

CS 305 Lab Tutorial

Lab 7 UDP&TCP&TLS

Dept. Computer Science and Engineering
Southern University of Science and Technology

Part A. UDP

- This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks.
- UDP assumes that the Internet Protocol (IP) is used as the underlying protocol.
- UDP is transaction oriented, and delivery and duplicate protection are NOT guaranteed.

<https://tools.ietf.org/html/rfc768>

a UDP segment(1)

No.	Time	Source	Destination	Protocol	Length	Info
2732	17.881663	10.21.3.80	172.18.1.92	DNS	83	Standard query 0x0004 A www.sustc.edu.cn.edu.cn
2733	17.924398	172.18.1.92	10.21.3.80	DNS	83	Standard query response 0x0004 Server failure A

> Frame 2732: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0

> Ethernet II, Src: IntelCor 5c:69:58 (90:61:ae:5c:69:58), Dst: JuniperN_aa:6d:c3 (2c:21:31:aa:6d:c3)

> **Internet Protocol Version 4**, Src: 10.21.3.80, Dst: 172.18.1.92

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 69

Identification: 0x4ca7 (19623)

> Flags: 0x0000

Time to live: 64

Protocol: UDP (17)

Header checksum: 0x732e [validation disabled]

[Header checksum status: Unverified]

Source: 10.21.3.80

Destination: 172.18.1.92

> **User Datagram Protocol**, Src Port: 64176, Dst Port: 53

Source Port: 64176

Destination Port: 53

Length: 49

Checksum: 0xc67f [unverified]

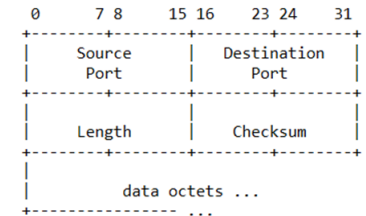
[Checksum Status: Unverified]

[Stream index: 468]

> Domain Name System (query)

UDP segment is the payload of IP package

UDP is identified by 17 in protocol field of IP package



User Datagram Header Format

a UDP segment(2)

Wireshark packet capture details for a DNS query over UDP.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
2732	17.881663	10.21.3.80	172.18.1.92	DNS	83	Standard query 0x0004 A www.sustc.edu.cn.edu.cn
2733	17.924398	172.18.1.92	10.21.3.80	DNS	83	Standard query response 0x0004 Server failure A

Packet 2732 Details:

- Frame 2732: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
- Ethernet II, Src: IntelCor_5c:69:58 (90:61:ae:5c:69:58), Dst: JuniperN_aa:6d:c3 (2c:21:31:aa:6d:c3)
- Internet Protocol Version 4, Src: 10.21.3.80, Dst: 172.18.1.92
- User Datagram Protocol, Src Port: 64176, Dst Port: 53
 - Source Port: 64176
 - Destination Port: 53
 - Length: 49
 - Checksum: 0xc67f [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 468]
- Domain Name System (query)
 - Transaction ID: 0x0004
 - Flags: 0x0100 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries

Packet 2733 Details:

- Response In: 2733

Hex Dump:

Offset	Hex	ASCII
0000	2c 21 31 aa 6d c3 90 61 ae 5c 69 58 08 00 45 00	,!1m..a .\iX..E.
0010	00 45 4c a7 00 00 40 11 73 2e 0a 15 03 50 ac 12	.EL...@ s...P..
0020	01 5c fa b0 00 35 00 31 c6 7f 00 04 01 00 00 01	.\...5.1
0030	00 00 00 00 00 00 03 77 77 77 05 73 75 73 74 63w ww.sustc
0040	03 65 64 75 02 63 6e 03 65 64 75 02 63 6e 00 00	.edu.cn.edu.cn..
0050	01 00 01	...

User Datagram Header Format:

0	7	8	15	16	23	24	31
Source Port				Destination Port			
Length				Checksum			
data octets ...							

While invoke an DNS query, this session is using UDP as transport protocol
You can use 'nslookup' or 'dig' to invoke an DNS query

Part B. TCP

- TCP a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.
- TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the Internet communication system.
 - Ports
 - connections
 - Flow control
 - Reliability

<https://tools.ietf.org/html/rfc793>

Part B.1 A TCP connection

tcp.stream eq 0

No.	Time	Source	Destination	Protoc	Info	
4	0.350305	192.168.88.149	14.215.177.39	TCP	60920 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1	connection establish
5	0.448978	14.215.177.39	192.168.88.149	TCP	80 → 60920 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM=1	
6	0.449087	192.168.88.149	14.215.177.39	TCP	60920 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0	
7	0.449211	192.168.88.149	14.215.177.39	HTTP	HEAD / HTTP/1.1	
8	0.487134	14.215.177.39	192.168.88.149	TCP	80 → 60920 [ACK] Seq=1 Ack=79 Win=24832 Len=0	http over tcp
9	0.493653	14.215.177.39	192.168.88.149	HTTP	HTTP/1.1 200 OK	
10	0.497383	192.168.88.149	14.215.177.39	TCP	60920 → 80 [FIN, ACK] Seq=79 Ack=333 Win=66304 Len=0	connection close
12	0.563547	14.215.177.39	192.168.88.149	TCP	80 → 60920 [ACK] Seq=333 Ack=80 Win=24832 Len=0	
13	0.566737	14.215.177.39	192.168.88.149	TCP	80 → 60920 [FIN, ACK] Seq=333 Ack=80 Win=24832 Len=0	
14	0.566805	192.168.88.149	14.215.177.39	TCP	60920 → 80 [ACK] Seq=80 Ack=334 Win=66304 Len=0	

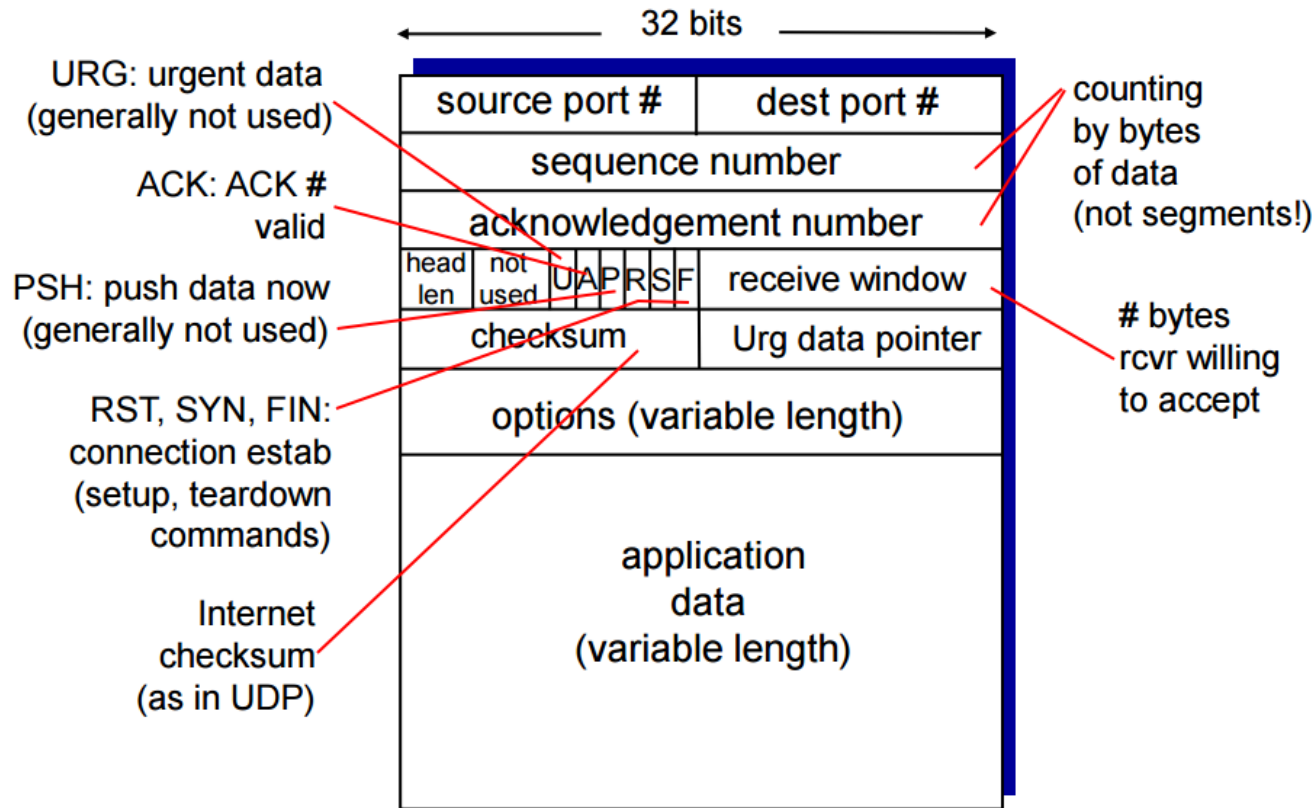
Source	Destination	Protoc	Info
192.168.88.149	14.215.177.39	TCP	60920 → 80

Source IP:192.168.88.149 port: 60920

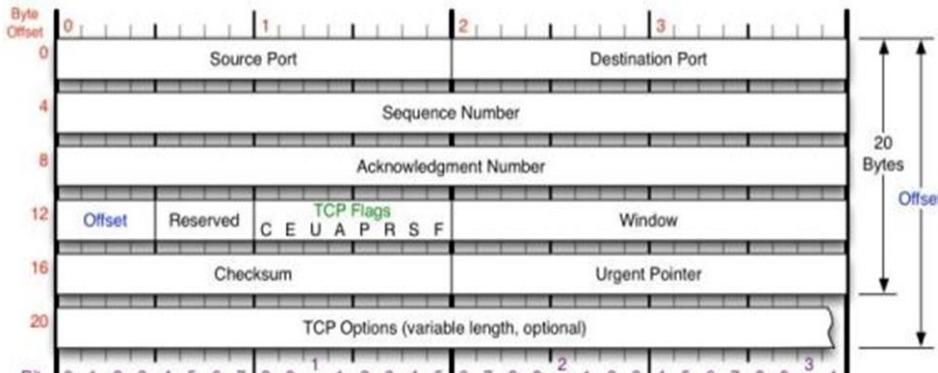
Destination IP:14.215.177.39 port:80

Tips: Using command 'curl' to invoke a http request which using TCP for transport
For example: curl -I www.baidu.com

Part B.2 TCP segment structure



'Header len'/'offset' field in TCP header



```

> Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window size value: 256
  [Calculated window size: 65536]
  [Window size scaling factor: 256]
  Checksum: 0x13ef [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  
```

head length is 20 byte
while there's no options

Data Offset: 4 bits

- The number of 32 bit words in the TCP Header. This indicates where the data begins.
- The TCP header (even one including options) is an integral number of 32 bits long.

```

> Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 0 (relative sequence number)
  [Next sequence number: 0 (relative sequence number)]
  Acknowledgment number: 0
  1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
  Window size value: 64240
  [Calculated window size: 64240]
  Checksum: 0x5335 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, N
  
```

32 bytes = 8 * 4 bytes

32 bytes = 20 (default length) + 12 (options length)

'Flags' in TCP header

```
Flags: 0x002 (SYN)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...0 = Acknowledgment: Not set
.... .... = Push: Not set
.... ..... = Reset: Not set
> ..... 1. = Syn: Set
```

```
Flags: 0x012 (SYN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... = Push: Not set
.... ..... = Reset: Not set
> ..... 1. = Syn: Set
.... .... = Fin: Not set
```

```
Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... = Push: Not set
.... ..... = Reset: Not set
.... .... = Syn: Not set
.... ..... = Fin: Not set
[TCP Flags: .....A....]
```

Control Bits:

URG: Urgent Pointer field significant

ACK: Acknowledgment field significant

PSH: Push Function

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: No more data from sender

```
Flags: 0x011 (FIN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... = Push: Not set
.... ..... = Reset: Not set
> ..... 1. = Fin: Set
[TCP Flags: .....A....F]
```

```
Flags: 0x019 (FIN, PSH, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... = Push: Set
.... ..... = Reset: Not set
.... .... = Syn: Not set
> ..... 1. = Fin: Set
```

Tips in Wireshark: Using 'tcp.flags.xxx==1' as filter to view the corresponding package
While xxx is the name of the flag, such as tcp.flags.syn==1

'Sequence number' and 'Ack number'(1)

Transmission is made reliable via the use of **sequence numbers** and **acknowledgments**.

- The sequence number **of the first octet of data in a segment** is transmitted with that segment and is called the segment **sequence number**.
- Segments also carry an **acknowledgment number** which is the sequence number of the next expected data octet of transmissions in the reverse direction.

When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer;

- when the acknowledgment for that data is received, the segment is deleted from the queue.
- If the acknowledgment is not received before the timer runs out, the segment is retransmitted.

An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken the responsibility to do so.

<https://tools.ietf.org/html/rfc793>

'Sequence number' and 'Ack number'(2)

```
Transmission Control Protocol, Src Port: 80, Dst Port: 54861, Seq: 81761, Ack: 333, Len: 1460
Source Port: 80
Destination Port: 54861
[Stream index: 2]
[TCP Segment Len: 1460]
Sequence number: 81761 (relative sequence number)
[Next sequence number: 83221 (relative sequence number)]
Acknowledgment number: 333 (relative ack number)
```

No.	Time	Source	Destination	Protoc	Info
234	10.752731	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=81761 Win=55296 Len=0
235	11.462632	128.119.245.12	192.168.88.149	TCP	80 → 54861 [ACK] Seq=81761 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
236	11.463266	128.119.245.12	192.168.88.149	TCP	80 → 54861 [ACK] Seq=83221 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
237	11.463358	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=84681 Win=52480 Len=0

54861->80: seq = 333 len=0

80->54861: ack=333+0 seq = 81761 len=1460

80->54861: ack=333+0 seq = 83221(81761+1460) len=1460

54861->80: Seq = 333(333+0) ack=84681(83221+1460) len=0

'Window' field in TCP header

- TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received.
- The window **indicates an allowed number of octets that the sender may transmit before receiving further permission.**

tcp.stream eq 2 && tcp.dstport==80

No.	Time	Source	Destination	Protoc	Info	
296	18.363331	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=127021 Win=9984 Len=0	the size of rcv window is dynamic changing
298	18.405271	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=128481 Win=8704 Len=0	
301	18.746754	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=131401 Win=5632 Len=0	
303	18.787241	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=132861 Win=4352 Len=0	
307	19.117577	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=135781 Win=1280 Len=0	

```

Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 0 (relative sequence number)
  [Next sequence number: 0 (relative sequence number)]
  Acknowledgment number: 0
  1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 64240
    [Calculated window size: 64240]
    Checksum: 0x5335 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale,
    > TCP Option - Maximum segment size: 1460 bytes
    > TCP Option - No-Operation (NOP)
    > TCP Option - Window scale: 8 (multiply by 256)
      Kind: Window Scale (3)
      Length: 3
      Shift count: 8
      [Multiplier: 256]
  
```

while in SYN, the multiplier on window is determined by 'window scale option'

```

No. Time Source Destination Protoc Info
296 18.363331 192.168.88.149 128.119.245.12 TCP 54861 → 80 [ACK] Seq=333 Ack=127021 Win=9984 Len=0
298 18.405271 192.168.88.149 128.119.245.12 TCP 54861 → 80 [ACK] Seq=333 Ack=128481 Win=8704 Len=0
301 18.746754 192.168.88.149 128.119.245.12 TCP 54861 → 80 [ACK] Seq=333 Ack=131401 Win=5632 Len=0
303 18.787241 192.168.88.149 128.119.245.12 TCP 54861 → 80 [ACK] Seq=333 Ack=132861 Win=4352 Len=0
307 19.117577 192.168.88.149 128.119.245.12 TCP 54861 → 80 [ACK] Seq=333 Ack=135781 Win=1280 Len=0

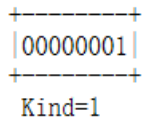
> Frame 296: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: IntelCor_Sc:69:58 (90:61:ae:5c:69:58), Dst: Routerbo_bd:b8:f5 (00:0c:42:bd:b8:f5)
> Internet Protocol Version 4, Src: 192.168.88.149, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 333, Ack: 127021, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 333 (relative sequence number)
  [Next sequence number: 333 (relative sequence number)]
  Acknowledgment number: 127021 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 39
    [Calculated window size: 9984]
    Window size scaling factor: 256
    Checksum: 0x234e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale,
    > TCP Option - Maximum segment size: 1460 bytes
    > TCP Option - No-Operation (NOP)
    > TCP Option - Window scale: 8 (multiply by 256)
      Kind: Window Scale (3)
      Length: 3
      Shift count: 8
      [Multiplier: 256]
  
```

$$9984 = 39(\text{size value}) * 256(\text{scaling factor})$$

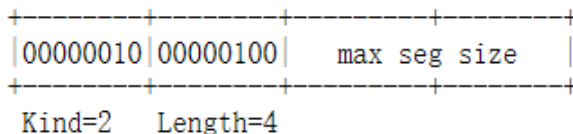
'Options' (variable) in TCP header

- May occupy **space at the end of the TCP header**
- **a multiple of 8 bits in length.**
- **No-operation** may be used between options, for example, to align the beginning of a subsequent option on a word boundary.

No-Operation



Maximum Segment Size



```

    Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Op
    ✓ TCP Option - Maximum segment size: 1460 bytes
      Kind: Maximum Segment Size (2)
      Length: 4
      MSS Value: 1460
    > TCP Option - No-Operation (NOP)
    ✓ TCP Option - Window scale: 8 (multiply by 256)
      Kind: Window Scale (3)
      Length: 3
      Shift count: 8
      [Multiplier: 256]
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    ✓ TCP Option - SACK permitted
      Kind: SACK Permitted (4)
      Length: 2
    ✓ [Timestamps]
    0000 00 0c 42 bd b8 f5 90 61 ae 5c 69 58 08 00 45 00 ..B....a..ix..E.
    0010 00 34 53 6b 40 00 40 06 58 97 c0 a8 58 95 80 77 .4Sk@.X.X.w
    0020 f5 0c d6 d4 00 50 d1 8b e9 fe 00 00 00 00 80 02 ...M.P.....
    0030 fa f0 53 35 00 00 02 04 05 b4 01 03 03 08 01 01 ..S5.....
    0040 04 02
  
```

mss

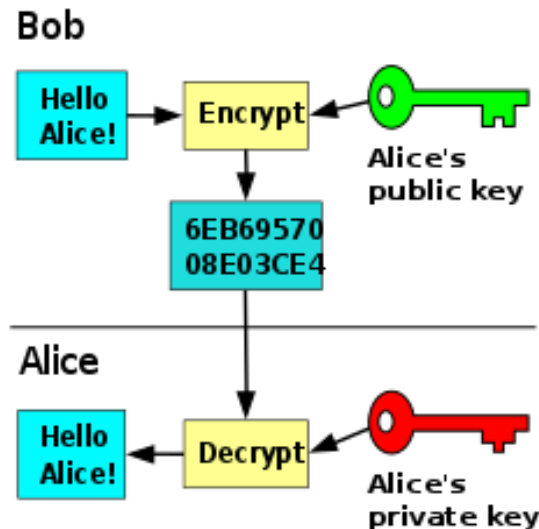
nop

Part C. TLS

- TLS stands for Transport Layer Security, which provide following features on TCP layer:
 - Encryption
 - Authentication of identity
 - Reliable transfer via integrity check (different from TCP reliable)

Part C. Public-key Cryptography

- Public-key cryptography, or asymmetric cryptography, is any cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner.



Hello! Let's start a encrypted conversation using TLS 1.2.

I want to talk to bank.com

I know the following cipher suites:

- ECDHE and RSA with 128bit AES in GCM mode and SHA256
- RSA with 128bit AES in GCM mode and SHA256

Here's a randomly chosen number:

3d86a5..04



Hi there, I think we can chat.



Let's use the cipher:

RSA with 128bit AES in GCM mode and SHA256

Here's my random number:

ca35f0..13

Here's my certificate chain:

[bank.com's certificate]

This certificate checks out: it was issued to bank.com and digitally signed by a certificate authority I trust.

Here's a secret encrypted with the RSA public key I took from your certificate:

[encrypted pre-master secret]

We can both derive the same key using this secret and the random numbers we exchanged.



I have decrypted the secret and derived the key.

From now on let's use the key to encrypt what we say.



[It's so great to speak privately]

[Can you get me the current balance of my checking account?]

[Sure thing, you have \$12.05 left in that account]

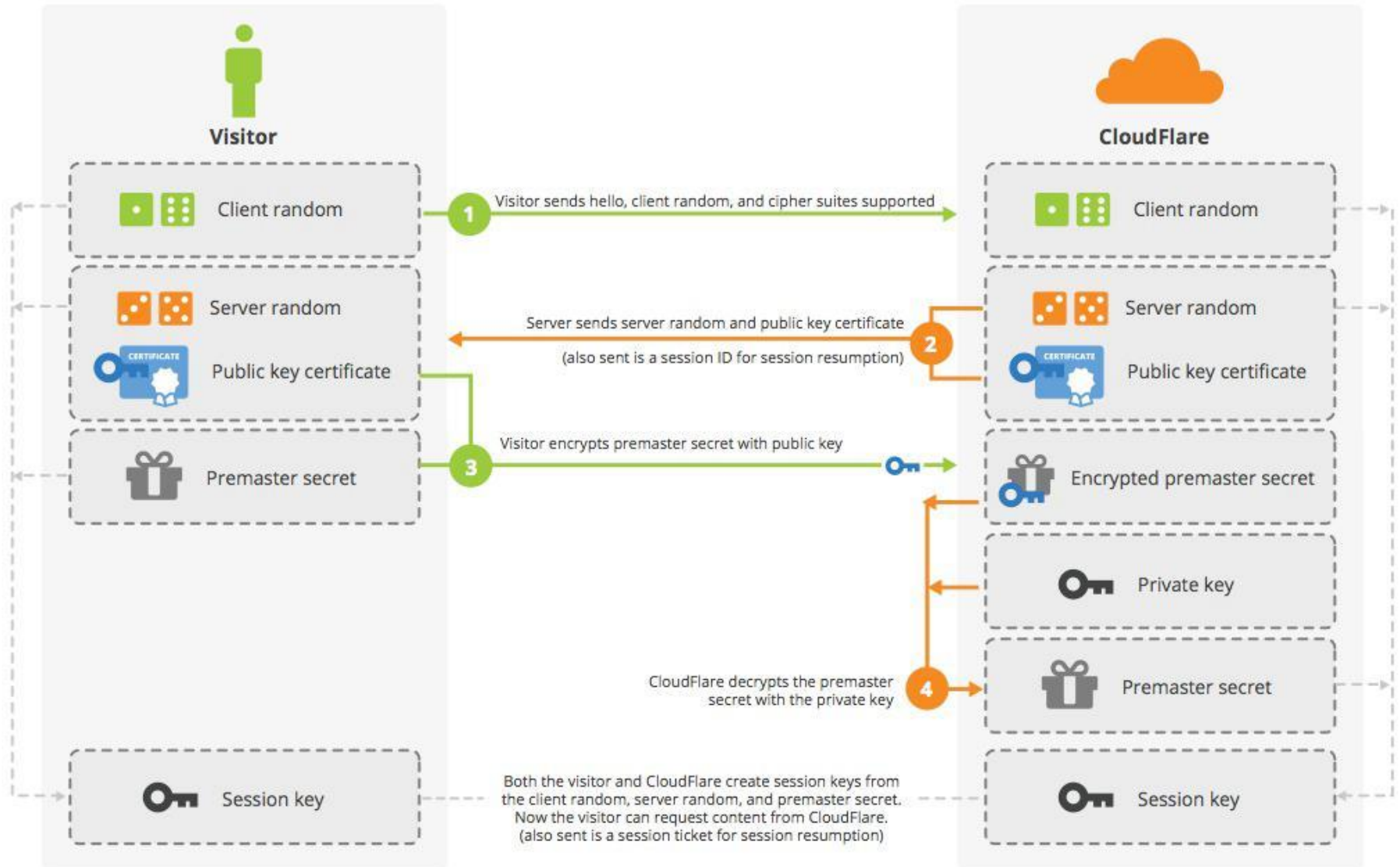


TLS Handshake (RSA without client cert)

- Client provide TLS version, a **Client random** and supported encryption method.
- Server check the TLS version and encryption method and provide server cert and **Server random**.
- Client validate the server cert and encrypt Premaster secret random using server public key.
- Server using private key to decrypt the **Premaster secret**.
- Server and Client using these three random numbers generate **Session key** standalone which will be used in the following session.

SSL Handshake (RSA) Without Keyless SSL

Handshake

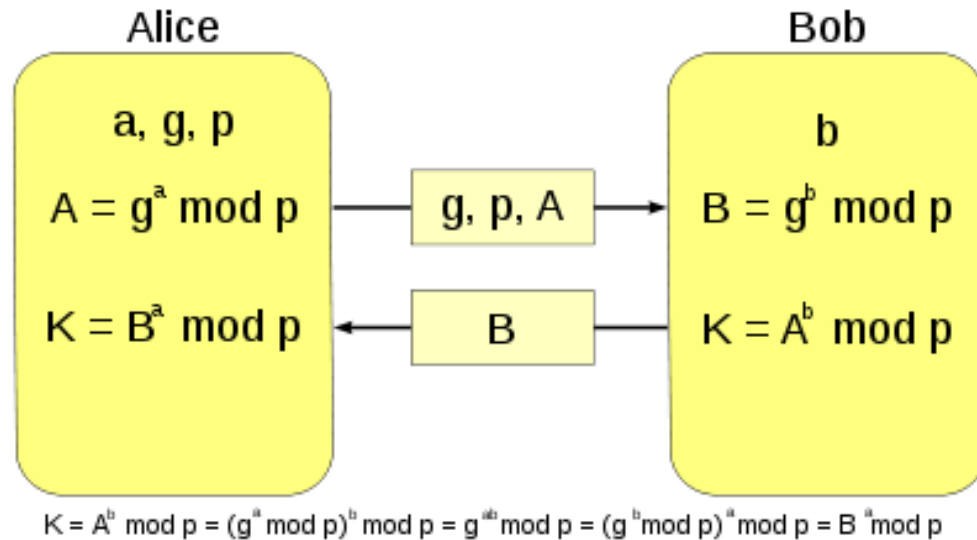


How could it be possible generate session key without encryption?

- If attacker is listening the TLS handshake, he will get the first two random numbers (**client random**, **server random**)
- The safety of session key depends on the **premaster secret**.
- If the RSA algorithm used is weak (using a 1024 bits cert example) can be cracked, the **premaster secret** can also be cracked. The entire session is not safe now.

Diffie-Hellman Key Exchange

- DH is a method of securely exchanging cryptographic keys over a public channel.



An DH Example

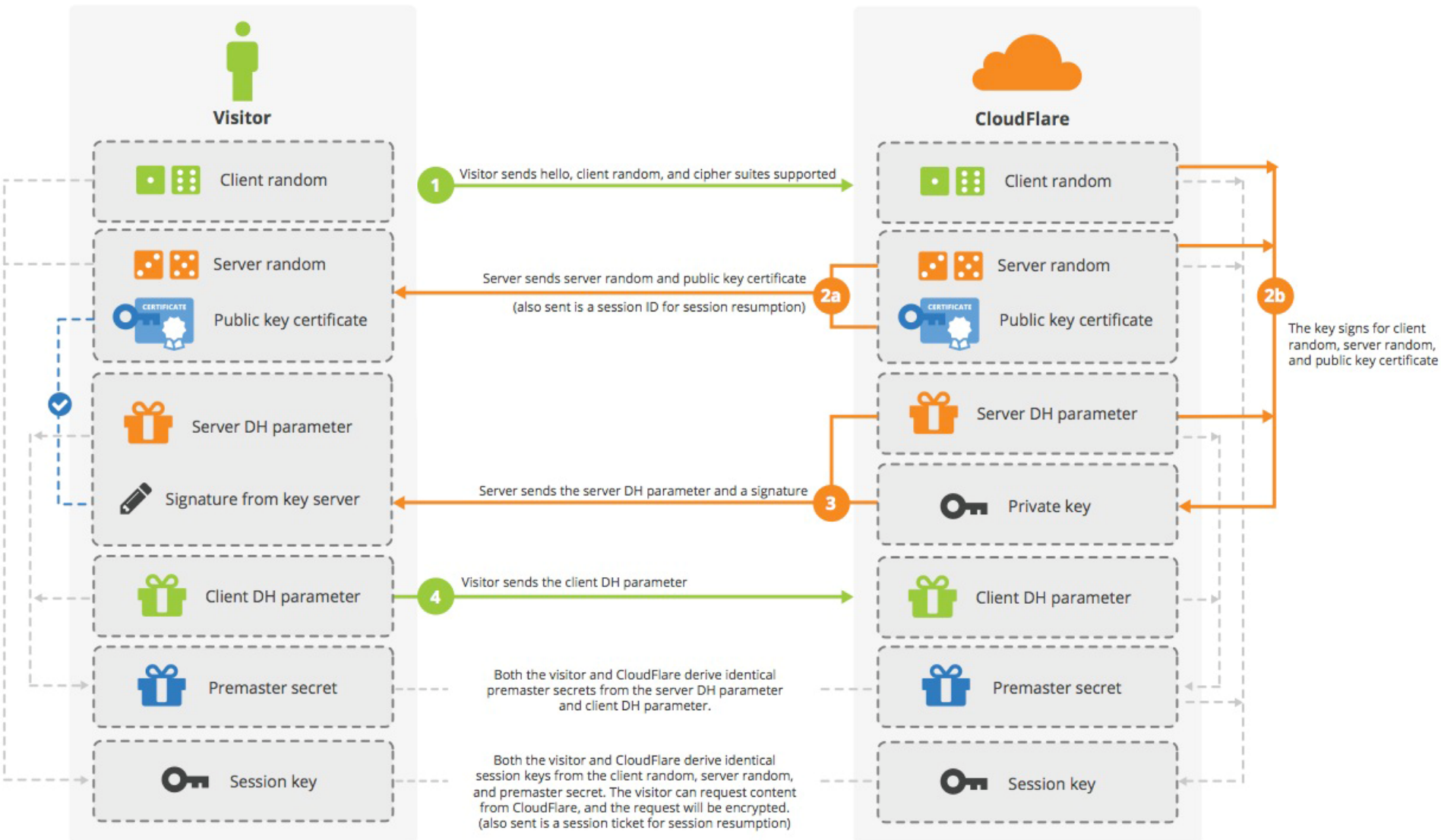
1. Alice and Bob agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $a = 4$, then sends Bob $A = g^a \bmod p$
 1. $A = 5^4 \bmod 23 = 4$
3. Bob chooses a secret integer $b = 3$, then sends Alice $B = g^b \bmod p$
 1. $B = 5^3 \bmod 23 = 10$
4. Alice computes $s = B^a \bmod p$
 1. $s = 10^4 \bmod 23 = 18$
5. Bob computes $s = A^b \bmod p$
 1. $s = 4^3 \bmod 23 = 18$
6. Alice and Bob now share a secret (the number 18).

TLS Handshake (DH without client cert)

- Client provide TLS version, a **Client random** and supported encryption method.
- Server check the TLS version and encryption method and provide server cert, **server random** and **server DH parameter** with signature.
- Client validate the server cert and send **client DH parameter**.
- Server and Client using the **DH parameters** to generate **premaster key** which is used for session key generation.

SSL Handshake (Diffie-Hellman) Without Keyless SSL

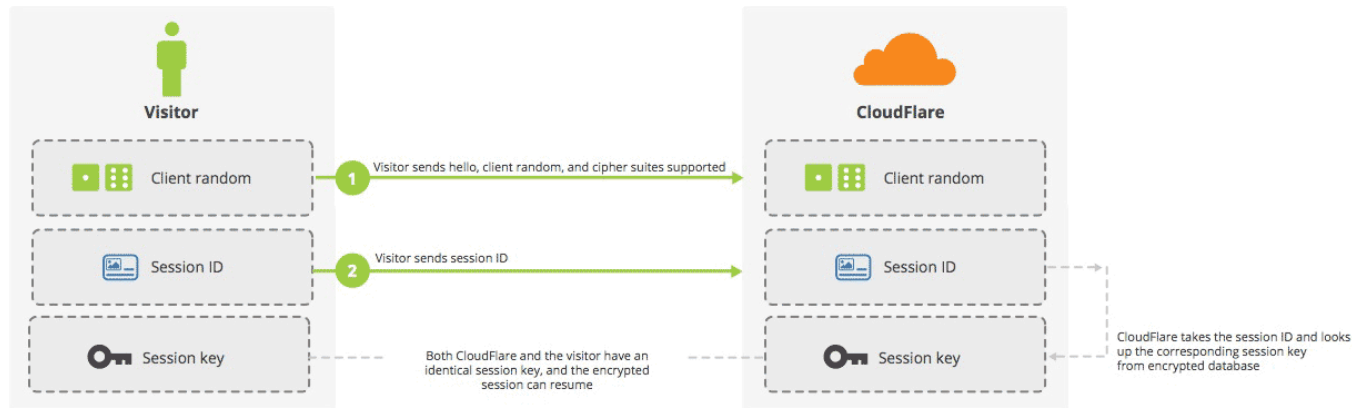
Handshake



Session resume

- If a TLS session is aborted, client can resume the session using session ID/session ticket.
 - No handshake needed (latency reduced)

Session resume with session ID



14801	26.204946	192.168.50.147	192.30.253.113	TLSv1.2	571 Client Hello
14815	26.709686	192.30.253.113	192.168.50.147	TLSv1.2	1514 Server Hello
14818	26.721227	192.30.253.113	192.168.50.147	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]

▼ Handshake Protocol: Client Hello

Handshake Type: Client Hello (1)

Length: 508

Version: TLS 1.2 (0x0303)

> Random: 9d840af65ff38f4ed04151b2545f2895c69009351152832d...

Client Random

Session ID Length: 32

Session ID: f77b857bdacd5caa7abb0cbe1271992ef4848dc2d325a8d5...

Cipher Suites Length: 36

▼ Cipher Suites (18 suites)

Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)

Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)

Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)

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_CHACHA20_POLY1305_SHA256 (0xcca9)

Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)

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_AES_256_CBC_SHA (0xc00a)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)

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_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)

Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)

Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)

Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)

Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)

Compression Methods Length: 1

> Compression Methods (1 method)

Extensions Length: 399

▼ Extension: server_name (len=15)

Type: server_name (0)

Length: 15

▼ Server Name Indication extension

Server Name list length: 13

Server Name Type: host_name (0)

Server Name length: 10

Server Name: github.com

14815	26.709686	192.30.253.113	192.168.50.147	TLSv1.2	1514 Server Hello
14818	26.721227	192.30.253.113	192.168.50.147	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]

- > Frame 14815: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
- > Ethernet II, Src: AsustekC_48:86:28 (18:31:bf:48:86:28), Dst: RivetNet_d3:eb:7f (9c:b6:d0:d3:eb:7f)
- > Internet Protocol Version 4, Src: 192.30.253.113, Dst: 192.168.50.147
- > Transmission Control Protocol, Src Port: 443, Dst Port: 14645, Seq: 1, Ack: 518, Len: 1460

Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 112

▼ Handshake Protocol: Server Hello

Handshake Type: Server Hello (2)

Length: 108

Version: TLS 1.2 (0x0303)

> Random: 3ce162659fede832ec967eaae51df4904e922733980b0a2b...

Server Random

Session ID Length: 32

Session ID: 66ed6a39d8a4fd9ada1769aac7a84376f7867fc6685fe48f...

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

Compression Method: null (0)

Extensions Length: 36

- > Extension: renegotiation_info (len=1)
- > Extension: server_name (len=0)
- > Extension: ec_point_formats (len=4)
- > Extension: extended_master_secret (len=0)

▼ Extension: application_layer_protocol_negotiation (len=11)

Type: application_layer_protocol_negotiation (16)

Length: 11

ALPN Extension Length: 9

▼ ALPN Protocol

ALPN string length: 8

ALPN Next Protocol: http/1.1

14818	26.721227	192.30.253.113	192.168.50.147	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]
14819	26.721368	192.30.253.113	192.168.50.147	TLSv1.2	100 Server Key Exchange, Server Hello Done
14821	26.726115	192.168.50.147	192.30.253.113	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

> Frame 14818: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
 > Ethernet II, Src: AsustekC_48:86:28 (18:31:bf:48:86:28), Dst: RivetNet_d3:eb:7f (9c:b6:d0:d3:eb:7f)
 > Internet Protocol Version 4, Src: 192.30.253.113, Dst: 192.168.50.147
 > Transmission Control Protocol, Src Port: 443, Dst Port: 14645, Seq: 2049, Ack: 518, Len: 1460
 > [3 Reassembled TCP Segments (3090 bytes): #14815(1343), #14816(588), #14818(1159)]

✓ Secure Sockets Layer

- ✓ TLSv1.2 Record Layer: Handshake Protocol: Certificate
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 3085
- ✓ Handshake Protocol: Certificate
 - Handshake Type: Certificate (11)
 - Length: 3081
 - Certificates Length: 3078
 - ✓ Certificates (3078 bytes)
 - Certificate Length: 1862
 - ✓ Certificate: 308207423082062aa00302010202100a0630427f5bbced69... (id-at-commonName=github.com,id-at-organizationName=GitHub, Inc.,id-at-...
 - > signedCertificate
 - > algorithmIdentifier (sha256WithRSAEncryption)
 - Padding: 0
 - encrypted: 700f5a96a758e5bf8a9da827982b007f26a907daba7b8254...
 - Certificate Length: 1210
 - ✓ Certificate: 308204b63082039ea00302010202100c79a944b08c119520... (id-at-commonName=DigiCert SHA2 Extended Validation Server CA,id-at-org...
 - > signedCertificate
 - > algorithmIdentifier (sha256WithRSAEncryption)
 - Padding: 0
 - encrypted: 9db6d09086e18602edc5a0f0341c74c18d76cc860aa8f04a...

14819	26.721368	192.30.253.113	192.168.50.147	TLSv1.2	100	Server Key Exchange, Server Hello Done
14821	26.726115	192.168.50.147	192.30.253.113	TLSv1.2	180	Client Key Exchange, Change Cipher Spec

- > Frame 14819: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
- > Ethernet II, Src: AsustekC_48:86:28 (18:31:bf:48:86:28), Dst: RivetNet_d3:eb:7f (9c:b6:d0:d3:eb:7f)
- > Internet Protocol Version 4, Src: 192.30.253.113, Dst: 192.168.50.147
- > Transmission Control Protocol, Src Port: 443, Dst Port: 14645, Seq: 3509, Ack: 518, Len: 46
- > [2 Reassembled TCP Segments (338 bytes): #14818(301), #14819(37)]

Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange

Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
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: 041addfedcf2891f68cc088af2a370c1532b33c43d1b7a1a...

server DH parameter

> Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
Signature Length: 256
Signature: 4d5f31b7eb32326db36b023500c44c5ac4bb7590f970b31b...

Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done

Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 4

▼ Handshake Protocol: Server Hello Done

Handshake Type: Server Hello Done (14)
Length: 0

14821	26.726115	192.168.50.147	192.30.253.113	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14829	26.821420	192.168.50.147	192.30.253.113	TLSv1.2	407 Application Data
14832	26.976118	192.30.253.113	192.168.50.147	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
14837	27.277675	192.30.253.113	192.168.50.147	TLSv1.2	1514 Application Data

> Frame 14821: 180 bytes on wire (1440 bits), 180 bytes captured (1440 bits) on interface 0
> Ethernet II, Src: RivetNet_d3:eb:7f (9c:b6:d0:d3:eb:7f), Dst: AsustekC_48:86:28 (18:31:bf:48:86:28)
> Internet Protocol Version 4, Src: 192.168.50.147, Dst: 192.30.253.113
> Transmission Control Protocol, Src Port: 14645, Dst Port: 443, Seq: 518, Ack: 3555, Len: 126
▼ Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 70

▼ Handshake Protocol: Client Key Exchange

Handshake Type: Client Key Exchange (16)

Length: 66

▼ EC Diffie-Hellman Client Params

Pubkey Length: 65

Pubkey: 042049f1720a9a9f5a2e357925528e547f75c1b9aa52af42...

client DH parameter

▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 40

Handshake Protocol: Encrypted Handshake Message

```

0000 18 31 bf 48 86 28 9c b6 d0 d3 eb 7f 08 00 45 00  .1.H.(.. ..E.
0010 00 a6 32 97 40 00 80 06 16 ef c0 a8 32 93 c0 1e  ..2.@... ..2...
0020 fd 71 39 35 01 bb c1 af 91 96 ce ef 49 b8 50 18  .q95.... ..I.P.
0030 01 00 0d b2 00 00 16 03 03 00 46 10 00 00 42 41  .....F...BA
0040 04 20 49 f1 72 0a 9a 9f 5a 2e 35 79 25 52 8e 54  .I.r... Z.5y%R.T
0050 7f 75 c1 b9 aa 52 af 42 68 46 e2 b2 63 91 98 57  .u...R.B hF..c..W
0060 a2 6d 18 d2 7b af f1 a1 92 bf 36 df ad 4b 2c 75  .m..{... ..6..K,u
0070 a2 53 22 63 96 db a9 b2 4a 42 fb e3 84 e2 6b 18  .S"c.... JB....k.
0080 ff 14 03 03 00 01 01 16 03 03 00 28 00 00 00 00  .....(....
0090 00 00 00 00 3b 93 f4 27 ae 57 96 5f c2 be c2 0d  ....;..' .W_....
00a0 8e 82 11 74 e2 3d df 45 62 93 07 69 35 bb a0 6f  ...t..E b..i5..o
00b0 af ff cf 5b                                     ...[

```

Practise 7.1

7.1 Select one UDP packet from your trace. From this packet,

- determine
 - 1) how many fields there are in the UDP header.
 - 2) the name and value of each fields in the UDP header.
 - 3) the length (in bytes) of each fields in the UDP header.
 - 4) What is the maximum number of bytes of a UDP packet ? (Hint: the answer to this question can be determined by your answer to 3) above)
 - 5) What is the largest possible destination port number? (Hint: same as the hint in 4) above.)
 - 6) What is the protocol ID for UDP in IP protocol?(Give your answer in both hexadecimal and decimal notation.)

Practise 7.2

- Finish the question 4, 5, 6, 7, 9, 10, 12 of Wireshark_TCP_v7.0.pdf