Modelo TCP/IP

Vulnerabildades

IPsec

SSL/TLS

SSH

Tema 5. Protocolos seguros

Seguridad en Sistemas Informáticos

Noviembre-2018

Modelo TCP/IP

Vulnerabildades

IPsec

SSL/TLS

SSH

Contenido

1. Conceptos básicos. Modelo TCP/IP
2. Vulnerabilidades en redes TCP/IP
3. Seguridad en nivel de RED: IPsec
4. Seguridad en nivel de TRANSPORTE: Capa SSL/TLS
5. Seguridad en nivel de APLICACIÓN: SSH

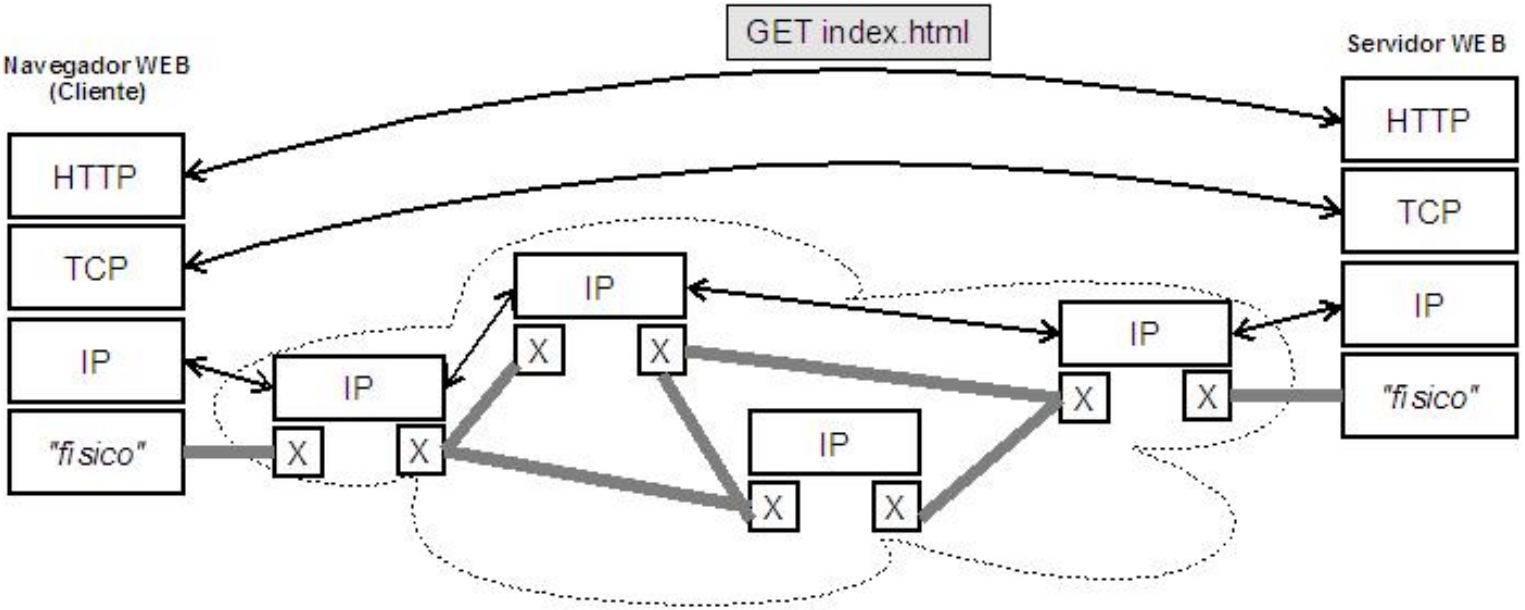
Conceptos básicos. Modelo TCP/IP

Pila de protocolos

APLICACIÓN	HTTP, TELNET, DNS, SMTP, ...
TRANSPORTE	TCP, UDP
RED	IP (ICMP) (ARP)
Enlace+Físico	ethernet, atm, ppp, ...

- **Capa "física"**
  - Protocolos de comunicación de tramas a bajo nivel: LLC (*Logical Link Control*), MAC (*Medium Access Control*)
    - ARP (*Address Resolution Protocol*): mapeo direcciones IP $\leftrightarrow$ direcciones hardware
  - Comunicación a nivel físico: por cable (coaxial, par trenzado, ...), inhalámbrica, ...

- **Capa de red** Define una comunicación **punto-a-punto** entre encaminadores (*routers*)
  - Confiere ”unidad” a todos los equipos de la red
  - Proporciona { asignación de direcciones únicas; encaminamiento de paquetes (no fiable); interconexión de redes; control de congestión }
  - Protocolos: IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol*) [protocolo de control]
- **Capa de transporte** Define una comunicación **extremo-a-extremo** entre sistemas finales
  - Da fiabilidad a la red para su uso desde las aplicaciones de nivel superior
  - Proporciona: { control de flujo y de errores; conexión y fiabilidad }
  - Protocolos: TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*)
- **Capa de aplicación** Comunicación entre aplicaciones de los sistemas que usan al red
  - Servicios y programas de nivel de usuario
  - Protocolos: HTTP (*HyperText Transfer Protocol*), Telnet, DNS (*Domain Name System*), SMTP (*Simple Mail Transfer Protocol*),...



Encapsulamiento

Cada capa { ofrece servicios a las capas superiores / solicita servicios de las capas inferiores

- Manejan sus propios mensajes
- Añade sus propias cabeceras de control

(a) Protocolo de red (IP)

- **Direccionamiento** En IP versión 4: direcciones de 32 bits (4 bytes)
  - 5 tipos de direcciones

A:	0.0.0.0	127.255.255.255
B:	128.0.0.0	191.255.255.255
C:	192.0.0.0	223.255.255.255
D:	224.0.0.0	239.255.255.255
E:	240.0.0.0	247.255.255.255

  - Rangos reservados para uso en redes privadas

A:	10.0.0.0	10.255.255.255
B:	172.16.0.0	172.31.255.255
C:	192.168.0.0	192.168.255.255

  - Direcciones de *bucle inverso* (loopback)

    127.0.0.0 – 127.255.255.255

    127.0.0.1 : interfaz de red para servicios locales
  - Direcciones de tipo D: para uso en *multidifusión* (broadcast)
  - Direcciones de tipo E: para usos experimentales

  En la cabecera de cada paquete IP del nivel de red se incluye la *dirección del equipo destino* y la del *equipo origen*

- **Subredes** Suele usarse una parte de la dirección IP para identificar la red y otra para identificar cada equipo concreto.
  - Una máscara binaria define la parte de la dir. IP que identifica a la red
  - Por defecto se usan las siguientes mácaras en cada tipo de dirección

	red	equipo	máscara	
A:	1 bytes	3 bytes	255.0.0.0	(16777216 equipos/red)
B:	2 bytes	2 bytes	255.255.0.0	(65536 equipos/red)
C:	3 bytes	1 bytes	255.255.255.0	(255 equipos/red)

  - División en subredes (ejemplo)
    - Red de tipo C, privada: 192.168.1.0

      Máscara: 255.255.255.0 $\Rightarrow$ 253 equipos (192.168.1.1 – 192.168.1.254)
    - Dividida en 2 subredes

      Máscara 255.255.255.128 $\Rightarrow$ 128 direcciones en cada subred
      Subred 1: 192.168.1.1 – 192.168.1.127 (126 equipos)

      Subred 2: 192.168.1.129 – 192.168.1.254 (126 equipos)

(b) Protocolos de transporte (TCP, UDP)

Dividen el flujo de bytes llegado del nivel de aplicación en paquetes de tamaño adecuado para su transmisión por el nivel inferior (red) añadiendo sus cabeceras de control.

- **TCP:** Orientado a conexión y fiable
  - Servicio de envío de mensajes con conexión
    - 3 etapas { establecimiento de conexión; transferencia; fin de conexión }
    - Negociación conexión en 3 pasos (SYN, SYN-ACK, ACK)
  - Garantiza entrega en destino, en orden, sin pérdidas y sin errores
    - Protocolo IP no es fiable: no garantiza recepción ni orden
  - Usa { núms. de secuencia; núms. de ACK } para mantener/confirmar el orden de entrega de los mensajes y evitar pérdidas
    - solicita retransmisión si es necesario

- **UDP:** No orientado a conexión y no fiable
  - No establece conexión $\Rightarrow$ cada mensaje (datagrama) incluye toda la información necesaria para que llegue a su destino
  - No garantiza el orden de llegada ni recuperación de datagramas perdidos
  - Usado en protocolos sencillos (DNS, DHCP) que no requieran confirmación de entrega

(c) Puertos y servicios

- **Puerto:** concepto usado para identificar las aplicaciones emisor y receptor
- Cada extermo de la conexión tiene asociado un número de puerto que indentifica sus mensajes (16 bits $\Rightarrow$ 0 .. 65535)
- Los puertos se especifican al establecer la conexión (en TCP) y se incluyen en las cabeceras de los paquetes de nivel de transporte intercambiados (puerto origen + puerto destino)
- **Puertos bien conocidos**
  - Puertos reservados asociados por convención a aplicaciones/servicios *"bien conocidos"*
  - Asignados por la IANA (*Internet Assigned Numbers Authority*) [`http://www.iana.org`]
  - Puertos del **0** al **1024**
  - Usados por el sistema o por procesos con privilegios (servidores)
  - Procesos "normales" (clientes) usarán puertos no privilegiados (de 1025 a 65535)
    - Se les asignan de forma aletatoria
  - Ejemplos:

Servicio	Puerto/s (protocolo)	Servicio	Puerto/s (protocolo)
ftp	21/tcp	ssh	22/tcp
telnet	23/tcp	smtp	25/tcp
dns	53/tcp, 53/udp	finger	79/tcp
pop3	110/tcp	http	80/tcp
https	443/tcp	rpc	11/tcp, 111/udp (*portmapper*)

Vulnerabilidades TCP/IP

Capa *"física"*

- Vulnerabilidades en el acceso directo al medio físico
- Problemas de { control de accesos; confidencialidad }
- Especialmente vulnerables las líneas punto a puntos, redes de difusión (bus) y redes inhalámbricas

Capa de red

- Problemas de { confidencialidad; autenticidad; integridad }
- Escucha y acceso a paquetes IP (*sniffing*)
  - Paquetes IP (vers. 4) van en claro y se pueden leer/modificar/falsificar
- Suplantación de mensajes/direcciones IP (*IP spoofing*)
  - Protocolo IP sólo "sabe" de 2 direcciones IP (origen y destino)
  - Pueden falsificarse $\Rightarrow$ no hay certeza sobre la autenticidad del origen
  - Posibilidad de reconstrucción de mensajes
- Ademas: en protocolo ARP, falsificación de respuestas ARP (*ARP poisoning*, *ARP spoofing*)

Capa de transporte

- Mismos problemas de { confidencialidad, autenticidad, integridad } que protocolo IP
  - Paquetes TCP y UDP van en claro $\Rightarrow$ posible acceso y modificación
- Además: deficiencias en diseño e implementación de TCP
  - Ataques de denegación de servicio (*DoS*) a nivel de transporte
    - Derivado de errores en algunas implementaciones de la pila TCP/IP
    - Debido a las esperas (*time-outs*) del protocolo de establecimiento de conexión de 3 pasos
    - Iniciar conexión (SYN) y no responder al asentimiento (SYN-ACK) dejando en espera al otro extremo
  - Posibilidad de interceptar conexiones TCP abiertas (secuestro conexiones TCP, *TCP hijacking*)
    - Generar paquetes TCP con fines malignos que *"encajen"* en el flujo de una conexión ya establecida
    - Debido a la poca exigencia en cuanto a autenticación (e integridad de mensajes) de los equipos en comunicación

Capa de aplicación

- Múltiples protocolos/aplicaciones $\Rightarrow$ multitud de deficiencias de seguridad específicas de cada uno (y de cada implementación concreta)
- Ejemplos
  - **DNS:** Suplantación de DNS (*DNS poisoning*)
    - Atacante puede modificar la información almacenada en la BD del servidor DNS
    - Atacante puede reemplazarlo completamente (suplantación de IP)
  - **Telnet:** Deficiencias en proceso de autenticación
    - Telnet autentica en sistema destino mediante login+passwrod
    - Login y password se transmiten en claro
    - Trivial caputar esa información mediante ténicas de *sniffing*
    - Solución: uso de SSH (cifra las conexiones)
  - **FTP:**
    - Mismas deficiencias que Telnet (toda la comunicación en claro)
    - Posibilidad de conexiones anónimas $\Rightarrow$ punto de entrada para ataques que aprovechen agujeros del software del servidor FTP.

Seguridad en redes TCP/IP

Dos aproximaciones **complementarias**.

- Filtrado y control de acceso a equipos/aplicaciones sensibles.
  $\Rightarrow$ uso de Cortafuegos/Firewalls
  $\Rightarrow$ uso de Detectores de intrusiones en RED (NIDS)
- Uso de protocolos seguros.

  Diferentes posibilidades para asegurar los protocolos de los distintos niveles

  - **Seguridad en el nivel de** RED
    - Se busca garantiza que el tráfico que envían los protocolos de nivel superior(TCP,UDP) se transmita protegido.
    - **Inconveniente:** puede ser necesario adaptar/reconfigurar la infraestructura de red (routers) para gestionar la nueva información añadida al protocolo IP.
  - **Seguridad en el nivel de** TANSPORTE
    - **Ventaja:** sólo precisa actualizar las implementaciones de TCP y/o UDP en los extremos de la comunicación.
    - Supone un cambio a nivel de software (S.O. o librerías de sistema)
  - **Seguridad en el nivel de** APLICACIÓN
    - **Ventaja:** mayor flexibilidad
    - Se puede responder mejor a las necesidades específicas de protocolos concretos (SSH, SET)

Seguridad en nivel de RED: IPsec

- Diseñado como alternativa protegida al protocolo IP
  - conj. protocolos definido como parte de IPv6 (nueva versión)
  - adaptado para su uso en IPv4 (actual)
- Permite cifrar y/o autentificar **todo** el tráfico a nivel IP
- Amenazas que evita
  - escucha y captura de paquetes IP
  - paquetes IP con dirección de origen falsa
    - evitar ”engaños” a aplicaciones que usan autentificación basada en direcciones IP

Servicios de seguridad que ofrece IPsec

servicio	**protocolos**	
control de acceso	AH	ESP
integridad sin conexión	AH	ESP (opc)
autenticación origen de datos	AH	ESP (opc)
rechazo paquetes reenviados	AH	ESP
confidencialidad (cifrado)		ESP

Protocolos básicos

- **protocolo AH** (*authentication header*)
  ofrece autenticación del origen de los paquetes IP (cabecera+datos)
- **protocolo ESP** (*encapsulating security payload*)
  ofrece confidencialidad + autenticación de origen de paquetes IP (cabecera+datos)
- **gestión de claves**
  Autenticación+confidencialidad hacen uso de claves secretas
  1. configuradas manualmente en los nodos/router IPSec
  2. acordadas mediante protocolos de distribución de claves
     - implementaciones del framework ISAKMP (*Internet Security Association and Key Management Protocol*)
     - prot. OAKLEY: variante del método de intercambio de claves Diffie-Hellmann
     - prot. IKE/IKEv2: basado en certificados digitales y/o Diffie-Hellmann

Arquitectura IPSec

(a) Componentes (dependerá de como se emplee IPSec)

- **nodos extremos.** origen y destino final de paquetes IPSec en configuraciones IPSec punto a punto
- **nodos intermedios.** (también pasarelas seguras) routers/firewall intermedios con soporte IPSec

Nota. Tráfico IPSec puede cruzar nodos/routers intermedios no-IPSec

$\rightarrow$ es tratado como tráfico IP normal (sin realizar comprobaciones)

(b) Asociaciones de Seguridad (SA) ($\approx$ enlaces IPSec)
Enlaces "virtuales" (no necesariamente directos) entre 2 nodos que intercambian tráfico IPSec.

- Pueden ser: $\left\{ \begin{array}{l} \text{extremos finales de la comunicación } \textit{(SA extremo-extremo)} \\ \text{puntos intermedios de la comunicación } \textit{(SA con pasarela segura)} \end{array} \right.$
- Son enlaces unidireccionales: $\left\{ \begin{array}{l} \mathbf{A} \xrightarrow{SA_1} \mathbf{B} \\ \mathbf{A} \xleftarrow{SA_2} \mathbf{B} \end{array} \right\}$

- Cada nodo IPSec mantiene información sobre sus SA's
  - Base datos de políticas de seguridad (SPD: *security polices database*)
    Asocia el tráfico IPSec con los correspondientes SA's
  - Cada SA se identifica por $\left\{ \begin{array}{l} \text{la dirección IP de su destino} \\ \text{un n}^{\text{o}} \textbf{ SPI} \text{ único} \\ \text{indicador de parametros de seguridad} \end{array} \right\}$
  - Para cada SA almacenado en SPD se mantiene información sobre
    - algoritmos criptograficos usados (cifrado, hash)
    - claves de cifrado y autenticación
- Cada paquete IP que recibe el nodo, se analiza y se consulta el SPD para decidir
  - tratarlo como paquete IPSec (AH o ESP) y enviarlo al siguiente destino
  - tratarlo como paquete IP normal
  - descartarlo

Protocolo AH

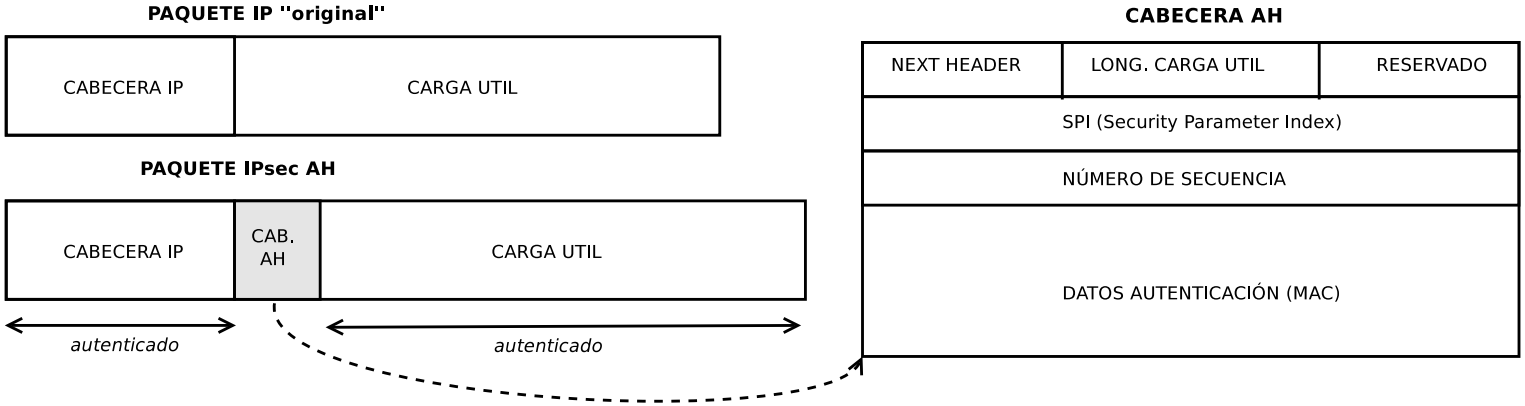
Soporta integridad+autentificación añadiendo al paquete IP un **código MAC** (*message authentication code*) basado en funciones HASH y claves secretas de autenticación.

NOTA: códigos MAC

- Mecanismo para preservar la integridad de los mensajes y garantizar la identidad del origen mediante la adición de un campo adicional (código MAC)
- Objetivos análogos a los de la *firma digital* (sin usar cifrado asimétrico)
- Esquema usual: HMAC (MAC con funciones HASH) [HMAC-MD5, HMAC-SHA1]
  - usa una *clave secreta de autenticación* K_s (compartida por origen y destino)
  - se aplica una función HASH al conjunto [MENSAJE+CLAVE_AUTENTICACIÓN]
  - el resumen resultante garantiza { integridad (HASH); origen (clave secreta) }

AH define una cabecera adicional donde se contiene información para: { autenticar del origen; asegurar integridad

Formato



Comportamiento de un nodo con un SA de tipo AH:

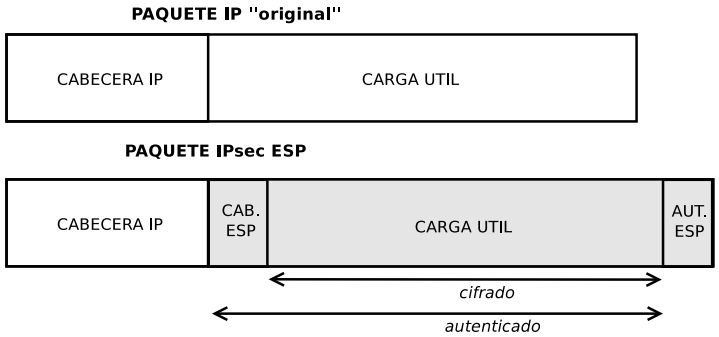
- Al recibir un paquete se comprueba su HMAC, empleando la K_s compartida (y opcionalmente el nº de secuencia)
  - Si es correcto: lo acepta y lo reenvía
    - al otro extremo de esa SA (en pasarelas IPSec)
    - a la capa de transporte (en un extremo final IPSec)
  - Si no: lo rechaza
- Sólo garantiza autenticación e integridad, no confidencialidad del tráfico

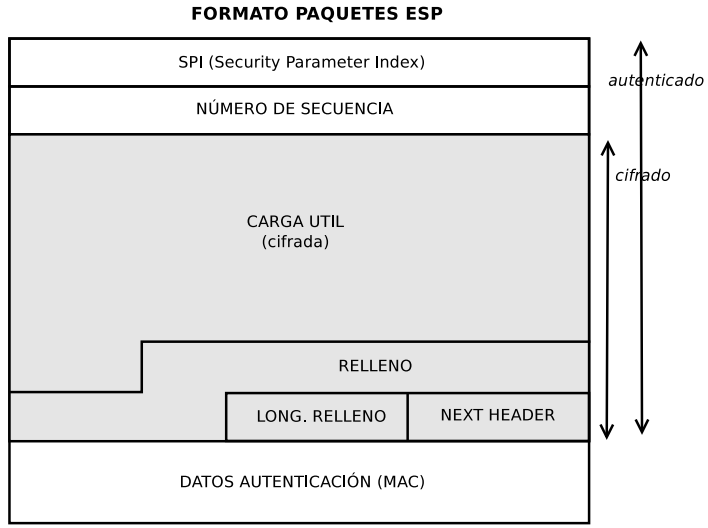
Protocolo ESP

Protocolo de encapsulado carga útil

- Nuevo formato de paquete con soporte para { confidencialidad; integridad/autenticidad (opc.) }
- En esa cabecera se incluye la carga util (opc. también la cabecera) del paquete IP original
- Usa cifrado simétrico + HMAC (opc.)

Formato paquetes ESP





2 tareas al procesar paquetes IPSec des protocolo ESP

1. verificar código HMAC de autenticación/integridad
2. descifrar los datos

Modos de uso IPSec

Existen 2 formas de usar AH y ESP

- en función de como se incluyan y manejen las cabeceras IP originales

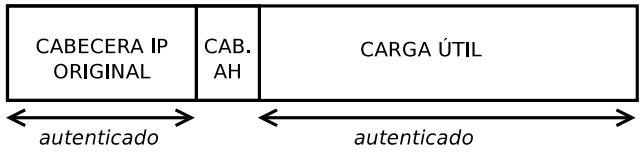
Modo transporte

- Protege **sólo** los datos del protocolo de nivel superior
- Las cabeceras AH ó ESP van a continuación de la cabecera IP original
  - forman parte de la carga útil del paquete IP
  - **Punto clave:** se mantiene la cabecera IP original
- Uso típico en SA extremo-extermo
- Los cálculos criptográficos se realizan **sólo** en los extremos de la comunicación.

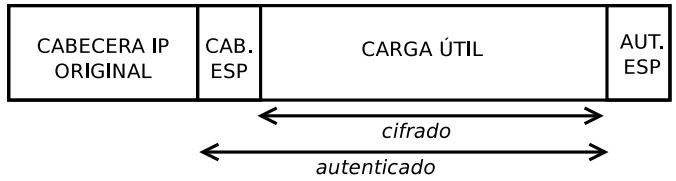
Modo túnel

- Protege el paquete IP original completo
- Paquete IP original se encapsula **completo** en un nuevo paquete IPSec con su **propia** cabecera IP
  - Las nuevas direcciones IP origen y destino serán las de los nodos inicial y final de la SA
  - La SA receptora *"deshace"* el paquete externo, realiza las compruebaciones y reenvía le paquete interno
- Se usa entre dos pasarelas IPSec
  - crea una ruta segura entre ellas
  - existe un *"tunel"* seguro dentro del cúal viajan los paquetes IP originales
- Permite interconectar de forma segura equipos no-IPSec mediante el establecimiento de pasarelas IPSec (implementadas sobre router o firewalls de borde)
  - estructura típica de las *redes virtuales privadas (VPN)*

PAQUETE IP "ORIGINAL"



AH en modo TRANSPORTE



ESP en modo TRANSPORTE

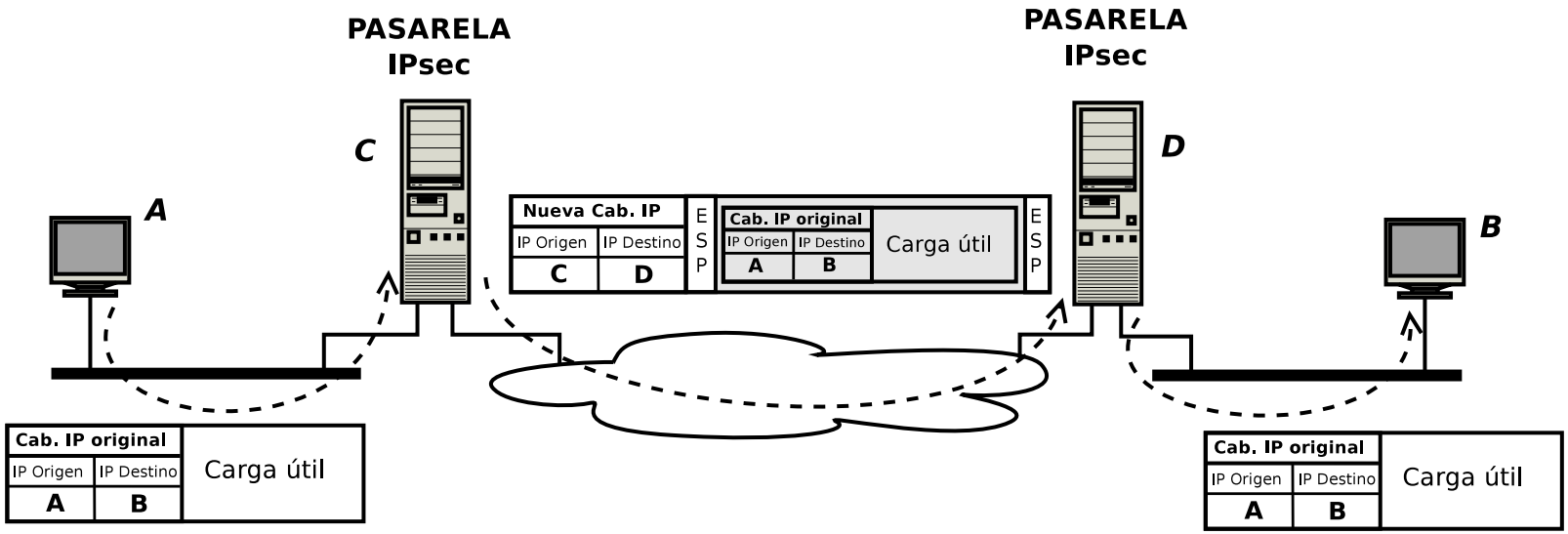
AH en modo TUNEL

ESP en modo TUNEL

Ejemplo

VPN basada en IPSec con ESP en modo túnel

(cifrado + autenticidad + integridad)



Características generales SSL/TLS

Objetivo inicial: proteger conex. web del protocolo HTTP

- permitir que cliente se asegure que está conectado a un servidor auténtico
- envio de datos confidencial (sólo servidor podrá ver los datos)

Protocolos SSL*(secure sockets layer)* y TLS*(transport layer security)*.

- SSL version 3 $\approx$ TLS version 1

Funciones de seguridad implantadas con una capa ”extra” en el nivel de transporte (a nivel de librerias)

- Nueva interfaz de acceso a servicios de transporte basada en el interfaz de *”sockets”*
- Reemplaza funciones de *sockets* (*connect, accept, send, recv*) por otras seguras (*SSL_connect, SSL_accept, SSL_send, SSL_recv*).
- Cualquier aplicación que use *sockets* puede adaptarse para usar SSL

Servicios de seguridad en SSL/TLS

1. **Confidencialidad.** Tráfico normal: intercambio de mensajes cifrados con claves simétricas
   - Al inicio de la sesión/conexión cliente y servidor acuerdan las claves que usarán
   - Dos claves, una por sentido: cliente $\rightarrow$ servidor, servidor $\rightarrow$ cliente
   - Diálogo inicial: mecanismo seguro de intercambio de claves basado en criptografía asimétrica (certif. digitales)
2. **Autenticación de entidades.** Cliente puede verificar la identidad del servidor mediante un mecanismo basado en firmas digitales + certificados digitales.
   - Exige conocer (y aceptar/confiar) la clave pública del servidor
     - mediante el empleo de *certificados digitales*
   - Esquema análogo para la autenticación del cliente frente al servidor (si es requerida)

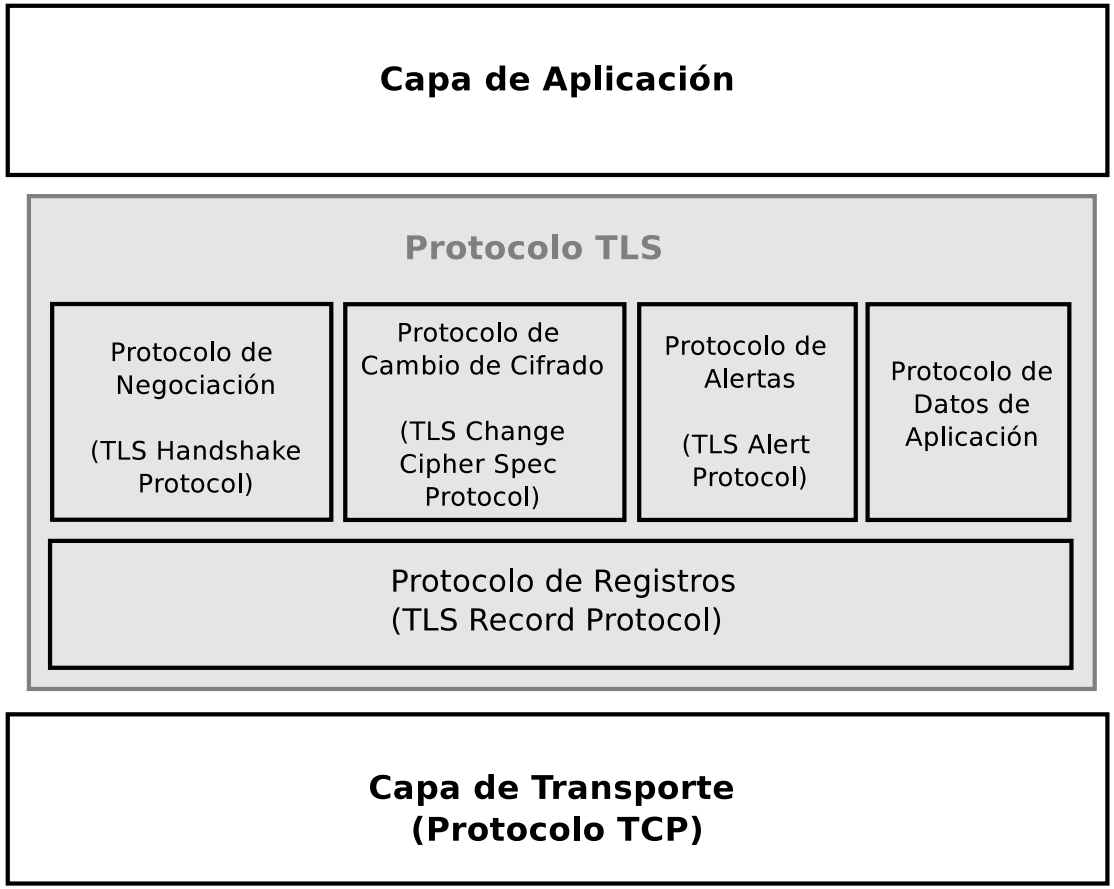
3. **Autenticación de mensajes.** Los paquetes SSL, además de ir cifrados, incluyen códigos HMAC para garantizar integridad y autenticidad
   - Dos claves secretas (una por sentido) acordadas al iniciar sesión/conexión

Consideraciones adicionales

- **Eficiencia y coste computacional.**
  - Minimiza uso de cifrado asimétrico (sólo al establecer sesión/conexión)
  - Evita repetir autenticación+intercambio de claves, permitiendo reutilizar parámetros negociados al iniciar la sesión entre varias conexiones.
  - Contempla la posibilidad de emplear algoritmos de compresión para reducir tamaño de mensajes
- **Extensibilidad.** Esquema abierto a distintos algoritmos (cifrado, compresión, intercambio claves, etc)
  - Los algoritmos a usar se negocian al establecer la sesión
    - en función de las capacidad del cliente y el servidor
    - uso de combinaciones predefinidas
  - Posibilidad de añadir nuevos algoritmos (garantiza compatibilidad hacia atrás)

Arquitectura SSL/TLS (I)

Capa de transporte seguro estructurada en 2 niveles.



Arquitectura SSL/TLS (II)

nivel superior. Tareas
- negociación parámetros de sesión/conexión
- transferencia de datos del nivel de aplicación
- funciones de control

nivel inferior. Ofrece servicios de intercambio seguro de mensajes a las tareas del nivel superior
- mensajes estructurados en *registro SSL*
- sobre ellos aplica compresión, autenticación y/o compresión

Protocolos

protocolo de registros SSL/TLS. Crea los mensajes protegidos (cifrados+autenticados) en el origen y los recupera y comprueba en el destino.

protocolo de negociación SSL/TLS. Establece de forma segura los parámetros de la sesión/conexión (combinaciones de algoritmos a usar + claves)

Sesiones y conexiones

Sesión. Asociación entre un cliente y un servidor (grupo de conexiones)

- Creadas por el *protocolo de negociación*
- Definen un conjunto de parámetros de seguridad (que se utilizarán en distintas conexiones)
- Cada sesión tiene asociado:
  - Identificador de sesión
  - Certificado de la entidad par (opc.) [certif. X.509 del otro extremo]
  - Método de compresión
  - Especificación del cifrado [combinación: cifrador simétrico + algoritmo HMAC + tamaño clave]
  - Clave maestra (*master-key*) [48 bytes compartidos por cliente y servidor]
  - Reanudable: SI/NO [indicación de si se pueden reutilizar loos parámetros]

Conexión. Conexión individual de nivel de transporte entre cliente y servidor

- Son transitorias y están asociadas a una sesión.
- Cada conexión tiene asociado:
  - Valores aleatorios compartidos por cliente y servidor
  - 2 pares de claves secretas (generadas a partir de *Clave Maestra* de sesión)
    - Clave secreta para HMAC de escritura del servidor
    - Clave secreta para HMAC de escritura del cliente
    - Clave secreta para cifrado del servidor
    - Clave secreta para cifrado del cliente
  - Vector de inicialización (VI) [creado a partir de la Clave Maestra]
  - Números de secuencia de los mensajes enviados y recibidos

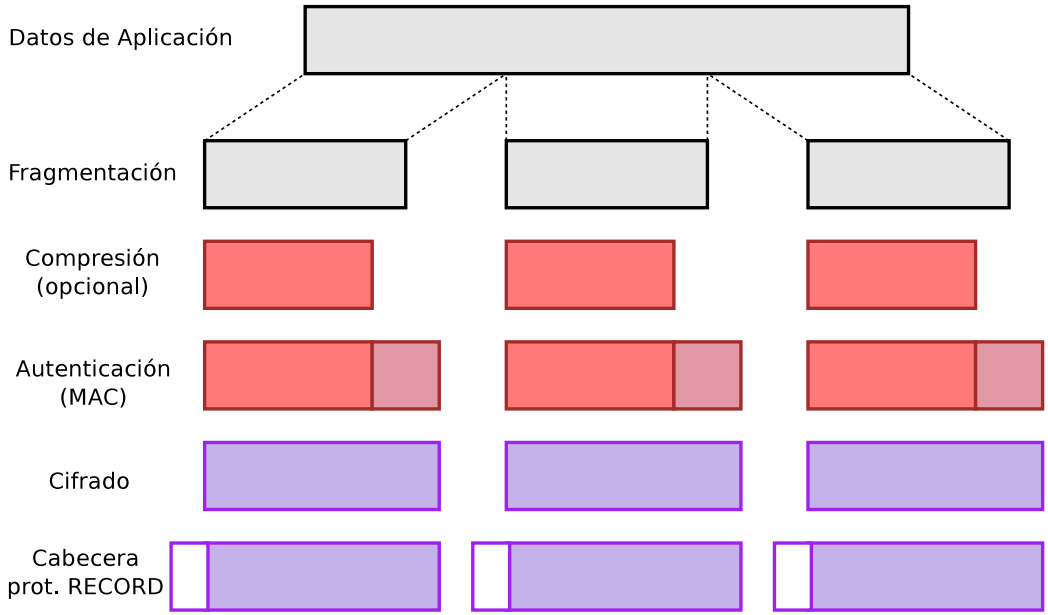
Protocolos SSL/TLS

(a) Protocolo de registros (RECORD)

Proporciona servicios de seguridad a niveles superirores

- Confidencialidad: mediate cifrado simétrico
- Integridad: mediante HMAC y claves secretas

Funcionamiento:



(b) Protocolo de negociación (HANDSHAKE)

Finalidad:

1. Autenticar el servidor ante el cliente (opc. al revés)
2. Acordar algoritmos: cifrado, HMAC, compresión
3. Acordar claves (*clave maestra*) de forma segura

Empleado antes de la transmisión de cualquier dato del nivel de aplicación.

Formato de mensajes intercambiados entre cliente y servidor (usa protocolo RECORD).

Esquema de la negociación:

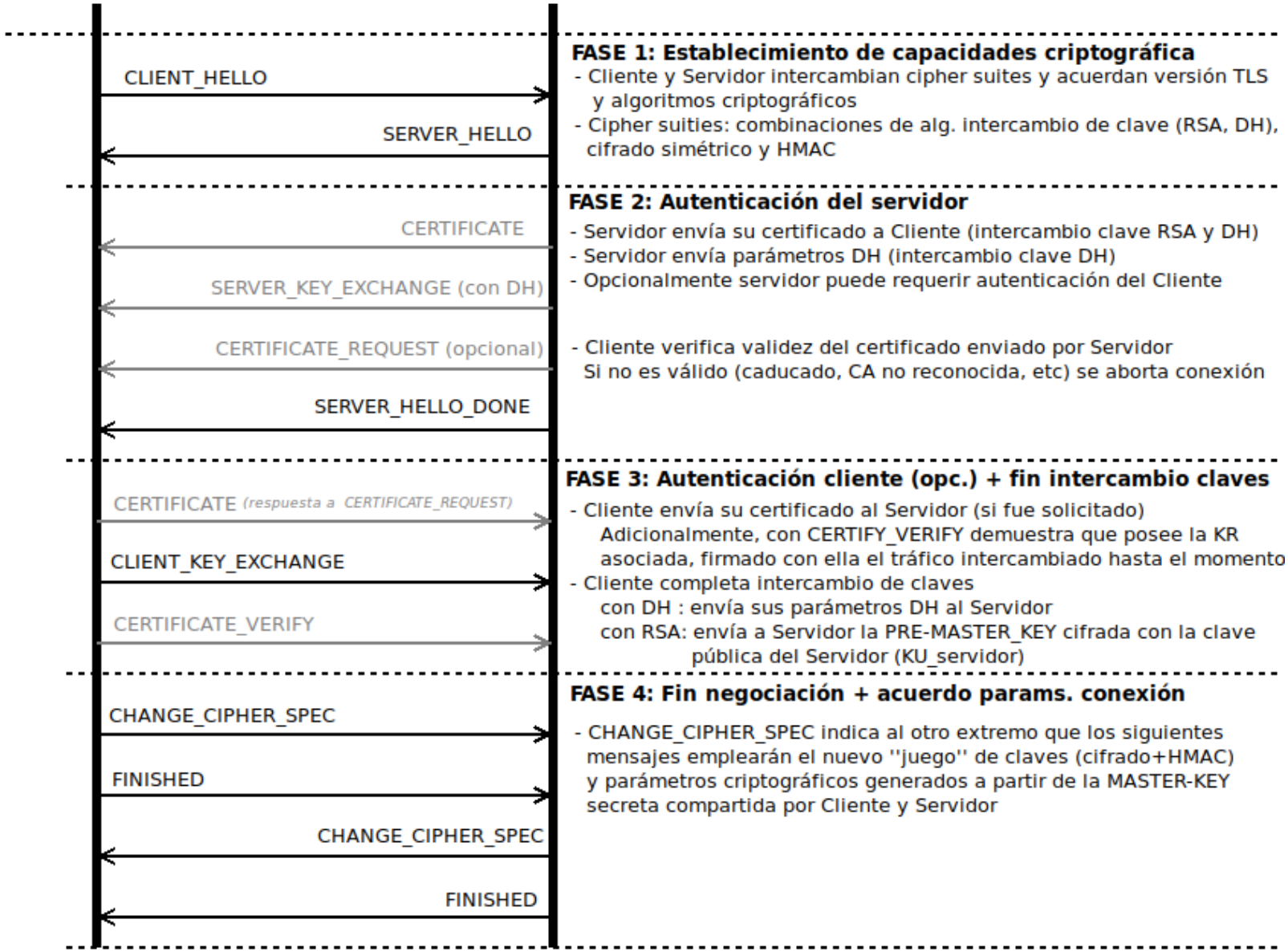
Fase 1. Establecer capacidades de cada extremo (combinaciones de algoritmos admitidos)

Fase 2. Autenticación del servidor (petición+envio certif. digital ó algorito de intercambio clave [RSA, diffie-Hellman])

Fase 3. (Opcional) Autenticación del cliente (petición+envio certif. digital ó algoritmo de intercambio clave)

Fase 4. Intercambio/acuerdo de parámetros concretos de la conexión

Mensajes prot. negociación TLS



Seguridad en nivel de APLICACIÓN: SSH

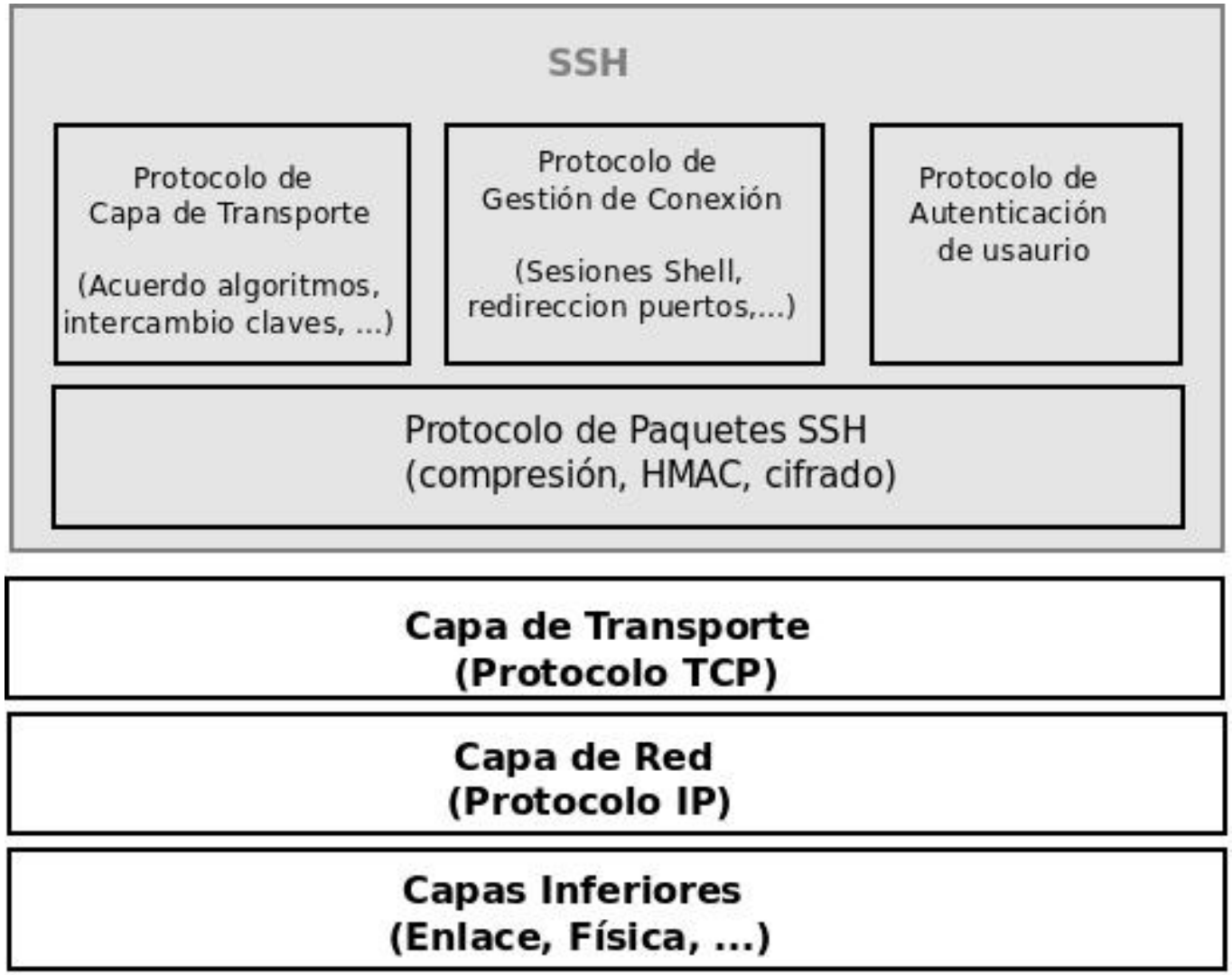
SSH (*Secure Shell*): reemplazo de aplicaciones acceso remoto `telnet` y `rlogin`

- No proporcionaban confidencialidad (tráfico en claro)
- SSH sigue una aproximación similar a TLS (combina cifrado asimétrico, simétrico y HMAC)

Servicios de seguridad en SSH

- Confidencialidad: cifrado simétrico con claves acordadas al inicio de conexión
- Autenticación e integridad de mensajes: HMAC con claves acordadas al inicio de conexión
- Autenticación de entidades: con cifrado asimétrico
- Autenticación de usuarios: diversos métodos (contraseña, cifrado asimétrico)
- Otros:
  - Eficiencia y extensibilidad
  - Redirección de tráfico sobre conexiones SSH

Arquitectura SSH



Protocolos SSH (I)

Protocolo de paquetes SSH: análogo a protocolo RECORD en SSL/TLS

- Compresión datos + autenticación/integridad + cifrado

Protocolo de Capa de Transporte: análogo a protocolo HANDSHAKE en SSL/TLS

- Establecimiento de conexión + acuerdo de algoritmos
- Autenticación del servidor
- Intercambio de claves

Pasos:

1. Negociación versión protocolo SSH
   - versión 1 [obsoleta] ó versión 2
   - deben coincidir en cliente y servidor
   - servidor decide su aborta conexión
2. Acuerdo de algoritmos
   - cliente y servidor proponen listas de algoritmos (inter. clave, cifrado simétrico, HMAC, compresión) en orden de preferencia
   - se selecciona la primera coincidencia
3. Intercambio de claves
   - SSH2 sólo prevé uso de Diffie-Helman
4. Mensaje `NEWKEYS` marca fin de negociación
   - siguiente mensaje usará nuevo juego de claves secretas
   - se regeneran las claves secretas cada hora de conexión o cada GB transferido

Protocolos SSH (II)

Protocolo autenticación usuario

- Intercambio de mensajes `USERAUTH_REQUEST`
- Disponibles distintos esquemas

Autentic. nula: servidor permite conexión de cq. usuario sin autenticarlo
Autentic. con lista de acceso: como autenticiación nula pero restringido a una lista de direcciones IP de equipos cliente autorizados

- vulnerable a falsificación dir. IP (*IP spoofing*)
- no soportado por SSH2

Autentic. con lista de acceso + autenticación cliente: incluye autenticación del equipo cliente mediante cifrado asimétrico

- cliente presenta su clave pública [KU] (en un certificado)
- debe firmar un mensaje con su clave privada [KR] para demostrar que la posee

Autentic. basada en contraseña: usuario debe proporcionar una contraseña

- servidor SSH delega en los sistemas de autenticación de usuario del sistema (proceso `login`, etc)

Autentic. basada en clave pública de usuario: (acceso SSH ”sin contraseña”)

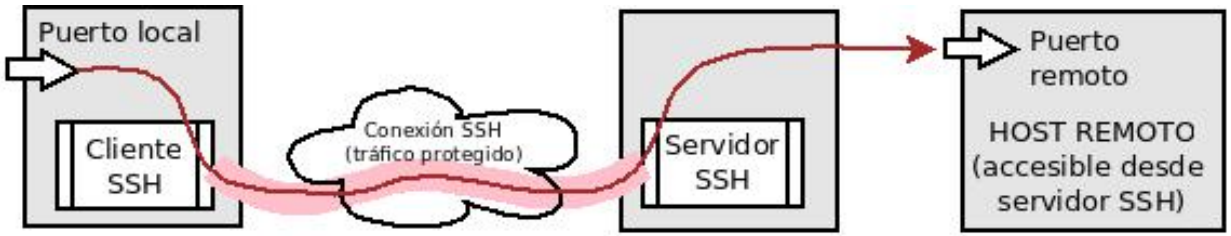
- cada usuario puede tener sus propios pares KU + KR (generados con `ssh-keygen`)
- usuario ”registra” su KU en los servidores donde pretende acceder (fichero `$HOME/.ssh/authorized_keys`)
- proceso cliente presenta la KU del usuario y demuestra que posee la respectiva KR firmando un mensaje con ella

Protocolos SSH (III)

Protocolo gestión de conexión

- Gestiona la sesión del shell y solicita la regeneración de claves secretas cuando corresponda ($>$ 1 hora ó $>$ 1 GB)
- Redirección de sesiones X11 sobre conexiones SSH (opción `-X` en OpenSSH)
- Redirección de puertos arbitrarios sobre conexiones SSH ($\approx$ tunel SSH)

Redirección local




```
ssh -l <usuario>  -L <puerto_local>:host:<puerto_remoto> <servidor>
```

Redirecciona a traves de la conexión SSH el tráfico de un puerto local del equipo donde se ejecuta el cliente SSH hacia un puerto "remoto" de un host accesible desde el servidor SSH

Redirección remota (redirección inversa)

```
ssh -l <usuario>  -R <puerto_servidor>:host:<puerto_destino> <servidor>
```

Redirecciona a traves de la conexión SSH el tráfico de un puerto del equipo donde se ejecuta el servidor SSH hacia un puerto de un host accesible desde el cliente SSH