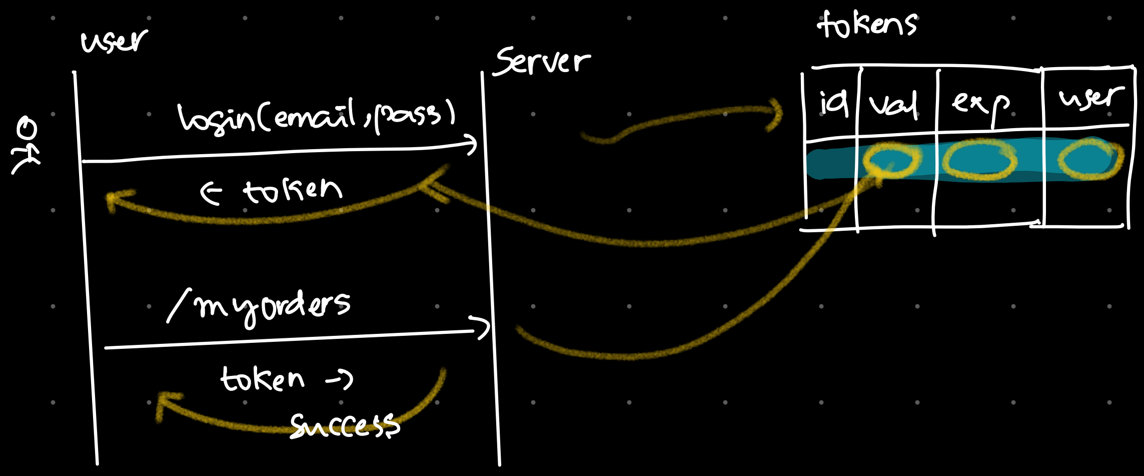


How to avoid the db call?

Sol^n 1: using cache.

create a replica of db (tokens table) in the cache, and everytime we validate the token using this.

pros: still faster than making a db call.

cons:

→ cost to share info in cache

→ still there's some latency

→ cache & db should be in sync.

Sol2: Self Validating Token.

what I need to verify --

1) user Id

2) expiry

3) token

Somehow, we'll need to store these info in the token.

```
{
    "user-id" : ________
    "exp"     : ________
    "token"   : ________
        -=
}
```

The data can be easily tampered, that becomes a security issue.

Tip: Along with the data, we need to make sure data is secured and cannot be modified.

encode

$1 \rightarrow A$

$2 \rightarrow B$

$3 \rightarrow C$

$\vdots$

$$\begin{tikzcd} ABCDE \arrow[r] \arrow[r, bend left] & \boxed{12345} \arrow[l, bend left] \end{tikzcd}$$

Knowing the logic is enough to decode/encode.

encryption

There's a secret key.

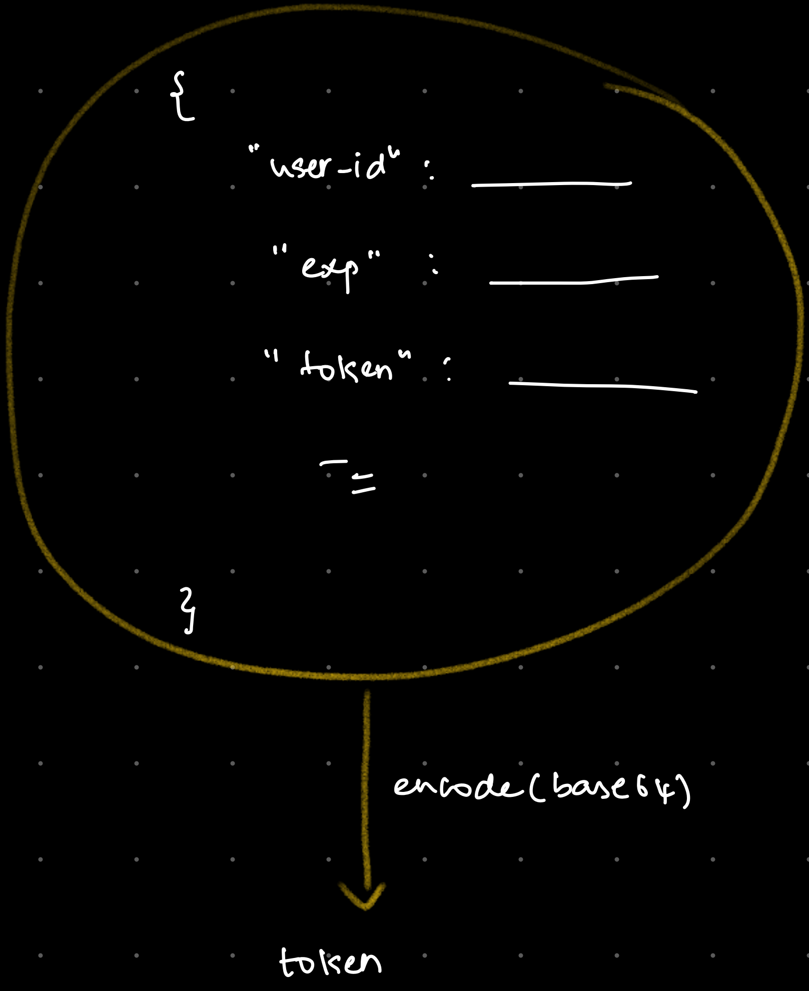
encode (data, secret key)

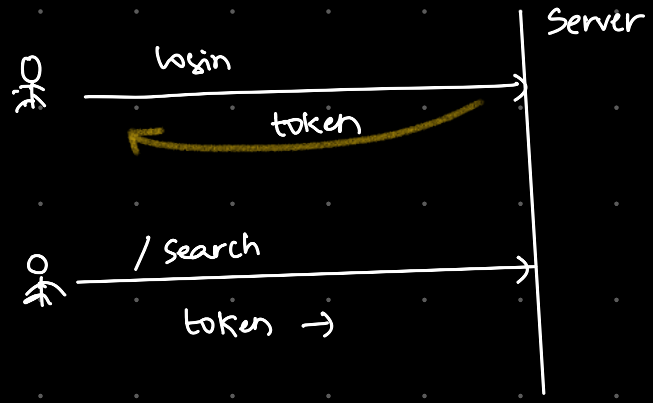
(It's not possible to decode without secret key).

encode + secret key

Encoding

Base 64 Encoding





search(user, token)

```
decodeToken = decode (token)
user_id = decodeToken.user_id
token_val = decodeToken . token
exp = decodeToken . exp
// run some logic & return
success if token is not
expired
```

Encryption

login

- secretKey
- login (email, password)

```
    // validate email & password
    value = base64encode(jsonobject)
    token = encrypt(value, secretKey)
    return token
```

Validation code

```
Validate(user, token)
    value = decrypt(token, secretKey)
    // here, if someone has changed the token, will I be
    // able to decrypt it using same secretKey.
    if decryption fails, immediately logout.
    if decryption is success
        -> user, value.user are same
        -> value.exp is not expired
            --
            -
            -
        return success.
```

JWT → JSON Web Token (Self validating token).

Contains 3 parts. let's say A, B, C

They'll be separated by a '.' operator.

$$\begin{tikzcd}
A . & B . & C \arrow[dr] & \\
\text{header} \arrow[from=1-1, u, phantom] & \text{data} \arrow[from=1-2] & & \text{signature.}
\end{tikzcd}$$

1) Algo we're using to encode

Actual payload

c = encrypt (a+b, secretKey)
if someone changes 'a' or 'b'
we won't be able to decrypt.

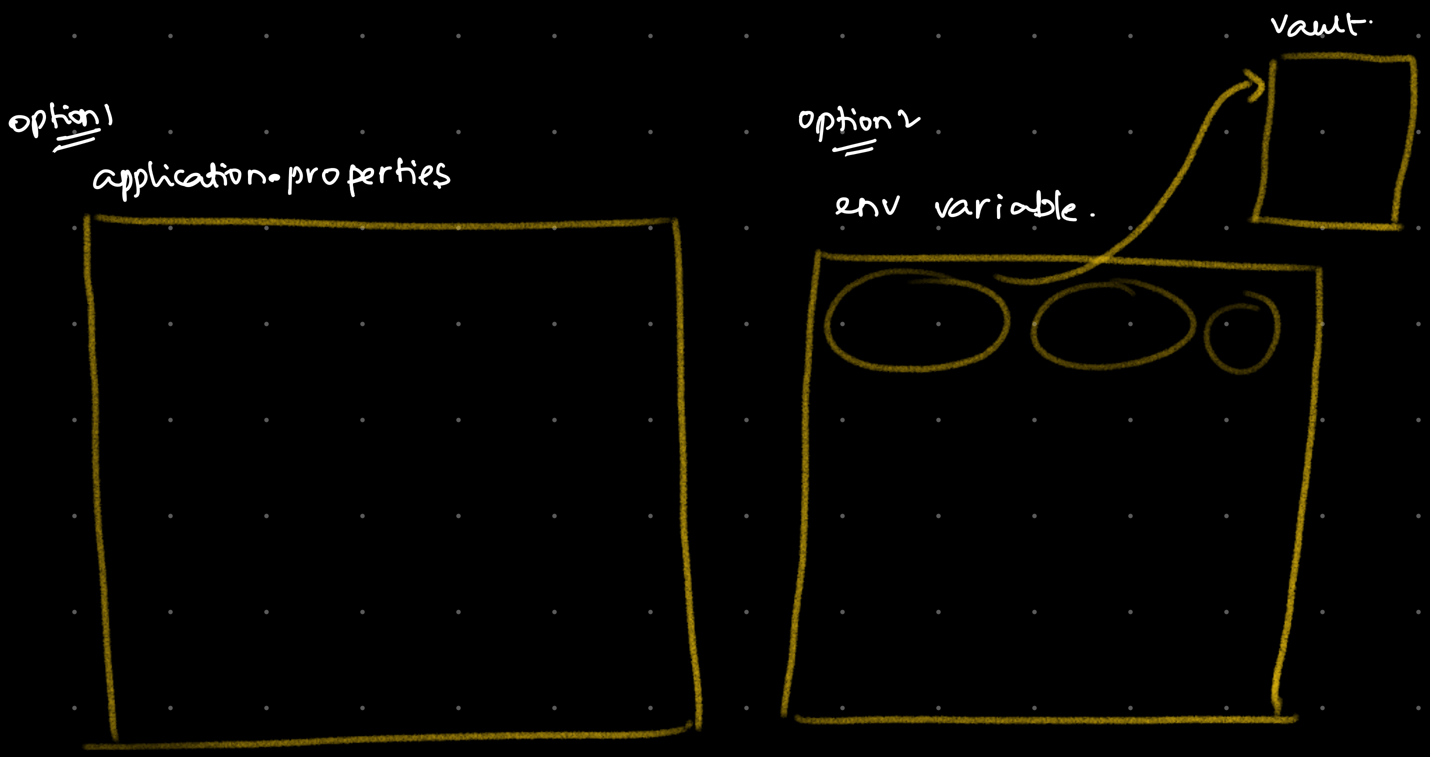
login

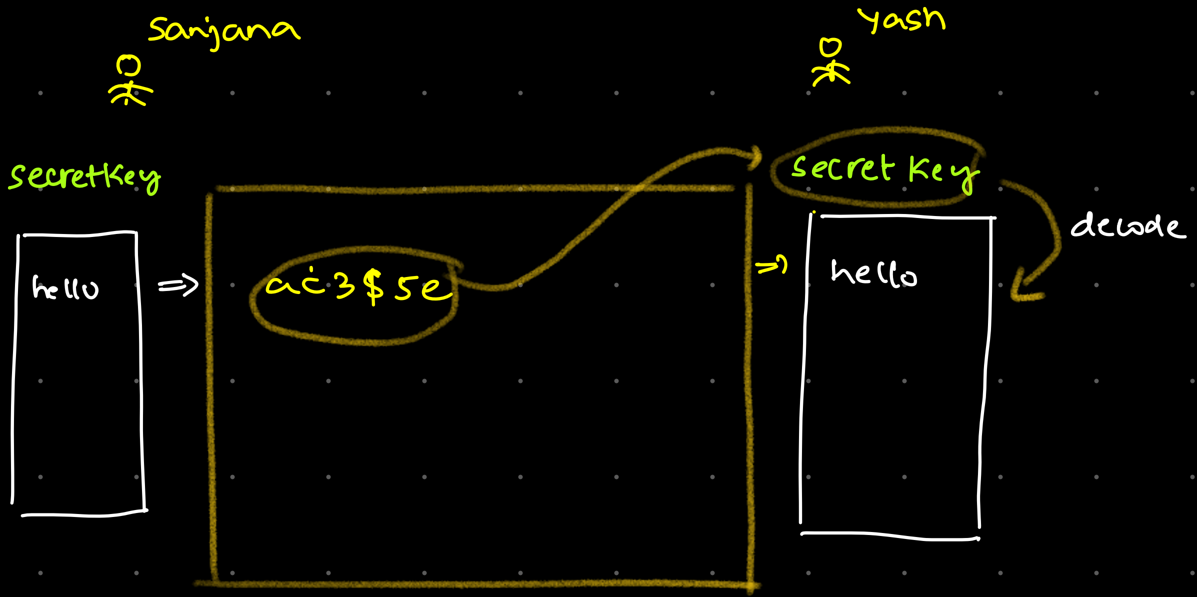
```
-secretKey
 login (email, password)
      // validate email & password combination
      a = base64encode ( { ≡ } )        → Algo
      b = base64encode ( { ≡ } )        → entire data of
                                           token like exp/
                                           userId / tokenValue

      c = encrypt ( a+b, secretKey )
      token = a+"."+b+"."+c
      return token
```

Validate code

```
 Validate Token (token)
      a, b, c = token.split (".")
      d = decrypt (c, secretKey)
      // if decryption fails => immediately logout
      // set b , b = base64decode( b)
      b.exp is not expired  -- etc
```

return success


option 1
application.properties
option 2
env variable.
vault.


sanjana
yash
secretKey
hello
ac3$5e
secret Key
decode
hello