

JWT

JWT

- JWT (JSON Web Tokens) to use to authenticate users (login, register)
- Authentication. JWT is mainly used for authentication. After a user logs into an application, the application will create a JWT and send it back to the user. Subsequent requests by the user will include the JWT. The token tells the server what routes, services, and resources the user is allowed to access



JWT Format

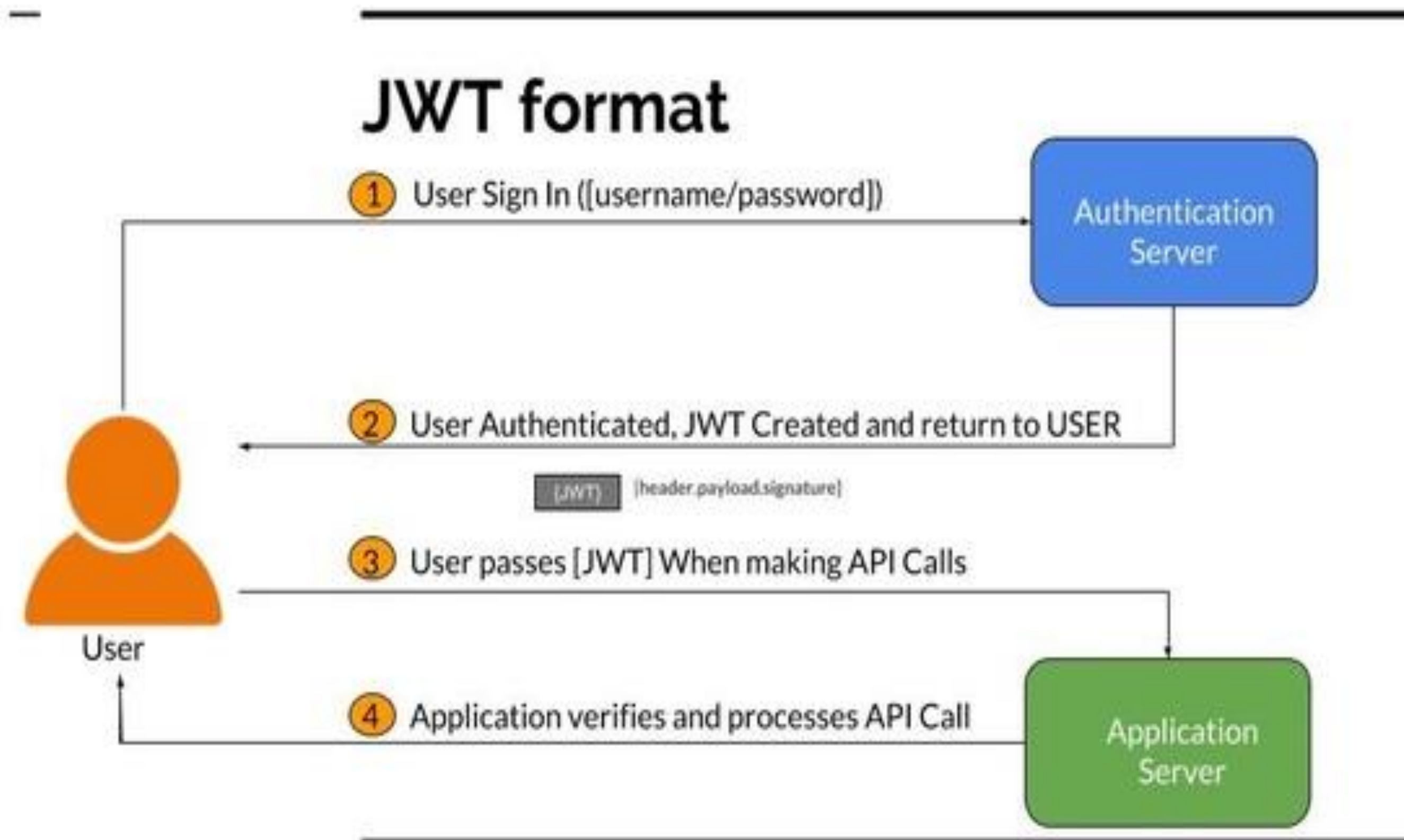


Image source: <https://www.mongodb.com/blog/post/the-modern-application-stack-part-2-using-mongodb-with-nodejs>

What is JWT token structure

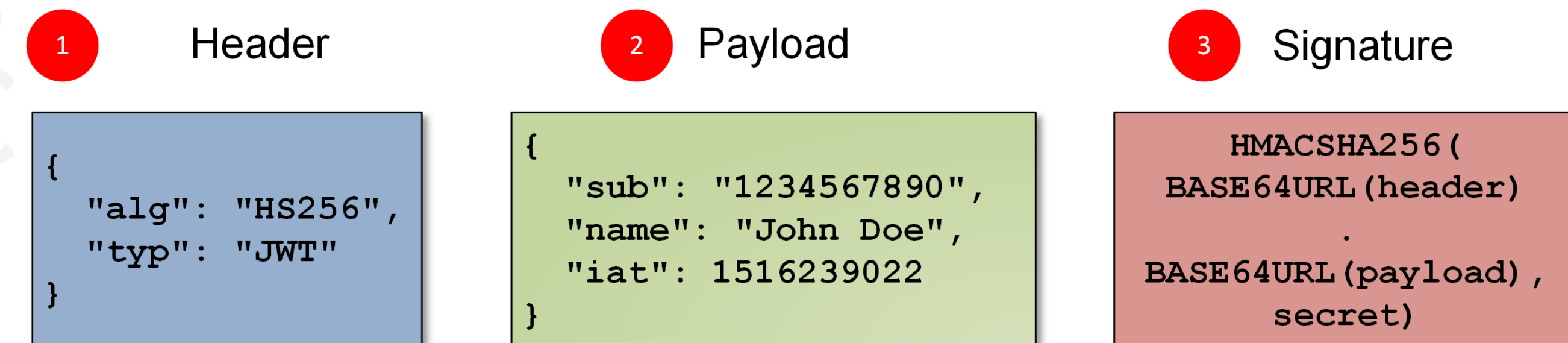
- Header:

- The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC, SHA256 or RSA.

- Payload :

- The second part of the token is the payload, which contains the claims (which has 3 claims), Some of them are: iss (issuer), exp (expiration time), sub (subject)

1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjQ.DIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrZ0ogtVhfEd2o 2 3

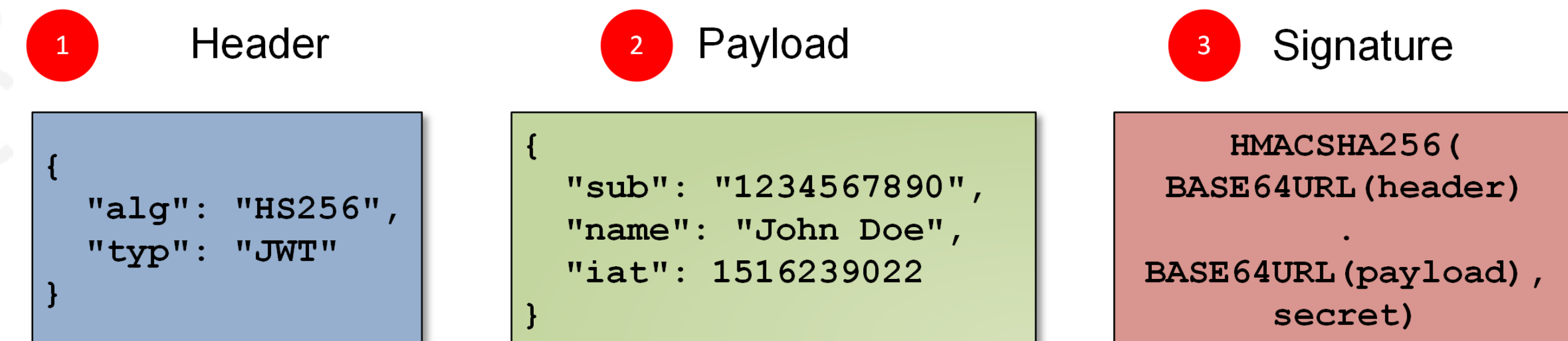


What is JWT token structure

Signature:

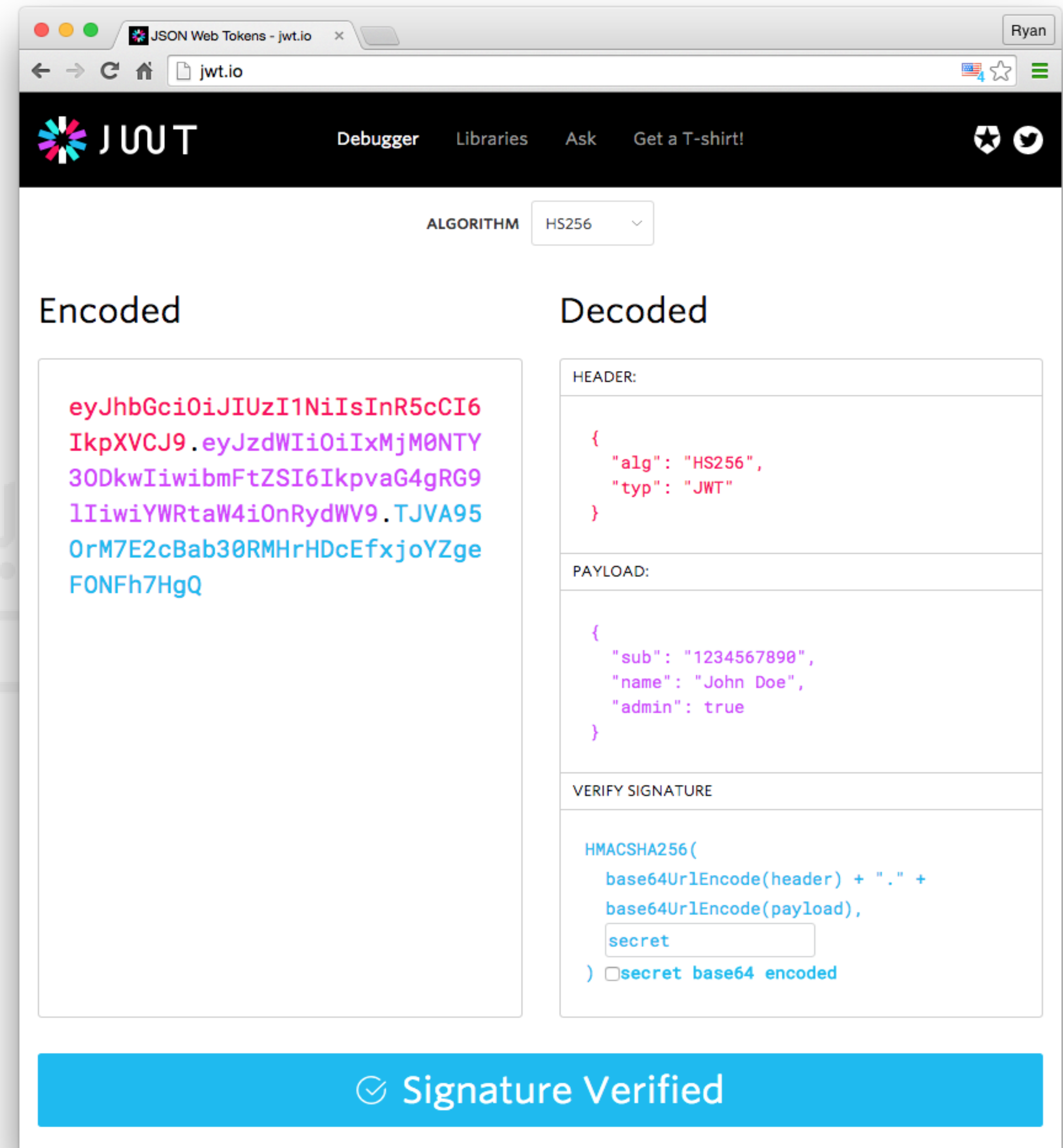
- To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.
- The signature is used to verify the message wasn't changed along the way, and, in the case of tokens signed with a private key, it can also verify that the sender of the JWT is who it says it is.

1
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJlMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjQ.DIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrZ0ogtVhfEd2o
2
3



How the Token Looks

- For more on [JWT](https://jwt.io)



The screenshot shows the JWT.io website interface. The browser address bar displays 'jwt.io'. The website has a dark header with the JWT logo, navigation links (Debugger, Libraries, Ask, Get a T-shirt!), and a user profile 'Ryan'. The main content area shows the 'ALGORITHM' set to 'HS256'. The 'Encoded' section displays a JWT token: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiNjUwMjE2MzQ0InQ.FONFh7HgQ`. The 'Decoded' section shows the token's structure:
HEADER: `{ "alg": "HS256", "typ": "JWT" }`
PAYLOAD: `{ "sub": "1234567890", "name": "John Doe", "admin": true }`
VERIFY SIGNATURE: `HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)` with a checkbox for 'secret base64 encoded'. A blue banner at the bottom states 'Signature Verified'.

install

- npm i jsonwebtoken



Working with JWT in User Model

- We will need to make sure that we encrypt and secure users when they are logged into the app

```
userSchema.methods.generateAuthToken = async function() {  
  const user = this;  
  // get current user  
  const token = jwt.sign({_id: user._id.toString()}, 'secret');  
  // jwt token has to be the same  
  //@@ create the token with the .sign  
  user.tokens = user.tokens.concat({ token }); // current user tokens are stored  
  await user.save(); // save the token to schema  
  return token;  
}  
  
// statics will be model methods  
userSchema.statics.findByCredentials = async (email, password) => {  
  const user = await User.findOne({email})  
  // find the user with the email passed in  
  if (!user) {  
    throw new Error('Unable to login')  
  }  
  const isMatch = await bcrypt.compare(password, user.password);  
  // matches the hashed password to log the user in  
  if (!isMatch) {  
    throw new Error('Unable to login')  
    // log here  
  }  
  return user  
  // if there is a match
```


Working with JWT in User Model

Hashing the password for security

```
1 // Hash the plain password
2 userSchema.pre('save', async function(next) {
3   // before users are saved we will run this method ( we will store passwords here)
4   const user = this // store the current user
5
6   if(user.isModified('password')){
7     // get current users password and hash it
8     user.password = await bcrypt.hash(user.password, 8);
9   }
10  console.log('Just before saving')
11  next() // will save the user when completed
12 })
13 const User = mongoose.model('User', userSchema)
14 module.exports = User;
```

Authentication Middleware

```
1  const jwt = require('jsonwebtoken');
2  const User = require('../models/User')
3  const auth = async (req, res, next) => {
4    try {
5      // validate user will be here
6      const token = req.header('Authorization').replace('Bearer ', '')
7      // Header will contain the Auth details
8      const decoded = jwt.verify(token, 'secret')
9      const user = await User.findOne({_id: decoded._id, 'tokens.token': token});
10     if(!user){
11       throw new Error('No User is found')
12     }
13     req.user = user;
14     req.token = token;
15     // can be used by other functions to be removed
16     // If there are no problem
17     next()
18     console.log(token)
19     console.log(decoded)
20   }
21   catch(e){
22     res.status(401).send({error: 'Please authenticate'})
23   }
24 }
25 module.exports = auth
```



Resources

- Node.js API Authentication With JWT: <https://www.youtube.com/watch?v=7nafaH9SddU>
- What are Json? : <https://www.freecodecamp.org/news/what-are-json-web-tokens-jwt-auth-tutorial/>