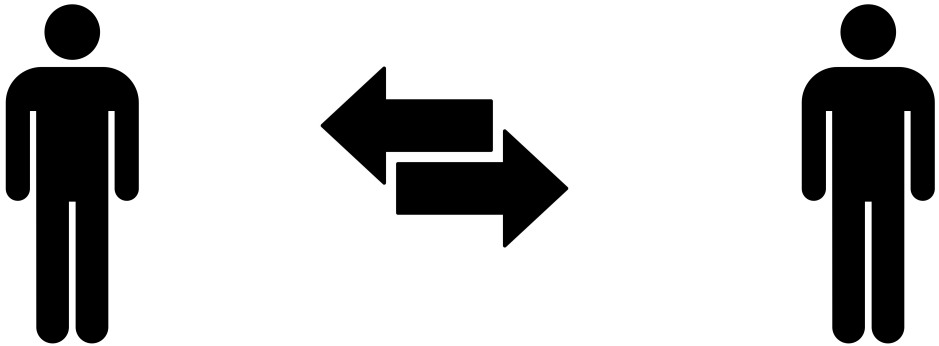
API Security

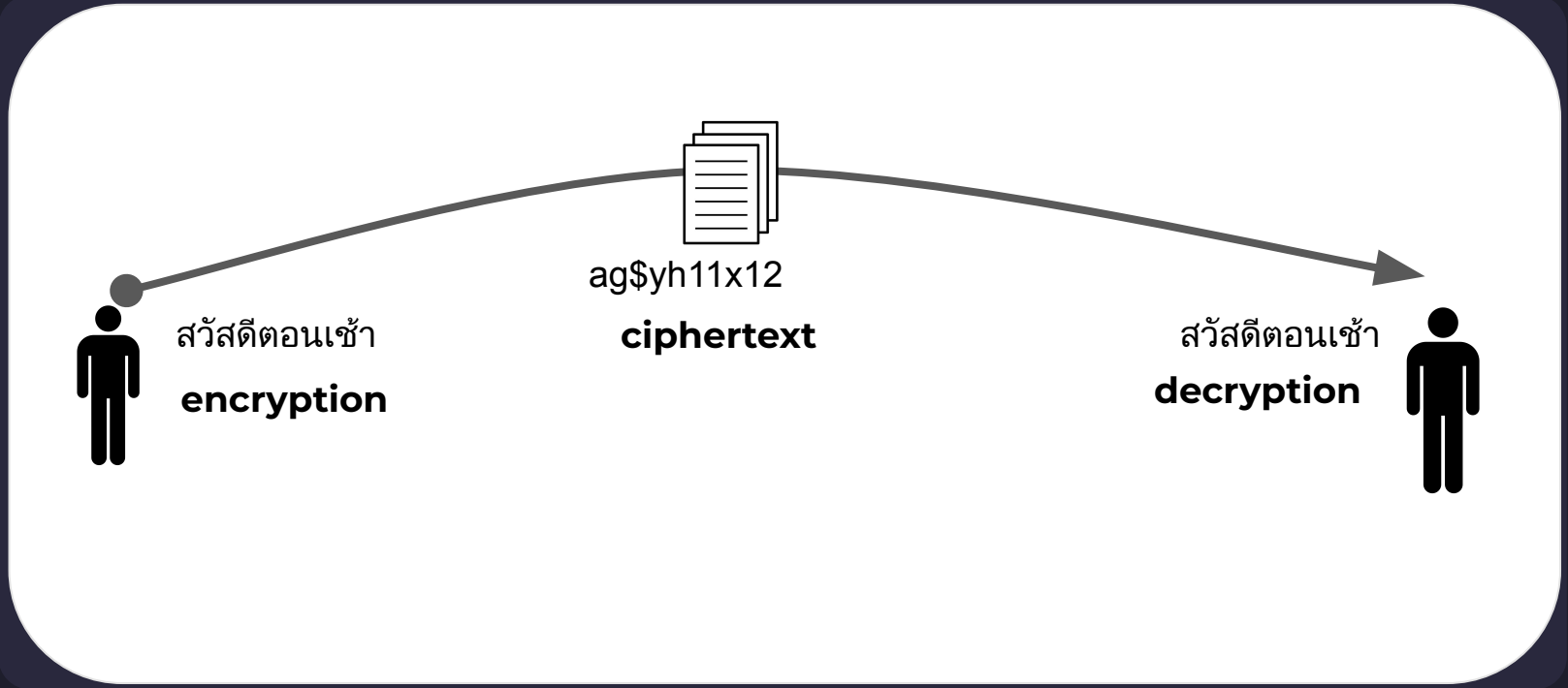
exchange information securely over the http protocol.

Communication



Communication

Encryption

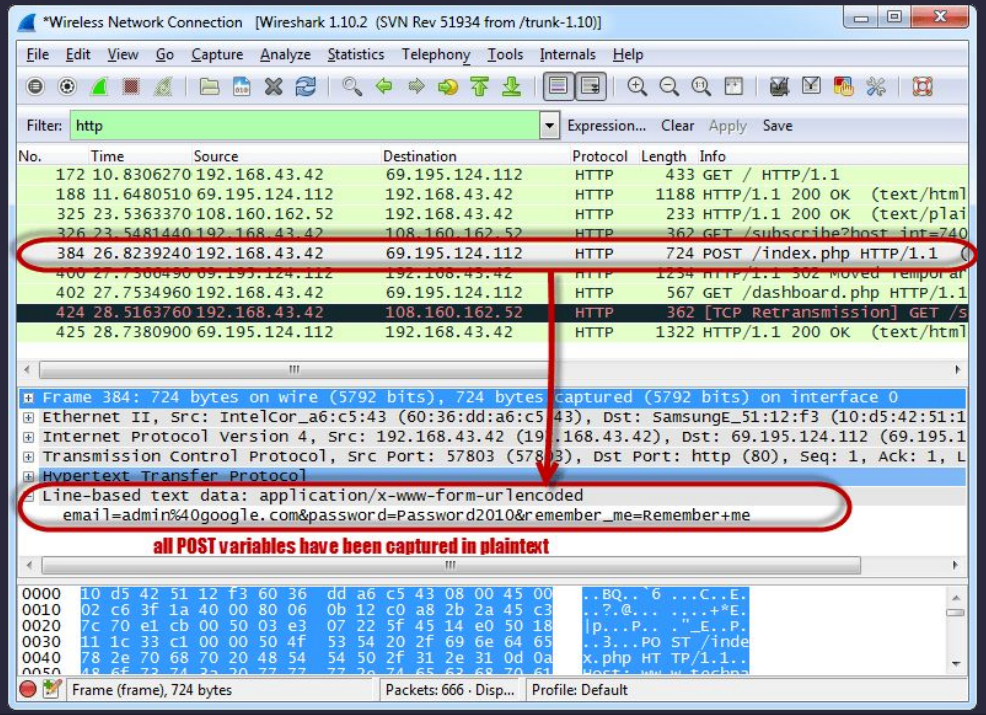

ag$yh11x12
ciphertext
สวัสดีตอนเช้า
encryption
สวัสดีตอนเช้า
decryption

HTTP

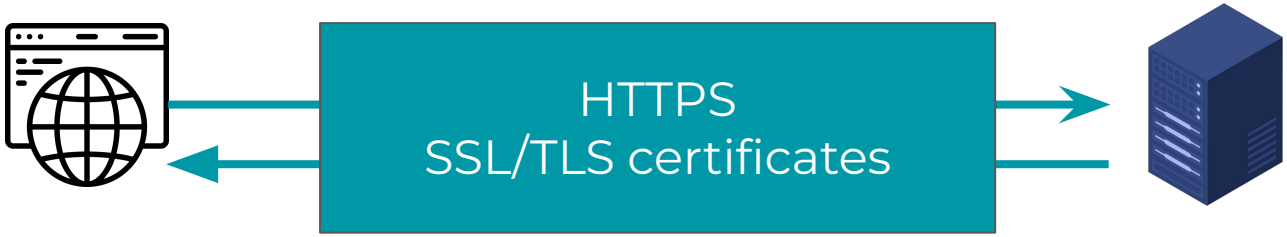

```
POST /login HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.124 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://www.example.com/login-page
Content-Type: application/x-www-form-urlencoded
Content-Length: 37

username=JohnDoe&password=Pass1234
```

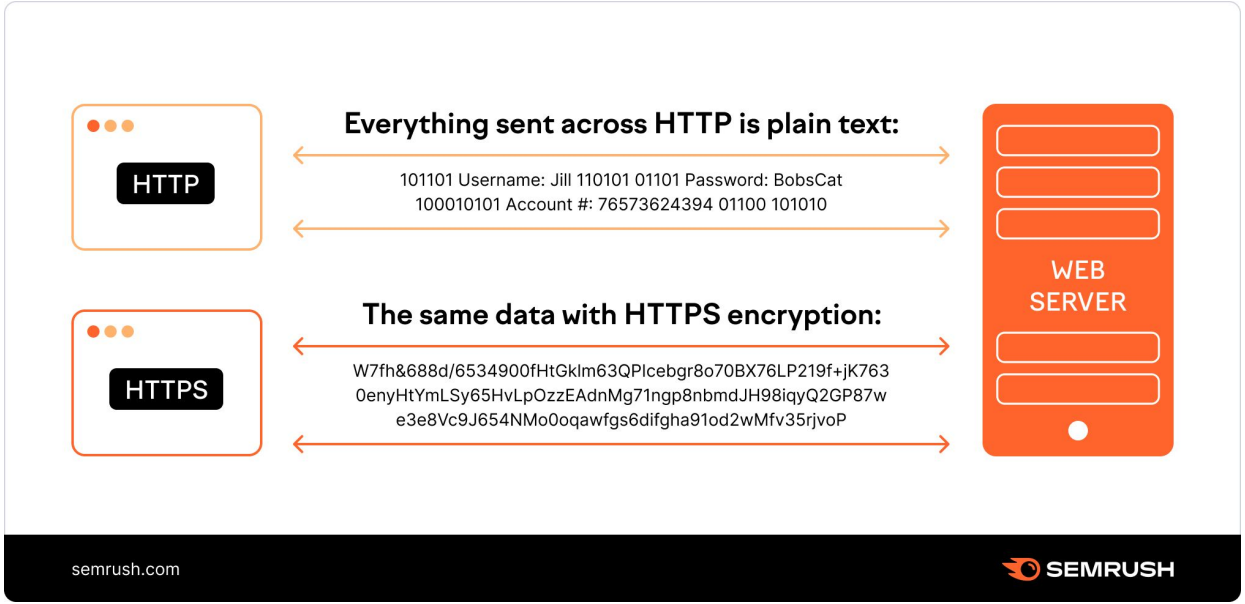
HTTP


*Wireless Network Connection [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
172 10.8306270 192.168.43.42 69.195.124.112 HTTP 433 GET / HTTP/1.1
188 11.6480510 69.195.124.112 192.168.43.42 HTTP 1188 HTTP/1.1 200 OK (text/html
325 23.5363370 108.160.162.52 192.168.43.42 HTTP 233 HTTP/1.1 200 OK (text/plai
384 26.8239240 192.168.43.42 69.195.124.112 HTTP 724 POST /index.php HTTP/1.1
402 27.7534960 192.168.43.42 69.195.124.112 HTTP 567 GET /dashboard.php HTTP/1.1
424 28.5163760 192.168.43.42 108.160.162.52 HTTP 362 [TCP Retransmission] GET /s
425 28.7380900 69.195.124.112 192.168.43.42 HTTP 1322 HTTP/1.1 200 OK (text/html
Frame 384: 724 bytes on wire (5792 bits), 724 bytes captured (5792 bits) on interface 0
Ethernet II, Src: IntelCor_a6:c5:43 (60:36:dd:a6:c5:43), Dst: SamsungE_51:12:f3 (10:d5:42:51:1
Internet Protocol Version 4, Src: 192.168.43.42 (192.168.43.42), Dst: 69.195.124.112 (69.195.1
Transmission Control Protocol, Src Port: 57803 (57803), Dst Port: http (80), Seq: 1, Ack: 1, L
Hypertext Transfer Protocol
Line-based text data: application/x-www-form-urlencoded
email=admin%40google.com&password=Password2010&remember_me=Remember+me
all POST variables have been captured in plaintext
0000 10 d5 42 51 12 f3 60 36 dd a6 c5 43 08 00 45 00 ..BQ.. 6 ...C..E.
0010 02 c6 3f 1a 40 00 80 06 0b 12 c0 a8 2b 2a 45 c3 ..?.@... ....+*E.
0020 7c 70 e1 cb 00 50 03 e3 07 22 5f 45 14 e0 50 18 |p...P.. ."_E..P.
0030 11 1c 33 c1 00 00 50 4f 53 54 20 2f 69 6e 64 65 ..3...PO ST /inde
0040 78 2e 70 68 70 20 48 54 54 50 2f 31 2e 31 0d 0a x.php HT TP/1.1..
Frame (frame), 724 bytes
Packets: 666 · Disp...
Profile: Default

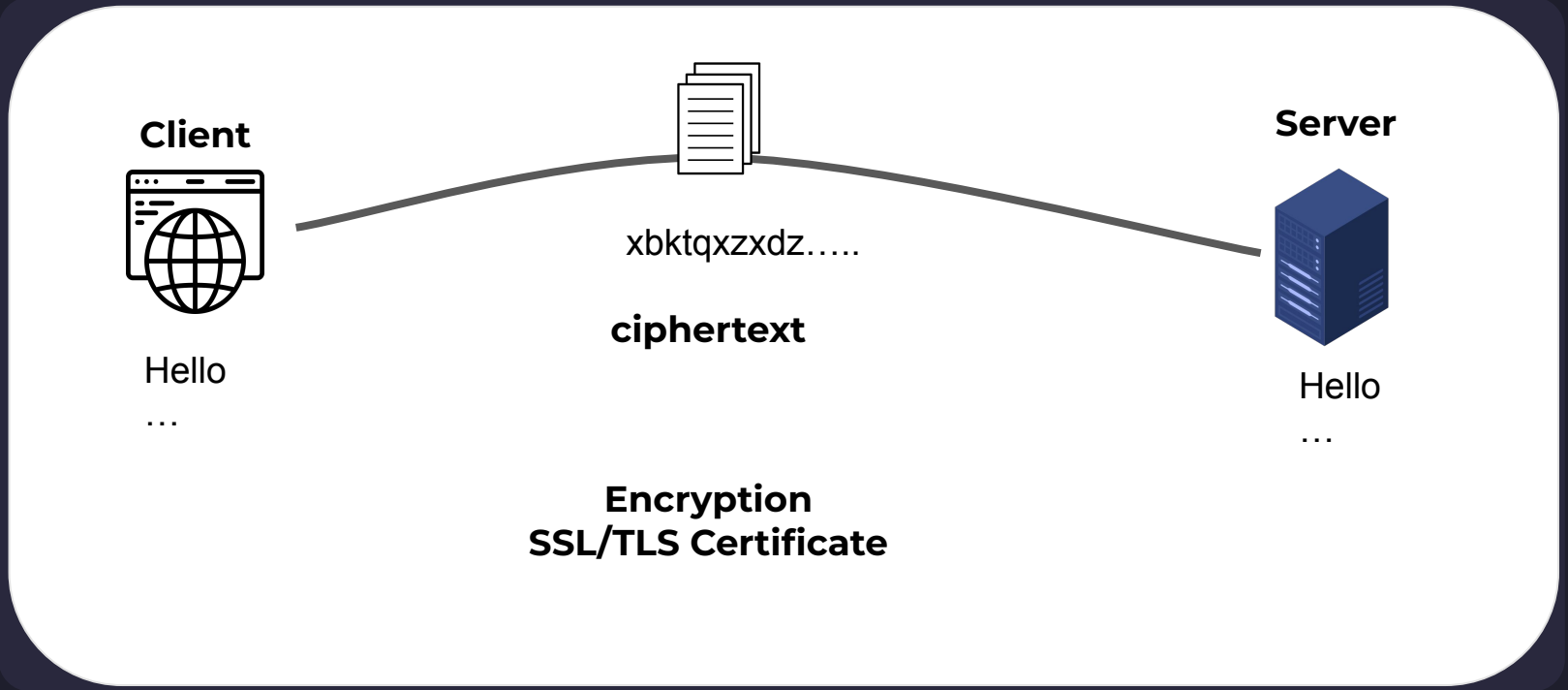
HTTPS



HTTPS


HTTP
Everything sent across HTTP is plain text:
101101 Username: Jill 110101 01101 Password: BobsCat
100010101 Account #: 76573624394 01100 101010
HTTPS
The same data with HTTPS encryption:
W7fh&688d/6534900fHtGklm63QPlcebgr8o70BX76LP219f+jK763
0enyHtYmLSy65HvLpOzzEAdnMg71ngp8nbmdJH98iqyQ2GP87w
e3e8Vc9J654NMo0oqawfgs6difgha91od2wMfv35rjvoP
WEB
SERVER
semrush.com
SEMRUSH

HTTPS


Client
Hello
…
xbktqxzxdz…..
ciphertext
Server
Hello
…
Encryption
SSL/TLS Certificate

HTTPS

"**HyperText Transfer Protocol <u>Secure</u>.**" It is the **secure version of HTTP**, the primary protocol used to send data between a web browser and a website

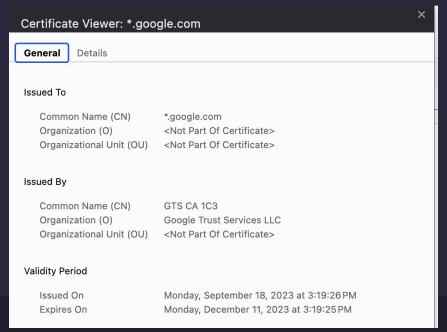
SSL/TLS Certificate

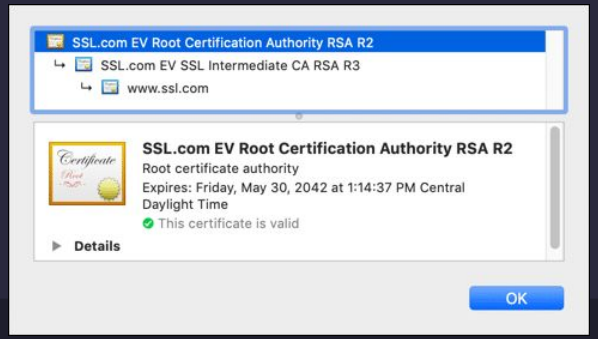
It is a **protocol** or communication rule that allows computer systems to talk to each other on the internet safely. SSL/TLS certificates allow web browsers **to identify and establish encrypted network connections** to websites using the SSL/TLS protocol.

Certificate Validation

A certificate authority (**CA**) is an organization that **sells SSL/TLS certificates** to web owners, web hosting companies, or businesses. **The CA validates the domain** and **owner details before issuing the SSL/TLS certificate.**

To be a CA, an organization must meet specific requirements set by the operating system, browsers, or mobile devices company and **apply to be listed as a root certificate authority.**

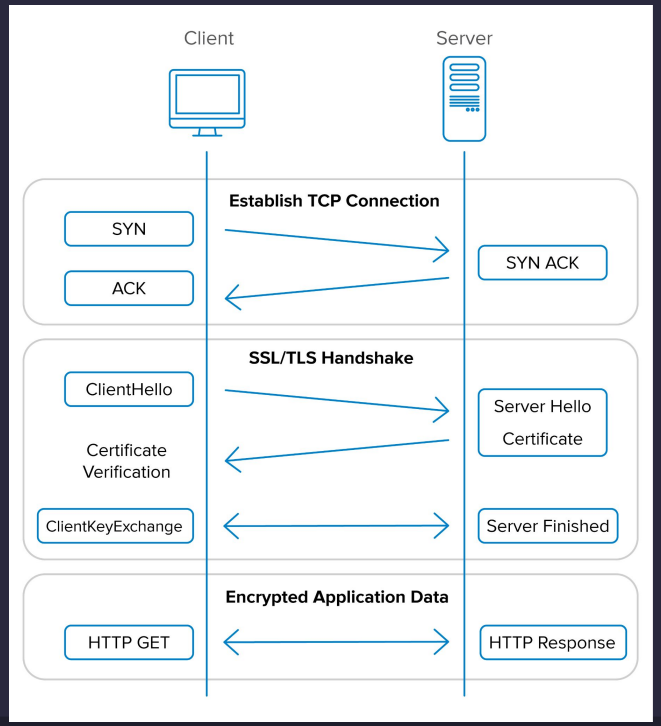




Establishing a SSL/TLS

Client private key

Client public key



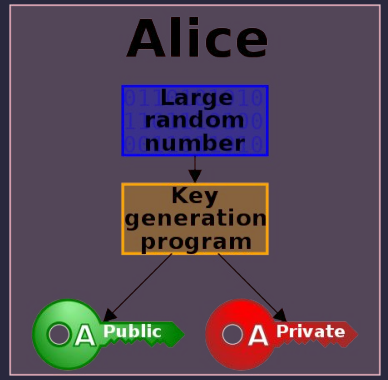
- **Certificate**
- **Key Exchange**
- **Asymmetric cryptography**

Server private key

Server public key

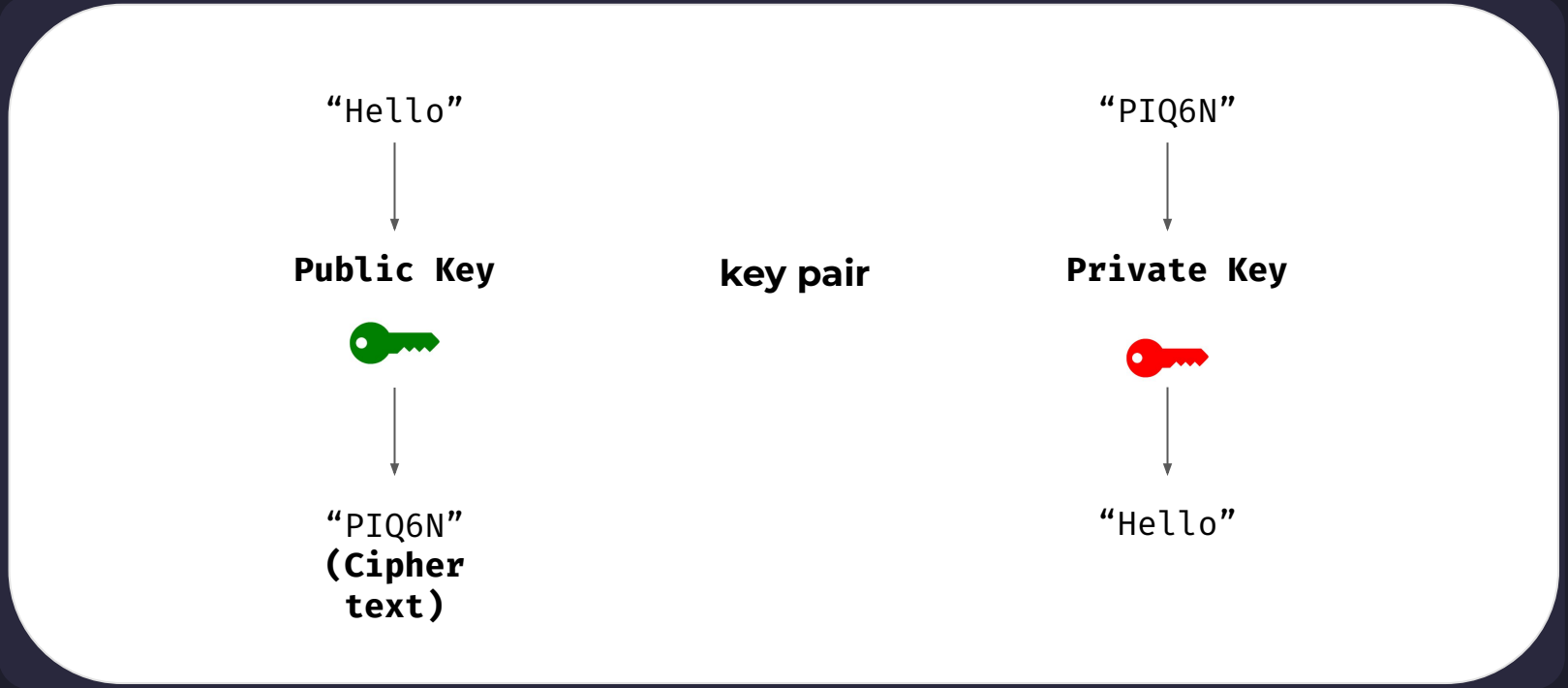
Asymmetric cryptography

In a public-key encryption system, anyone with a **public key can encrypt a message**, yielding **a ciphertext**, but only those who know the corresponding **private key can decrypt the ciphertext** to obtain the **original message.**

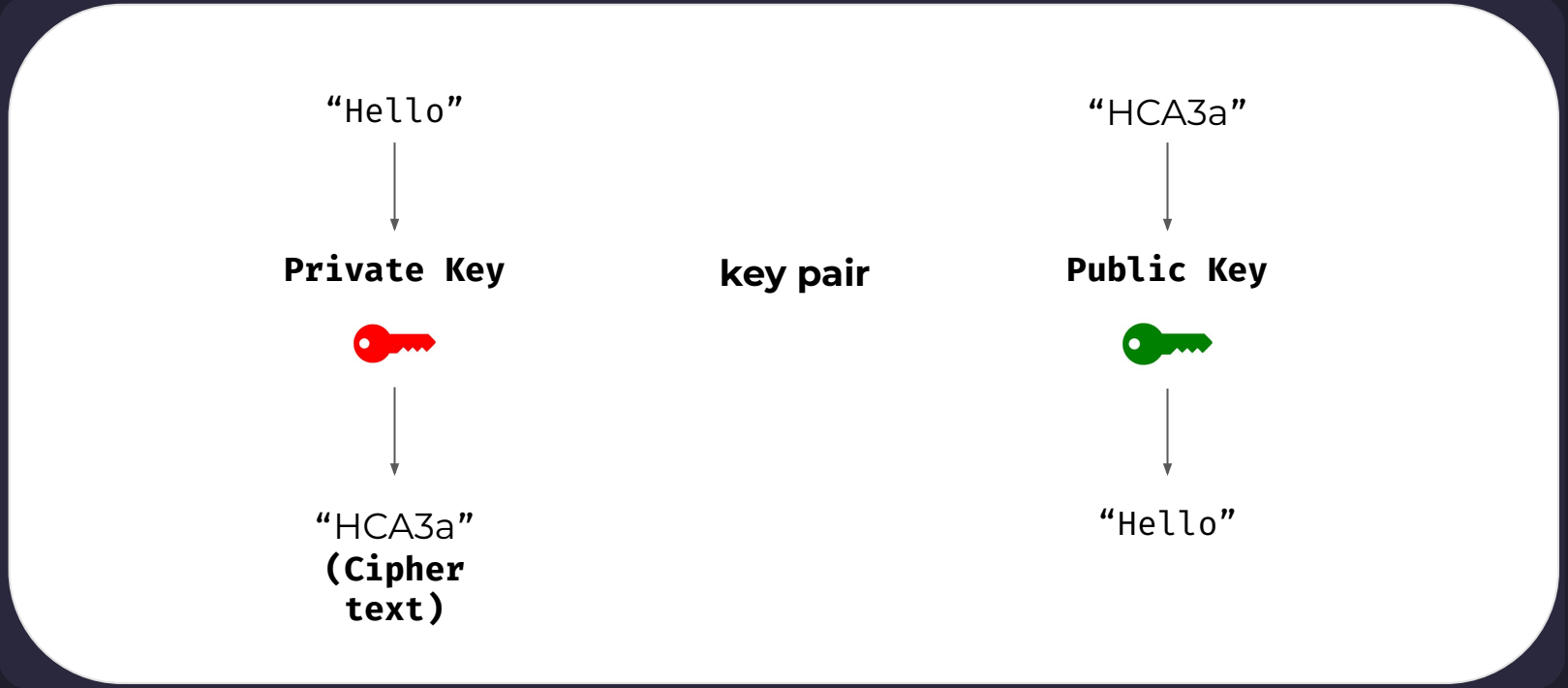


Asymmetric cryptography

Asymmetric cryptography



Asymmetric cryptography



Secure Connection

Secure connection ≠ trusted user

Authentication

Example of Authentication

What’s Authentication?

Authentication is the **process of verifying the identity of a user, system or application**. It's a means to **ensure** that the entity **requesting access** is **who** or **what it claims to be**

Type of Authentication

- 🔑 🔢 Something you know - **passwords, PINs**, and answers
- 📱 💳 Something you have - **smart cards**, hardware tokens, or a smartphone **OTP**
- 👁️ 👍 Something you are - **fingerprints**, retina scans, or **facial recognition**
- 🌍 📍 Somewhere you are - authentication based on **geolocation**
- ✍️ ⌨️ Something you do - gesture-based mobile **unlock patterns** or typing rhythm

Authentication method

- API Keys
- Basic Authentication
- Token-Based Authentication
  - OAuth
  - JWT (JSON Web Tokens)
- etc.

Basic Authentication

1. The user’s username and password are combined with a colon.
2. The resulting string is base64 encoded.

Authorization: Basic <credentials>

API KEY

- As a query parameter in the URL
  (e.g., https://api.example.com/data?apikey=YOUR_API_KEY)

- In the request header
  (e.g., Authorization: Bearer YOUR_API_KEY)

- As part of the request body in POST requests.

JWT

```
Authorization: Bearer <jwt_token>
```

Authorization

Example of Authorization

What’s Authorization?

Authorization refers to the process of determining **what actions a verified user or system** is **allowed to perform after** they have been **authenticated.**

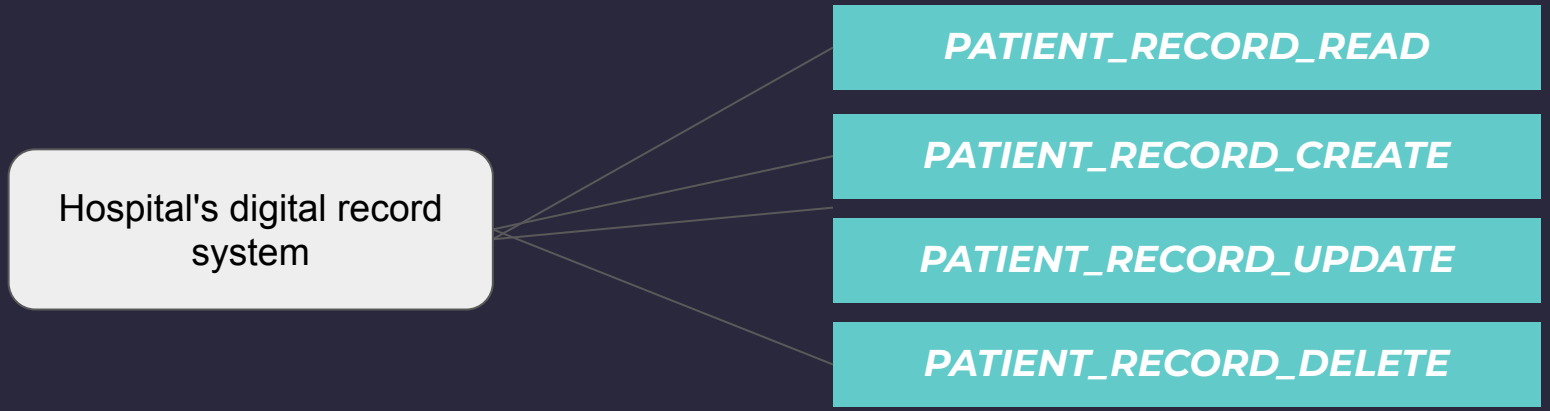
Components of Authorization

- **Permissions**: These define the **specific actions** or **operations** a user can perform. For instance, read, write, edit, delete, etc.

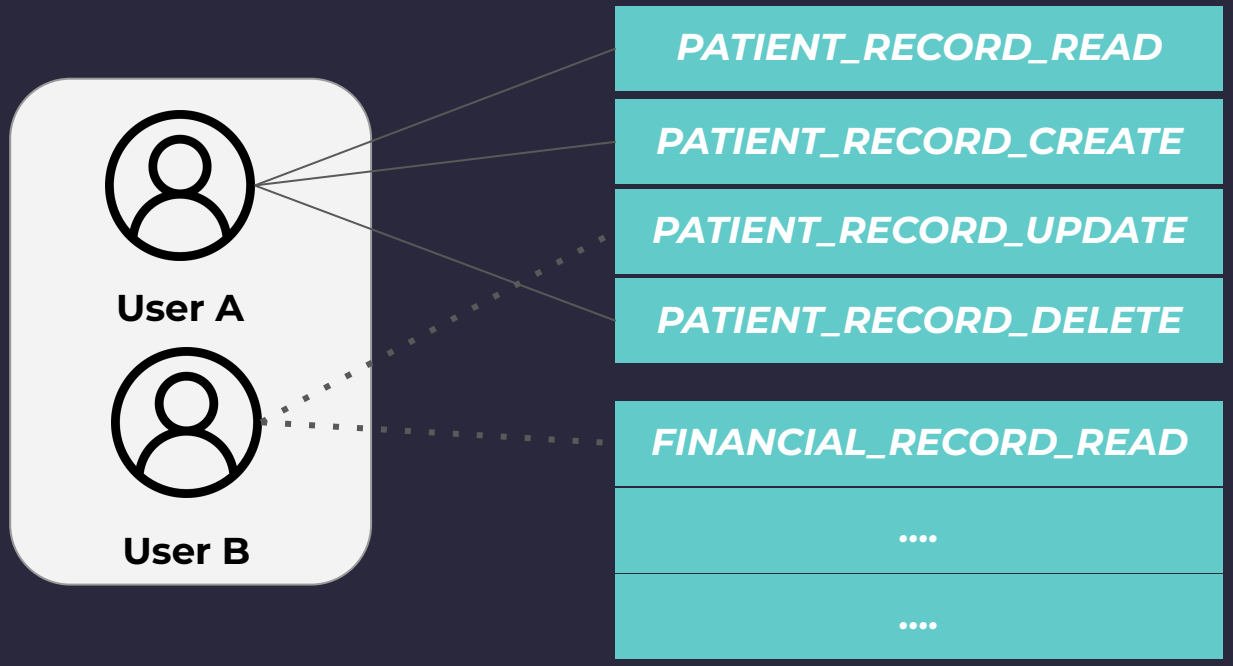
- **Roles**: A **collection of permissions**. Instead of assigning individual permissions to users, roles are assigned. A user with the "admin" role might have permissions to create, edit, and delete records, while a "viewer" might only have the read permission.

- **Policies**: High-level definitions that govern access. For instance, a policy might define that only employees in the HR department can access employee salary details.

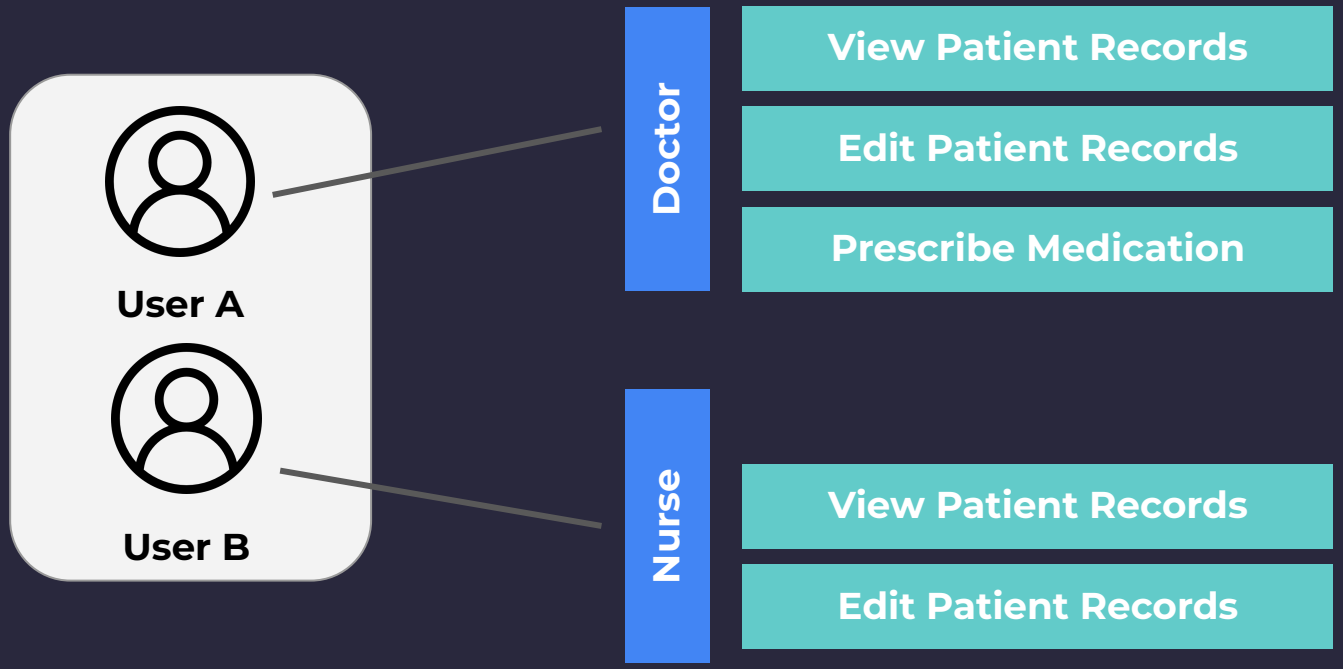
Permission-based



Permission-based



Role-based


User A
User B
Doctor
View Patient Records
Edit Patient Records
Prescribe Medication
Nurse
View Patient Records
Edit Patient Records

JSON Web Token (JWT)

The Key to Modern Web Security

What's JWT?

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for **securely transmitting information between parties** as a **JSON object.** This information can be **verified** and **trusted** because it is **digitally signed**.

JWTs can be **signed** using a secret **(with the HMAC algorithm)** or a **public/private key pair** using RSA or ECDSA.

Why JWT?

- Stateless Authentication - Information contained within the JWT
- Scalability - no needing to share session data
- Decentralized Issuance - Tokens can be issued and verified by multiple parties or services
- Fine-grained Authorization
- Short-lived & Expiry
- Standardized

Imagination of JWT

- **Single package** - single string
- **Contain information inside** - payload
- **Can verify** if **information** inside if it got changed
- Have **expiry time**

JWT Structure

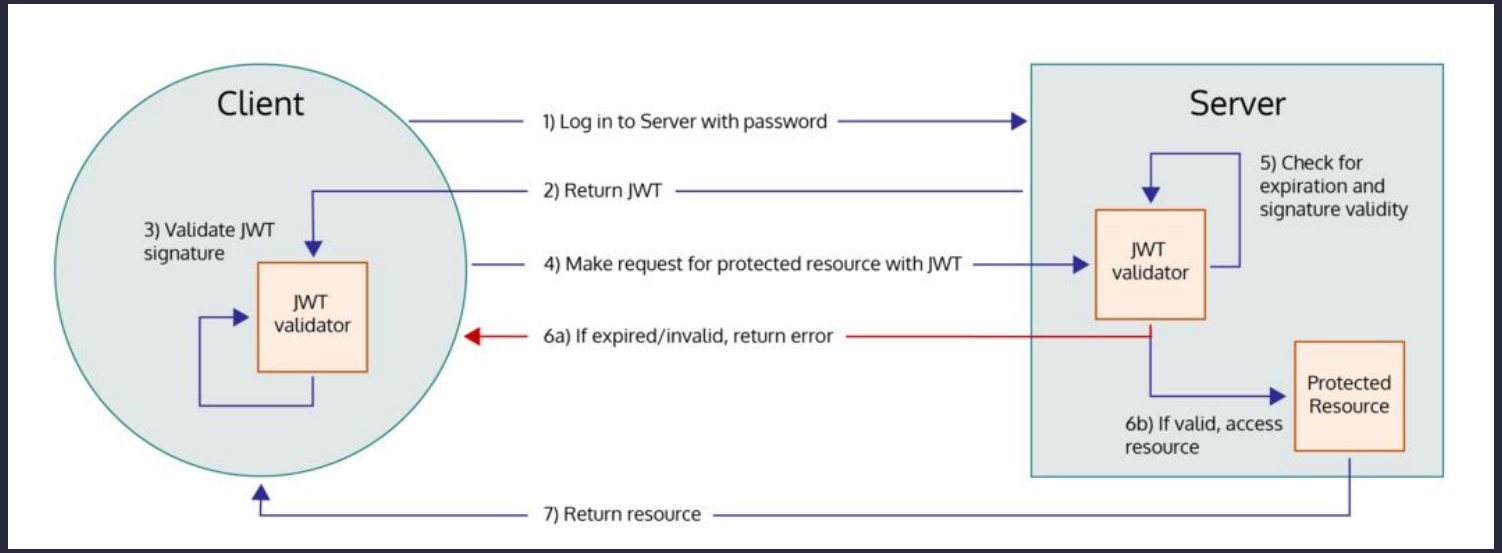
xxxxx.yyyyy.zzzzz

Header: The header typically consists of two parts: the type of the token (JWT) and the signing **algorithm** being used, such as HMAC SHA256 or RSA.

Payload: This is where the actual **claims** are located. Claims are statements about an entity (typically, the user) and additional data.

Signature: To create the **signature** part, you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign it.

Understanding JWT



JWT Validation

xxxxx.yyyyy.zzzzz

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

xxxxx.yyyyy.zzzzz

Implementing JWT - Access Token

- Generate JWT
- Validation
- Middleware

JWT - Consideration & Practices

- **Transmission**: JWTs should **always** be sent **over encrypted channels**, such as **HTTPS**.

- **Storage**: If stored on the **client-side**, **JWTs** should be **stored securely** (e.g., using Web Storage with caution due to potential XSS attacks).

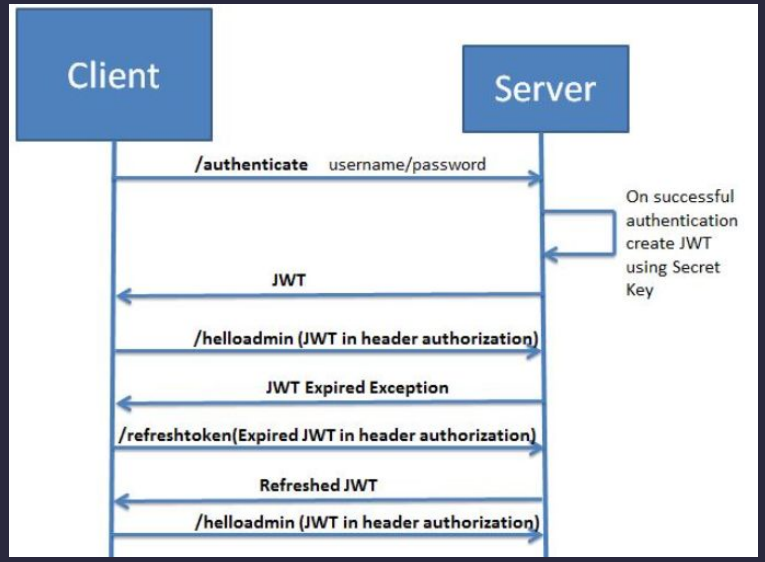
- **Secret**: If symmetric **signing algorithms** like HMAC SHA256 are used, the **secret key must be protected**, as anyone with the key can create and verify JWTs. Asymmetric algorithms (e.g., RSA) have a private key (for signing) and a public key (for verification), offering a separation of capabilities.

- **Expiration**: JWTs typically have an expiration claim (exp) that determines how long they are valid. **Short-lived tokens** are more secure because they limit the potential damage of token leaks.

JWT - Refresh Token

accessToken (short-lived)

refreshToken (long-lived)

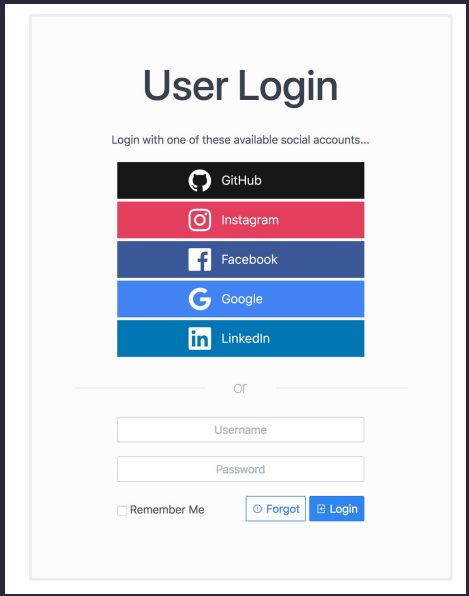


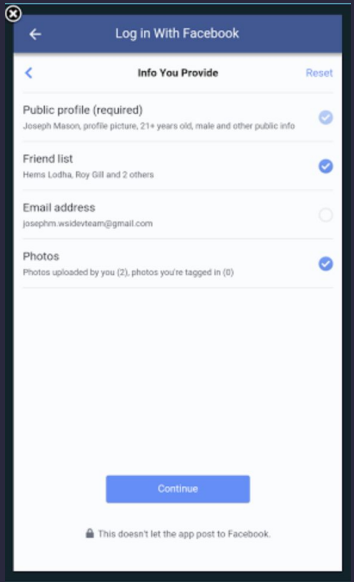
refreshToken (long-lived)

accessToken (short-lived)

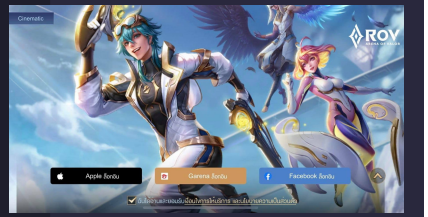
OAUTH 2.0

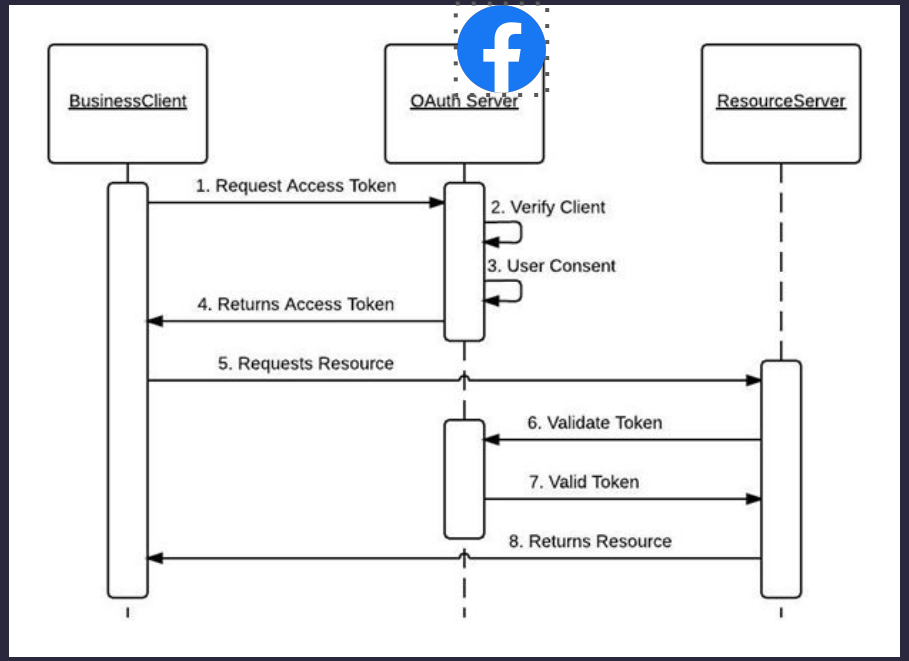
OAUTH 2.0


User Login
Login with one of these available social accounts...
GitHub
Instagram
Facebook
Google
LinkedIn
or
Username
Password
Remember Me
Forgot
Login


Log in With Facebook
Info You Provide
Reset
Public profile (required)
Joseph Mason, profile picture, 21+ years old, male and other public info
Friend list
Hems Lodha, Roy Gill and 2 others
Email address
josephm.wsidevteam@gmail.com
Photos
Photos uploaded by you (2), photos you're tagged in (0)
Continue
This doesn't let the app post to Facebook.

OAUTH 2.0





Resource Receive: email,name etc.

OAUTH 2.0 - Benefits

User Security: OAuth allows third-party applications to access user data without requiring the user to share their credentials. Instead of sharing passwords, tokens are shared.

Simplified User Experience: Users can leverage their existing accounts (like Google, Facebook, or Twitter) to sign into various applications without needing to create new accounts or remember additional passwords.

OAUTH 2.0 - Benefits

Flexible Access Control: OAuth's token mechanism allows for detailed specifications about what third-party applications can and can't do. This is managed through "scopes" in the token. For instance, an application might gain access to read a user's email and cannot post on their behalf.

Scalability for Developers: Developers can delegate the responsibility of secure authentication to large providers that specialize in security. This means less custom code for handling user credentials and potentially fewer security vulnerabilities.

etc.

API Security Practices

Use Strong authentication mechanisms

- Oauth 2.0
- JWT (short-lived token)
- Basic Authentication/API Key + other security measure

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

Payment Gateway

🔑Pub

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

Register merchantID, APIKEY

Payment Gateway

🔑Pub

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

Register merchantID, APIKEY

Payment Gateway

🔑Pub

APIKEY_1: 🔑PubKey

…Other key…

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

Download Payment GW Pub

🔑Pub

Payment Gateway

🔑Pub

APIKEY_1: 🔑PubKey

…Other key…

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

🔑Pub

Payment Gateway

🔑Pub

APIKEY_1: 🔑PubKey

…Other key…

API KEY + Key Pair

Merchant

🔐Private Key

🔑Pub Key

🔑Pub

Merchant want to send data?

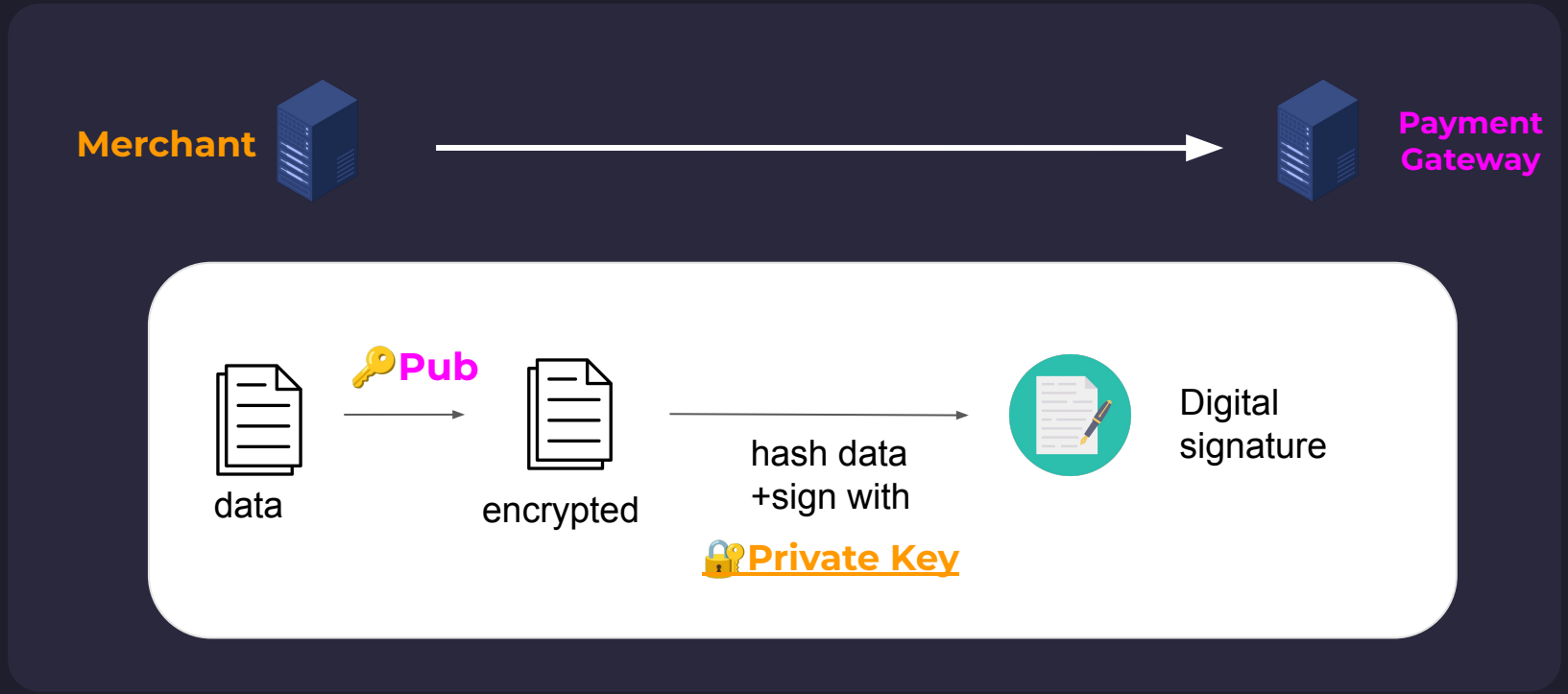
Payment Gateway

🔑Pub

APIKEY_1: 🔑PubKey

…Other key…

API KEY + Key Pair



API KEY + Key Pair

```
POST /initiatePayment
Headers:
    Content-Type: application/json
    Authorization: Bearer API_KEY_PROVIDED_TO_MERCHANT

Body:
{
    "merchantID": "MID12345",
    "encryptedData": "abc123xyz...",
    "digitalSignature": "def456uvw..."
}
```

1. check authentication
2. matching merchant public key
3. Decrypt data with **private key**
4. Hash data from (3)
5. Decrypt digital signature with **merchant public key** from (2) and get hashed data
6. Compare (4), (5)

Rate Limiting

Rate limiting is the process of **controlling the frequency** at which a user can access a resource within a given time frame.

pros.

1. Prevents resource exhaustion and ensures fair access.
2. Protects against Denial-of-Service (DoS) attacks.
3. Maintains system stability and performance.

example.

1. API calls: Limiting an application to 1000 requests per minute.
2. Login attempts: Allowing only 5 login attempts every 10 minutes.
3. Data uploads: Allowing only 500MB of data upload every hour.

Input Validation

Input validation is the process of **verifying** and **sanitizing** user-provided **data before processing** it.

pros.
1. Protects against malicious input, such as **SQL injection** and script injection.
2. **Ensures data integrity** and **prevents errors in processing**.
3. Enhances user experience by providing immediate **feedback on invalid data**.

example.
1. Email format: Ensuring that user-provided emails match the pattern user@example.com.
2. Date input: Making sure a user enters a valid date format, such as MM/DD/YYYY.

Data Protection

Data protection involves implementing measures to safeguard sensitive information from unauthorized access, breaches, and potential threats.

example.
1. Password Storage: **Passwords** should be **hashed** using **strong cryptographic** algorithms **before** being **stored** in databases like **bcrypt** or Argon2.

2. Data Encryption: **Sensitive data**, such as credit card numbers, should be **encrypted** both at rest (when stored) and in transit (when transmitted)..

Logging and Monitoring

Logging and Monitoring involve the systematic collection, storage, and analysis of system activities to detect, diagnose, and respond to potential issues or threats.

example.
1. User Activity Logs: Tracking user activities such as **login attempts, data access, and modifications.**

2. Intrusion Detection: **Using monitoring tools** to **identify unusual patterns** or unauthorized access attempts.

CORS (Cross-Origin Resource Sharing)

CORS is a security feature implemented by web browsers that controls how web pages in one domain can request resources from another domain.

```
Access to XMLHttpRequest at 'http://localhost:5000/global_config'   step1:1
from origin 'http://localhost:8080' has been blocked by CORS policy:
Response to preflight request doesn't pass access control check: No 'Access-
Control-Allow-Origin' header is present on the requested resource.
```

avoid using the wildcard (*)

API Access: A weather app on **weatherapp.com** requests data from an **API** on weatherdata.com using AJAX. With CORS, **weatherdata.com can allow** this cross-origin request.

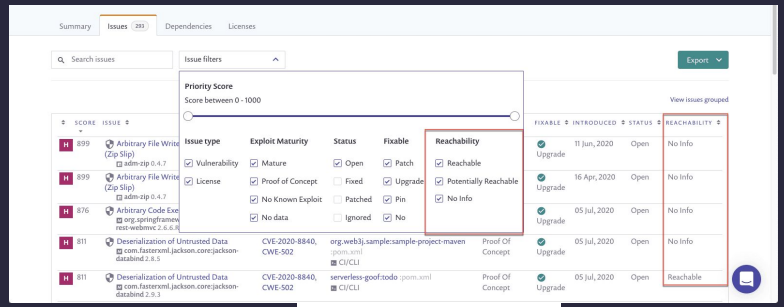
Error Handling

Provide generic error messages. Don't reveal too much information, such as **stack traces** or **detailed database errors**, as they can provide attackers with clues

```
java.sql.SQLException: No suitable driver found for jdbc:mysql://db-server:3306/sensitiveDB?user=admin&password=secret123
```

Updating Dependencies

Dependencies are **external libraries** or **modules** that software relies on to function. Not updating them means using **outdated versions** that **might have known vulnerabilities** or compatibility issues.



snyk

govulncheck

Govulncheck only reads binaries compiled with Go 1.18 and later.

Conclusion

- Use Strong authentication mechanisms
- Rate Limiting
- Input Validation
- Logging and Monitoring
- CORS (Cross-Origin Resource Sharing)
- Error Handling