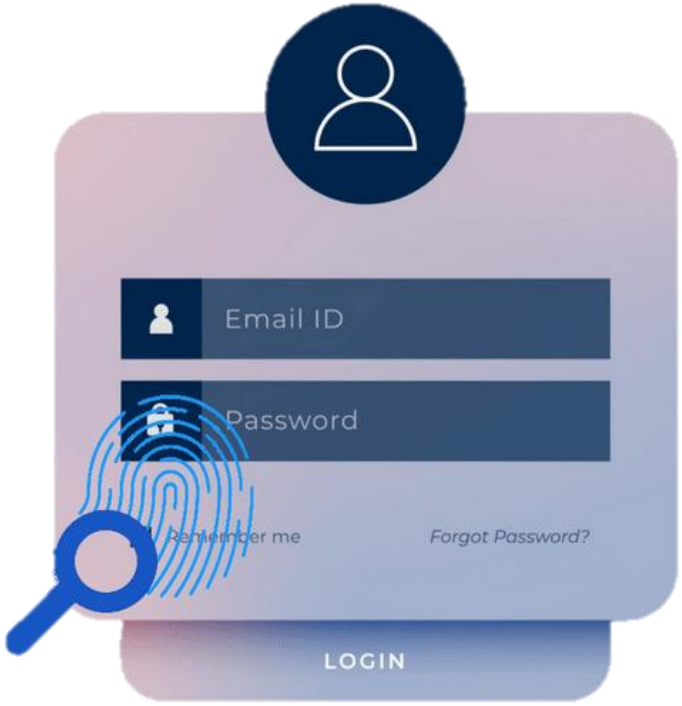
AUTHENTICATION: METHODS, PROTOCOLS, AND STRATEGIES

MOHAMMAD JAWAD KADHIM

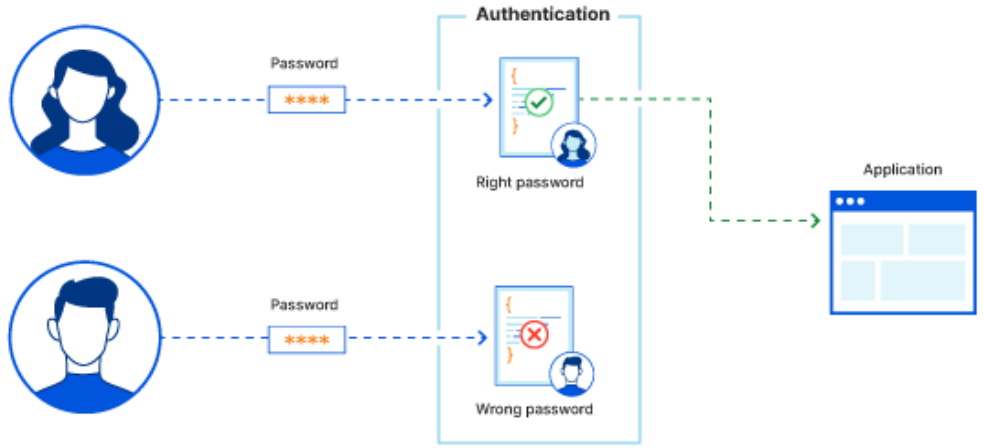
WHAT IS AUTHENTICATION?

- Authentication is the process of determining if the person or entity accessing a computing system really is who they claim to be.
- Authentication systems make a binary decision.
- They allow or deny access based on credentials or other proof provided by those requesting access.
- Authentication typically works together with authorization systems, which determine what type or level of access a user should have.



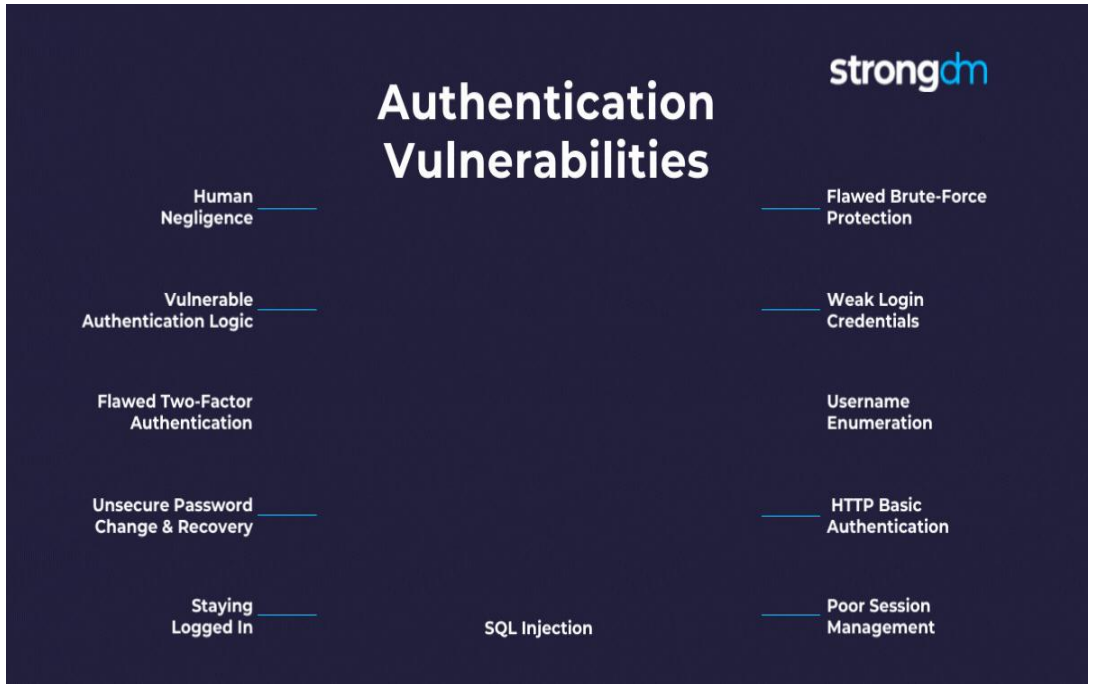
HOW DOES AUTHENTICATION WORK?

- In a traditional authentication process, the user types in their credentials, such as a username and a password. The authentication system queries a user directory, which is either stored in the local operating system or on an authentication server. If the credentials match, the user is allowed to access the system. In the second stage, permissions assigned to users determine what objects or operations they are allowed to access, and other access rights, such as allowed access times and rate limits.



TRADITIONAL AUTHENTICATION CHALLENGES:

- Traditionally, each IT system or application would handle its own authentication. This created a burden on application developers and meant that authentication systems were non-standard, in many cases not secure.
- Applications are increasingly delivered over the web. Today most applications communicate via HTTP and HTTP/S. These are stateless protocols, meaning that in a traditional authentication model, users would have to login to a web application every time they accessed it.
- Password-based authentication is very easily compromised by attackers, and is also inconvenient to users.



WHAT ARE AUTHENTICATION FACTORS?

Knowledge

Possession

Inheritance

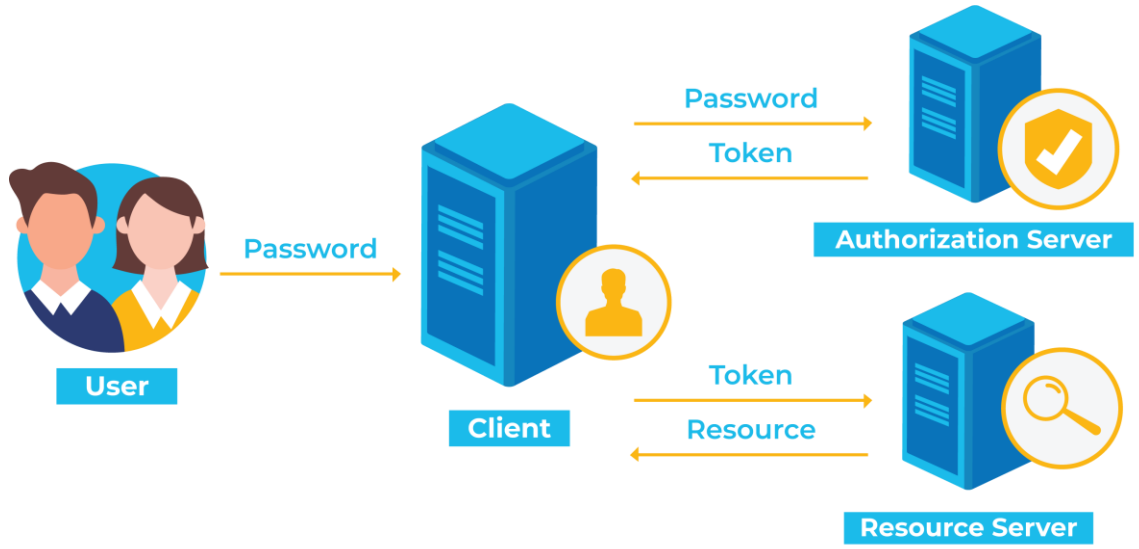
Knowledge Factor: A knowledge factor is a category of credentials that users are expected to know. Common knowledge factors include usernames, passwords, personal identification numbers (PINs), and answers to security questions.

Possession Factor: This factor requires users to provide evidence of possessing physical items, such as SIM cards, smart cards, mobile phones, FIDO2 security keys, and hardware OTP tokens. By checking whether the user has a certain piece of hardware, organizations can make it much more difficult to breach.

Inherence Factor: Inherence is considered the strongest authentication factor because it asks users to confirm their identity by presenting evidence inherent to unique features. Common inherence factor examples include biometrics like fingerprint scans, retina pattern scans, and facial recognition.

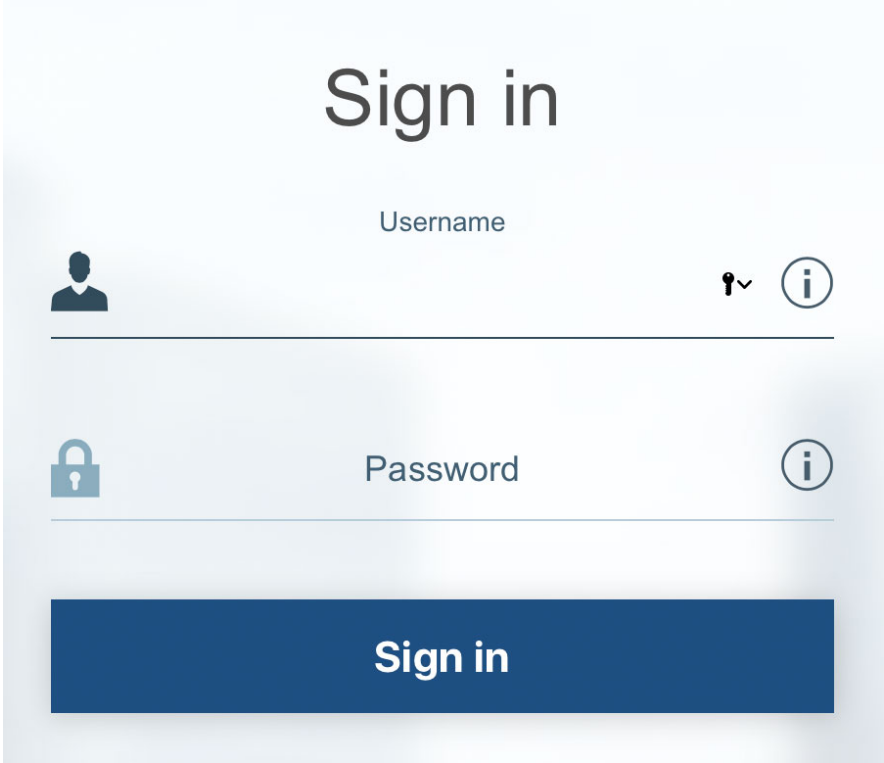
TYPES OF AUTHENTICATION

- **Token authentication** – a commonly-used authentication protocol that allows users to authenticate themselves once and receive a token verifying their identity. As long as the token is valid, the user can access the website or application without signing in again.



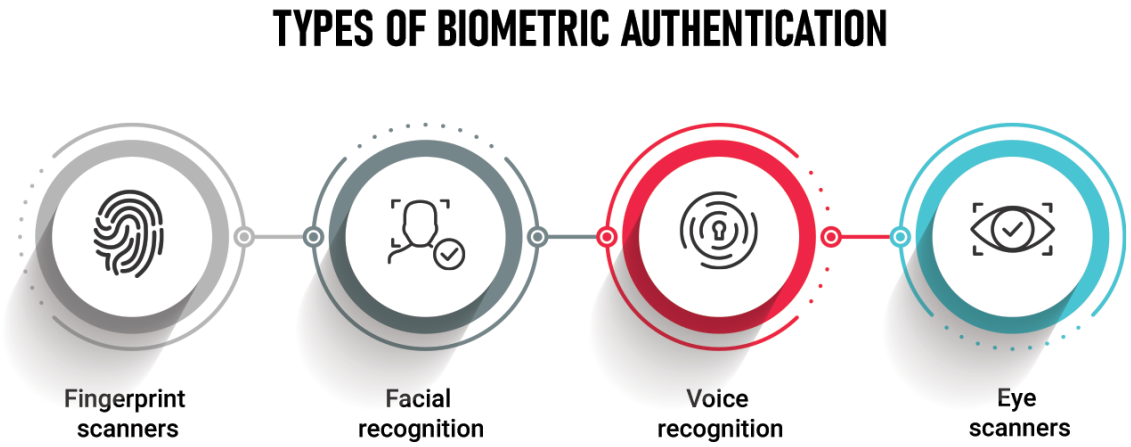
TYPES OF AUTHENTICATION

- **Password authentication** – this requires users to memorize their credentials—typically a username and password in the form of letters, numbers, or special characters. The combination of username and password verifies the user's identity. The more complex the password and the more frequently it is renewed, the more secure the account.



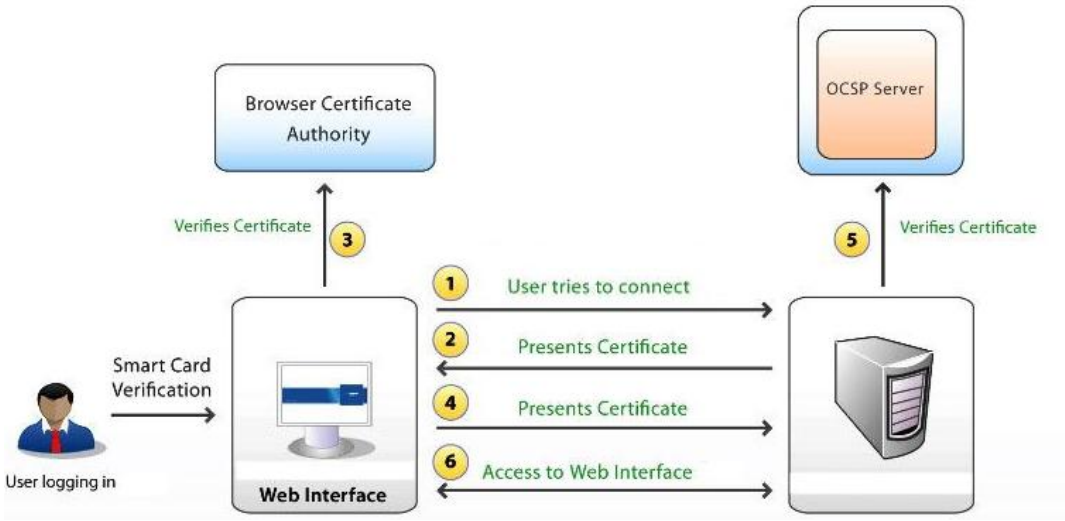
TYPES OF AUTHENTICATION

- **Biometric authentication** – identifies individuals based on their unique biological characteristics. It stores data about an individual—for example, their fingerprint or the shape of their iris—and then compares a real-time reading with this stored data. Biometric authentication is convenient for users and is inherently secure.



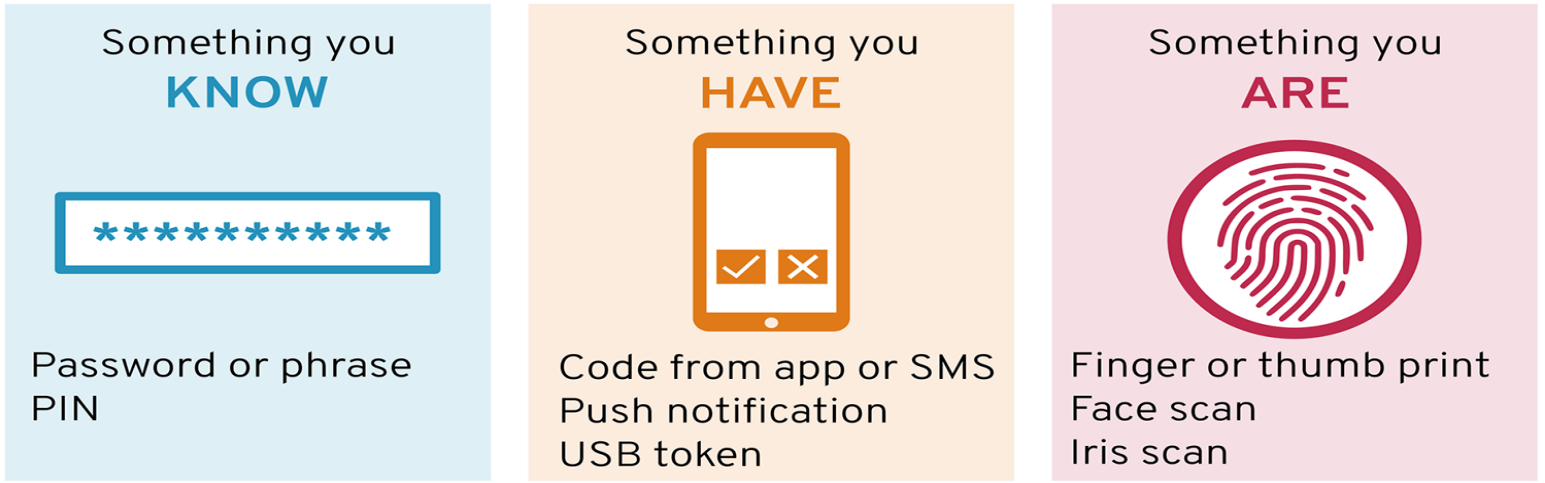
TYPES OF AUTHENTICATION

- **Certificate-based authentication** – uses a digital certificate to identify a user before accessing a resource. Digital certificates are impossible to forge without possessing the private key. It can be used to authenticate a user, device, or service account. Most certificate-based authentication solutions come with a cloud-based management system.



TYPES OF AUTHENTICATION

- **Multi-factor authentication (MFA)** – a combination of two or more authentication factors. These can include any of the above types. Combining several factors significantly improves security and makes it much more difficult for attackers to compromise accounts.



TYPES OF AUTHENTICATION

- **Passwordless authentication** – allows a user to access an app or IT system without entering a password or answering security questions. Instead, users provide other forms of proof such as fingerprints, proximity badges, or codes generated by hardware tokens. This authentication is often combined with MFA and single sign-on (SSO).

AUTHENTICATION PROTOCOLS

- **1. OAuth 2.0: it** Delegated authorization framework for secure API access. Uses access tokens to allow third-party apps to access resources on behalf of users, and it used in social logins (Google, Facebook, GitHub), API authentication.
- **2. OpenID Connect (OIDC): An** Identity authentication protocol built on top of OAuth. Provides an ID token containing user authentication details. Used with Single Sign-On (SSO), user authentication in web apps.
- **3. Security Assertion Markup Language (SAML): is a** XML-based authentication and authorization standard for identity federation. exchanges authentication data between an Identity Provider (IdP) and a Service Provider (SP). Used in enterprise authentication, cloud services, federated identity management.
- **4. JSON Web Token (JWT): is a** Compact, self-contained token format for authentication and secure data exchange. Encodes authentication information in a digitally signed token. Used Stateless authentication, API security, session management.
- **5. Lightweight Directory Access Protocol (LDAP): is a** Protocol for accessing and managing directory-based authentication systems. queries a centralized directory service (e.g., Active Directory) to authenticate users. Used in Enterprise authentication, centralized identity management.
- **6. Kerberos: is a** Secure network authentication protocol using encrypted tickets. Issues time-limited tickets for user authentication without transmitting passwords. Used in windows Active Directory, enterprise authentication.
- **7. Extensible Authentication Protocol (EAP): is a** Framework for authentication in network access environments (e.g., Wi-Fi, VPNs). Supports multiple authentication methods (e.g., certificates, passwords, biometrics). Used in secure wireless authentication (WPA2-Enterprise), VPN access.

AUTHENTICATION PROTOCOLS

Protocol	Purpose	Common Use Cases
OAuth 2.0	Secure delegated authorization	Social logins, API authentication
OpenID Connect (OIDC)	Identity authentication over OAuth 2.0	SSO, user authentication
SAML	Identity federation for authentication	Enterprise SSO, cloud services
JWT	Token-based authentication	API security, stateless authentication
LDAP	Directory-based authentication	Enterprise networks, Active Directory
Kerberos	Secure ticket-based authentication	Windows domains, enterprise authentication
EAP	Flexible authentication for network access	Wi-Fi security, VPN authentication

AUTHENTICATION STRATEGIES

- Basics Authentication
- Session Based Authentication
- Token-Based Authentication
- JWT Authentication
- OAuth - Open Authorization
- Single Sign On (SSO)

BASICS AUTHENTICATION

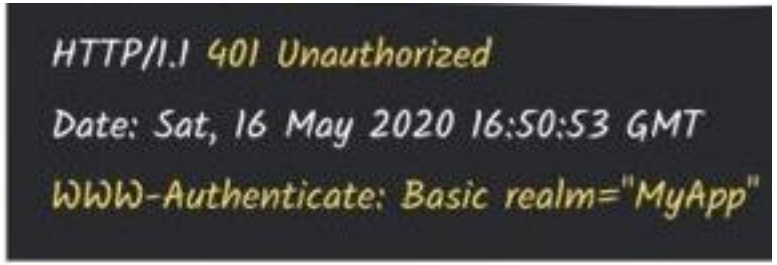
- Mechanism to authenticate access to resources over HTTP
- Credential are sent in the request headers
- Base 64 encoded username & password

Authorization: Basic aGltOm922XJhY2hp2X2lcg==

How does it work

- Step1: client tries to access some protected URL

- Step2: server checks if the request has: **Authorization header with valid username & password?**



Response to the client

- Step 3: browser notices the www-authenticate header in response and present the alert for credentials.
- Step 4: user submit the credentials. Browser encodes then using base64 and sends in the next request. Credentials are sent in the Authorization header in request as follows

```
Authorization: Basic aGltOm92ZXJhY2hpZXZlcg==
```

- Step5: go to step 2. server does the authentication and cycle continue.

Note: basic authentication is not secure unless used with TLS/HTTPS. Because every one can eavesdrop and decode the credentials

SESSION BASED AUTHENTICATION

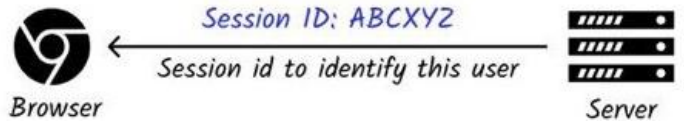
- User is assigned some unique identifiers and this identifiers is stored on the server in memory client sends this session id in all the requests and server uses it to identify the user

- Step1:

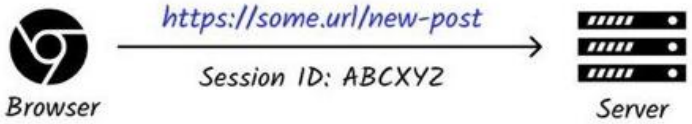
Server stores the generated session id in memory. This could be anywhere e.g. redis, memory, database, filesystem etc.

ABCXYZ now refers to user with id 1224

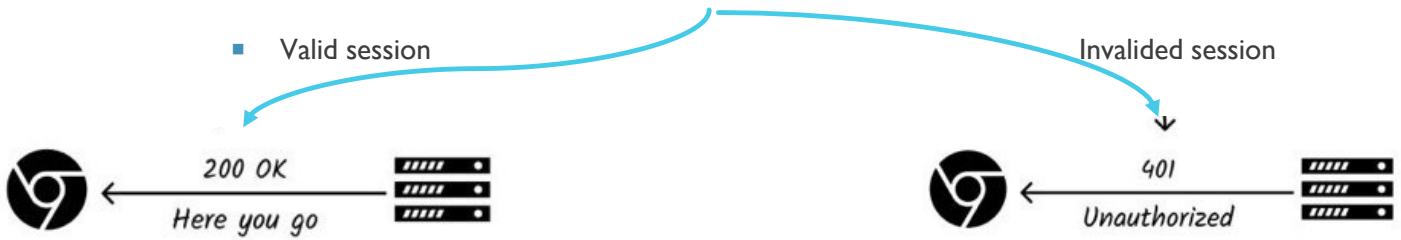
- Step2:



- Step3:



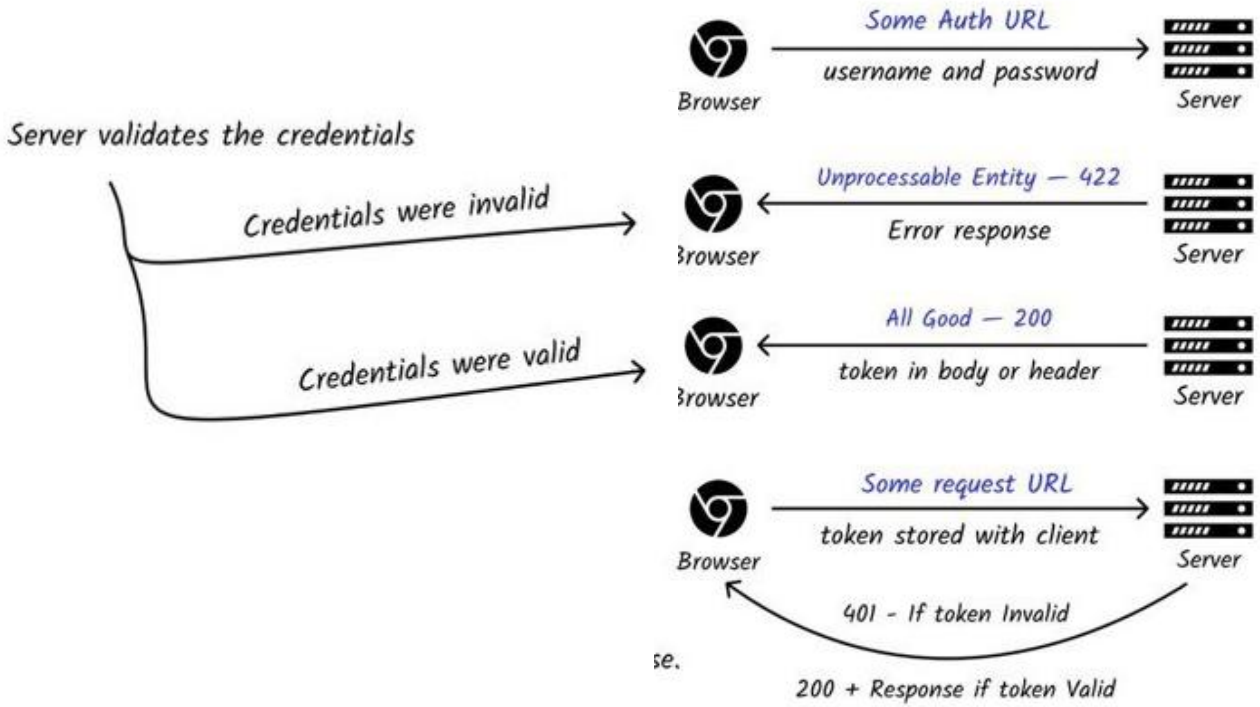
- Valid session
- Invalided session



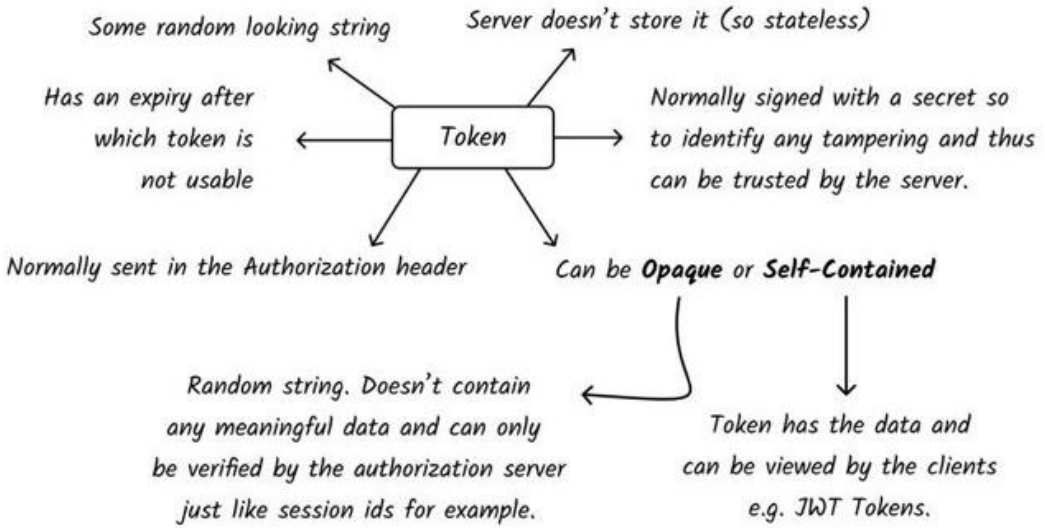
TOKEN BASED AUTHENTICATION

- Unlike the basic authentication where username and password are sent in each request, in token based authentication a token is sent from client to server in each request: some string generated by the server for client to send in each request.

- Client store this token in local storage or cookies and sends in subsequent requests.



CHARACTERISTICS OF TOKEN



Examples of Token Based Authentication Strategies

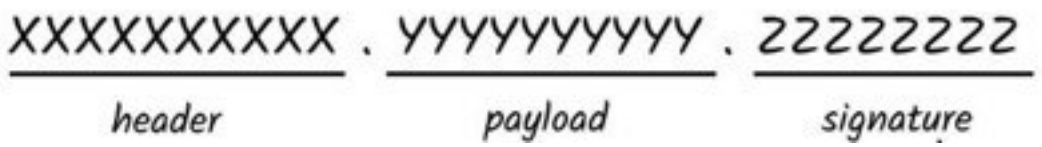
- SWT (Simple Web Tokens)
- JWT (JSON Web Tokens)
- OAuth (Open Authorization)
- SAML (Security Assertions Markup Language)
- OpenID

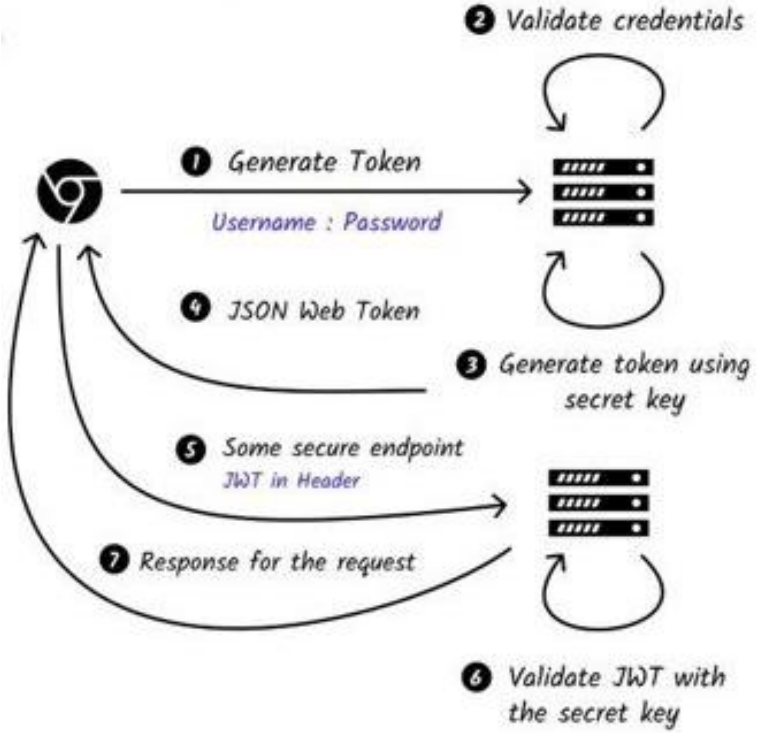
JSON WEB TOKENS (JWT) AUTHENTICATION

Form of token based authentication. Based on open standards (RFC 7519) can be used for authorization as well as secure info exchange

Characteristics of token

- Normal URL-safe string
- Token self contained.
- Anyone can view the content.
- Has three parts separated by a dot





Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIyMzQxMzEiLCJuYW1lIjoiIiwiaWF0IjoxNTE2MjM5MDIyfQ.j5LM4BDvq_c3rRiOAQnBU20UYyt9NHIBmGI3zzuMG9A
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
  "sub": "234131",
  "name": "",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

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

hashing algorithm used for the signature part
e.g. in this case SHA256

Type of token

String generated using base64(ourData)
where ourData is the data that we want
to embed in the token (aka JWT Claims).

String generated by hashing the header,
payload with a secret i.e.

HMACSHA256(header + '.' + payload, 'secret')

held at server and used to generate and verify tokens

Public Claims

Claims which we can define and use for our own data e..g userId, email, above etc.

Private Claims

Names without any meaning to anyone except the consumer and producer of tokens.

Registered Claims

Standard names which are reserved for app usage

iat -> issued at (issuance timestamp)
iss -> issuer (who issued it, app name for example)
sub -> token subject
exp -> expiry time (expiry timestamp)
aud -> token audience (app URL or some string for example)
nbf -> not before (timestamp before which token is not usable)
jti -> unique token identifier (can be used to revoke existing JWT token)

In our sample payload above, we have **exp** and **iat** claims.

OPEN AUTHENTICATION (OAUTH)

- Open protocol for authorization that allow users to share their private resources to a third party
- Example

1. Allowing some application to log you I using twitter
2. Authorizing your applications fronted from your API

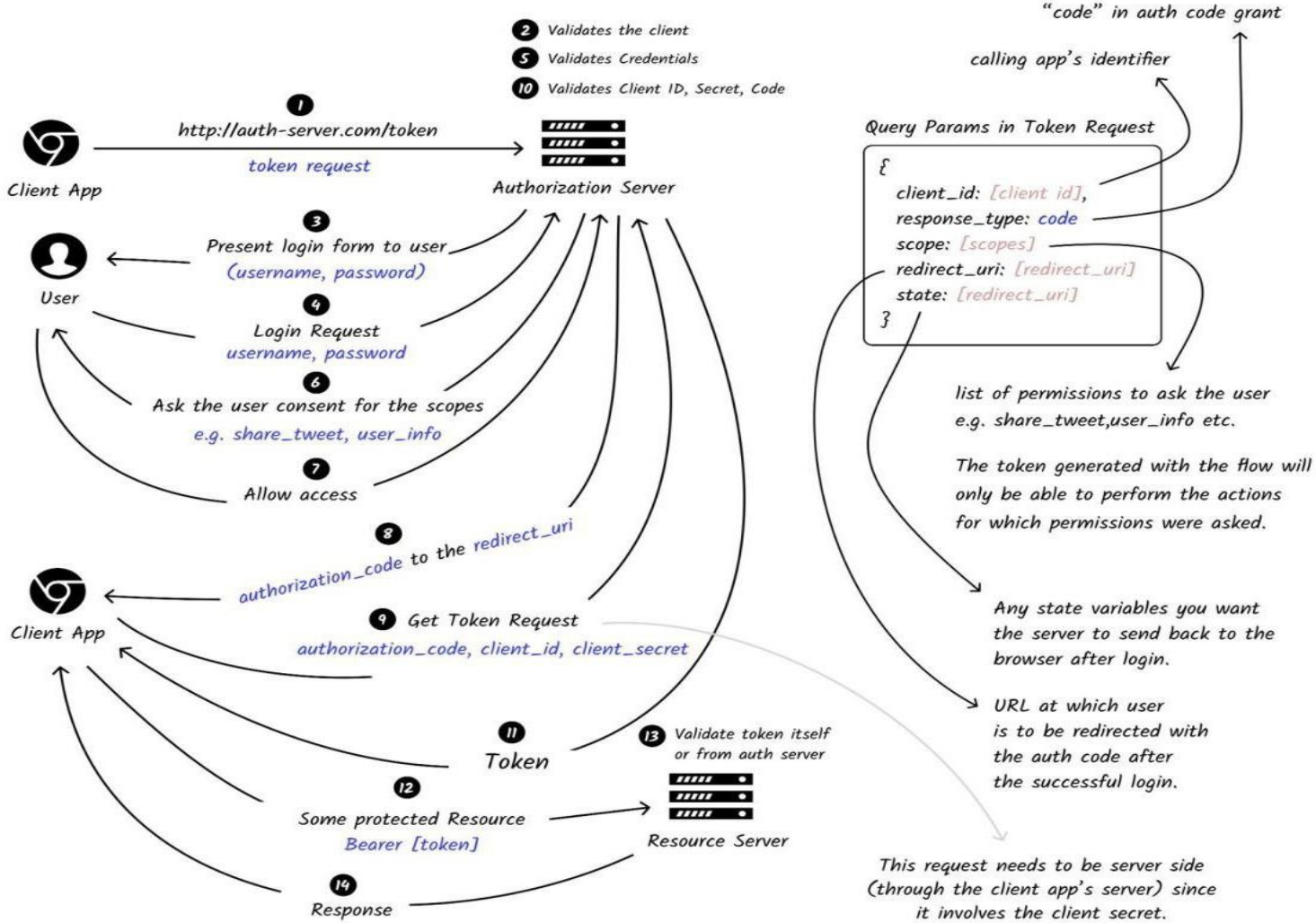
There are two versions — OAuth 1.0 — OAuth 2.0 — OAuth 2.1 (draft)

1 Authorization Code Grant Flow

Terminology

client_id -> Like a username for your app
client_secret -> Like a password for your app
authorization server -> server that handles authentication and authorization
resource server -> server that exposes protected resources

} -> They could both be a single server also



❷ Implicit Grant Flow

Similar to the "Authorization Code Grant Flow" but ***in step 8 there is a token instead of authorization code.***
So no need for 9, 10, and 11 steps. Also in the token request the ***response_type parameter is set to*** `token` ***instead of*** `code`

❸ Client Credential Grant Flow

Similar to the "Authorization Code Grant Flow" but there is no user interaction. It starts at ***step 9 and immediately gets the token.***
So step 9 would be step 1 in this grant flow. Also in the ***request at step 9 there is no*** `authorization_code` ***and an additional parameter of the*** `grant_type` ***set to*** `token`

❹ Password Grant Flow

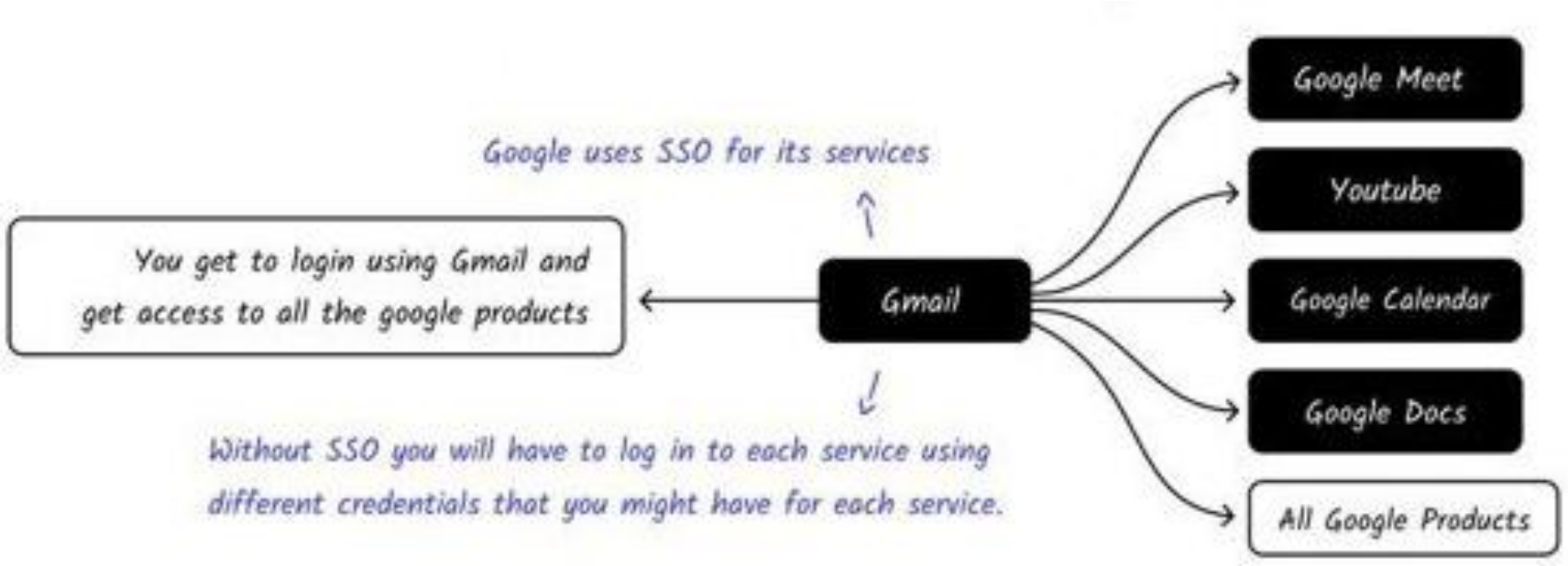
Similar to the "Authorization Code Grant Flow" but user enters the credentials owned by the the authorization server instead of the app.
So ***step 4 to 8 are replaced by the get token request (i.e. step 9) and the credentials are directly sent to the authorization server along with an additional parameter of the*** `grant_type` ***set to*** `password`.

Note

Token response in all grant types is normally accompanied by an expiry date for the token and a refresh token which is used to refresh the token when expired.

SINGLE SIGN ON (SSO)

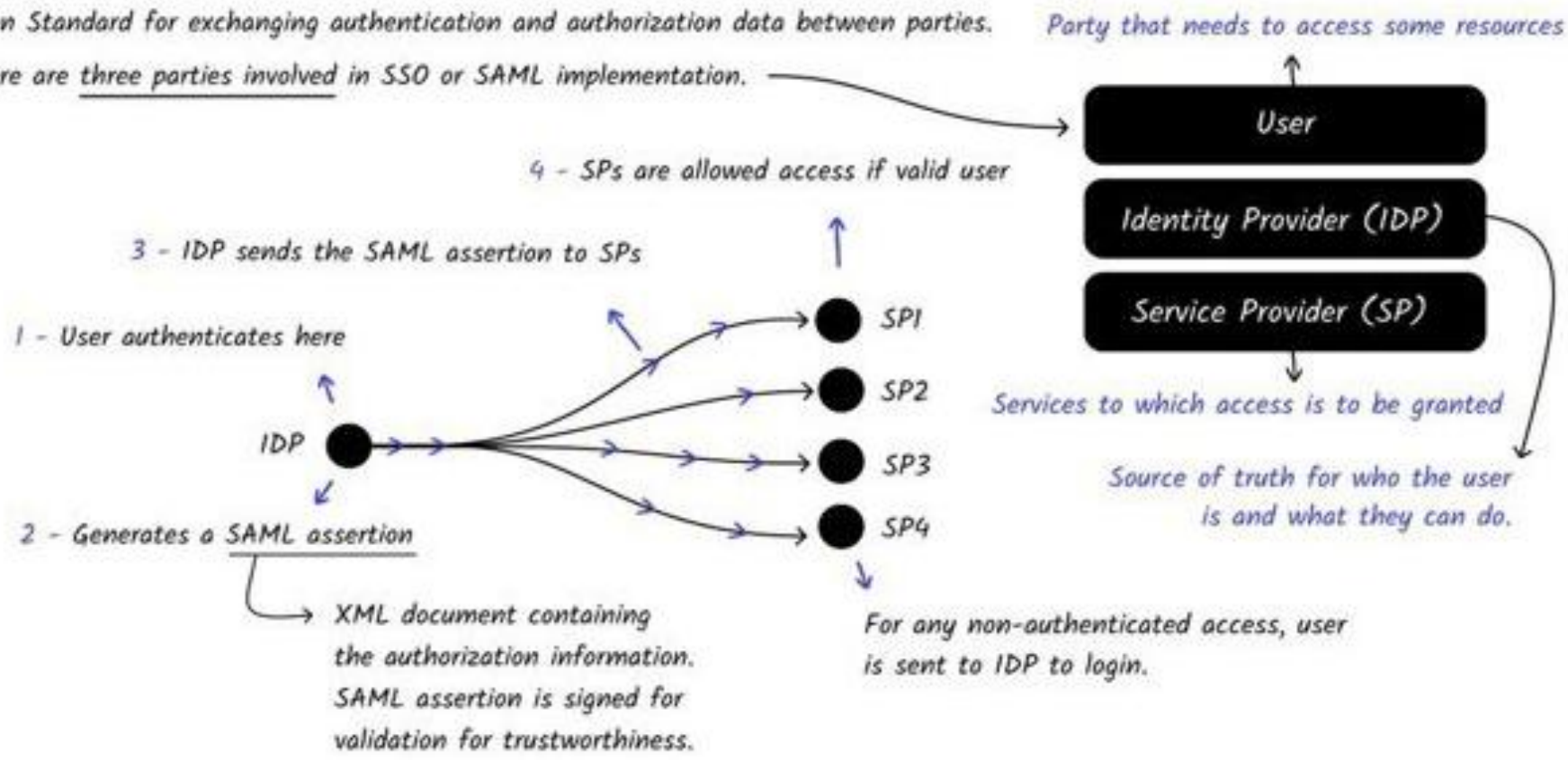
- Authentication strategy that allow user to login with single username and password to several related but independent services

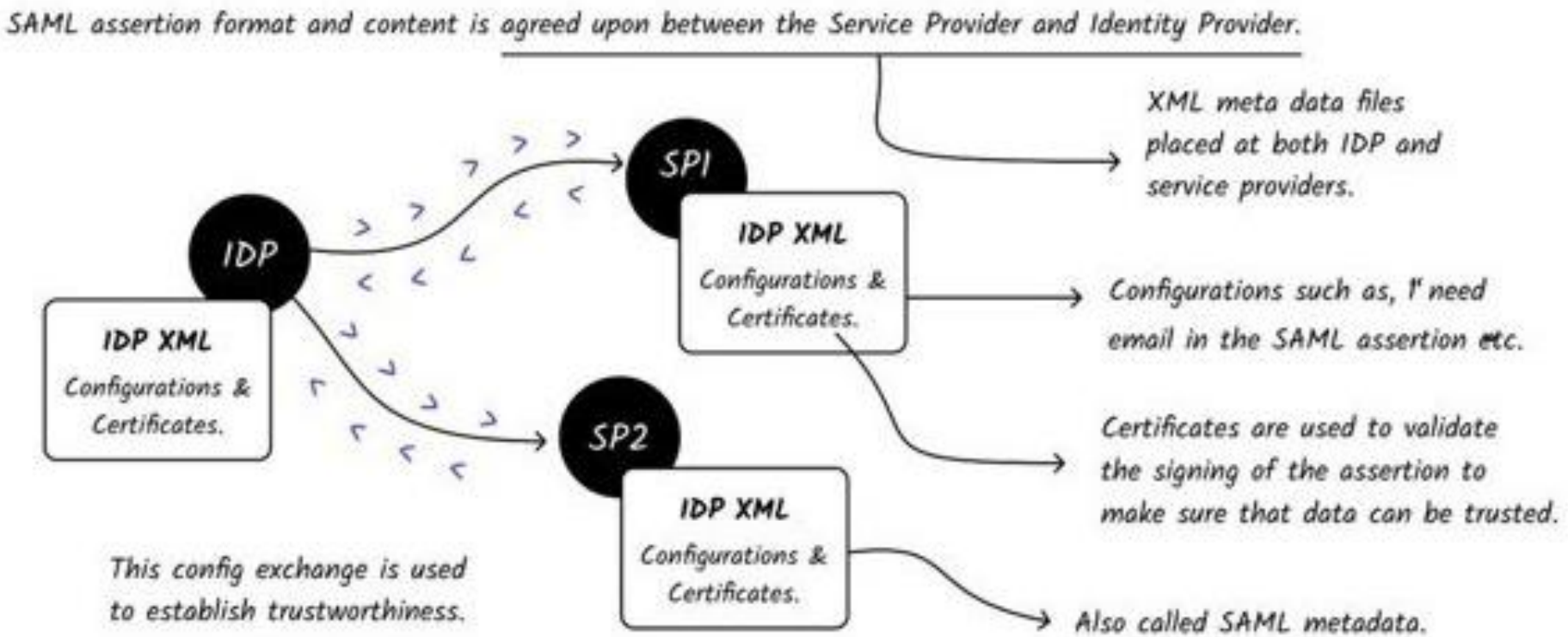


SAML — Security Assertion Markup Language

Open Standard for exchanging authentication and authorization data between parties.

There are three parties involved in SSO or SAML implementation.




SAML assertion format and content is agreed upon between the Service Provider and Identity Provider.
XML meta data files placed at both IDP and service providers.
SP1
IDP
IDP XML
Configurations & Certificates.
Configurations such as, I' need email in the SAML assertion etc.
IDP XML
Configurations & Certificates.
Certificates are used to validate the signing of the assertion to make sure that data can be trusted.
SP2
IDP XML
Configurations & Certificates.
This config exchange is used to establish trustworthiness.
Also called SAML metadata.

How does it work?

There are two main ways of initiating the authentication flow.

- Identity Provider Initiated Flow
- Service Provider Initiated Flow

Identity Provider Initiated Flow

Client requests the IDP to start the authentication flow