ASSIGNMENT NO 02

**JSON Web Token (JWT)**is an open standard that defines a compact ( Because of its size, it can be sent through an URL, POST parameter, or inside an HTTP header. Additionally, due to its size its transmission is fast.) and self-contained (The payload contains all the required information about the user, to avoid querying the database more than once.) way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.

These are some scenarios where JSON Web Tokens are useful:

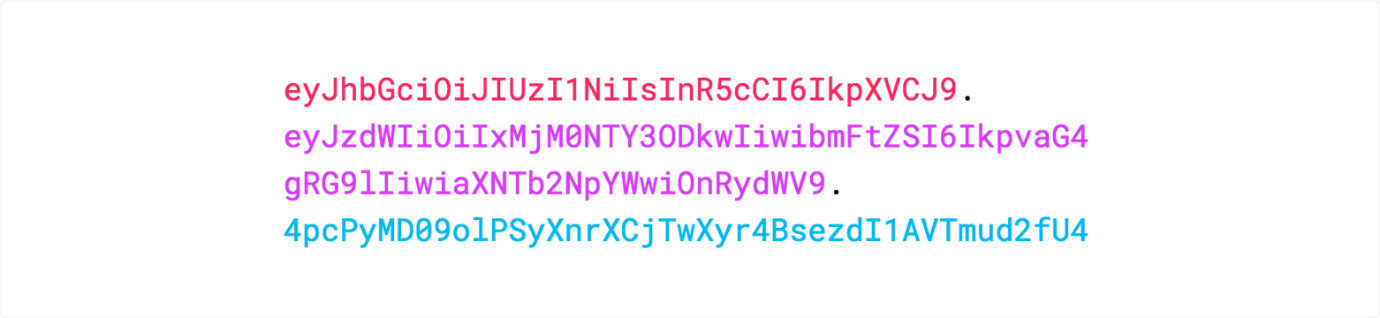
* **Authentication**: This is the typical scenario for using JWT, once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used among systems of different domains.
* **Information Exchange**: JWTs are a good way of securely transmitting information between parties, because as they can be signed, for example using a public/private key pair, you can be sure that the sender is who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn’t changed.

JWTs consist of three parts separated by dots (.), which are:

* **Header**
* **Payload**
* **Signature**

The output is above three Base64 strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact compared to XML-based standards such as SAML.

The following shows a JWT that has the previous header and payload encoded and it is signed with a secret.



**JWT Benefits –**

1. JWT is a stateless authentication mechanism as the user state is never saved in the database. As JWTs are self-contained, all the necessary information is there, reducing the need of going back and forward to the database. With JWT we don't need to query database to authenticate the user for every API call.
2. Protects against CSRF (Cross Site Request Forgery) attacks.
3. JWT is compact. Because of its size, it can be sent through an URL, POST parameter, or inside an HTTP header.
4. You can authorize only the requests you wish to authorize. Cookies are sent for every single request.
5. You can send JWT to any domain. This is especially useful for single page applications that are consuming multiple services that are requiring authorization - so I can have a web app on the domain myapp.com that can make authorized client-side requests to myservice1.com and to myservice2.com. Cookies are bound to a single domain. A cookie created on the domain foo.com can't be read by the domain bar.com.

\*\*\*\* Thanks \*\*\*\*