

OpenSSH: Intercambio de Claves ECDH y Protocolos Poscuánticos (Kyber)

March 6, 2025

① Introducción a SSH

Concepto y Funcionamiento

② Intercambio de Claves en SSH

Algoritmos Tradicionales

③ ECDH en OpenSSH

Criptografía de Curva Elíptica

④ Protocolos Poscuánticos

Amenaza Cuántica

⑤ Kyber en OpenSSH

Implementación y Características

⑥ Kyber: Criptografía Poscuántica

¿Qué es Kyber?

Funcionamiento de Kyber

Matemáticas detrás de Kyber

¿Qué es SSH?

Secure Shell (o **Secure Socket Shell**) es un protocolo de red que proporciona:

- ① Acceso seguro a ordenadores a través de redes no seguras
- ② Autenticación robusta para administradores de sistemas

SSH también se refiere al conjunto de utilidades que implementan este protocolo.

Importancia

SSH es importante para mantener la seguridad de los sistemas, ya que el protocolo actúa como una forma segura de proporcionar acceso y gestión de sistemas en red.

Además de proporcionar un fuerte cifrado, SSH es ampliamente utilizado por los administradores de red para:

- ① Gestionar sistemas y aplicaciones de forma remota
- ② Iniciar sesión en otro ordenador a través de una red
- ③ Ejecutar comandos
- ④ Mover archivos de un ordenador a otro

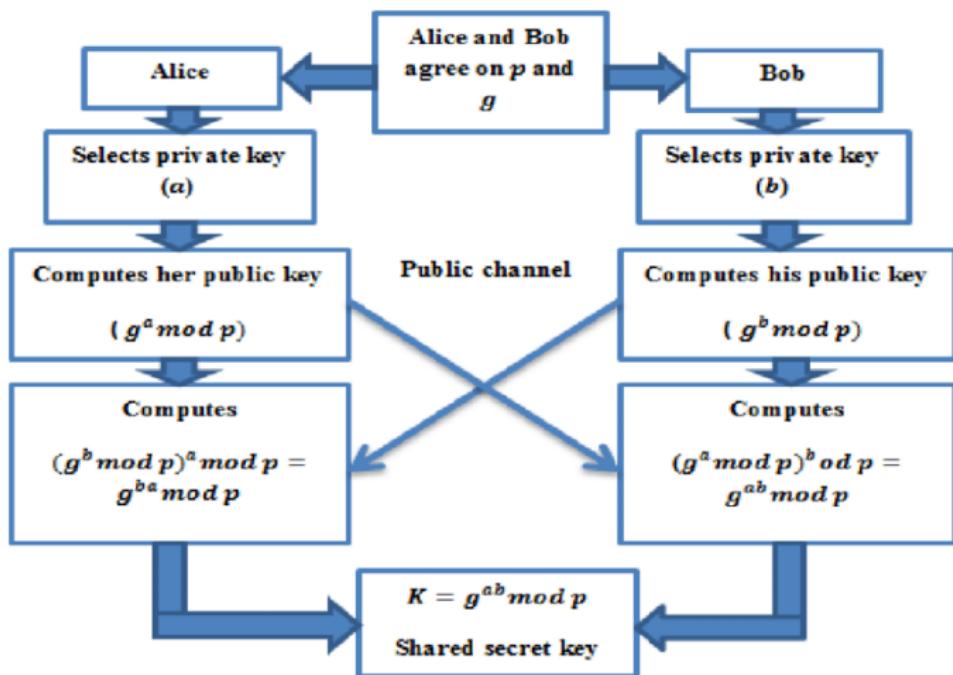
Modelo Cliente-Servidor

SSH utiliza el modelo cliente-servidor:

- ① La aplicación cliente muestra la sesión
- ② El servidor SSH ejecuta la sesión
- ③ Por defecto, escucha en el puerto TCP 22

Las implementaciones incluyen soporte para emulación de terminal y transferencias de archivos.

Intercambio DH



Problema del logaritmo

$$2^n = 16$$

¿Cuál es el valor de n ?

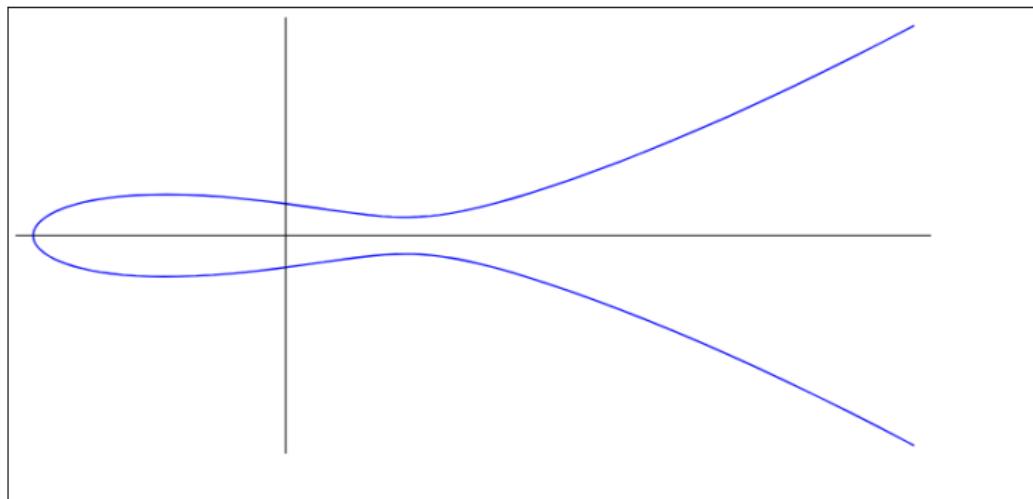
Problema del logaritmo discreto

$$2^n \mod 17 = 16$$

¿Cuál es el valor de n ?

Algoritmos Tradicionales

Curvas Elípticas al rescate



Aritmética en $GF(p)$

Sea $E(a, b)$ una curva elíptica sobre $GF(p)$ con puntos P, Q . Se cumple:

$$P + O = P, \quad -P = (x_1, -y_1), \quad P + (-P) = O.$$

Si $Q \neq -P$, entonces $P + Q = (x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1.$$

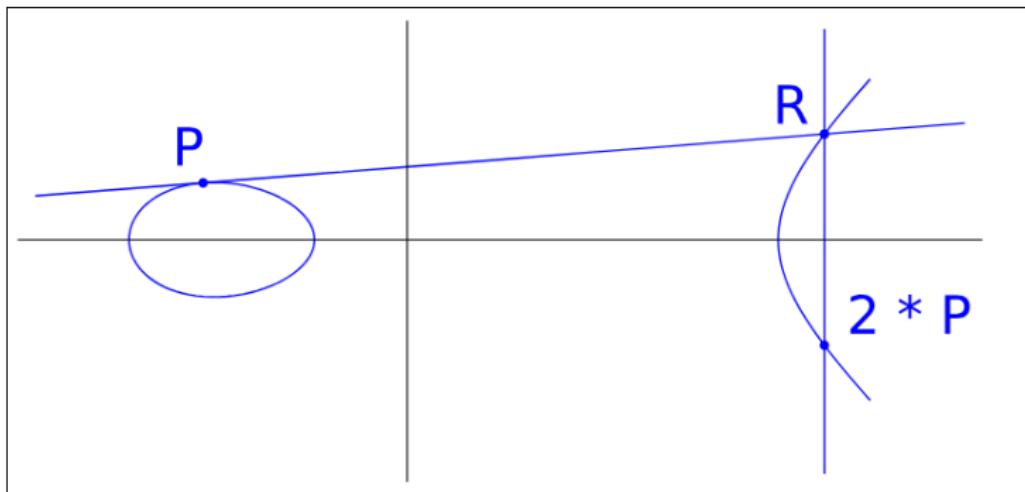
Con:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & P = Q \end{cases}$$

Algoritmos Tradicionales

Visualización puntos

Algoritmos Tradicionales



Problema del logaritmo discreto en curvas elípticas

Dada una curva elíptica E sobre un campo finito \mathbb{F}_p , un punto generador G y otro punto Q en la curva:

$$nG = Q$$

Algoritmos Tradicionales

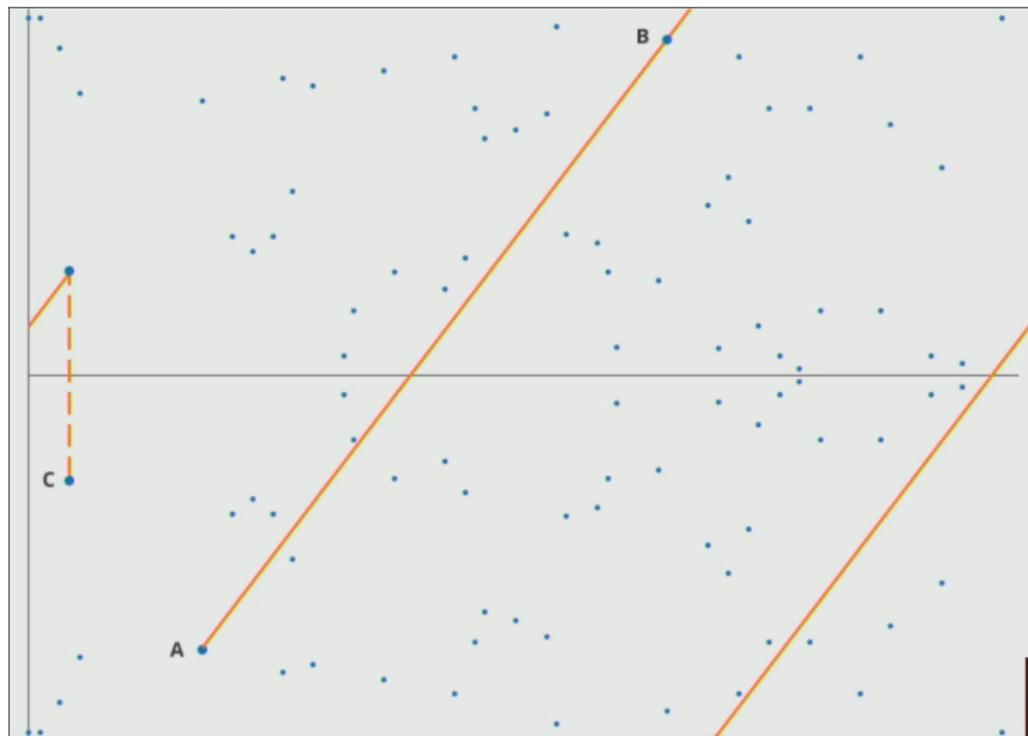
DH

$$(G^n)^c = (G^c)^n$$

EC

$$c(nG) = n(cG)$$

Algoritmos Tradicionales



ECDH vs DH tradicional

Problema Matemático:

- DH usa el **logaritmo discreto** en números primos.
- ECDH usa el **logaritmo discreto en curvas elípticas (ECDLP)**, más difícil de resolver.

Comparación de tamaños de clave:

Seguridad	DH tradicional	ECDH
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
192 bits	7680 bits	384 bits
256 bits	15360 bits	512 bits

Intercambio de Claves con ECDH

Parámetros Públicos: Curva elíptica E , punto generador G

Paso 1: Generación de Claves

- Alice elige una clave privada d_A y calcula su clave pública $Q_A = d_A \cdot G$
- Bob elige una clave privada d_B y calcula su clave pública $Q_B = d_B \cdot G$

Paso 2: Intercambio de Claves Públicas

- Alice y Bob intercambian sus claves públicas Q_A y Q_B

Paso 3: Cálculo del Secreto Compartido

- Alice calcula $S = d_A \cdot Q_B = d_A d_B G$
- Bob calcula $S = d_B \cdot Q_A = d_B d_A G$

Resultado: Ambos comparten la misma clave secreta S

Criptografía de Curva Elíptica

Private key (d_A)**Public key:**
 $Q_A = d_A \times G$ **Private key (d_B)****Public key:**
 $Q_B = d_B \times G$  Q_A Q_B **Shared key:**

$$\text{Share} = d_A \times Q_B$$

**Shared key:**

$$\text{Share} = d_A \times d_B \times G$$

Shared key:

$$\text{Share} = d_B \times Q_A$$

**Shared key:**

$$\text{Share} = d_B \times d_A \times G$$

Criptografía de Curva Elíptica

El proceso del Intercambio de Claves Diffie-Hellman está definido en la sección 8 del RFC4253 y el diagrama a continuación lo ilustra:

```
$ ssh -Q kex <remote ip>
diffie-hellman-group1-sha1
diffie-hellman-group14-sha1
diffie-hellman-group14-sha256
diffie-hellman-group16-sha512
diffie-hellman-group18-sha512
diffie-hellman-group-exchange-sha1
diffie-hellman-group-exchange-sha256
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521
curve25519-sha256
curve25519-sha256@libssh.org
sntrup4591761x25519-sha512@tinyssh.org
```

LINK RFC 4253, sección 8

Grupos Diffie-Hellman

Un grupo Diffie-Hellman define el tamaño del número primo, el valor del generador y el tipo de algoritmo para generar la clave secreta.

Existen dos tipos de grupos Diffie-Hellman:

- **MODP (Exponenciación Modular)**: Definido en el RFC3526.
- **ECP (Criptografía de Curvas Elípticas)**: Definido en el RFC4753.

ssh -o KexAlgorithms=<nombre del grupo dh> <ip remota> LINK RFC 3526

Criptografía de Curva Elíptica

Type	Group name	Size (bits)
MODP	Diffie-Hellman Group 1	768
MODP	Diffie-Hellman Group 2	1024
MODP	Diffie-Hellman Group 5	1536
MODP	Diffie-Hellman Group 14	2048
MODP	Diffie-Hellman Group 15	3072
ECP	Diffie-Hellman Group 19	256
ECP	Diffie-Hellman Group 20	384
ECP	Diffie-Hellman Group 21	512

Criptografía de Curva Elíptica

MITM y DF en linux

```
$ ssh user@192.168.1.68
@@@@@@@@@@@oooooo
@      WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!      @
@@@@@@@@@@@oooooo
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:SsSxnPC8Z6BTy/JdftxNc6PJcJOKLRRxcGeINRJTviM.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of
this message.
Offending ECDSA key in /home/user/.ssh/known_hosts:86
    remove with:
        ssh-keygen -f "/home/user/.ssh/known_hosts" -R "192.168.1.68"
Host key for 192.168.1.68 has changed and you have requested strict checking.
Host key verification failed.
```

Criptografía de Curva Elíptica

Curvas Utilizadas en OpenSSH

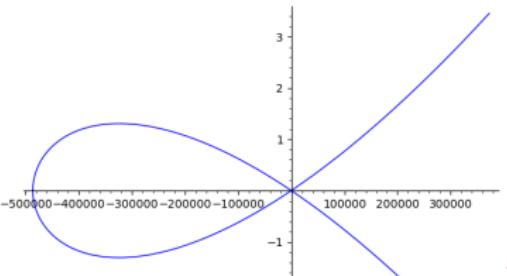
OpenSSH implementa varias curvas elípticas:

- ① Curve25519 (preferida por seguridad y rendimiento)
- ② NIST P-256, P-384 y P-521
- ③ Brainpool (en algunas configuraciones específicas)

Criptografía de Curva Elíptica

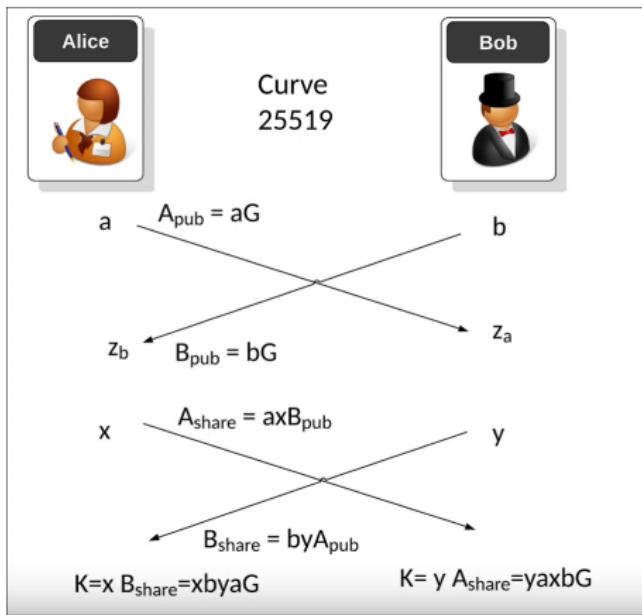
Curva 25519

- Creada por **Daniel J Bernstein**.
- Curve 25519 es uno de los métodos de criptografía ECC más utilizados.
- Utiliza la curva $y^2 = x^3 + 486662x^2 + x$, que es una curva de Montgomery.
- El número primo utilizado es $2^{255} - 19$.
- El punto base G está en $(9, 1478161944758954479102059356840998688726460 6134616475288964881837755586237401)$.



Criptografía de Curva Elíptica

Curva en el protocolo



Desafío de la Computación Cuántica

Los ordenadores cuánticos suponen una amenaza para:

- ① Algoritmos basados en factorización (RSA)
- ② Algoritmos basados en logaritmo discreto (DH)
- ③ ECDH (vulnerable al algoritmo de Shor)

Necesidad de Alternativas Resistentes

La seguridad a largo plazo requiere:

- ① Nuevos algoritmos resistentes a ataques cuánticos
- ② Implementaciones eficientes y seguras
- ③ Compatibilidad con infraestructuras existentes
- ④ Estandarización internacional

Integración en OpenSSH

OpenSSH ha implementado soporte para Kyber:

- ① Integración híbrida con ECDH para mayor seguridad
- ② Disponible en versiones recientes (Open Quantum-Safe)
- ③ Configuración mediante directivas específicas
- ④ Compatibilidad con infraestructuras existentes

Implementación y Características

Algoritmo Kyber

Kyber es un finalista del NIST para criptografía poscuántica:

- ① Basado en problemas de retículos
- ② Resistente a ataques cuánticos conocidos
- ③ Ofrece buen equilibrio entre seguridad y eficiencia
- ④ Seleccionado para estandarización por el NIST

Implementación y Características

Implementación de MITM

```
cripto@cripto:~/Desktop$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/cripto/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cripto/.ssh/id_ed25519
Your public key has been saved in /home/cripto/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:b3PZthGSyejKp4xiFdgjHIVinoOxxxy4lH1HQ0krLt+U ccripto@cripto
The key's randomart image is:
+--[ED25519 256]--+
| .o
| +
| =
| ..= = + o o
| +o.o . oS . = .
| ++o. . o + .
| oBo.+ = o +
| ++++.oE + o.o . o
| + o. . =o . .
+---[SHA256]-----+
cripto@cripto:~/Desktop$
```

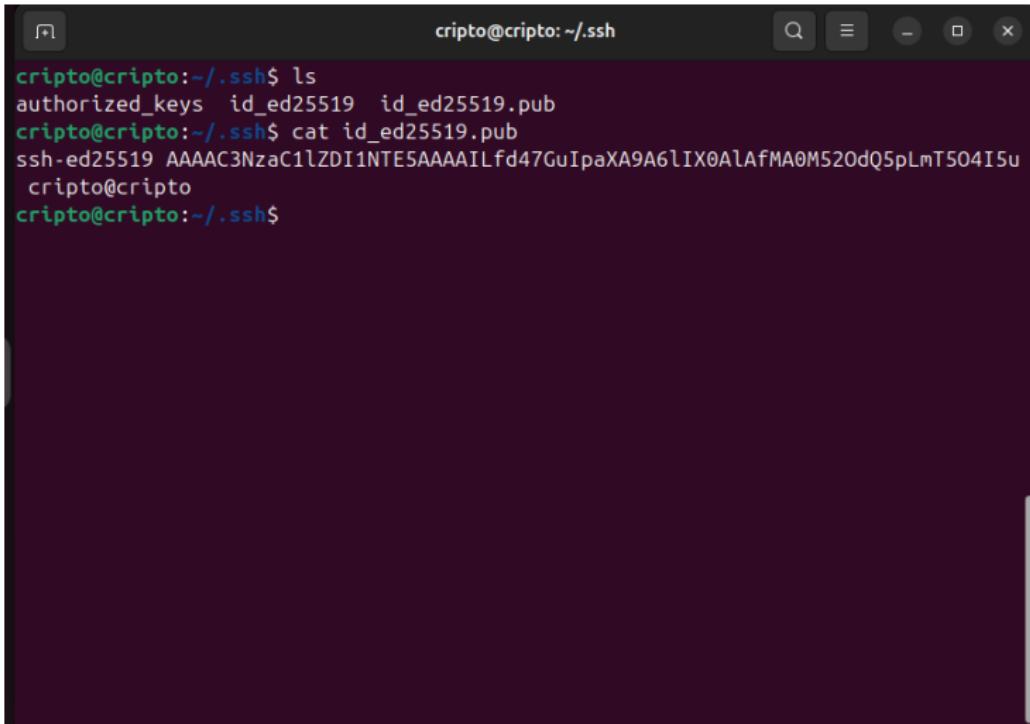
Figure: Generación de llave ECC en el servidor (Ubuntu) - Elaboración propia.

Implementación y Características

```
+----[SNIAZ30]-----+
cripto@cripto-2024:~/ssh$ ls -l
total 8
-rw----- 1 cripto cripto 411 Feb 22 11:41 id_ed25519
-rw-r--r-- 1 cripto cripto 100 Feb 22 11:41 id_ed25519.pub
cripto@cripto-2024:~/ssh$
```

Figure: Llaves públicas y privadas ECC

Implementación y Características

A screenshot of a terminal window titled "cripto@cripto: ~/ssh". The window contains the following text:

```
cripto@cripto:~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
cripto@cripto:~/ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAILfd47GuIpaXA9A6lIX0AlAfMA0M520dQ5pLmT504I5u
cripto@cripto:~/ssh$
```

The terminal has a dark background and light-colored text. The title bar and window controls are visible at the top.

Figure: Visualización de llave pública.

Implementación y Características

```
cripto@cripto: ~/ssh
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

cripto2@DESKTOP-8CLBMD4 C:\Users\cripto2>exit
Connection to 192.168.1.19 closed.

cripto@cripto:~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old
cripto@cripto:~/ssh$ ssh-copy-id cripo2@192.168.1.19
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
cripto2@192.168.1.19's password:
'exec' is not recognized as an internal or external command,
operable program or batch file.

Number of key(s) added: 1

Now try logging into the machine, with:    "ssh 'cripto2@192.168.1.19'"
and check to make sure that only the key(s) you wanted were added.
```

Figure: Asignando la llave a la maquina Cliente(Windows)

Implementación y Características

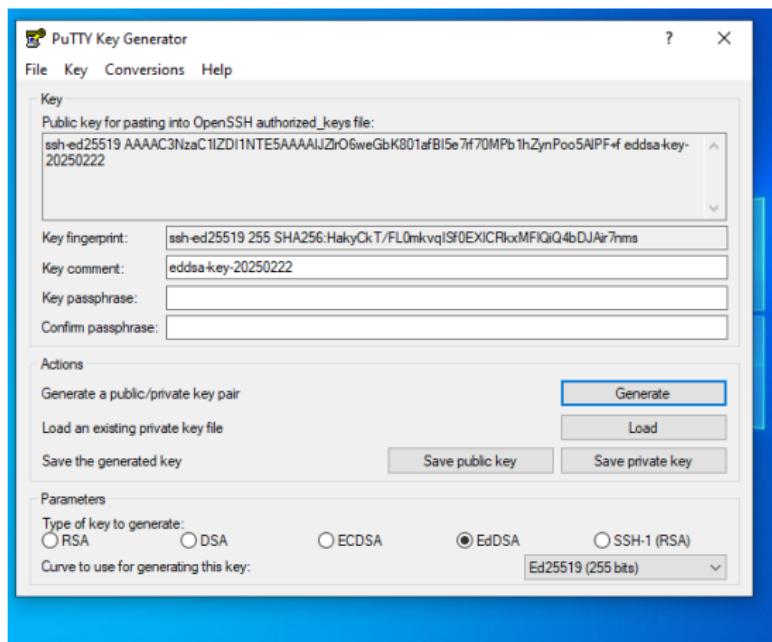


Figure: Generación llave ECC con Putty

Implementación y Características

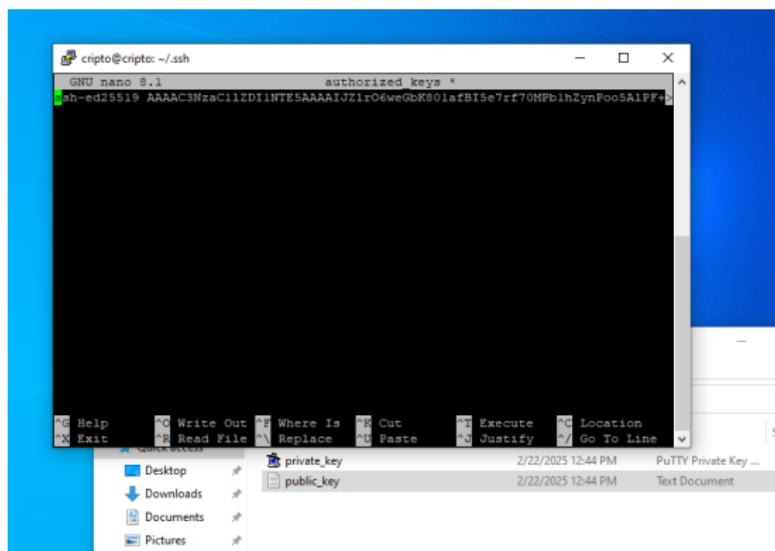


Figure: Visualización llave

Implementación y Características

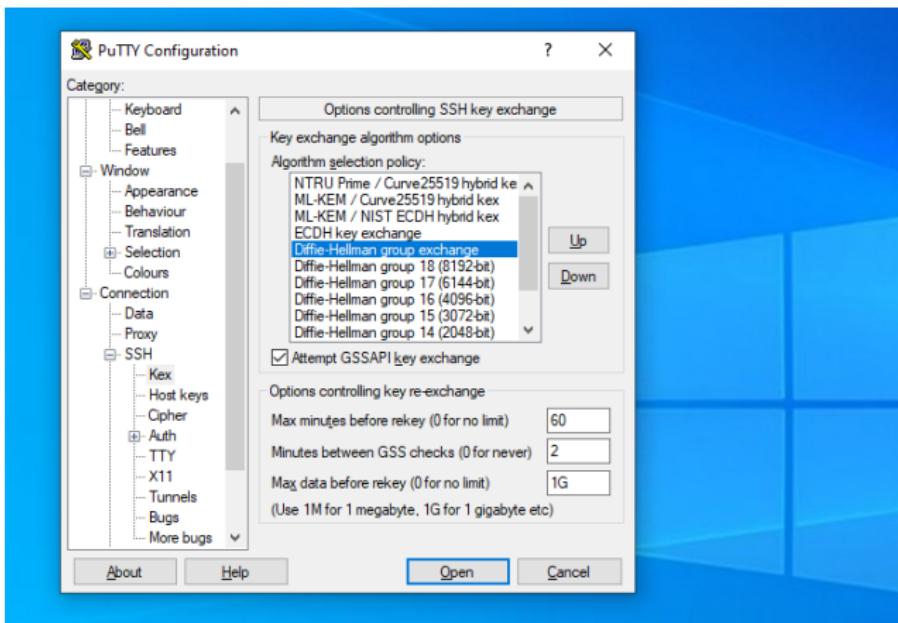


Figure: Intercambio de llaves Diffie-Hellman

Implementación y Características

Identificando los Hosts de ataque

The screenshot shows the Ettercap interface with the title "Ettercap 0.8.3.1 (EB)". The main window displays a "Host List" table with columns: IP Address, MAC Address, and Description. The table contains the following data:

IP Address	MAC Address	Description
192.168.1.7	3C:9C:0F:6E:07:D5	
fe80::20c:29ff:fe38:66e8	00:0C:29:38:66:E8	
fe80::4602:d1b:c26b:ce7d	00:0C:29:76:B3:CF	
fe80::b29c:dbbf:d252:e5c2	3C:9C:0F:6E:07:D5	
192.168.1.17	00:0C:29:38:66:E8	
192.168.1.19	00:0C:29:76:B3:CF	

At the bottom of the host list table, there are three buttons: "Delete Host", "Add to Target 1", and "Add to Target 2". Below the table, the status message "GROUP 1: 192.168.1.7 3C:9C:0F:6E:07:D5" is displayed. Further down, it says "GROUP 2 : 192.168.1.19 00:0C:29:76:B3:CF" and "Scanning for merged targets (2 hosts)...".

At the very bottom of the interface, a message states "2 hosts added to the hosts list...".

Implementación y Características

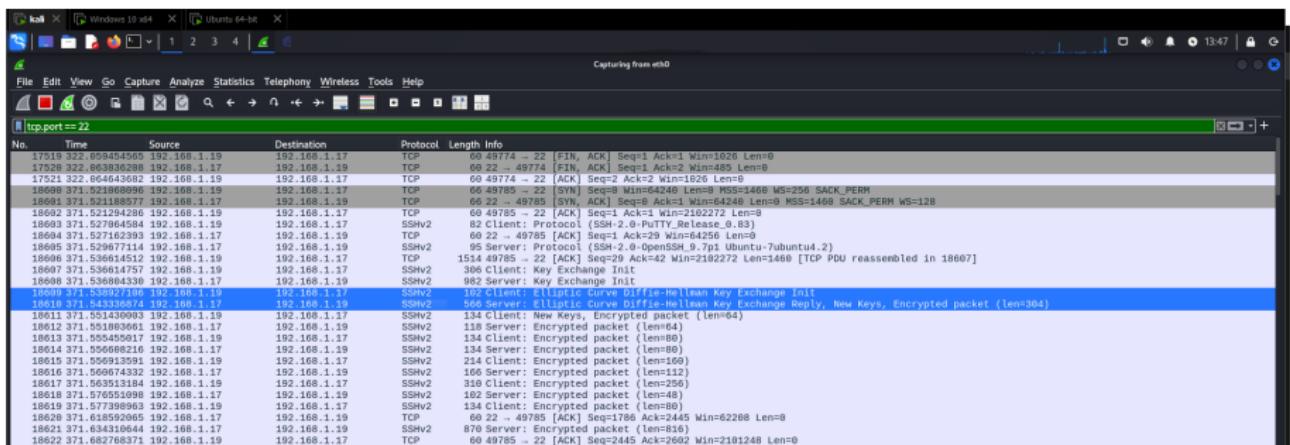


Figure: Captura de la conexión con Wireshark

Implementación y Características

```

tcp.port == 22
No. Time Source Destination Protocol Length Info
25217 466. 593095739 192.168.1.17 192.168.1.19 TCP 66 22 - 49822 [SYN, ACK] Seq=8 Ack=1 Win=64240 Len=0 MSS=1468 SACK_PERM WS=128
25218 466. 593193682 192.168.1.17 192.168.1.17 TCP 68 49822 - 22 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
25219 466. 512346382 192.168.1.17 192.168.1.19 SSHv2 95 Server: Protocol (SSH-2.0-OpenSSH_9.7p1 Ubuntu-7ubuntu4.2)
25220 466. 512918489 192.168.1.19 192.168.1.17 SSHv2 82 Client: Protocol (SSH-2.0-PuTTY_Release_0.83)
25221 466. 512976742 192.168.1.17 192.168.1.19 TCP 66 22 - 49822 [ACK] Seq=42 Ack=29 Win=64256 Len=0
25222 466. 514163956 192.168.1.17 192.168.1.19 SSHv2 98 Server: Key Exchange Init
25223 466. 515469656 192.168.1.19 192.168.1.17 TCP 1514 49822 - 22 [ACK] Seq=29 Ack=970 Win=2101248 Len=1400 [TCP PDU reassembled in 252]
25224 466. 515469718 192.168.1.19 192.168.1.17 SSHv2 308 Client: Key Exchange Init
25225 466. 515598247 192.168.1.17 192.168.1.19 TCP 66 22 - 49822 [ACK] Seq=970 Ack=1741 Win=62592 Len=0
25226 466. 518749433 192.168.1.19 192.168.1.17 SSHv2 102 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
25227 466. 522299652 192.168.1.17 192.168.1.19 SSHv2 566 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted pa...
25228 466. 530183426 192.168.1.19 192.168.1.17 SSHv2 134 Client: New Keys, Encrypted packet (len=64)
25229 466. 530381525 192.168.1.17 192.168.1.19 SSHv2 118 Server: Encrypted packet (len=64)

> Frame 25226: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface eth0, id 0
> Ethernet II, Src: VMware_76:b3:c f (00:0c:29:76:b3:c f), Dst: VMware_38:e8 (00:0c:29:38:66:e8)
> Internet Protocol Version 4, Src: 192.168.1.19, Dst: 192.168.1.17
    0100 ... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 88
    Identification: 0xfdc0 (64960)
> 010. .... = Flags: 0x2, Don't fragment
... 0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x796a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.19
    Destination Address: 192.168.1.17
    [Stream index: 50]
> Transmission Control Protocol, Src Port: 49822, Dst Port: 22, Seq: 1741, Ack: 970, Len: 48
> SSH Protocol
    - SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
        Packet Length: 44
        Padding Length: 6
        > Key Exchange (method:curve25519-sha256)
            Padding String: 0df56771b217
            [Sequence number: 1]
            [Direction: client-to-server]

● Text Item (text), 48 bytes
    Packets: 25466 · Displayed:
```



Implementación y Características

```

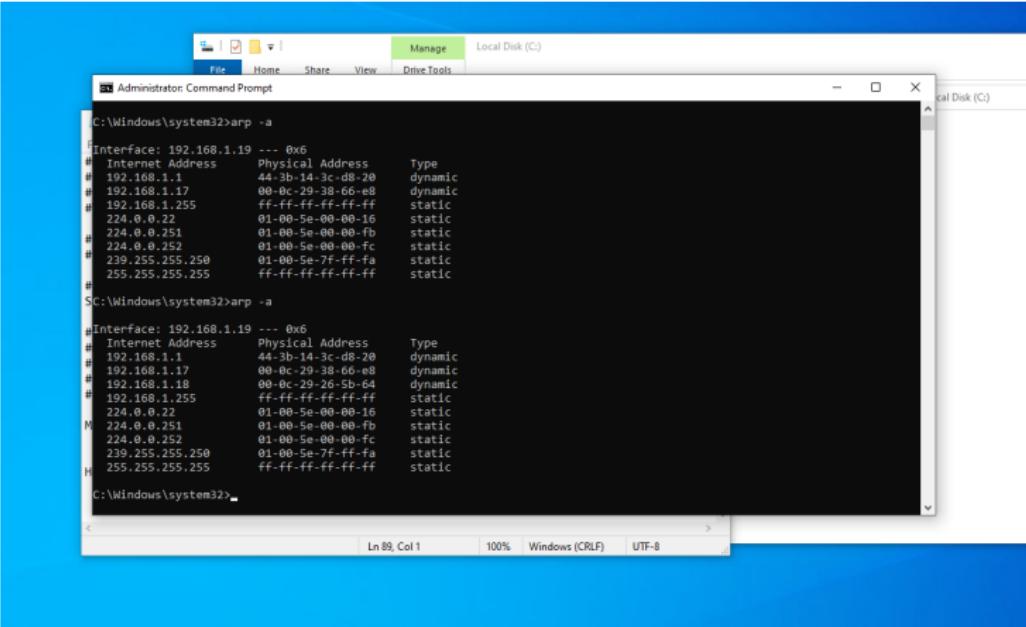
[...]
25226 466.518749433 192.168.1.19      192.168.1.17      SSHv2      162 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
25227 466 522229453 192.168.1.19      192.168.1.17      SSHv2      564 Client: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=64)
25228 466 538145425 192.168.1.19      192.168.1.17      SSHv2      184 Client: New Keys, Encrypted packet (len=64)
25229 466 538145425 192.168.1.19      192.168.1.17      SSHv2      118 Server: Encrypted packet (len=64)
Frame 25227: 566 bytes on wire (4528 bits), 566 bytes captured (4528 bits) on interface eth0, adq
Ethernet II, Src: VMware_7e:b3:cf (00:0c:29:7e:b3:cf), Dst: VMware_7e:b3:cf (00:0c:29:7e:b3:cf)
Internet Protocol Version 4, Src: 192.168.1.19, Dst: 192.168.1.19
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    + Differentiated Services Field: 0x00 (DSQoS: CS6, ECN: Not-ECT)
        Total Length: 566
        Identification: 0xdaf (17583)
    > 0100 .... = Flags: 0x2, Don't fragment
    ..0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x70ac [validation disabled]
        [Header checksum status: Unverified]
    Source Address: 192.168.1.19
    Destination Address: 192.168.1.19
    [Stream index: 50]
    Transmission Control Protocol, Src Port: 22, Dst Port: 49822, Seq: 976, Ack: 1789, Len: 512
    + SSH Protocol
        -> SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
            Packets: 188
            Padding Length: 188
            Padding String: 
            Key Exchange (method:curve25519-sha256)
            Padding String: 0000000000000000
            [Sequence number: 1]
        -> SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
            Packets: 192 bytes
[...]
0000 01 e9 77 1f 00 00 00 00 00 bc 00 5f 00 00 00 33  w   3
0000 00 00 00 0b 73 73 60 2d 65 64 32 35 35 31 39 00  ss - ed25519
0000 00 00 20 86 ea b1 99 61 52 64 ba 42 2b 61 00 e4  R_B:a
0000 81 1e 32 b6 a2 d9 be a2 4c 50 dd 86 ec 39 1a 9d  2   NP .9
0070 78 9e 44 00 00 00 20 ec bd 4f 1b f5 f4 95 eb 6b  x   0
0000 e2 5c 97 d0 a9 50 4f 13 80 74 97 c4 2a 78 ec  \P0 t .
0090 73 6d 2d 65 64 32 35 31 39 00 00 00 00 00 00 00 00 00
0000 73 6d 2d 65 64 32 35 31 39 00 00 00 00 00 00 00 00 00
0000 07 9c 3f 07 fe 5b df d6 29 19 56 a0 b1 b7 39 3b  g? z ) V .00
0000 51 51 87 0d 72 08 e5 e1 e2 29 0a cc 70 56 a3 f4  QQ r ) pV .
0000 2a 00 36 f3 7c 23 e2 a9 91 1d c6 e2 38 76 26 8e  * 6 |a .0v.
0000 e4 7d 2d 9c 57 0b e3 73 32 31 d3 58 d5 00 00 00  p-W s 23 X
0070 00 00 00 00 00 00 00 00 00 00 0c 0a 15 08 00 00 00 00
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0120 16 0c 0a 0a bd 13 58 56 0e 00 14 8c 37 a6 00 00 00 00
0130 04 51 ba 2f c3 09 0f 2a 77 b1 1f 20 fd f1 50 c6 0-Q .z .w .P.
0140 aa ac 24 c5 b7 5c 08 c8 30 92 3b 57 15 6c c5 b5 0-S .\ .6 |M .v .
0150 70 0f 09 d3 09 fd 82 c5 44 49 ae 34 5b 59 61 d3 }----- 01-EY .
0160 cb 5d 1f 51 fa 4b fe 09 9a a6 98 33 18 86 ee e8 }-0-K .A .
0170 d3 0d 37 93 05 9c 71 4a 69 3a 0f 5c 46 5e 59 36 }-7 .Q J :A [E
0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }----- 0-Y .B .P .
0190 4b cb 18 01 cc 35 0d 64 ea c5 43 7f 61 00 00 5 .C .Q .0
0180 7c 00 00 db cc bc 69 29 22 58 5e 30 0a 37 eb e9 }----- 1 "XZB .7 .
01b0 d9 32 59 91 ed 97 59 18 90 2e 0f 50 c9 f6 cf 4e 2ZY .Y .P .N
01c0 04 54 29 92 de 56 04 c5 43 d3 d8 30 44 0d c8 f5 }----- 2T .V .A .00
[...]

```

Packets: 25750 · Displayed: 69 (0.3%)

Text Item (text), 192 bytes

Implementación y Características



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". It displays two separate ARP table outputs from the command `arp -a`.

The first output (Interface: 192.168.1.19) shows the following entries:

Internet Address	Physical Address	Type
192.168.1.1	44-3b-14-3c-d8-20	dynamic
192.168.1.17	00-0c-29-38-66-e8	dynamic
192.168.1.255	ff-ff-ff-ff-ff-ff	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

The second output (Interface: 192.168.1.19) shows the same set of entries, indicating no change.

At the bottom of the window, there is a status bar with the text "Ln 89, Col 1" and a zoom level of "100%".

Figure: Suplantación de dirección MAC

Implementación y Características

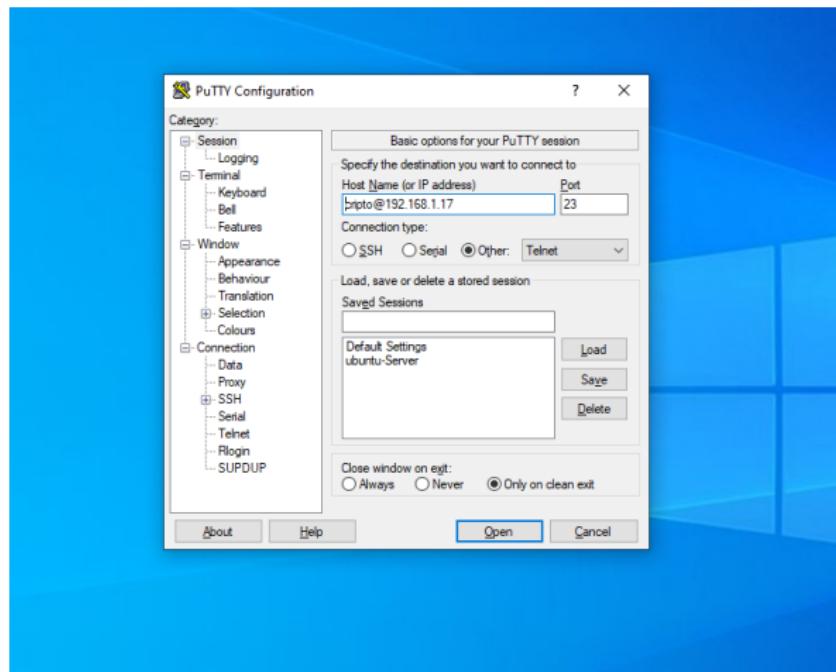
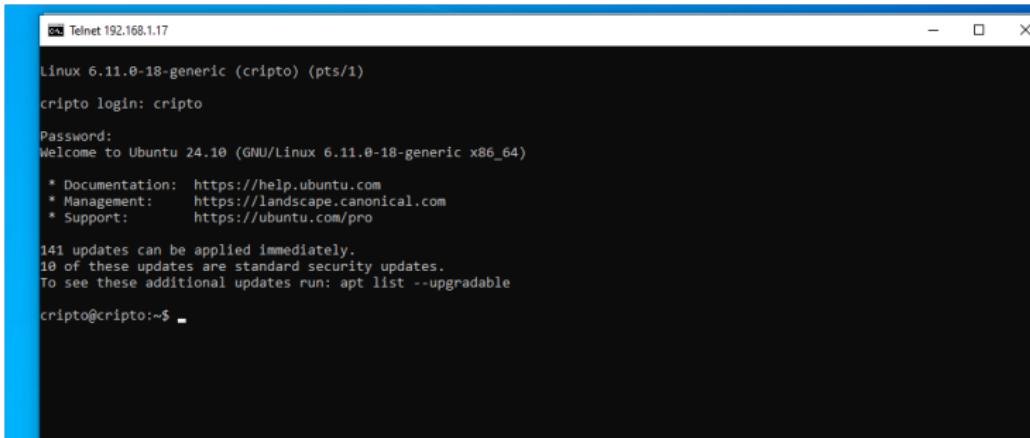


Figure: Conexión a través de Telnet

Implementación y Características



The screenshot shows a Windows Telnet window titled "Telnet 192.168.1.17". The session is connected to a Linux 6.11.0-18-generic (crypto) (pts/1) machine. A user named "cripto" has logged in. The password was accepted. The system is identified as "Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-18-generic x86_64)". It provides documentation at <https://help.ubuntu.com>, management at <https://landscape.canonical.com>, and support at <https://ubuntu.com/pro>. It indicates 141 updates are available, with 10 being standard security updates. To see additional updates, run "apt list --upgradable". The prompt shows the user is now at the root level ("cripto@cripto:~\$").

Figure: Ingreso a la maquina Ubuntu

Implementación y Características

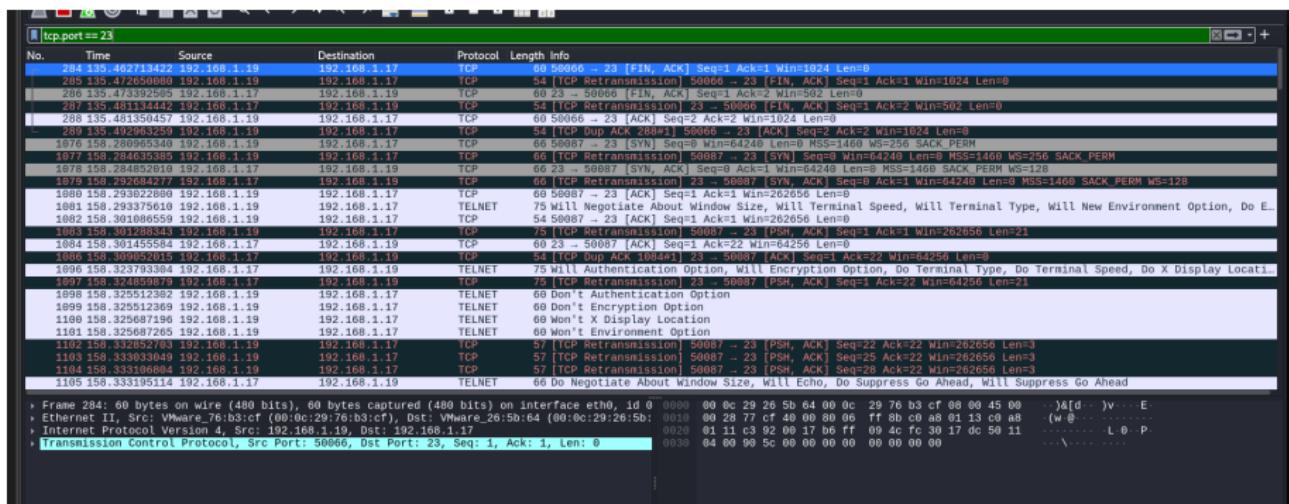


Figure: Captura de tráfico Telnet

Implementación y Características

No.	Time	Source	Destination	Protocol	Length	Info
1193	16:16.846074012	192.168.1.19	192.168.1.17	TELNET	66	[TCP] Dup ACK
1194	16:16.846074012	192.168.1.19	192.168.1.17	TCP	55	[TCP] Keep-Alive
1195	16:16.849207563	192.168.1.17	192.168.1.19	TCP	55	[TCP] Keep-Alive
1196	16:16.861317981	192.168.1.17	192.168.1.19	TCP	54	[TCP] Keep-Alive
1197	16:16.879491001	192.168.1.19	192.168.1.17	TELNET	66	1 byte data
1198	16:16.879491001	192.168.1.19	192.168.1.17	TCP	55	[TCP] Keep-Alive
1199	16:16.873474084	192.168.1.17	192.168.1.19	TCP	68	23 - 50087
1200	16:16.848875864	192.168.1.17	192.168.1.19	TCP	54	[TCP] Keep-Alive
1201	16:16.253348834	192.168.1.19	192.168.1.17	TELNET	66	1 byte data
1202	16:16.264762408	192.168.1.19	192.168.1.17	TCP	55	[TCP] Keep-Alive
1203	16:16.265935324	192.168.1.17	192.168.1.19	TCP	66	23 - 50087
1204	16:16.269999862	192.168.1.17	192.168.1.19	TCP	54	[TCP] Keep-Alive
1205	16:16.741638139	192.168.1.19	192.168.1.17	TELNET	66	1 byte data
1206	16:16.744749470	192.168.1.19	192.168.1.17	TCP	55	[TCP] Keep-Alive
1207	16:16.745015243	192.168.1.17	192.168.1.19	TCP	66	23 - 50087
1208	16:16.745393677	192.168.1.17	192.168.1.19	TELNET	66	2 bytes data
1209	16:16.748622570	192.168.1.17	192.168.1.19	TCP	54	53 - 50087
1210	16:16.748670658	192.168.1.17	192.168.1.19	TCP	56	[TCP] Retransmit
1211	16:16.779986842	192.168.1.19	192.168.1.17	TCP	68	50087 - 23
1212	16:16.804676499	192.168.1.19	192.168.1.17	TCP	54	[TCP] dup ACK
1213	16:16.804825085	192.168.1.17	192.168.1.19	TELNET	414	306 bytes data
1214	16:16.816697143	192.168.1.17	192.168.1.19	TCP	414	[TCP] Retransmit
1215	16:16.820000000	192.168.1.19	192.168.1.17	TCP	55	[TCP] Keep-Alive
1216	16:16.860963497	192.168.1.19	192.168.1.17	TCP	54	[TCP] dup ACK
1217	16:16.888897672	192.168.1.17	192.168.1.19	TELNET	108	46 bytes data

```
|> Frame 1205: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, j
|> Ethernet II, Src: VMware_76:b3:c9 (00:0c:29:76:b3:c9), Dst: VMWare_26:b6:64 (00:0c:29:26:f
|> Internet Protocol Version 4, Src: 192.168.1.19, Dst: 192.168.1.17
|> Transmission Control Protocol, Src Port: 50087, Dst Port: 23, Seq: 110, Ack: 126, Len: 1
|> Telnet
```

Wireshark - Follow TCP Stream (tcp.stream eq 22) - eth0

..38400,38400...T,USER,cripto,XTERM,

Linux 6.11.0-18-generic (cripto) (pts/1)

Password:
123

Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-18-generic x86_64)

* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/pro>

141 updates can be applied immediately.
10 of these updates are standard security updates.

Packet 7746.19 client pkts, 9 server pkts, 13 items. Click to select.

Entire conversation (643 bytes) Show as ASCII No delta times St
Find: Case sensitive

Filter Out This Stream Print Save as... Back × Close

Figure: Captura de información sensible 1

Implementación y Características

The screenshot shows a terminal window titled 'driftnet' running on a Kali Linux desktop environment. The terminal displays the following output:

```
Kali㉿kali: ~/Desktop
File Actions Edit View Help
driftnet

Summary:
Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 1332
Download size: 41.8 kB
Space needed: 116 kB / 5,249 MB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 driftnet amd64 1.5.0-0.2+b1 [41.8 kB]
Fetched 41.8 kB in 1s (77.6 kB/s)
Selecting previously unselected package driftnet.
(Reading database ... 404376 files and directories currently ...
Preparing to unpack .../driftnet_1.5.0-0.2+b1_amd64.deb ...
Unpacking driftnet (1.5.0-0.2+b1) ...
Setting up driftnet (1.5.0-0.2+b1) ...
Processing triggers for desktop-file-utils (0.27-2) ...
Processing triggers for man-db (2.13.0-1) ...
Processing triggers for kali-menu (2024.4.0) ...

[(kali㉿kali)-~/Desktop]
$ driftnet -i eth0
Tue Feb 25 20:27:42 2025 - error: pcap_activate: You don't have permission to perform this capture on that device

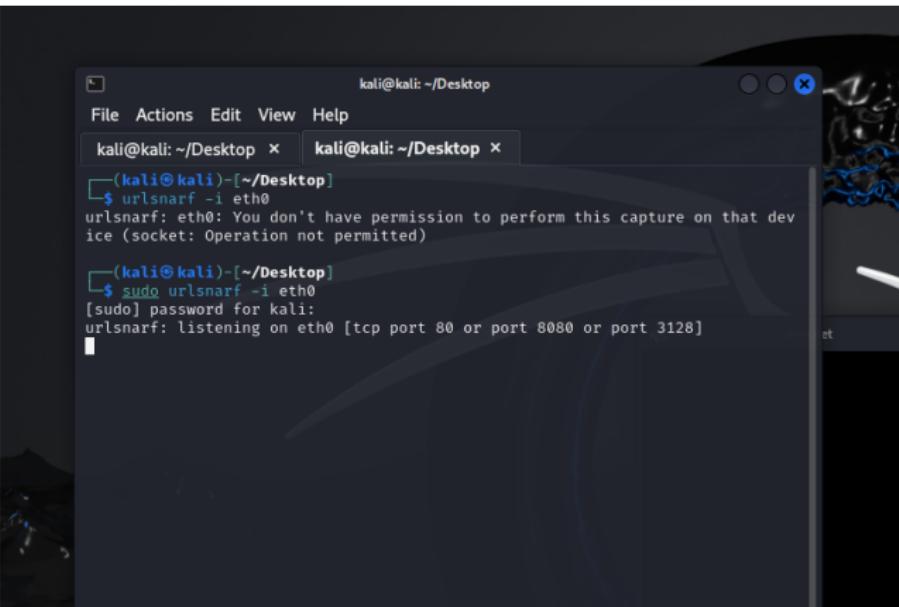
[(kali㉿kali)-~/Desktop]
$ sudo driftnet -i eth0
[]
```

The terminal shows the installation of the 'driftnet' package from the Kali Rolling repository. It then attempts to run the program with root privileges using 'sudo'. The output indicates that the user lacks the necessary permissions to capture traffic on the 'eth0' interface.

Figure: Herramientas de captura de información Driftnet

Implementación y Características

Ejemplo de diapositiva 18



The screenshot shows a terminal window titled "kali@kali: ~/Desktop". It contains two tabs, both labeled "kali@kali: ~/Desktop". The bottom tab is active and displays the following command and its output:

```
└─(kali㉿kali)-[~/Desktop]
$ urlsnarf -i eth0
urlsnarf: eth0: You don't have permission to perform this capture on that dev
ice (socket: Operation not permitted)

└─(kali㉿kali)-[~/Desktop]
$ sudo urlsnarf -i eth0
[sudo] password for kali:
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
```

Figure: Herramientas de captura de información UrlSnarf

Implementación y Características

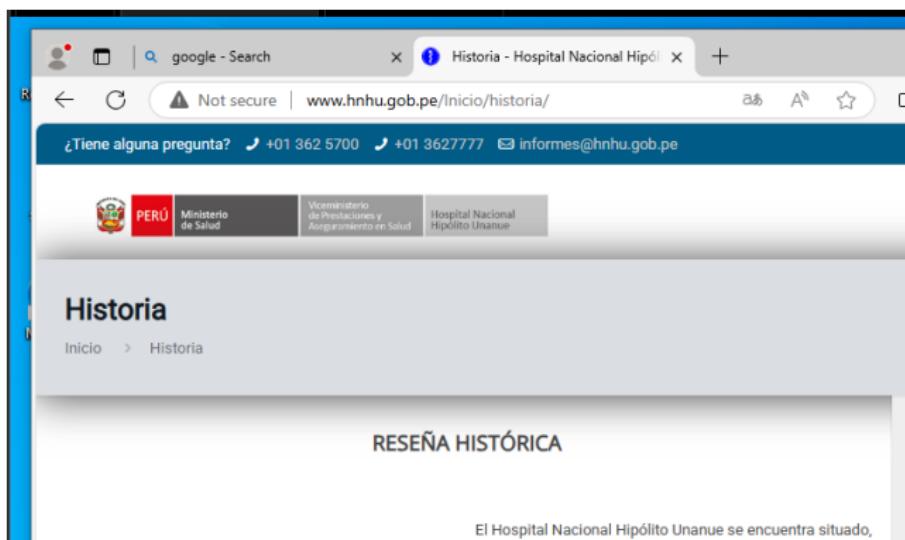
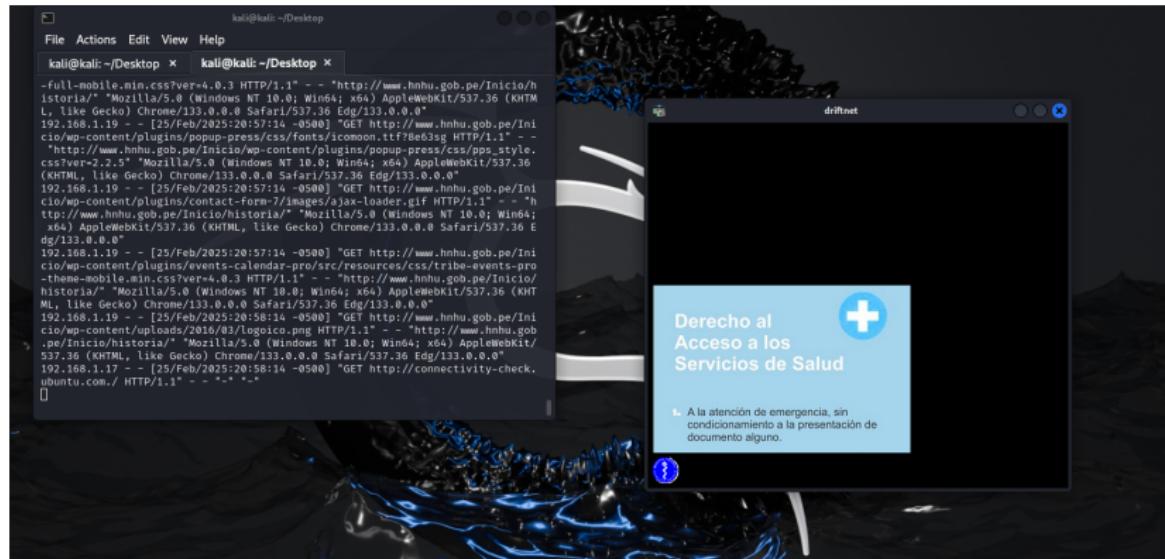


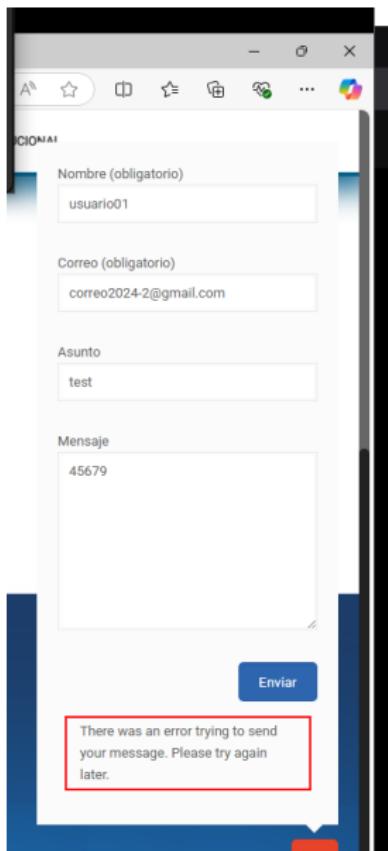
Figure: Ejemplo ingreso web insegura

Implementación y Características

Ejemplo de diapositiva 20



Implementación y Características



Implementación y Características

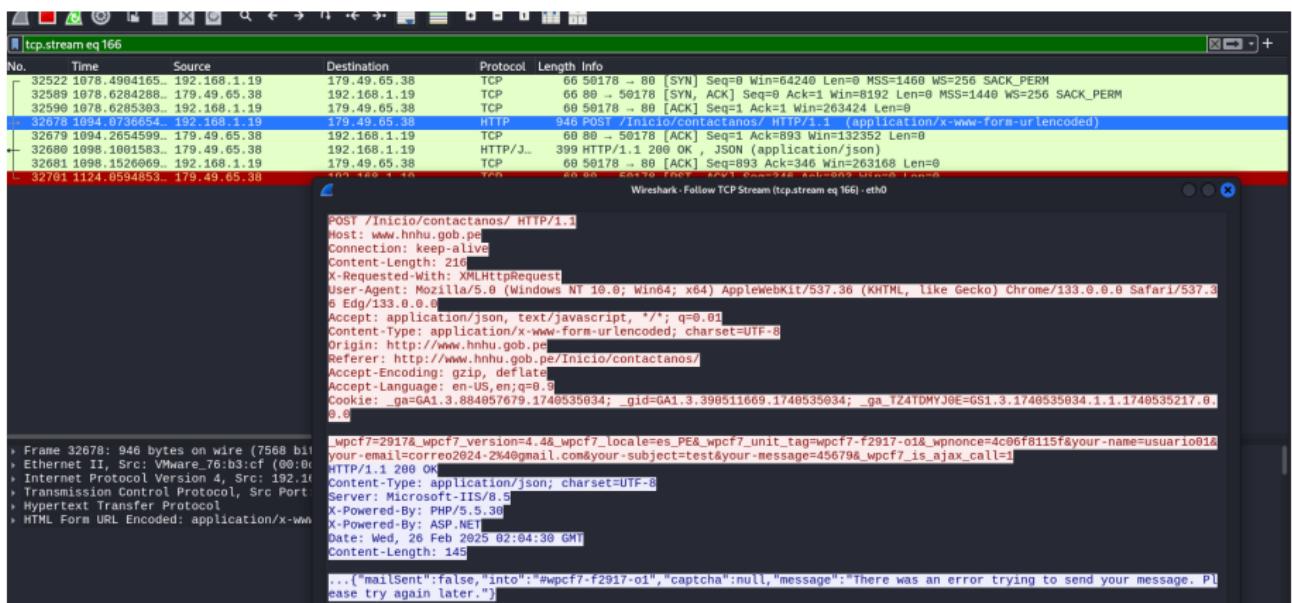


Figure: Captura de información Sensible 2

¿Qué es Kyber?

Kyber

Kyber es un esquema de cifrado basado en retículos (lattice-based cryptography) diseñado para ser resistente a ataques cuánticos. Fue seleccionado por el NIST como uno de los finalistas para la estandarización de criptografía poscuántica.

- **Tipo:** Mecanismo de encapsulamiento de llave (KEM).
- **Base Matemática:** Problemas de retículos.
- **Seguridad:** Resistente a ataques cuánticos, como el algoritmo de Shor.
 - **Kyber-512:** Nivel de seguridad semejante a **AES-128**.
 - **Kyber-768:** Nivel de seguridad semejante a **AES-192**.
 - **Kyber-1024:** Nivel de seguridad semejante a **AES-256**.

Funcionamiento de Kyber

Funcionamiento de Kyber

Kyber opera en tres fases principales:

- ① Generación de Claves:** Se generan una clave pública y una clave privada.
- ② Encapsulamiento:** Se encapsula una clave simétrica usando la clave pública.
- ③ Desencapsulamiento:** Se recupera la clave simétrica usando la clave privada.

Kyber utiliza operaciones matriciales y polinomiales sobre un retículo para garantizar la seguridad.

Matemáticas detrás de Kyber

Kyber se basa en problemas de retículos, específicamente en el problema de aprendizaje con errores (Learning With Errors, LWE).

- **Retículos:** Estructuras algebraicas en espacios vectoriales.
- **LWE:** Dado un conjunto de ecuaciones lineales con ruido, es difícil recuperar la solución original.
- **Operaciones:** Kyber utiliza multiplicación de matrices y polinomios en un anillo $R_q = \mathbb{Z}_q[X]/(X^n + 1)$.

VARIABLES EN EL ESKUEMA KYBER

- **Dimensión del Retículo (n):** Define el tamaño del retículo y afecta la seguridad del esquema. Valores comunes son $n = 256, 512$.
- **Módulo (q):** Un número primo que define el anillo $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. Kyber usa $q = 3329$.
- **Nivel de Seguridad (k):** Determina el tamaño de las matrices y vectores en el esquema. Valores típicos son $k = 2, 3, 4$ para diferentes niveles de seguridad.
- **Distribución de Ruido (χ):** Una distribución de probabilidad usada para añadir ruido a las ecuaciones. Kyber usa una distribución centrada binomial.
- **Parámetro de Compresión (d):** Controla la compresión de los coeficientes para reducir el tamaño de las claves y cifrados.

Intercambio de Claves: Generación de Claves (Alice)

Paso 1: Alice genera su par de claves.

- **Entrada:** Parámetros públicos (q, n, A) , donde:
 - $q = 3329$ (módulo).
 - $n = 256$ (dimensión del retículo).
 - A : Matriz pública de tamaño $k \times k$ en R_q .
- **Generación de claves:**
 - Alice elige un vector secreto s y un vector de error e , ambos con coeficientes pequeños (ruido).
 - Calcula:

$$t = A \cdot s + e$$

- La **clave pública** es $pk = (A, t)$.
- La **clave privada** es $sk = s$.

Intercambio de Claves: Encapsulamiento (Bob)

Paso 2: Bob encapsula una clave simétrica.

- **Entrada:** Clave pública de Alice $pk = (A, t)$.
- **Generación de valores aleatorios:**
 - Bob elige un vector secreto r y vectores de error e_1, e_2 , con coeficientes pequeños.
- **Cálculo del cifrado:**
 - Bob calcula:

$$u = A^T \cdot r + e_1$$

$$v = t^T \cdot r + e_2 + (m)$$

Donde m es el mensaje (clave simétrica) que Bob quiere compartir.

- Comprime u y v :

$$u_{\text{compressed}} = \text{Compress}(u, d)$$

$$v_{\text{compressed}} = \text{Compress}(v, d)$$

- **Envío del cifrado:**

- Bob envía $c = (u_{\text{compressed}}, v_{\text{compressed}})$ a Alice.

Intercambio de Claves: Desencapsulamiento (Alice)

Paso 3: Alice recupera la clave simétrica.

- **Entrada:** Cifrado $c = (u_{\text{compressed}}, v_{\text{compressed}})$ y clave privada $sk = s$.
- **Descompresión:**
 - Alice descomprime u y v :

$$u = \text{Decompress}(u_{\text{compressed}}, d)$$

$$v = \text{Decompress}(v_{\text{compressed}}, d)$$

- **Recuperación del mensaje:**

- Alice calcula:

$$m = (v - s^T \cdot u)$$

- Esto le permite recuperar la clave simétrica m .

Matemáticas detrás de Kyber

$$\begin{aligned}m_n &= v - s^T u \\&= t^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b - s^T (A^T r + e_1) \\&= (As + e)^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b - s^T (A^T r + e_1) \\&= e^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b - s^T e_1\end{aligned}$$

Implementación de Kyber en SSH Poscuántico en IBM Cloud

Recurso recomendado:

- Para ver una implementación detallada de Kyber en SSH poscuántico en IBM Cloud:

IBM Quantum-Safe SSH Tutorial

Herramientas utilizadas:

- OpenSSH OQS:** Una bifurcación de OpenSSH que incluye soporte para algoritmos cuánticamente seguros.
- liboqs:** Una biblioteca de código abierto que implementa algoritmos poscuánticos, incluido Kyber.

Referencias

- **A Security Site - ECDH**
- **Wikipedia - Curve25519**
- **IBM - Conexión SSH Post-Cuántica**
- **Demystifying Elliptic Curves for Cryptography**