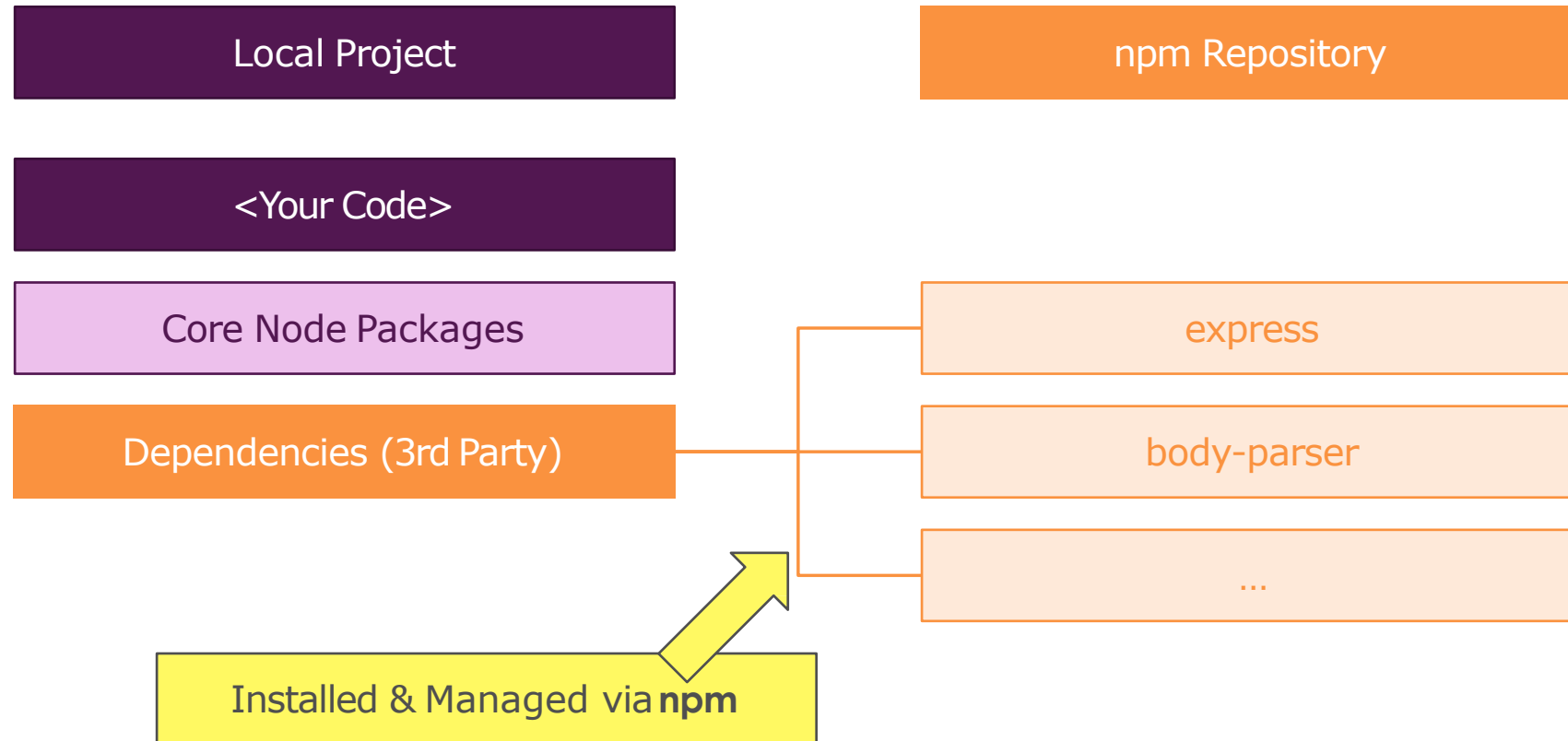




NPM & Modules

npm & packages Intro



What is npm?

- ▶ **npm** is a node package manager for Node.js packages, or modules if you like.
- ▶ www.npmjs.com hosts thousands of free packages to download and use.
- ▶ The NPM program is installed on your computer when you install Node.js.

- ▶ When we install a package:
 - ▶ Notice dependencies changes in `package.json`
 - ▶ notice folder: `node_modules`
 - ▶ This structure separate our app code to the dependencies. Later when we share/deploy our application, there's no need to copy `node_modules`, run:
`npm install` will read all dependencies and install them locally.

What is a Package?

- ▶ A package in Node.js contains all the files you need for a module.
- ▶ Modules are JavaScript libraries you can include in your project.
- ▶ A package contains:
 - ▶ JS files
 - ▶ package.json (manifest)
 - ▶ package-lock.json (maybe)

npm CLI Commands

```
npm -v // will print npm version
npm init // will create package.json
npm install <package> --S // download & install the code from last commit of git repo
                        // "--save" option will update package.json automatically
                        // other options are: --save-dev (-D) --save-optional (-O)
npm i <package> -g // download & install a package globally
npm i <package> --dry-run
npm ls -g --depth=0 // show all global packages in your system
npm update // check versions in package.json and update
npm i npm -g // update npm
npm outdated -g // show all outdated global packages
npm prune // if a package is installed without --save then delete and clean

npm config list // display the default npm settings
npm config set init-author-name "Josh Edward"
npm config delete init-author-name
npm config set save true // automatically --save (-S)

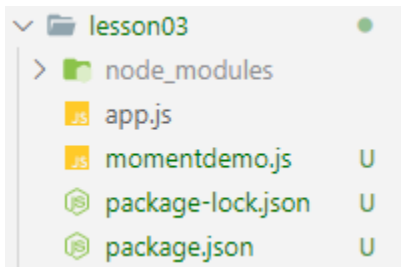
npm search lint // search online for package with lint in the name
npm home <package> // open browser to package homepage
npm repo <package> // open browser to package repository
```

Create & use a new package

`npm init` //follow the instruction to set up your project

`npm install moment --save`

// moment is a package that parse, validate, manipulate and display dates



momentdemo.js

```
const moment = require('moment');  
console.log(moment().format("ddd, hA")); // Mon, 10AM
```

package.json Manifest

```
{
  "name": "lesson03-demo",
  "version": "1.0.0",
  "description": "lesson 3 demos",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rujuan Xing",
  "license": "ISC",
  "dependencies": {
    "moment": "^2.24.0"
  }
}
```

```
// "scripts" defines commands we can run using (npm run commandName)
// ^ in version means it's ok to automatically update to anything
//   within this major release
// ~ means only update patches. Not even minor updates
```

Semantic Versioning

4.7.6

Major Version

Major Changes
Breaks the API
(maybe)

Minor Version

Minor Changes
Does not break the API

Patches

Bug fixes

package-lock.json

- ▶ Introduced by NPM version 5 to capture the exact dependency tree installed at any point in time.
- ▶ Describes the exact tree
- ▶ Guarantee the dependencies on all environments.
- ▶ Use `npm ci` if you want to use dependencies in package-lock.json file
- ▶ Don't modify this file manually.
- ▶ Always use npm CLI to change dependencies, it'll automatically update package-lock.json

```
{
  "name": "lesson03-demo",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "moment": {
      "version": "2.24.0",
      "resolved": "https://registry.npmjs.org/moment/-/moment-2.24.0.tgz",
      "integrity": "sha512-bV7f+6l2QigeBBZSM/6yTNq4P2fNpSWj/0e7jQcy87A8e7o2nAfP/34/2ky5Vw4B9S446EtIhodAzkFCcR4dQg=="
    }
  }
}
```


More About Packages

- ▶ **Development Dependencies:** Needed only while I'm developing the app. It's not needed for running the app.
 - ▶ `npm install mocha --save-dev`
`// notice devDependencies entry now in package.json`
- ▶ **Global Dependencies:** Available to all applications
 - ▶ `npm install -g nodemon`
 - ▶ `nodemon app.js` `//auto detects changes and restarts your project`

Recall: Create a Node Server

```
const http = require('http');

const server = http.createServer((req, res) => {
  console.log(req);
  // process.exit(); //typically you don't do this.
});

server.listen(3000);
```

Understanding Request & Response

- ▶ A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.
- ▶ After receiving and interpreting a request message, a server responds with an HTTP response message.

```
const http = require('http');

const server = http.createServer((req, res) => {
  console.log(req.url, req.method, req.headers);

  res.setHeader('Content-Type', 'text/html');
  res.write('<html>');
  res.write('<head><title>My First Page</title></head>');
  res.write('<body><h1>Hellow from Node.js</h1></body>');
  res.write('</html>');
  res.end();

  // process.exit(); //typically you don't do this.
});

server.listen(3000);
```

HTTP Request: Reading Get and Post Data

- ▶ Handling basic GET & POST requests is relatively simple with Node.js.
- ▶ We use the `url` module to parse and read information from the URL.
- ▶ The `url` module uses the WHATWG URL Standard (<https://url.spec.whatwg.org/>)

Elements in URL Object

href									
protocol		auth		host		path		hash	
				hostname	port	pathname	search		
							query		
" https:	//	user	: pass	@	sub.host.com	:	8080	/p/a/t/h	? query=string #hash "
protocol		username	password		hostname		port		
origin				host		pathname		search	
				origin		pathname		search	
href									

Using URL Module

- ▶ `url.parse(str)` will return URL object with properties (protocol, hostname, port, pathname, hash, etc...)

```
const url = require('url');  
const myURL = url.parse('https://user:pass@sub.host.com:8080/p/a/t/h?course1=nodejs&course2=angular#hash');  
console.log(myURL);
```

```
Url {  
  protocol: 'https:',  
  slashes: true,  
  auth: 'user:pass',  
  host: 'sub.host.com:8080',  
  port: '8080',  
  hostname: 'sub.host.com',  
  hash: '#hash',  
  search: '?course1=nodejs&course2=angular',  
  query: 'course1=nodejs&course2=angular',  
  pathname: '/p/a/t/h',  
  path: '/p/a/t/h?course1=nodejs&course2=angular',  
  href: 'https://user:pass@sub.host.com:8080/p/a/t/h?course1=nodejs&course2=angular#hash'  
}
```

Parsing the Query String

```
const url = require('url');  
const myURL2 = url.parse('https://user:pass@sub.host.com:8080/p/a/t/h?course1=nodejs&course2=angular#hash', true);  
console.log(myURL2);
```

```
Url {  
  protocol: 'https:',  
  slashes: true,  
  auth: 'user:pass',  
  host: 'sub.host.com:8080',  
  port: '8080',  
  hostname: 'sub.host.com',  
  hash: '#hash',  
  search: '?course1=nodejs&course2=angular',  
  query: [Object: null prototype] { course1: 'nodejs', course2: 'angular' },  
  pathname: '/p/a/t/h',  
  path: '/p/a/t/h?course1=nodejs&course2=angular',  
  href: 'https://user:pass@sub.host.com:8080/p/a/t/h?course1=nodejs&course2=angular#hash'  
}
```

Format a URL

```
const urlObject = {  
  protocol: 'http',  
  host: 'www.mim.edu',  
  search: '?q=cs',  
  pathname: '/search', };  
  
console.log( url.format(urlObject) );  
// http://www.mim.edu/search?q=CS
```


Using querystring module

```
const querystring = require('querystring');

const result1 = querystring.stringify({
  firstname: 'Josh',
  lastname: 'Edward'
})

console.log(result1); //firstname=Josh&lastname=Edward

const result2 = querystring.parse('firstname=Josh&lastname=Edward');
console.log(result2); // {firstname: 'Josh', lastname: 'Edward'}
```

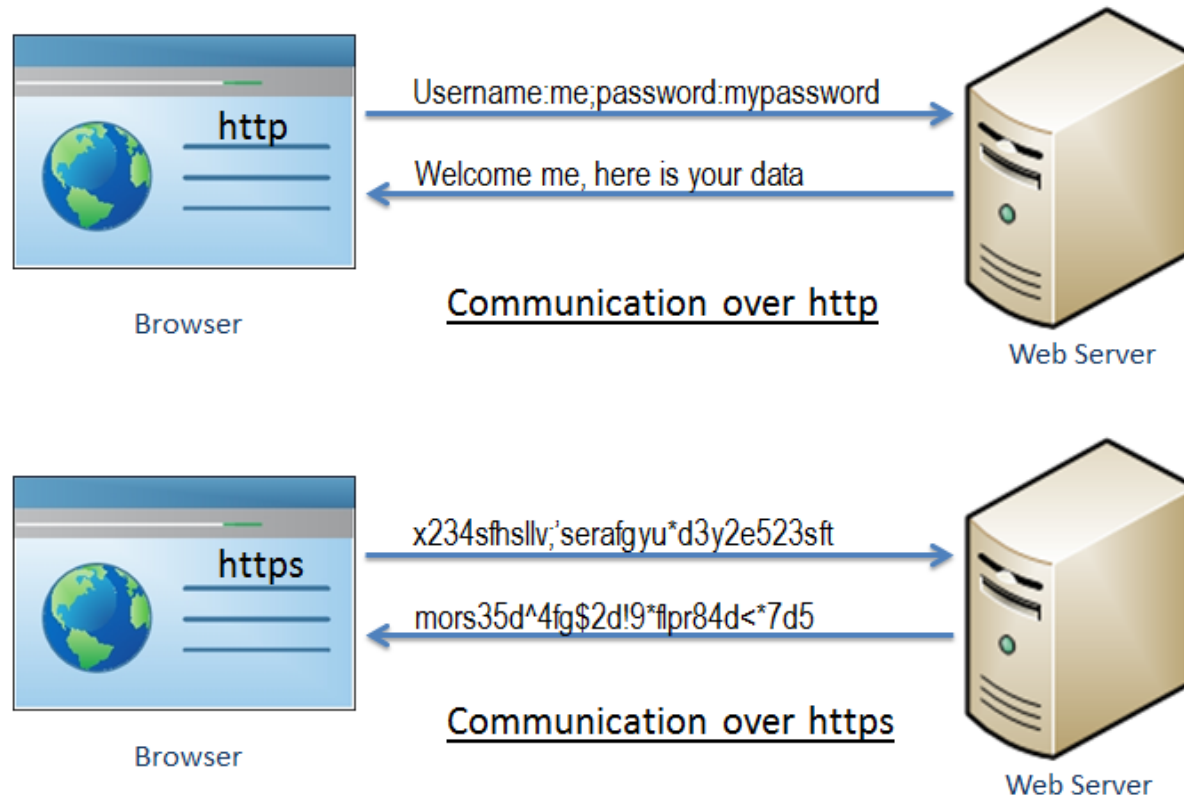
Routing Requests

```
var http = require('http');
var fs = require('fs');

http.createServer(function(req, res) {
  if (req.url === '/') {
    fs.createReadStream(__dirname + '/index.htm').pipe(res);
  } else if (req.url === '/api') {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    var obj = { firstname: 'Josh', lastname: 'Edward' };
    res.end(JSON.stringify(obj));
  } else {
    res.writeHead(404);
    res.end();
  }
}).listen(1337, '127.0.0.1');
```

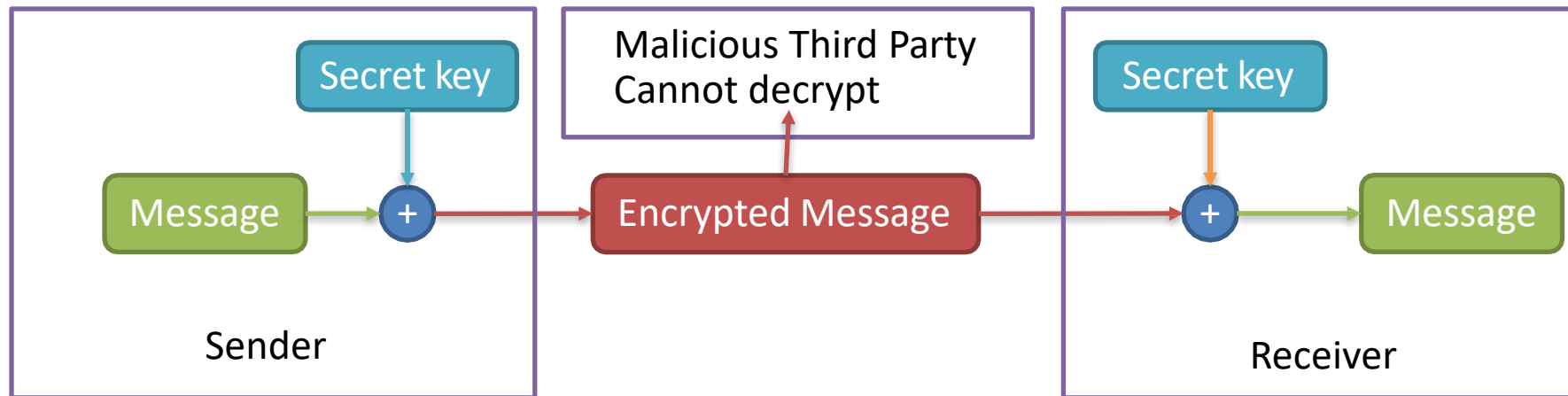
HTTPS and Secure Communication

- ▶ HTTPS stands for Hyper Text Transfer Protocol Secure. It is a protocol for securing the communication between two systems e.g. the browser and the web server.
- ▶ The following figure illustrates the difference between communication over http and https:



Symmetric Key Cryptography

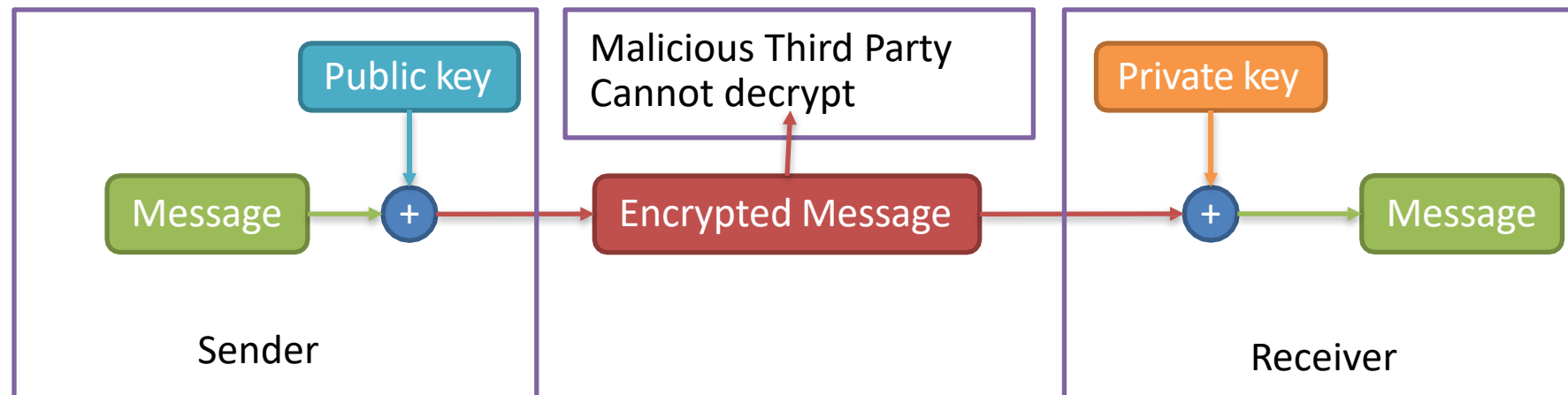
- ▶ Symmetric Encryption
 - ▶ Shared secret key between the two parties



Public Key Cryptography

► Asymmetric Encryption

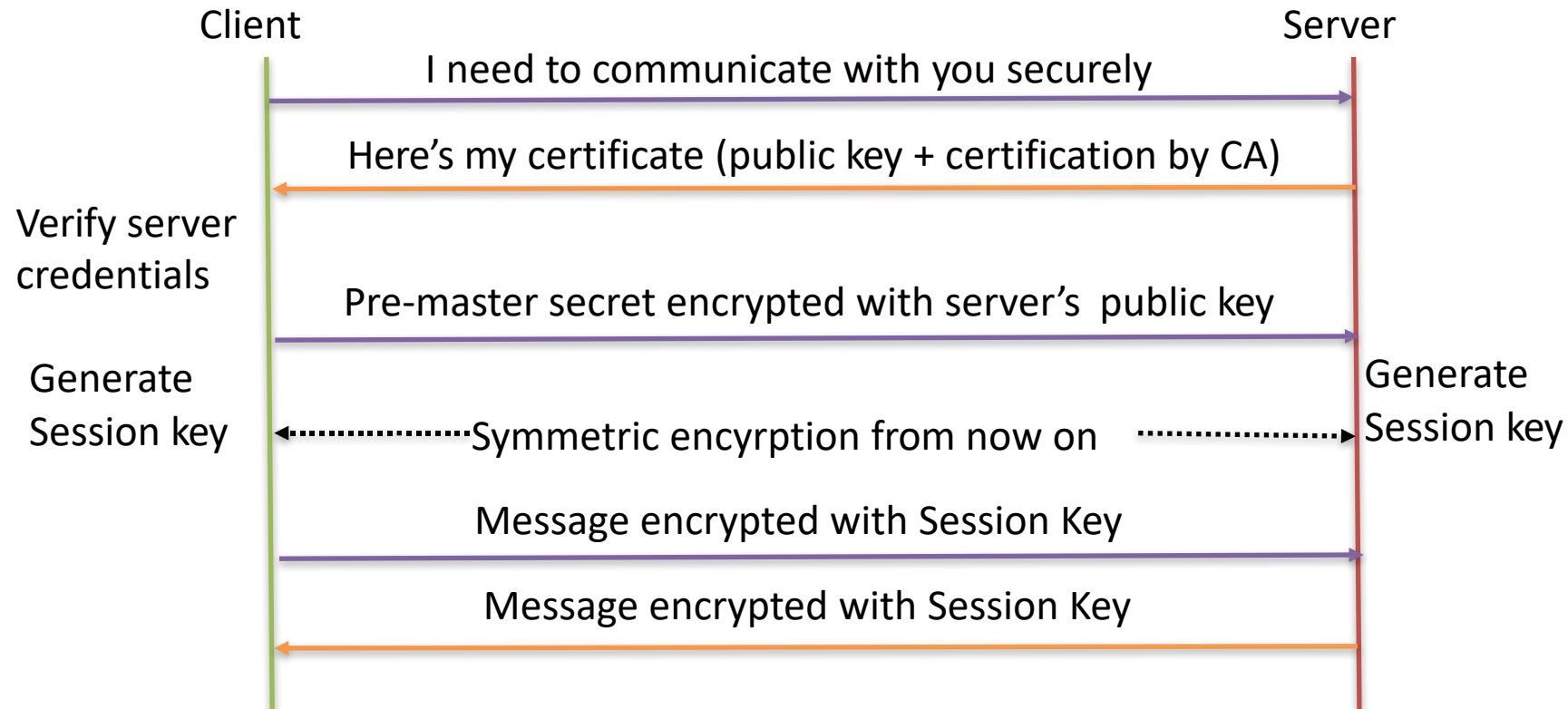
- Public key that can be widely distributed
- Private key that is only known to the receiver



Secure Sockets Layer (SSL) / Transport Layer Security (TLS)

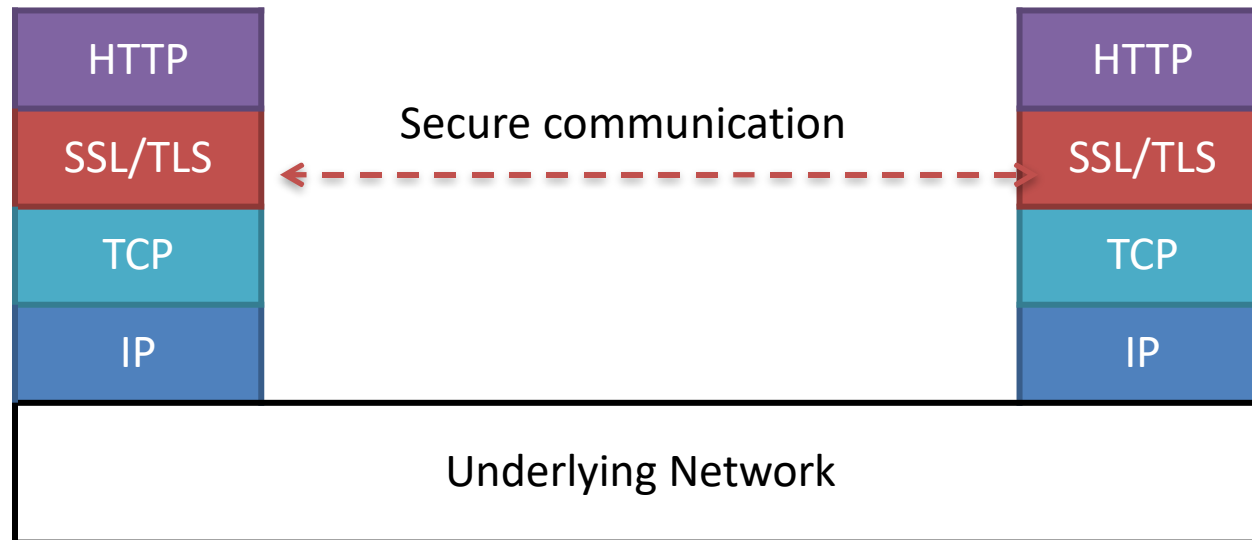
- ▶ Cryptographic protocols that enable secure communication over an insecure network like the Internet
- ▶ Privacy and Integrity of the communication protected
 - ▶ Uses a combination of public-key cryptography and symmetric cryptography

SSL/TLS Handshake



<https://www.ssl.com/article/ssl-tls-handshake-overview/>

HTTPS

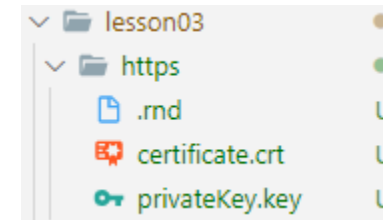


Generating Keys

- ▶ OpenSSL Windows installer is here
<http://slproweb.com/products/Win32OpenSSL.html>
- ▶ Navigate to the OpenSSL bin directory.
 - ▶ C:\openssl_X64\bin in our example. You can also add openssl.exe as Path environment variable, then you can use command “openssl” under all direcotries.
- ▶ Right-click the **openssl.exe** file and select **Run as administrator**.
- ▶ Enter the following command to begin generating a certificate and private key:
 - ▶ **req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout privateKey.key -out certificate.crt**
- ▶ For production environment / deploying to a production server you need to get the keys and certificate from a certification authority (CA) e.g., Verisign, Thawte

<https://helpcenter.gsx.com/hc/en-us/articles/115015960428-How-to-Generate-a-Self-Signed-Certificate-and-Private-Key-using-OpenSSL>

```
OpenSSL> req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout privateKey.key -out certificate.crt
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'privateKey.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Iowa
Locality Name (eg, city) []:Fairfield
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MIU
Organizational Unit Name (eg, section) []:MSD
Common Name (eg, YOUR name) []:Rujuan Xing
Email Address []:rxing@miu.edu
```



Create HTTPS Server

```
const fs = require('fs');

var options = {
  key: fs.readFileSync('./privateKey.key'),
  cert: fs.readFileSync('./certificate.crt')
};

const server = require('https')
  .createServer(options);

server.on('request', (req, res) => {
  res.writeHead(200, { 'content-type': 'text/plain' });
  res.end('Hello from my HTTPS Web server!!!\n');
});

server.listen(443);
```

Try this link if the previous slides doesn't work for you:
<https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/>

Resources

▶ Node and NPM

- ▶ [Nodejs.org](https://nodejs.org)
- ▶ [Npmjs.com](https://npmjs.com)
- ▶ [Node API Documentation](https://nodejs.org/en/docs/guides/node-api/documentation)
- ▶ [NPM Documentation](https://docs.npmjs.com)

▶ HTTPS

- ▶ [HTTPS \(Wikipedia\)](https://en.wikipedia.org/wiki/HTTPS)
- ▶ [Public Key Cryptography](#)
- ▶ [Transport Layer Security](#)
- ▶ [Node HTTPS Server](#)
- ▶ [Kurose, James F., and Keith W. Ross. Computer networking: a top-down approach. Pearson, 2017, ISBN-10: 0134522206 • ISBN-13: 9780134522203.](#)

▶ Other Resources

- ▶ [Howto: Make Your Own Cert With OpenSSL on Windows](#)
- ▶ [OpenSSL for Windows](#)
- ▶ [How to Use SSL/TLS with Node.js](#)
- ▶ [Adding HTTPS \(SSL\) to Express 4.X Applications](#)
- ▶ [How does HTTPS actually work?](#)

Homework

- ▶ Create a http server which is listen to 3000 port. The home page “/” which displays an html page which one input to enter any text message, after click “Submit” button, the user’s inputs are stored in a local file on the server side. User will be redirect to home page after saving successfully.