

Encryption

HTTP Request

ip.addr == 204.174.63.5

No.	Time	Source	Destination	Protocol	Length	Info
472	54.276885	10.0.0.200	204.174.63.5	TCP	78	49742 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...
482	54.277326	204.174.63.5	10.0.0.200	TCP	66	80 → 49742 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W...
484	54.277360	10.0.0.200	204.174.63.5	TCP	54	49742 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
488	54.278039	10.0.0.200	204.174.63.5	TCP	78	49744 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...
490	54.278195	204.174.63.5	10.0.0.200	TCP	66	80 → 49744 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W...
491	54.278218	10.0.0.200	204.174.63.5	TCP	54	49744 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
496	54.278923	10.0.0.200	204.174.63.5	HTTP	413	GET / HTTP/1.1
499	54.279171	204.174.63.5	10.0.0.200	TCP	60	80 → 49744 [ACK] Seq=1 Ack=360 Win=30336 Len=0
530	54.348738	204.174.63.5	10.0.0.200	HTTP	506	HTTP/1.1 302 Found (text/html)
531	54.348740	204.174.63.5	10.0.0.200	TCP	60	80 → 49744 [FIN, ACK] Seq=453 Ack=360 Win=30336 Len=0
532	54.348806	10.0.0.200	204.174.63.5	TCP	54	49744 → 80 [ACK] Seq=360 Ack=453 Win=261632 Len=0
533	54.348807	10.0.0.200	204.174.63.5	TCP	54	49744 → 80 [ACK] Seq=360 Ack=454 Win=261632 Len=0
534	54.349515	10.0.0.200	204.174.63.5	TCP	54	49744 → 80 [FIN, ACK] Seq=360 Ack=454 Win=262144 Len=0
535	54.350134	204.174.63.5	10.0.0.200	TCP	60	80 → 49744 [ACK] Seq=454 Ack=361 Win=30336 Len=0
630	54.673465	10.0.0.200	204.174.63.5	TCP	78	49752 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...

Internet Protocol Version 4, Src: 10.0.0.200, Dst: 204.174.63.5

Transmission Control Protocol, Src Port: 49744, Dst Port: 80, Seq: 1, Ack: 1, Len: 359

Hypertext Transfer Protocol

GET / HTTP/1.1\r\n

[Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]

Request Method: GET

Request URI: /

Request Version: HTTP/1.1

Host: www.camosun.bc.ca\r\n

Upgrade-Insecure-Requests: 1\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.2 Safari/605.1.15\r\n

Accept-Language: en-ca\r\n

0030 10 00 17 fd 00 00 47 45 54 20 2f 20 48 54 54 50GE T / HTTP

0040 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e /1.1..Ho st: www.

0050 63 61 6d 6f 73 75 6e 2e 62 63 2e 63 61 0d 0a 55 camosun. bc.ca..U

0060 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65 2d pgrade-I nsecure-

0070 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 41 63 63 Requests : 1..Acc

0080 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61 ept: tex t/html,a

0090 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c pplicati on/xhtmll

00a0 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e +xml,app lication

00b0 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 2a 2f 2a 3b 71 /xml;q= .9,*/*;q

00c0 3d 30 2e 38 0d 0a 55 73 65 72 2d 41 67 65 6e 74 =0.8..Us er-Agent

00d0 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 4d : Mozill a/5.0 (M

00e0 61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65 6c 20 acintosh ; Intel

00f0 4d 61 63 20 4f 53 20 58 20 31 30 5f 31 35 5f 36 Mac OS X 10_15_6

0100 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 36 30) AppleW ebKit/60

0110 35 2e 31 2e 31 35 20 28 4b 48 54 4d 4c 2c 20 6c 5.1.15 (KHTML, l

0120 69 6b 65 20 47 65 63 6b 6f 29 20 56 65 72 73 69 ike Geck o) Versi

0130 6f 6e 2f 31 33 2e 31 2e 32 20 53 61 66 61 72 69 on/13.1. 2 Safari

0140 2f 36 30 35 2e 31 2e 31 35 0d 0a 41 63 63 65 70 /605.1.1 5..Accep

0150 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 63 t-Langua ge: en-c

0160 61 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 a..Accep t-Encodi

0170 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 74 ng: gzip , deflat

0180 65 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b e..Conne ction: k

Hypertext Transfer Protocol (http), 359 bytes

Packets: 5969 · Displayed: 3771 (63.2%) · Dropped: 0 (0.0%) · Profile: Default

Note that HTTP header and body (HTML) are sent in plain text; easy to intercept

HTTP Search

ip.addr == 204.174.63.5

No.	Time	Source	Destination	Protocol	Length	Info
73	9.082273	10.0.0.200	204.174.63.5	TCP	78	49895 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...
76	9.082741	204.174.63.5	10.0.0.200	TCP	66	80 → 49895 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W...
77	9.082790	10.0.0.200	204.174.63.5	TCP	54	49895 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
79	9.083479	10.0.0.200	204.174.63.5	HTTP	769	GET /search?q=Computer+Science HTTP/1.1
80	9.083759	204.174.63.5	10.0.0.200	TCP	60	80 → 49895 [ACK] Seq=1 Ack=716 Win=30720 Len=0
83	9.134675	204.174.63.5	10.0.0.200	HTTP	600	HTTP/1.1 301 Moved Permanently (text/html)
84	9.134678	204.174.63.5	10.0.0.200	TCP	60	80 → 49895 [FIN, ACK] Seq=547 Ack=716 Win=30720 Len=0
85	9.134769	10.0.0.200	204.174.63.5	TCP	54	49895 → 80 [ACK] Seq=716 Ack=547 Win=261568 Len=0
86	9.134772	10.0.0.200	204.174.63.5	TCP	54	49895 → 80 [ACK] Seq=716 Ack=548 Win=261568 Len=0
87	9.135527	10.0.0.200	204.174.63.5	TCP	54	49895 → 80 [FIN, ACK] Seq=716 Ack=548 Win=262144 Len=0
88	9.135885	204.174.63.5	10.0.0.200	TCP	60	80 → 49895 [ACK] Seq=548 Ack=717 Win=30720 Len=0
89	9.141470	10.0.0.200	204.174.63.5	TCP	78	49897 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...
91	9.141832	204.174.63.5	10.0.0.200	TCP	66	80 → 49897 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W...
92	9.141868	10.0.0.200	204.174.63.5	TCP	54	49897 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
94	9.142001	10.0.0.200	204.174.63.5	HTTP	770	GET /search/?q=Computer+Science HTTP/1.1

Frame 79: 769 bytes on wire (6152 bits), 769 bytes captured (6152 bits) on interface en0, id 0

Ethernet II, Src: Apple_f3:ac:6f (f0:18:98:f3:ac:6f), Dst: RealtekU_64:30:1c (52:54:00:64:30:1c)

Internet Protocol Version 4, Src: 10.0.0.200, Dst: 204.174.63.5

Transmission Control Protocol, Src Port: 49895, Dst Port: 80, Seq: 1, Ack: 1, Len: 715

Hypertext Transfer Protocol

GET /search?q=Computer+Science HTTP/1.1\r\n

[Expert Info (Chat/Sequence): GET /search?q=Computer+Science HTTP/1.1\r\n]

Request Method: GET

Request URI: /search?q=Computer+Science

Request Version: HTTP/1.1

Host: camosun.ca\r\n

[truncated]Cookie: __qca=P0-126811573-1600204349872; __utma=46804917.1259099860.1600204349.1600204349.1600204349.1; __utmb=46804917.1.10.1600204349; __

Note that the search text is encoded in plain text in the GET request

0030 10 00 19 61 00 00 47 45 54 20 2f 73 65 61 72 63 ...a...GE T /search
0040 68 3f 71 3d 43 6f 6d 70 75 74 65 72 2b 53 63 69 h?q=Comp uter+Sci
0050 65 6e 63 65 20 48 54 54 50 2f 31 2e 31 0d 0a 48 ence HTT P/1.1..H
0060 6f 73 74 3a 20 63 61 6d 6f 73 75 6e 2e 63 61 0d ost: cam osun.ca.
0070 0a 43 6f 6f 6b 69 65 3a 20 5f 5f 71 63 61 3d 50 .Cookie: __qca=P
0080 30 2d 31 32 36 38 31 31 35 37 33 2d 31 36 30 30 0-126811 573-1600
0090 32 30 34 33 34 39 38 37 32 3b 20 5f 5f 75 74 6d 20434987 2; __utm
00a0 61 3d 34 36 38 30 34 39 31 37 2e 31 32 35 39 30 a=468049 17.12590
00b0 39 39 38 36 30 2e 31 36 30 30 32 30 34 33 34 39 99860.16 00204349
00c0 2e 31 36 30 30 32 30 34 33 34 39 2e 31 36 30 30 .1600204 349.1600
00d0 32 30 34 33 34 39 2e 31 3b 20 5f 5f 75 74 6d 62 204349.1; __utmb
00e0 3d 34 36 38 30 34 39 31 37 2e 31 2e 31 30 2e 31 =4680491 7.1.10.1
00f0 36 30 30 32 30 34 33 34 39 3b 20 5f 5f 75 74 6d 60020434 9; __utm
0100 63 3d 34 36 38 30 34 39 31 37 3b 20 5f 5f 75 74 c=468049 17; __ut
0110 6d 74 3d 31 3b 20 5f 5f 75 74 6d 7a 3d 34 36 38 mt=1; __ utmz=468
0120 30 34 39 31 37 2e 31 36 30 30 32 30 34 33 34 39 04917.16 00204349
0130 2e 31 2e 31 2e 75 74 6d 63 73 72 3d 28 64 69 72 .1.1.utm csr=(dir
0140 65 63 74 29 7c 75 74 6d 63 63 6e 3d 28 64 69 72 ect)|utm ccn=(dir
0150 65 63 74 29 7c 75 74 6d 63 6d 64 3d 28 6e 6f 6e ect)|utm cmd=(non
0160 65 29 3b 20 5f 67 61 3d 47 41 31 2e 32 2e 31 32 e); __ga= GA1.2.12
0170 35 39 30 39 39 38 36 30 2e 31 36 30 30 32 30 34 59099860 .1600204
0180 33 34 39 3b 20 5f 67 69 64 3d 47 41 31 2e 32 2e 349; __gi d=GA1.2.

Hypertext Transfer Protocol (http), 715 bytes

Packets: 1189 · Displayed: 169 (14.2%) · Dropped: 0 (0.0%) · Profile: Default

Security, Privacy, Integrity Issues

- When transmitting data in plain text over a network:
 - Online banking credentials can be intercepted and used to make fraudulent transactions
 - Email content can be read and disclosed to unauthorized parties
 - Text messages can be modified or forged
 - etc.

Ethical, Legal, Social Implications

- There are many software tools that make it easy to watch and/or modify traffic, both in wired and wireless environments
- Point to Ponder:
 - I can vs. I should vs. I should not
 - parental control vs. crime prevention vs. social protest vs. eliminating dissent
- Many jurisdictions (including Canada) view privacy as a right and make it illegal to spy on others (some exceptions apply)

Symmetric Encryption in Python

Setup on Ubuntu

```
sudo apt-get install python3-pip # if not already installed  
pipenv install cryptography # from inside the PyCharm terminal
```

Symmetric Encryption

```
#!/usr/bin/python3.11

from cryptography.fernet import Fernet

key = Fernet.generate_key()
print(key)

f = Fernet(key)
encrypted = f.encrypt(b"Hello World")
print(encrypted)

decrypted = f.decrypt(encrypted)
print(decrypted)
```


Symmetric Encryption

```
#!/usr/bin/python3.11
```

```
from cryptography.fernet import Fernet
```

```
key = Fernet.generate_key()
```

```
b'RE_9SKxNg5Aj8x7TteJTyzVZ87EDW5xM5c7XjDdazrGE='
```

```
f = Fernet(key)
```

```
encrypted = f.encrypt(b"Hello World")
```

```
print(encrypted)
```

```
decrypted = f.decrypt(encrypted)
```

```
print(decrypted)
```

Symmetric Encryption

```
#!/usr/bin/python3.11

from cryptography.fernet import Fernet

key = Fernet.generate_key()
print(key)

f = Fernet(key)
encrypted = f.encrypt(b"Hello World")

decrypted = f.decrypt(encrypted)
print(decrypted)
```

```
b'gAAAAABfX_WN08Z0U3pZXwzMI6wEKDPUALB_l6kafBEujKl8SZFCShmlolGACV5kLU4hkI-PjBt0Pd8KSdhPhauxmCIcj8qwsA=='
```

Symmetric Encryption

```
#!/usr/bin/python3.11

from cryptography.fernet import Fernet

key = Fernet.generate_key()
print(key)

f = Fernet(key)
encrypted = f.encrypt(b"Hello World")
print(encrypted)

decrypted = f.decrypt(encrypted)
print(decrypted)
```

```
b'Hello World'
```

Symmetric Encryption

- Can now encrypt communication between a server and a client
- This was the easy part

Symmetric Encryption

- Problem: We must keep the key secret at all times
 - How do we distribute the key to clients?
 - How do we protect the key once it is shared with a client?
- Problem: Without a timestamp or other nonce, the same plain text string will always return the same encrypted string, e.g.,
 - 'Yes', given the previous key, will ALWAYS generate
b'gAAAAABfX_Z0dZC-PO0maWc9TM-
LGWxuTKUwK1N6mHixvwjR6FBAMC3WW3rpbKcNge3WtcJcBho
33NsI3G7Eceb67EnhaMNK_Q=='
 - Given time, an attacker can figure this out
- These are hard problems! How do we solve them?

Expert Advice

- DON'T!

Expert Advice

- Solving these issues requires in-depth analysis
- Cryptographic expertise is required
- Involves the use of mathematical proofs, verification tools, and expert peer review
- Better to use established, well-tested libraries

Public Key Encryption in Python

Basic Setup

- For demonstration purposes, we'll create a self-signed certificate
- In a production environment, we would have to obtain a certificate from an official source (e.g., <https://letsencrypt.org/>)

Create a Configuration File (server.cnf)

```
[ req ]
prompt = no
distinguished_name = req_distinguished_name

[ req_distinguished_name ]
countryName = ca
stateOrProvinceName = British Columbia
localityName = Victoria
organizationName = Camosun College
organizationalUnitName = Department of Computer Science
emailAddress = horiem@camosun.bc.ca
commonName = horiem.ca
```

Modify /etc/hosts

```
127.0.0.1 localhost
127.0.1.1 ics226
127.0.0.1 horiem.ca
```

```
# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Generate a Self-Signed Certificate

```
openssl req -x509 -nodes -days 1095 -newkey rsa:4096 -config  
server.cnf -out server.crt -keyout server.key
```

Review the Certificate

```
openssl x509 -text -in server.crt -noout
```

Create a PEM File for the Server

```
cat server.crt server.key > server.pem
```

Protect this file! It contains the private key AND the public key

TLS Server

```
#!/usr/bin/python3.11

from socket import socket, AF_INET, SOCK_STREAM, SOL_SOCKET,
SO_REUSEADDR
from ssl import create_default_context, Purpose

HOST = ''
PORT = 12345
NUM_CLIENTS = 1

purpose = Purpose.CLIENT_AUTH
context = create_default_context(purpose, cafile=None)
context.load_cert_chain('server.pem')

listener = socket(AF_INET, SOCK_STREAM)
listener.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
listener.bind((HOST, PORT))
listener.listen(NUM_CLIENTS)
raw_sock, address = listener.accept()

ssl_sock = context.wrap_socket(raw_sock, server_side=True)
ssl_sock.sendall('Hello World!'.encode('utf-8'))

ssl_sock.close()
```

TLS Client

```
#!/usr/bin/env python3.11

from socket import socket, AF_INET, SOCK_STREAM
from ssl import create_default_context, Purpose

HOST = 'horiem.ca'
PORT = 12345
BUF_SIZE = 1024

purpose = Purpose.SERVER_AUTH
context = create_default_context(purpose, cafile='server.crt')

raw_sock = socket(AF_INET, SOCK_STREAM)
raw_sock.connect((HOST, PORT))

ssl_sock = context.wrap_socket(raw_sock, server_hostname=HOST)
while True:
    data = ssl_sock.recv(BUF_SIZE)
    if not data:
        break
    print(data)
```


Expert Advice

- DON'T!

Python Requests

Alternatives

- Prime Directive of Security: Only cryptography experts should write cryptography code; everybody else should use that code
- Can use *stunnel* to set up secure tunnels; Python socket code needs not be modified, except to make sure that the correct port is chosen so that packets travel through the tunnel
- Can also use *ssh* to tunnel traffic
- Can place *nginx* in front of the Python socket code to handle encryption and load balancing of HTTPS traffic

requests Library

- Can install and then use the *requests* library to make HTTPS client requests

```
import requests
r = requests.get('https://www.camosun.ca')
print(r.status_code)
print(r.text)
```

TLS

HTTPS Request

The image shows a Wireshark packet capture of an HTTPS request. The top pane displays a list of packets, with packet 57 selected, which is a TLSv1.2 Client Hello. The middle pane shows the details of this packet, including the Handshake Protocol: Client Hello. The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
51	3.898398	10.0.0.200	52.60.154.26	TCP	78	50047 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=67...
52	3.898777	52.60.154.26	10.0.0.200	TCP	66	80 → 50047 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W...
53	3.898816	10.0.0.200	52.60.154.26	TCP	54	50047 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
54	3.905523	10.0.0.200	52.60.154.26	TCP	78	50049 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=6...
55	3.905966	52.60.154.26	10.0.0.200	TCP	66	443 → 50049 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 ...
56	3.906027	10.0.0.200	52.60.154.26	TCP	54	50049 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
57	3.906075	10.0.0.200	52.60.154.26	TLSv1...	571	Client Hello
58	3.906372	52.60.154.26	10.0.0.200	TCP	60	443 → 50049 [ACK] Seq=1 Ack=518 Win=30336 Len=0
97	4.061372	52.60.154.26	10.0.0.200	TLSv1...	1514	Server Hello
98	4.061454	10.0.0.200	52.60.154.26	TCP	54	50049 → 443 [ACK] Seq=518 Ack=1461 Win=260672 Len=0
99	4.061642	52.60.154.26	10.0.0.200	TCP	1514	443 → 50049 [ACK] Seq=1461 Ack=518 Win=30336 Len=1460 [TCP segment of a ...
100	4.061644	52.60.154.26	10.0.0.200	TCP	1514	443 → 50049 [PSH, ACK] Seq=2921 Ack=518 Win=30336 Len=1460 [TCP segment ...
101	4.061694	10.0.0.200	52.60.154.26	TCP	54	50049 → 443 [ACK] Seq=518 Ack=4381 Win=259200 Len=0
102	4.061853	10.0.0.200	52.60.154.26	TCP	54	[TCP Window Update] 50049 → 443 [ACK] Seq=518 Ack=4381 Win=262144 Len=0
103	4.062206	52.60.154.26	10.0.0.200	TLSv1...	1303	Certificate, Server Key Exchange, Server Hello Done

Frame 57: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface en0, id 0

Ethernet II, Src: Apple_f3:ac:6f (f0:18:98:f3:ac:6f), Dst: RealtekU_64:30:1c (52:54:00:64:30:1c)

Internet Protocol Version 4, Src: 10.0.0.200, Dst: 52.60.154.26

Transmission Control Protocol, Src Port: 50049, Dst Port: 443, Seq: 1, Ack: 1, Len: 517

Transport Layer Security

TLSv1.2 Record Layer: Handshake Protocol: Client Hello

Content Type: Handshake (22)

Version: TLS 1.0 (0x0301)

Length: 512

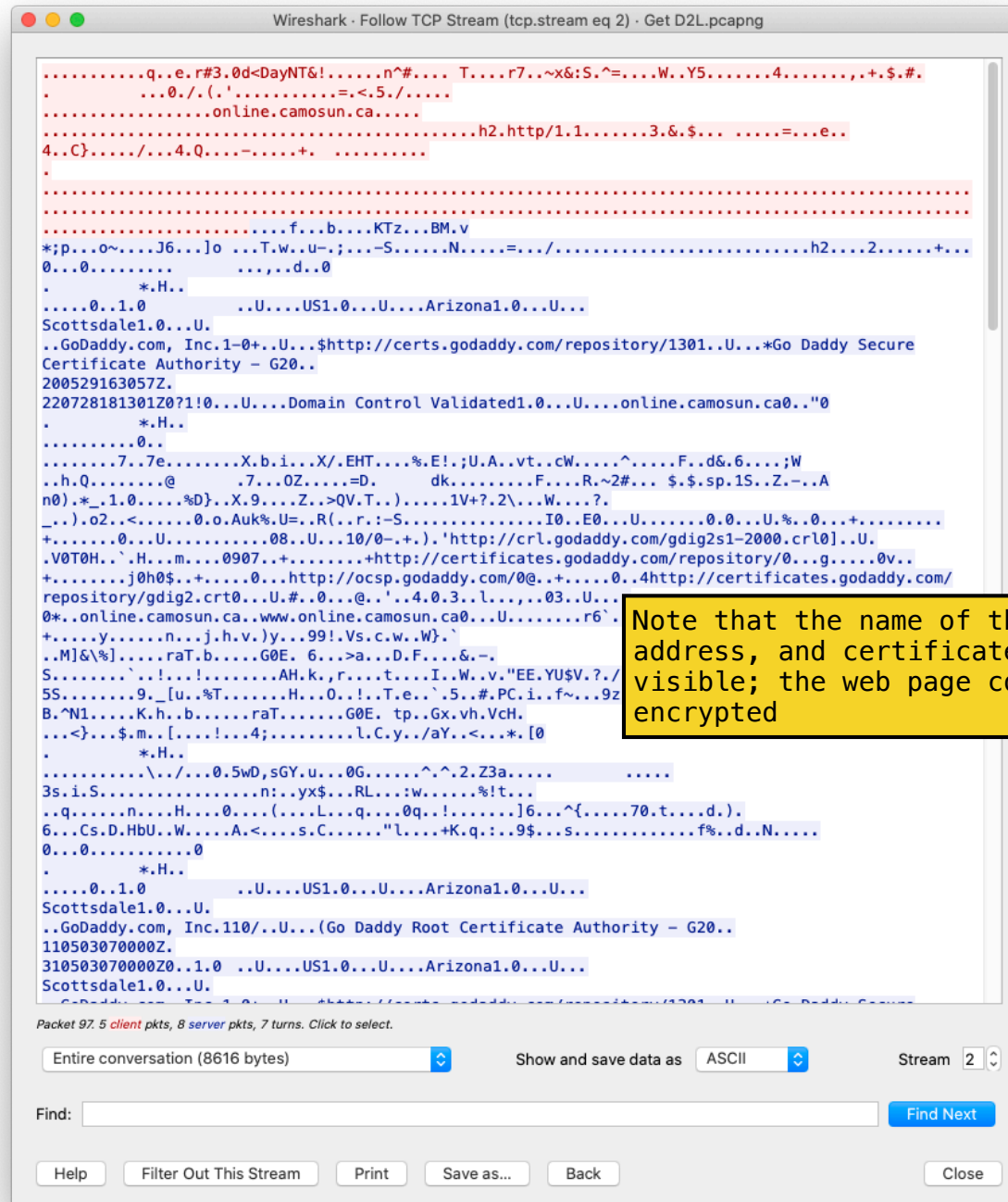
Handshake Protocol: Client Hello

Note that the name of the web server, its IP address, and certificate information is still visible; the web page content, however, will be encrypted

Record Layer (tls.record), 517 bytes

Packets: 283 · Displayed: 34 (12.0%) · Dropped: 0 (0.0%) · Profile: Default

HTTPS Request



Notes on DNS

- Even though we went to an HTTPS site, the DNS request for `www.camosun.ca` is not encrypted, meaning that a network observer knows that we are visiting a Camosun web site
- Even if we encrypted DNS traffic, the fact that your computer shortly thereafter connects to an IP associated with Camosun is not hidden

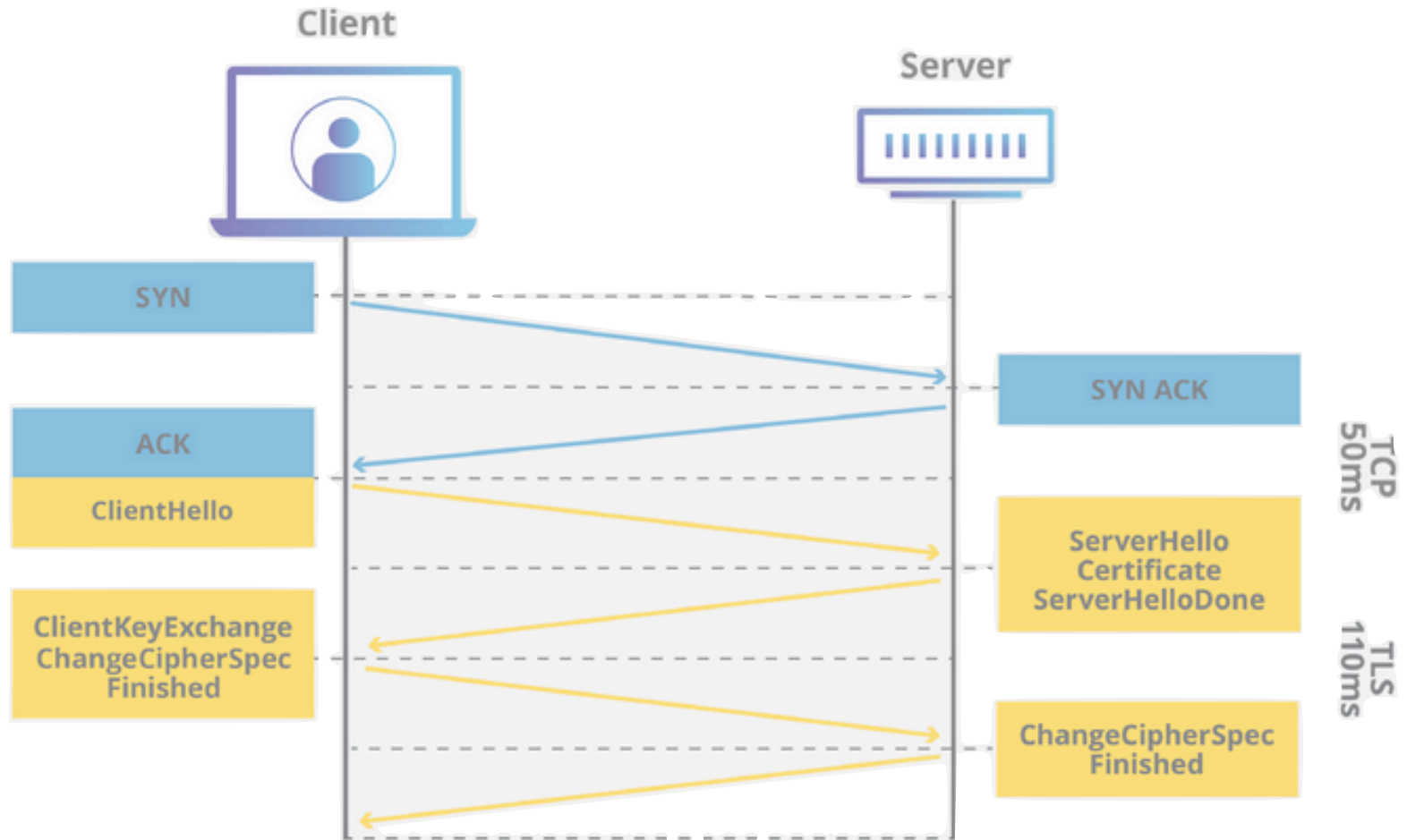
Notes on HTTPS

- Note that `www.camosun.ca` appears in plain text!
- The certificate check also shows that a GoDaddy certificate is being used
- The actual payload is encrypted, but based on data lengths, an observer may still figure out which web pages are being viewed
- ICS 228 is about exploring further weaknesses

TLS Handshake

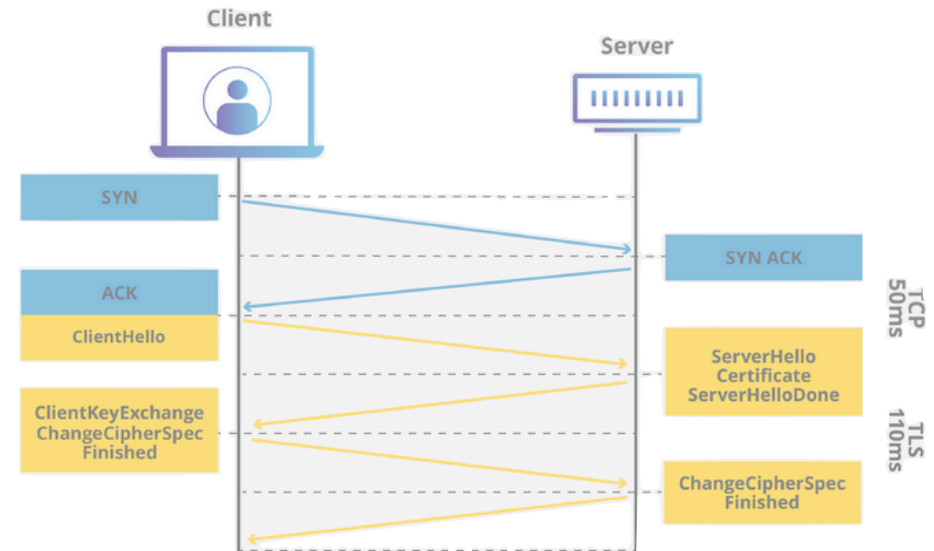
- When establishing an HTTPS connection, modern websites rely on TLS to provide the actual authentication and encryption
- During the actual setup (the TLS Handshake), the client and server establish
 - what version of TLS to use
 - what cipher suite to use
 - that the server could be authenticated successfully
 - session keys
- Details may differ, depending on what cipher was chosen

TLS 1.2 Handshake



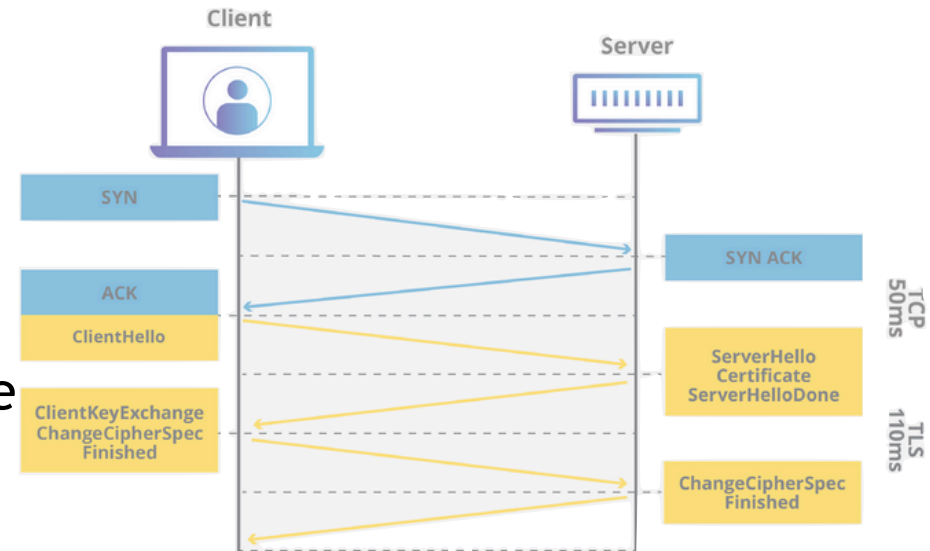
Client Hello

- Initiates the handshake
- Includes
 - TLS version that the client uses
 - Cipher suites that the client supports
 - A random byte string for later use

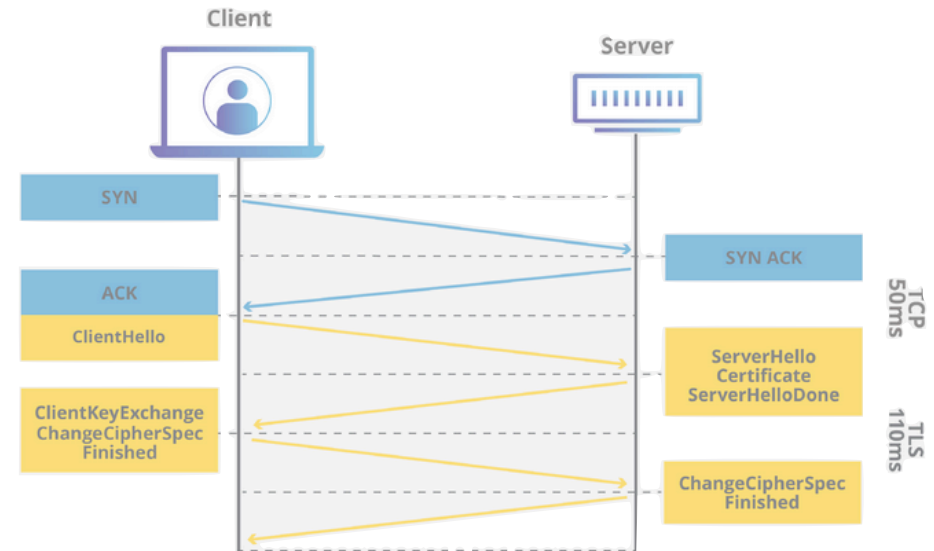


Server Hello, Certificate, Server Hello Done

- Replies to the client hello message
- Includes
 - Another random byte string for later use
 - The chosen cipher suite
 - The server certificate

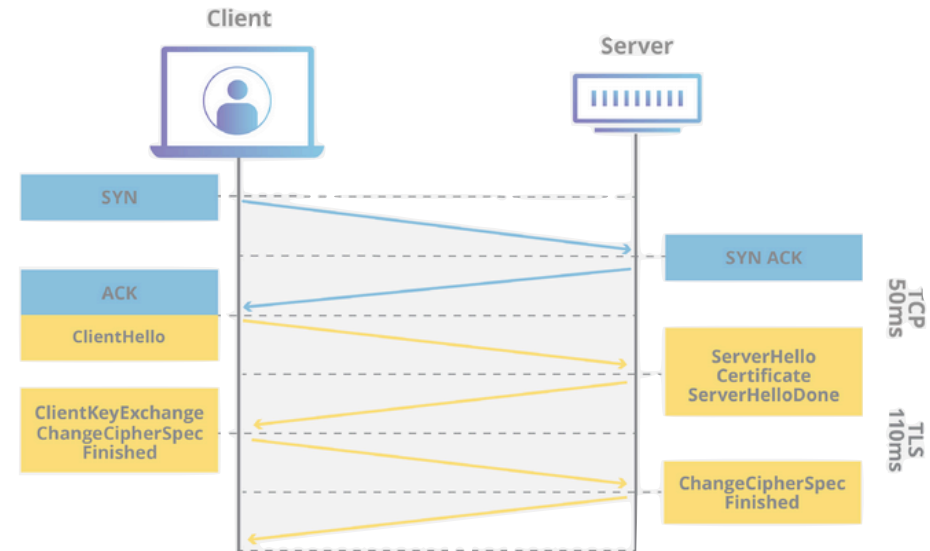


Client Key Exchange



- Provides the server with information needed to generate an encryption key for the session

Change Cipher Spec Finished



- Ready for encrypted session
- Change Cipher Spec possibly piggybacked on top of data
- This part of the handshake is already encrypted

Alert

- Encrypted
- Can be as simple as notifying a close of the connection
- Can also indicate a range of errors

TLS 1.3

- Reduces the number of exchanges
- Client Hello and Server Hello are still used
- Client Hello includes client key information based on an expected cipher protocol
- Assuming the server is willing to support the expected cipher, it will start encrypting data as part of its reply
- Subsequent connections even allow the Client Hello to include encrypted data. This is also known as 0-RTT (zero round trip time); some replay issues exist; see <http://blog.cloudflare.com/introducing-0-rtt>

Starting Points

- <https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/>
- <http://blog.fourthbit.com/2014/12/23/traffic-analysis-of-an-ssl-slash-tls-session/>
- Rhodes and Goerzen, Foundations of Python Network Programming, Apress, 2014

Key Skills

- Explain how a TLS session is established and terminated
- Explain the role of stunnel, nginx, requests
- Explain how to ensure security, privacy, and integrity of communication while recognizing the ethical, legal, and social implications of network programming