Redes Privadas Virtuales (VPN)


HACKED
©Beyond Security® All rights reserved
www.SecuriTeam.com
BY DALE BRADEN
I'M NOT USING THIS CIRCUIT UNTIL I KNOW FOR SURE THAT OUR CONVERSATION WILL OCCUR OVER A SECURE CONNECTION!
BRADEN

Redes Privadas virtuales

Se pueden definir redes privadas virtuales utilizando protocolos de distinto nivel. Ejemplos más utilizados:

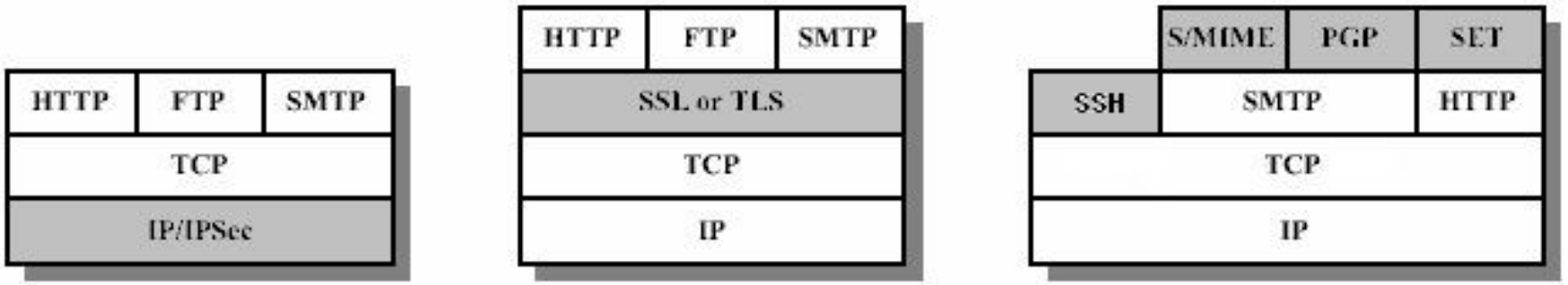
IPSEC, componente obligatorio en IPv6 , opcional para IPv4

VPN Basada en TLS. Se puede asegurar un servicio o toda la comunicación entre dos redes.

VPN Basada en SSH. Se pueden hacer túneles para un solo servicio, túneles dinámicos con socks o túneles completos en capa 2 y 3.

Otros

Protocolos


HTTP
FTP
SMTP
TCP
IP/IPSec
HTTP
FTP
SMTP
SSL or TLS
TCP
IP
S/MIME
PGP
SET
SSH
SMTP
HTTP
TCP
IP

Capa 2

802.1q VLANS

802.1x Port Based Network Access Control

802.1AE: MAC Security (MACsec)

Ver https://www.juniper.net/documentation/en_US/junos/topics/topic-map/understanding_media_access_control_security_qfx_ex.html

¿Por qué IPSEC?

- Verificación de los extremos de la comunicación
  - Verificar que un paquete se originó realmente en la contraparte.
- Integridad de los datos
  - Saber que los paquetes no han sido alterados.
- Confidencialidad
  - Saber que nadie ha visto el contenido de la comunicación.
- Protección contra el Replay
  - A veces no hace falta conocer el contenido del paquete para causar efectos indeseados. Por ejemplo capturar un paquete con el comando de formatear el disco y luego reenviarlo a otro equipo.
- Confidencialidad del flujo de datos
  - Que nadie sepa que estamos comunicándonos.

IPSEC: Seguridad en IP

- Específico para IP.
  - Opcional en IP v4.
  - Implementación obligatoria en IP v6.
- Transparente para aplicaciones y usuarios.
  - *Virtual Private Network* (VPN).
  - Usuario remoto, puede ser móvil *(road warrior).*

IPSEC RFCs

- RFC 1829: The ESP DES-CBC Transform
- RFC 2403: The Use of HMAC-MD5-96 within ESP and AH
- RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH
- RFC 2405: The ESP DES-CBC Cipher Algorithm With Explicit IV
- RFC 2410: The NULL Encryption Algorithm and Its Use With IPsec
- RFC 2451: The ESP CBC-Mode Cipher Algorithms
- RFC 2857: The Use of HMAC-RIPEMD-160-96 within ESP and AH
- RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
- RFC 3602: The AES-CBC Cipher Algorithm and Its Use with IPsec
- RFC 3686: Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)
- RFC 3947: Negotiation of NAT-Traversal in the IKE
- RFC 3948: UDP Encapsulation of IPsec ESP Packets
- RFC 4106: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
- RFC 4301: Security Architecture for the Internet Protocol
- RFC 4302: IP Authentication Header
- RFC 4303: IP Encapsulating Security Payload
- RFC 4304: Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)
- RFC 4307: Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)
- RFC 4308: Cryptographic Suites for IPsec
- RFC 4309: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)
- RFC 4543: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH
- RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)
- RFC 4806: Online Certificate Status Protocol (OCSP) Extensions to IKEv2
- RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec
- RFC 4945: The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX
- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
- RFC 5282: Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol
- RFC 5386: Better-Than-Nothing Security: An Unauthenticated Mode of IPsec
- RFC 5529: Modes of Operation for Camellia for Use with IPsec
- RFC 5685: Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)
- RFC 5723: Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption
- RFC 5857: IKEv2 Extensions to Support Robust Header Compression over IPsec
- RFC 5858: IPsec Extensions to Support Robust Header Compression over IPsec
- RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)
- RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)
- RFC 7383: Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation
- RFC 7427: Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)
- RFC 7634: ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec

Modos de operación

- Transporte: protege los datos de la capa de transporte, **extremo a extremo**.
  → Típicamente, entre *hosts.*

- Túnel: protege el datagrama IP completo, **encapsulándolo** dentro de otro.
  → Obligatorio cuando uno o ambos extremos se comportan como *gateway.*

Tres protocolos

- *Internet Key Exchange* (IKE).
  - **Negociación** de parámetros e intercambio seguro de **claves**.
- *Authentication Header* (AH).
  - Proporciona **autenticación** e **integridad**.
- *Encapsulating Security Payload* (ESP).
  - Además, puede proporcionar **confidencialidad**.

IKE

Internet Key Exchange (IKE) es el protocolo utilizado en IPSEC para intercambiar claves. Está definido en el RFC 2409. Utiliza intercambio de claves con el método de Diffie-Hellman para definir una clave secreta de sesión. Utiliza UDP/500. Tambien esta IKEv2

IKE opera en dos fases: La fase uno establece un canal para seguro. En la fase dos se eligen los protocolos a usar.

Se pueden utilizar 3 mecanismos de autenticación mutua:

- Preshared-keys
- Pares de claves (firma digital)
- Certificados Digitales.

Authentication Header (AH)

- Servicios de seguridad:
  - Integridad
  - Autenticación del origen de los datos
  - Anti-replay (opcional)

- Inicialmente usa estos algoritmos de MAC:
  - HMAC-MD5
  - HMAC-SHA1

Authentication Header (AH)

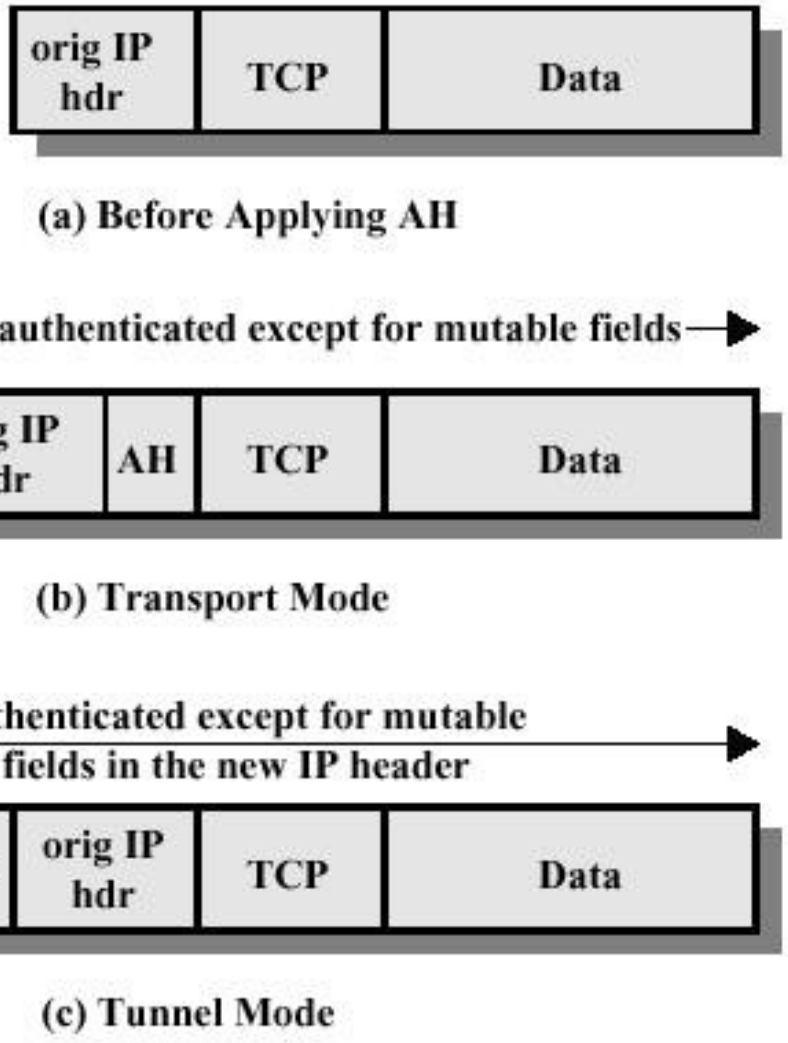
El protocolo AH se define como el protocolo 51 de IP

Su objetivo es proveer verificación de la fuente, integridad de paquetes y anti-replay pero no se preocupa por la confidencialidad de los datos o del flujo de datos.

El header de autenticación esta insertado en dos posibles lugares:

1. Entre el header IP original y el IP data payload (Modo transporte)
2. Como un prefijo del header original al cual se le agrega un nuevo header IP (Modo tunel).

Authentication Header (AH)

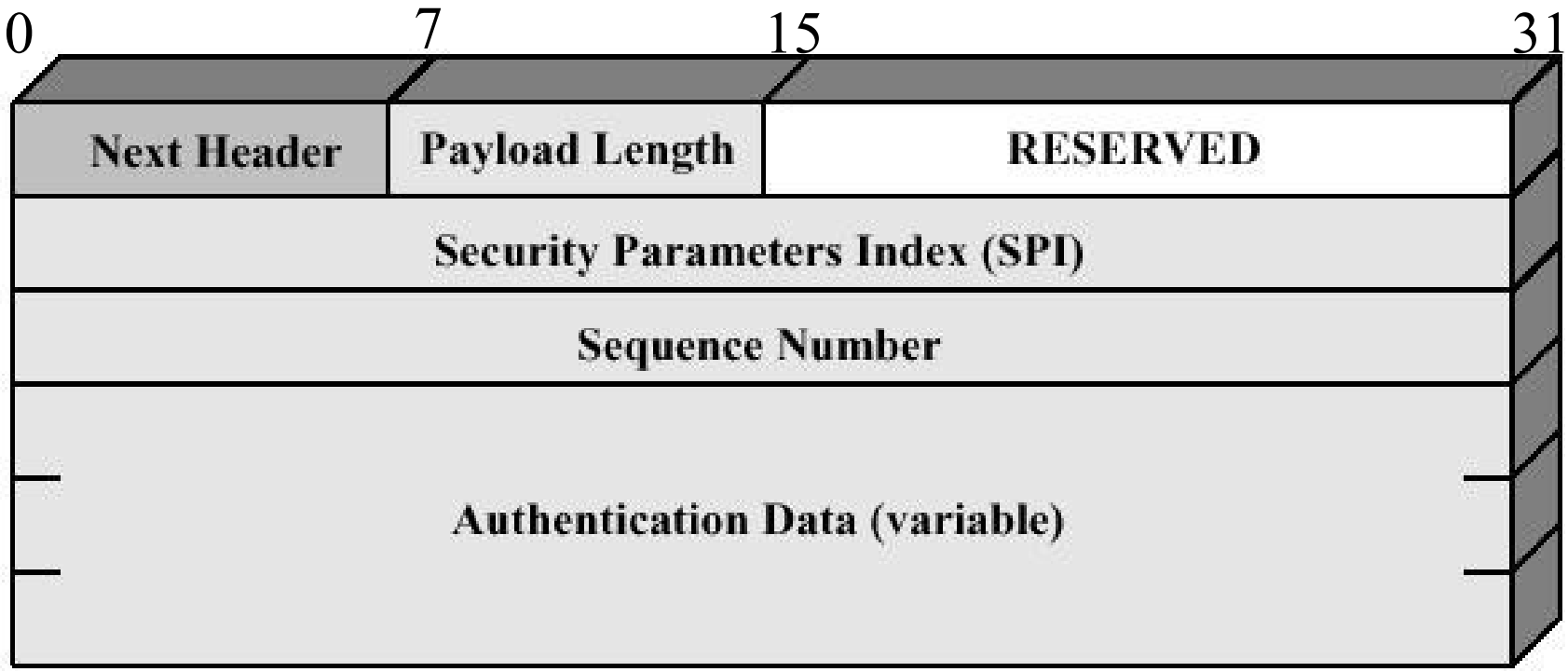


Authentication Header (AH)

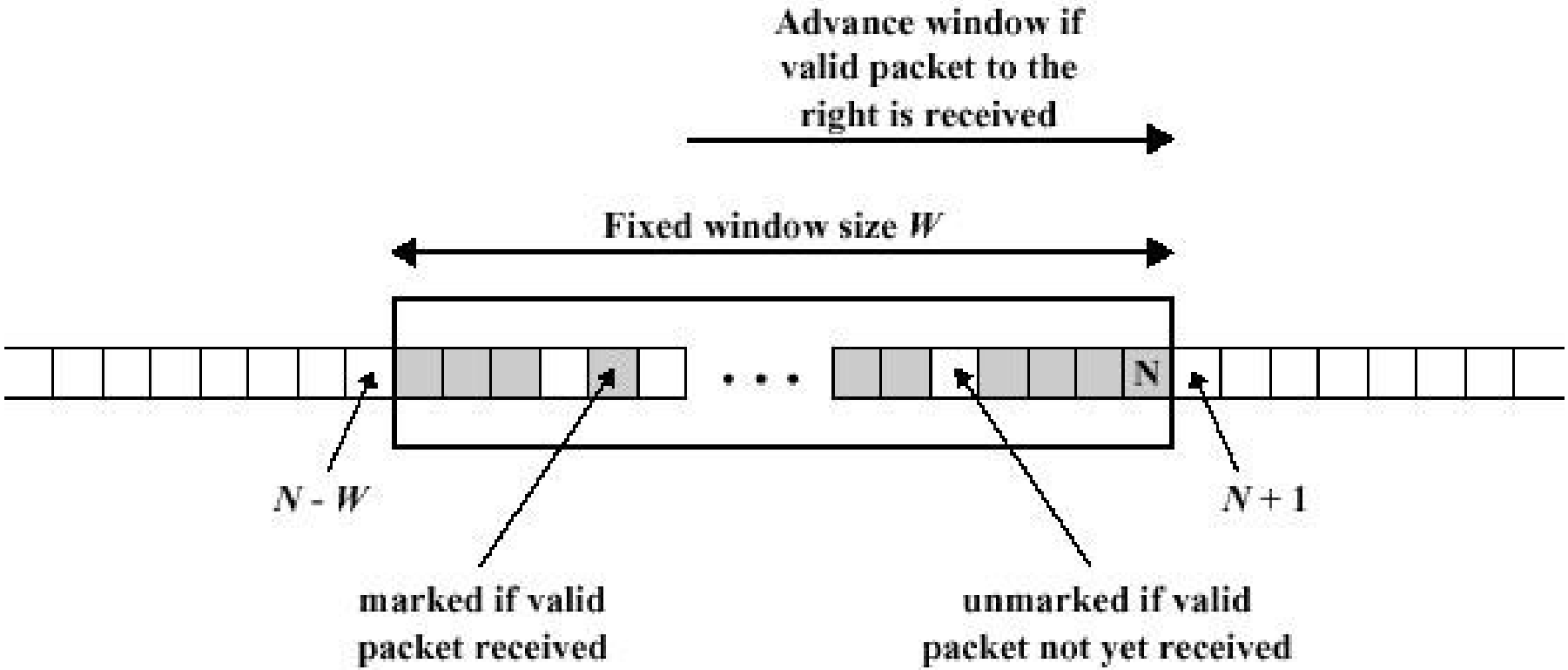
El Header AH contiene los siguientes campos:

- **Next Header**: El tipo de protocolo del paquete (8 bits).
- **Payload Length**: Longitud del AH. (8 bits)
- **Reserved**: 16 bits reservados para uso futuro. Hoy en día debe ser todos ceros.
- **SPI**: 32 bit "Security Parameter Index". Consideraciones:
  - Los valores entre 1 y 255 están reservados para uso futuro.
  - El SPI "0" esta reservado para uso local y no debe ser enviado a la red.
- **Sequence Number**: 32 bits utilizado para protección de replays. Esto tiene un efecto colateral, que limita la cantidad de paquetes a mil millones, una vez pasado ese limite, si la SA esta utilizando protección anti replay, deberá ser reestablecida.
- **Authentication Data**: una cadena de bits de longitud arbitraria para autenticación del paquete.

Authentication Header (AH)



Mecanismo anti-replay



Encapsulating Security Payload (ESP)

Los paquetes ESP son similares a AH, pero en el protocolo 50 de IP.

ESP provee confidencialidad de datos y en las condiciones correctas también puede proveer confidencialidad del flujo de datos.

La posición del header ESP es la misma que la del header AH, dependiendo del modo en que se este utilizando la SA.

ESP puede ser utilizado para proveer la misma funcionalidad que AH utilizando el algoritmo de cifrado “NULL”

Encapsulating Security Payload (ESP)

- Servicios de seguridad (uno, otro o los dos conjuntos):
  - Integridad
  - Autenticación del origen de los datos
  - Anti-replay (opcional)
    → Son los mismos que presenta AH
  - Confidencialidad
  - Resiste el análisis de tráfico (modo túnel)
    → Solamente disponibles en ESP.

- Emplea MAC:
  - HMAC-MD5
  - HMAC-SHA1

Encapsulating Security Payload (ESP)

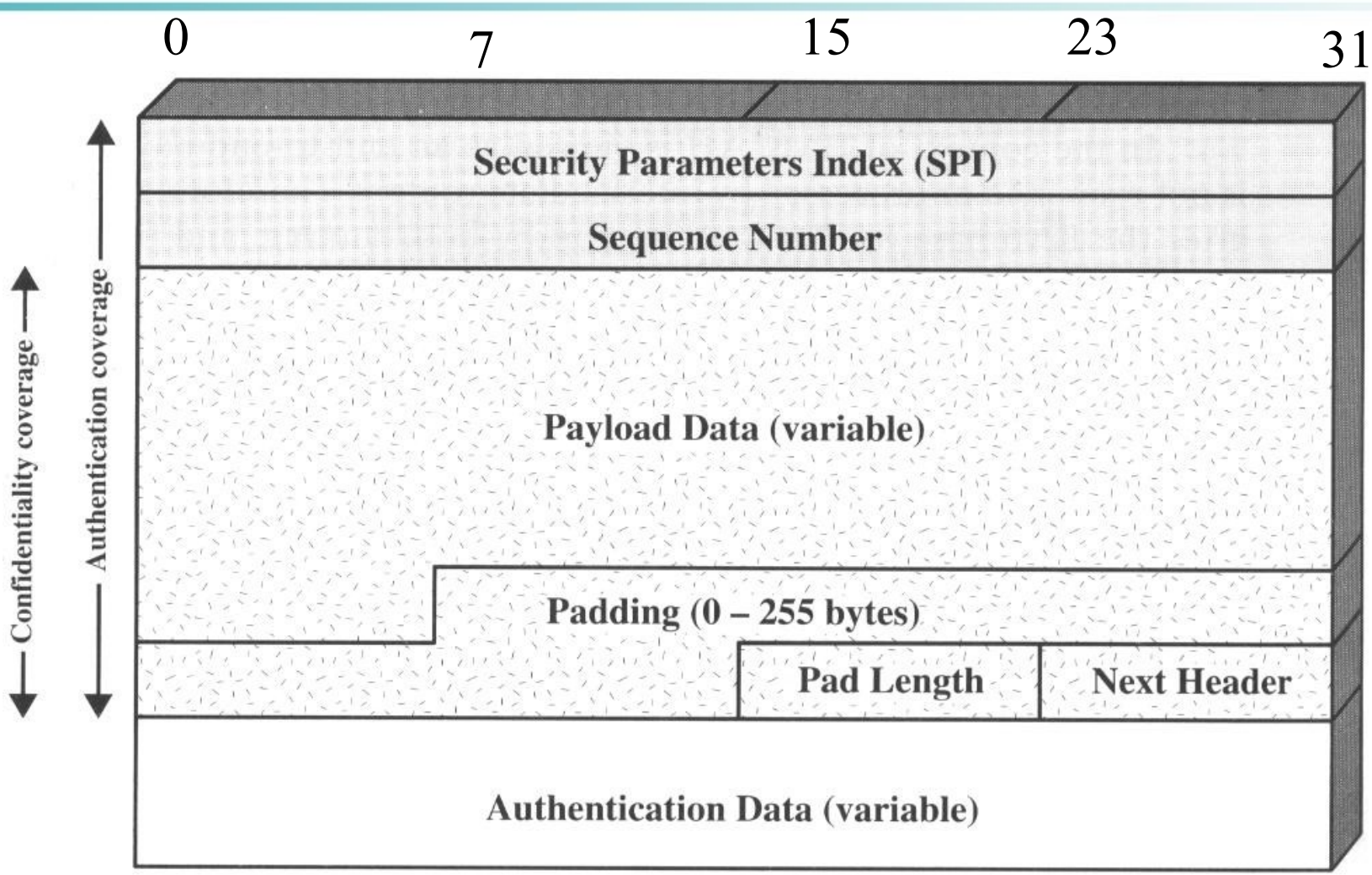
- Cifrado para confidencialidad:
  - Algoritmos obligatorios de cifrado:
    DES (RFC 2405) y NULL (RFC 2410)
  - Algoritmos opcionales de Cifrado (RFC 2451):
    3DES, CAST-128, RC5, IDEA, Blowfish y AES
  - Modo CBC

Encapsulating Security Payload (ESP)

El header ESP contiene los siguientes campos:

- **SPI**: valor de 32 bits
- **Sequence Number**: 32 bits de protección anti-replay
- **Payload Data**: Longitud variable. Payload cifrado.
- **Padding**: 0-255 bytes. Utilizados para proteger al payload cifrado de análisis criptoanalítico.
- **Pad Length**: 8 bits, indican la longitud del padding.
- **Next Header**: Tipo de protocolo del paquete. 8 bits.
- **Authentication Data**: Un checksum del paquete.

Encapsulating Security Payload (ESP)



NAT traversal

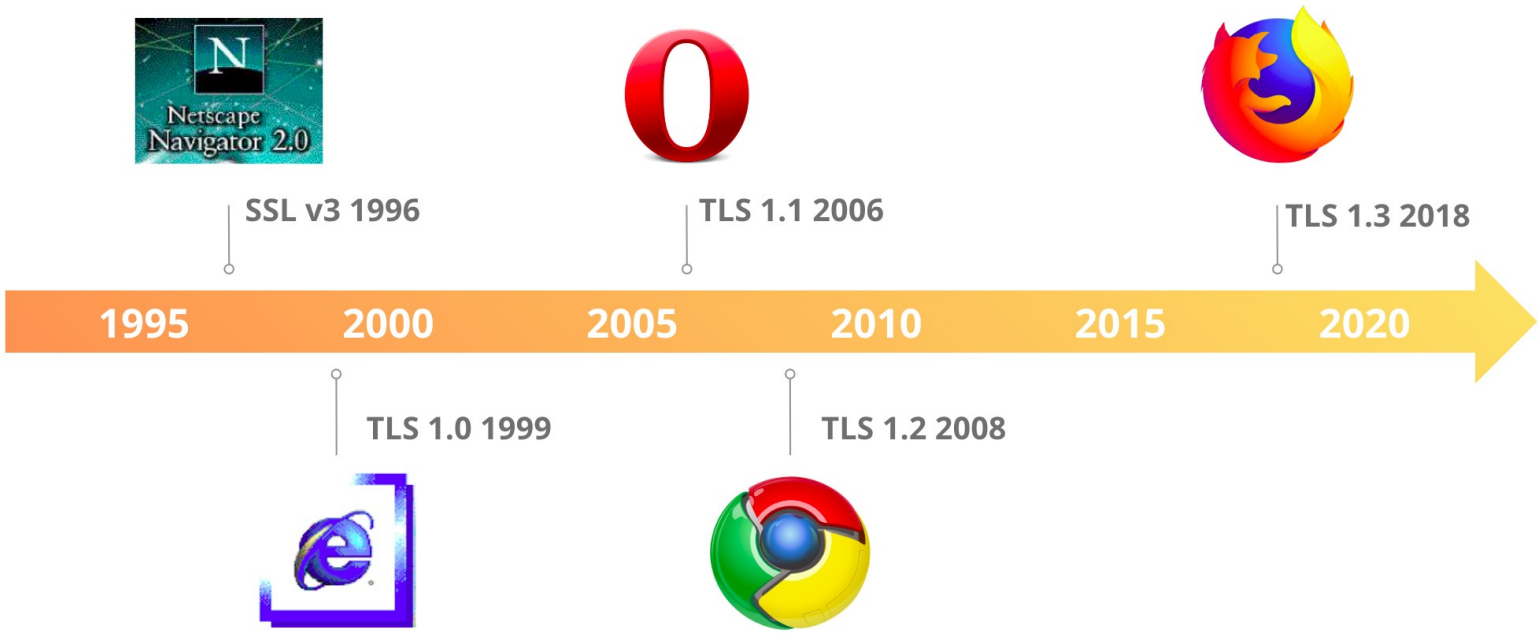
- NAT supone la **manipulación** y por ende **modificación** del paquete IP original para resolver el problema del direccionamiento IP (público/privado)

- IPSec considera la acción de **NAT** como un **ataque** a la integridad del paquete y la conexión no se realiza.

- Nat-Traversal: Solución propuesta definida en RFCs 3947 y 3948.

  Consiste en **encapsular** la trama cifrada por IPSec (antes de añadirle la cabecera IP final) con un campo **UDP** y sobre esta cabecera extra se realizan las operaciones de NAT.

TLS / SSL


Netscape Navigator 2.0
SSL v3 1996
TLS 1.1 2006
TLS 1.3 2018
1995
2000
2005
2010
2015
2020
TLS 1.0 1999
TLS 1.2 2008

Transport Layer Security (TLS)

- Versión actualizada de **SSL** (Secure Sockets Layer)
  - La última versión de SSL (Netscape) fue 3.0
  - TLS se identifica como SSL v 3.1
  - Similar, pero no compatible directamente.
  - Especificado en RFC 2246 (1999). Extendido posteriormente en RFC 3546 (2003), 5246 y 8446
- Protege una sesión entre **cliente** y **servidor**.
  - Típicamente, HTTP (navegador y web server).
- Requiere protocolo de transporte confiable.
  - Por ejemplo **TCP**.

Transport Layer Security (TLS)

TIPS:

- RFC 2246
- Basado en SSL 3.0.
- Incompatible con este SSL 3.0
- Una de las ventajas que proporciona sobre SSL es que puede ser iniciado a partir de una conexión TCP ya existente, lo cual permite seguir trabajando con los mismos puertos que los protocolos no cifrados.
- SSL es un protocolo incompatible con TCP, lo cual significa que no podemos establecer una conexión de un cliente TCP a un servidor SSL ni al revés, y por lo que es necesario diferenciarlos utilizando distintos puertos.
- Con TLS puede establecerse la conexión normalmente a través de TCP, y luego activar sobre el mismo el protocolo TLS.

Problemas con SSL v2

- **Utiliza la misma clave para integridad y confidencialidad.**
- **Utiliza una función MAC dónde el secreto es un prefijo y usa MD5, por ende es vulnerable a “length extension attacks”**
- **No protege el handshake, por ende es posible mediante un man-in-the-middle forzar la utilización de algoritmos de cifrado débiles.**
- **No tiene un mensaje de fin de conexión, por ende es posible terminar la conexión con un TCP FIN (IP spoofing).**

Transport Layer Security (TLS)

Mecanismos de seguridad de SSLv3 y TLS:

- Numeración secuencial de los paquetes que contienen datos.
- Incorporación de esa numeración al cálculo de los MAC.
- Protección frente a ataques que intentan forzar el empleo de versiones antiguas del protocolo o cifrados más débiles.
- El mensaje que finaliza la fase de establecimiento de la conexión incorpora un hash de todos los datos intercambiados por ambos interlocutores.

Mecanismos de seguridad adicionales de TLS:

- Se utilizan dos funciones de hash (MD5 y SHA-1) para calcular el MAC.
- La construcción de MAC está basada en el HMAC de RFC 2104.

TLS 1.2 y TLS 1.3

- **RFC 8996 Marzo 2021: Depreca tls 1.0 y tls1.1**
- **TLS 1.2 No usa md5 ni sha1, sino sha-256**
- **No puede negociar SSLv2**

- **Ver**

https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

Tls 1.3, RFC 8446 (2018)

Deshabilita un montón de algoritmos, es más performante para iniciar sesiones, permite ocultar el FQDN. Mecanismos para dificultar downgrade attacks. Soportado en navegadores modernos

Servicios de Seguridad

- Autenticación
  - del servidor frente al cliente;
  - opcionalmente, del cliente frente al servidor.

  →Mediante **certificados** de clave pública.

- Integridad

  →Mediante **MAC** y números de secuencia.

- Confidencialidad
  - opcional

  →Mediante **cifrado** con algoritmo simétrico.

Fases de TLS

Una comunicación a través de TLS implica tres fases:

- Establecimiento de la conexión y negociación de los algoritmos criptográficos que van a usarse en la comunicación, a partir del conjunto de algoritmos soportados por cada uno de los interlocutores.

- Intercambio de claves, empleando algún mecanismo de clave pública y autentificación de los interlocutores a partir de sus certificados digitales.

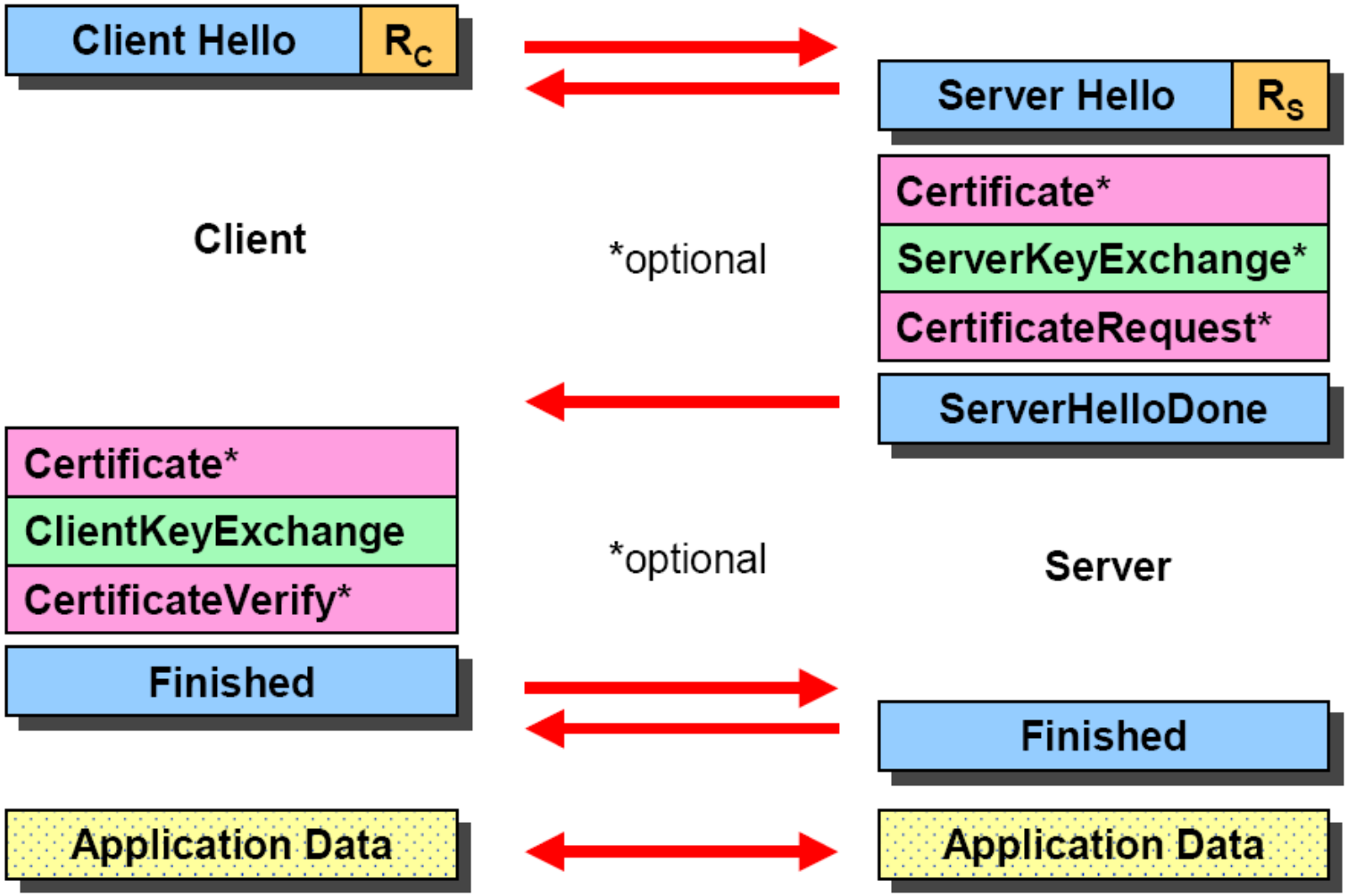
- Cifrado simétrico del tráfico.

Fases

- Handshake:
  - Negociación de **algoritmos** y parámetros.
  - **Autenticación** (del servidor o mutua).
  - Canal seguro para **compartir** un secreto inicial.
  - Derivación de **claves** en cada extremo.
  - **Integridad** de todo el intercambio.

- Transferencia de datos:
  - Usa las **claves** anteriormente derivadas.
  - Provee **integridad**.
  - Opcionalmente, provee **confidencialidad**.
  - Autentica el **cierre** de cada conexión.

Handshake


Client Hello
R_C
Server Hello
R_S
Certificate*
ServerKeyExchange*
CertificateRequest*
Client
*optional
ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
*optional
Server
Finished
Finished
Application Data
Application Data

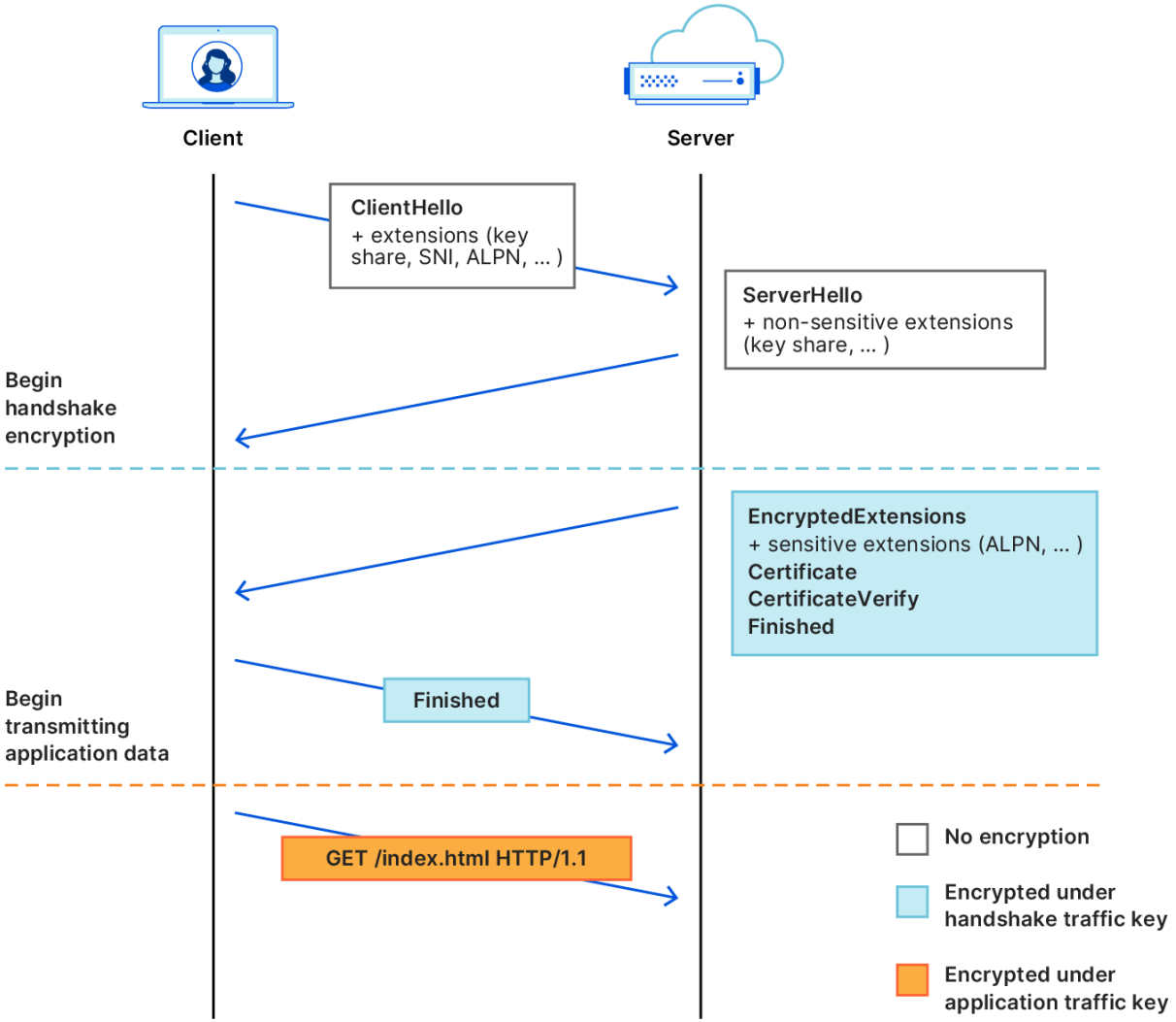
Handshake

El estado de la sesión SSL es controlado por el protocolo handshake de TLS. Se emplea para negociar los atributos de sesión.

Cuándo un cliente TLS y un servidor TLS comienzan a comunicarse, acuerdan:

- algoritmos criptográficos,
- versión del protocolo,
- opcionalmente se autentican
- con algoritmos de cifrado asimétrico generan claves de sesión.

Handshake TLS 1.3



PFS

- **Perfect Forward Secrecy o Forward Secrecy**

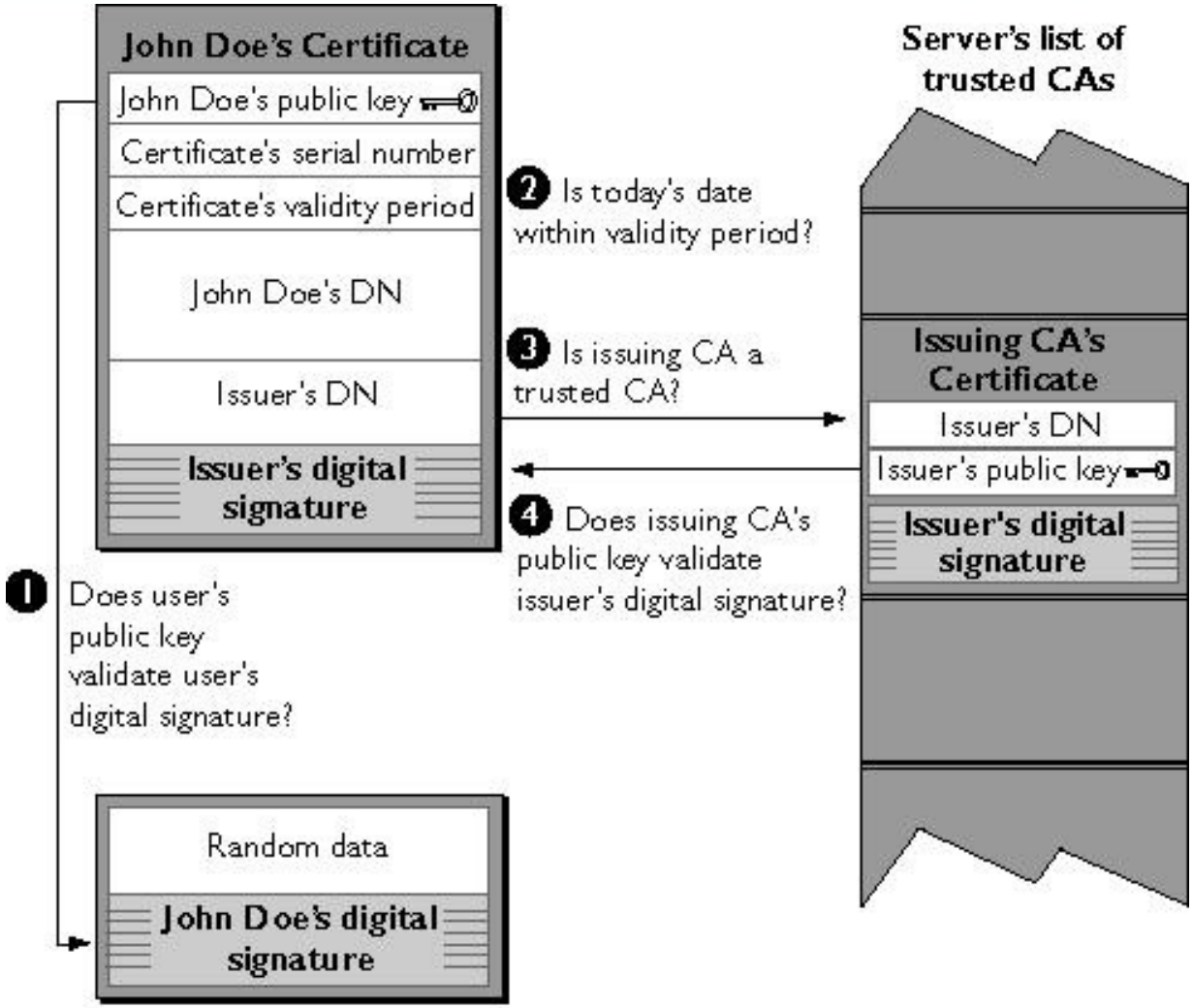
- **RFC 4251 (SSH): PFS is essentially defined as the cryptographic property of a key-establishment protocol in which the compromise of a session key or long-term private key after a given session does not cause the compromise of any earlier session**

  https://community.qualys.com/blogs/securitylabs/2013/06/25/ssl-labs-deploying-forward-secrecy

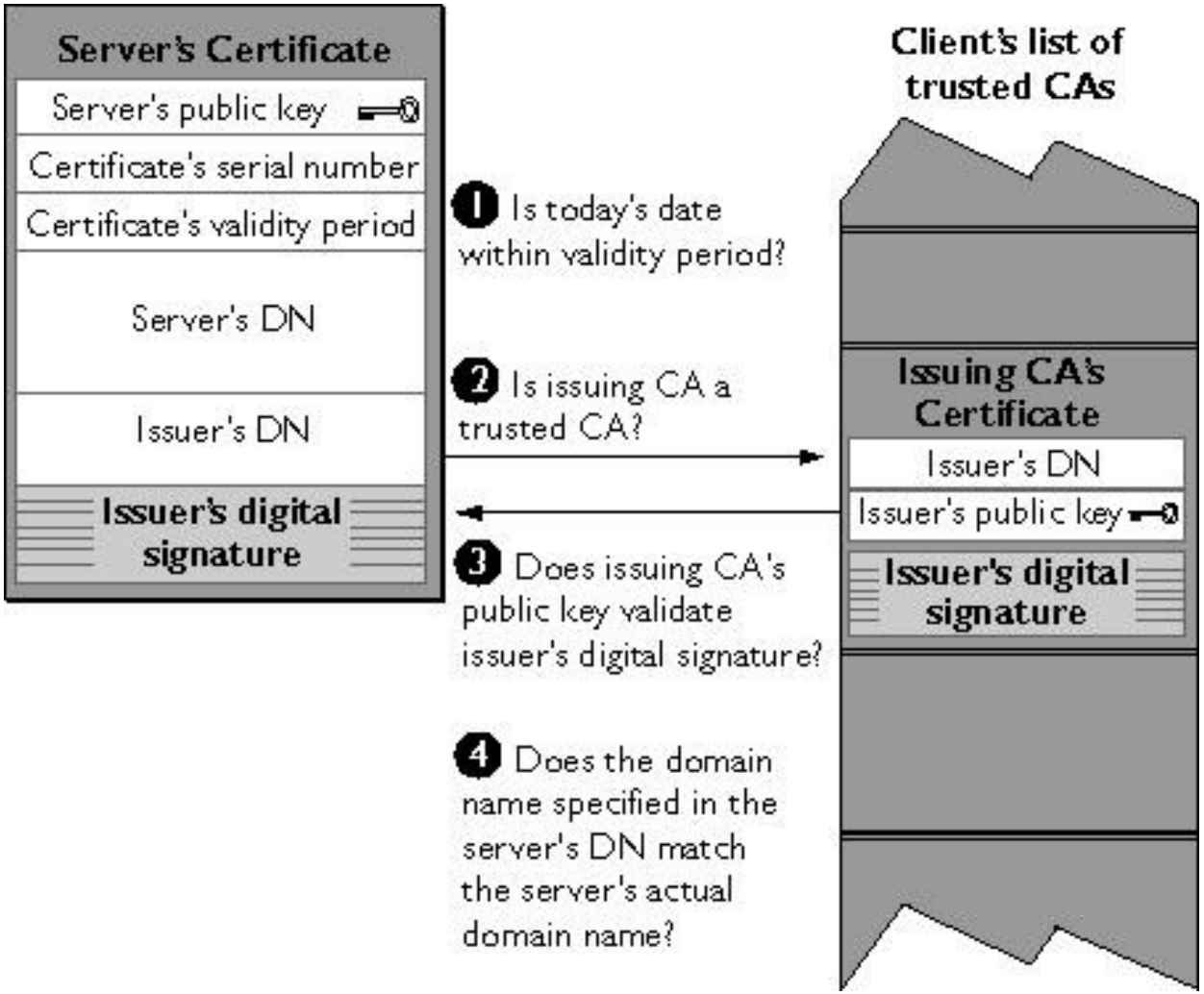
Autenticación del cliente

- Con el propósito de controlar el acceso, el servidor puede requerir la **autenticación** del **cliente**, solicitándole su certificado de clave pública y verificando que posea la clave privada:
  - Después de presentar su certificado, el servidor se lo solicita al cliente mediante *CertificateRequest.*
  - El cliente lo envía, mediante *Certificate.*
  - Como comprobación de la posesión de su clave privada **KRA**, el cliente envía el mensaje *Certificate Verify*, que consiste en la **firma** de los mensajes de *handshake* intercambiados hasta ese momento.

Autenticación del cliente



Verif. Certificado Servidor



OpenVPN

- **Busca implementar VPNs de forma más portable y sencilla que IPSEC, utilizando TLS.**
- **No requiere de la complejidad del protocolo IKE.**
- **Puede trabajar en modo bridging o en modo routing**
- **Utiliza la implementación de openssl**
- **www.openvpn.net**

Wireguard

- **Incorporado en el kernel de linux desde la versión 5.6 (más performance)**
- **Connection-less**
- **Más moderno, menos líneas de código, menos algoritmos. Más sencillo de usar**

https://www.wireguard.com/protocol/

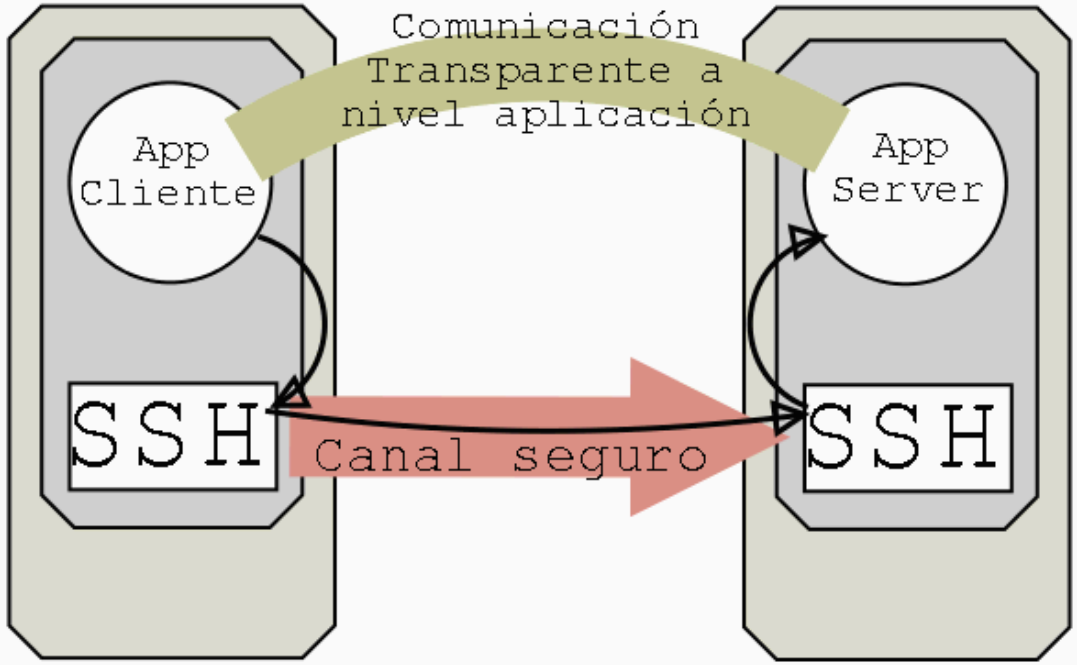
https://www.wireguard.com/papers/wireguard.pdf

SSH

Túneles SSH – Port forwarding

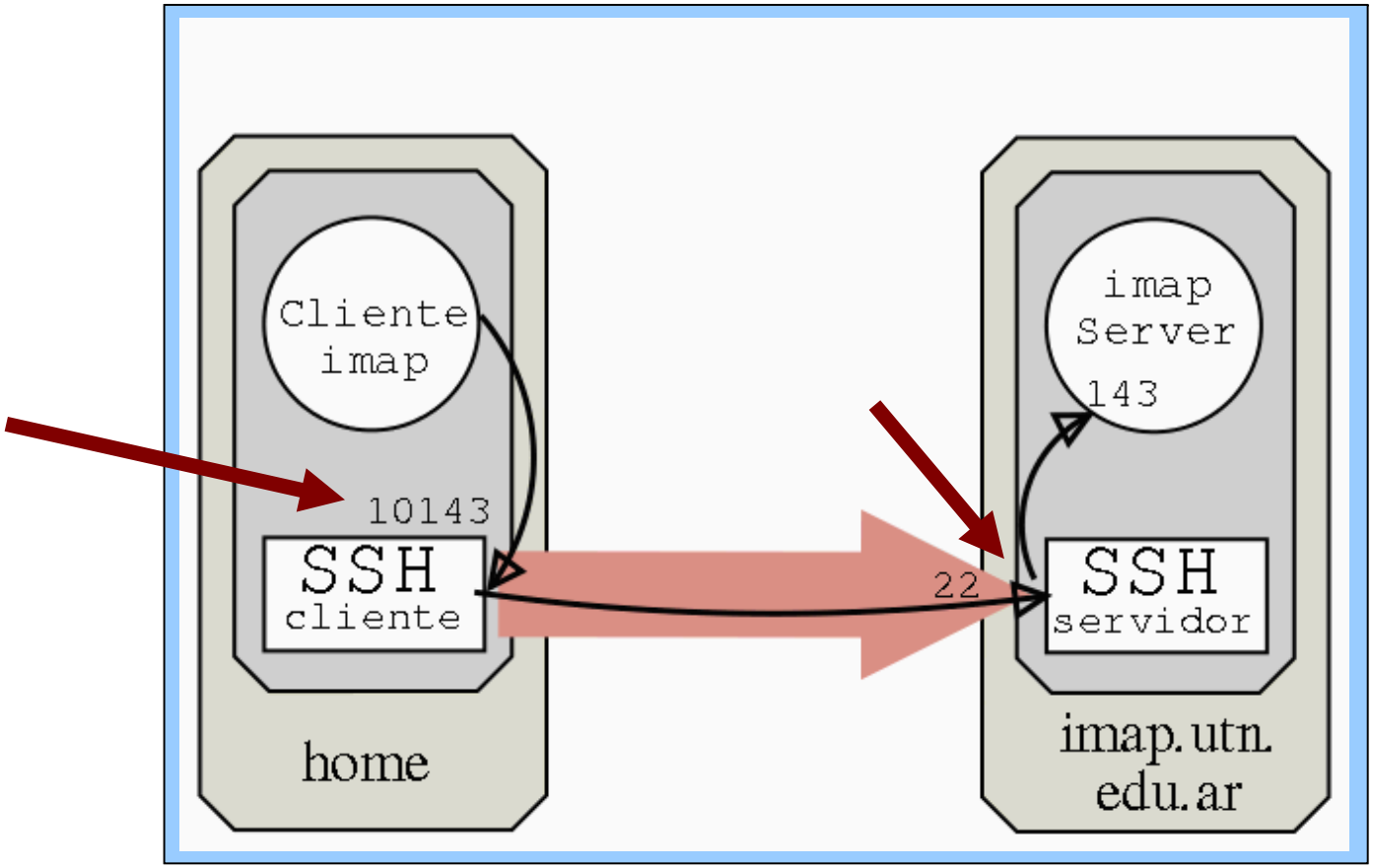
- Ahorrarse el software necesario para una complicada VPN
- Proteger mi clave plana de FTP, HTTP, Telnet, POP3, IMAP, authSMTP…
- Atravesar un firewall donde solo el servicio de SSH está permitido
- Acceder a servicios TCP internos de una LAN con direcciones privadas

Los túneles SSH básicos no permiten forwardear paquetes UDP o protocolos no IP



Túneles SSH – Local simple

```
home:~$> ssh -L 10143:localhost:143 imap.utn.edu.ar
```



Túneles con SSH – SOCKS (dynamic forwarding)

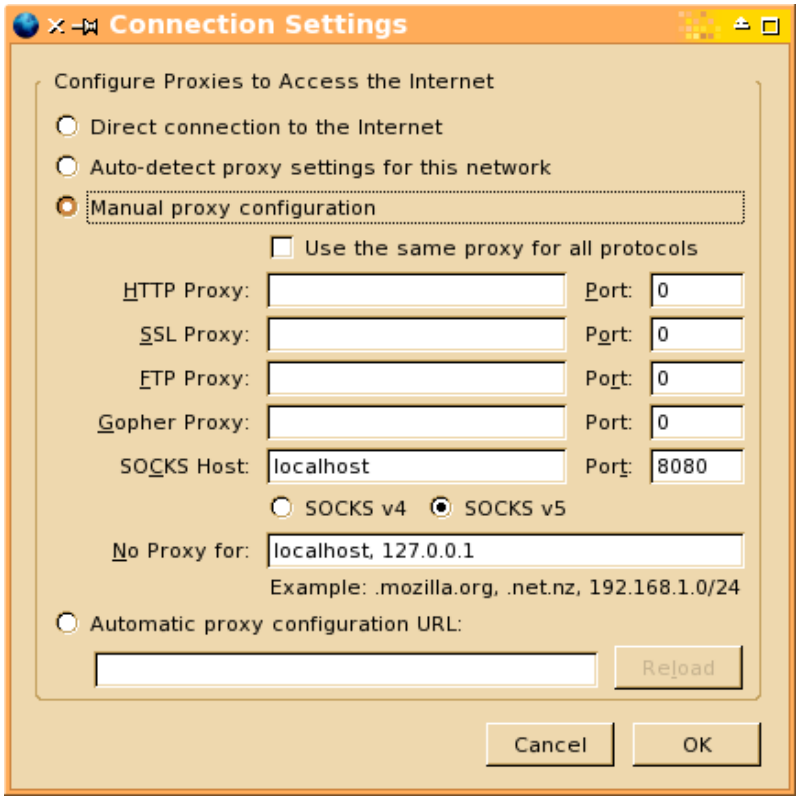
- SOCKet Secure es una forma de conectarse a una red, un proxy server
- El modificador *-D* genera un túnel local “dinámico”, haciendo que el cliente SSH se transforme en un SOCKS server

- Para emular un SOCKS server en el puerto 8080:

```
$> ssh -D8080 <user>@server
```

- La aplicación a forwardear debe ser SOCKS aware, o hay que usar un “proxifier”:

https://github.com/rofl0r/proxychains-ng



OpenSSH's built in tunneling

- **OpenSSH tiene soporte TUN/TAP usando -w<local-tun-number>:<remote-tun-number>.**

PermitTunnel

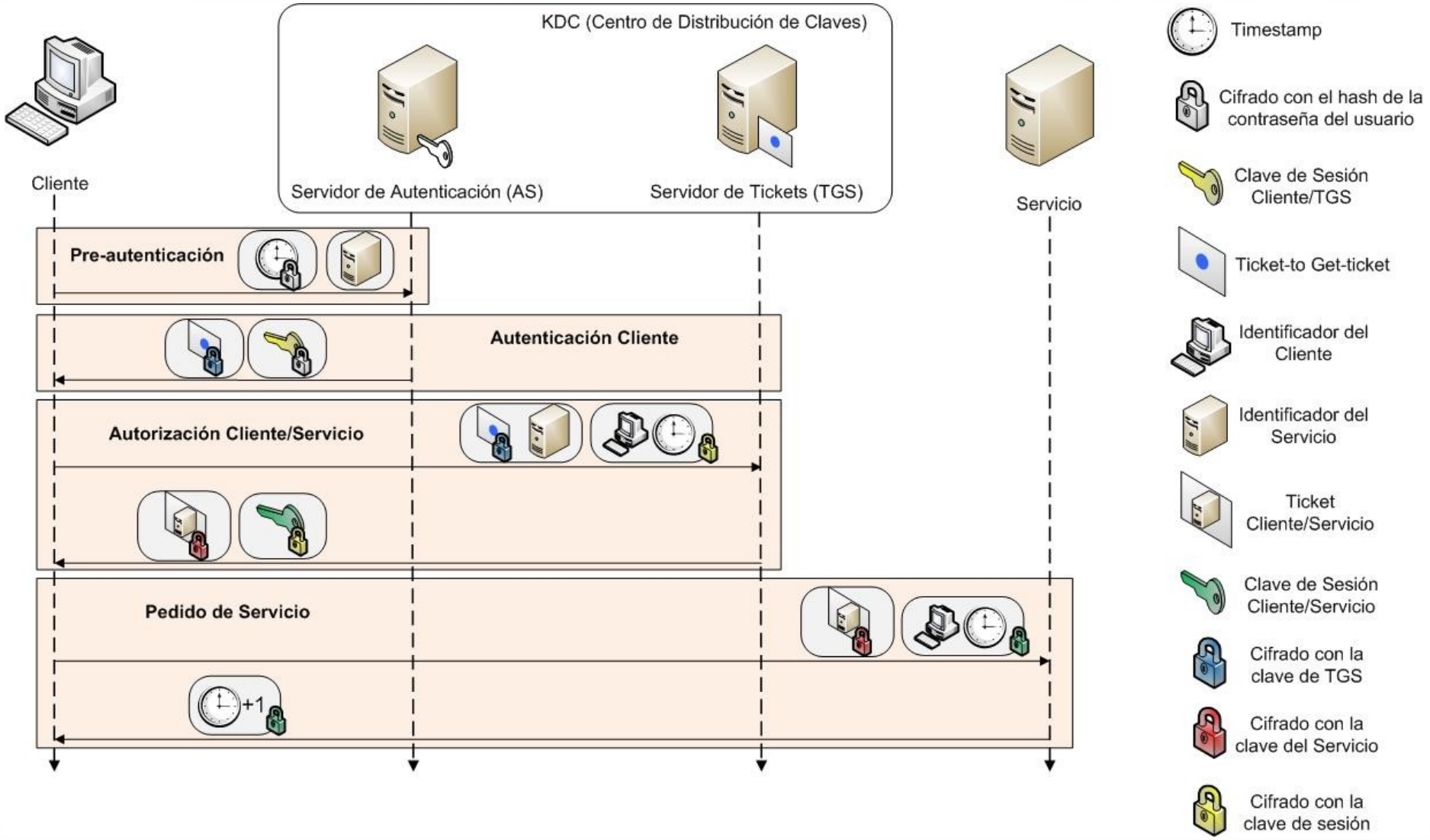
Specifies whether tun(4) device forwarding is allowed. The argument must be yes, point-to-point (layer 3), ethernet (layer 2), or no. Specifying yes permits both point-to-point and ethernet.

The default is no.

Kerberos

- Protocolo diseñado para proveer autenticación sobre un canal inseguro, desarrollado por el MIT.
- Utiliza criptografía simétrica y requiere de un tercero confiable (con quien todas las partes involucradas comparten un secreto).
- Provee autenticación mutua, y los mensajes empleados por el protocolo se encuentran protegidos contra ataques de replay y de eavesdropping.
- La versión 4 se publicó a finales de los ’80, la versión actual (v5) se estandarizó en 1993 (RFC 1510) y se modificó en 2005 (RFC 4120).
- Windows 2000 y superior utilizan Kerberos v5 como su mecanismo de autenticación por defecto.

Kerberos: Diagrama



Kerberos: ¿Qué contienen los tickets?

- ***Ticket-to Get-Ticket*:** Identificador del cliente, dirección de red del cliente, timestamp, período de validez del ticket, clave de sesión C*liente/TGS.*

- ***Client-to-service ticket:*** Identificador del cliente, identificador del servicio, dirección de red del cliente, timestamp, período de validez del ticket, clave de sesión *Cliente/Servicio.*

Zero Trust

El modelo Zero-trust se basa en varios principios:

Garantizar el acceso seguro a todos los recursos independientemente de donde estén localizados.

Seguir una estrategia de mínimo privilegio y seguir una política estricta de control de acceso.

Inspeccionar y registrar todo el tráfico para validar la actividad en la red.