Date:
M T W T F S S

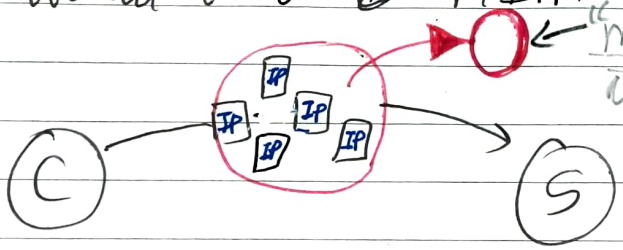
Security & HTTPS

Stuff you gotta know to understand HTTPS

I] Man In the Middle Attack (MITM)

i) An attack in which the attacker intercepts a line of comm" i.e. thought to be pvt by its 2 communicating parties

ii) A malicious actor intercepted and mutated an IP packet on its way from client to server, that would be a MITM attack



"malicious client" intercepts IP packets
① Goes thru sensitive info that they contain
② Manipulates data exchanged b/w client - server for some other purpose

iii) MITM attacks are the primary threat to privacy, cybersecurity and what encryp", HTTPS aim to defend against

2] Encryp": A way of securing data exchanged b/w parties by converting data into an unrecognizable string of random bytes and giving only concerned parties the power to reconvert it back into understandable data.

2 types of encryp^n

(1) Symmetric encryp^n

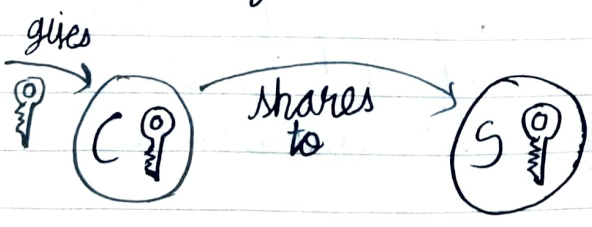
i) Type of encryp^n that relies on a single key (🔑) to both encrypt, decrypt data

ii) Key must be known to all parties involved in comm^n and must ∴ typically be shared b/w parties at 1 point or another → this is a vulnerable pt. as sharing can be intercepted by a MITM attack.

iii) Faster

iv) Symmetric key algos are (mostly part of the AES) used to generate the key.

symm. key algo gives



Use : for comm^n after secure connec^n b/w client and server has been established

(2) Asymmetric encryp^n

The type of encryp^n that relies on 2 keys → a public key (🔑) and a prvt. key (🔑) to encrypt, decrypt

The party involved should only know the key it is concerned with i.e. if it is concerned with
a) decryptin → prvt. key
b) encryptin → public key.

Slower

Keys generated usin cryptographic algos and are mathematically connected s.t. the data encrypted with public key can only be decrypted with prvt. key.

Use : to establish a secure connec^n b/w client-server

3] AES (Advanced Encrypⁿ Std.)

i) A widely used encrypⁿ std. that has 3 symm. key algos
- AES-128
- AES-192
- AES-256

ii) "Gold std" of encrypⁿ

iii) Use → U.S. National Security Agency to encrypt top secret info.

4] HTTPS (Hyper Text Transfer Protocol)

i) Extension of HTTP that runs on top of TLS (Transport Layer security). It a.k.a. HTTP on top of TLS

ii) Used for secure commⁿ online

iii) Requires servers to pr trusted certificates (SSL certificates) and uses the TLS, a security protocol built on top of TCP to encrypt data communicated b/w a client and a server. This process is a.k.a. "TLS Handshake"

5] TLS

The Transport Layer Security (TSL) is a protocol over which HTTP runs in order to achieve secure commⁿ online. HTTP over TLS is a.k.a. HTTPS

6) SSL certificate (Secure socket layer)

1) A digital certificate granted to a server by a certification authority (CA) → contains

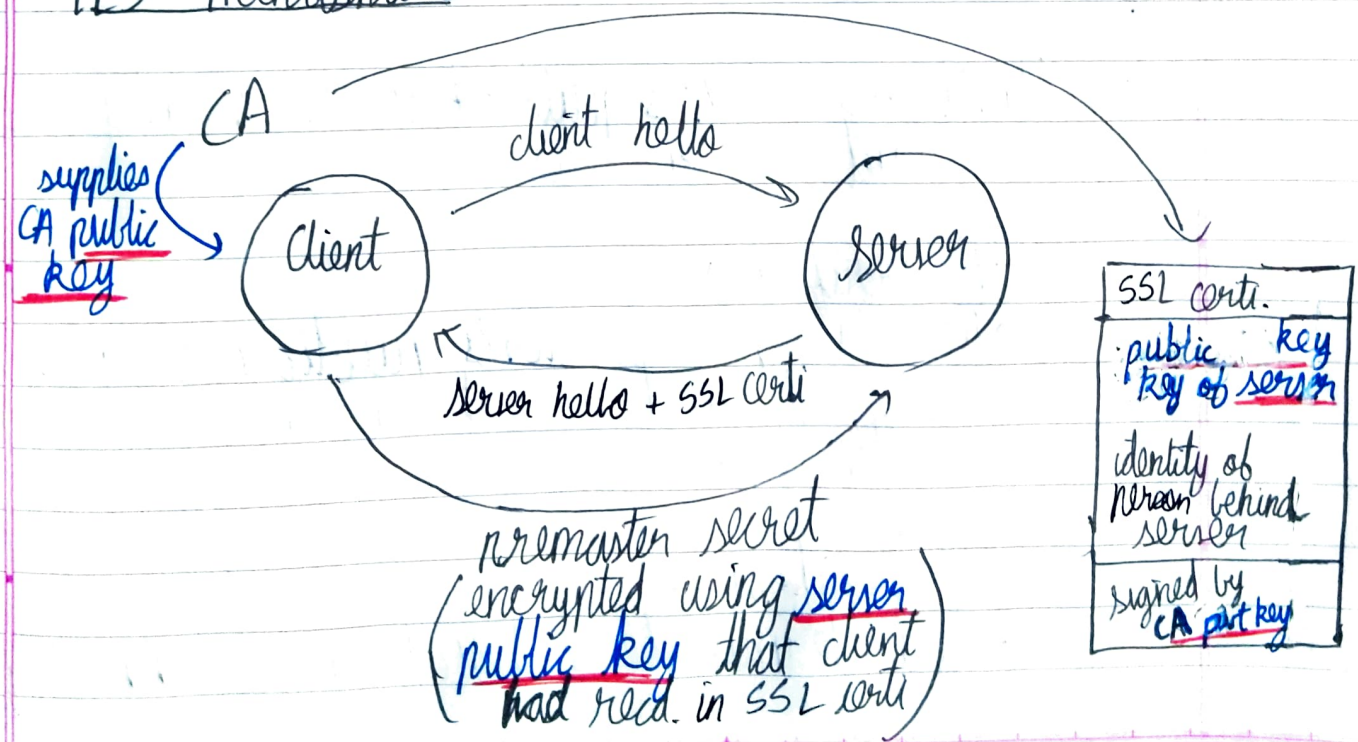
- (1) server's public key
- (2) name of entity owning public-pvt key pair i.e. the company whose server this is
- signed by CA's pvt key

2) Server's public key is used as part of the TLS handshake process in an HTTPS connection

3) An SSL certi effectively confirms that the public key belongs to the server claiming that it's their key.

4) SSL certi are a crucial step in defending against MITM attacks as they
   - ↳ provide identity of server
   - ↳ check and verify this identity

7) TLS Handshake



CA (Certificate Authority) → A trusted entity that signs digital certs namely SSL certs that are relied on in HTTPS connec^n's

TLS Handshake steps:

1) Client sends "client hello" – str. of random bytes – to server to start handshake process

2) Server responds by generation server's public-pvt key pair and sendin back 2 things:
   i) "Server Hello" – str of random bytes acknowledgin initia^n of HTTPS connec^n by client
   ii) SSL certi. contains → server's public key
                           → CA's pvt. key
                           → server owner's identity

Meanwhile, client obtains CA's public key from CA itself.

iii) Client uses CA's public key to verify SSL certi. sent by server
   a) If verified ✓
      client accepts server's public key given in SSL certi and uses it in the next step of the process
   b) If not verified X
      client terminate's handshake, connec^n fails

iv) Client generates a "premaster secret" encrypted using the server's public key it just recd.

v) Server uses its own pvt. key i.e. server's pvt. key to gain access to 'premaster secret'

vi) Now client and server both hv access to 3 things:

- client hello
- server hello
- premaster secret.

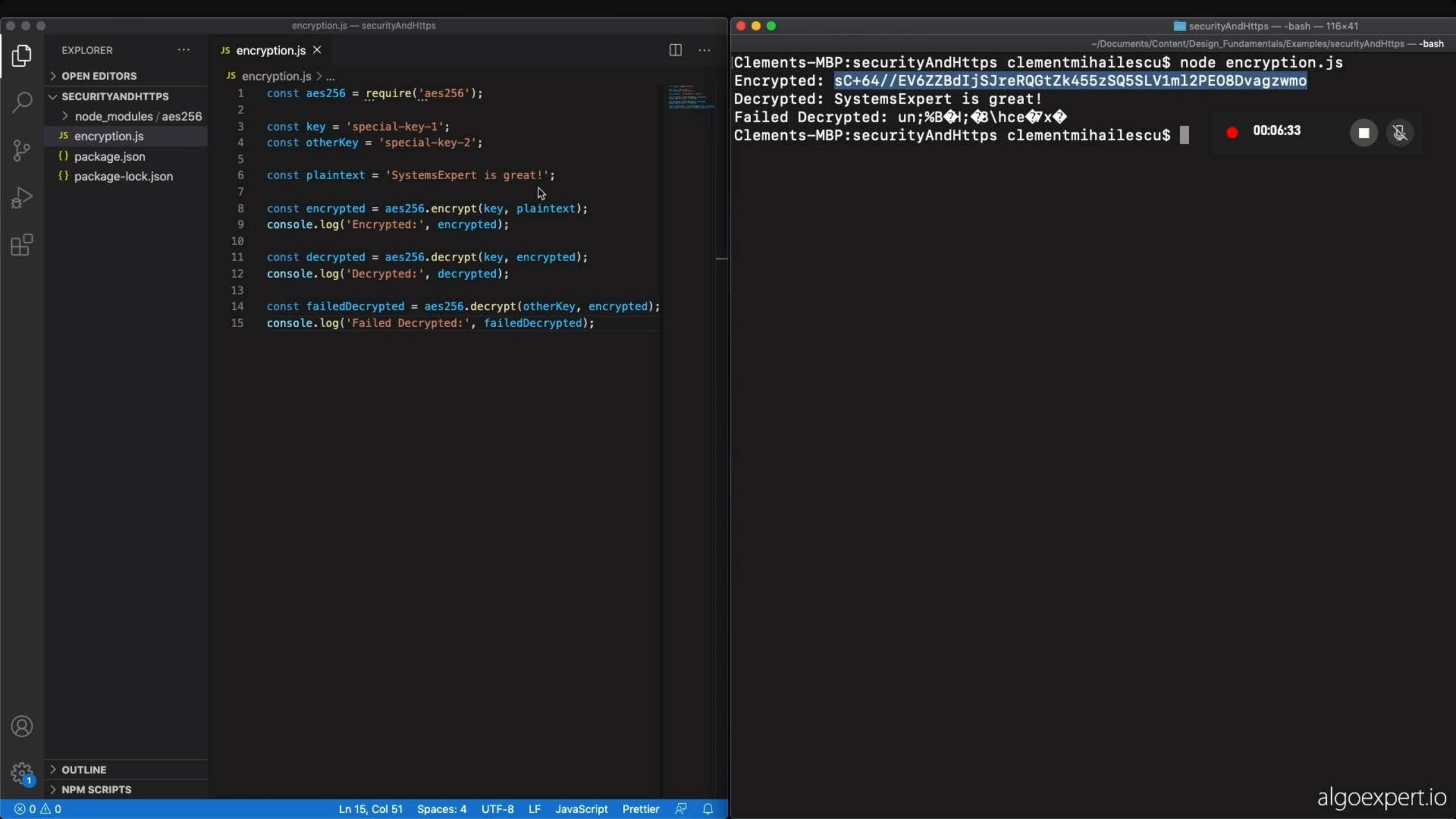
} + key algo = session key

HTTPS connec" established

vii) Using these 3 things and a symm. key algo, they generate a symmetric "session key" that they'll both store and use to decrypt/encrypt and data exchange in that session.

HTTPS connec" terminated

viii) Once connec" is terminated, curr. session key is disposed off and a new one will be generated by the "TLS Handshake" process next time around.


encryption.js — securityAndHttps
EXPLORER
OPEN EDITORS
SECURITYANDHTTPS
node_modules / aes256
encryption.js
package.json
package-lock.json
encryption.js
const aes256 = require('aes256');
const key = 'special-key-1';
const otherKey = 'special-key-2';
const plaintext = 'SystemsExpert is great!';
const encrypted = aes256.encrypt(key, plaintext);
console.log('Encrypted:', encrypted);
const decrypted = aes256.decrypt(key, encrypted);
console.log('Decrypted:', decrypted);
const failedDecrypted = aes256.decrypt(otherKey, encrypted);
console.log('Failed Decrypted:', failedDecrypted);
OUTLINE
NPM SCRIPTS
Ln 15, Col 51
Spaces: 4
UTF-8
LF
JavaScript
Prettier
securityAndHttps — -bash — 116×41
~/Documents/Content/Design_Fundamentals/Examples/securityAndHttps — -bash
Clements-MBP:securityAndHttps clementmihailescu$ node encryption.js
Encrypted: sC+64//EV6ZZBdIjSJreRQGtZk455zSQ5SLV1ml2PEO8Dvagzwmo
Decrypted: SystemsExpert is great!
Failed Decrypted: un;%B?H;?8\hce??x?
Clements-MBP:securityAndHttps clementmihailescu$
00:06:33
algoexpert.io