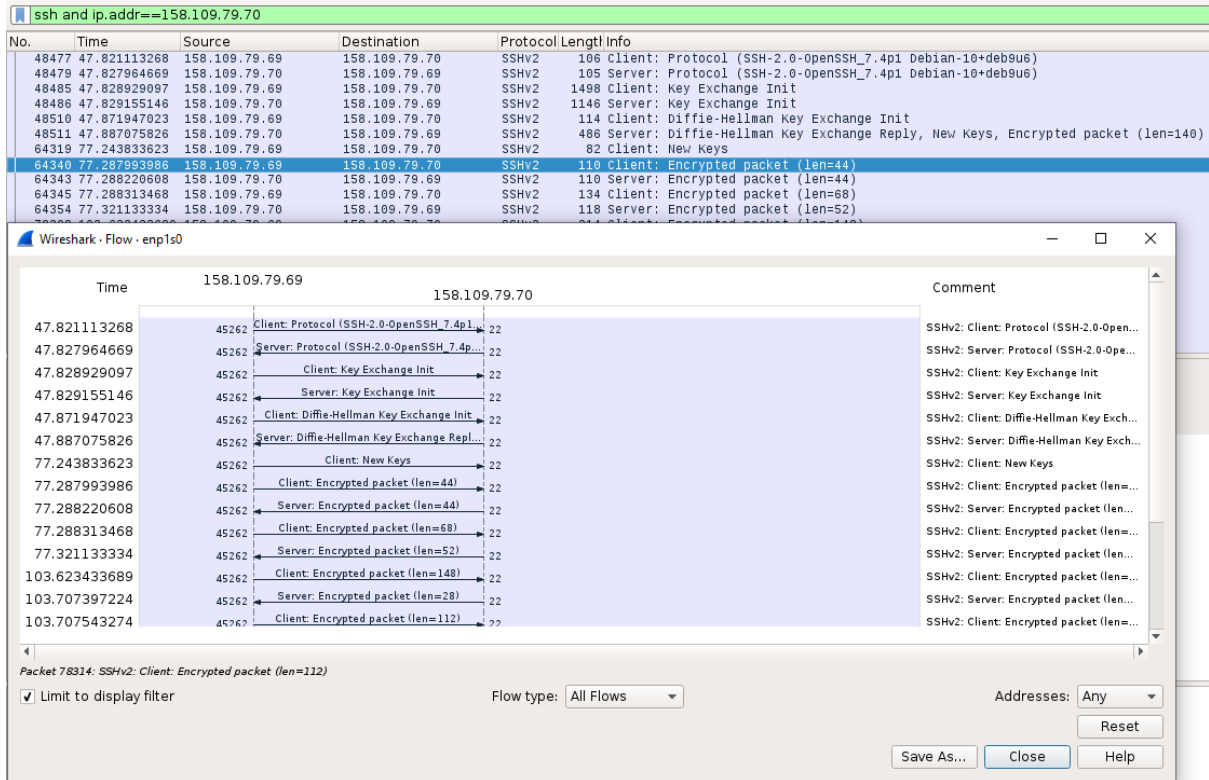


### 3.3. Tráfico SSH

1. (1 punto) Filtre en Wireshark para que capture solo tráfico SSH. Conéctese a través de SSH a otra maquina con `ssh usuario@DEIC-DCX`. Una vez iniciada la sesión detenga la captura. Utilice la herramienta Flow Graph, haga una captura del gráfico desde el inicio de la negociación TCP hasta el envío de los 3 primeros datos cifrados. Resalte el método (dentro de un canal público) de intercambio de llaves utilizado.



**2. (1 punto) ¿Cuál es el puerto local que se utiliza para la conexión SSH?**

Cómo se puede observar el ssh utiliza el puerto 22 y el de vuelta 45262.

**3. (1 punto) ¿Qué versión de SSH, cifrado, Message Authentication Code (MAC) y compresión se negoció?. Colocar una imagen**

```
SSH Protocol
└─ SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)
   Packet Length: 260
   Padding Length: 10
   └─ Key Exchange
      Message Code: Diffie-Hellman Key Exchange Reply (31)
      └─ KEX host key (type: ecdsa-sha2-nistp256)
         Host key length: 104
         Host key type: ecdsa-sha2-nistp256
         ECDSA elliptic curve identifier length: 8
         ECDSA elliptic curve identifier: nistp256
         ECDSA public key length: 65
         ECDSA public key (Q): 045a8c68060de8e27f95fce31c5d27c42cf9c9d7e4766a61...
         Multi Precision Integer Length: 32
         DH server f: 0b28cf10d4c56fcd4a7b3b7dbc9ce5c842bfa2e96aa4aec...
         KEX H signature length: 100
         KEX H signature: 0000001365636473612d736861322d6e6973747032353600...
         Padding String: 000000000000000000000000

SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)
Packet Length: 12
Padding Length: 10
└─ Key Exchange
   Message Code: New Keys (21)
   Padding String: 000000000000000000000000

SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)
Packet Length (encrypted): 5412189e
Encrypted Packet: 2fbb398c25fac42c4451bf74d95536e7f26292d5615f97a8...
MAC: a0ca7f20238c499ee50ac448935276c4
```

### 3.4 Tráfico HTTPS

A continuación vamos a crear un servidor que sirve una página Web en HTTPS y utilizaremos Wireshark para ver como se puede capturar esta información y qué información podemos obtener.

1. Comenzaremos creando una llave RSA y un certificado para el servidor, abrimos una consola y los generamos con el siguiente comando:

```
openssl req -new -x509 -out server.crt -nodes -keyout server.pem -subj /CN=localhost
```

Con este pedido solicitamos a OpenSSL que nos genere un certificado (-NEW) autofirmado (-x509) que se guardara en el archivo ( -out server.crt), pedimos que nos genere una llave privada (-keyout server.pem) y que no la cifra (-nodes) y especificamos el CommonName del certificado (-subj / CN = localhost).

2. Una vez que tengamos el certificado y la llave privada, podemos abrir el servidor con el siguiente comando:

```
openssl s server -tls1 2 -www -cipher AES256-SHA -key server.pem -cert server.crt
```

En este caso estamos requiriendo a OpenSSL que nos abra un servidor que utiliza AES para cifrar ( -cipher AES256-SHA) que utilizara la llave privada (-key server.pem) y el certificado (-cert server.crt) que acabamos de crear y que nos responderá a las peticiones con un mensaje de estado (-www).

3. Ahora ejecutamos Wireshark para capturar el tráfico que se genere hacia y desde localhost. Se recomienda aplicar como filtro de pantalla "ssl"

4. Una vez iniciemos la captura del tráfico, en una nueva consola haremos la petición Web con el comando:

```
printf 'GET / HTTP/1.0\r\n\r\n' | openssl s_client -ign_eof
```

5. (1 punto) Explica lo que has obtenido en esta captura. Colocar una imagen

Al no tener la clave para el descifrado SSL, Wireshark ve todos los paquetes cifrados sin saber la información que contienen.

ip.src=127.0.0.1					
No.	Time	Source	Destination	Protocol	Length Info
1	0.000000000	88.24.171.249	158.109.79.69	SSH	150 Client: Encrypted packet (len=96)
2	0.000238305	158.109.79.69	88.24.171.249	SSH	166 Server: Encrypted packet (len=112)
3	0.000693850	158.109.79.69	88.24.171.249	SSH	150 Server: Encrypted packet (len=96)
4	0.000948313	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
5	0.001368330	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
6	0.001649181	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
7	0.002156803	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
8	0.002458180	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
9	0.002986780	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
10	0.003288987	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
11	0.003816766	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
12	0.004113108	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
13	0.004366855	88.24.171.249	158.109.79.69	TCP	60 62557 → 22 [ACK] Seq=97 Ack=209 Win=515 Len=0
14	0.004660899	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
15	0.004959873	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
16	0.005489630	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
17	0.005716255	88.24.171.249	158.109.79.69	SSH	150 Client: Encrypted packet (len=96)
18	0.005804472	158.109.79.69	88.24.171.249	SSH	134 Server: Encrypted packet (len=80)
19	0.006375451	158.109.79.69	88.24.171.249	SSH	182 Server: Encrypted packet (len=128)

▶ Frame 1: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0

▶ Ethernet II, Src: IntelCor\_91:7c:e9 (00:15:17:91:7c:e9), Dst: Micro-St\_93:0f:1a (6c:62:6d:93:0f:1a)

▶ Internet Protocol Version 4, Src: 88.24.171.249, Dst: 158.109.79.69

▼ Transmission Control Protocol, Src Port: 62557, Dst Port: 22, Seq: 1, Ack: 1, Len: 96

Source Port: 62557  
Destination Port: 22  
[Stream index: 0]  
[TCP Segment Len: 96]  
Sequence number: 1 (relative sequence number)  
[Next sequence number: 97 (relative sequence number)]  
Acknowledgment number: 1 (relative ack number)  
0101 .... = Header Length: 20 bytes (5)  
▶ Flags: 0x018 (PSH, ACK)  
Window size value: 516  
[Calculated window size: 516]  
[Window size scaling factor: -1 (unknown)]  
Checksum: 0xe503 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
▶ [SEQ/ACK analysis]  
▶ [Timestamps]

0000	6c 62 6d 93 0f 1a 00 15	17 91 7c e9 08 00 45 00	1bm..... ...E
0010	00 88 5b 0b 40 00 75 06	b8 a0 58 18 ab f9 9e 6d	..[.@.u...X...m

ip.src!=127.0.0.1						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	88.24.171.249	158.109.79.69	SSH	150	Client: Encrypted packet (len=96)
2	0.000238305	158.109.79.69	88.24.171.249	SSH	166	Server: Encrypted packet (len=112)
3	0.000693850	158.109.79.69	88.24.171.249	SSH	150	Server: Encrypted packet (len=96)
4	0.000948313	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
5	0.001368330	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
6	0.001649181	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
7	0.002156803	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
8	0.002458180	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
9	0.002986780	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
10	0.003288987	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
11	0.003816766	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
12	0.004113108	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
13	0.004366855	88.24.171.249	158.109.79.69	TCP	60	62557 → 22 [ACK] Seq=97 Ack=209 Win=515 Len=0
14	0.004660899	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
15	0.004959873	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
16	0.005489630	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
17	0.005716255	88.24.171.249	158.109.79.69	SSH	150	Client: Encrypted packet (len=96)
18	0.005804472	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
19	0.006375451	158.109.79.69	88.24.171.249	SSH	182	Server: Encrypted packet (len=128)
20	0.009617521	88.24.171.249	158.109.79.69	TCP	60	62557 → 22 [ACK] Seq=193 Ack=529 Win=514 Len=0
21	0.009642310	158.109.79.69	88.24.171.249	SSH	774	Server: Encrypted packet (len=720)
22	0.010066959	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
23	0.010364362	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
24	0.010892721	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
25	0.012514620	88.24.171.249	158.109.79.69	TCP	60	62557 → 22 [ACK] Seq=193 Ack=1377 Win=510 Len=0
26	0.012540749	158.109.79.69	88.24.171.249	SSH	374	Server: Encrypted packet (len=320)
27	0.012807756	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
28	0.013262278	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
29	0.013532186	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
30	0.013635928	88.24.171.249	158.109.79.69	TCP	60	62557 → 22 [ACK] Seq=193 Ack=2177 Win=516 Len=0
31	0.013835289	88.24.171.249	158.109.79.69	TCP	60	62557 → 22 [ACK] Seq=193 Ack=2337 Win=515 Len=0
32	0.014108083	158.109.79.69	88.24.171.249	SSH	134	Server: Encrypted packet (len=80)
33	0.014355739	158.109.79.69	88.24.171.249	SSH	150	Server: Encrypted packet (len=96)
34	0.014566958	158.109.79.69	88.24.171.249	SSH	310	Server: Encrypted packet (len=256)
35	0.014781216	158.109.79.69	88.24.171.249	SSH	374	Server: Encrypted packet (len=320)

Frame 1: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0

Ethernet II, Src: IntelCor\_91:7c:e9 (00:15:17:91:7c:e9), Dst: Micro-St\_93:0f:1a (6c:62:6d:93:0f:1a)

Internet Protocol Version 4, Src: 88.24.171.249, Dst: 158.109.79.69

Transmission Control Protocol, Src Port: 62557, Dst Port: 22, Seq: 1, Ack: 1, Len: 96

Source Port: 62557

Destination Port: 22

0000 6c 62 6d 93 0f 1a 00 15 17 91 7c e9 08 00 45 00 1bm.....E

0010 00 88 5b 0b 40 00 75 06 b8 a0 58 18 ab f9 9e 6d ..[.@u..X...m

6. Ahora le diremos a Wireshark cual es la llave privada que se está utilizando en esta comunicación, para hacer esto iremos a “Edit”→“Preferences”, dentro de Protocolos buscaremos SSL y modificaremos la RSA keys lista, con los parámetros: IP address - 127.0.0.1, Puerto - 4433, Protocolo - http y Key File - server.pem. (En caso de que no se descifran los mensajes cambiar la IP address por ::1). Finalmente al darle doble click sobre uno de los paquetes cifrados se habilitará una nueva pestaña Decrypted SSL al lado de Frame

7. (1 punto) ¿Qué cambios se han producido en los paquetes? ¿Cuál es la respuesta del servidor a la petición GET?. Colocar la imagen obtenida al ingresar a Follow SSL Stream.

Como ahora Wireshark tiene acceso a la clave de cifrado puede descifrar los paquetes que ha capturado y leer su contenido. Gracias a eso es capaz de ver los paquetes con los mensajes de “Client Hello” y “Server Hello” por ejemplo.

ssl						
No.	Time	Source	Destination	Protocol	Length	Info
1908	7.115829824	::1	::1	TLSv1.2	262	Client Hello
1910	7.116117447	::1	::1	TLSv1.2	946	Server Hello, Certificate, Server Hello Done
1912	7.117364706	::1	::1	TLSv1.2	432	Client Key Exchange, Change Cipher Spec, Finished
1913	7.121939882	::1	::1	TLSv1.2	340	New Session Ticket, Change Cipher Spec, Finished
1914	7.122527479	::1	::1	HTTP	159	GET / HTTP/1.0
1915	7.122726341	::1	::1	TLSv1.2	1871	[SSL segment of a reassembled PDU]
1918	7.122937559	::1	::1	TLSv1.2	143	Alert (Level: Warning, Description: Close Notify)