PKI on the Internet

Toni Bluher

2018 Women and Mathematics Program

Disclaimer: The opinions expressed are those of the writer and not necessarily those of NSA/CSS, the Department of Defense, or the U.S. Government.

Outline

Goal of this lecture: describe cryptography on the internet, specifically https.

1. Brief history of internet, http, and internet security
2. Live examples using Wire Shark
3. TLS
4. Cipher suites
5. Internet protocols: envelopes within envelopes

Brief history of internet

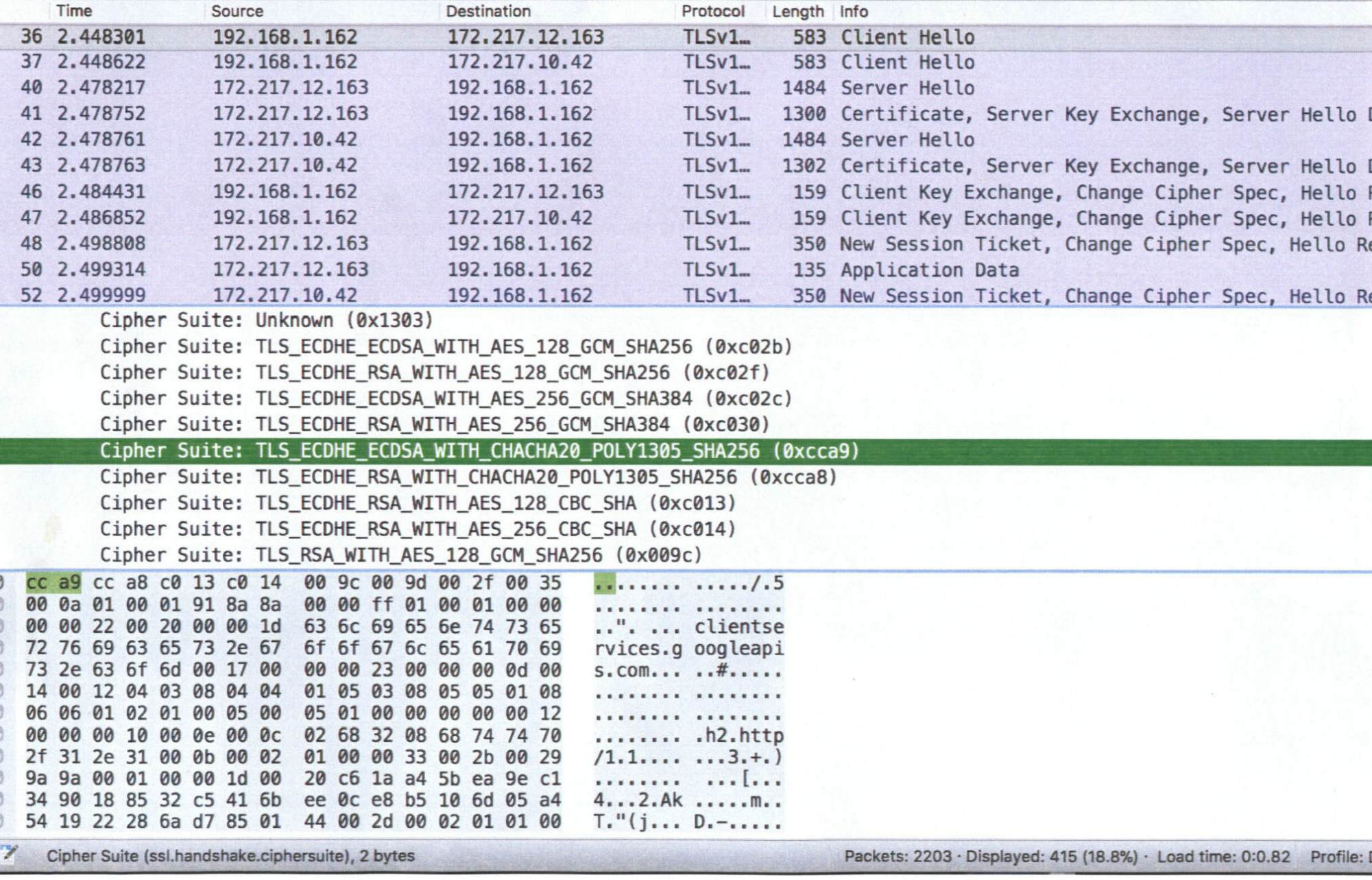
- 1960's: ARPANET was developed as a research tool for universities and military or government laboratories - this was "first internet".
- 1987 - NSFnet connected six large mainframes over phone lines; solicited contracts for higher throughput.
- HTTP = Hypertext Transfer Protocol invented at CERN in 1989.
- Browsers using HTTP were widely available by 1996.

Internet security

- Mid-1980's: ATHENA project at MIT – a non-PKI system of internet security called KERBEROS.
- 1992: "Pretty Good Privacy" by Phillip Zimmerman
- SSL = Secure Socket Layer was developed by Netscape in 1994 for endpoint-to-endpoint computer security using PKI. Later superseded by TLS = Transport Layer Security.
- SSL/TLS was first proposed as internet standard in 1999, updated in 2008 and 2011.
- HTTPS = HTTP + TLS
- IPsec provides security between routers; developed in mid-1990's. (Basis for Virtual Private Networks)

Wireshark

- Shows IP packets that enter or leave your computer
- Shows source and destination IP, current protocol, and other info
- Emily and I sat in cafe and brought up wireshark.
- Then did https://www.amazon.com in our browsers.
- Filtering on ssl brings up the packets that deal with PKI.
- Next two slides are screenshots of some output that we got.


Time Source Destination Protocol Length Info
36 2.448301 192.168.1.162 172.217.12.163 TLSv1… 583 Client Hello
37 2.448622 192.168.1.162 172.217.10.42 TLSv1… 583 Client Hello
40 2.478217 172.217.12.163 192.168.1.162 TLSv1… 1484 Server Hello
41 2.478752 172.217.12.163 192.168.1.162 TLSv1… 1300 Certificate, Server Key Exchange, Server Hello
42 2.478761 172.217.10.42 192.168.1.162 TLSv1… 1484 Server Hello
43 2.478763 172.217.10.42 192.168.1.162 TLSv1… 1302 Certificate, Server Key Exchange, Server Hello
46 2.484431 192.168.1.162 172.217.12.163 TLSv1… 159 Client Key Exchange, Change Cipher Spec, Hello
47 2.486852 192.168.1.162 172.217.10.42 TLSv1… 159 Client Key Exchange, Change Cipher Spec, Hello
48 2.498808 172.217.12.163 192.168.1.162 TLSv1… 350 New Session Ticket, Change Cipher Spec, Hello
50 2.499314 172.217.12.163 192.168.1.162 TLSv1… 135 Application Data
52 2.499999 172.217.10.42 192.168.1.162 TLSv1… 350 New Session Ticket, Change Cipher Spec, Hello
Cipher Suite: Unknown (0x1303)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
cc a9 cc a8 c0 13 c0 14 00 9c 00 9d 00 2f 00 35 ..../.5
00 0a 01 00 01 91 8a 8a 00 00 ff 01 00 01 00 00 ....
00 00 22 00 20 00 00 1d 63 6c 69 65 6e 74 73 65 ..". ... clientse
72 76 69 63 65 73 2e 67 6f 6f 67 6c 65 61 70 69 rvices.g oogleapi
73 2e 63 6f 6d 00 17 00 00 00 23 00 00 00 0d 00 s.com... ..#.....
14 00 12 04 03 08 04 04 01 05 03 08 05 05 01 08 ....
06 06 01 02 01 00 05 00 05 01 00 00 00 00 00 12 ....
00 00 00 10 00 0e 00 0c 02 68 32 08 68 74 74 70 ..... .h2.http
2f 31 2e 31 00 0b 00 02 01 00 00 33 00 2b 00 29 /1.1.... ...3.+.)
9a 9a 00 01 00 00 1d 00 20 c6 1a a4 5b ea 9e c1 ....[...
34 90 18 85 32 c5 41 6b ee 0c e8 b5 10 6d 05 a4 4...2.Ak .....m..
54 19 22 28 6a d7 85 01 44 00 2d 00 02 01 01 00 T."(j... D.-.....
Cipher Suite (ssl.handshake.ciphersuite), 2 bytes
Packets: 2203 · Displayed: 415 (18.8%) · Load time: 0:0.82
Profile:

ssl

	Time	Source	Destination	Protocol	Length	Info
62	27.644814	192.168.1.46	40.114.95.106	TLSv1.2	427	Application Data
63	27.645108	192.168.1.46	40.114.95.106	TLSv1.2	283	Application Data
278	43.835398	13.91.52.255	192.168.1.46	TLSv1.2	139	Encrypted Alert
279	43.836007	192.168.1.46	13.91.52.255	TLSv1.2	139	Encrypted Alert
351	48.438247	192.168.1.46	13.33.60.247	TLSv1.2	571	Client Hello
353	48.455307	13.33.60.247	192.168.1.46	TLSv1.2	1514	Server Hello
356	48.456625	13.33.60.247	192.168.1.46	TLSv1.2	1514	Certificate [TCP segment of a reassembled PDU]
357	48.457011	13.33.60.247	192.168.1.46	TLSv1.2	841	Certificate Status, Server Key Exchange, Server Hello
359	48.459265	192.168.1.46	13.33.60.247	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Ha
360	48.472274	13.33.60.247	192.168.1.46	TLSv1.2	296	New Session Ticket, Change Cipher Spec, Encrypted Han

```
    Length: 333
  ▲ Handshake Protocol: Server Key Exchange
      Handshake Type: Server Key Exchange (12)
      Length: 329
    ▲ EC Diffie-Hellman Server Params
        Curve Type: named_curve (0x03)
        Named Curve: secp256r1 (0x0017)
        Pubkey Length: 65
        Pubkey: 049f1adff70f8f7bee3da6754c08354bd3ae4aa2ecedd58f...
      ▷ Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
        Signature Length: 256
        Signature: 07fda17033fb9f4020b0747b2da781f42d7bfaab8777c189
```

```
0000  a4 db 30 53 22 97 00 7f  28 e0 59 c7 08 00 45 00   ..0S"... (.Y...E.
0010  03 3b be fa 40 00 f7 06  b5 d3 0d 21 3c f7 c0 a8   .;..@... ...!<...
0020  01 2e 01 bb d0 c0 19 73  b0 48 51 49 bb 08 50 18   .......s .HQI..P.
0030  00 77 e5 1b 00 00 82 01  b5 30 81 9e a2 16 04 14   .w...... .0......
0040  24 6e 2b 2d d0 6a 92 51  51 25 69 01 aa 9a 47 a6   $n+-.j.Q Q%i...G.
0050  89 e7 40 20 18 0f 32 30  31 38 30 34 32 30 30 36   ..@ ..20 18042006
0060  31 32 31 36 5a 30 73 30  71 30 49 30 09 06 05 2b   1216Z0s0 q0I0...+
0070  0e 03 02 1a 05 00 04 14  a8 7e 30 31 06 e4 e8 85   ........ .~01....
0080  65 cf e9 52 59 8f a6 da  7c 00 53 2f 04 14 24 6e   e..RY... |.S/..$n
0090  2b 2d d0 6a 92 51 51 25  69 01 aa 9a 47 a6 89 e7   +-.j.QQ% i...G...
00a0  40 20 02 10 07 dc 7d 69  44 60 e6 47 b7 6c 6e 78   @ ....}i D`.G.lnx
```

ame (841 bytes) | Reassembled TCP (484 bytes)

wireshark_BC817898-4A5F-4804-ABAB-C01161C1FD3E_20180420101025_a06324

TLS Handshake

We see the following interactions:

1. Client Hello
2. Server Hello
3. Certificate, Server Key Exchange, Server Hello Done
4. Client Key Exchange, Change Cipher Spec, Client Hello Done
5. Application Data

We are witnessing a TLS handshake!

TLS Handshake

- During the handshake, the client (my browser) and server (amazon.com) use public key cryptography to obtain a secret key that will be used to encrypt our session.
- One-sided authentication: only amazon has a public key certificate. (But I have a password and a credit card ...)
- Client Hello and Server Hello has main purpose of agreeing on a cipher suite.

Cipher Suites

- Cipher suite is a compatible choice of algorithms needed to establish a secure, authenticated connection. There are hundreds of them in the internet standards, each with its own 2-digit hex label.
- The first cipher suite that the client suggests is TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.
- Label is 0xc02b.
- Cipher suites are fully described by IETF in series of RFC's. (You can find them online.)

Cipher Suite example

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.

- ECDHE = Elliptic Curve Diffie-Hellman with Ephemeral key.
- ECDSA is elliptic curve digital signature algorithm. Server has a public key certificate for this.
- AES128 is AES with a 128-bit key, encrypts 128-bit blocks.
- GCM means “Galois counter mode”: encrypt *j*th 128-bit block of the message by XOR-ing it bitwise with $\mathrm{AES}_K(j)$.
- SHA256 is a hash function with 256-bit output; for use with the key exchange and signature.

Explanation of TLS Handshake

- Client Hello - My browser's favorite cipher suites
- Server Hello - Amazon picks one
- Certificate - Amazon sends its public key certificate
- Server Key Exchange - half of W/DH key exchange
- Client Key Exchange - other half of W/DH key exchange
- Change Cipher Spec - "Start encryption!"
- Application data - encrypted http session

What are all those numbers at the bottom?

- Wireshark screenshots show bytes at bottom.
- Byte = 8 bits = 2 hexadecimal digits, 0–255.
- Bytes in line 0a0 of first screenshot are: cc a9 cc a8 c0 13 c0 14 00 9c.
- These are the cipher suite codes!

Bytes in wireshark screenshot

- Second screenshot shows first 10 lines of the packet, with 16 bytes per line. What do these numbers mean?
- Note that the highlighted line at the top shows Source IP address is 13.33.60.247, destination IP address is 192.168.1.46.
- In hexadecimal these are 0d.21.3c.f7 and c0.a8.01.2e.
- These numbers can be found on lines 2-3 of the screenshot!

Bytes in second screenshot

- First 14 bytes are a4 db 30 ... 08 00. These probably belong to the link layer (and I don't understand them).
- Next 20 bytes are the IP header, which follow very exact specs given by the IP Protocol.
- 45 00 03 3b b3 fa 40 00 f7 06 b5 d3 0d 21 3c f7 c0 a8 01 2e
- To explain these bytes, check "IPV4" article, Section 3.1 on wikipedia.

Bytes in second screenshot

45 00 03 3b be fa 40 00 f7 06 b5 d3 0d 21 3c f7 c0 a8 01 2e

- 45: 4 is version number, i.e. IPV4. 5 is length of header, measured in units of 4 bytes. So it says the header has 20 bytes.
- 00: Used for VOIP, N/A for this example
- 03 3b is total length of the packet in bytes. $0x033b = 3 \cdot 16^2 + 3 \cdot 16 + 11 = 827$ bytes. Note that the highlighted line at the top of wireshark shows 841 bytes, which is 827+14. (Recall we had 14 initial bytes that we don't have a full explanation for.)
- be fa is an "identification field"

Bytes in second screenshot

45 00 03 3b be fa **40 00 f7 06 b5 d3 0d 21 3c f7 c0 a8 01 2e**

- 40 00 - high bit is always 0; next two bits (10) are flags; low 13 bits (=0) are fragment offset. This particular flag setting indicates “don’t fragment”.
- f7 is Time To Live (TTL); decrements by 1 each time packet is forwarded to a router.
- 06 is the protocol code for TCP (Transmission Control Protocol). Indicates that TCP will follow the IP header.
- b5 d3 is a 16-bit checksum for detecting errors. If error check fails, packet is discarded.
- 0d.21.3c.f7 and c0.a8.01.2e are source and destination IP addresses.

Next 20 bytes: TCP Header

- Next 20 bytes are TCP header. (Recall the protocol field 06 indicated TCP would follow the IP header.)
- TCP accepts stream of data, breaks it into chunks, and adds a TCP header to each chunk. The TCP header gives enough info (such as sequence numbers) for recipient to reassemble the data stream.
- TCP also keeps track of which packets were recieved, via ACK messages. Resends if needed.
- In our example, the data stream consists of the TLS handshake, followed by an encrypted HTTP session.
- For format of TCP header, look up TCP on wikipedia.

Next 20 bytes: TCP

01 bb d0 c0 19 73 b0 48 51 49 bb 08 50 18 00 77 e5 1b 00 00

- 01bb is source port (443), standard port for HTTPS.
- d0c0 is destination port, randomish.
- 1973b048 is sequence number, used by recipient to correctly order all packets that are received.
- 5149bb08 is ACK number. Indicates next sequence # expected (if ACK flag is set).
- 5 is size of TCP header in 4-byte units. I.e. this TCP header is 20 bytes.

TCP header

01 bb d0 c0 19 73 b0 48 51 49 bb 08 5**0 18 00 77 e5 1b 00 00**

- 018: Flag bits. In this example, the ACK and PSH flags are activated. PSH means push buffered data to receiving application. ACK means an acknowledgment is being sent using ACK number.
- 0077 is how many bytes the sender is willing to receive in a TCP packet.
- e51b is a checksum on the TCP header. (Packet will be discarded if checksum is wrong.)
- 0000 is urgent pointer (if URG flag is set).

After TCP header

82 01 b5 30 81 9e ...

- After the TCP header comes a chunk of the data stream.
- Our data stream contains the TLS handshake. The bytes shown are probably part of Amazon's certificate.
- The screenshot indicates that RSA is being used for signatures and that the signature has 256 bytes = 2048 bits. Thus, the above bytes are likely part of Amazon's 2048-bit RSA modulus.