Authentication

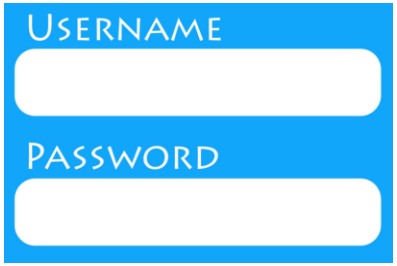
Authentication

- Password-based authentication
- 2-Factor Authentication (2FA)
- 2FA in Practice
- Single Sign-On using Token-based Security
- Delegated Authentication

Application Security Aspects

- **Authentication (Identity verification):**
  - Verify the identity of the user given the credentials received
  - Making sure the user is who he claims to be
- **Authorization:**
  - Determine if the user should be granted access to a particular resource.
- **Confidentiality:**
  - Encrypt sensitive data to prevent unauthorized access in transit or in storage
- **Data Integrality:**
  - Sign sensitive data to prevent the content from being tampered (e.g., changed in transit)

Password-based Authentication



Back

Passwords

- Passwords are the most commonly used authenticator in computer systems
  - A combination of letters, numbers, and special characters that a user supplies in order to prove their identity
  - Frequently passwords are chosen by the user
- But security problems associated with passwords:

81%

Data breaches in 2016 that involved **weak, default, or stolen passwords** *(VDBR)*

1 IN 14

Phishing attacks were successful in 2016 *(VDBR)*

1,579

Breaches in 2017, **a 45% increase over 2016** *(ITRC)*

Password Vulnerabilities

- Actually not very secure
  - Weak password: Users tend to pick simple passwords and easy to guess passwords
    - Good (random) passwords are almost impossible to remember
  - Password Reuse: re-use passwords between services
    - If one service is hacked, then someone knows your password for all other services
  - Vulnerable to key loggers
  - Vulnerable to phishing attacks

How Much Password Complexity is Needed?

- In general, a password can be thought of as good if brute-forcing it would require the same amount of work as a 64-bit encryption key
- How do we measure brute-force complexity of a password?

Brute force Complexity for Passwords

- How many different 4 character, all lowercase passwords are there?

? ? ? ?

$$26 * 26 * 26 * 26$$

$$=26^4$$

Translating into Key Complexity

$26^4 = 2^x$

$x=\log_2(26^4)$

$x=18.8$ bits

$26^4 \approx 2^{19}$

So, a 4 character password of all lowercase letters has the same brute force complexity as a ~19 bit encryption key

Let's make it better!

- 8 characters, all lowercase
  - 26^8 -> 37.6 bits
- 8 characters, lower and uppercase
  - 52^8 -> 45.6 bits
- 8 characters, lower, upper and numbers
  - 62^8 -> 47.6 bits
- 8 character, lower, upper, numbers, special
  - 94^8 -> 52.4 bits

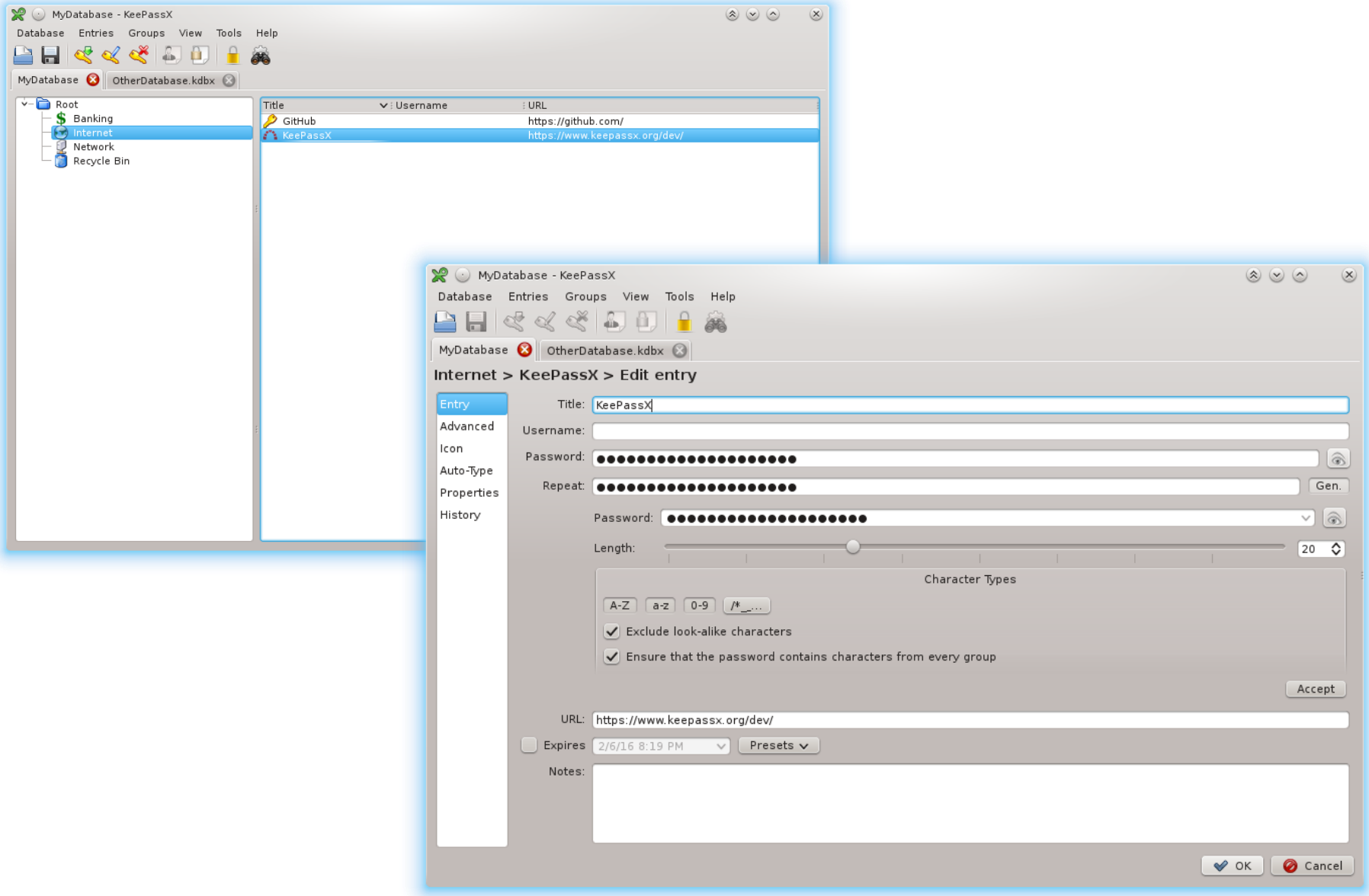
=> A randomly generated 8 character password with lowercase, uppercase, numbers, and special characters *is still not secure enough*

- How long would the password need to be for 64-bits of security?
  - 94^{10} -> 65.54 bits

Improvements - End-User Password Hygiene

- Use a different password for every account
- Make your passwords complex or randomly generated
- Use a password safe to generate and store passwords so that you don't need to remember them
  - All your passwords can be truly random!
- Open source password manager
  - http://www.keepassx.org/
- Commercial password managers
  - https://www.pcmag.com/article2/0,2817,2407168,00.asp

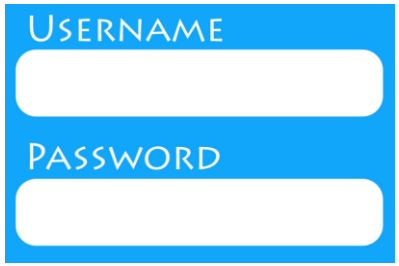
Password Safe



Improvements - Service Provider

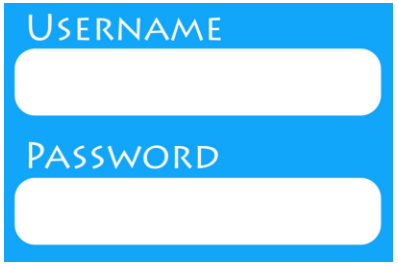
- Lock the account after 3 unsuccessful attempts
- Use 2-Factor Authentication (2FA) as Single Factor Authentication is insufficiency

2-Factor Authentication (2FA)

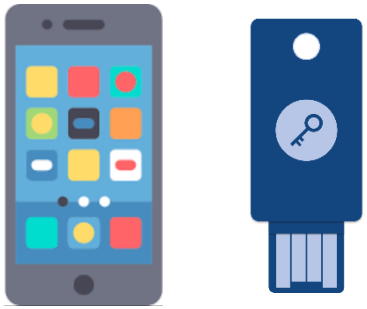


Back

Authentication Factors



Something the user knows
(PIN, password)



Something the user has
(mobile phone, device)

Something the user is
(biometric, retina, fingerprint)

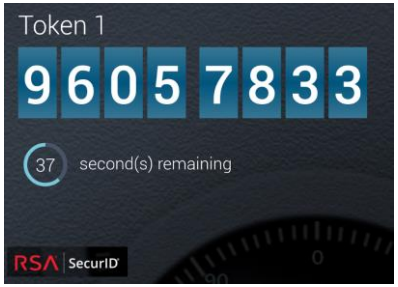
1. Something the user knows

- Information that would only be known by the user
- Examples:
  - Password (most common)
  - PIN number
  - Security questions (usually biographical information)

2. Something the user has

- A physical item the user has
- Examples
  - Smart phone
  - Smart card
  - Security token

- Security Token is Device or App that displays a number that changes ~every 30 seconds
  - Commonly used for corporate VPNs, generating one-time passwords (OTPs), etc.



- USB security key

3. Something the user is

- *Biometrics*
  - Based on physical characteristics of the user
- Examples
  - Fingerprints and Retina/Iris scan used in Qatar for e-government and e-gate systems
- Not revokeable or transferable…
- But the use of biometric information is less common since fingerprint or retina recognition software is expensive and difficult to implement.

Multi-Factor Authentication

- Using multiple factors of authentication increases security
  - Must include *unique* factors
  - In practice 2-Factor Authentication (2FA) is used
- Example:
  - Requiring a password and a security question is still only one-factor (what the user knows)
  - Requiring a password and the code from an SMS sent your phone is two-factor (what the user knows and what the user has)
  - Requiring a password and fingerprint: *two-factor* authentication (what the user knows and what the user is)

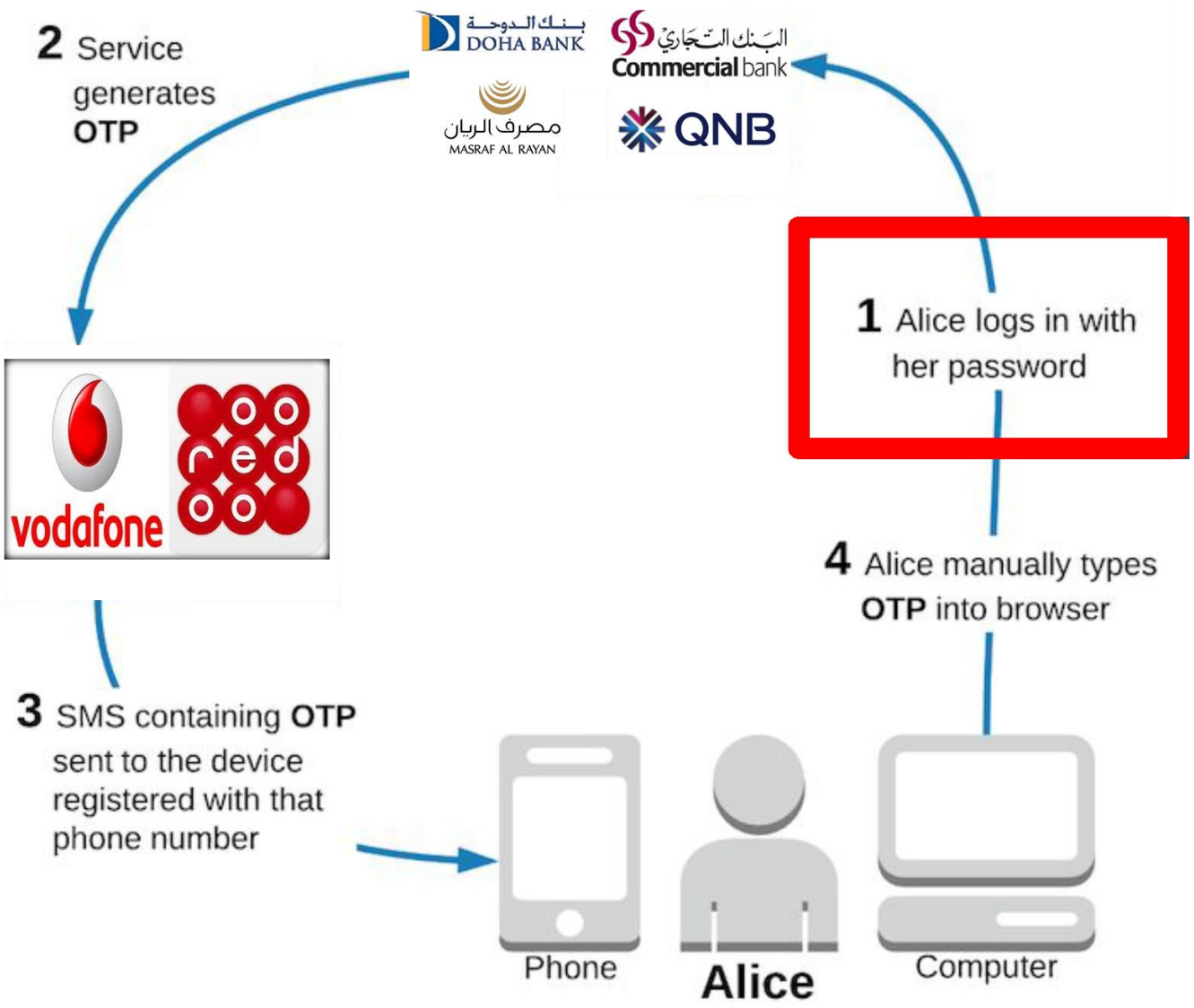
2FA in Practice

Back

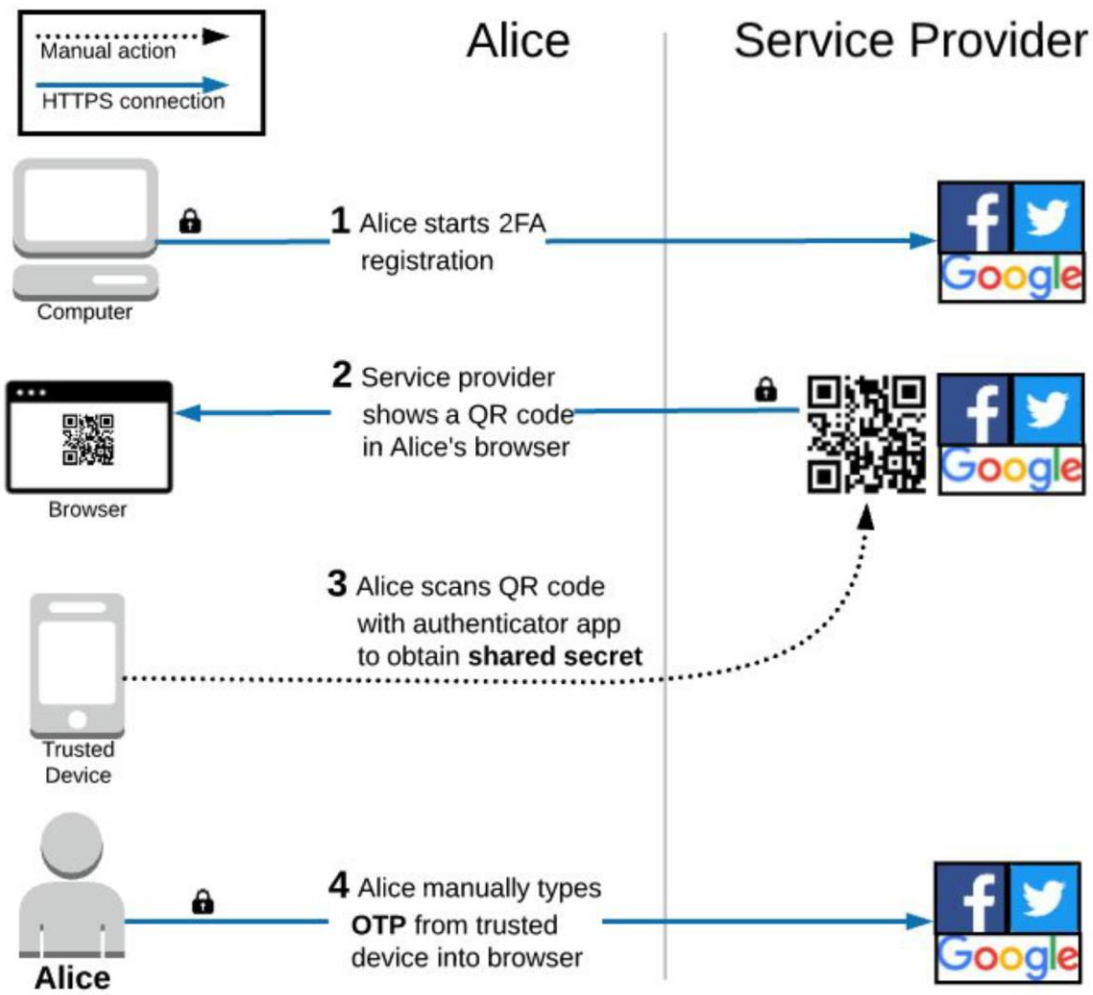
2FA Methods in Practice

1. SMS
2. Time-based One-time Passwords
   - e.g. Google Authenticator, RSA SecurID Security Token
3. Push notifications
   - e.g. Google Prompt
4. Universal 2nd Factor (U2F)
   - e.g. USB security keys

SMS – 2FA Authentication Flow

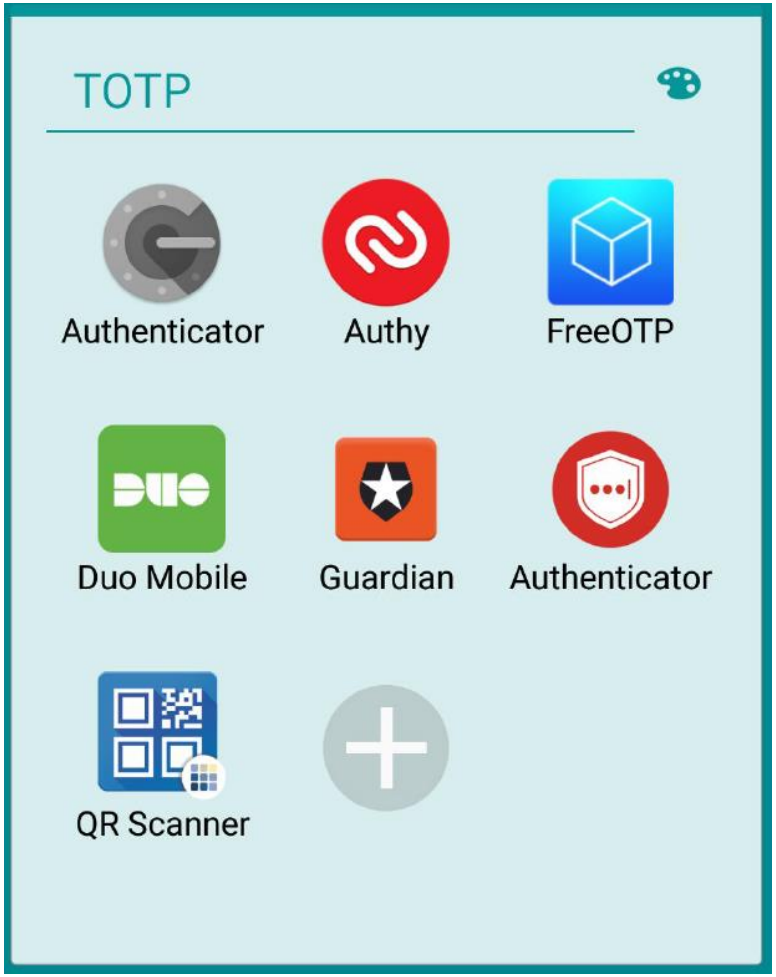


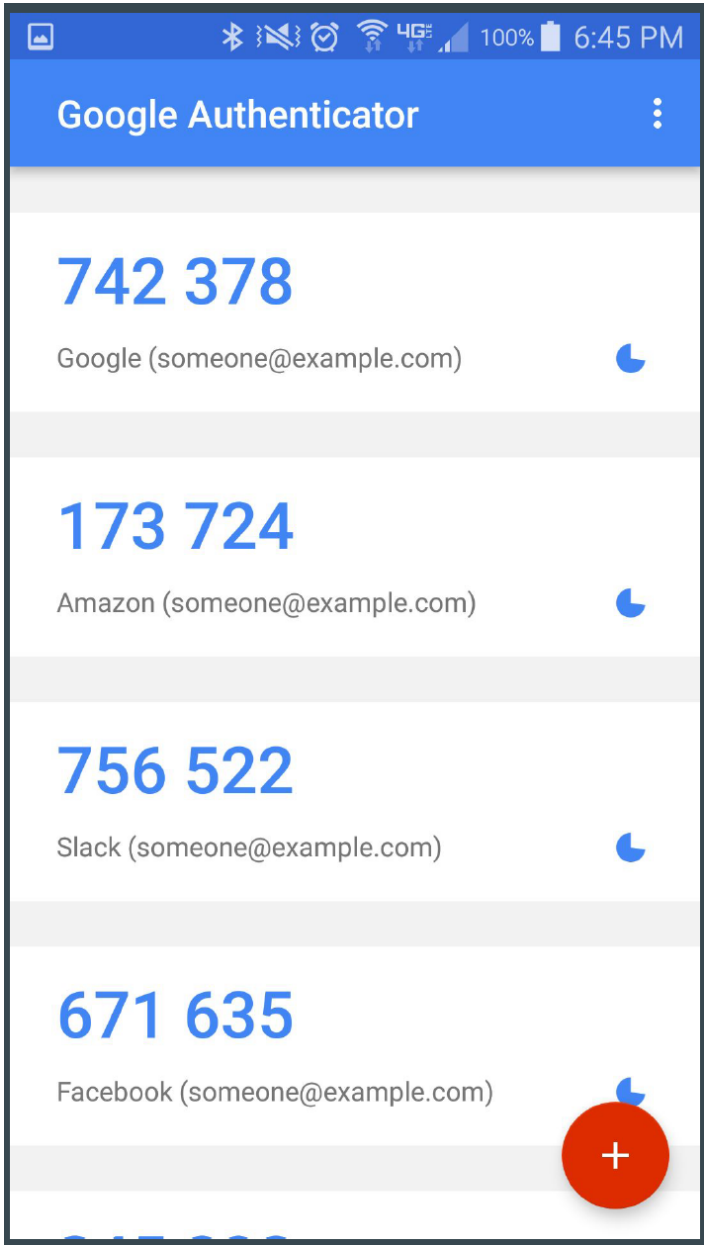
Time-based One-Time Password algorithm (TOTP)



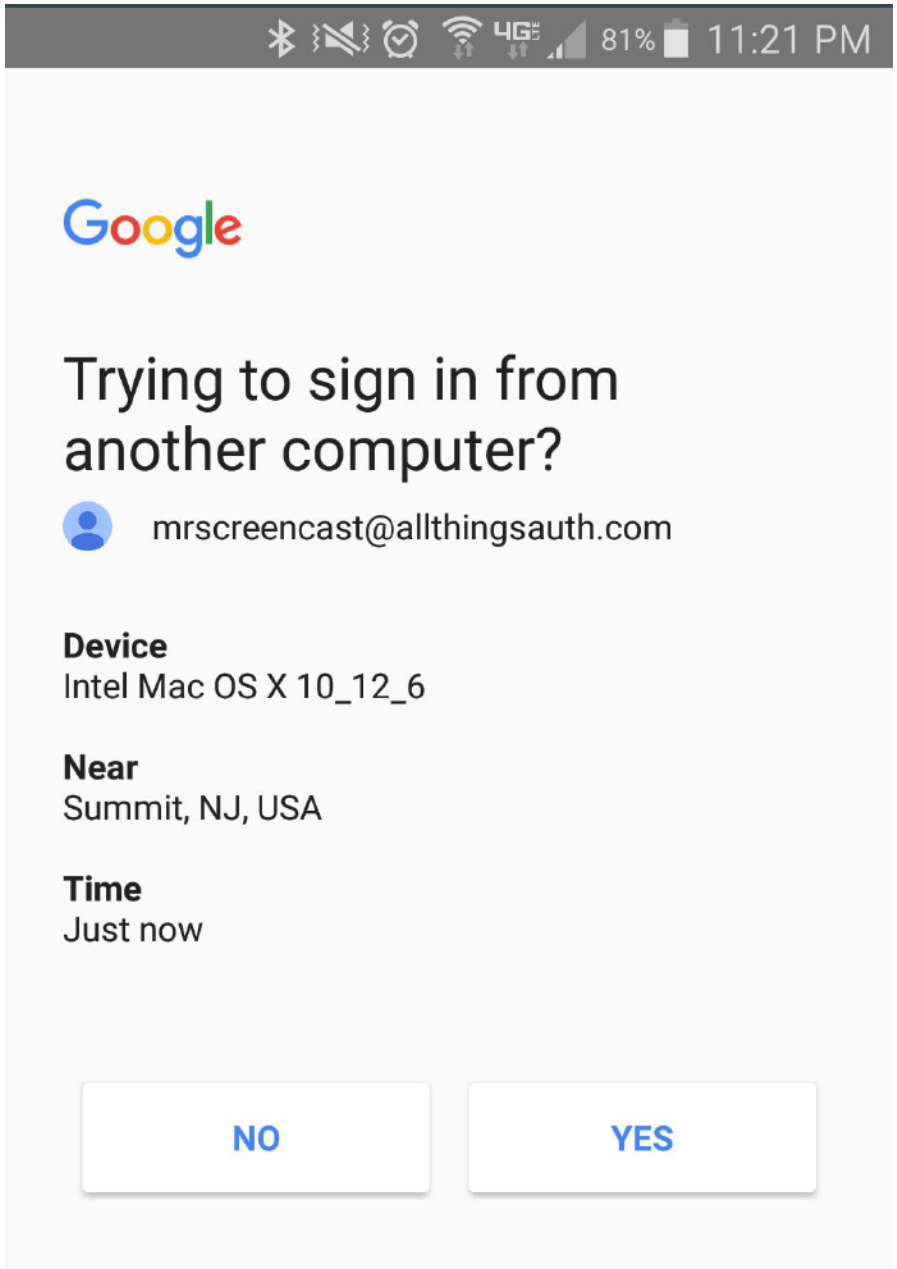
HMAC-SHA-1 (shared secret + time) ≈ TOTP

TOTP: Example Authenticator Apps

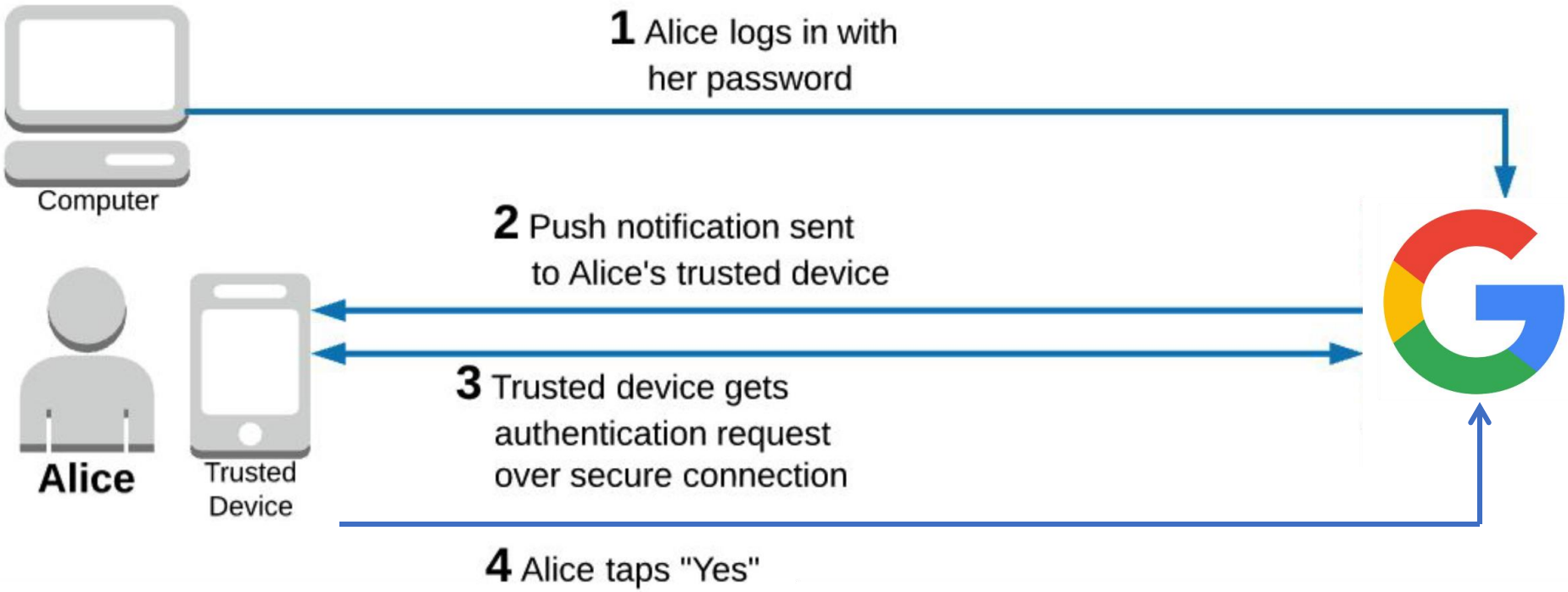




Push: Authentication prompt

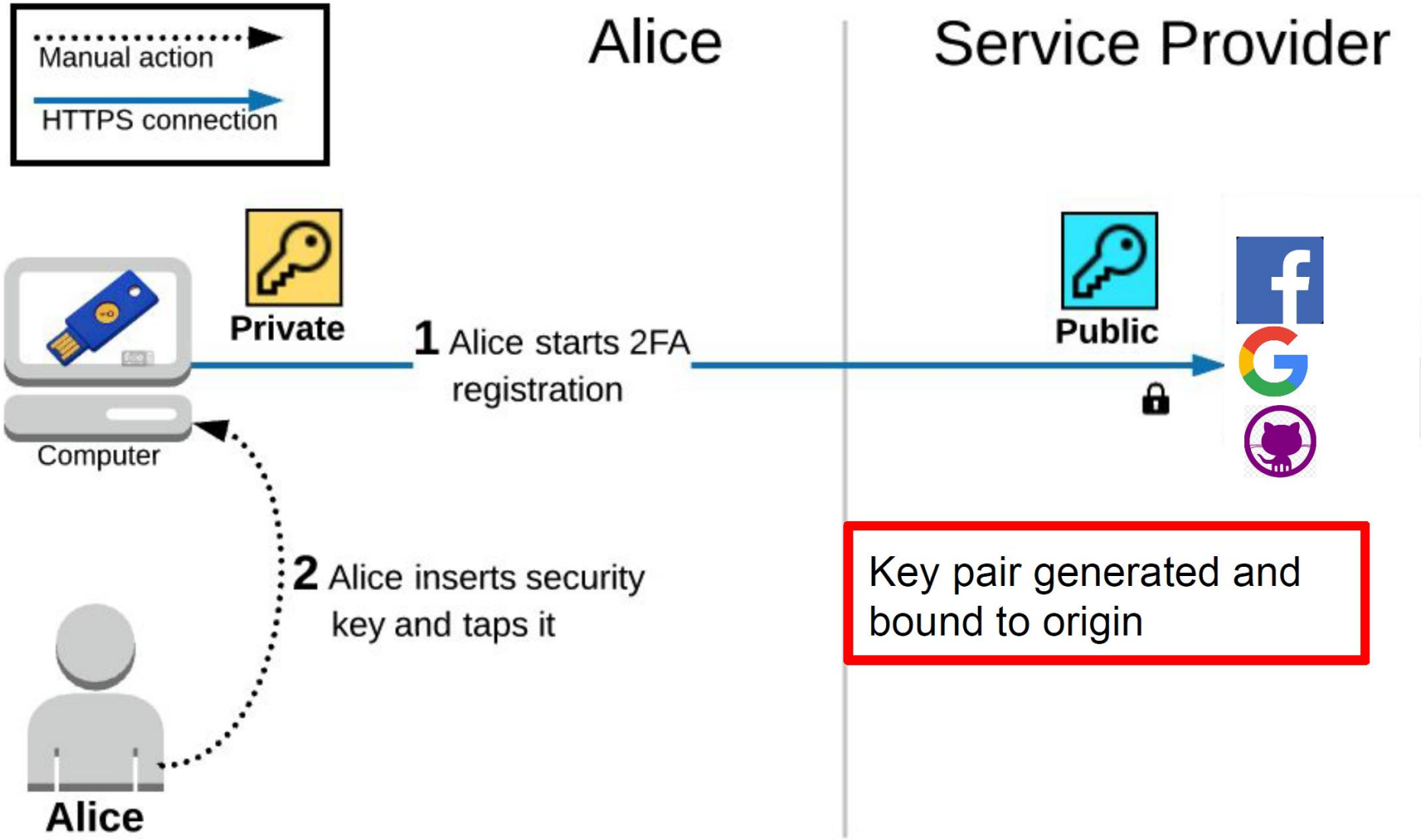


Push: Authentication flow

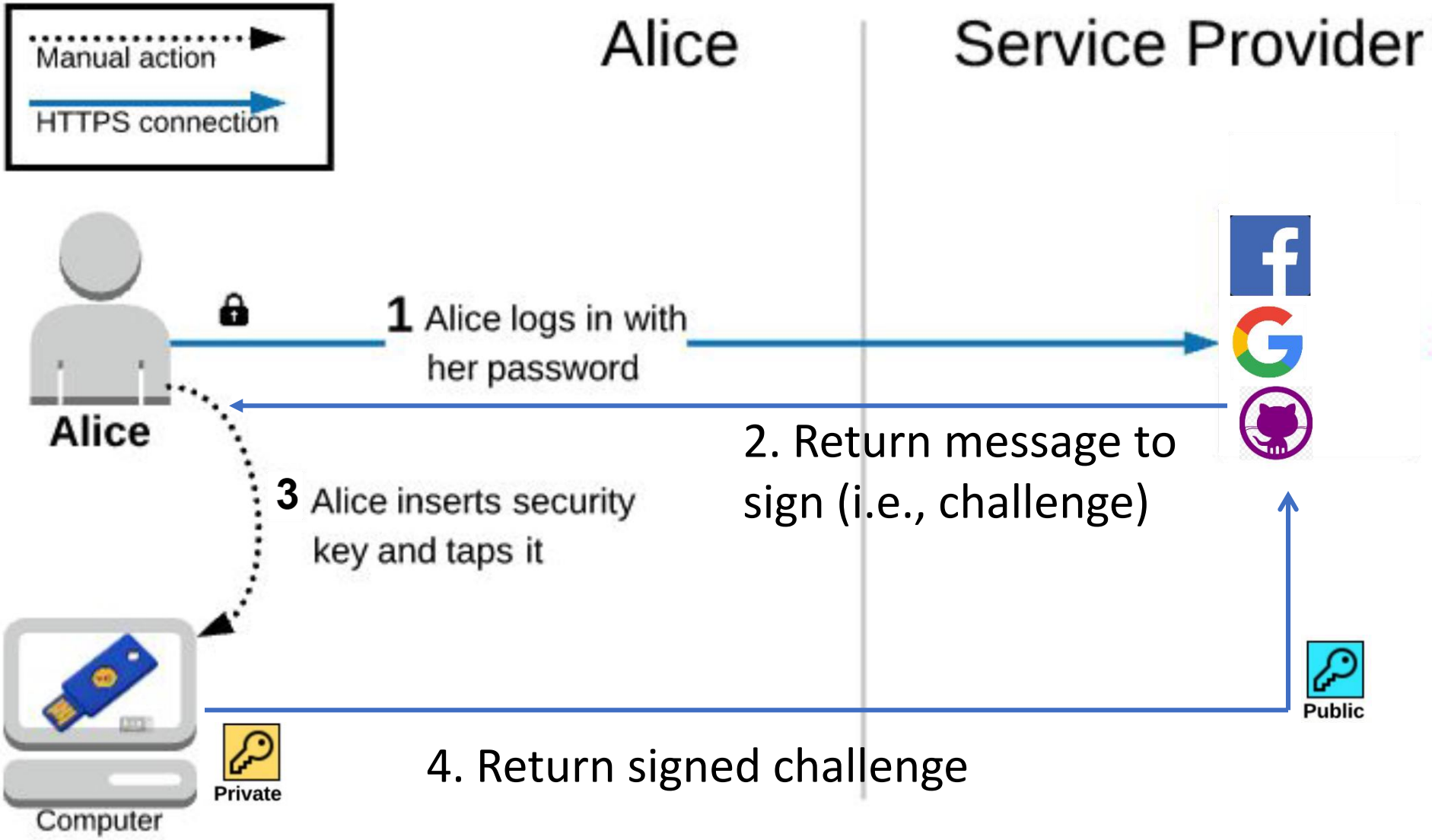


- Secure communication over HTTPS using public key cryptography
- Other solutions: Duo, Authy

U2F: Registration flow



U2F: Authentication flow



Single Sign-On Using Token-Based Security

Token based security

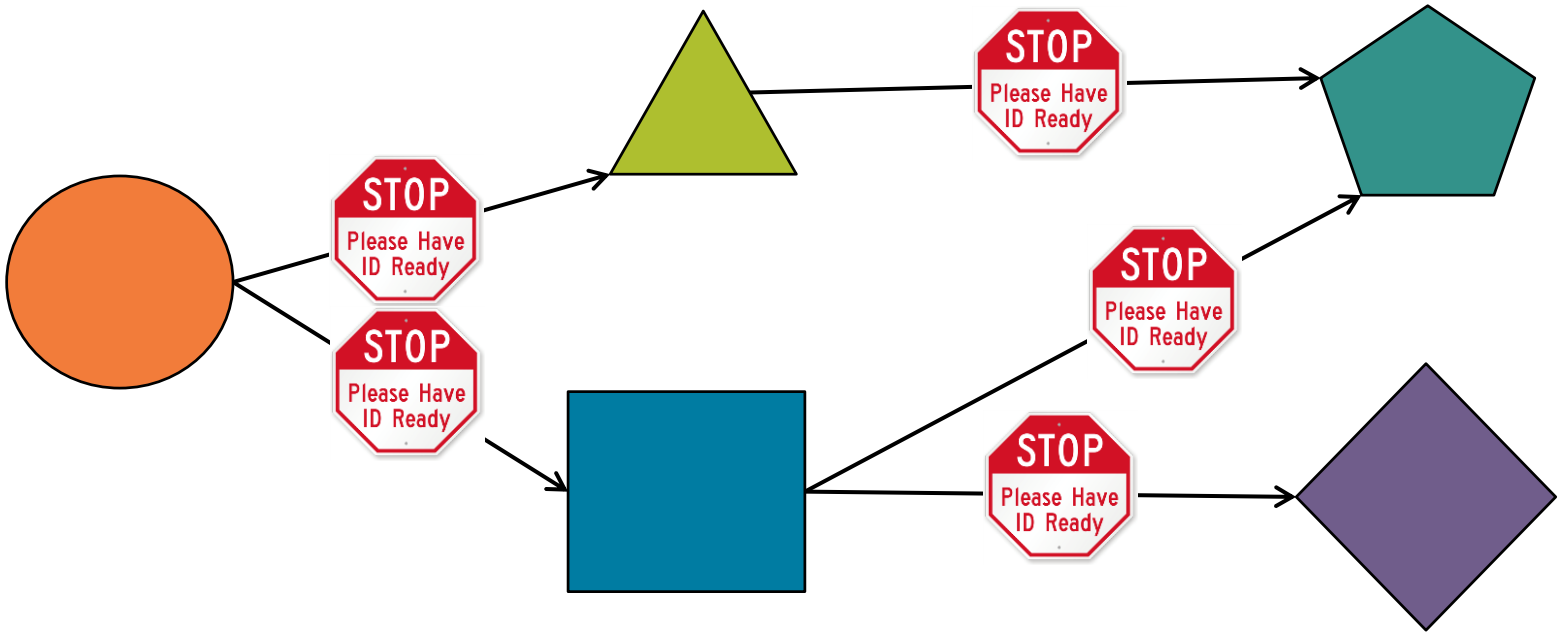
- After a successful authentication a JWT token is issued by the server and communicated to the client
- JWT token is a signed json object
  - contain information about issuer and subject (claims)
  - signed (tamper proof & authenticity)
  - typically contain an expiration time
- JWT is added to the HTTP header of subsequent requests to Web API
- A Web API (i.e., a resource) validate a token

JWT-based Single Sign-On Model

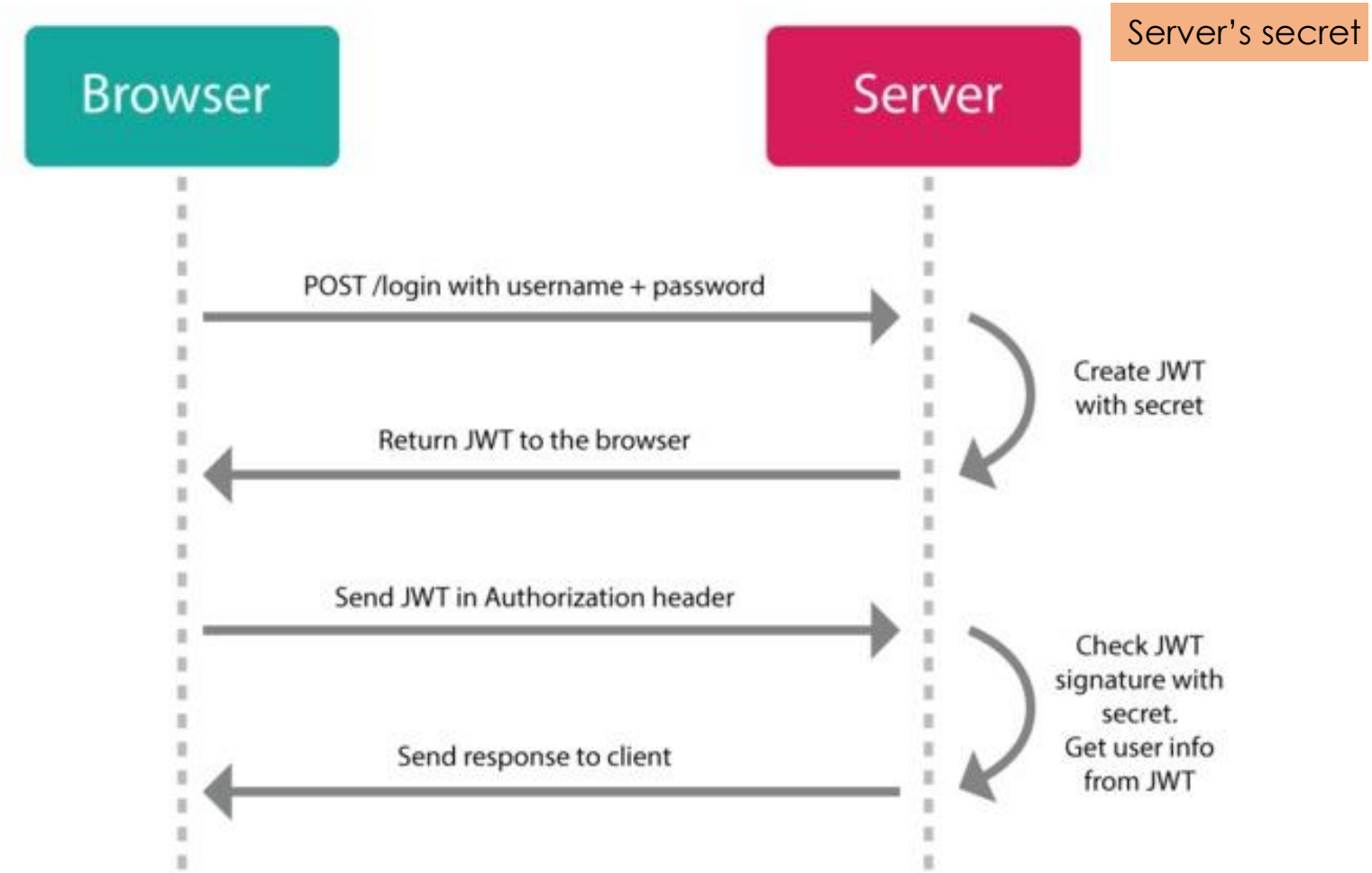
- JWT can be used for **Single Sign-On:**
  - Sharing the JWT between different applications
- A Web App gets a request that includes a JWT token
- Web App checks that the JWT token is valid
  - JSON Web Token (JWT) is a widely used token format
  - Token contains the roles/scopes that the user is authorized to access
  - Web App uses info in the token to make an access control decision

Securing Web App

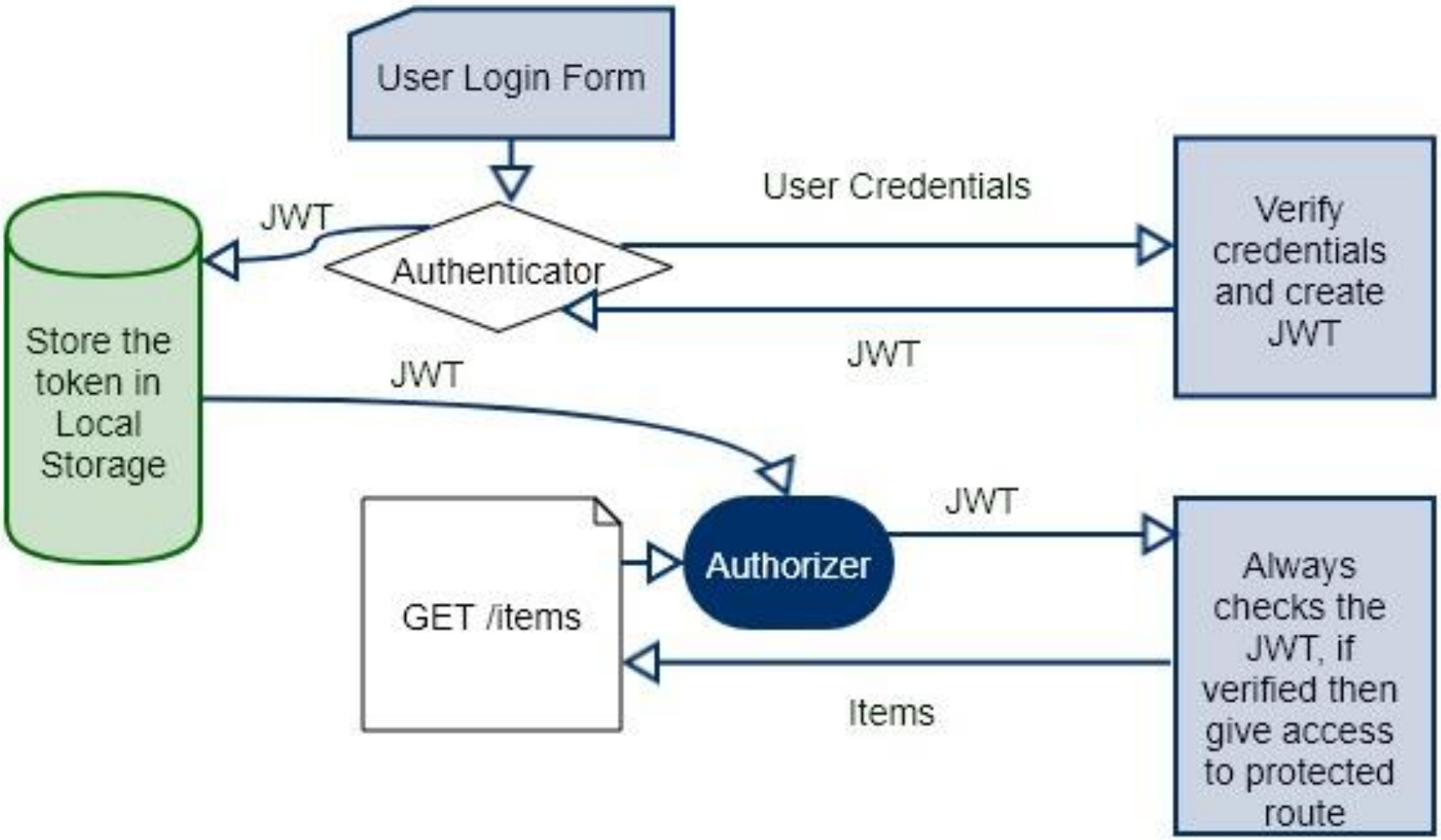
- Every request to a Web App must include a **security token** that the Web App can easily verify and use for making authorization decisions.



JSON Web Token (JWT)



How JWT Works



JWT Structure

Header

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

Claims

```
{
    role: "Admin",
    given_name: "Abdelkarim",
    family_name: "Erradi",
    name: "erradi",
    email: "erradi@jwt.org",
    iat: 1526597430,
    exp: 1526604630
}
```

eyJhbGciOiJub25lIn0.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD.4MTkzODAsDQogImh0dHA6Ly9leGFt

Header **Claims** **Signature**

Successful Login using JWT

- Sign in @ http://localhost:3040/auth/login

POST http://localhost:3040/auth/login

Authorization | Headers (1) | Body | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

```
1 {
2     "name": "erradi",
3     "password":"secret"
4 }
```

Body | Cookies | Headers (7) | Test Results

Pretty | Raw | Preview | JSON

```
1 {
2     "idToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJvaWRQcm92aWRlciI6ImxvY2Fs
          oiZXJyYWRpIiwiZW1haWwiOiJlcnJhZGlAand0Lm9yZyIsIl9fdiI6MCwiaWF0IjoxNTI2NTk
3 }
```

Use JWT to Access Protected Resource

- Get users http://localhost:3040/auth/users

GET http://localhost:3040/auth/users

Authorization | Headers (1) | Body | Pre-request Script | Tests

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJvaWRQcm92aWRlciI6ImxvY2FsIiwi
New key	Valu

Body | Cookies | Headers (7) | Test R

Pretty | Raw | Preview | JSON


```
[
    {
        "oidProvider": "local",
        "role": "Visitor",
        "defaultUrl": "/",
        "_id": "5afe02cbac0f4c4260622e46",
        "given_name": "Abdelkarim",
        "family_name": "Erradi",
        "name": "erradi",
        "email": "erradi@jwt.org",
        "password": "$2b$10$I/DbUjlOeja.dMnekCLZTOqnBi4xyI9zPCsE3e48xn5Dwpcs45NWe"
    }
]
```


Add the JWT token to standard **Authorization** header of HTTP requests to allow the Web API to verify it and allow access to resources

https://jwtinspector.io/

JWT Inspector is a chrome extension that lets you **decode** and **inspect** JWT in requests, and local storage

Overview (0) | Debug

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJvaWRQcm92aWRlciI6ImxvY2FsIiwicm9sZSI6IlZpc2l0b3IiLCJkZWZhdWx0VXJsIjoiLyIsIl9pZCI6IjVhZmUwMmNiYWMwZjRjNDI2MDYyMmU0NiIsImdpdmVuX25hbWUiOiJBYmRlbGthcmltIiwiZmFtaWx5X25hbWUiOiJFcnJhZGkiLCJuYW1lIjoiZXJyYWRpIiwiZW1haWwiOiJlcnJhZGlAand0Lm9yZyIsIl9fdiI6MCwiaWF0IjoxNTI2NTk2NDQ1LCJleHAiOjE1MjY2MDM2NDV9.fwG_o7zbvdEIRnNifQ5Bj8sZ5Q4VxtaC5c6VPbxaDLc

Header


```
{
  alg: "HS256",
  typ: "JWT"
}
```


Payload


```
{
  oidProvider: "local",
  role: "Visitor",
  defaultUrl: "/",
  _id: "5afe02cbac0f4c4260622e46",
  given_name: "Abdelkarim",
  family_name: "Erradi",
  name: "erradi",
  email: "erradi@jwt.org",
  __v: 0,
  iat: 1526596445,
  exp: 1526603645
}
```


Signature

fwG_o7zbvdEIRnNifQ5Bj8sZ5Q4VxtaC5c6VPbxaDLc

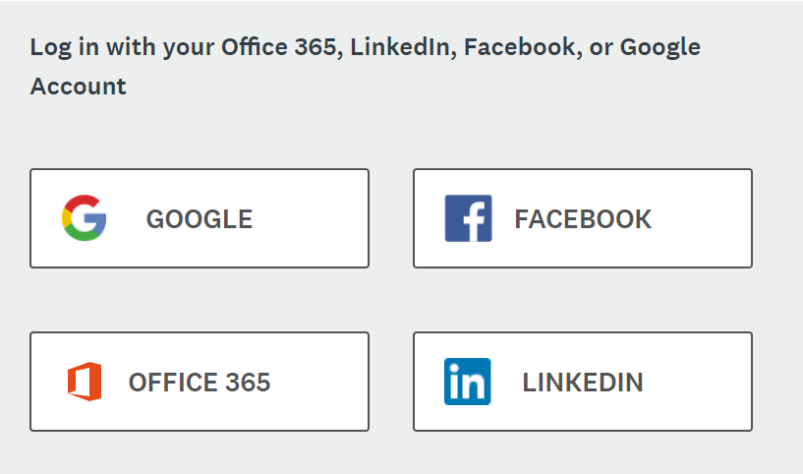
Delegated Authentication

Authentication is hard

- Trying to write your own login system is difficult:
  - Need to save passwords securely
  - Provide recovery of forgotten passwords
  - Make sure users set a good password
  - Detect logins from suspecious locations or new devices
  - etc.
- Luckily, **you don't have to build your own authentication!**
- You can use **OpenID Connect** to delegate login to an **Identity Provider** and get the user's profile

OpenID Connect

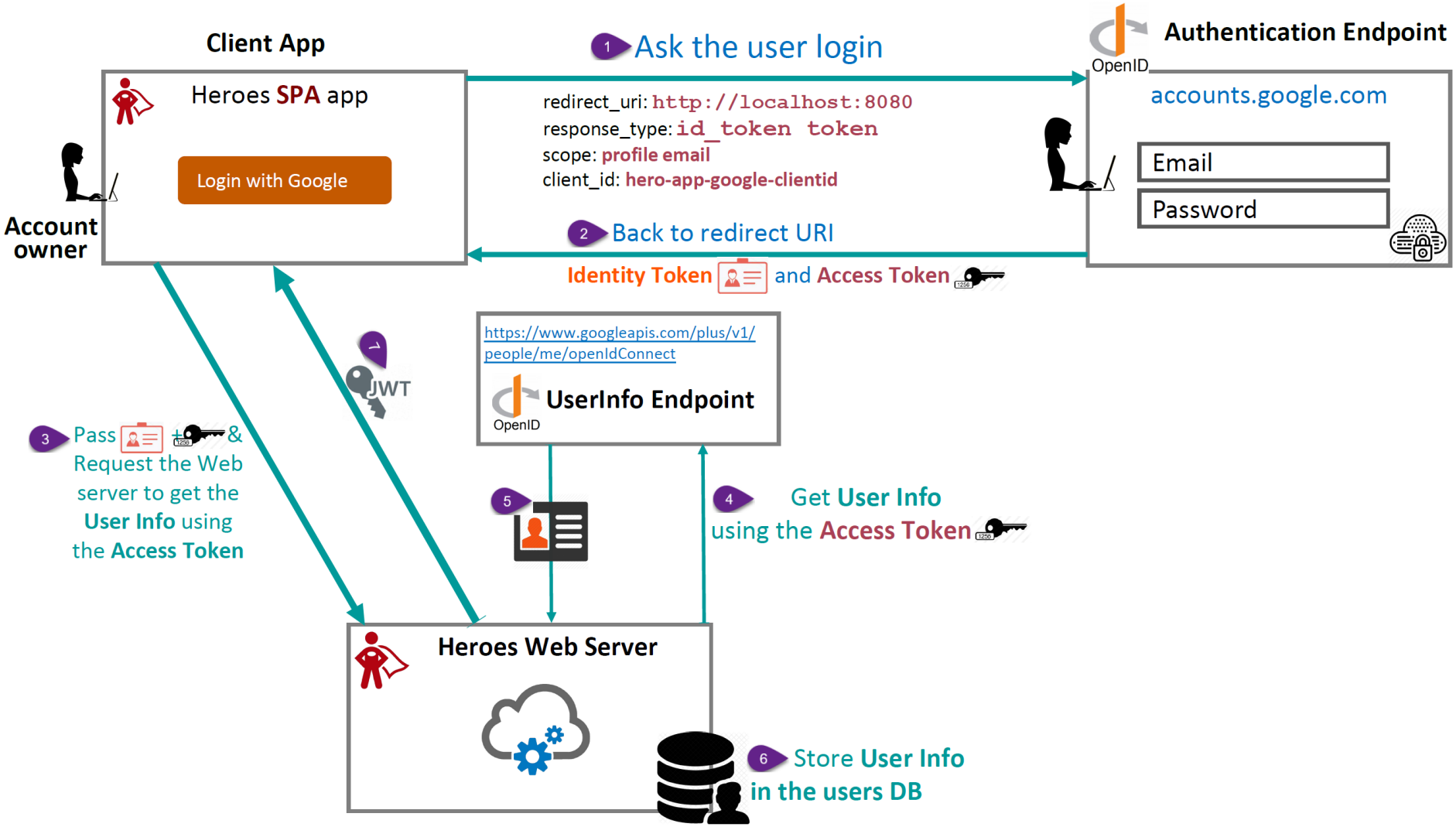
- **OpenID Connect** is a standard for user authentication
  - For users:
    - It allows a user to log into a website like AirBnB via some other service, like Google or Facebook
  - For developers:
    - It lets you authenticate a user without having to implement log in
  - Examples: "Log in with Facebook"



OpenID Connect APIs

- Companies like Google, Facebook, Twitter, and GitHub offer OpenID Connect APIs:
  - Google Sign-in API
  - Facebook Login API
  - Twitter Login API
  - GitHub Apps/Integrations
  - OpenID Connect is standardized, but the API that these services provide are slightly different
  - You must read the documentation to understand how to connect via their API
- After the user logins, you will get the user profile such name, email, etc

OpenID Connect Authentication Flow



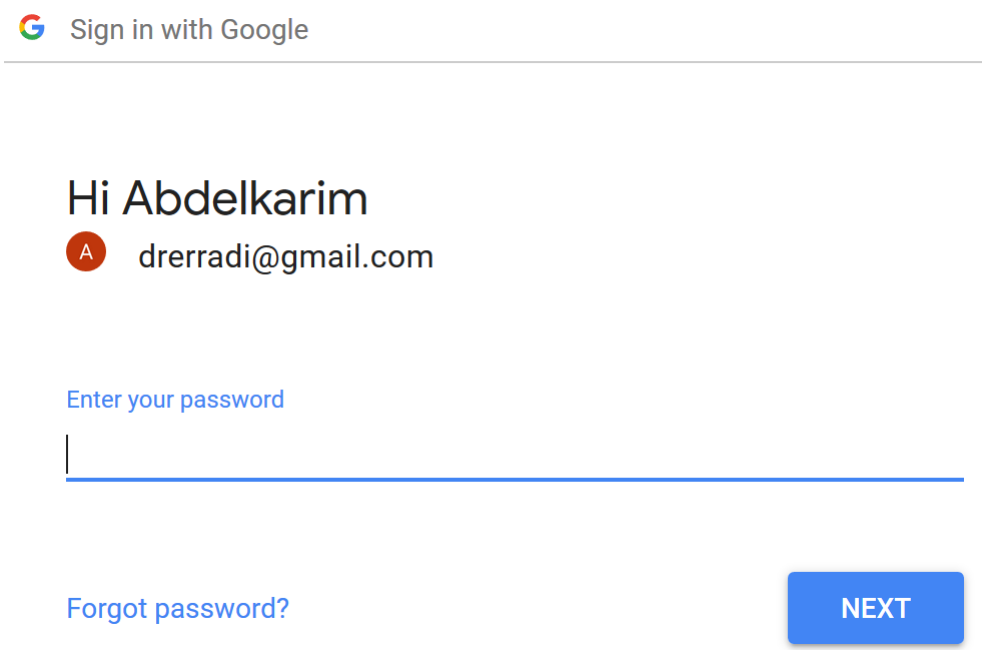
Authenticating via a SPA App

- **User** starts the flow by visiting a SPA App
- **Client** sends authentication request with `profile` scope via browser redirect to the **Authorization** endpoint
- **User** authenticates and consents to **Client** to access user's identity
- **ID Token** and **Access Token** is returned to **Client** via browser redirect
- **Client** optionally fetches additional user info with the **Access Token** from **UserInfo** endpoint

Authorization Request

- Ask the user to login via browser redirect to the Authentication Endpoint

https://accounts.google.com/o/oauth2/auth



- This will return an **Access Token** to the client to allow it to request the user's profile from the UserInfo Endpoint

Authentication Parameters

GET https://accounts.google.com/o/oauth2/auth?scope=profile%20email%20phone&client_... Params **Send**

Key	Value
scope	profile%20email%20phone
client_id	866457396346-piq09ek9kiofq9uspsnjuh...
response_type	id_token%20token
redirect_uri	http://localhost:8080

Scope = what user info the client needs access to?

Need to register and get **client_id** from https://console.developers.google.com/apis/credentials

What is the desired response type to get from the Authentication EndPoint?

- **id_token**: jwt of the authentication user
- **token**: access-token to be able to access the UserInfo endpoint

Redirect_udrl = callback address google will use to deliver to access_token and id_token

Body Cookies (2) Headers (15) Test Results Status: 200 OK Time: 461 ms Size: 72 KB

Google

One account. All of Google.

Sign in with your Google Account

Email or phone

Next

ID Token

Example ID Token from Google

```
{
  iss: "accounts.google.com",
  aud: "lv1muk.apps.googleusercontent.com",
  sub: "111893194175723488203",
  email: "karimerradi@gmail.com",
  email_verified: true,
  exp: 1526656174,
  iat: 1526652574
}
```

- JWT representing logged-in user
- Claims:
  - `iss` - Issuer
  - `sub` - User Identifier
  - `aud` - Audience for ID Token
  - `exp` - Expiration time
  - `iat` - Time token was issued

Scopes for Identify Claim Requests

- Scopes = what user info you need access to?
- Standard scopes:

  `openid` – JWT representing logged-in user

  `profile` – Profile info

  `email` – Email address & verification status

  `address` – Postal address

  `phone` – Phone number & verification status

Calling the UserInfo Endpoint

- Get the user's profile from the UserInfo Endpoint

GET https://www.googleapis.com/plus/v1/people/me/openIdConnect Params Send

Authorization Headers (1) Body Pre-request Script Tests

Key	Value	Description
Authorization	Bearer ya29.Gly_BcvwEQdaZgHKJIk2nB7N2g3falZmpCxp3NXM7UjoWxou_1Jp4v...	
New key	Value	Description

Bulk Edit

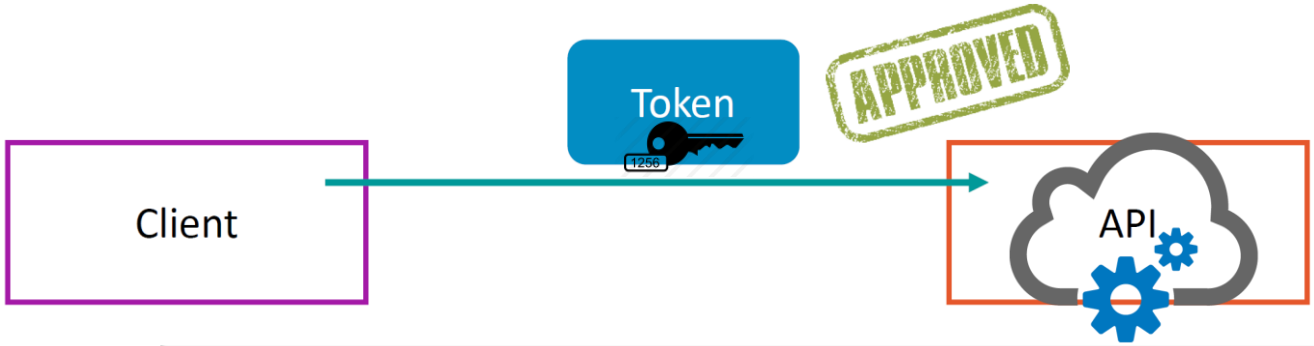
Body Cookies Headers (14) Test Results Status: 200 OK Time: 663 ms

Pretty Raw Preview JSON


```
{
    "kind": "plus#personOpenIdConnect",
    "gender": "male",
    "sub": "111893194175723488203",
    "name": "Erradi",
    "given_name": "Erradi",
    "family_name": "",
    "profile": "https://plus.google.com/111893194175723488203",
    "picture": "https://lh6.googleusercontent.com/-iuZD8qYF0xQ/AAAAAAAAAAI/AAAAAAAAGIM/ll35MtiUkJ8/photo.jpg?sz=50",
    "email": "karimerradi@gmail.com",
    "email_verified": "true",
    "locale": "en"
}
```


Send the **access token** received after the authentication. Add it to the **Authorization** header.

Use the Access Token to access Web Resources



- Validate token
- Grant access to the resource

GET https://people.googleapis.com/v1/people/me/connections?personFields=names,emailAddresses,phoneNum... Params Send

Key	Value	Description
personFields	names,emailAddresses,phoneNumbers,addresses,pho...	

Authorization Headers (1) Body Pre-request Script Tests

Key	Value	Description
Authorization	Bearer ya29.GlvABUS5jPiHTOywJTpKr6cCwUEsACshu3...	

Body Cookies (1) Headers (13) Test Results Status: 200 OK Time: 253 ms

Pretty Raw Preview JSON


```
1  {
2      "connections": [
3          {
4              "resourceName": "people/c3945308633452445077",
5              "etag": "%EggBAgMJCxA3LhoMAQIDBAUGBwgJCgsMIgxuWmJUQU5BT3FKOD0=",
6              "names": [
7                  {
8                      "metadata": {
9                          "primary": true,
10                         "source": {
11                             "type": "CONTACT",
12                             "id": "36c08d4c8a36fd95"
13                         }
14                     },
15                     "displayName": "Qatar University",
16                     "familyName": "University",
17                     "givenName": "Qatar",
18                     "displayNameLastFirst": "University, Qatar"
19                 }
20             ],
```

Summary

- Three types of authentication factors
  1. What the user knows
  2. What the user has
  3. What the user is
- 2 Factors Authentication (2FA) is better
- JWT is easy to create, transmit and validate to protect Web Apps in a scalable way
- Use OpenID Connect for **Delegated Authentication**

Resources

- NIST Digital Identity Guidelines

https://pages.nist.gov/800-63-3/

- JWT Handbook

https://auth0.com/resources/ebooks/jwt-handbook

- Authentication Survival Guide

https://auth0.com/resources/ebooks/authentication-survival-guide

- What the Heck is OpenID Connect?

https://www.youtube.com/watch?v=6ypYXxRPKgk