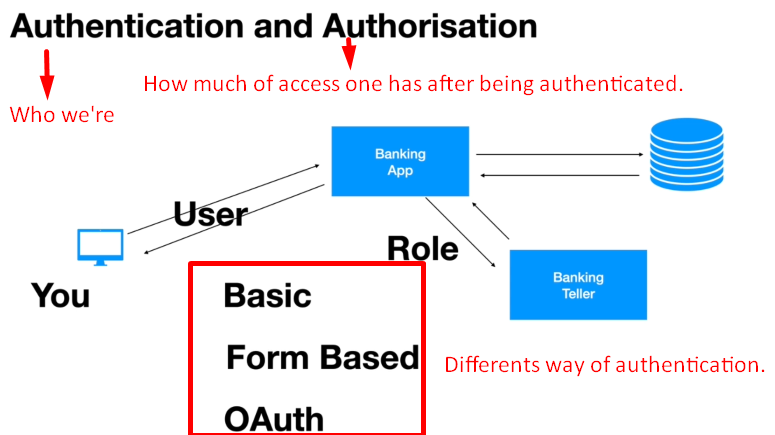
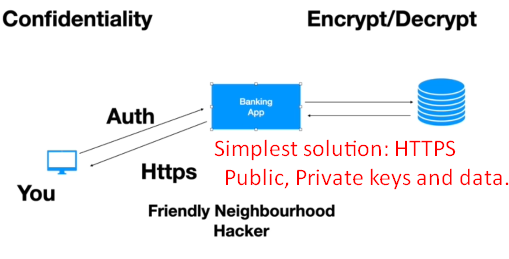
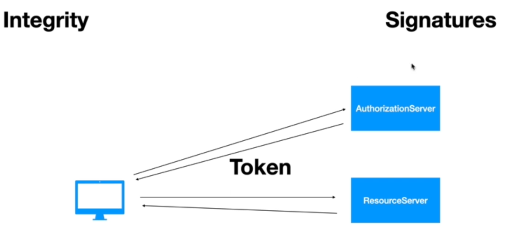
1. **Authentication**: Letting our app know who we’re.
2. **Authorization:** How much access a user or other system has.
   1. It uses roles.
   2. Each role is mapped to certain URLs or methods.



1. There are certain ways of authentications in the HTTP or web app world.
   1. **Basic**:
   2. **Form Based**:
   3. **OAuth**:
      1. In the RESTFul world, we can implement OAuth Single Sign-on where we implement authentication and authorization for all applications within our organization through Single Sign-On.
   4. **Custom**: We can use our custom login mechanism instead of using basic, form based or OAuth for authentication and authorization.
2. **Confidentiality**:
   1. This is where applications ensure that the data they are sharing is not vulnerable to hackers.
   2. For example, you’re logging in providing authentication details, what if a friendly neighborhood hacker hacks your username and password over the network?
   3. This is where **encryption and decryption** come in**.**
   4. The simplest way is by using HTTPS.
      1. Once the communication is encrypted, if the data being shared is captured by a Hacker, he will not make sense of it as the sender(Browser) will send the details with public key and the receiver would use private key to encrypt the data.
   5. 
3. **Integrity**:  
   
   1. Application ensures that data is coming from the expected user and is not changed in the exchange process.
   2. **For example**: If a hacker captures a request and changes the data or add some other data, the app must know this.
   3. **Solution**: Signature
   4. Later on, when you will see OAuth or JWT, you will learn authorization server and resource server.  
      Authorization server creates a token and gives it to application and that app sends that token to resource server.  
      How does resource server make sure that this token is really produced by that authentication server?  
      **Solution**: Signature.  
      The authentication token will use a private key again and it will sign this token.  
      Then the resource server with the public key corresponding to the private key will be able to verify the signature.  
      If a Hacker tweaks the token, the resource server will use the private key to verify the signature and one signature is passed with the token. They must match.
4. **CSRF and CORS**:
   1. **CSRF:** Will prevent other sites to submit data on our behalf. 