**JSON WEB TOKENS (JWT):**

JWT helps in ensuring trust and security in many applications. JWT allow claims also called as attributes, such as user data, which are to be represented in a secure manner.

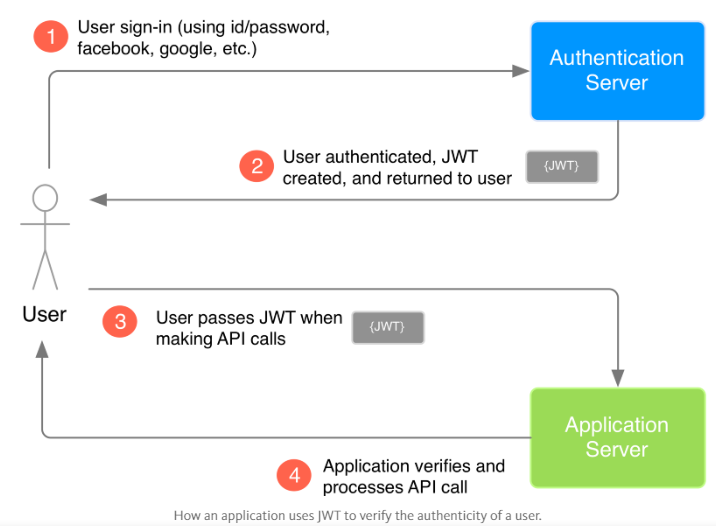
A JSON Web Token (JWT) is a JSON object that is contained in RFC 7519 standard as a secure way to signify a set of information between two parties. A single JSON Web Token has a header, a payload, and a signature.

Format of a token can be represented as:

**header.payload.signature**

To illustrate how and why JWTS are actually used, we will use a 3 entity example. The three entities are:

* Authentication server-It provides the JWT to user
* User-Receives the JWT
* Application server-Communicates between user and application safely



**Anatomy of JWT:**

When the user signs into the respective authentication server using the application’s login system. Unique JWT is created to that user by the authentication server .API calls with JWT attached to them are sent to application upon logged in. The configured application server verifies the creator of JWT. The creator should be the authentication server and that particular authenticated user should be the one who making the API calls.

**How to use JWT?**

**1)Create HEADER**

The header component of the JWT has the data about how the JWT signature is figured.

**Format:**

**{**

**"typ": "JWT",**

**"alg": "HS256"**

**}**

* “typ” key - states the object is JWT
* “alg” key - states the type of algorithm being used to create JWT signature
* HMAC-SHA256 - algorithm, a hashing algorithm that computes the signature using secret key

**2)Create PAYLOAD**

It has the data about user information (“user id”) in the form of claims(attributes). Three types of claims can be used:

* Registered/Reserved claims - Standard claims contained in JWT (“iss”,”sub”,”exp”)
* Userdefined claims – Defined by users, are the two most used claims (“userId”).

**Format:**

**{**

**"userId": "b08f86af-35da-48f2-8fab-cef3904660bd"**

**}**

**3)Create SIGNATURE**

The signature is figured using the following pseudo code:

**Fomat:**

**data = base64urlEncode( header ) + “.” + base64urlEncode( payload )**

**signature = Hash( data, secret );**

* base64url-Encodes the header and the payload
* Period(.)-Joins the encoded strings
* Secret key –Creates the JWT signature using hashing algorithm

**Encoding the HEADER and PAYLOAD:**

// header

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9

// payload

eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ

**Signature obtained looks like:**

// signature

-xN\_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM

**Final JTW looks like:**

// JWT Token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ.-xN\_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM

**How the data is protected using JTW?**

JWT do not hide or obscure the user data. It is just a verification of API calls from authenticated user. The user data in a JWT is not ENCRYPTED but just ENCODED and SIGNED.

* ‘Encoding’-Transforms the data’s structure
* ‘Signing’-verifies the authenticity

**JWT is ENCODING not ENCRYPTION**

**ENCODING-** transform data so that it can be safely consumed by a different type of system.

**ENCRYPTION-** transforms the data in order to keep it secret from others.