핀테크 산업 응용

엄현상 교수님 강의 실습 4차시
송만섭, 반지훈
2022/05/24

Contents

1 실습 1 - traceroute (tracert) 명령어 (Cont'd)

1) 구글 (google.com)에 어떻게 접속할 수 있는지 확인해보기

- 시작 메뉴 -> cmd 실행
- cmd에서 tracert google.com 입력

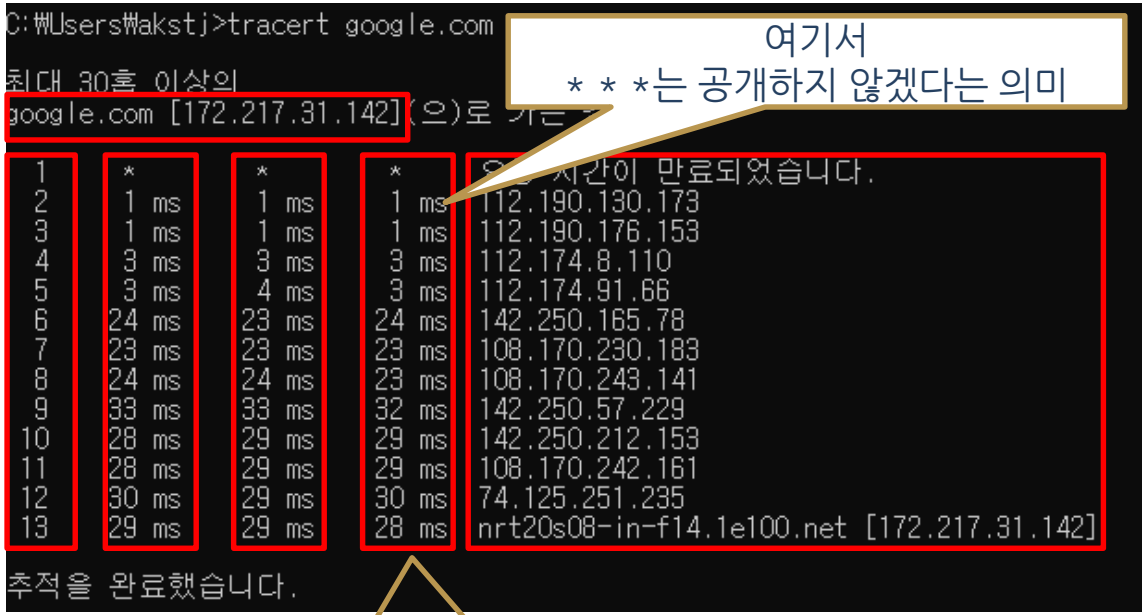
```
C:₩Users₩akstj>tracert google.com

최대 30홉 이상의
google.com [172.217.31.142](으)로 가는 경로 추적:

  1     *        *        *     요청 시간이 만료되었습니다.
  2     1 ms     1 ms     1 ms  112.190.130.173
  3     1 ms     1 ms     1 ms  112.190.176.153
  4     3 ms     3 ms     3 ms  112.174.8.110
  5     3 ms     4 ms     3 ms  112.174.91.66
  6    24 ms    23 ms    24 ms  142.250.165.78
  7    23 ms    23 ms    23 ms  108.170.230.183
  8    24 ms    24 ms    23 ms  108.170.243.141
  9    33 ms    33 ms    32 ms  142.250.57.229
 10    28 ms    29 ms    29 ms  142.250.212.153
 11    28 ms    29 ms    29 ms  108.170.242.161
 12    30 ms    29 ms    30 ms  74.125.251.235
 13    29 ms    29 ms    28 ms  nrt20s08-in-f14.1e100.net [172.217.31.142]

추적을 완료했습니다.
```

1 실습 1 - traceroute (tracert) 명령어 (Cont'd)



① 목적지 주소
② 컴퓨터에서 목적지 주소인 google.com의 웹서버까지 가는데 거친 홉 수
③ ④ ⑤ RTT (Round Trip Time, 왕복 시간)
3개인 이유: 3개의 패킷들을 전송함 (reliability)
⑥ 해당 홉의 호스트 이름과 IP

1) tracert 동작 원리

1. PC1에서 TTL 값을 1 증가시켜 IP packet을 R1으로 전송
2. R1은 수신된 IP 패킷을 다음 라우터로 전달하기 전에 IP header의 TTL 값을 1 감소
3. TTL 값이 0이 되면 송신측(PC1)에 time exceed error 패킷으로 응답
4. PC1은 TTL 1 더 증가시킨 TTL=2인 IP packet을 R1으로 전송
5. R1에서 R2로 전송 (TTL = 0)
6. 이와 같이, 여러 번의 try and error들을 통해 PC1에서 PC2로 패킷이 전달됨

But, 유독 오래 걸린다? -> network bottleneck

1 traceroute (tracert) 명령어 예시

1) Example of a Good Result

```
1    13 ms    22 ms    25 ms  52.93.114.64
1    <1 ms    <1 ms     1 ms  54.239.108.177
2    <1 ms    <1 ms    <1 ms  64.125.12.29
3     2 ms     2 ms     2 ms  64.125.31.41
4     65 ms    65 ms    65 ms  64.125.30.248
5     65 ms    66 ms    65 ms  64.125.29.45
6     65 ms    65 ms    65 ms  64.125.28.103
7     82 ms    65 ms    65 ms  64.125.26.183
8     65 ms    65 ms    65 ms  64.125.31.49
9     *        *        *      216.200.159.42
10    64 ms    65 ms    64 ms  64.93.85.25
11    66 ms    66 ms    66 ms  64.93.75.18
12    65 ms    65 ms    65 ms  72.10.63.118
13    64 ms    64 ms    64 ms  72.47.244.140
```

2) Example Tracert Result of a Failed Hop

```
1    13 ms    22 ms    25 ms  52.93.114.64
1    <1 ms    <1 ms     1 ms  54.239.108.177
2    <1 ms    <1 ms    <1 ms  64.125.12.29
3     2 ms     2 ms     2 ms  64.125.31.41
4     65 ms    65 ms    65 ms  64.125.30.248
5     65 ms    66 ms    65 ms  64.125.29.45
6     65 ms    65 ms    65 ms  64.125.28.103
7     82 ms    65 ms    65 ms  64.125.26.183
8     65 ms    65 ms    65 ms  64.125.31.49
9     *        *        *      Request timed out
10    *        *        *      Request timed out
11    *        *        *      Request timed out
12    *        *        *      Request timed out
13    *        *        *      Request timed out
```

3) Example Tracert Result of Routing Loop Problem

```
1    13 ms    22 ms    25 ms  52.93.114.64
1    <1 ms    <1 ms     1 ms  54.239.108.177
2    <1 ms    <1 ms    <1 ms  64.125.12.29
3     2 ms     2 ms     2 ms  64.125.31.41
4     65 ms    65 ms    65 ms  64.125.30.248
5     65 ms    66 ms    65 ms  64.125.29.45
6     65 ms    65 ms    65 ms  64.125.28.103
7     82 ms    65 ms    65 ms  64.125.26.183
8     65 ms    65 ms    65 ms  64.125.31.49
9     65 ms    65 ms    72 ms  216.200.159.42
10    64 ms    65 ms    64 ms  64.125.31.49
11    66 ms    66 ms    66 ms  216.200.159.42
…
…
…
29    64 ms    65 ms    66 ms  216.200.159.42
30    64 ms    64 ms    64 ms  64.125.31.49
```

1 tracert options

구문

복사

```
tracert [/d] [/h <maximumhops>] [/j <hostlist>] [/w <timeout>] [/R] [/S <srcaddr>] [/4][/6] <targetname>
```

매개 변수

매개 변수	Description
/d	중간 라우터의 IP 주소를 해당 이름으로 확인하려는 시도를 중지합니다. 이렇게 하면 결과의 반환 속도가 빨라지게 될 수 있습니다.
/h `<maximumhops>`	대상(대상)을 검색할 경로의 최대 홉 수를 지정합니다. 기본값은 30 홉입니다.
/j `<hostlist>`	에코 요청 메시지가 IP 헤더에서 중간 대상 집합이 지정된 느슨한 소스 경로 옵션을 사용하도록 지정 `<hostlist>`합니다. 느슨한 원본 라우팅을 사용 하 여 연속 중간 대상 하나 또는 여러 개의 라우터에 의해 분리할 수 있습니다. 목록의 최대 주소 또는 이름 수는 9개입니다. 공백 `<hostlist>` 으로 구분된 일련의 IP 주소(점선 소수점 표기법)입니다. IPv4 주소를 추적 하는 경우에이 매개 변수를 사용 합니다.
/w `<timeout>`	지정된 에코 요청 메시지에 해당하는 ICMP 시간이 초과되거나 에코 회신 메시지가 수신될 때까지 대기하는 시간(밀리초)을 지정합니다. 제한 시간 내에 수신되지 않으면 별표(`*`)가 표시됩니다. 기본 제한 시간은 4000 (4 초)입니다.
/R	IPv6 라우팅 확장 헤더를 사용하여 대상을 중간 대상으로 사용하고 역방향 경로를 테스트하여 로컬 호스트에 에코 요청 메시지를 보내도록 지정합니다.
/S `<srcaddr>`	에코 요청 메시지에 사용할 원본 주소를 지정합니다. IPv6 주소를 추적 하는 경우에이 매개 변수를 사용 합니다.
/4	tracert.exe 이 추적에 IPv4만 사용할 수 있도록 지정합니다.
/6	tracert.exe 이 추적에 IPv6만 사용할 수 있도록 지정합니다.
`<targetname>`	식별 된 대상 지정 IP 주소 또는 호스트 이름으로 합니다.
/?	명령 프롬프트에 도움말을 표시합니다.

source: https://docs.microsoft.com/ko-kr/windows-server/administration/windows-commands/tracert

1 나에게 맞는 DNS 찾기

Best Free & Public DNS Servers

Provider	Primary DNS	Secondary DNS
Google	8.8.8.8	8.8.4.4
Quad9	9.9.9.9	149.112.112.112
OpenDNS Home	208.67.222.222	208.67.220.220
Cloudflare	1.1.1.1	1.0.0.1
CleanBrowsing	185.228.168.9	185.228.169.9
Alternate DNS	76.76.19.19	76.223.122.150
AdGuard DNS	94.140.14.14	94.140.15.15

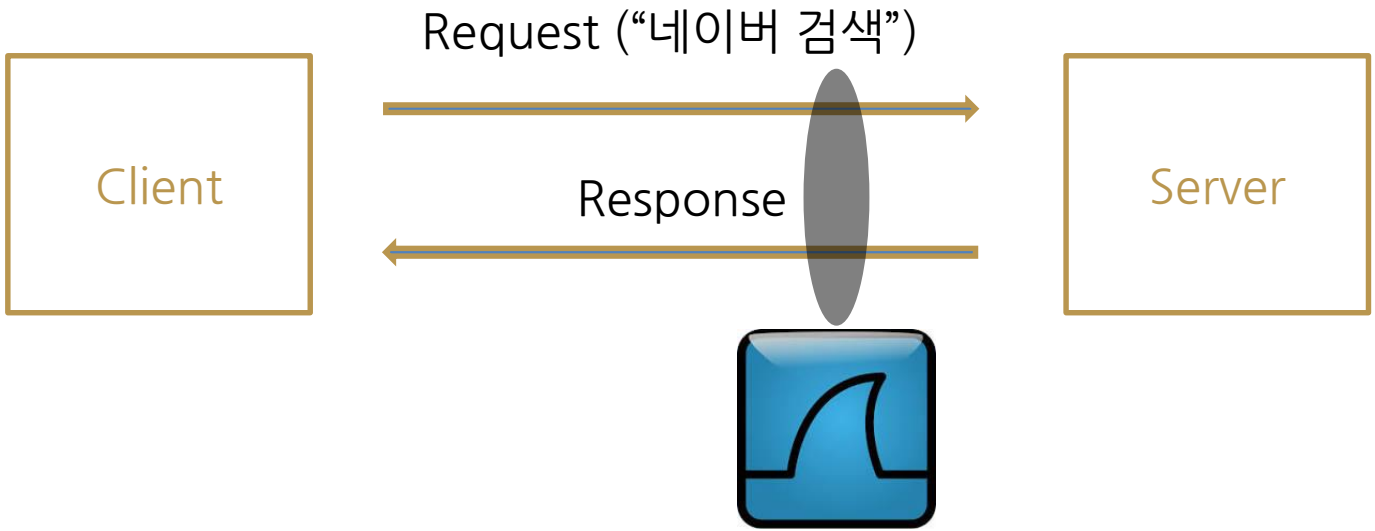
한번씩 재미삼아 해보아요!!

2 와이어샤크 (Wireshark)

1) 와이어샤크(Wireshark)란?

- 네트워크 패킷을 캡처하고 분석하는 오픈소스 도구
- 패킷 분석에 가장 널리 사용되는 프로그램
- Windows, Linux, Mac 등 다양한 OS 호환 가능
- Live Capture 및 Offline 분석 가능
- 암호화된 패킷 분석 가능
- 필터링 가능 - 원하는 패킷만 캡처 가능
- 강력하고 쉬운 사용법때문에 해킹뿐만 아니라 보안 취약점 분석, 보안 컨설팅, 개인정보 영향평가 등 여러 분야들에서 폭 넓게 사용됨

2 와이어샤크 (Wireshark) (Cont'd)



2 와이어샤크 (Wireshark) (Cont'd)

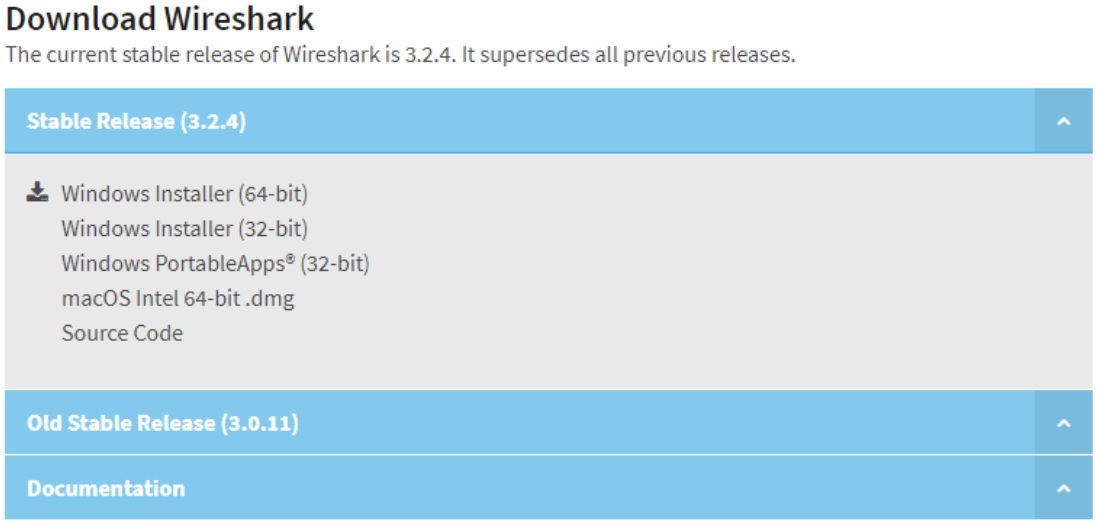
1) Wireshark를 사용하여 네트워크로 전송되는 패킷(네트워크상의 데이터)을 수신 및 저장

- PCAP이라는 포맷으로 파일을 저장
- PCAP 은 Packet Capture의 약자로 네트워크 트래픽을 캡처하는 API구성
- Wireshark는 자체 프로그램으로 네트워크 트래픽을 캡처하는 것이 아니고, 운영체제에서 지원하는 캡처 라이브러리를 이용하여 수집
  - 유닉스 : libpcap
  - 윈도우 : Winpcap

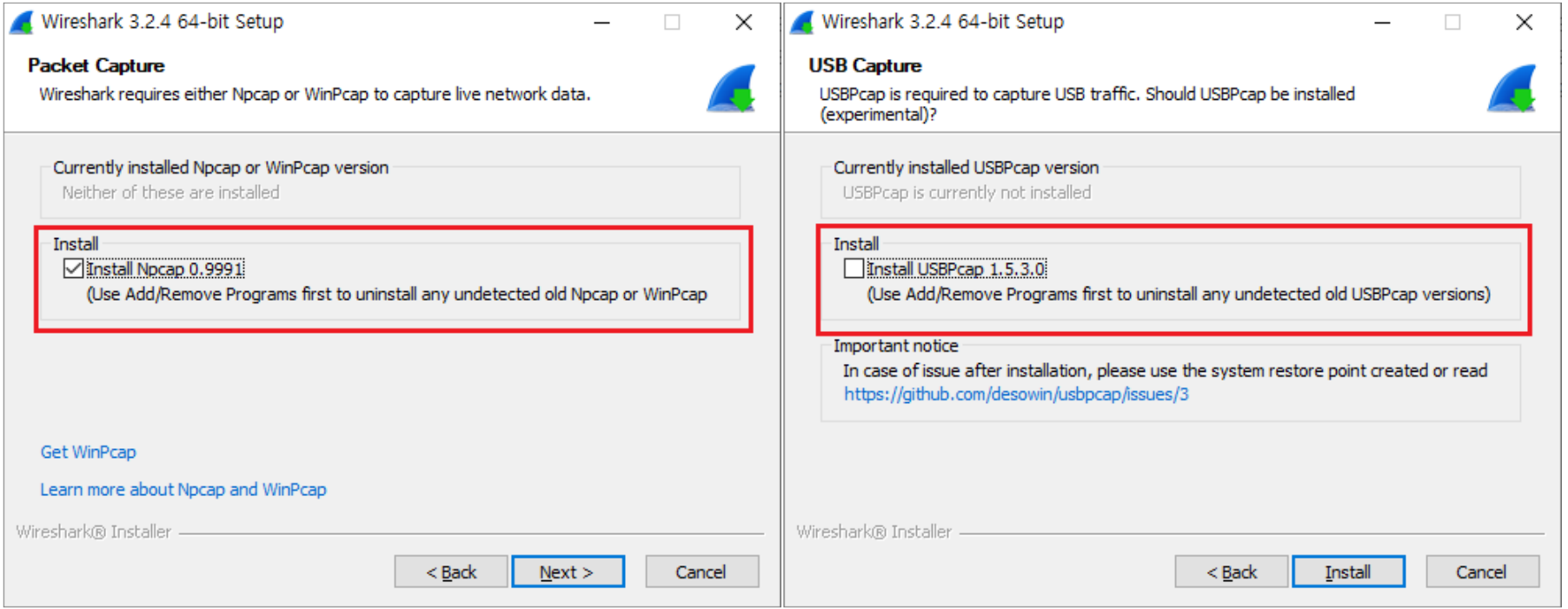
1) Wireshark 설치

> 본인 환경에 적합한 파일 다운로드

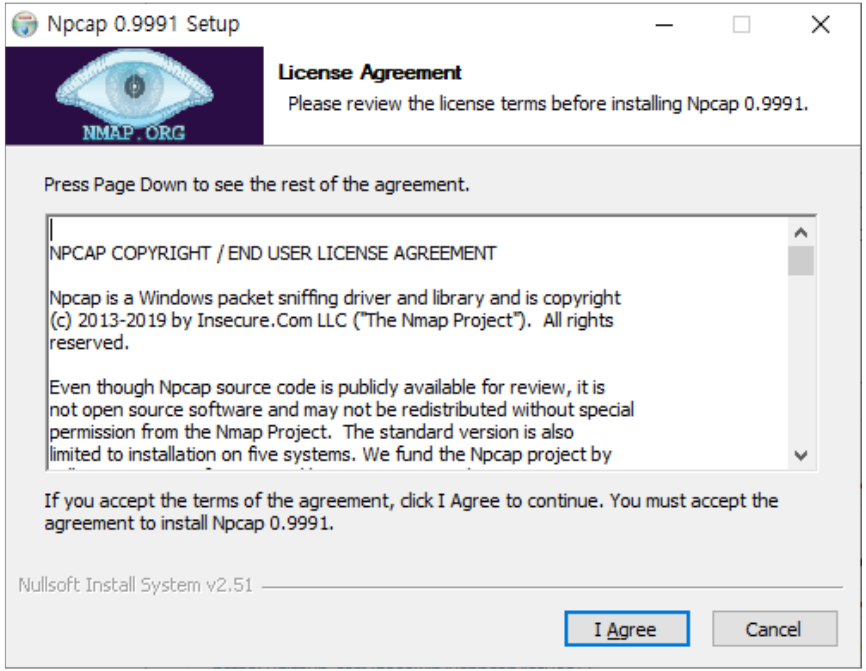
- (link): https://www.wireshark.org/download.html

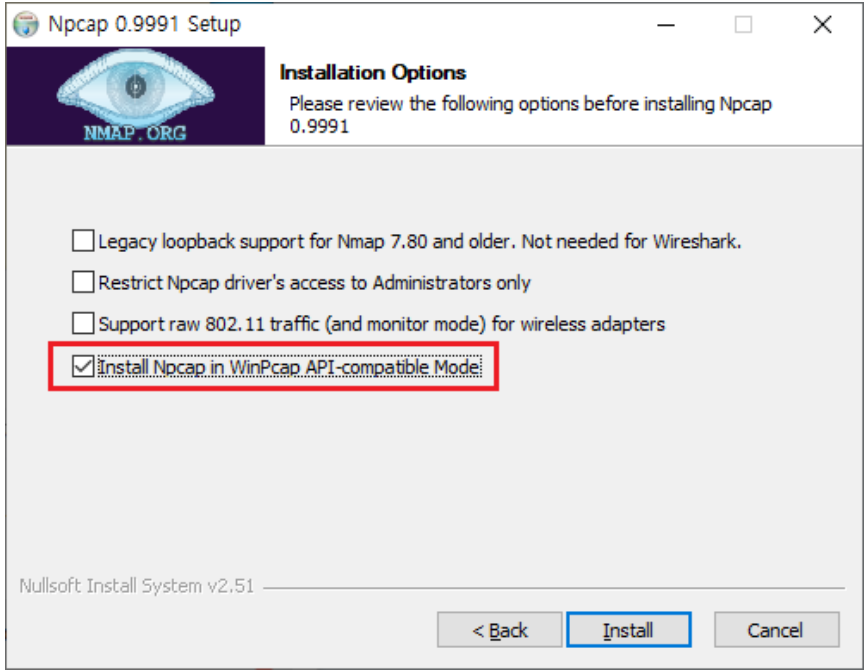


2 와이어샤크 (Wireshark) 설치 (Cont'd)



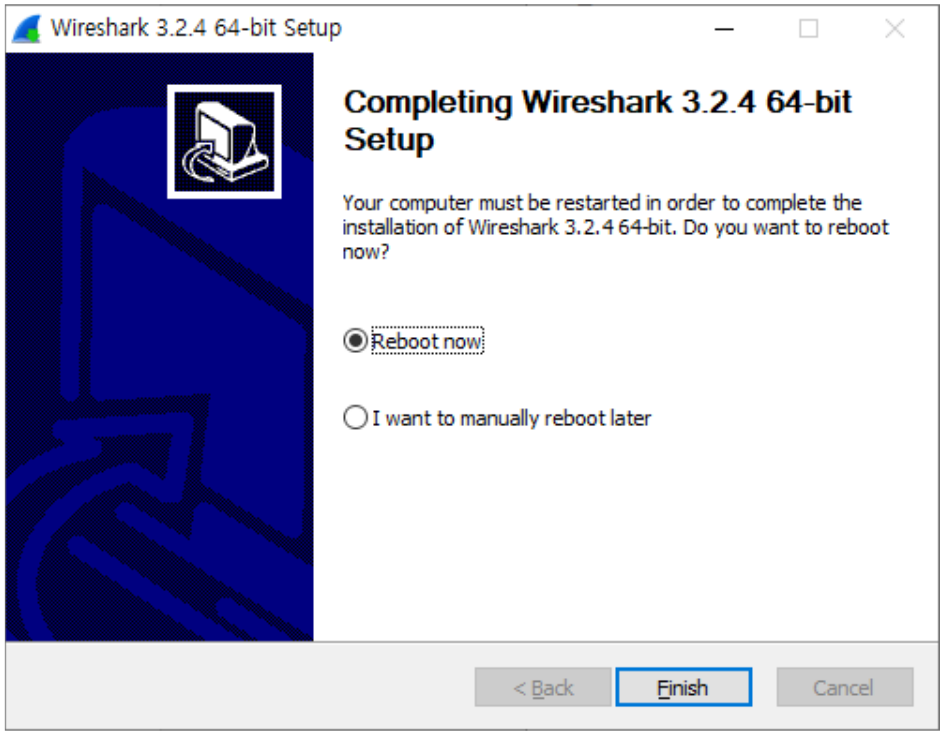
2 와이어샤크 (Wireshark) 설치 (Cont'd)





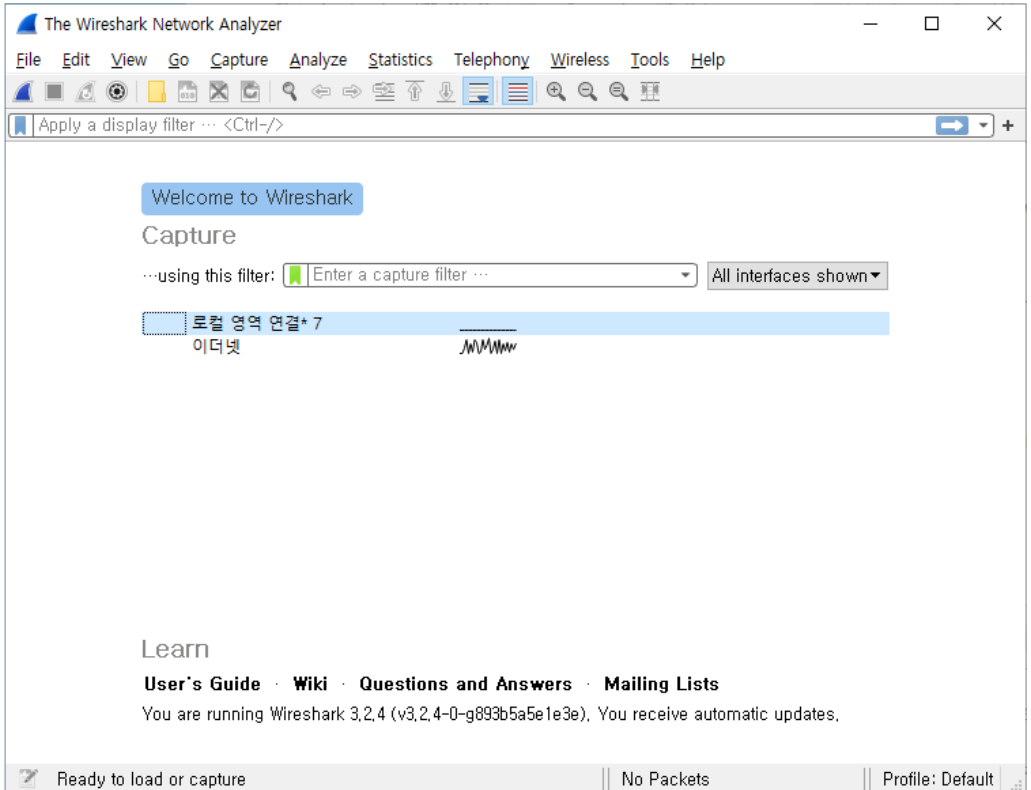
2 와이어샤크 (Wireshark) 설치 (Cont'd)

> 설치 완료

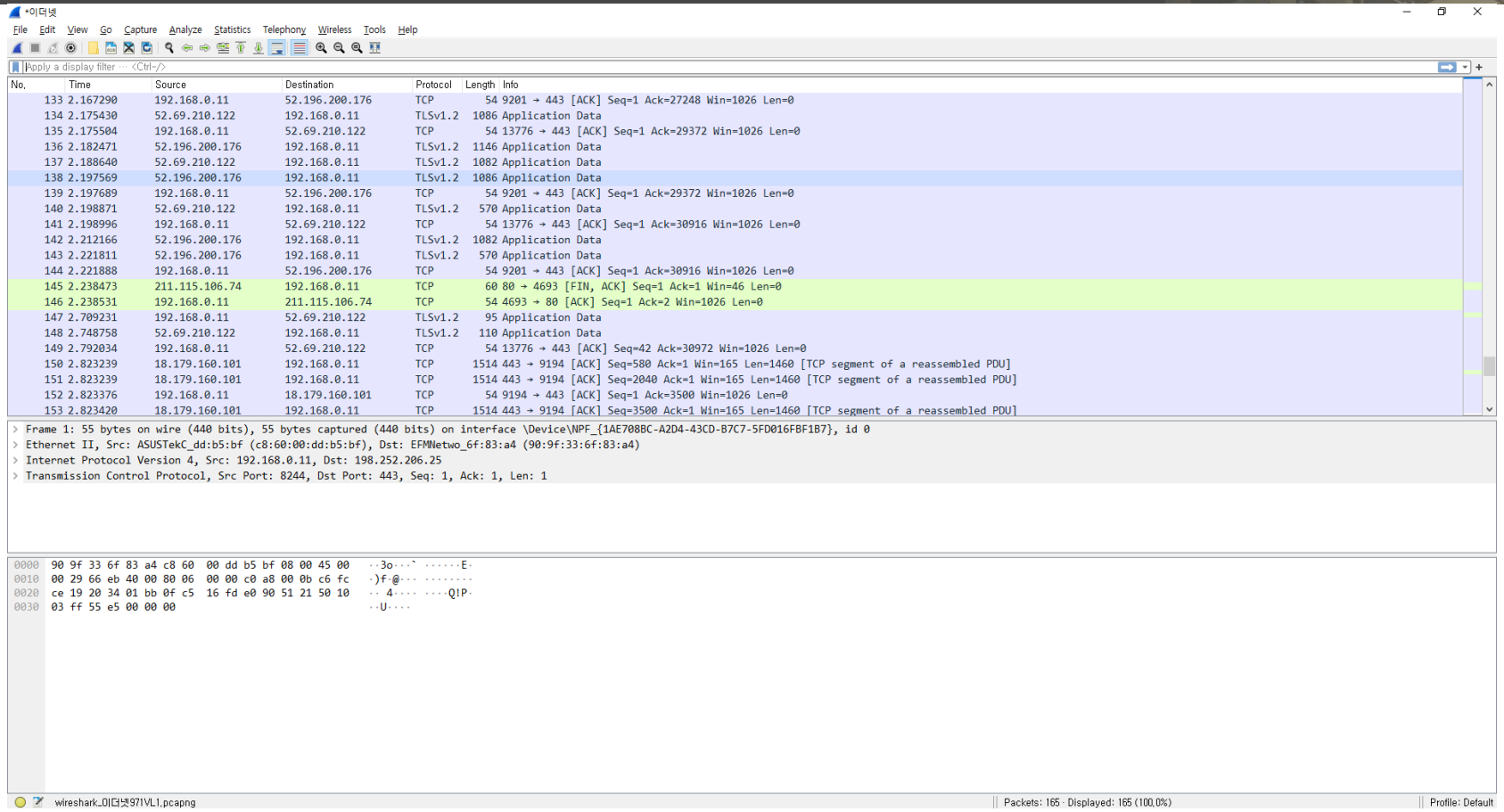


2 와이어샤크 (Wireshark) 실행

1) Wieshark 실행



2 Wireshark 패킷 캡처



2 Wireshark 필터링

1) IP 필터링

ip.addr == 192.168.0.11

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.11	198.252.206.25	TCP	55	8244 → 443 [ACK] Seq=1 Ack=1 Win=1023 Len=1 [TCP segment of a reassembled PDU]
2	0.022267	18.179.160.101	192.168.0.11	TLSv1.2	261	Application Data
3	0.026911	13.114.192.111	192.168.0.11	TLSv1.2	340	Application Data
4	0.066953	13.114.192.111	192.168.0.11	TLSv1.2	300	Application Data
5	0.067052	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=533 Win=1024 Len=0
6	0.076955	192.168.0.11	18.179.160.101	TCP	54	9194 → 443 [ACK] Seq=1 Ack=208 Win=1025 Len=0
7	0.098614	52.69.210.122	192.168.0.11	TLSv1.2	1060	Application Data
8	0.106478	52.196.200.176	192.168.0.11	TLSv1.2	1060	Application Data
9	0.109697	52.69.210.122	192.168.0.11	TLSv1.2	788	Application Data
10	0.109785	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
11	0.117377	52.196.200.176	192.168.0.11	TLSv1.2	788	Application Data
12	0.117475	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
13	0.122218	52.69.210.122	192.168.0.11	TLSv1.2	1145	Application Data
14	0.131393	52.196.200.176	192.168.0.11	TLSv1.2	1145	Application Data
15	0.136293	52.69.210.122	192.168.0.11	TLSv1.2	1157	Application Data
16	0.136386	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
17	0.146371	52.196.200.176	192.168.0.11	TLSv1.2	1157	Application Data
18	0.146471	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
19	0.149626	52.69.210.122	192.168.0.11	TLSv1.2	1080	Application Data
20	0.160147	52.196.200.176	192.168.0.11	TLSv1.2	1080	Application Data
21	0.163108	52.69.210.122	192.168.0.11	TLSv1.2	1087	Application Data

2 Wireshark 필터링 (Cont'd)

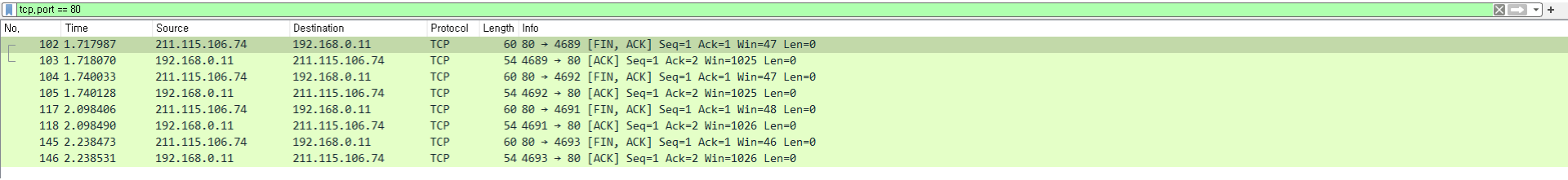
1) Source IP 필터링

ip.src == 192.168.0.11

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.11	198.252.206.25	TCP	55	8244 → 443 [ACK] Seq=1 Ack=1 Win=1023 Len=1 [TCP segment of a reassembled PDU]
5	0.067052	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=533 Win=1024 Len=0
6	0.076955	192.168.0.11	18.179.160.101	TCP	54	9194 → 443 [ACK] Seq=1 Ack=208 Win=1025 Len=0
10	0.109785	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
12	0.117475	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
16	0.136386	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
18	0.146471	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
22	0.163174	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
24	0.174039	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
28	0.190413	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
31	0.203589	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
35	0.219192	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
37	0.224800	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
39	0.281386	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=709 Win=1023 Len=0
40	0.706163	192.168.0.11	18.178.61.111	TLSv1.2	108	Application Data
41	0.708228	192.168.0.11	147.46.241.13	SSH	146	Client: Encrypted packet (len=92)
45	0.797356	192.168.0.11	18.178.61.111	TCP	54	14836 → 443 [ACK] Seq=55 Ack=57 Win=1024 Len=0
49	0.918604	192.168.0.11	13.114.192.111	TLSv1.2	95	Application Data
51	0.944471	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
53	0.946375	192.168.0.11	52.68.38.34	TCP	54	13762 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
59	1.023891	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=3384 Win=1026 Len=0

2 Wireshark 필터링 (Cont'd)

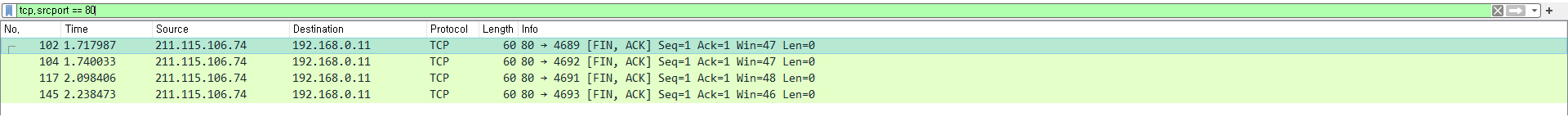
1) TCP port 필터링



No.	Time	Source	Destination	Protocol	Length	Info
102	1.717987	211.115.106.74	192.168.0.11	TCP	60	80 → 4689 [FIN, ACK] Seq=1 Ack=1 Win=47 Len=0
103	1.718070	192.168.0.11	211.115.106.74	TCP	54	4689 → 80 [ACK] Seq=1 Ack=2 Win=1025 Len=0
104	1.740033	211.115.106.74	192.168.0.11	TCP	60	80 → 4692 [FIN, ACK] Seq=1 Ack=1 Win=47 Len=0
105	1.740128	192.168.0.11	211.115.106.74	TCP	54	4692 → 80 [ACK] Seq=1 Ack=2 Win=1025 Len=0
117	2.098406	211.115.106.74	192.168.0.11	TCP	60	80 → 4691 [FIN, ACK] Seq=1 Ack=1 Win=48 Len=0
118	2.098490	192.168.0.11	211.115.106.74	TCP	54	4691 → 80 [ACK] Seq=1 Ack=2 Win=1026 Len=0
145	2.238473	211.115.106.74	192.168.0.11	TCP	60	80 → 4693 [FIN, ACK] Seq=1 Ack=1 Win=46 Len=0
146	2.238531	192.168.0.11	211.115.106.74	TCP	54	4693 → 80 [ACK] Seq=1 Ack=2 Win=1026 Len=0

2 Wireshark 필터링 (Cont'd)

1) TCP source port 필터링



2 Wireshark 필터링 (Cont'd)

1) 혼합 사용 (and, or)

tcp && ip.src == 192.168.0.11

tcp and ip.src == 192.168.0.11

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.11	198.252.206.25	TCP	55	8244 → 443 [ACK] Seq=1 Ack=1 Win=1023 Len=1 [TCP segment of a reassembled PDU]
5	0.067052	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=533 Win=1024 Len=0
6	0.076955	192.168.0.11	18.179.160.101	TCP	54	9194 → 443 [ACK] Seq=1 Ack=208 Win=1025 Len=0
10	0.109785	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
12	0.117475	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
16	0.136386	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
18	0.146471	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
22	0.163174	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
24	0.174039	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
28	0.190413	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
31	0.203589	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
35	0.219192	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
37	0.224800	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
39	0.281386	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=709 Win=1023 Len=0
40	0.706163	192.168.0.11	18.178.61.111	TLSv1.2	108	Application Data
41	0.708228	192.168.0.11	147.46.241.13	SSH	146	Client: Encrypted packet (len=92)
45	0.797356	192.168.0.11	18.178.61.111	TCP	54	14836 → 443 [ACK] Seq=55 Ack=57 Win=1024 Len=0
49	0.918604	192.168.0.11	13.114.192.111	TLSv1.2	95	Application Data
51	0.944471	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
53	0.946375	192.168.0.11	52.68.38.34	TCP	54	13762 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
59	1.023891	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=3384 Win=1026 Len=0

tcp || ip.src == 192.168.0.11

tcp or ip.src == 192.168.0.11

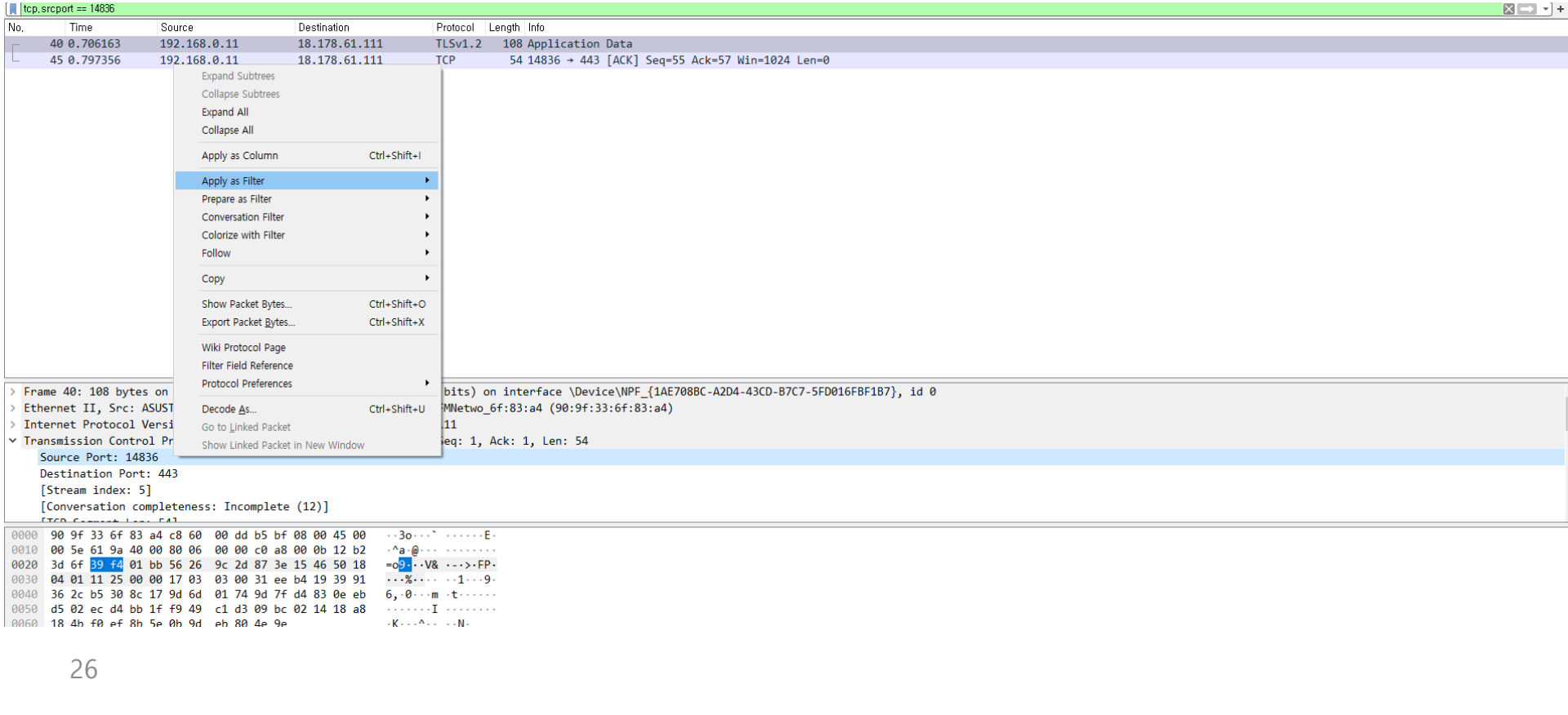
2 Wireshark 필터링 (Cont'd)

1) 혼합 사용 (not)

!ip.dst == 192.168.0.11

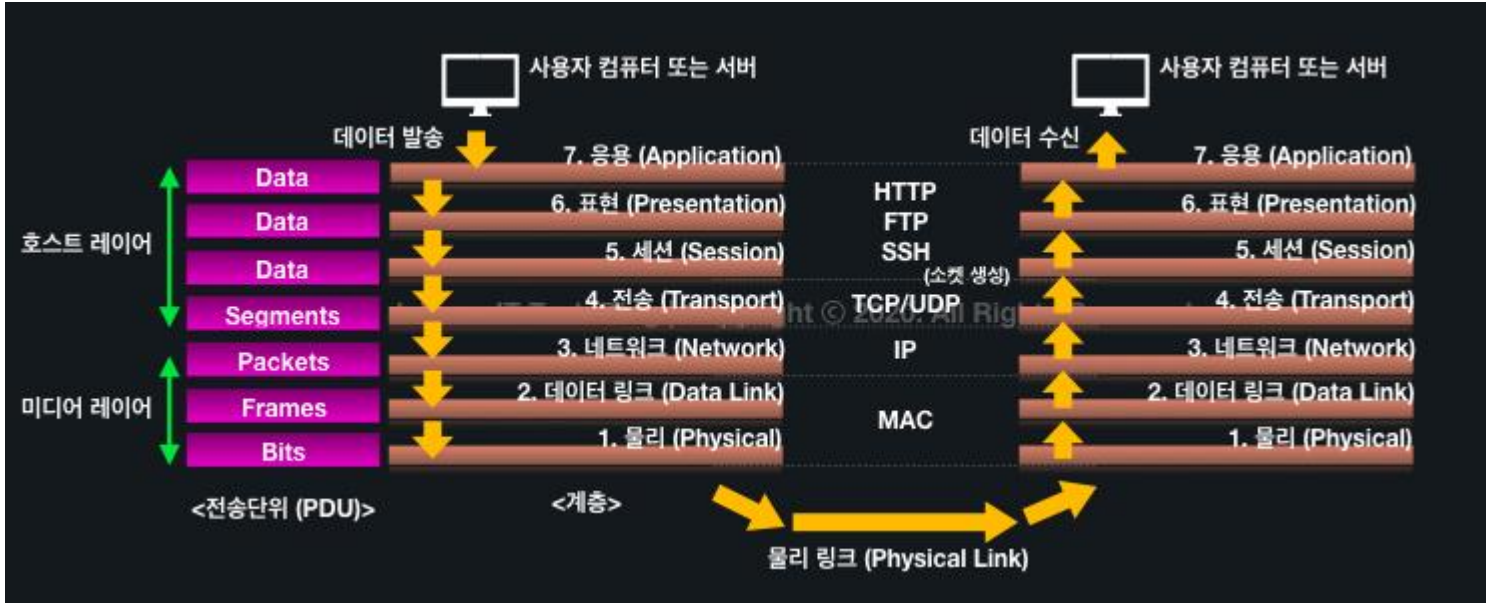
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.11	198.252.206.25	TCP	55	8244 → 443 [ACK] Seq=1 Ack=1 Win=1023 Len=1 [TCP segment of a reassembled PDU]
5	0.067052	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=533 Win=1024 Len=0
6	0.076955	192.168.0.11	18.179.160.101	TCP	54	9194 → 443 [ACK] Seq=1 Ack=208 Win=1025 Len=0
10	0.109785	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
12	0.117475	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=1741 Win=1026 Len=0
16	0.136386	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
18	0.146471	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=3935 Win=1026 Len=0
22	0.163174	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
24	0.174039	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=5994 Win=1026 Len=0
28	0.190413	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
31	0.203589	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=8192 Win=1026 Len=0
35	0.219192	192.168.0.11	52.69.210.122	TCP	54	13776 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
37	0.224800	192.168.0.11	52.196.200.176	TCP	54	9201 → 443 [ACK] Seq=1 Ack=9809 Win=1026 Len=0
39	0.281386	192.168.0.11	13.114.192.111	TCP	54	13760 → 443 [ACK] Seq=1 Ack=709 Win=1023 Len=0
40	0.706163	192.168.0.11	18.178.61.111	TLSv1.2	108	Application Data
41	0.708228	192.168.0.11	147.46.241.13	SSH	146	Client: Encrypted packet (len=92)
45	0.797356	192.168.0.11	18.178.61.111	TCP	54	14836 → 443 [ACK] Seq=55 Ack=57 Win=1024 Len=0
49	0.918604	192.168.0.11	13.114.192.111	TLSv1.2	95	Application Data
51	0.944471	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
53	0.946375	192.168.0.11	52.68.38.34	TCP	54	13762 → 443 [ACK] Seq=1 Ack=1692 Win=1026 Len=0
59	1.023891	192.168.0.11	52.192.84.97	TCP	54	9196 → 443 [ACK] Seq=1 Ack=3384 Win=1026 Len=0

2 Wireshark 필터링 (Cont'd)



(참고) OSI 7 Layer

1) 각 Layer별 전송 단위



source: https://www.stevenjlee.net/wp-content/uploads/2020/02/image.png

(참고) OSI 7 Layer (Cont'd)

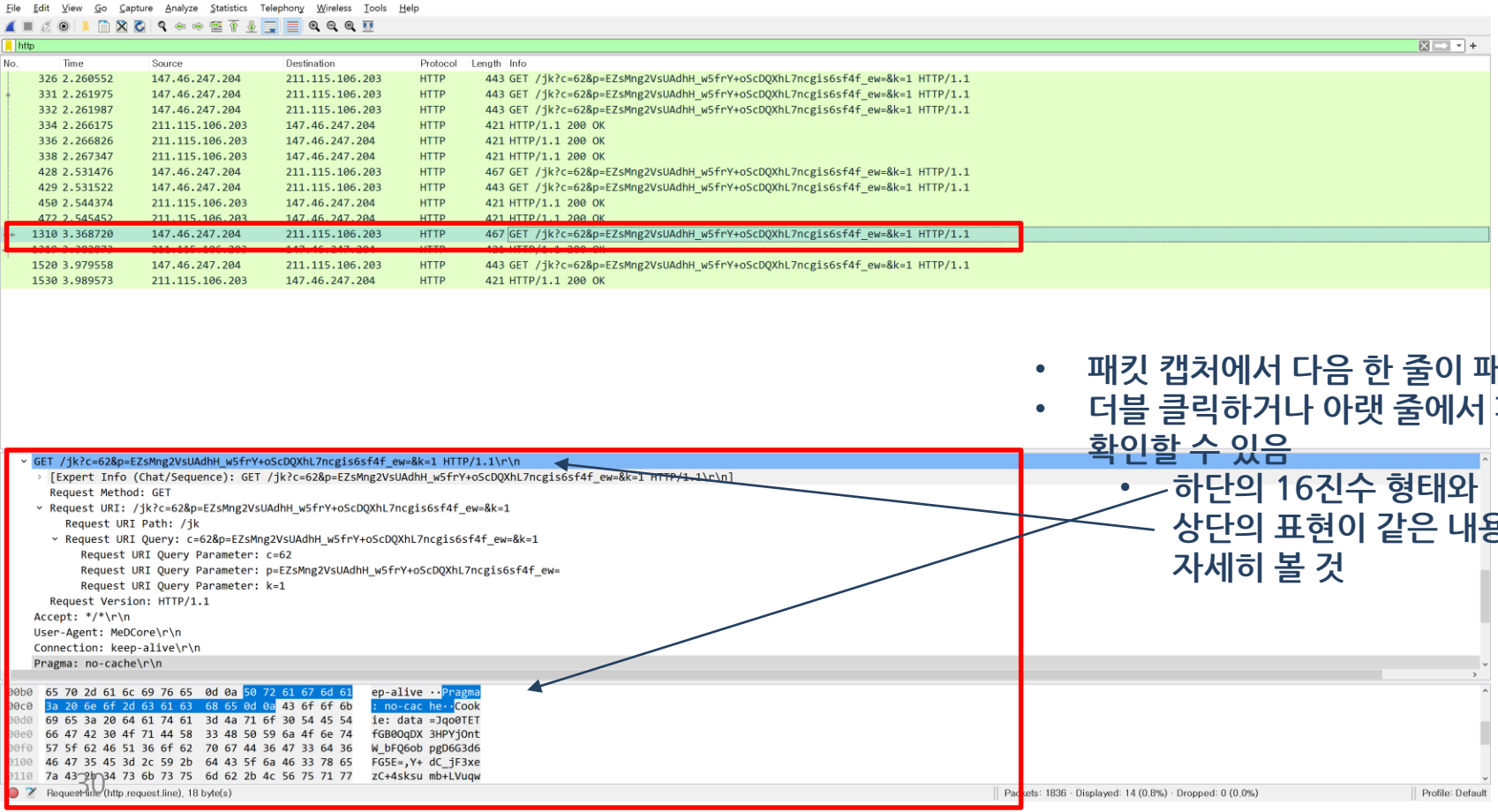
source: https://madplay.github.io/post/network-osi-7-layer

- 네트워크에서 통신이 일어나는 과정을 7단계로 표현
- 통신이 일어나는 과정을 단계별로 파악 가능
- 송신 시 윗 계층에서 헤더를 붙이면서 아랫 계층으로 이동하며, 수신 시 헤더를 분리하면서 윗 계층으로 이동
- 헤더는 각 계층에서 필요한 정보들이 들어있음

* Packet 내용 살펴보기

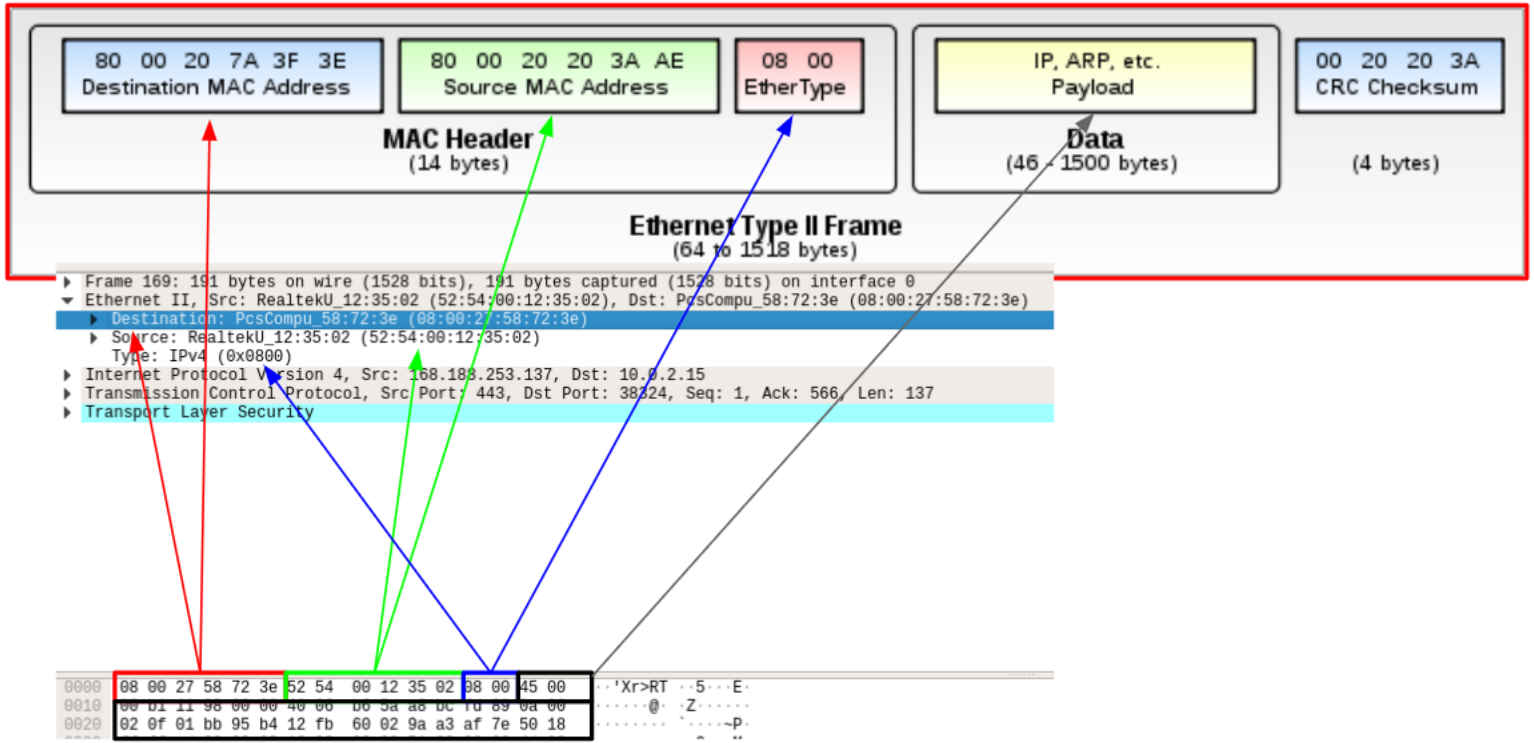
1) Wireshark를 통해 각 layer에서 packet header에 대한 구조 파악해보자

Packet 내용 살펴보기 (Cont'd)



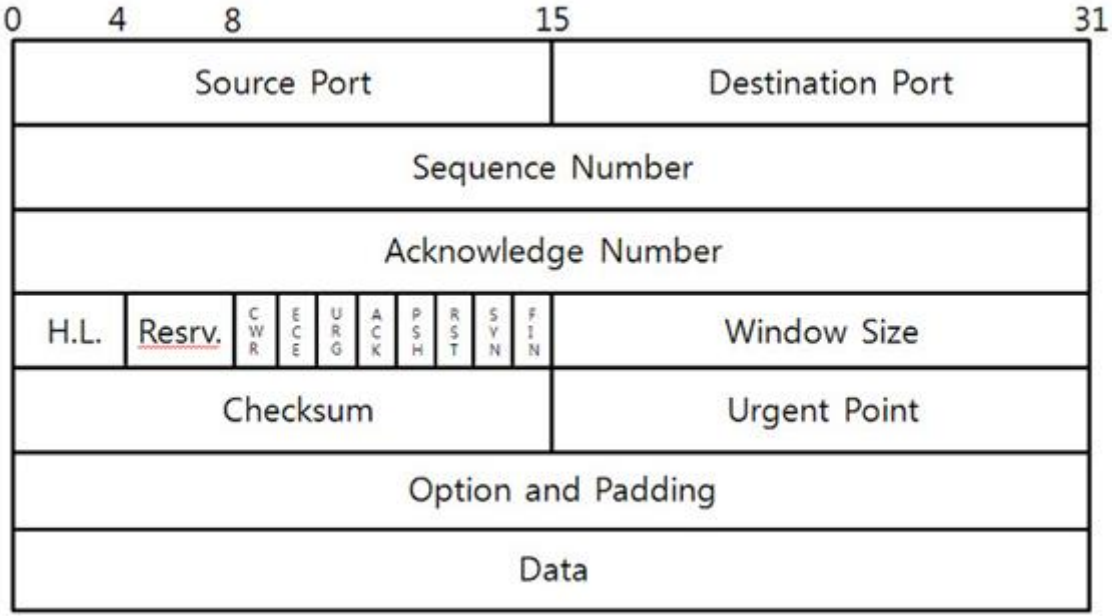
- 패킷 캡처에서 다음 한 줄이 패킷 한 개
- 더블 클릭하거나 아랫 줄에서 패킷 내용을 확인할 수 있음
  - 하단의 16진수 형태와 상단의 표현이 같은 내용이므로 자세히 볼 것

Ethernet header structure



TCP header structure

1) (실습) Wireshark로 분석하기



실습 - Questions

1) 패킷 분석하기

> 다양한 사이트를 선정 (택 1)

- www.naver.com
- www.daum.net
- www.youtube.com
- etc

> 패킷 캡처 후 하나의 패킷을 선정

Questions

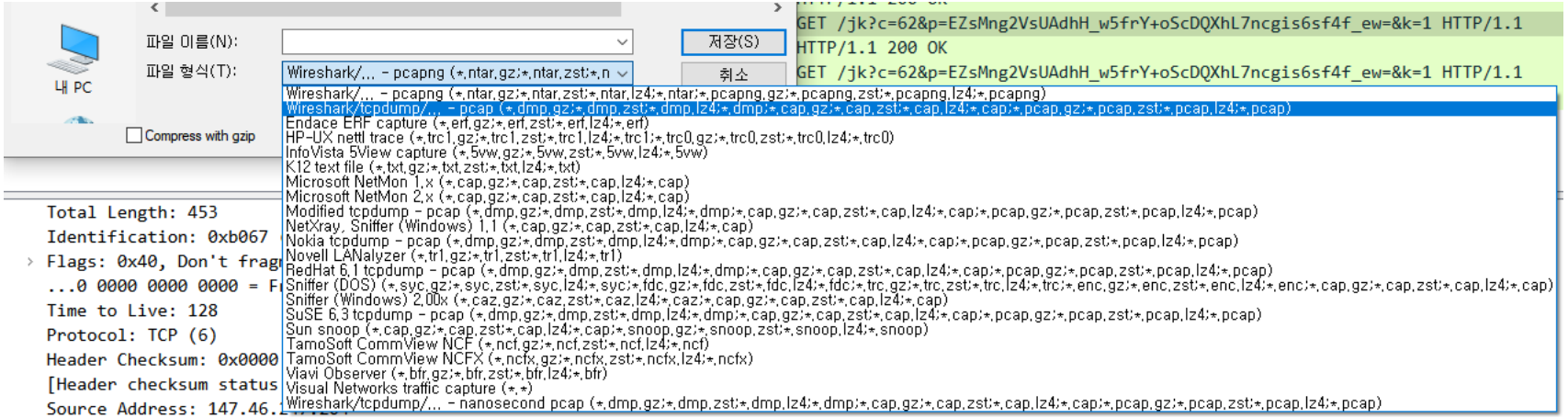
1) Frame이 내 컴퓨터 기준으로 source인가 destination인가?
2) IP header에서 Source IP address의 크기, 16진수로 값은?
3) IP header에서 Destination IP address의 크기, 16진수로 어떻게 표현되어 있는가?
4) 하나의 frame는 총 몇 바이트인가?
5) Ethernet header는 몇 바이트인가?
6) IP header는 몇 바이트인가?
7) TCP header는 몇 바이트인가?

* 제출 방법

1) 파일명: 우리은행_이름.zip
2) .pcap 파일과 보고서를 함께 압축하여 제출
3) 보고서는 pdf로 제출 (hwp, docs로 제출 X!!)
   - > 과제 목표
   - > 과제 해결 방법 (질문에 대한 답을 보고서에 작성, 답을 증명할 캡처 사진 필요)

* .pcap 파일 저장하는 방법

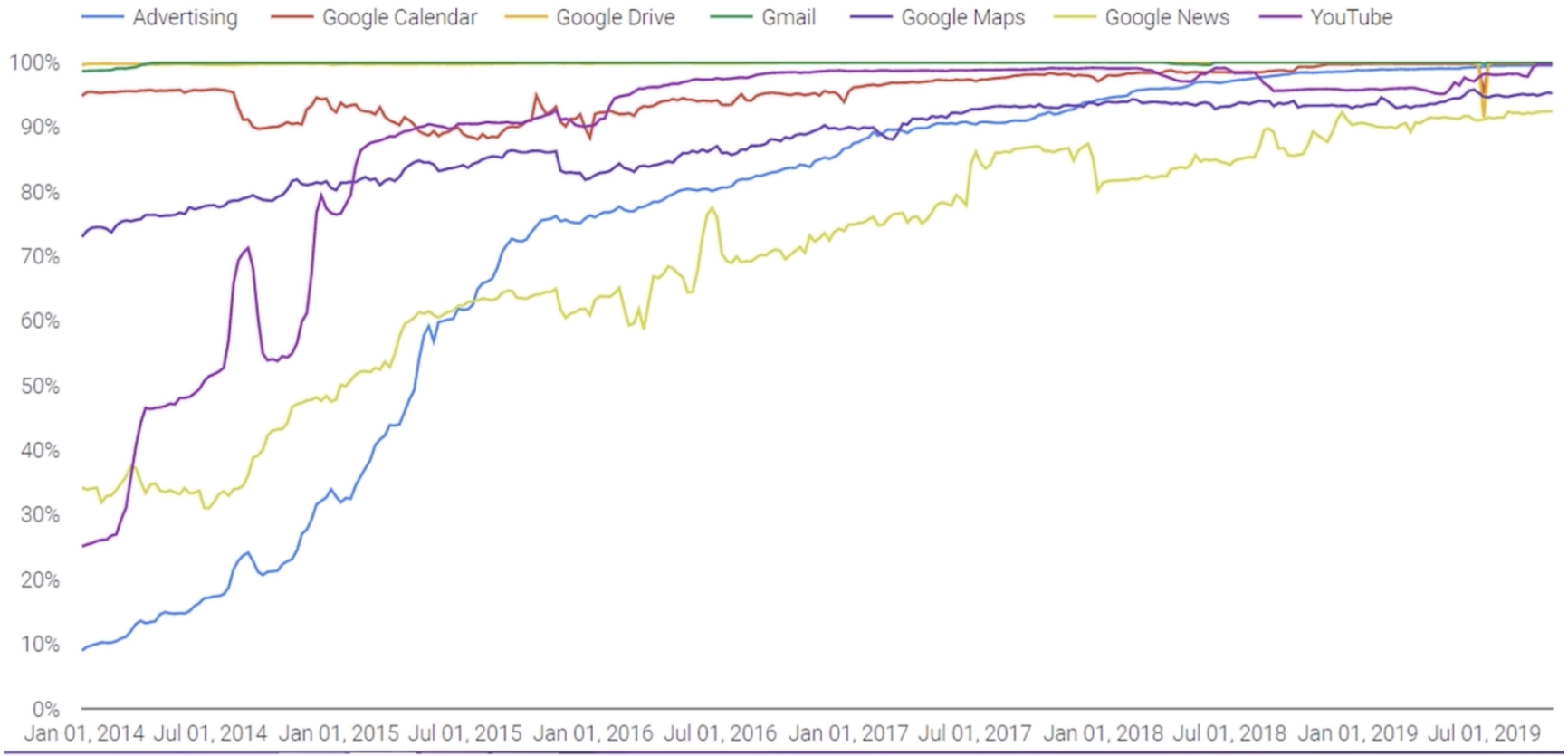
1) 메뉴에서 File -> Save as -> pcap 형식으로 변경 후 저장



1 http vs https

http	https
• Hypertext Transfer Protocol • No certificate • No encryption • TLS not used • No privacy	• Hypertext Transfer Protocol Secure • Certificate • Encryption • Use TLS • Privacy

1 https traffic is increasing



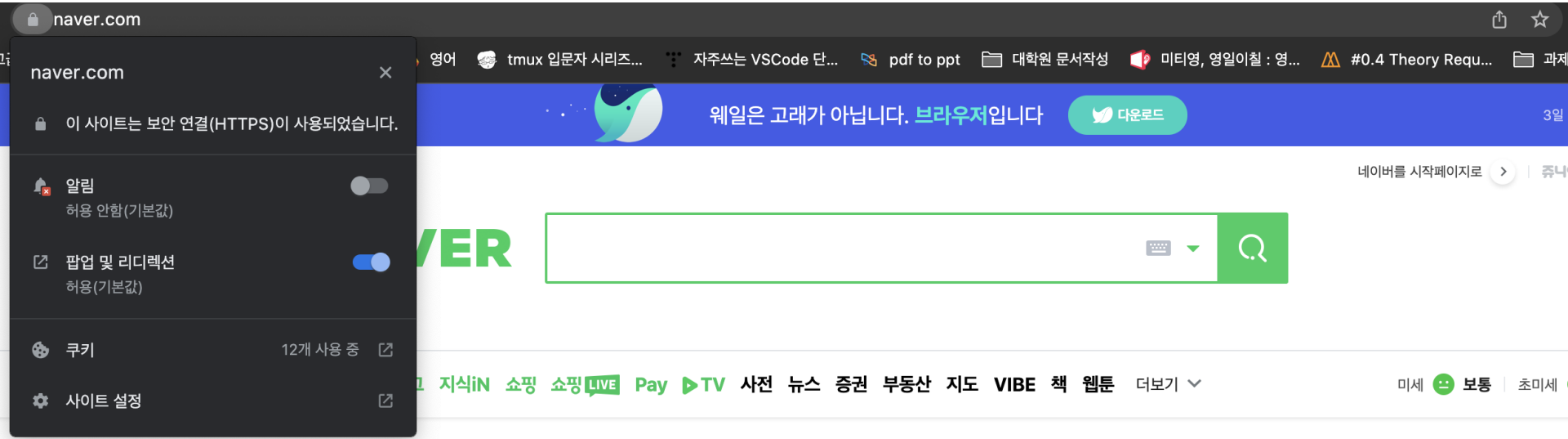
source: https://transparencyreport.google.com/https/overview?hl=en

1 What is SSL/TLS?

1) Transport Layer Security (TLS) protocol
   - > De facto standard for Internet security
   - > “The primary goal of the TLS protocol is to provide authentication, confidentiality and data integrity between two communicating applications”
   - > In practice, it is used to protect information transmitted between browsers and Web servers
2) Based on Secure Sockets Layer (SSL)
   - > SSL is the old version and deprecated
3) Deployed in every Web browser

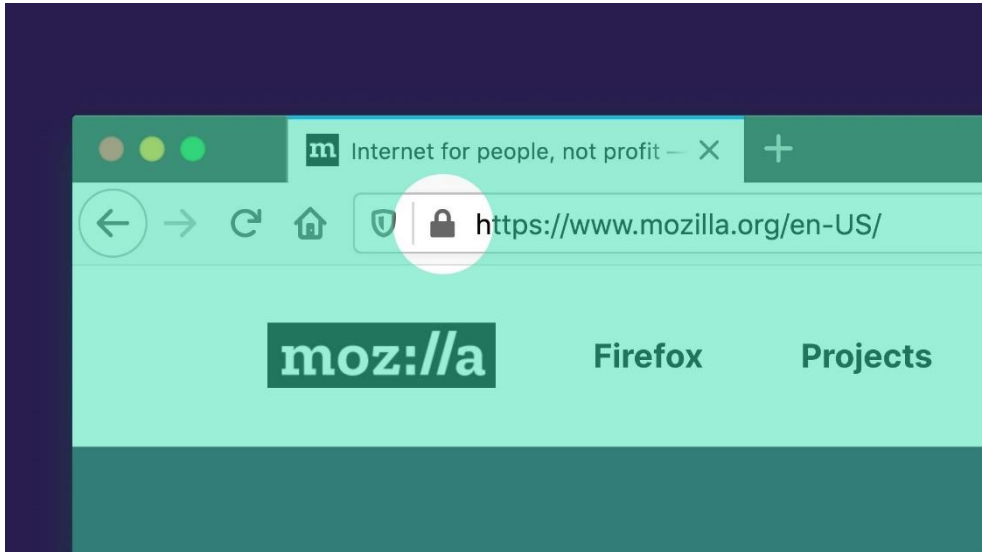
1 How can I know TLS is used?

1) Look at the address line in the browser



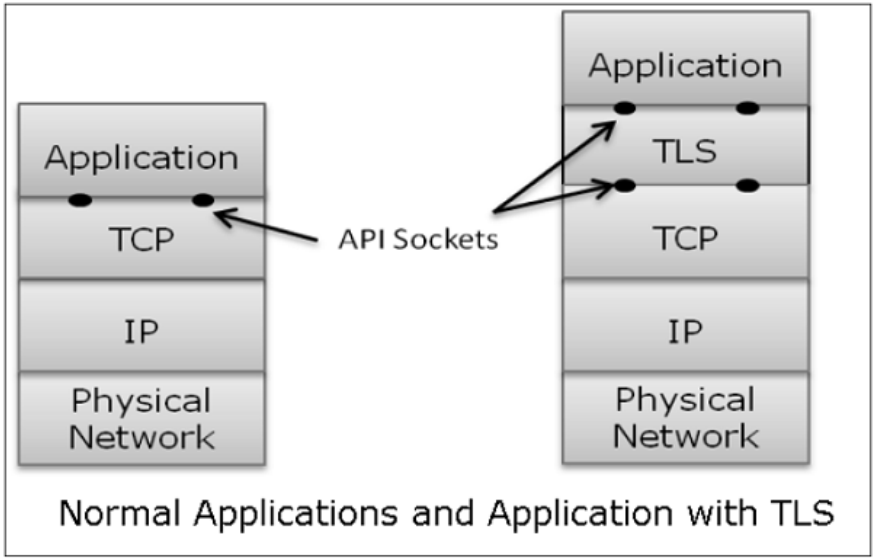
1 How can I know TLS is used? (Cont'd)

1) Look at the address line in the browser



1 TLS provides..

1) Application level protection
2) End-to-end security
3) Insecure Transport Layer VS Secure Transport Layer

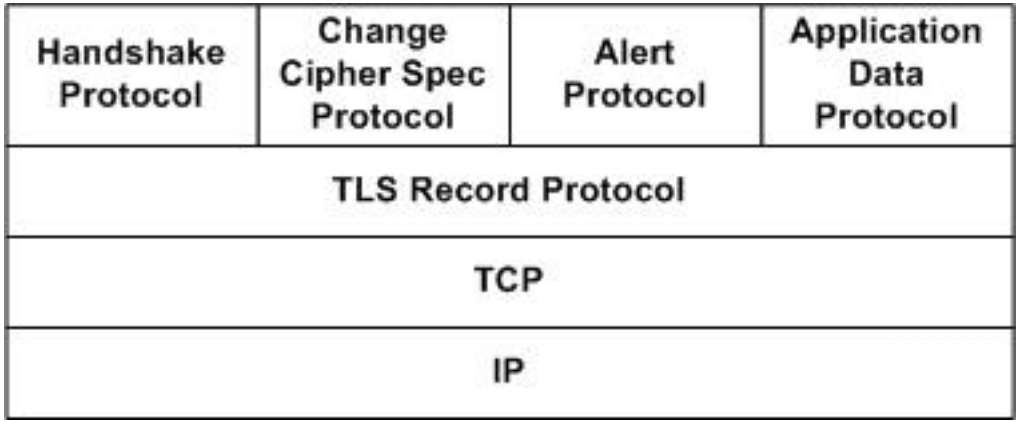


source: https://www.tutorialspoint.com/network_security/network_security_transport_layer.ht

1 TLS basics

1) TLS consists of two main protocols
   > total four protocols
2) Handshake protocol
   > Use public-key cryptography to establish a shared secret key between the client and the server
3) Record protocol
   > Use the secret key established in the handshake protocol to protect communication between the client and the server
4) We will focus on the handshake protocol

1 TLS protocol architecture



source: A Secured Service Level Negotiation In Ubiquitous Environments

2 TLS handshake: overview

1) Two parties: client and server
2) Negotiating the version of the protocol and the set of cryptographic algorithms to be used
   - > Interoperability between different versions
3) Authenticating the server
   - > Using digital certificates to learn the server's public key and verify the server's identity
4) Authenticating the client optionally
5) Using the server's public key to establish a shared secret

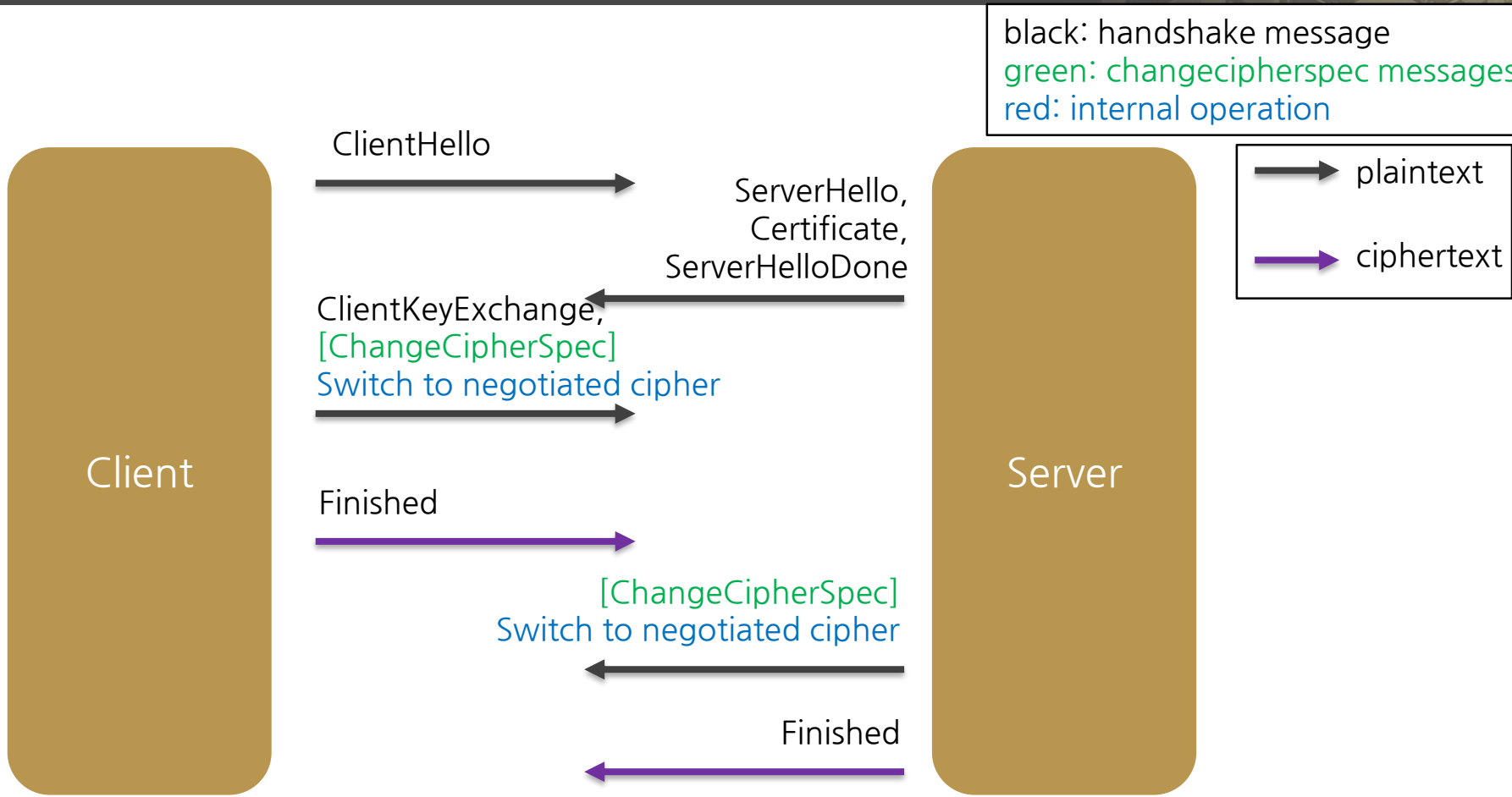
2 TLS handshake: overview (Cont'd)

6) Symmetric key is generated from the secret

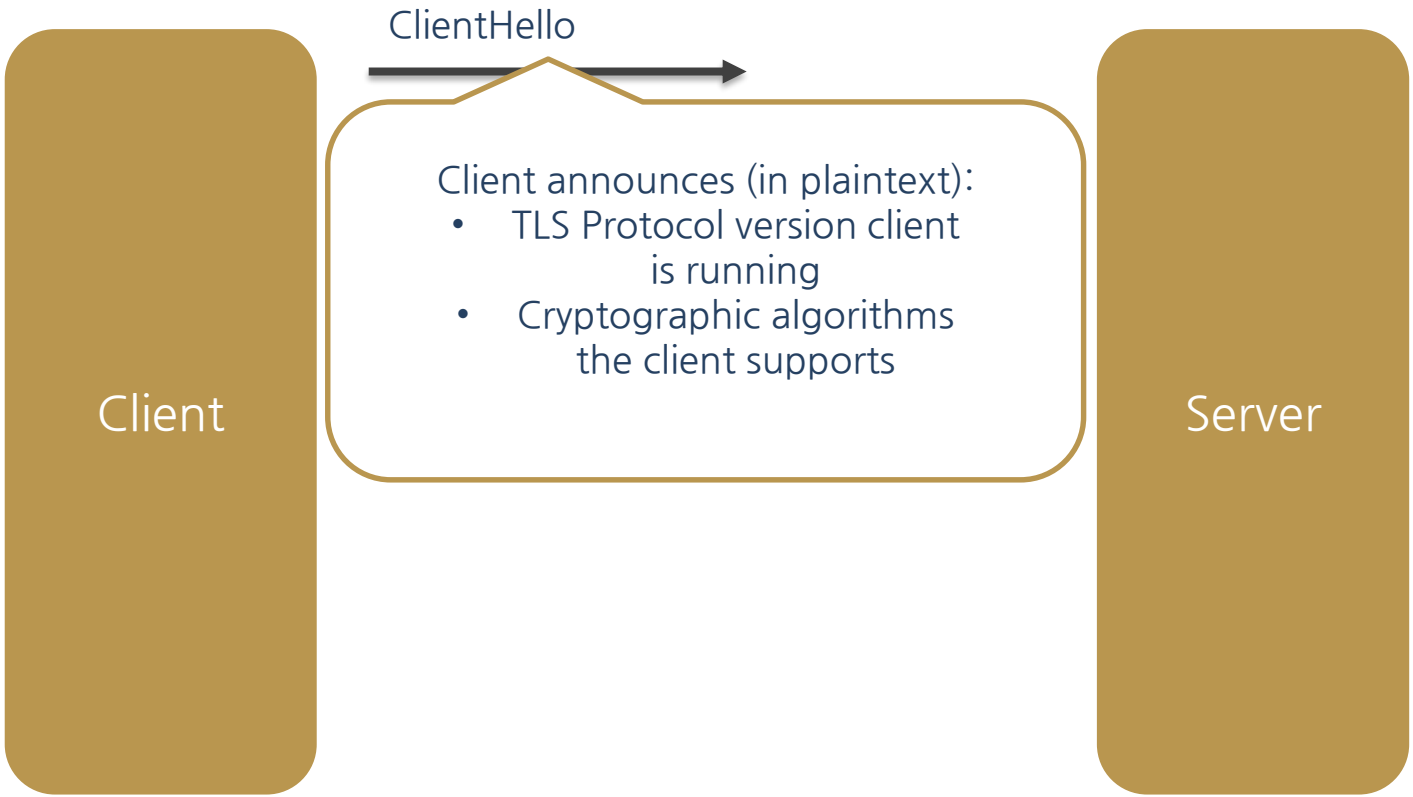
7) The following is based on TLS 1.1 & 1.2

We simplify the TLS message flow in the following slides. There are many variations in the message flow depending on PKC ciphers and certificates.

2 TLS handshake message flow



2 ClientHello



2 ClientHello fields

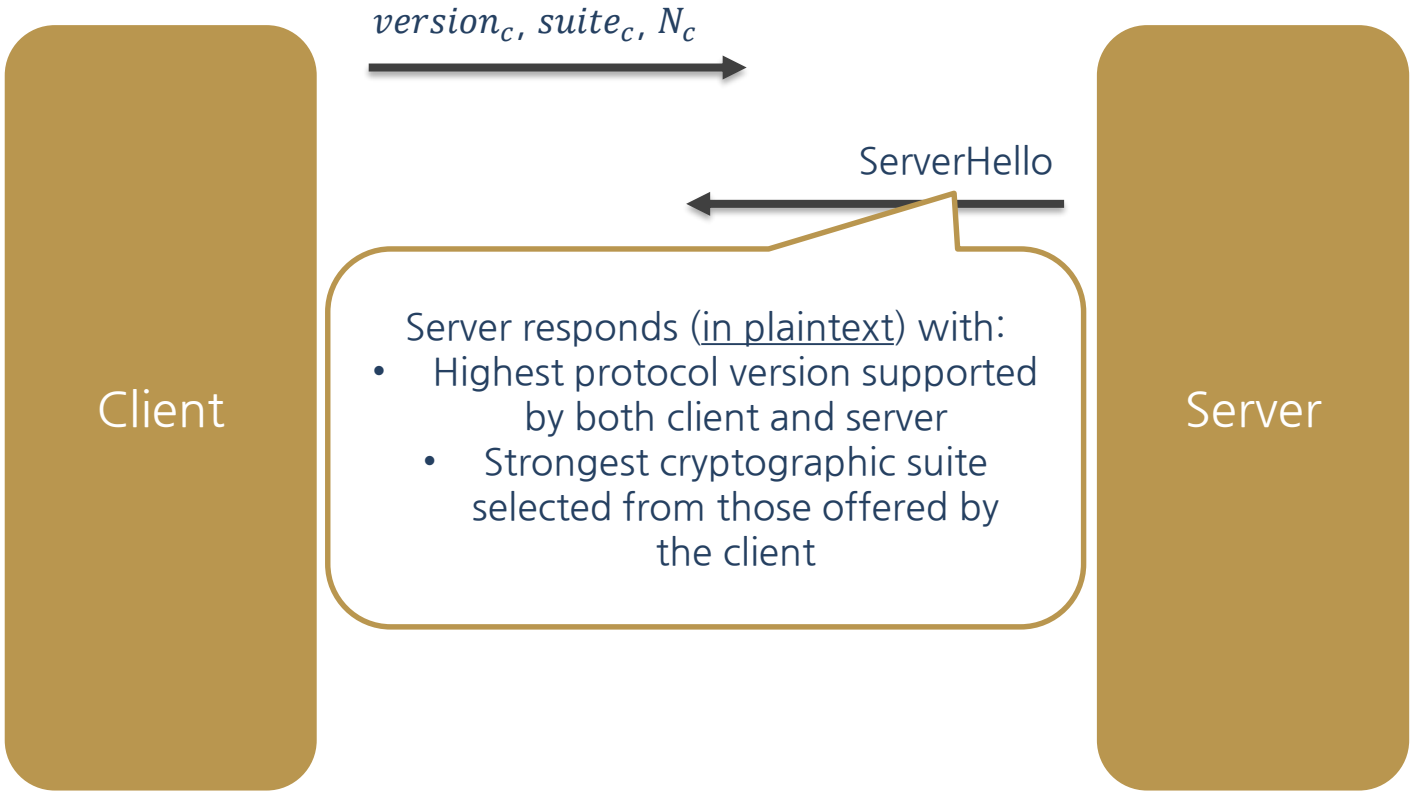
```
struct {
        PortocolVersion cleint_version;
        Random random;
        SessionID session_id;
        CipherSuite cipher_suites;
        CompressionMethod compression_methods;
} ClientHello
```

Highest version of the protocol supported by the client

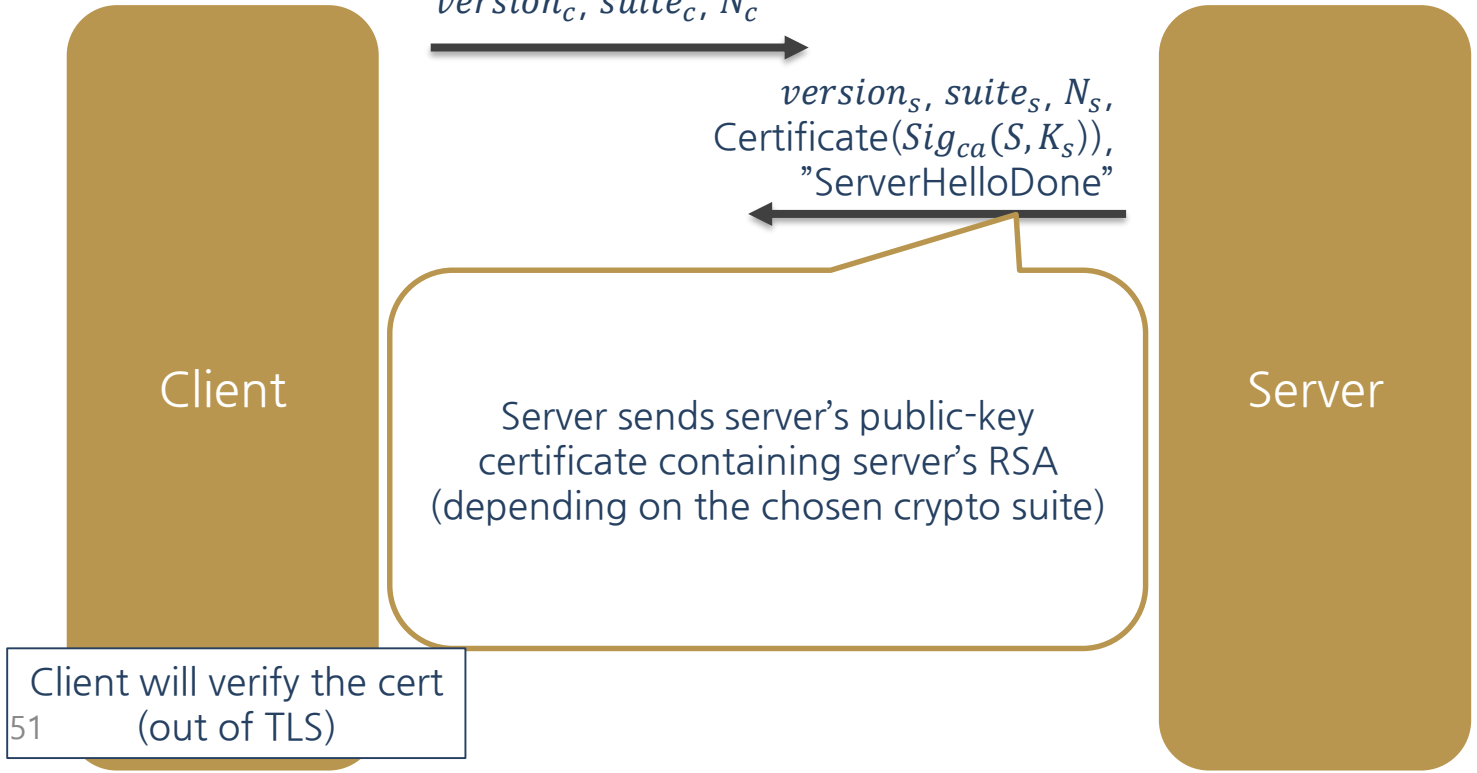
Session id (if the client wants to resume an old session)

Set of cryptographic algorithms supported by the client (e.g., RSA)

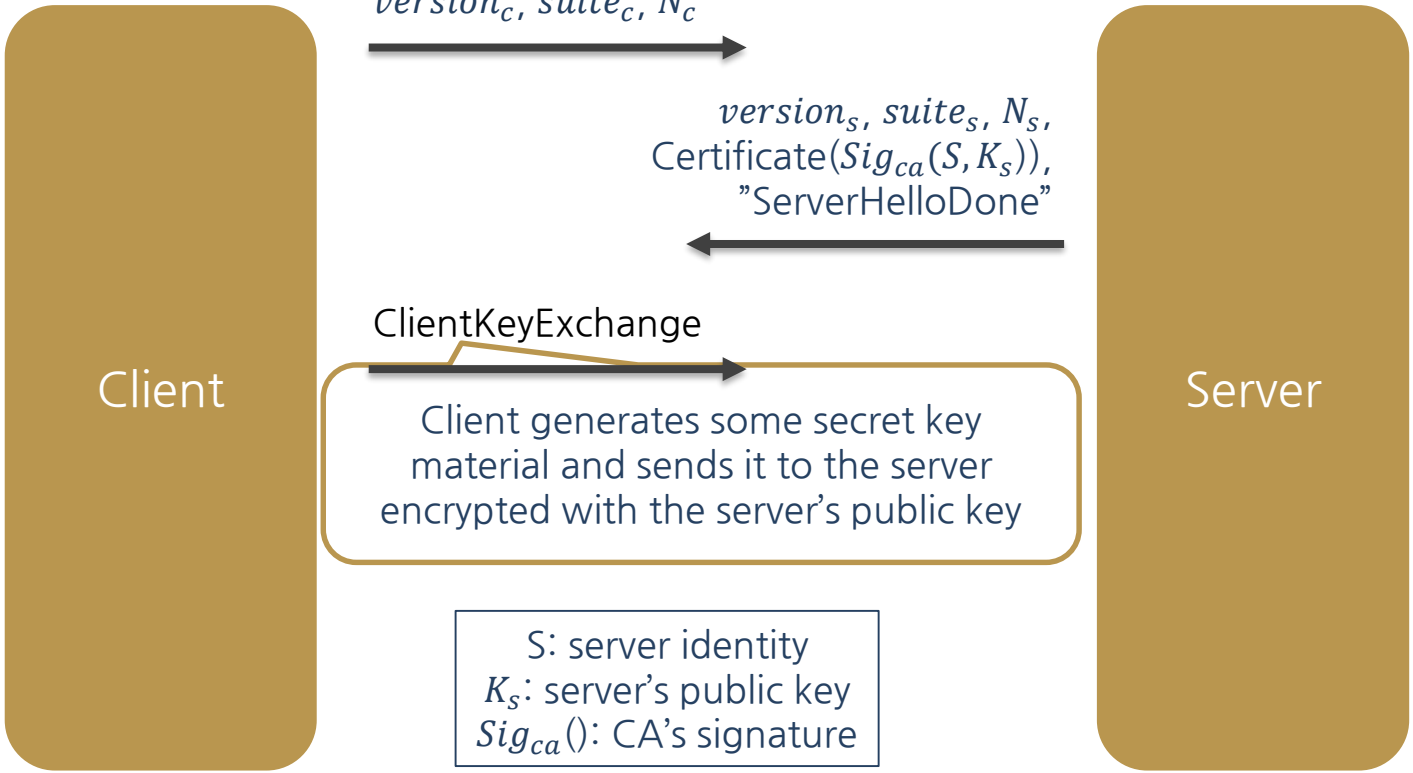
2 ServerHello



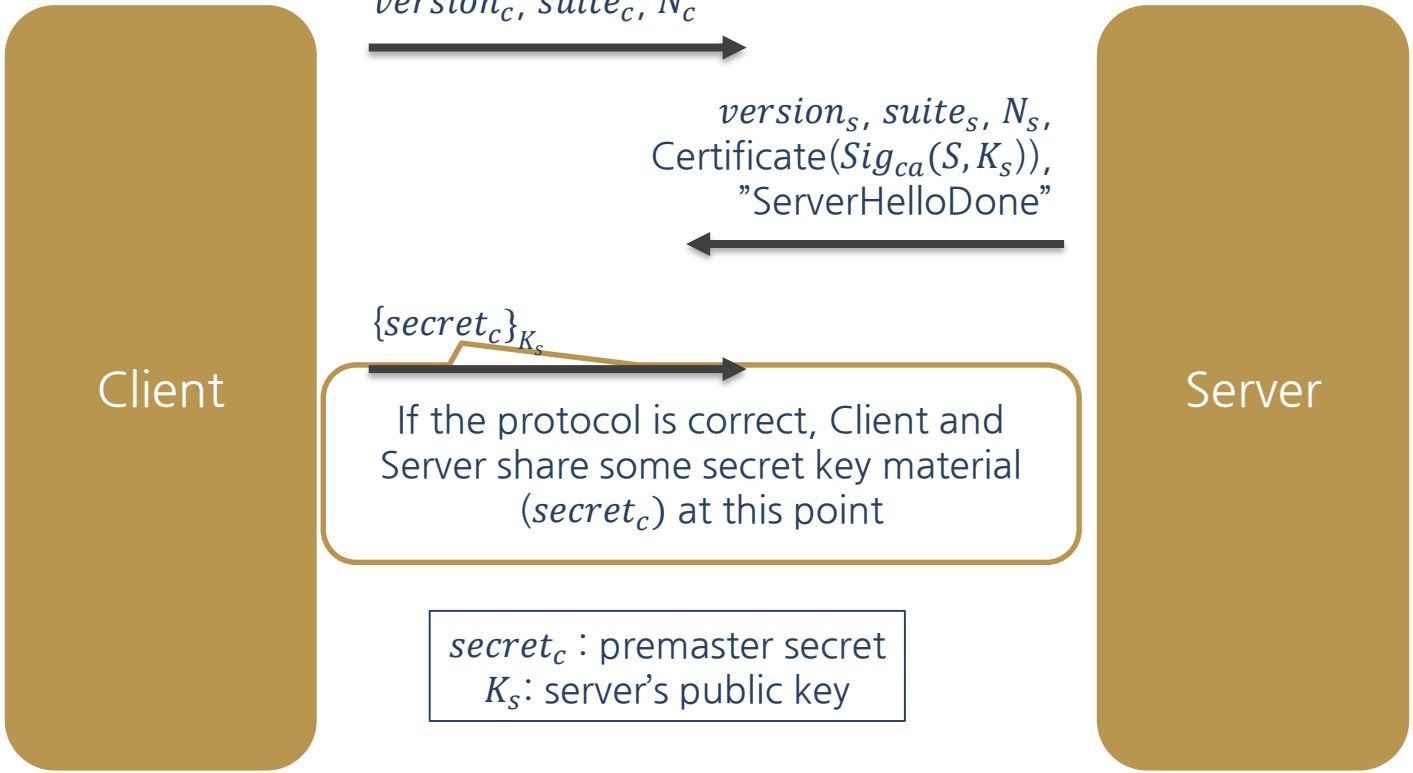
2 Certificate (of server)



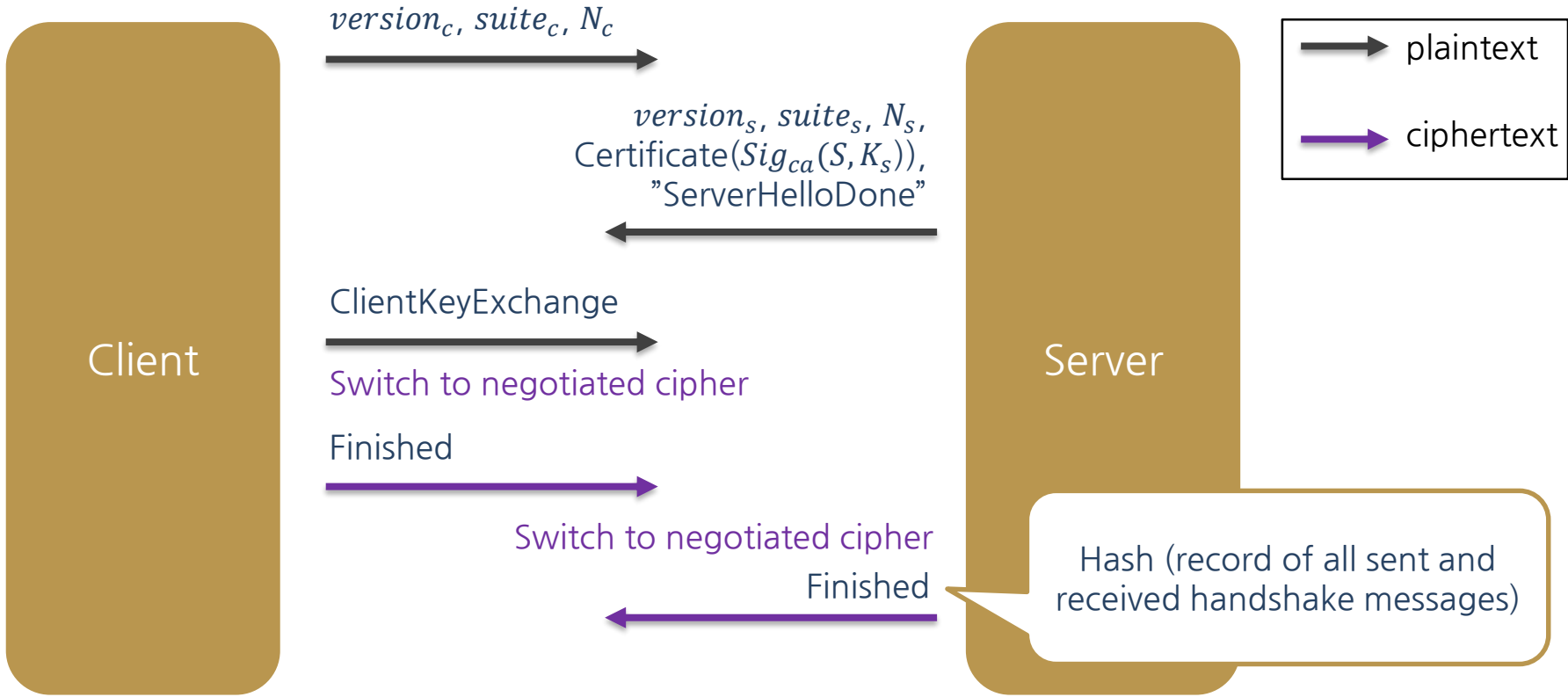
2 ClientKeyExchange



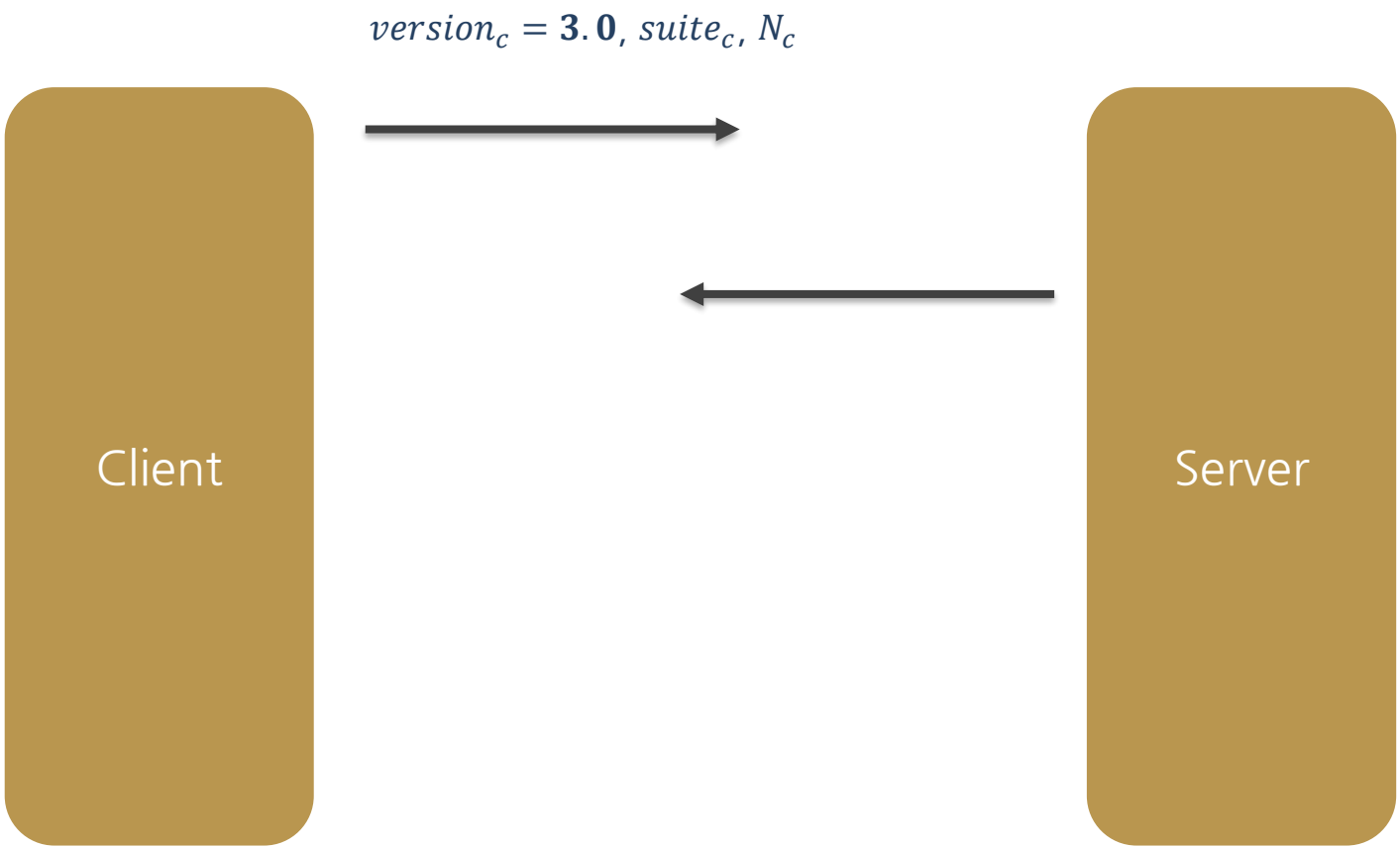
2 TLS message flow: RSA case



2 Finished

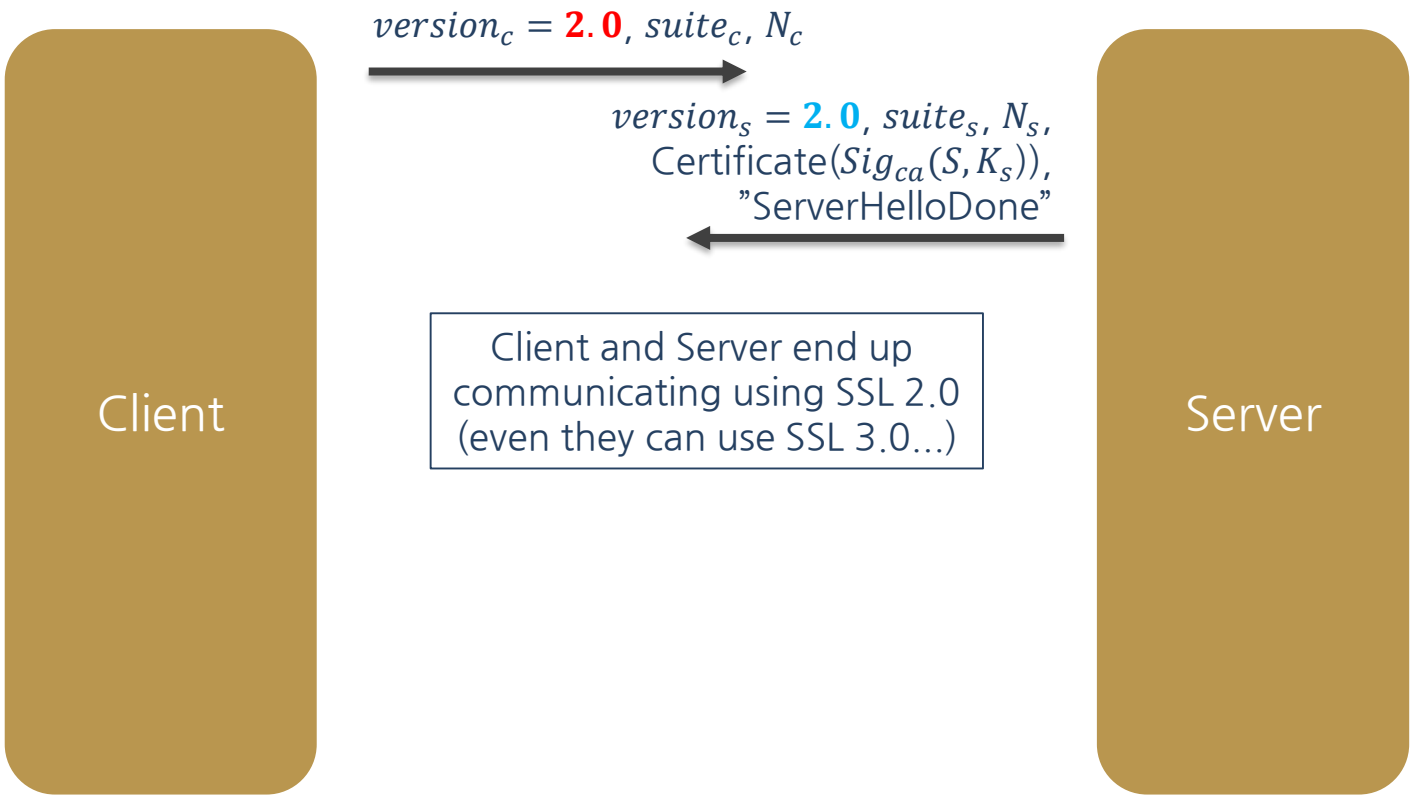

Client
Server
$version_c$, $suite_c$, N_c
$version_s$, $suite_s$, N_s,
Certificate($Sig_{ca}(S, K_s)$),
"ServerHelloDone"
ClientKeyExchange
Switch to negotiated cipher
Finished
Switch to negotiated cipher
Finished
Hash (record of all sent and received handshake messages)
plaintext
ciphertext

3 Version rollback attack

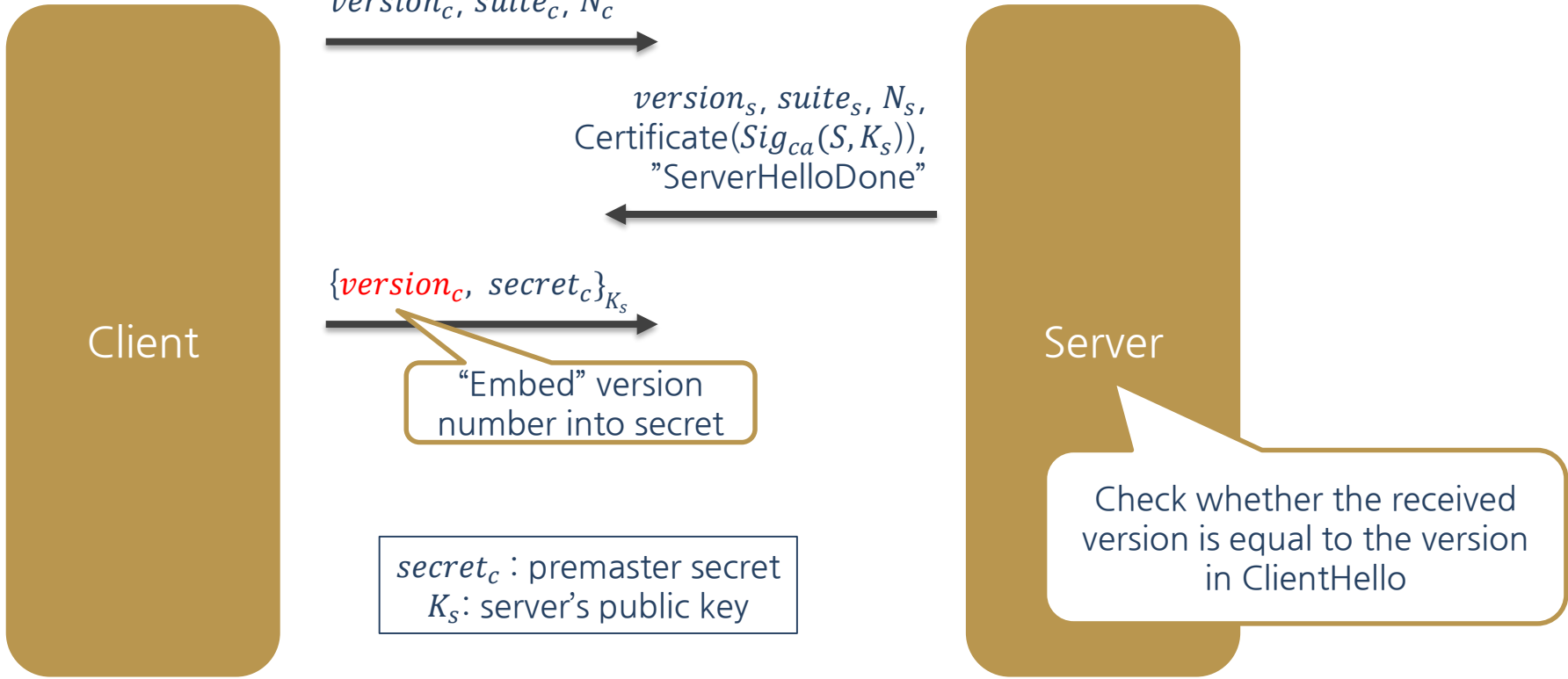


3 Version rollback attack (Cont'd)

$secret_c$: premaster secret
K_s: server's public key



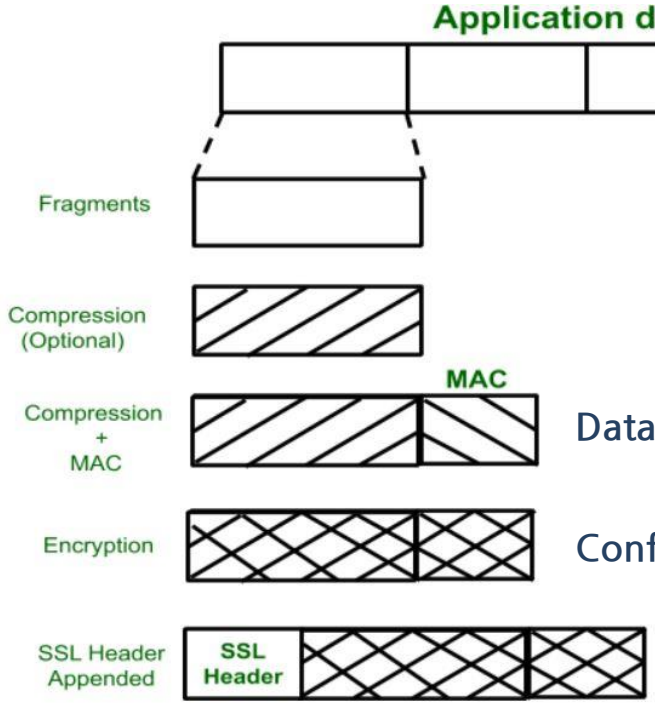
3 Version check in SSL 3.0



3 Finished message

1) It contains the hash of all the exchanged messages
2) To thwart any attempt of tampering messages in the middle
3) If the two hashes do not match, there has been message modification during the TLS handshake

1) keys for MAC and for encryption are different



Data Integrity ensures that tampering will be detected

Confidentiality ensures that the data will be private

source: https://www.geeksforgeeks.org/secure-socket-layer-ssl/

3 Other TLS protocols

1) Alert protocol
   - > Management of SSL/TLS session with error messages
   - > Fatal errors and warning
2) ChangeCipherSpec protocol
   - > Not part of Handshake protocol
   - > Used to indicate that the entity is changing to the recently agreed cipher suite
3) Both protocols running over the Record protocol

TLS를 활용한 https 로 “Hello WooriBank~”화면에 출력하기

crypto/x509 package

x509 ?

: ITU-T가 만든 PKI(Public Key Infrastructure, 공개키 기반 구조)의 표준

ITU-T(국제전기통신연합 전기통신표준화부문, International Telecommunication Union Telecommunication Standardization Sector): 모든 전기통신 분야에 적용하는 표준을 만들어내는 단체

Key Generation Code

```
func main() {
    max := new(big.Int).Lsh(big.NewInt(1), 128)
    serialNumber, _ := rand.Int(rand.Reader, max)

    subject := pkix.Name{
        Organization:       []string{"test Organization"},
        OrganizationalUnit: []string{"test"},
        CommonName:         "Go Web Programming",
    }

    template := x509.Certificate{
        SerialNumber: serialNumber,
        Subject:      subject,
        NotBefore:    time.Now(),
        NotAfter:     time.Now().Add(365 * 24 * time.Hour),
        KeyUsage:     x509.KeyUsageKeyEncipherment | x509.KeyUsageDigitalSignature,
        ExtKeyUsage:  []x509.ExtKeyUsage{x509.ExtKeyUsageServerAuth},
        IPAddresses:  []net.IP{net.ParseIP("127.0.0.1")},
    }
    [redacted]
}
```

HTTPS Server Code

```go
package main

import (
    "fmt"
    "net/http"
)

type handler struct{}

func (h *handler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello WooriBank~")
}

func main() {
    handler := handler{}
    server := http.Server{
        Addr:    "127.0.0.1:8080",
        Handler: &handler,
    }
    server.ListenAndServeTLS("cert.pem", "key.pem")
}
```

cert.pem
key.pem

Thank you!
수고하셨습니다!