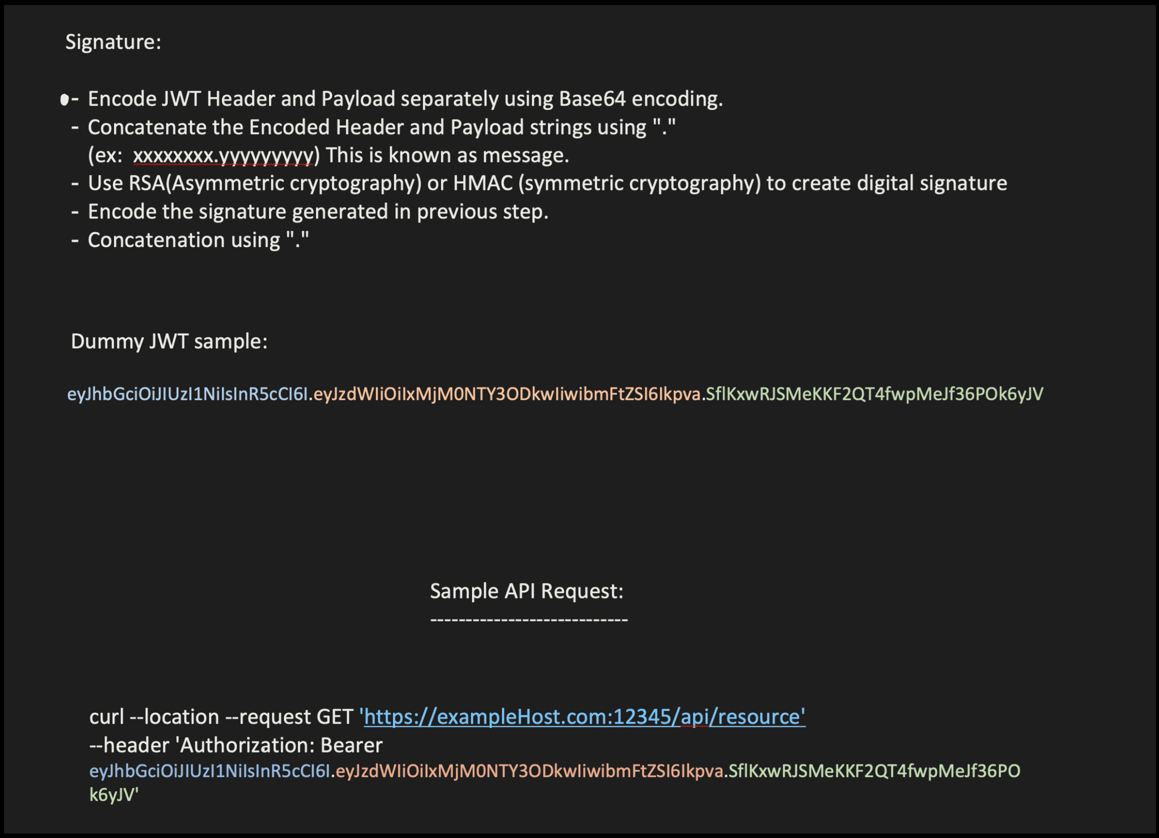


Registered Claims
(predefined names and meaning)

- iss (Issuer): entity that issued the JWT.
- sub (Subject) : identifies the user.
- aud (Audience): identifies the recipient for which token is intended.
- exp (Expiration Time) : After this time, token is not valid.
- nbf (Not Before): Time before this, Token should not be accepted.
- iat (Issued At): time at which token is issued.
- jti (JWT Id): Unique JWT ID.

Public Claims
(It's a custom claim, which can be shared and understood by multiple parties)

Private Claims
(It's a custom claim, which are indented for internal use only and not standardized, nor expected that other parties understand this)



Signature:

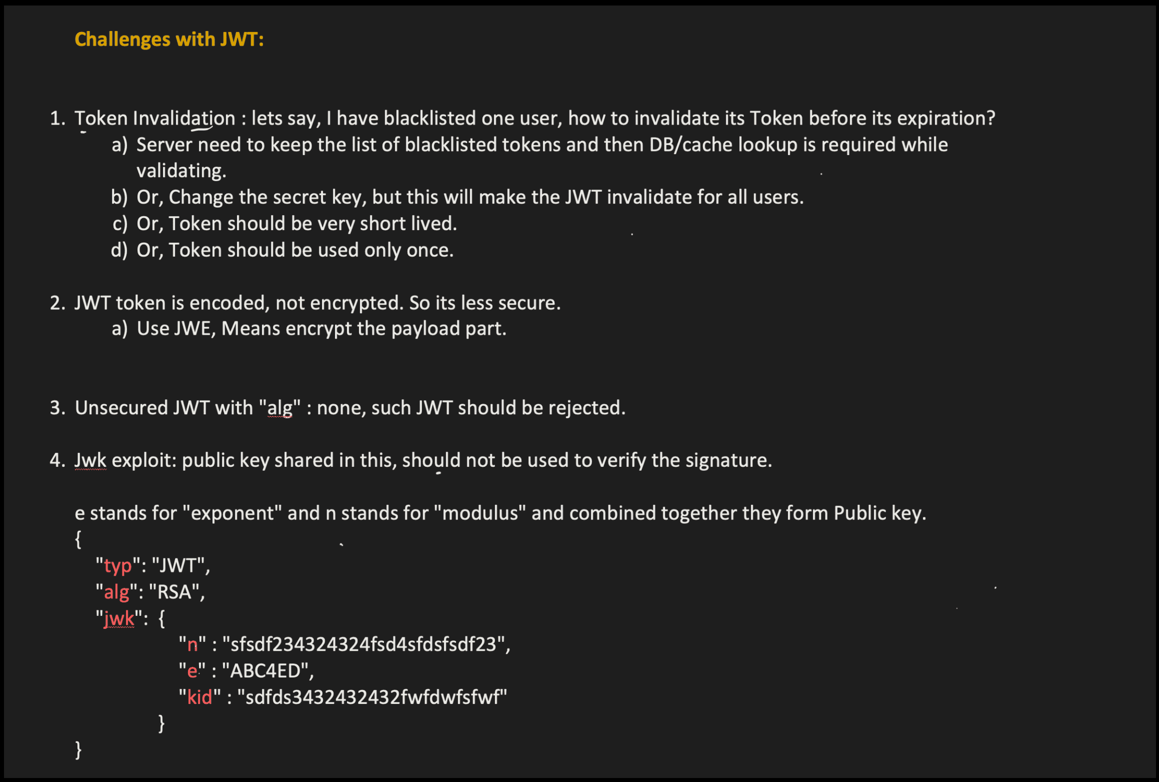
- Encode JWT Header and Payload separately using Base64 encoding.
- Concatenate the Encoded Header and Payload strings using "."
  (ex: xxxxxxxx.yyyyyyyyy) This is known as message.
- Use RSA(Asymmetric cryptography) or HMAC (symmetric cryptography) to create digital signature
- Encode the signature generated in previous step.
- Concatenation using "."

Dummy JWT sample:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6I.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV
```

Sample API Request:

```
curl --location --request GET 'https://exampleHost.com:12345/api/resource'
--header 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6I.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva.SflKxwRJSMeKKF2QT4fwpMeJf36PO
k6yJV'
```



Challenges with JWT:

1. Token Invalidation : lets say, I have blacklisted one user, how to invalidate its Token before its expiration?
   a) Server need to keep the list of blacklisted tokens and then DB/cache lookup is required while validating.
   b) Or, Change the secret key, but this will make the JWT invalidate for all users.
   c) Or, Token should be very short lived.
   d) Or, Token should be used only once.

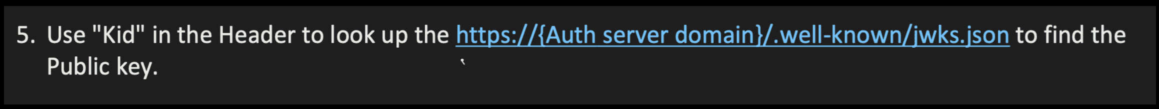
2. JWT token is encoded, not encrypted. So its less secure.
   a) Use JWE, Means encrypt the payload part.

3. Unsecured JWT with "alg" : none, such JWT should be rejected.

4. Jwk exploit: public key shared in this, should not be used to verify the signature.

   e stands for "exponent" and n stands for "modulus" and combined together they form Public key.

```
{
   "typ": "JWT",
   "alg": "RSA",
   "jwk": {
              "n" : "sfsdf234324324fsd4sfdsfsdf23",
              "e" : "ABC4ED",
              "kid" : "sdfds3432432432fwfdwfsfwf"
          }
}
```



5. Use "Kid" in the Header to look up the https://{Auth server domain}/.well-known/jwks.json to find the Public key.