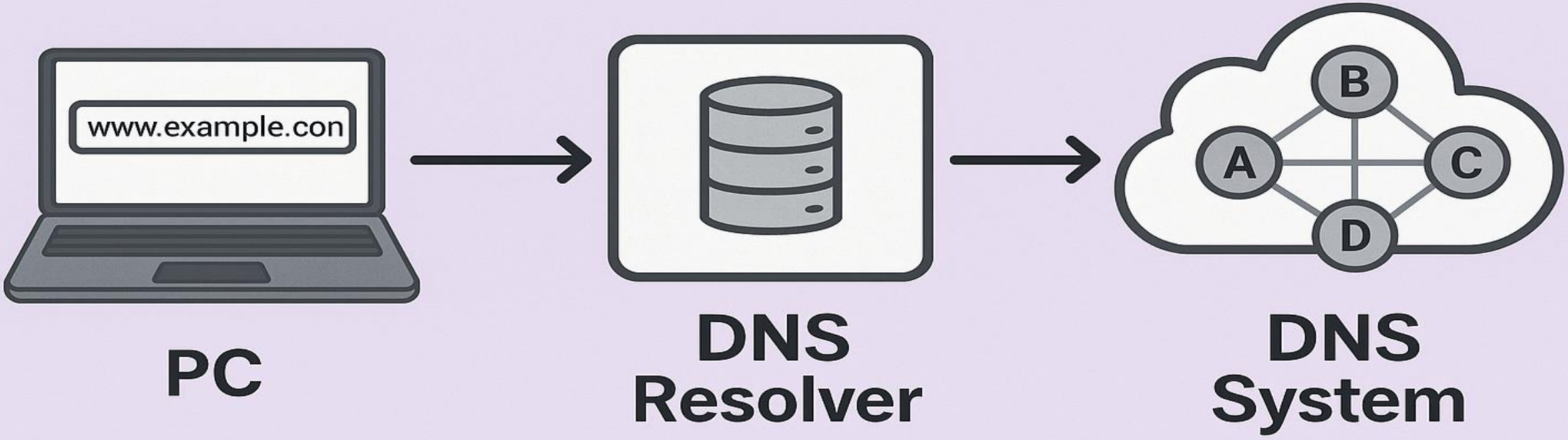
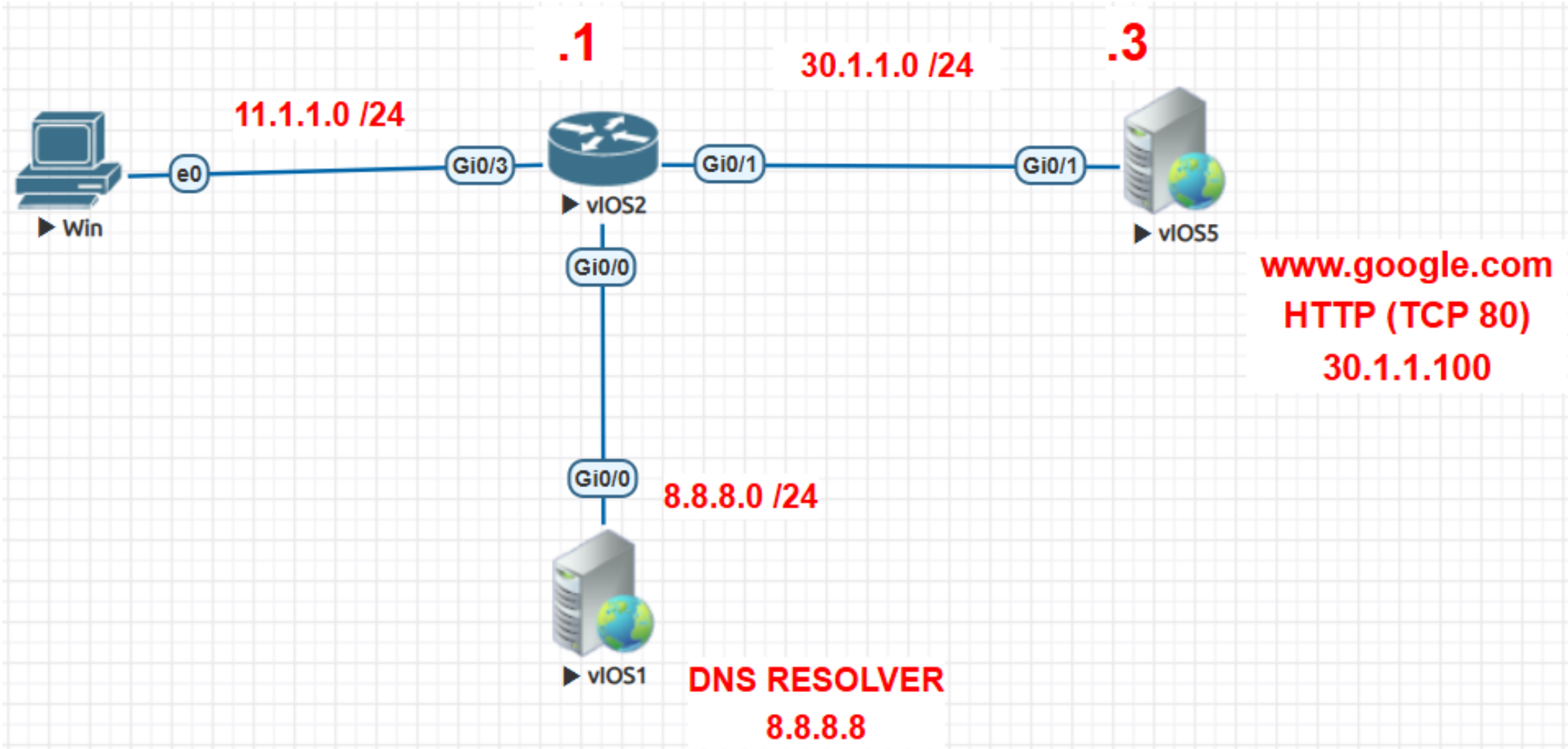

www.example.con
B
A
C
D
PC
DNS
Resolver
DNS
System

TOPOLOGY


.1
30.1.1.0 /24
.3
11.1.1.0 /24
e0
Gi0/3
Gi0/1
Gi0/1
Win
vIOS2
vIOS5
Gi0/0
www.google.com
HTTP (TCP 80)
30.1.1.100
Gi0/0
8.8.8.0 /24
vIOS1
DNS RESOLVER
8.8.8.8

When we enters the domain name in the browser and domain is not found in browser cache, host file, and os cache. Then it sends a request to the dns resolver. It uses:

source ip (win ip)
destination ip (dns server ip)
src mac (win mac)
dst mac (next hop/ gateway)
src port (random)
dst port (53)

Windows Pc -> DNS

```
> Frame 512: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface -, id 0
> Ethernet II, Src: 50:00:00:06:00:00 (50:00:00:06:00:00), Dst: 50:00:00:02:00:03 (50:00:00:02:00:03)
> Internet Protocol Version 4, Src: 11.1.1.11, Dst: 8.8.8.8
> User Datagram Protocol, Src Port: 62472, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0xd01d
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.google.com: type A, class IN
        Name: www.google.com
        [Name Length: 14]
        [Label Count: 3]
        Type: A (1) (Host Address)
        Class: IN (0x0001)
    [Response In: 513]
```

WinPC:
Ip: 10.1.1.11
Mac: 5000:0060:0000
Router:
Mac - G0/2 - 5000.0002.0003
- G0/0 - 5000.0002.0000
- G0/1 - 5000.0002.0001
Web-Web Server:
G0/0 -5000.0005.0000

When the DNS Resolver sends a reply in response to the DNS request of the client, it provides the IP address of the requested domain. If ip address found in the cache.
It uses

source ip (server ip)
destination ip (wind ip)
src mac (server mac)
dest mac (next hop)
src port (53)
dest port (random)

WinPC:
IP: 11.1.1.11
Mac: 5000:0060:0000
Router:
Mac - G0/2 - 5000.0002.0003
- G0/0 - 5000.0002.0000
- G0/1 - 5000.0002.0001
Web-Web Server:
G0/0 -5000.0005.0000

DNS -> Windows Pc

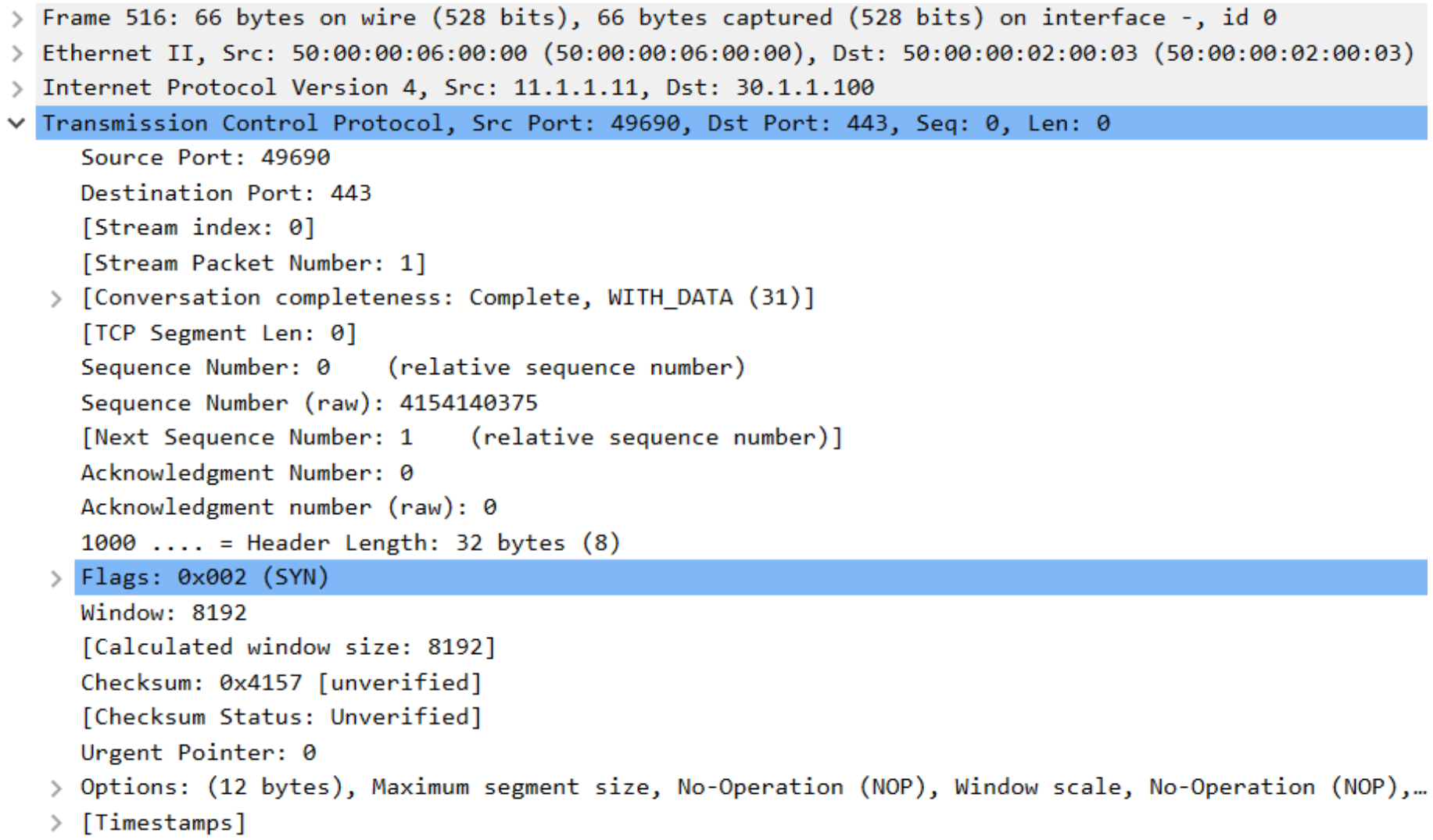
```
> Frame 497: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface -, id 0
> Ethernet II, Src: 50:00:00:01:00:00 (50:00:00:01:00:00), Dst: 50:00:00:02:00:00 (50:00:00:02:00:00)
> Internet Protocol Version 4, Src: 8.8.8.8, Dst: 11.1.1.11
> User Datagram Protocol, Src Port: 53, Dst Port: 62472
v Domain Name System (response)
    Transaction ID: 0xd01d
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.google.com: type A, class IN
  v Answers
    > www.google.com: type A, class IN, addr 30.1.1.100
    [Request In: 496]
    [Time: 0.049401000 seconds]
```

After getting the response from the DNS Resolver, the Windows PC sends a TCP packet with the SYN flag to the web server to establish a connection. The client's TCP state changes from CLOSED to SYN-SENT, and it uses the resolved IP address to initiate the connection.
It uses

Source ip(wind ip)
Dest ip(dns server ip)
Sourceport(random)
Destination port is 443 (https)
mac address (next-hop)
seq=0

WinPC:
IP: 11.1.1.11
Mac: 5000:0060:0000
Router:
Mac - G0/2 - 5000.0002.0003
- G0/0 - 5000.0002.0000
- G0/1 - 5000.0002.0001
Web-Web Server:
G0/0 -5000.0005.0000

client -> web server



After receiving the TCP (syn) flag from client, the web-server sends ack in response to the syn flag and sends its own syn to establish the connection with the user, then state of the server changes from listen to sync received, and it uses:

web server -> client

Source ip: server ip
dest ip: client ip
Source port is (443) https server
dest port (client port no)
ack = received seq+1
syn = 0

Phantom byte (segment)

```
> Frame 242: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
> Ethernet II, Src: 50:00:00:05:00:01 (50:00:00:05:00:01), Dst: 50:00:00:02:00:01 (50:00:00:02:00:01)
> Internet Protocol Version 4, Src: 30.1.1.100, Dst: 11.1.1.11
v Transmission Control Protocol, Src Port: 443, Dst Port: 49690, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 49690
    [Stream index: 0]
    [Stream Packet Number: 2]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1715819089
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 4154140376
    0110 .... = Header Length: 24 bytes (6)
  > Flags: 0x012 (SYN, ACK)
    Window: 4128
    [Calculated window size: 4128]
    Checksum: 0xc541 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (4 bytes), Maximum segment size
  > [Timestamps]
  > [SEQ/ACK analysis]
```

After receiving the TCP (syn + ack) flag from server, the windows pc(browser) sends ack in response to the sync and its state change from syn sent to established.
it uses:

Source ip: own ip
dest ip: client ip
Source port (443)
dest port (client port no)
ack = received seq+1
(received ack if correct)

WinPC:
IP: 11.1.1.11
Mac: 5000:0060:0000
Router:
Mac - G0/2 - 5000.0002.0003
- G0/0 - 5000.0002.0000
- G0/1 - 5000.0002.0001
Web-Web Server:
G0/0 -5000.0005.0000

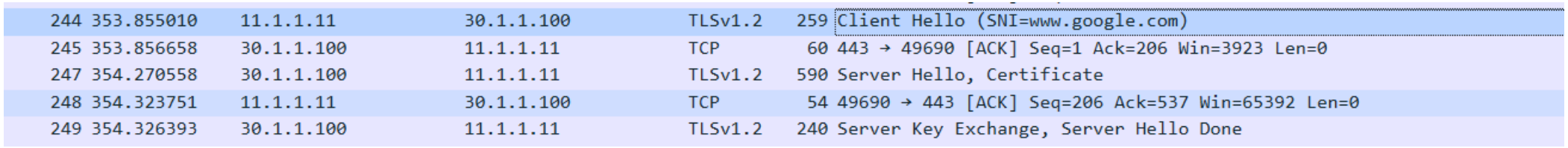
client -> web server

```
> Frame 518: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface -, id 0
> Ethernet II, Src: 50:00:00:06:00:00 (50:00:00:06:00:00), Dst: 50:00:00:02:00:03 (50:00:00:02:00:03)
> Internet Protocol Version 4, Src: 11.1.1.11, Dst: 30.1.1.100
v Transmission Control Protocol, Src Port: 49690, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 49690
    Destination Port: 443
    [Stream index: 0]
    [Stream Packet Number: 3]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 4154140376
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1715819090
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 65392
    [Calculated window size: 65392]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xea11 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
```

When server receives ack from client, then the server state changes from sync received to established. Now the client can send the data.

SSL later version - TLS
TLS 1.2 is a cryptographic protocol used to secure communications over networks, such as internet
HTTPS = HTTP + TLS

Before sending data, tls handshake occurs between winpc and server.



244	353.855010	11.1.1.11	30.1.1.100	TLSv1.2	259 Client Hello (SNI=www.google.com)
245	353.856658	30.1.1.100	11.1.1.11	TCP	60 443 → 49690 [ACK] Seq=1 Ack=206 Win=3923 Len=0
247	354.270558	30.1.1.100	11.1.1.11	TLSv1.2	590 Server Hello, Certificate
248	354.323751	11.1.1.11	30.1.1.100	TCP	54 49690 → 443 [ACK] Seq=206 Ack=537 Win=65392 Len=0
249	354.326393	30.1.1.100	11.1.1.11	TLSv1.2	240 Server Key Exchange, Server Hello Done

1. Windows Pc - Web Server

Windows Pc Hello
Supported TLS versions
Supported cipher suites
Windows Pc random
SNI (server name indication — the domain it wants)
(used when share hosting comes into picture)

2. Web Server-Windows Pc

Ack for Windows Pc packets
Web Server Hello
Web Server picks TLS version, cipher suite
Sends:

- Web Server random
- Digital certificate (used to verify identity)

3. Windows Pc – Web Server

Ack packet for received hello and other messages

4. Web Server-Windows Pc

SKE: Sends server's part of key exchange info
Hello done: Indicates end of server handshake part

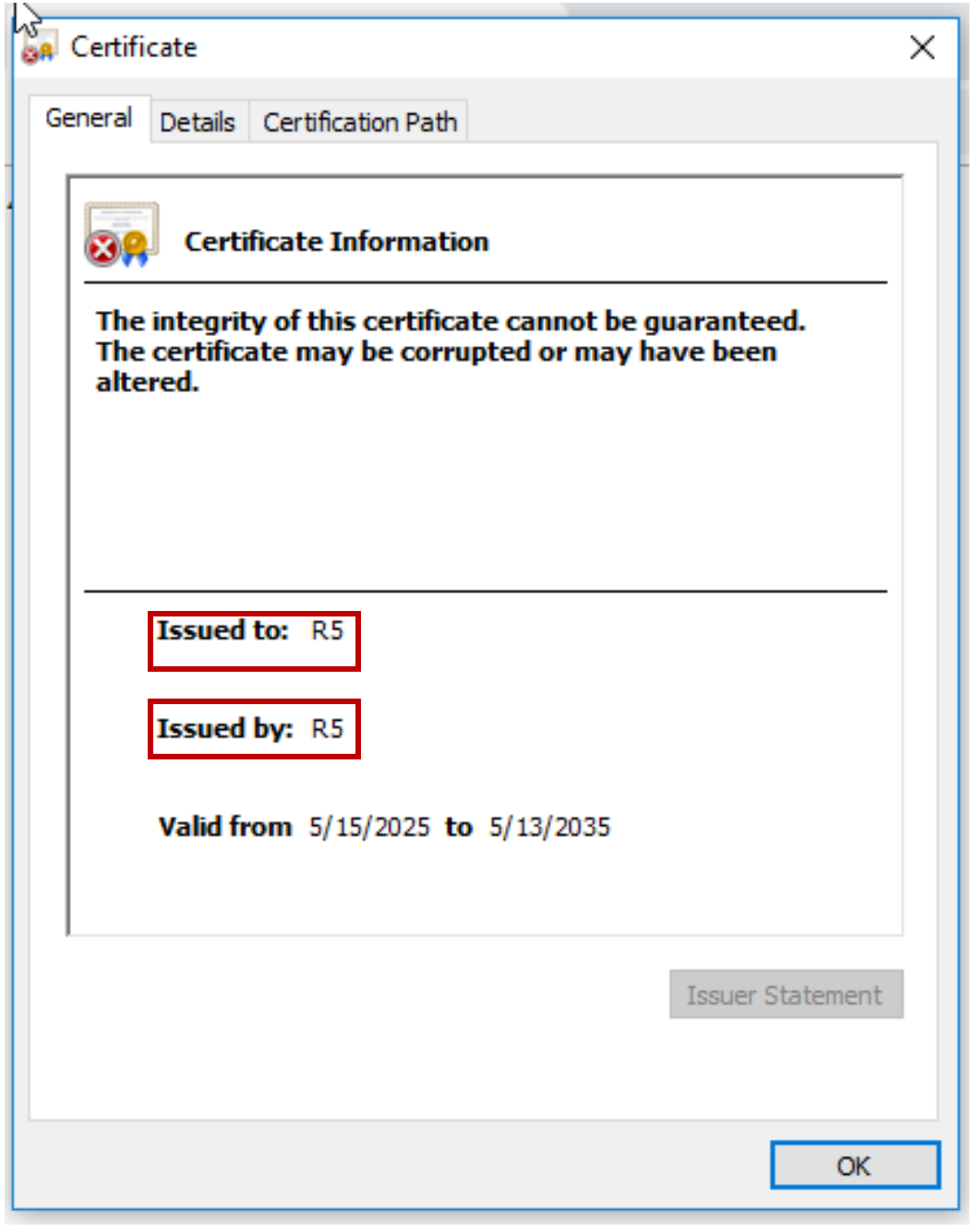
11.1.1.11	30.1.1.100	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
30.1.1.100	11.1.1.11	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message

5. Windows Pc – Web Server

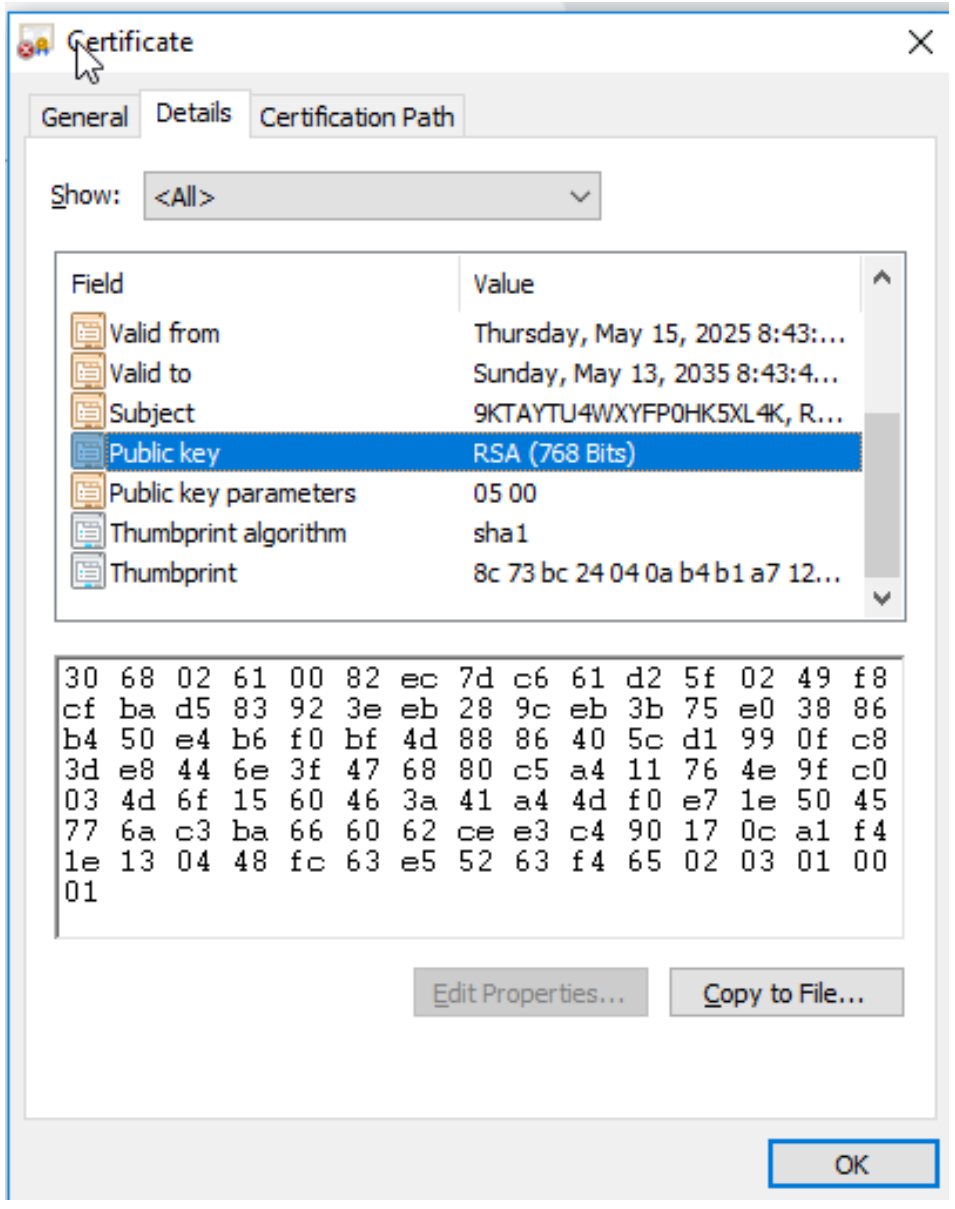
Generates pre-master key
Encrypts it with server public key
Sends Change Cipher Spec (about to start encrypted communication)
Sends a Finished message, it is like the first message encrypted with the session key to confirm handshake integrity

6. Web Server – Windows Pc

Web Server also switches to encrypted mode
and Sends final Finished message



When I enter the credential, browser is showing this site is not secure, means it uses self-signed certificate



Now, the Windows browser is sending data to the Web server using HTTP over a TLS-encrypted TCP connection. The data is encrypted, and the server responds with an ACK flag to acknowledge the received packet.

client -> web server

242	178.892102	11.1.1.11	30.1.1.100	TLSv1.2	561	Application Data
243	178.902072	30.1.1.100	11.1.1.11	TCP	60	443 → 49760 [ACK] Seq=774 Ack=871 Win=4128 Len=0

```
> Frame 242: 561 bytes on wire (4488 bits), 561 bytes captured (4488 bits) on interface -, id 0
> Ethernet II, Src: 50:00:00:02:00:01 (50:00:00:02:00:01), Dst: 50:00:00:05:00:01 (50:00:00:05:00:01)
> Internet Protocol Version 4, Src: 11.1.1.11, Dst: 30.1.1.100
> Transmission Control Protocol, Src Port: 49760, Dst Port: 443, Seq: 364, Ack: 774, Len: 507
v Transport Layer Security
  v TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 502
      Encrypted Application Data […]: 00000000000000175dbb0e2acd72a6c6080c3f35b02b7fa7d0f9cae8b9fc
      [Application Data Protocol: Hypertext Transfer Protocol]
```

Now, Web server is sending data to the Windows Pc (browser) using HTTP over a TLS-encrypted TCP connection. The data is encrypted, and the Client responds with an ACK flag to acknowledge the received packet of the Web Server.

245	178.954353	30.1.1.100	11.1.1.11	TLSv1.2	590 Application Data, Application Data
246	178.956313	30.1.1.100	11.1.1.11	TLSv1.2	543 Application Data
247	178.957377	11.1.1.11	30.1.1.100	TCP	54 49760 → 443 [ACK] Seq=871 Ack=1595 Win=65392 Len=0
248	178.962859	30.1.1.100	11.1.1.11	TLSv1.2	590 Application Data, Application Data
249	178.965131	11.1.1.11	30.1.1.100	TCP	54 49760 → 443 [ACK] Seq=871 Ack=2620 Win=65392 Len=0

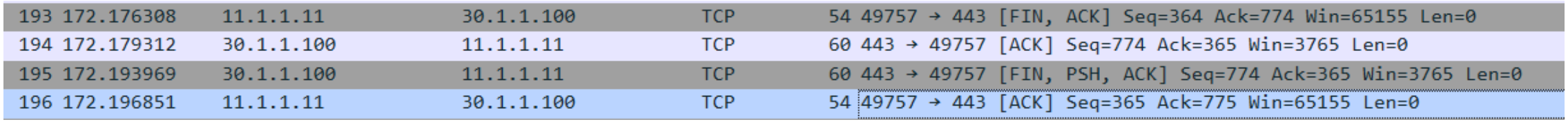
TCP 4 way handshake to break the connection:

At last, Windows Pc (br) sends fin flag to indicate that I have done sending data from my side, and it also sends ack for previous data it received from web server, the TCP state changes from established to fin_wait1. Then the server sends the ack flag for received FIN flag of client and the TCP state change from established to close wait.

After receiving the ACK flag from the web server, the TCP state of the PC changes from FIN_WAIT1 to FIN_WAIT2. Lastly, the server sends the buffered data using the PUSH flag, sends an ACK flag for the received sequence from the client, and then sends its own FIN flag to close the connection. The tcp state of the Web server changes from close_wait to last ack.

When pc receives the fin flag from the server, the TCP state of the pc changes from fin_wait2 to time-wait, after 2 milli seconds, the TCP state changes from time_wait to closed.

After receiving the ack flag from Windows pc, the TCP state of server changes from last_ack to closed. Finally the connection was closed from both side.



193	172.176308	11.1.1.11	30.1.1.100	TCP	54 49757 → 443 [FIN, ACK] Seq=364 Ack=774 Win=65155 Len=0
194	172.179312	30.1.1.100	11.1.1.11	TCP	60 443 → 49757 [ACK] Seq=774 Ack=365 Win=3765 Len=0
195	172.193969	30.1.1.100	11.1.1.11	TCP	60 443 → 49757 [FIN, PSH, ACK] Seq=774 Ack=365 Win=3765 Len=0
196	172.196851	11.1.1.11	30.1.1.100	TCP	54 49757 → 443 [ACK] Seq=365 Ack=775 Win=65155 Len=0