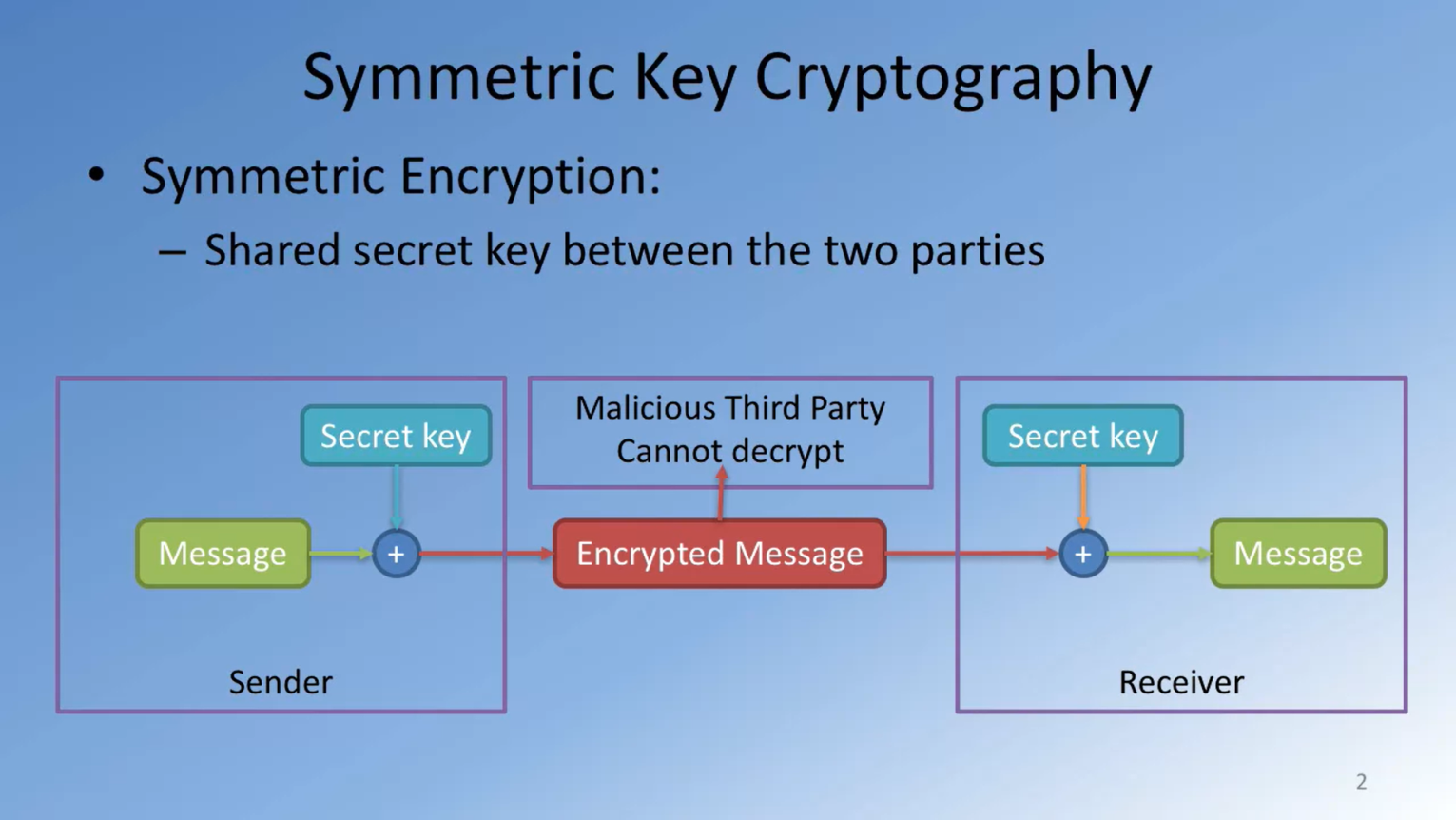
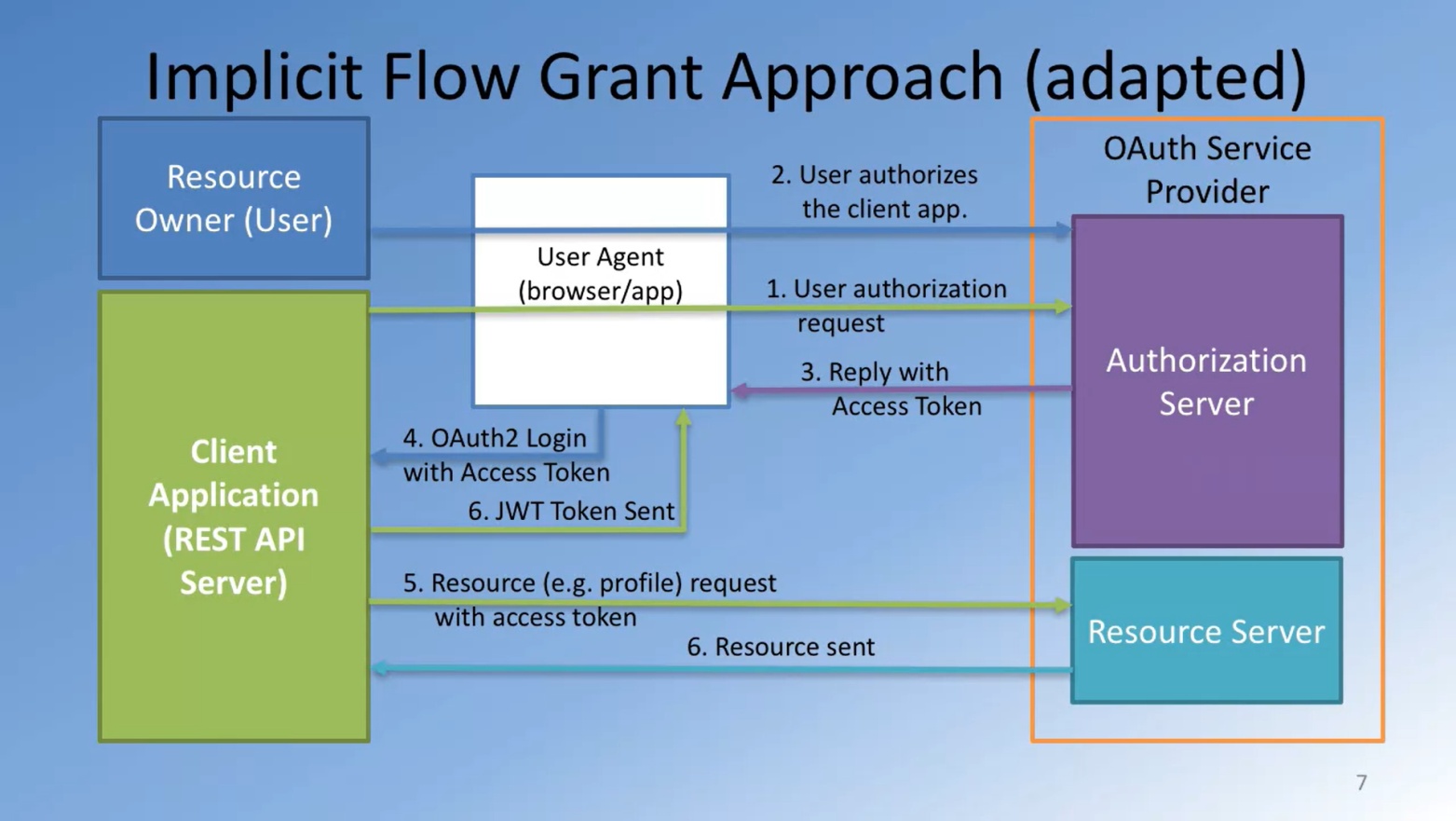
* **Authentication** can be done simply without using any modules in express.
* But **passport** module makes the code smaller for authentication.
* **JSON Web Tokens** creates a token to be used by the particular user. It makes the authentication procedure even more secure. It does not save the password directly, but encrypts it and then save it.
* **Population** is kind of inner join, outer join(SQL). Since these operations are not directly allowed in mongo, Population is used.(retrieving data from more than one table). Should be used only when its really required coz it increases the server side work.
* Any third party can get access, since the information is being passes through header. So **https** is used to establish a secure connection between the user and the server.



* **Cors(Cross-Origin Resource Sharing)**
* **OAuth** - These service providers in turn, are willing to provide authentication services on your behalf. So, for example, you see the proliferation of a number of websites and mobile applications where you are allowed to login using your social media accounts. Now, how does this actually work? Many of these social media account providers act as authentication service providers using a protocol called as OAuth.

**Example:** Pubg asking for fb details for logging into an account, instead of creating an user itself.  
  
These authentication service providers promise to the users of these authentication service that they will authenticate the identity of the user based upon their submission of their credentials for these social media services.



Authenticate.getToken 🡪 This will get us the JSON web token.

1. Essentially, the user is sending the access token to the express server,

2. The express server uses the accessToken to go to Facebook and

then fetch the profile of the user. And if the user doesn't exist, we'll create a new user with that Facebook ID.

3. After that, then our express server will generate a JSON web token and then return the JSON web token to our client.

All subsequent accesses from our user will have to include this JSON

web token that we have just returned by using this approach.

So at this point you no longer need the Facebook access token anymore.

**You can discard the Facebook access token at this point because the JSON web token is the one that keeps the users authentication active for whatever duration that this JSON web token is active.**

**Backend as a Service (BaaS): Loopback**

npm i -g @loopback/cli

lb appname

* . To create a Loopback model for dishes, type the following at the prompt:
* lb model
* For the model name type dishes. For the data source select db. For the model's base class select PersistedModel.
* Say Yes to REST API, and select the common model.
* Loopback will prompt for the properties of the model. Create the following properties, all of String type: name, description, category, image, label and price. Add also a property named featured of Boolean type that is required and a default value of false. All are required except label. For label, select the default value as ' ', and for price the default value as 0.

? Enter the model name: dishes

? Select the datasource to attach dishes to: db (memory)

? Select model's base class PersistedModel

? Expose dishes via the REST API? Yes

? Custom plural form (used to build REST URL):

? Common model or server only? Common

 we will look at how we can connect

our LoopBack server to a back-end database,

or MongoDB in this instance,

and we'll also see how we can set up various access controls on the recipient endpoints.

So using the access controls,

we can specify what kind of operations can be carried out on the recipient endpoints,

and who will be allowed to carry out those operations.

lb datasource

Enter the datasource name: MongoDB

? Select the connector for MongoDB: MongoDB (supported by StrongLoop)

? Connection String url to override other settings (eg: mongodb://username:password@hostname:port/database):

? host: localhost

? port: 27017

? user:

? password:

? database: conFusion

? Install loopback-connector-mongodb@^4.0.0 (Y/n) Run-async wrapped function (sync) returned a promise but async() callback must be executed to resolve.

? Install loopback-connector-mongodb@^4.0.0 Yes

**For access control in Loopback**

lb acl

. (all existing models)

. All methods and properties

. All (match all types)

. All users

. Explicitly deny access

{accordingly give access-- example}

dishes

A single method

create

Other users

role: admin

Explicitly grant access

Lb relation

Select the model to create the relationship from: dish

? Relation type: has many

? Choose a model to create a relationship with: comments

? Enter the property name for the relation: (comments) loopback-datasource-juggler deprecated Scope method "getAsync()" is deprecated, use

"find()" instead. C:\Users\hp\AppData\Roaming\npm\node\_modules\loopback-cli\node\_modules\generator-loopback\lib\helpers.js:132:37

? Enter the property name for the relation: comments

? Optionally enter a custom foreign key:

? Require a through model? No

? Allow the relation to be nested in REST APIs: No

? Disable the relation from being included: No