

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

February 26, 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

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

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

4 / 45

# 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.

## SSH es fundamental para:

- 1 Gestionar sistemas y aplicaciones de forma remota
- 2 Iniciar sesión en ordenadores remotos con seguridad
- 3 Ejecutar comandos en sistemas distantes
- 4 Transferir archivos con cifrado completo

# Intercambio de Claves Diffie-Hellman de Curva Elíptica

ECDH ofrece ventajas significativas:

- ➊ Mayor seguridad con claves más pequeñas
- ➋ Operaciones criptográficas más eficientes
- ➌ Menor consumo de recursos computacionales
- ➍ Ampliamente implementado en OpenSSH moderno

# 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)



# 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:

- ❶ Integración híbrida con ECDH para mayor seguridad
- ❷ Disponible en versiones recientes (OpenSSH 9.0+)
- ❸ Configuración mediante directivas específicas
- ❹ Compatibilidad con infraestructuras existentes

# Algoritmo Kyber

Kyber es un finalista del NIST para criptografía poscuántica:

- ➊ Basado en problemas de retículos
- ➋ Resistente a ataques cuánticos conocidos
- ➌ Ofrece buen equilibrio entre seguridad y eficiencia
- ➍ Seleccionado para estandarización por el NIST

# Implementación de MITM

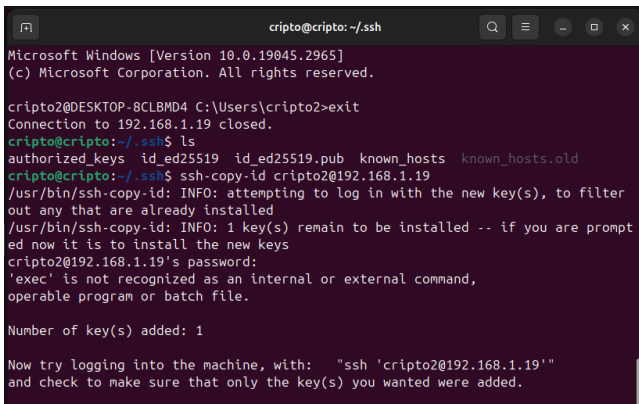
```
cripto@cripto:~/Desktop$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/cripto/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cripto/.ssh/id_ed25519
Your public key has been saved in /home/cripto/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:b3PZthGSyejKp4xiFdgjHIVino0xxy4LH1HQ0krLt+U cripto@cripto
The key's randomart image is:
+--[ED25519 256]--+
|  .o                |
|    +               |
|   = =             |
|..= = +   o o      |
|+o.o . oS . = .    |
|++o. .. o  + .     |
|oBo.+      = o +   |
|+++o.E + o.o . o   |
| +o. .. =o        . |
+-----[SHA256]-----+
cripto@cripto:~/Desktop$
```

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





## 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 cripto2@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

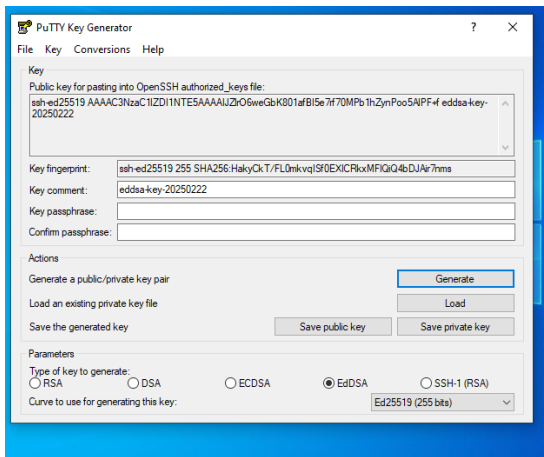


Figure: Generación llave ECC con Putty

## Implementación y Características

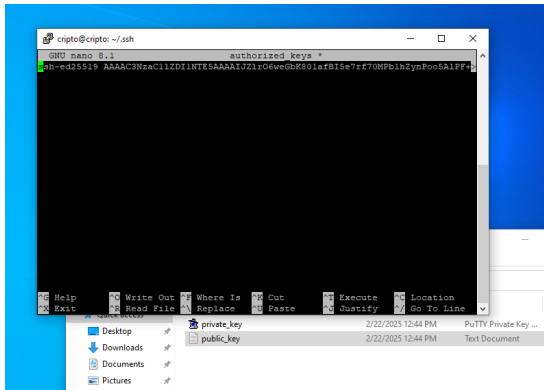


Figure: Visualización llave

## Implementación y Características

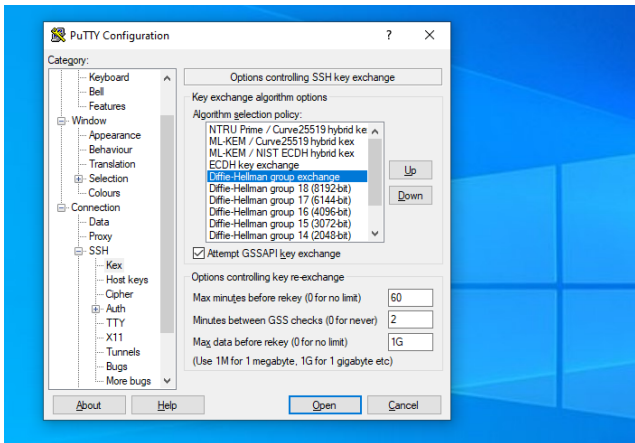
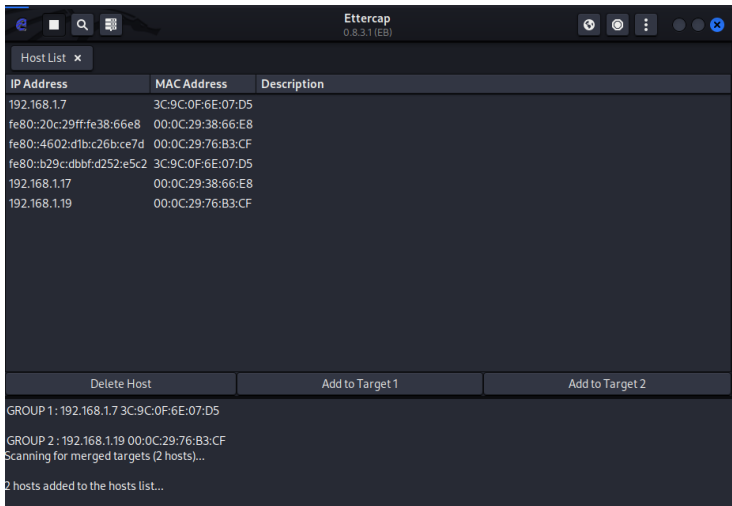


Figure: Intercambio de llaves Diffie-Hellman

# Identificando los Hosts de ataque



## Implementación y Características

No.	Time	Source	Destination	Protocol	Length	Info
17519	322.859454568	192.168.1.19	192.168.1.17	TCP	60	49774 -> 22 [FIN, ACK] Seq=1 Ack=1 Win=1826 Len=0
17520	322.863836208	192.168.1.17	192.168.1.19	TCP	60	22 -> 49774 [FIN, ACK] Seq=1 Ack=2 Win=485 Len=0
17521	322.864643682	192.168.1.19	192.168.1.17	TCP	60	49774 -> 22 [ACK] Seq=2 Ack=2 Win=1826 Len=0
18608	371.521868896	192.168.1.19	192.168.1.17	TCP	60	49785 -> 22 [SYN] Seq=8 Win=64248 Len=0 MSS=1468 WS=256 SACK_PERM
18609	371.521188577	192.168.1.17	192.168.1.19	TCP	60	22 -> 49785 [SYN, ACK] Seq=8 Ack=1 Win=64248 Len=0 MSS=1468 SACK_PERM WS=128
18602	371.521294286	192.168.1.19	192.168.1.17	TCP	60	49785 -> 22 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
18603	371.527064584	192.168.1.19	192.168.1.17	SSHv2	82	Client: Protocol (SSH-2.0-PuTTY_Release_0.83)
18604	371.527162393	192.168.1.17	192.168.1.19	TCP	60	22 -> 49785 [ACK] Seq=1 Ack=29 Win=64256 Len=0
18605	371.529677114	192.168.1.17	192.168.1.19	SSHv2	95	Server: Protocol (SSH-2.0-OpenSSH_9.7p1 Ubuntu-7ubuntu4.2)
18606	371.536614512	192.168.1.19	192.168.1.17	TCP	1514	49785 -> 22 [ACK] Seq=29 Ack=42 Win=2102272 Len=1468 [TCP PDU reassembled in 18607]
18607	371.536614757	192.168.1.19	192.168.1.17	SSHv2	386	Client: Key Exchange Init
18608	371.536804330	192.168.1.17	192.168.1.19	SSHv2	982	Server: Key Exchange Init
18609	371.538927188	192.168.1.19	192.168.1.17	SSHv2	192	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
18610	371.543336874	192.168.1.17	192.168.1.19	SSHv2	566	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=304)
18611	371.551430003	192.168.1.19	192.168.1.17	SSHv2	134	Client: New Keys, Encrypted packet (len=64)
18612	371.551883661	192.168.1.17	192.168.1.19	SSHv2	118	Server: Encrypted packet (len=64)
18613	371.555455917	192.168.1.19	192.168.1.17	SSHv2	134	Client: Encrypted packet (len=80)
18614	371.556088210	192.168.1.17	192.168.1.19	SSHv2	134	Server: Encrypted packet (len=80)
18615	371.556913551	192.168.1.19	192.168.1.17	SSHv2	214	Client: Encrypted packet (len=160)
18616	371.560674332	192.168.1.17	192.168.1.19	SSHv2	166	Server: Encrypted packet (len=112)
18617	371.563513184	192.168.1.19	192.168.1.17	SSHv2	310	Client: Encrypted packet (len=256)
18618	371.576551098	192.168.1.17	192.168.1.19	SSHv2	162	Server: Encrypted packet (len=48)
18619	371.577389863	192.168.1.19	192.168.1.17	SSHv2	134	Client: Encrypted packet (len=80)
18620	371.618592965	192.168.1.17	192.168.1.19	TCP	60	22 -> 49785 [ACK] Seq=1786 Ack=2445 Win=62288 Len=0
18621	371.634318644	192.168.1.17	192.168.1.19	SSHv2	870	Server: Encrypted packet (len=816)
18622	371.682768371	192.168.1.19	192.168.1.17	TCP	60	49785 -> 22 [ACK] Seq=2445 Ack=2662 Win=2181248 Len=0

Figure: Captura de la conexión con Wireshark

## Implementación y Características

tcp.port == 22

No.	Time	Source	Destination	Protocol	Length	Info
25217	466.593095738	192.168.1.17	192.168.1.19	TCP	66	22 → 49822 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
25218	466.593193682	192.168.1.19	192.168.1.17	TCP	60	49822 → 22 [ACK] Seq=1 Ack=1 Win=2182272 Len=0
25219	466.512346382	192.168.1.19	192.168.1.17	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	60	22 → 49822 [ACK] Seq=42 Ack=29 Win=64256 Len=0
25222	466.514163956	192.168.1.19	192.168.1.19	SSHv2	982	Server: Key Exchange Init
25223	466.515469656	192.168.1.17	192.168.1.19	TCP	1514	49822 → 22 [ACK] Seq=29 Ack=970 Win=2181248 Len=1460 [TCP PDU reassembled in 25224]
25224	466.515469718	192.168.1.19	192.168.1.17	SSHv2	386	Client: Key Exchange Init
25225	466.515598247	192.168.1.17	192.168.1.19	TCP	60	22 → 49822 [ACK] Seq=970 Ack=1741 Win=62592 Len=0
25226	466.518749433	192.168.1.19	192.168.1.17	SSHv2	182	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
25227	466.522229652	192.168.1.17	192.168.1.19	SSHv2	566	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet
25228	466.530183420	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: 182 bytes on wire (816 bits), 182 bytes captured (816 bits) on interface eth0, id 0						
» Ethernet II, Src: VMware_76:b3:cf (00:0c:29:76:b3:cf), Dst: VMware_38:66: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]						
0000 00 0c 29 38 66 e8 00 0c 29 76 b3 cf 08 00 45						
0010 00 58 fd c0 40 00 80 06 79 6a c0 a8 01 13 c0						
0020 01 11 c2 9e 00 16 fd ee 95 1b 63 cc 31 d7 50						
0030 20 10 1f 49 00 00 00 00 00 2c 06 1e 00 00 00						
0040 9a 5d 06 73 7d f4 47 3d d3 89 e0 79 b8 ff 1b						
0050 6d 60 03 e4 a6 be c8 d8 f1 b1 a3 63 56 0b 85						
0060 0d f5 67 71 b2 17						
Text item (text), 48 bytes						
Packets: 25466 · Displayed:						

## Implementación y Características

```

25226 466.918749433 192.168.1.19 192.168.1.17 SSHv2 192 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
25227 466.922296522 192.168.1.17 192.168.1.19 SSHv2 569 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New keys, Encrypted packet (len=304)
25228 466.938183420 192.168.1.19 192.168.1.17 SSHv2 134 Client: New Keys, Encrypted packet (len=64)
25229 466.938389192 192.168.1.19 192.168.1.17 SSHv2 198 Server: Encrypted packet (len=64)

Frame 25227: 566 bytes on wire (4528 bits), 566 bytes captured (4528 bits) on interface eth0, id 0
Ethernet II, Src: VMware_38:66:08 (00:0c:29:38:66:08), Dst: VMware_76:b3:c0f (00:0c:29:76:b3:c0f)
Internet Protocol Version 4, Src: 192.168.1.17, Dst: 192.168.1.19
0100 ... = Version 4
... 0101 = Header Length: 20 bytes (5)
... Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 552
Identification: 0x4d4f (17593)
... 010 ... = Flags: Bx2, 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: 192.168.1.17
Destination Address: 192.168.1.19
[Stream index: 50]
Transmission Control Protocol, Src Port: 22, Dst Port: 49822, Seq: 978, Ack: 1789, Len: 512
[SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)]
Packet Length: 188
Padding Length: 8
Key Exchange (method:curve25519-sha256)
Padding String: 0000000000000000
[Sequence number: 0]
SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)

```

## Implementación y Características

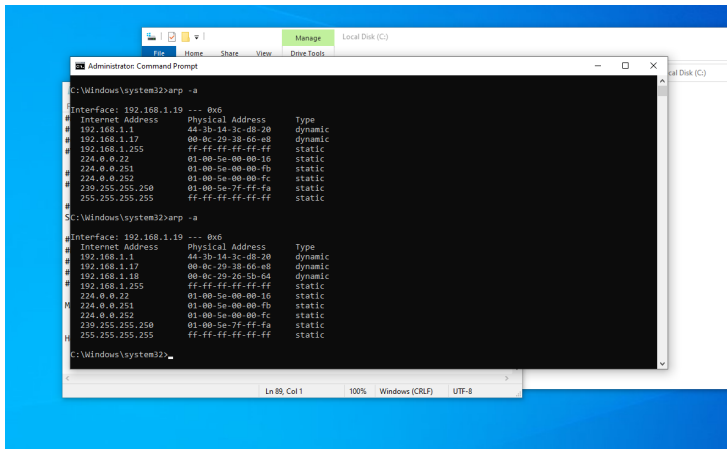


Figure: Suplantación de dirección MAC



## Implementación y Características

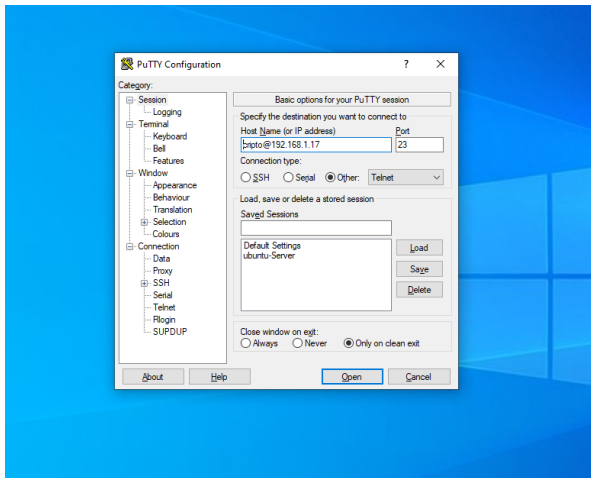


Figure: Conexión a través de Telnet

## Implementación y Características

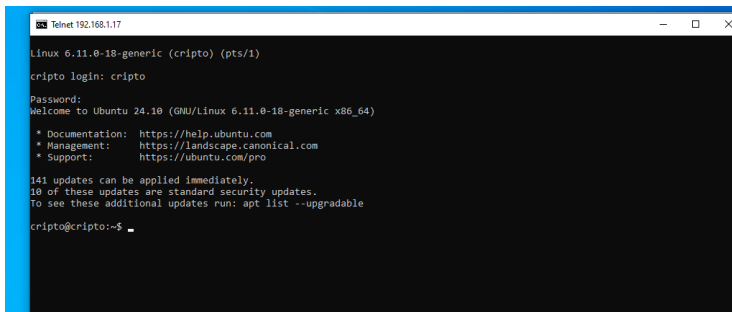
A terminal window titled 'Telnet 192.168.1.17' with standard window controls. The terminal output shows a successful SSH login to a Linux 6.11.0-18-generic machine named 'cripto'. The user 'cripto' provides their login name, and the system prompts for a password. After login, the system displays the Ubuntu 24.10 welcome message, including links for documentation, management, and support. It also informs the user that 141 updates can be applied immediately, with 10 being standard security updates. The prompt returns to 'cripto@cripto:~\$'.

Figure: Ingreso a la maquina Ubuntu

## Implementación y Características

tcp.port == 23						
No.	Time	Source	Destination	Protocol	Length	Info
284	135.462713422	192.168.1.19	192.168.1.17	TCP	60	50066 → 23 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
285	135.472600089	192.168.1.19	192.168.1.17	TCP	54	[TCP Retransmission] 50066 → 23 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
286	135.472600505	192.168.1.17	192.168.1.19	TCP	60	23 → 50066 [FIN, ACK] Seq=1 Ack=2 Win=502 Len=0
287	135.481134442	192.168.1.17	192.168.1.19	TCP	54	[TCP Retransmission] 23 → 50066 [FIN, ACK] Seq=1 Ack=2 Win=502 Len=0
288	135.481350457	192.168.1.19	192.168.1.17	TCP	60	50066 → 23 [ACK] Seq=2 Ack=2 Win=1024 Len=0
289	135.492082550	192.168.1.19	192.168.1.17	TCP	54	[TCP Dup ACK] 50066 → 23 [ACK] Seq=2 Ack=2 Win=1024 Len=0
1676	158.280965340	192.168.1.19	192.168.1.17	TCP	60	50087 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1677	158.284635385	192.168.1.19	192.168.1.17	TCP	66	[TCP Retransmission] 50087 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1678	158.284852010	192.168.1.17	192.168.1.19	TCP	66	23 → 50087 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
1679	158.292042777	192.168.1.19	192.168.1.19	TCP	66	[TCP Retransmission] 23 → 50087 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
1680	158.293022080	192.168.1.19	192.168.1.17	TCP	60	50087 → 23 [ACK] Seq=1 Ack=1 Win=262656 Len=0
1681	158.293375610	192.168.1.19	192.168.1.17	TELNET	75	Will Negotiate About Window Size, Will Terminal Speed, Will Terminal Type, Will New Environment Option, Do E..
1682	158.301006559	192.168.1.19	192.168.1.17	TCP	54	50087 → 23 [ACK] Seq=1 Ack=1 Win=262656 Len=0
1683	158.301268344	192.168.1.19	192.168.1.17	TCP	75	[TCP Retransmission] 50087 → 23 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=21
1684	158.301455584	192.168.1.17	192.168.1.19	TCP	60	23 → 50087 [ACK] Seq=1 Ack=22 Win=64256 Len=0
1686	158.309052015	192.168.1.17	192.168.1.19	TCP	54	[TCP Dup ACK] 1684 → 23 → 50087 [ACK] Seq=1 Ack=22 Win=64256 Len=0
1696	158.323793384	192.168.1.17	192.168.1.19	TELNET	75	Will Authentication Option, Will Encryption Option, Do Terminal Type, Do Terminal Speed, Do X Display Locati..
1697	158.324859679	192.168.1.17	192.168.1.19	TCP	75	[TCP Retransmission] 23 → 50087 [PSH, ACK] Seq=1 Ack=22 Win=64256 Len=21
1698	158.325512362	192.168.1.19	192.168.1.17	TELNET	60	Don't Authentication Option
1699	158.325512369	192.168.1.19	192.168.1.17	TELNET	60	Don't Encryption Option
1700	158.325687196	192.168.1.19	192.168.1.17	TELNET	60	Don't X Display Location
1701	158.325687265	192.168.1.19	192.168.1.17	TELNET	60	Don't Environment Option
1702	158.332852783	192.168.1.19	192.168.1.17	TCP	57	[TCP Retransmission] 50087 → 23 [PSH, ACK] Seq=22 Ack=22 Win=262656 Len=3
1703	158.333033049	192.168.1.19	192.168.1.17	TCP	57	[TCP Retransmission] 50087 → 23 [PSH, ACK] Seq=25 Ack=22 Win=262656 Len=3
1704	158.333106804	192.168.1.19	192.168.1.17	TCP	57	[TCP Retransmission] 50087 → 23 [PSH, ACK] Seq=28 Ack=22 Win=262656 Len=3
1705	158.333195114	192.168.1.17	192.168.1.19	TELNET	66	Do Negotiate About Window Size, Will Echo, Do Suppress go Ahead, Will Suppress go Ahead

Frame 284: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0 0000 00 0c 29 26 5b 64 00 0c 29 76 b3 cf 00 00 45 00 ... )S(d...v...E  
Ethernet II, Src: VMware\_76:b3:cf (00:0c:29:76:b3:cf), Dst: VMware\_26:5b:64 (00:0c:29:26:5b:64) ... (w...  
Internet Protocol Version 4, Src: 192.168.1.19, Dst: 192.168.1.17 0020 01 11 c3 92 00 17 b6 ff 09 4c fc 30 17 dc 50 11 ..... L.O.P.  
Transmission Control Protocol, Src Port: 50066, Dst Port: 23, Seq: 1, Ack: 1, Len: 0 0030 04 00 00 5c 00 00 00 00 00 00 00

Figure: Captura de tráfico Telnet

## Implementación y Características

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

No.	Time	Source	Destination	Protocol	Length	Info
1154	158.460891784	192.168.1.19	192.168.1.17	TCP	54	TCP Dup ACK
1193	160.846074012	192.168.1.19	192.168.1.17	TELNET	60	1 byte data
1194	160.849011951	192.168.1.19	192.168.1.17	TCP	55	[TCP Keep-Alive]
1195	160.849297653	192.168.1.17	192.168.1.19	TCP	60	23 - 50087 [ACK]
1196	160.861317901	192.168.1.17	192.168.1.19	TCP	54	[TCP Keep-Alive]
1197	161.029491001	192.168.1.19	192.168.1.17	TELNET	60	1 byte data
1198	161.037102549	192.168.1.19	192.168.1.17	TCP	55	[TCP Keep-Alive]
1199	161.037470404	192.168.1.17	192.168.1.19	TCP	60	23 - 50087 [ACK]
1200	161.040875884	192.168.1.17	192.168.1.19	TCP	54	[TCP Keep-Alive]
1201	161.253348834	192.168.1.19	192.168.1.17	TELNET	60	1 byte data
1202	161.264702408	192.168.1.19	192.168.1.17	TCP	55	[TCP Keep-Alive]
1203	161.265035324	192.168.1.17	192.168.1.19	TCP	60	23 - 50087 [ACK]
1204	161.269099862	192.168.1.17	192.168.1.19	TCP	54	[TCP Keep-Alive]
1205	161.741638139	192.168.1.19	192.168.1.17	TELNET	60	1 byte data
1206	161.744749470	192.168.1.19	192.168.1.17	TCP	55	[TCP Keep-Alive]
1207	161.745015243	192.168.1.17	192.168.1.19	TCP	60	23 - 50087 [ACK]
1208	161.745303677	192.168.1.17	192.168.1.19	TELNET	60	2 bytes data
1209	161.748622570	192.168.1.19	192.168.1.17	TCP	54	23 - 50087 [ACK]
1210	161.748700587	192.168.1.17	192.168.1.19	TCP	56	[TCP Retransmission]
1211	161.799868642	192.168.1.19	192.168.1.17	TCP	60	50087 -> 23 [ACK]
1212	161.004076499	192.168.1.19	192.168.1.17	TCP	54	[TCP Dup ACK]
1213	161.804025085	192.168.1.17	192.168.1.19	TELNET	414	308 bytes data
1214	161.812500723	192.168.1.19	192.168.1.17	TCP	414	[TCP Corruption]
1215	161.861749464	192.168.1.19	192.168.1.17	TCP	60	50087 -> 23 [ACK]
1216	161.868023482	192.168.1.19	192.168.1.17	TCP	54	[TCP Dup ACK]
1217	161.868097677	192.168.1.17	192.168.1.19	TELNET	100	46 bytes data

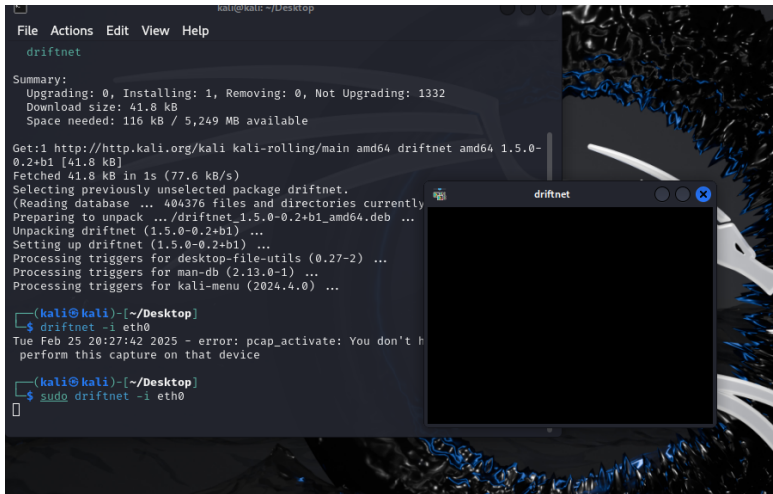
Frame 1205: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, ...  
Ethernet II, Src: VMware\_7e:b3:cf (00:0c:29:7e:b3:cf), Dst: VMware\_26:5b:64 (00:0c:29:26:5b:64)  
Internet Protocol Version 4, Src: 192.168.1.19, Dst: 192.168.1.17  
Transmission Control Protocol, Src Port: 50087, Dst Port: 23, Seq: 110, Ack: 126, Len: 1  
Telnet

Linux 6.11.0-18-generic (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 1746: 10 bytes on wire (80 bits), 10 bytes captured (80 bits) on interface eth0, ...

Entire conversation (643 bytes) Show as ASCII No delta times Case sensitive  
Find:  
Filter Out This Stream Print Save as... Back Close

Figure: Captura de información sensible 1

## Implementación y Características



The screenshot shows a Kali Linux desktop environment with a dark theme and a background image of a dragon. A terminal window is open, displaying the installation of the 'driftnet' package. The terminal output shows the package being downloaded from the Kali rolling repository and installed. After installation, the user runs 'driftnet -i eth0', which results in an error: 'pcap\_activate: You don't have permission to perform this capture on that device'. The user then runs 'sudo driftnet -i eth0'. A smaller window titled 'driftnet' is also visible on the desktop, but it is empty.

```
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 installed.)
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
```

Figure: Herramientas de captura de información Driftnet

# Ejemplo de diapositiva 18

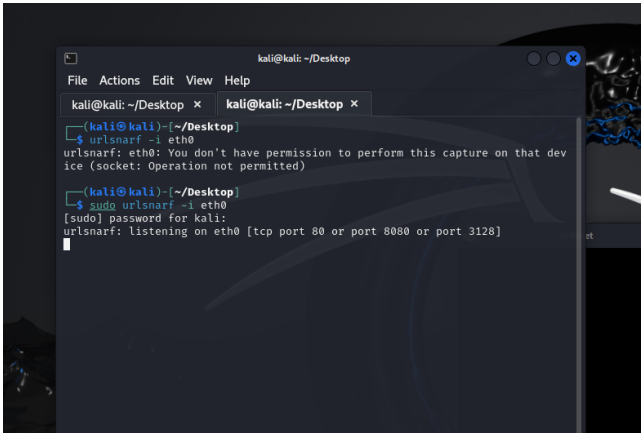


Figure: Herramientas de captura de información UrlSnarf

## Implementación y Características

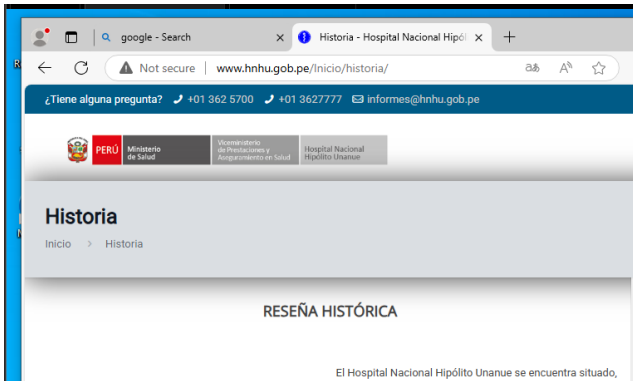


Figure: Ejemplo ingreso web insegura

## Ejemplo de diapositiva 20





## Implementación y Características

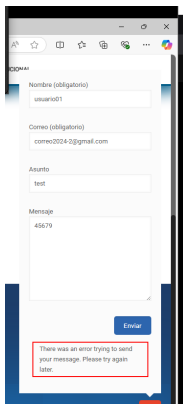


Figure: Test de trafico capturado

## Implementación y Características

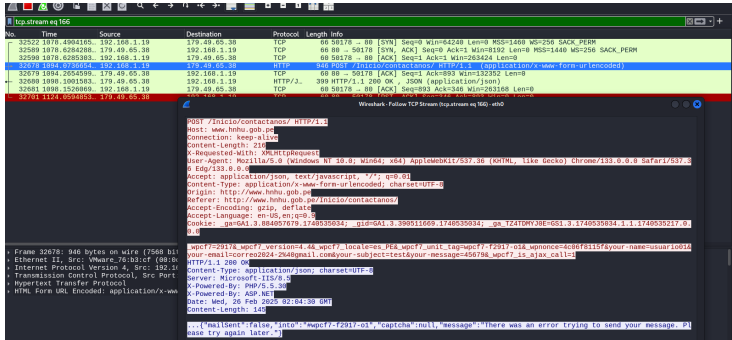


Figure: Captura de información Sensible 2

# 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 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  $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ .

- **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 ( $\chi$ ):** 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.



### 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:

$$v = t^T \cdot r + e_2 + \text{Encode}(m)$$

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.



### Paso 3: Alice recupera la clave simétrica.

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

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

- Alice calcula:

$$m = \text{Decode}(v - s^T \cdot u)$$

- ◀ ◻ ▶ ◀ ▢ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

# Intercambio de Claves: Generación de Claves (Alice)

## Paso 1: Alice genera su par de claves.

- **Entrada:** Parámetros públicos  $(q, n, A)$ :
  - $q = 3329$  (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  $sk = s$ .



### Paso 3: Alice recupera la clave simétrica.

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

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

- Alice calcula:

$$m = \text{Decode}(v - s^T \cdot u)$$

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

**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.

## Contenido del tutorial:

- Configuración de un servidor remoto en IBM Cloud.