Ciberseguridad en la nube

Seguridad en redes

Área de Ingeniería Telemática

Álvaro Alesanco

Departamento de Ingeniería Electrónica y Comunicaciones

Universidad Zaragoza

Protocolos Básicos de Seguridad y VPNs
TLS, IPSEC y SSH

Área de Ingeniería Telemática

Álvaro Alesanco

Departamento de
Ingeniería Electrónica
y Comunicaciones

Universidad Zaragoza

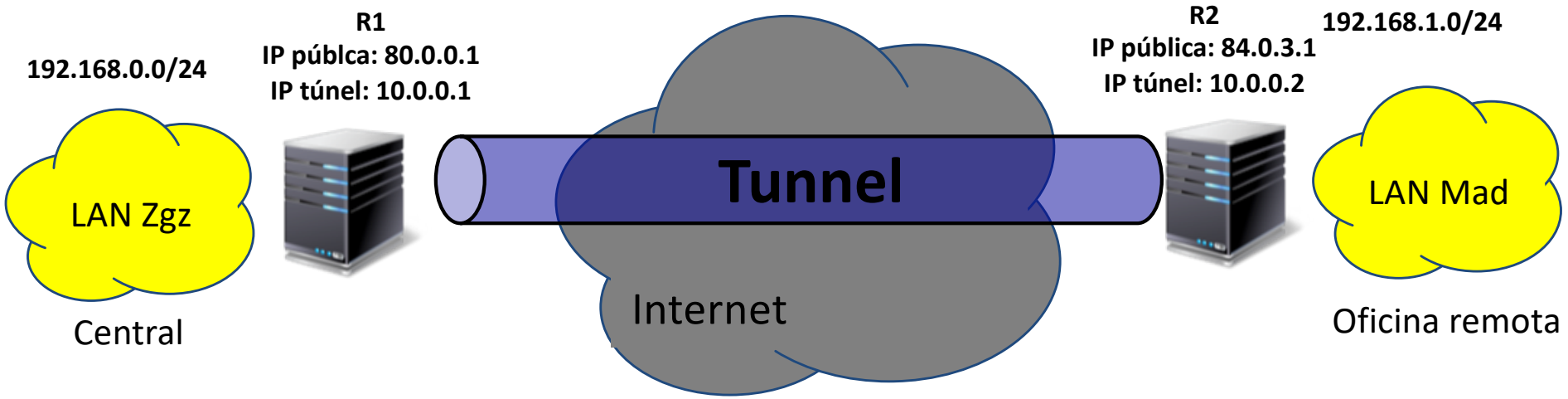
Virtual Private Network

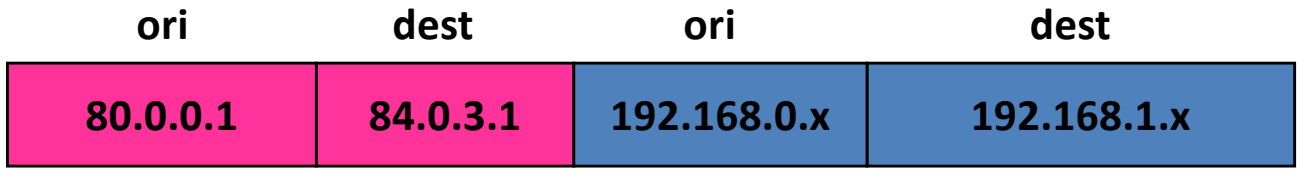
- A virtual private network (VPN) extends a private network across a public network, such as the Internet.
- It enables a computer to send and receive data across shared or public networks as if it is directly connected to the private network, while benefiting from the functionality, security and management policies of the private network.
- A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryptions.

https://en.wikipedia.org/wiki/Virtual_private_network

Virtual Private Network

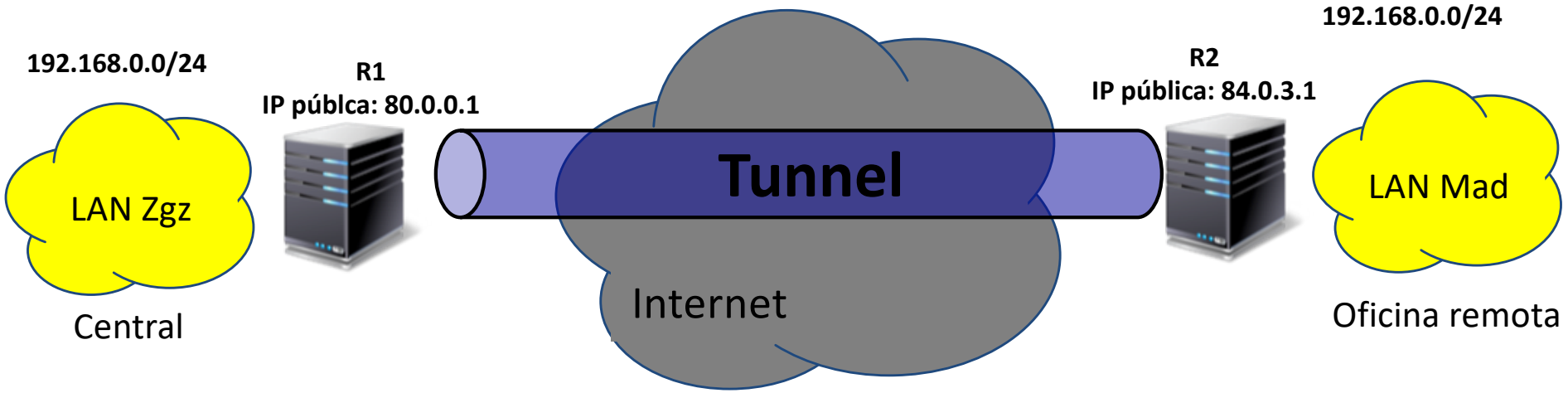
- Escenario típico 1a: túnel IP entre 2 redes privadas

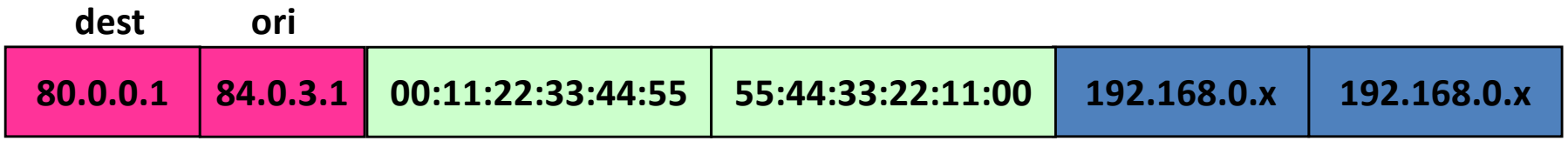




Virtual Private Network

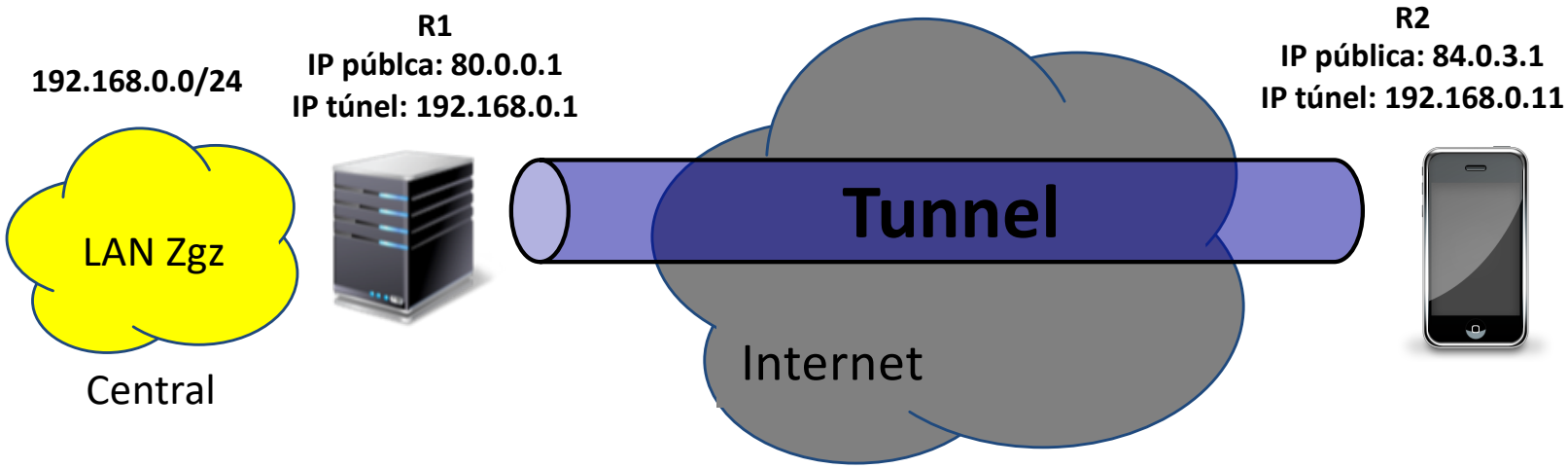
- Escenario típico 1b: túnel TAP entre 2 localizaciones de 1 red privada





Virtual Private Network

- Escenario típico 2a: Roadwarrior con túnel IP

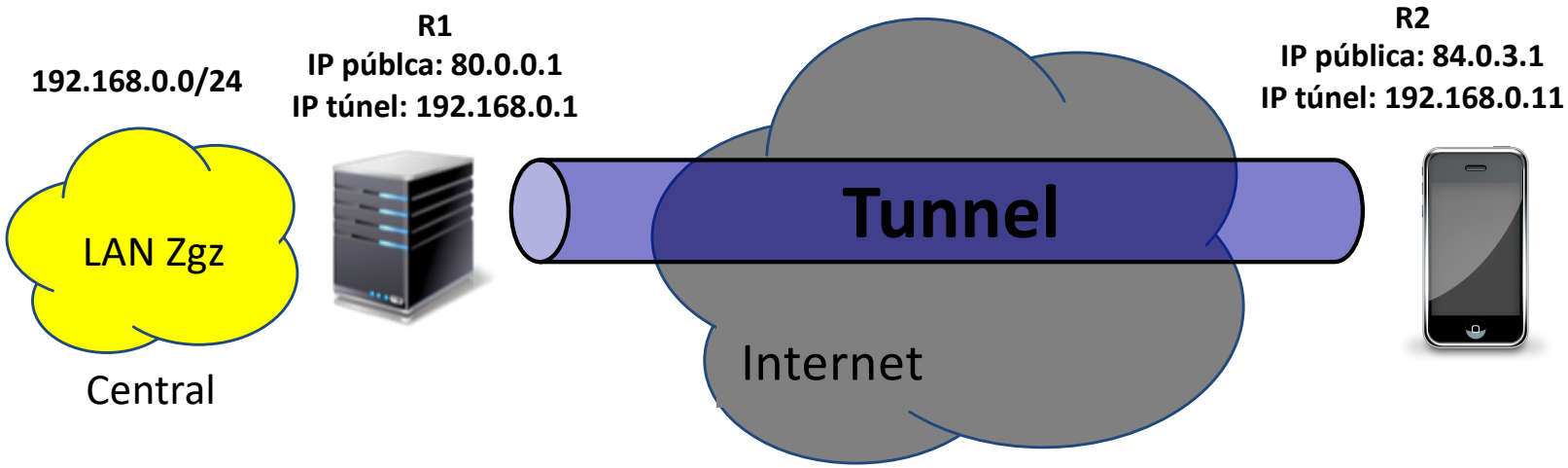


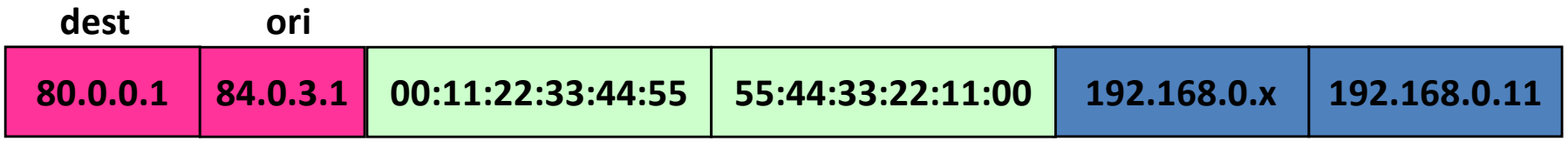
ori	dest	ori	dest
80.0.3.1	84.0.0.1	192.168.0.11	192.168.0.x

ori	dest	ori	dest
80.0.0.1	84.0.3.1	192.168.0.x	192.168.0.11

Virtual Private Network

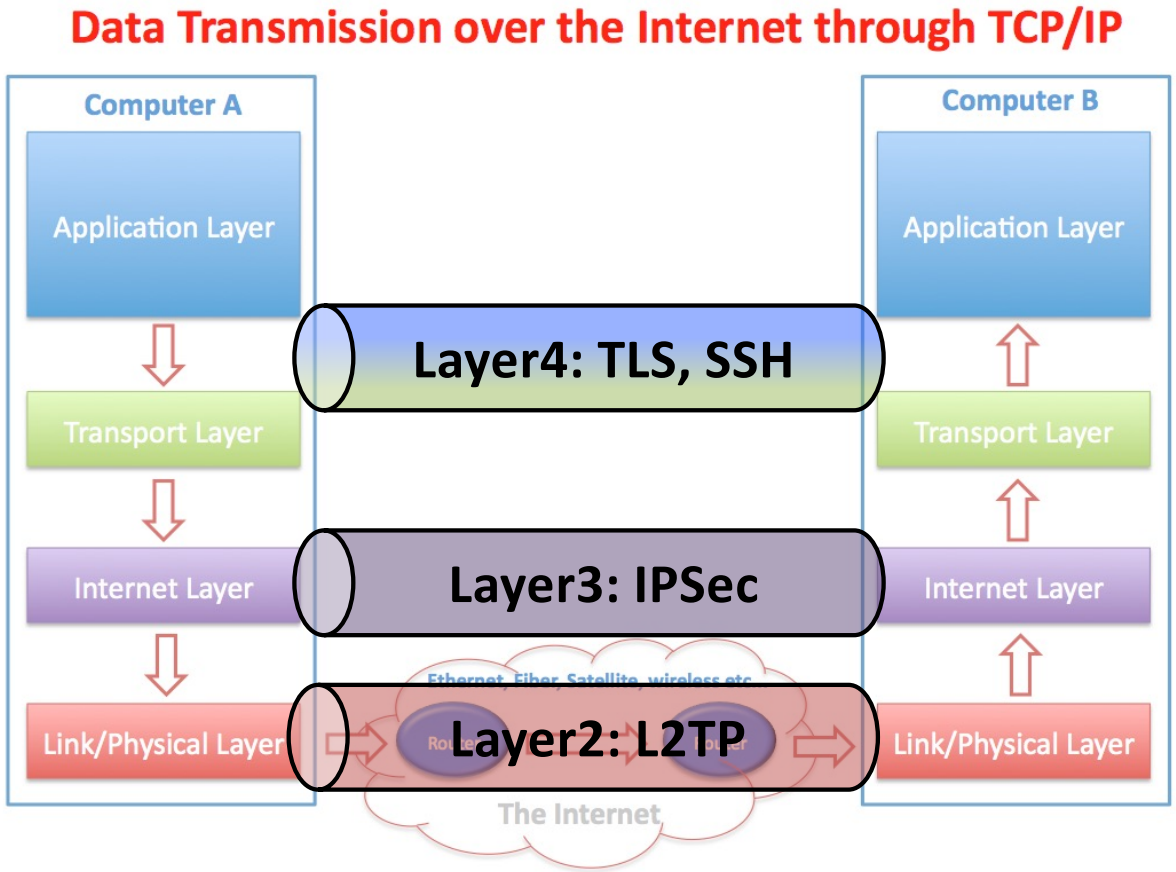
- Escenario típico 2b: Roadwarrior con túnel TAP





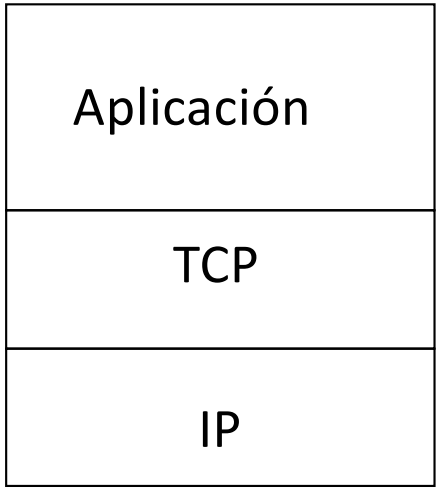
Virtual Private Network

- ¿Cómo se cifra la información? Niveles

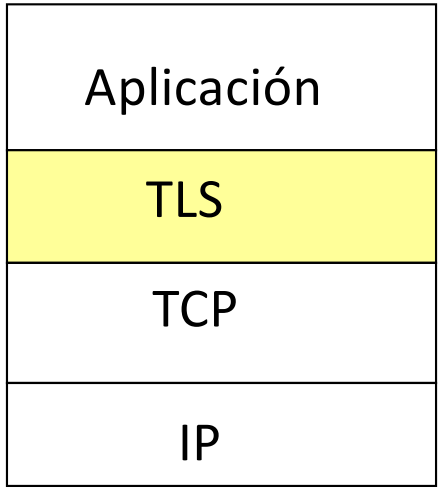


Layer4: TLS/SSL (1.3 RFC 8446)

- Proporciona
  - Confidencialidad
  - Integridad
  - Autenticación



Aplicación normal



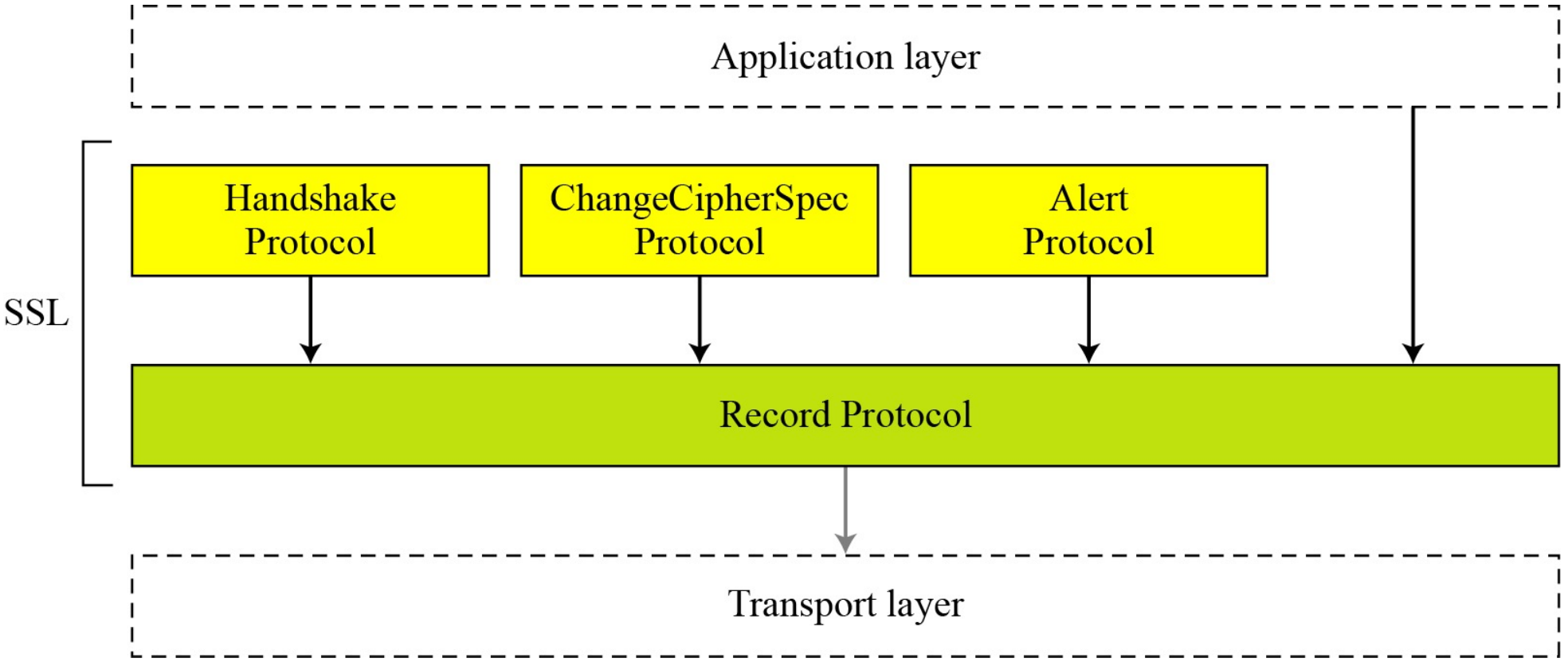
Aplicación con TLS

Layer4: TLS/SSL

- Overview del funcionamiento
  - Handshake: A y B usan sus certificados y claves privadas para autenticarse el uno al otro (no siempre) e intercambiar el secreto compartido.
  - Calculo de claves: A y B usan el secreto compartido para derivar un conjunto de claves.
  - Transferencia de datos: Los datos que van a ser transferidos son divididos en una serie de registros.
  - Cierre de conexión: Se envían mensajes especiales para cerrar de forma segura la conexión.

Layer4: TLS/SSL

- TLS --> 4 protocolos en 2 niveles



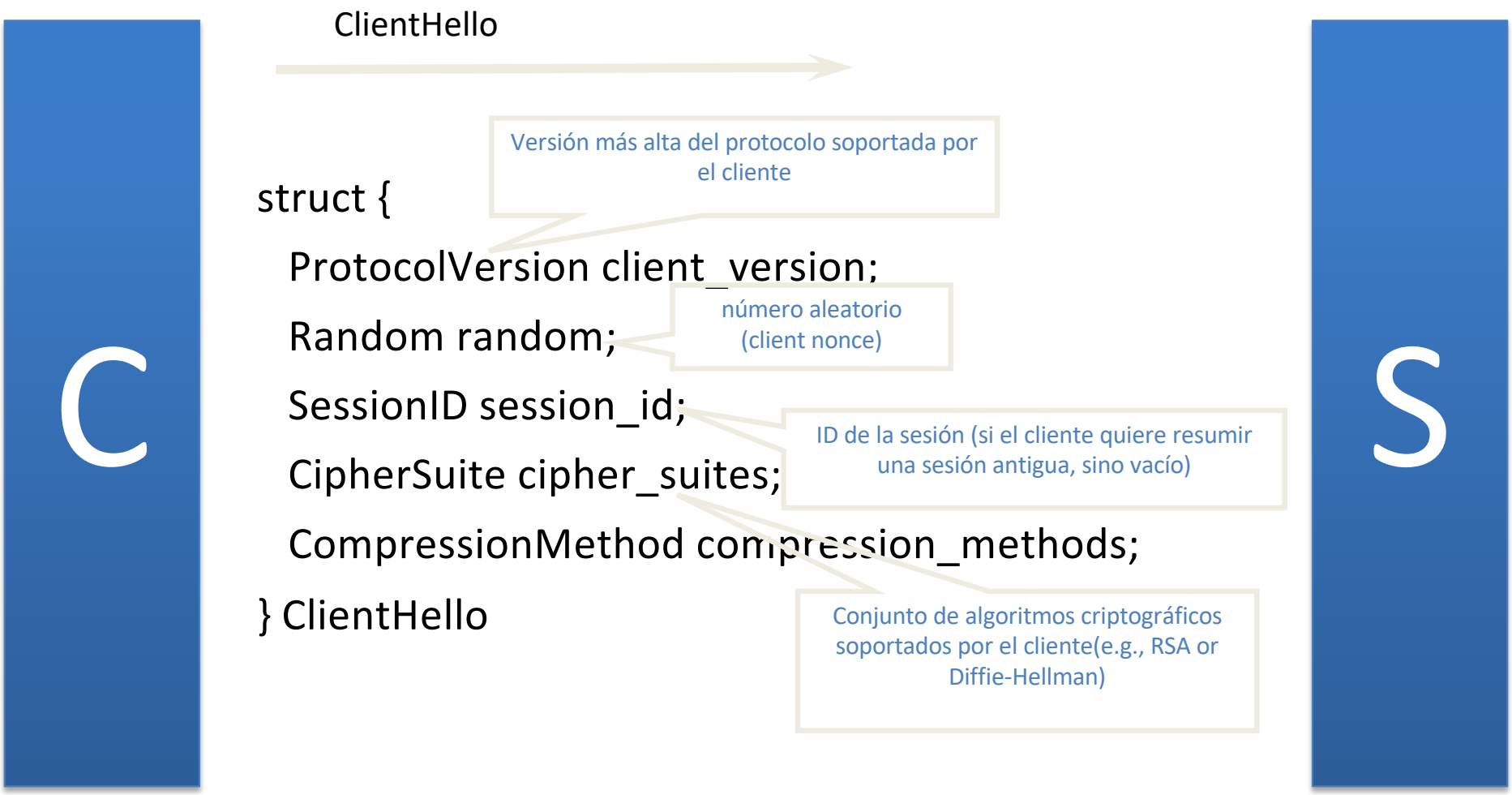
Layer4: TLS/SSL

- Handshake Protocol: propósito

- Autenticación del servidor
- Negociado: ponerse de acuerdo en algoritmos de cifrado
- Establecer claves
- Autenticación del cliente (opcional)

Layer4: TLS/SSL

- Handshake Protocol: fase 1 (RFC 2246)



Layer4: TLS/SSL

- Cipher Suite
  - Método para el intercambio de claves
  - Algoritmo de firma
  - Algoritmo de cifrado de datos
  - Algoritmo para el cálculo del Hash

[SSL/TLS]_[key exchange]_[signature algorithm]_WITH_[block cipher]_[authentication hash]

Layer4: TLS/SSL

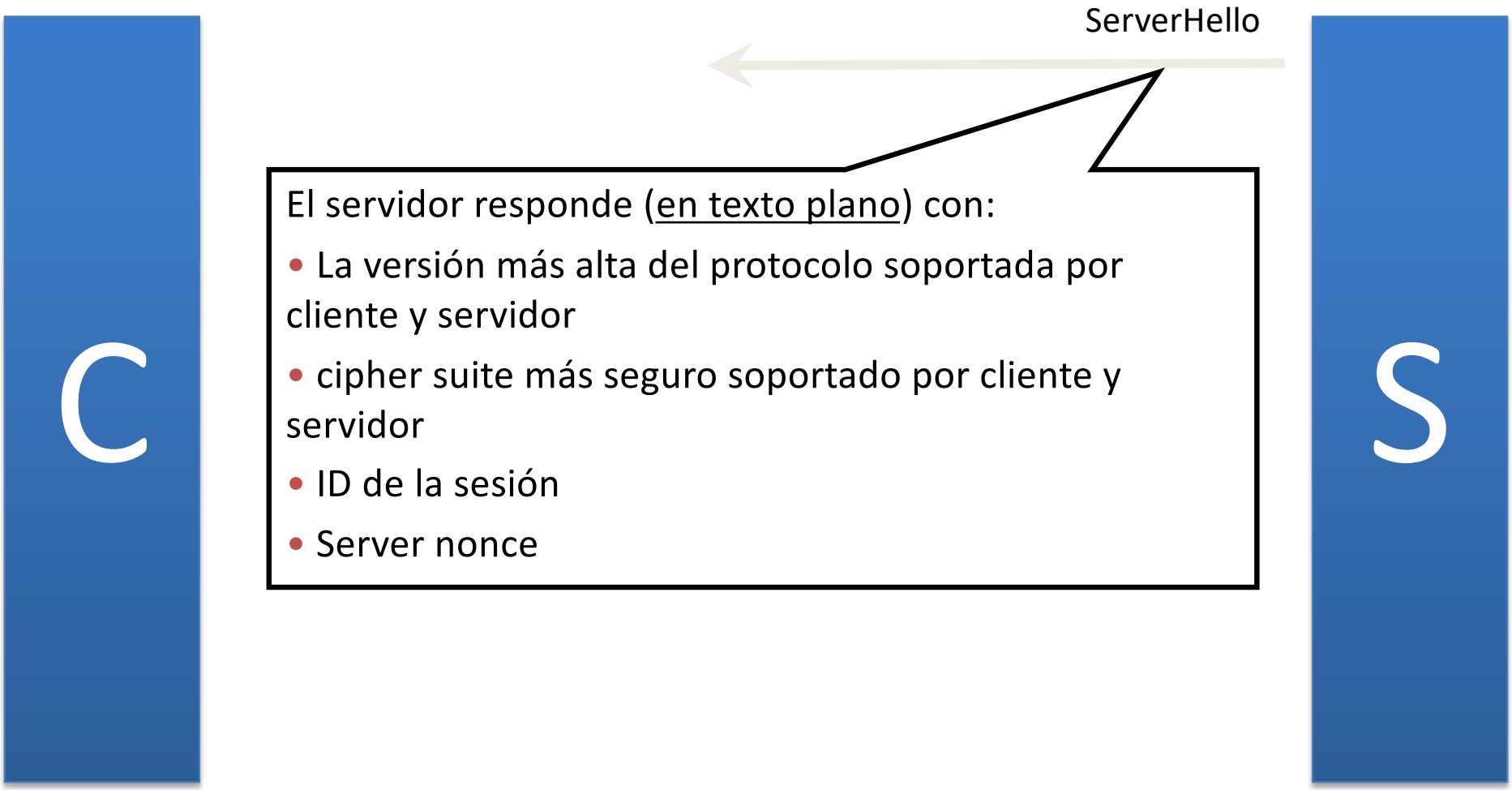
- Cipher Suite
  - Método para el intercambio de claves
    - None: No hay intercambio. A y B saben de antemano el pre-master secret
    - RSA
    - Diffie-Hellman / DH con curvas elípticas
      - Anonymous
      - Ephemeral
      - Fixed
  - Algoritmo de firma
    - RSA
    - DSS
    - ECDSA

Layer4: TLS/SSL

- Cipher Suite
  - Algoritmo para el cálculo del Hash
    - None: A y B declinan a usar un Hash
    - MD5: 128 bits
    - SHA: 160 o 256 bits
  - Algoritmo de cifrado de datos
    - 3DES
    - AES

Layer4: TLS/SSL

- Handshake Protocol: fase 2a



Layer4: TLS/SSL

- Handshake Protocol: fase 2b

C

S

ServerCertificate

- public-key certificate (clave pública RSA o DH en función del cipher suite)

Issued To

Common Name (CN)	mail.google.com
Organization (O)	Google Inc
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	54:F5:75:0C:BF:3D:E7:5E

Issued By

Common Name (CN)	Google Internet Authority G2
Organization (O)	Google Inc
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

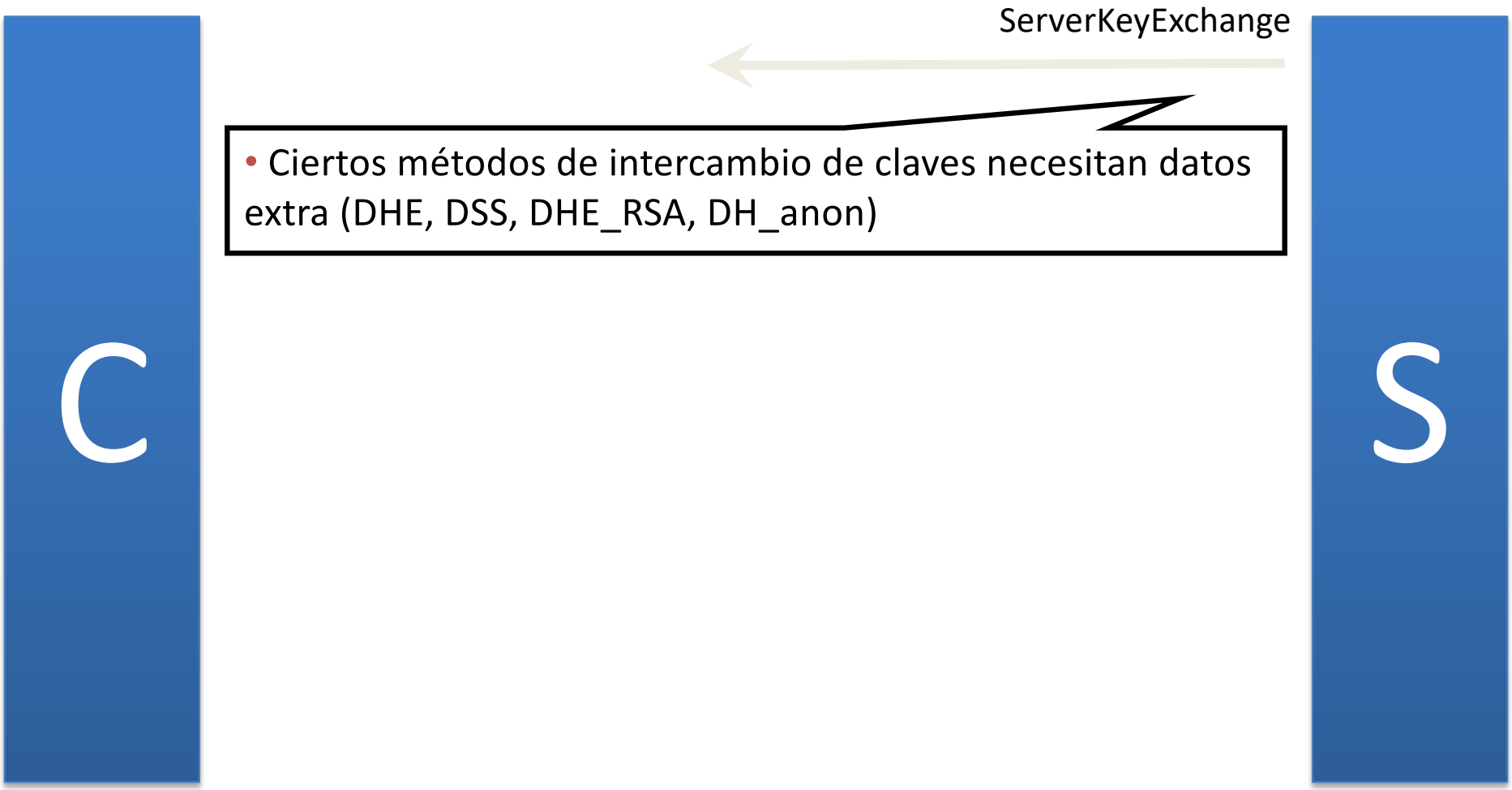
Issued On	07/05/14
Expires On	05/08/14

Fingerprints

SHA1 Fingerprint	9E:32:87:26:CA:A4:5F:D0:6C:91:30:24:F0:01:09:21:6C:E7:5D:98
MD5 Fingerprint	6B:11:D3:4E:ED:24:6A:E2:5E:23:70:9A:3C:FA:5C:EB

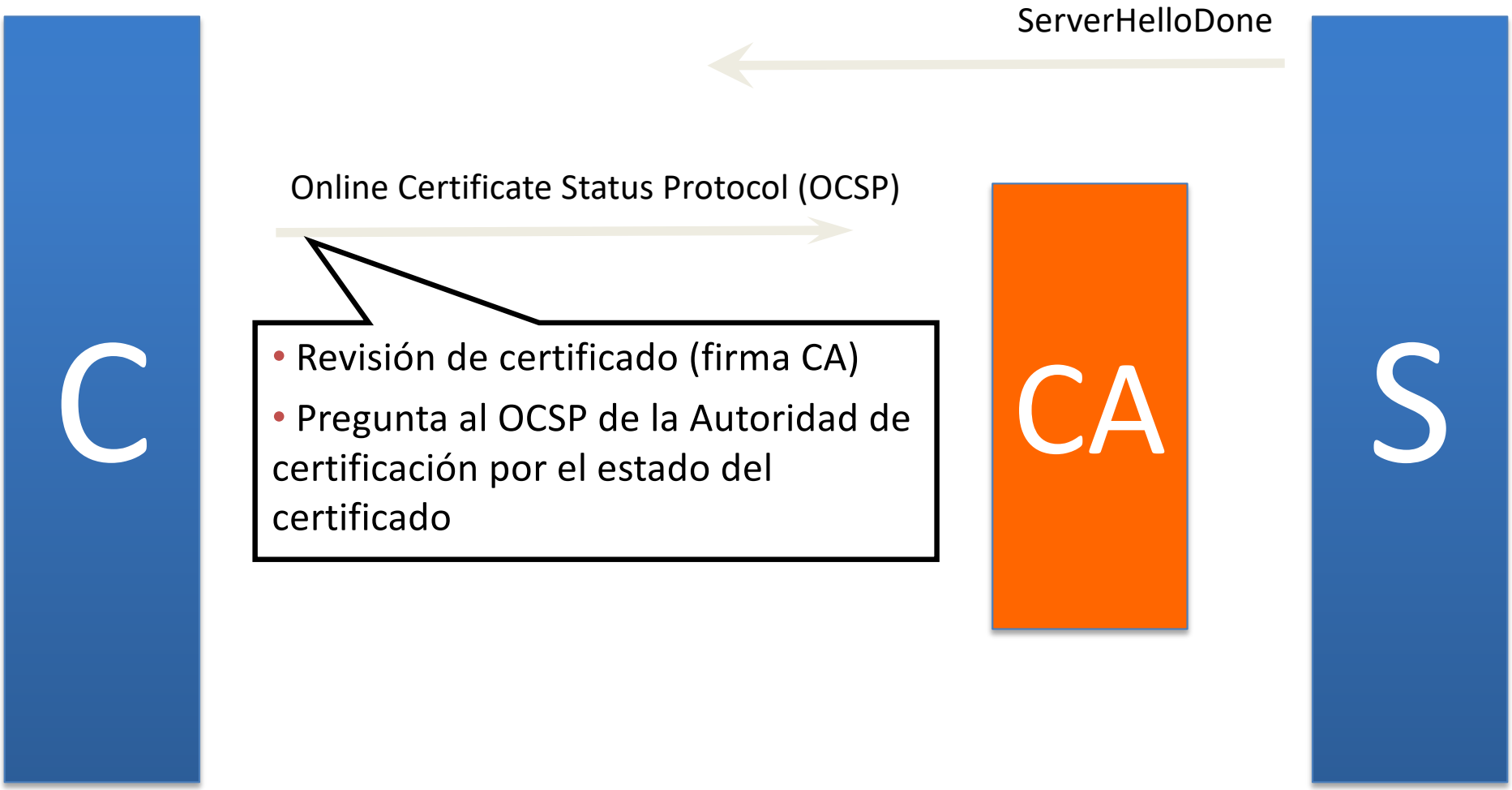
Layer4: TLS/SSL

- Handshake Protocol: fase 2b



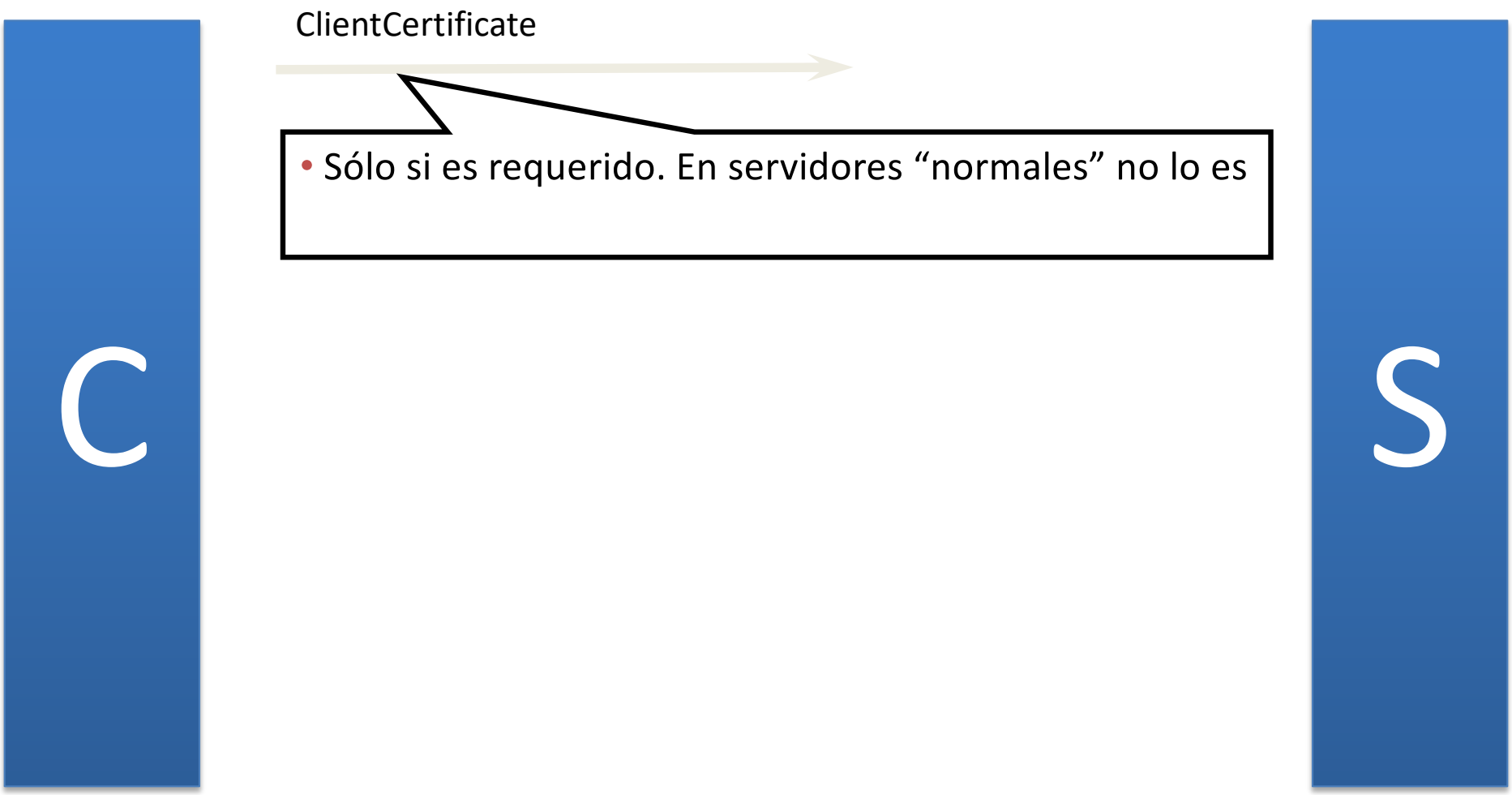
Layer4: TLS/SSL

- Handshake Protocol: fase 2c



Layer4: TLS/SSL

- Handshake Protocol: fase 3a



Layer4: TLS/SSL

- Handshake Protocol: fase 3b

C

ClientKeyExchange

```
struct {
        select (KeyExchangeAlgorithm) {
           case rsa:
              EncryptedPreMasterSecret;
           case dhe_dss:
           case dhe_rsa:
           case dh_dss:
           case dh_rsa:
           case dh_anon:
              ClientDiffieHellmanPublic;
        } exchange_keys;
     } ClientKeyExchange;
```

genera *pre_master_sectret (48 bytes aleatorios generados de forma segura*,)

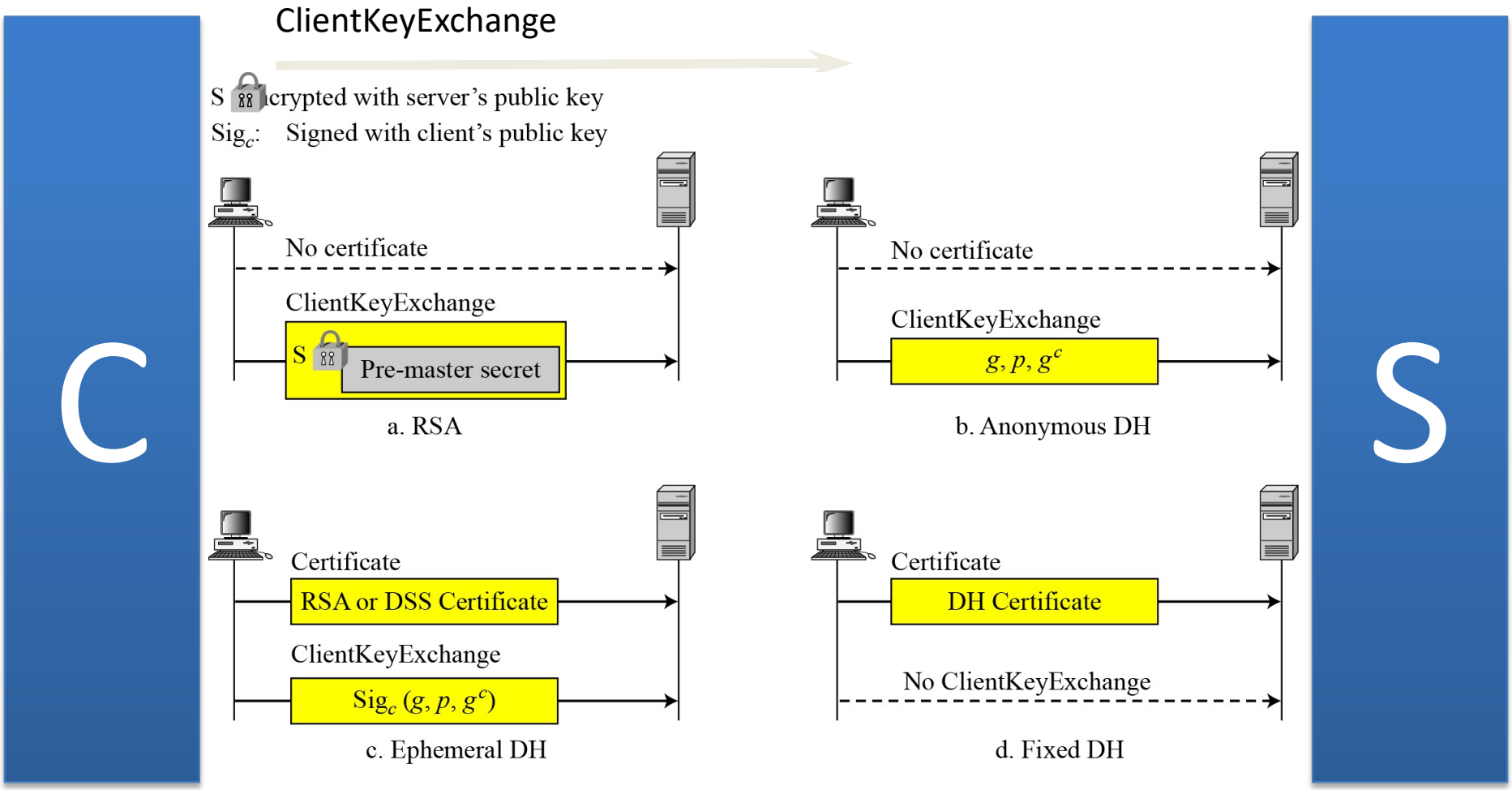
cifra con la clave pública del servidor (EncryptedPreMasterSecret)

envía al servidor

S

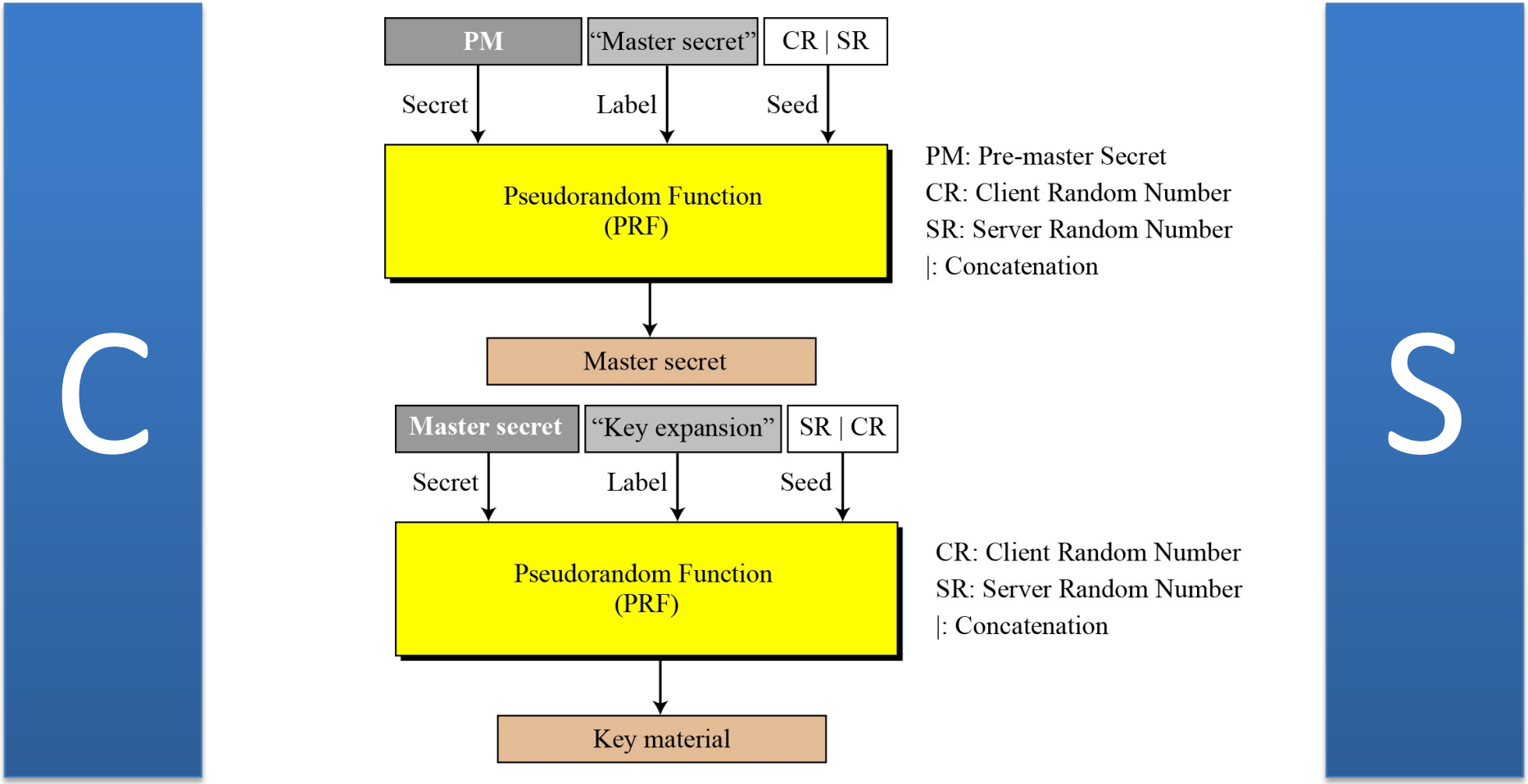
Layer4: TLS/SSL

- Handshake Protocol: fase 3b (otras opciones)



Layer4: TLS/SSL

- Handshake Protocol: fase 4 (cálculo de claves)



Layer4: TLS/SSL

- Handshake Protocol: fase 4 (6 claves)

C

Key material se divide en 6 claves

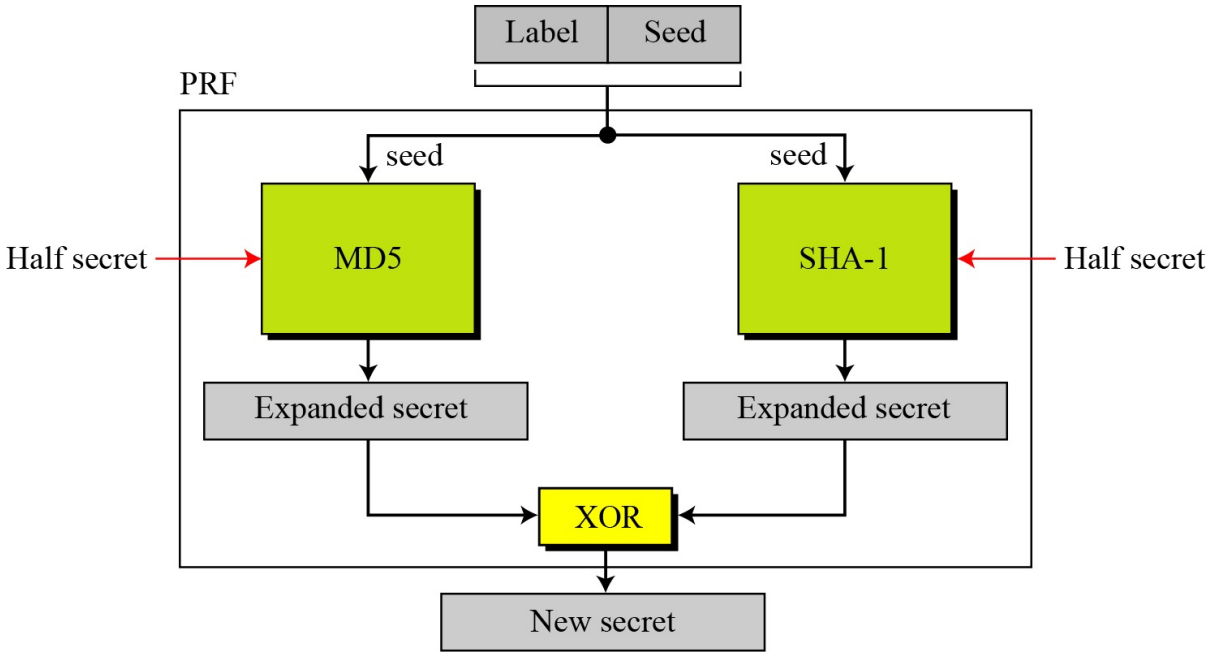
- clave cliente para MAC (client write MAC key)
- clave servidor para MAC (server write MAC key)
- clave de cifrado del cliente (client write encryption key)
- clave de cifrado del servidor (server write encryption key)
- vector de inicialización del cliente (IV) (client write IV)
- vector de inicialización del servidor (IV) (server write IV)

S

Layer4: TLS/SSL

- Handshake Protocol: fase 4 (PRF)

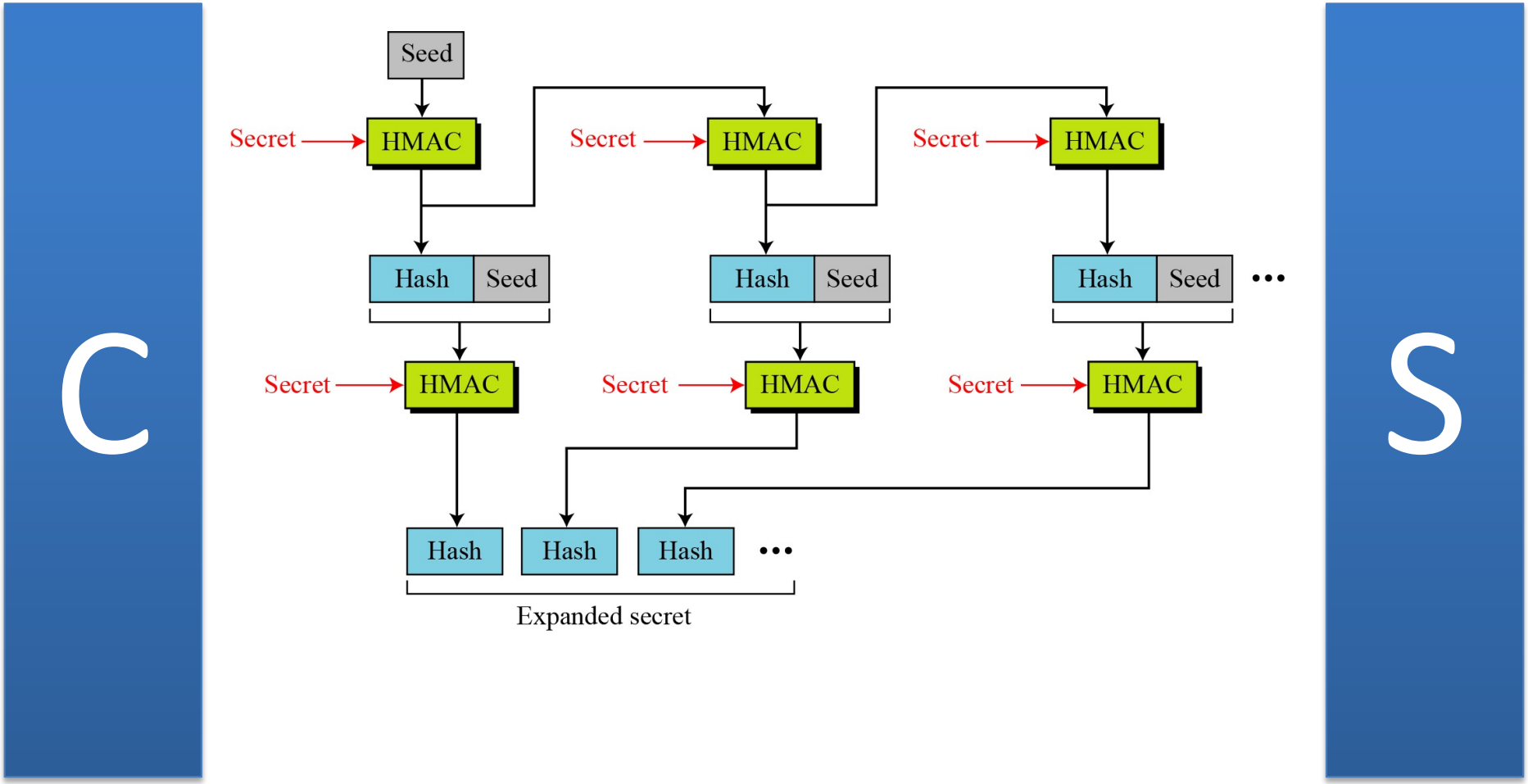
C



S

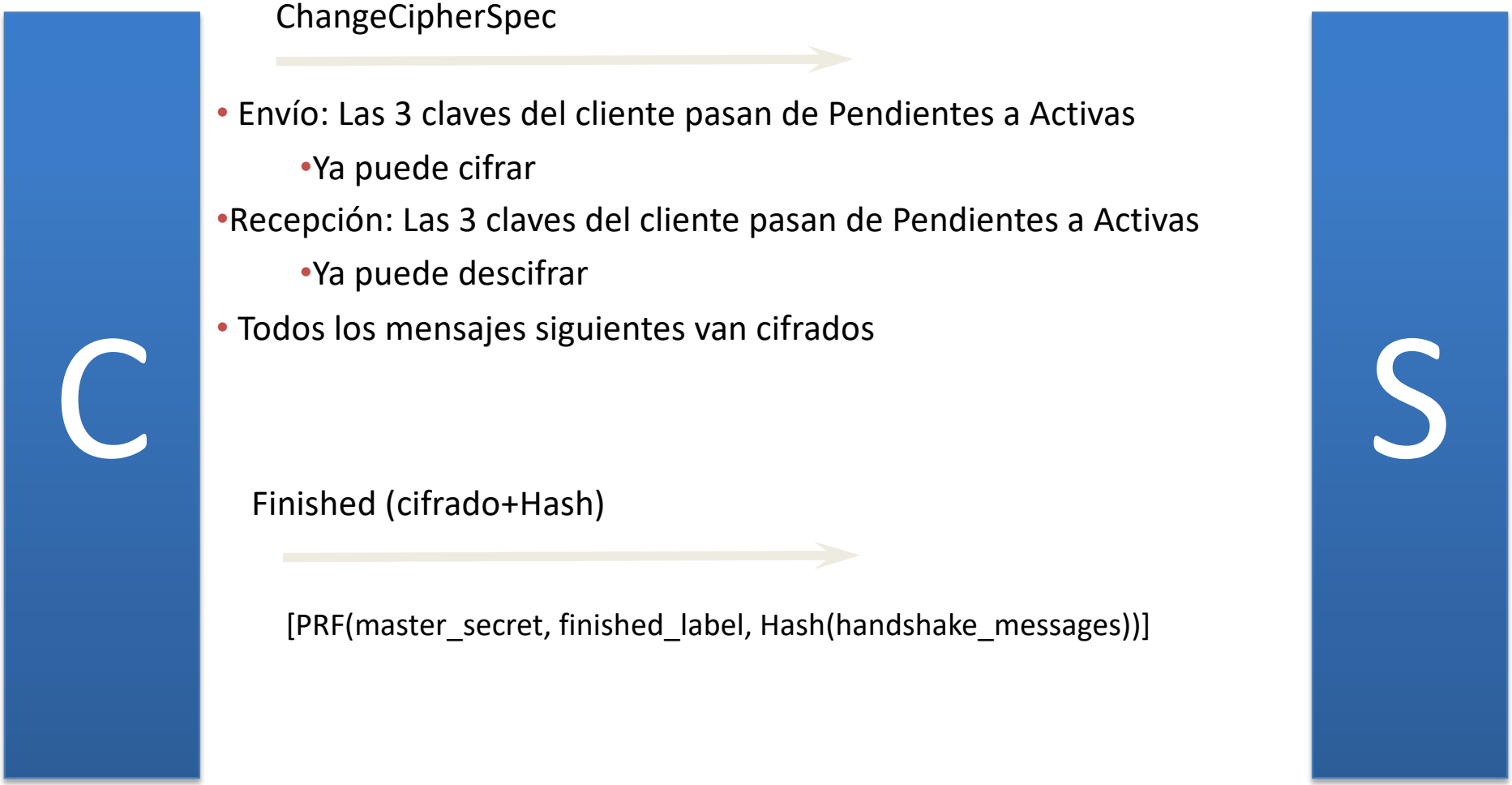
Layer4: TLS/SSL

- Handshake Protocol: fase 4 (Data expansion function)



Layer4: TLS/SSL

- ChangeCipherSpec Protocol: fase 1 (cliente)
- Handshake Protocol: fase 5 (final cliente)



Layer4: TLS/SSL

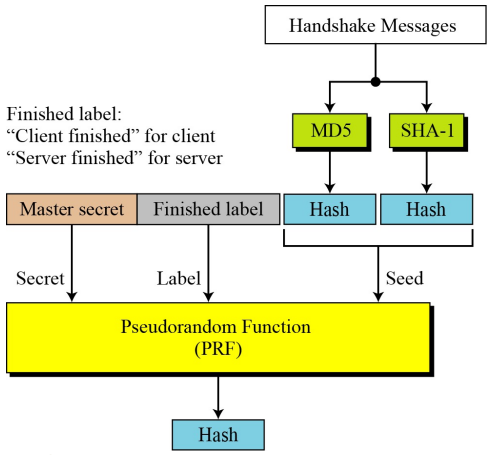
- ChangeCipherSpec Protocol: fase 2 (servidor)
- Handshake Protocol: fase 5 (final servidor)

C

S

ChangeCipherSpec

- Envío: Las 3 claves del servidor pasan de Pendientes a Activas
  - Ya puede cifrar
- Recepción: Las 3 claves del servidor pasan de Pendientes a Activas
  - Ya puede descifrar
- Todos los mensajes siguientes van cifrados

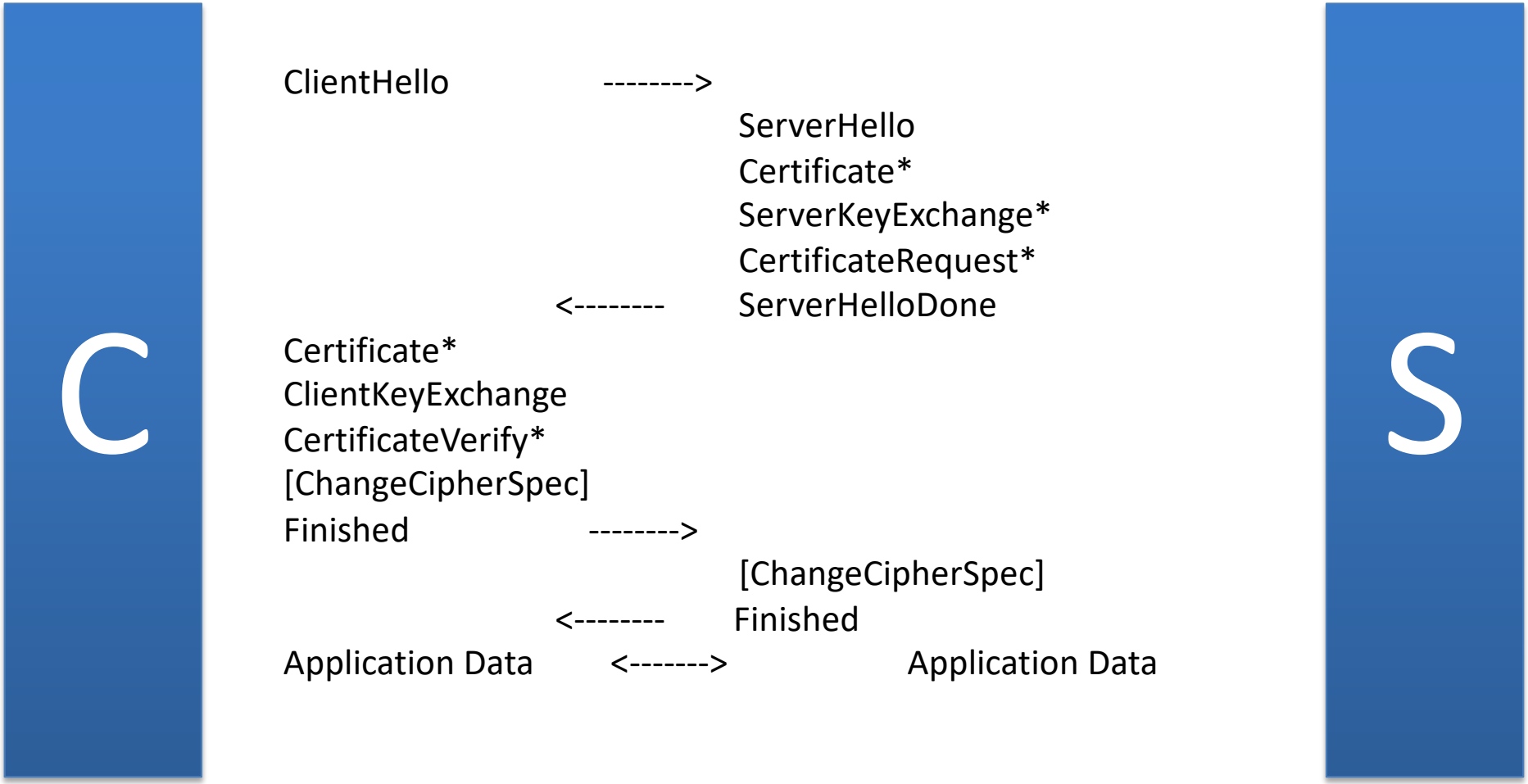


Finished (cifrado + Hash)

Datos cifrados

Layer4: TLS/SSL

- Resumen



Layer4: TLS/SSL

- Transmisión de información: Registros (https://tools.ietf.org/html/rfc5246#page-19)
  - Toda la información de los 3 protocolos, TLS Handshaking Protocols (Handshake, ChangeCipherSpec y Alert) y los datos se encapsula en mensajes del Record Protocol


```
struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[TLSPlaintext.length];
    } TLSPlaintext;
```


```
struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[TLSCompressed.length];
    } TLSCompressed;
```

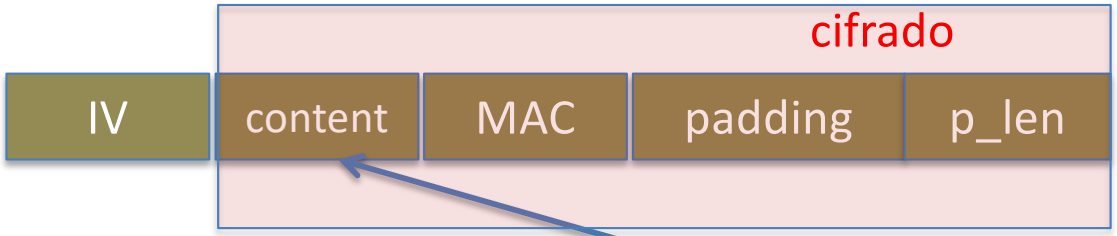

```
enum {
        change_cipher_spec(20), alert(21),
        handshake(22), application_data(23), (255)
    } ContentType;

struct {
        uint8 major;
        uint8 minor;
    } ProtocolVersion;
```

```
struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        select (SecurityParameters.cipher_type) {
            case stream: GenericStreamCipher;
            case block:  GenericBlockCipher;
            case aead:   GenericAEADCipher;
        } fragment;
    } TLSCiphertext;
```

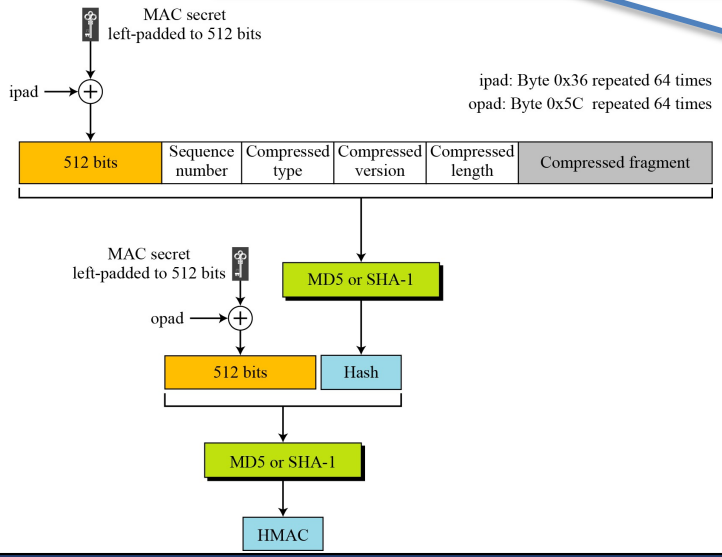
Layer4: TLS/SSL

- Transmisión de información: Registros (https://tools.ietf.org/html/rfc5246#page-19)
  - TLSCiphertext




```
struct {
   opaque IV[SecurityParameters.record_iv_length];
   block-ciphered struct {
      opaque content[TLSCompressed.length];
      opaque MAC[SecurityParameters.mac_length];
      uint8 padding[GenericBlockCipher.padding_length];
      uint8 padding_length;
   };
} GenericBlockCipher;
```

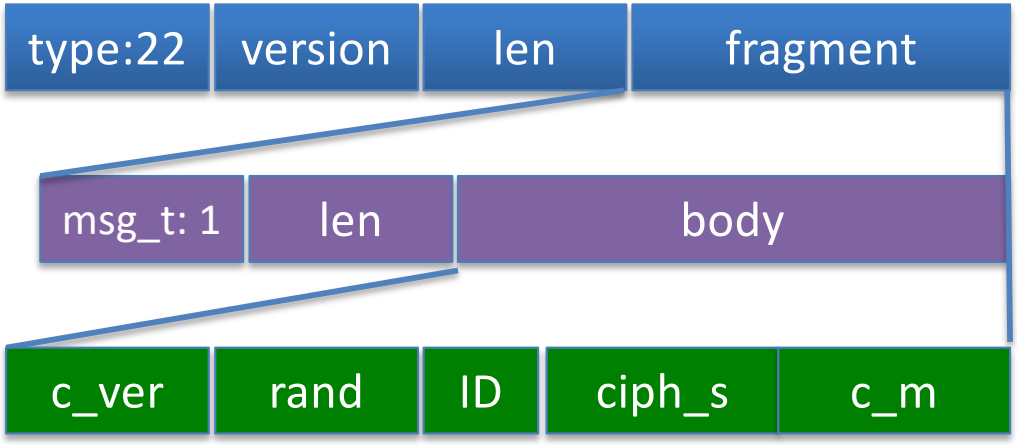
content --> fragment compressed




```
MAC(MAC_write_key, seq_num +
                   TLSCompressed.type +
                   TLSCompressed.version +
                   TLSCompressed.length +
                   TLSCompressed.fragment);
```

Layer4: TLS/SSL

- Registros del Handshake (TLSPlaintext)



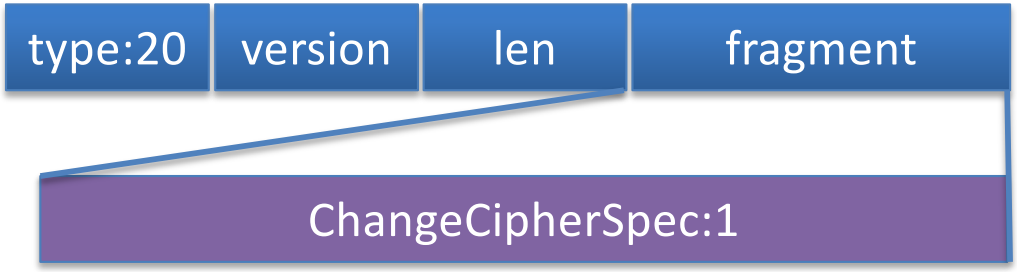

```
enum {
      hello_request(0), client_hello(1), server_hello(2),
      certificate(11), server_key_exchange (12),
      certificate_request(13), server_hello_done(14),
      certificate_verify(15), client_key_exchange(16),
      finished(20), (255)
   } HandshakeType;

   struct {
      HandshakeType msg_type;    /* handshake type */
      uint24 length;             /* bytes in message */
      select (HandshakeType) {
         case hello_request:       HelloRequest;
         case client_hello:        ClientHello;
         case server_hello:        ServerHello;
         case certificate:         Certificate;
         case server_key_exchange: ServerKeyExchange;
         case certificate_request: CertificateRequest;
         case server_hello_done:   ServerHelloDone;
         case certificate_verify:  CertificateVerify;
         case client_key_exchange: ClientKeyExchange;
         case finished:            Finished; /* único cifrado*/
      } body;
   } Handshake;
```

```
struct {
      ProtocolVersion client_version;
      Random random;
      SessionID session_id;
      CipherSuite cipher_suites<2..2^16-2>;
      CompressionMethod compression_methods<1..2^8-1>;
      select (extensions_present) {
         case false:
            struct {};
         case true:
            Extension extensions<0..2^16-1>;
      };
   } ClientHello;
```

Layer4: TLS/SSL

- Registro de ChangeCipherSpec (TLSCiphertext)

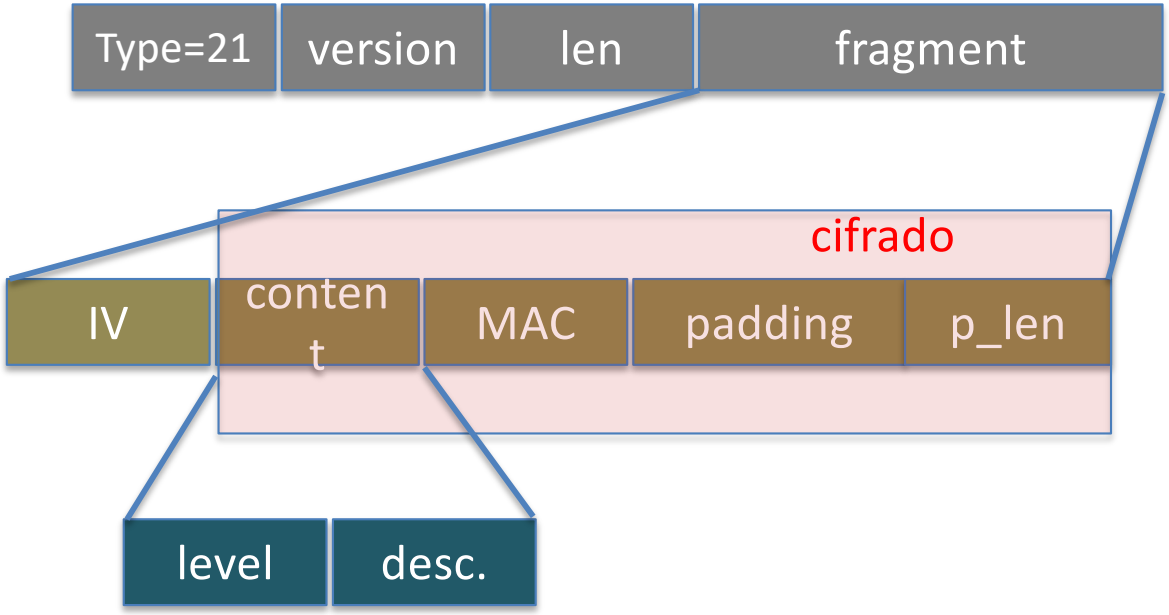



```
struct {
        enum { change_cipher_spec(1), (255) } type;
    } ChangeCipherSpec;
```

- Nota: se cifra con las claves activas (no pendientes) por lo que normalmente va sin cifrar (al inicio de la conexión no hay claves activas y este es, precisamente, el mensaje que hace que las claves pase a activas)

Layer4: TLS/SSL

- Registro de Alert (TLSCiphertext)



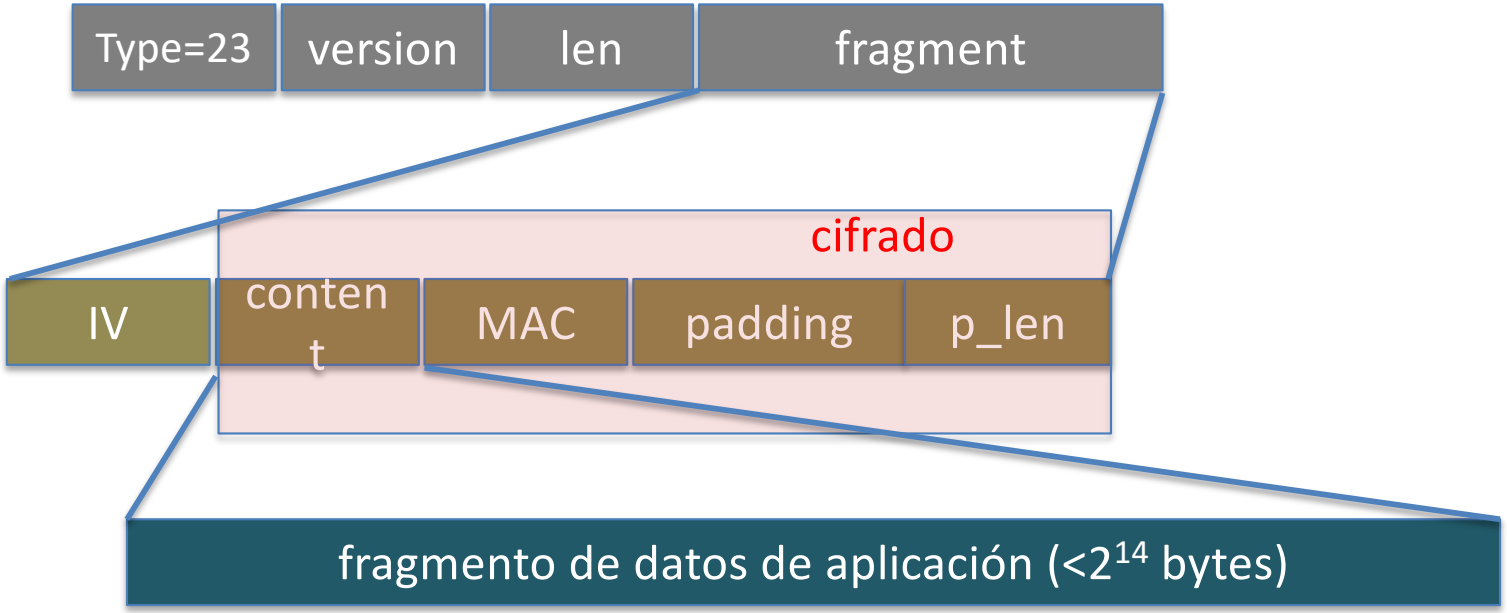

```
enum { warning(1), fatal(2), (255) } AlertLevel;

   enum {
      close_notify(0),
      unexpected_message(10),
      bad_record_mac(20),
      decryption_failed_RESERVED(21),
      record_overflow(22),
      decompression_failure(30),
      handshake_failure(40),
      no_certificate_RESERVED(41),
      bad_certificate(42),
      unsupported_certificate(43),
      certificate_revoked(44),
      certificate_expired(45),
      certificate_unknown(46),
      illegal_parameter(47),
      unknown_ca(48),
      access_denied(49),
      decode_error(50),
      decrypt_error(51),
      export_restriction_RESERVED(60),
      protocol_version(70),
      insufficient_security(71),
      internal_error(80),
      user_canceled(90),
      no_renegotiation(100),
      unsupported_extension(110),
      (255)
   } AlertDescription;

   struct {
      AlertLevel level;
      AlertDescription description;
   } Alert;
```

Layer4: TLS/SSL

- Registro de Datos (TLSCiphertext)

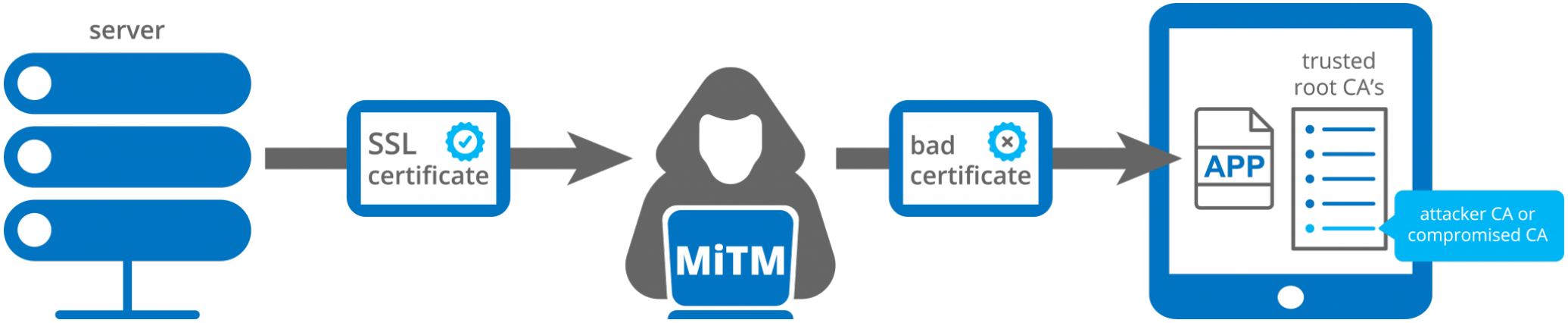


Importante: el número de secuencia no es un campo el registro sino que se incluye en el cálculo del MAC

Layer4: TLS/SSL

- Posibles ataques
  - MiM elimina cipher suites robustos del ClientHello --> Detectable con el Hash del Finished
  - MiM inyecta un paquete repetido (ataque de Replay de paquete) --> Detectable por los números de secuencia
  - MiM intercepta toda la transacción y la repite posteriormente (ataque de Replay de la conexión) --> detectable por el uso de nonces
  - Renegociación de parámetros por parte del cliente --> no debería ser soportada (vulnerabilidad)
  - Heartbleed: vulnerabilidad del OpenSSL
    - Ya ha sido parcheada

Man in the Middle



Layer4: TLS/SSL

- Recursos
  - https://www.ssllabs.com/index.html
  - https://www.openssl.org

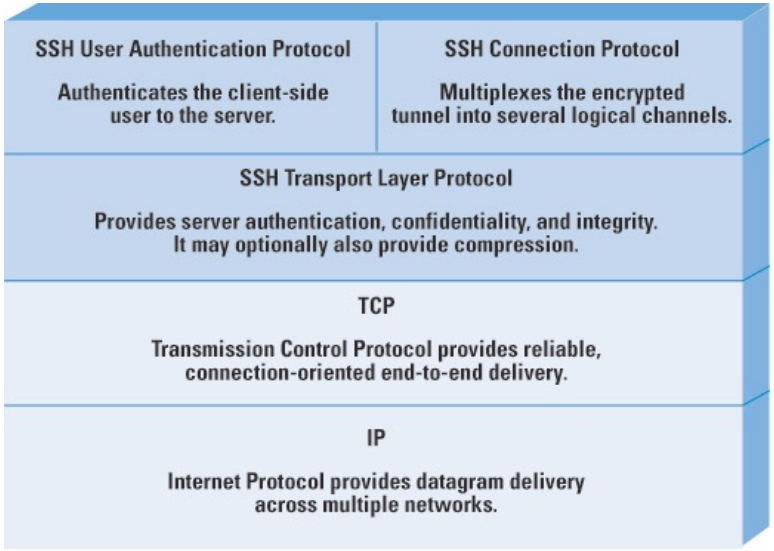
Layer4: SSH (v2 RFC 4251)

http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-4/124_ssh.html

- Secure Shell (SSH) proporciona
  - Confidencialidad
  - Integridad
  - Autenticación
  - Perfect Forward Secrecy
    - 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)
- Para
  - **login en un shell de un host remoto**
  - comunicaciones de datos
  - transferencia de ficheros segura (SFTP)
  - securización de otros protocolos a través de forwarding

Layer4: SSH

- Transport Layer Protocol (SSH-TRANS)
  - proporciona autenticación del servidor, confidencialidad e integridad con perfect forward secrecy.
- User Authentication Protocol (SSH-USERAUTH)
  - autentica al cliente frente al servidor. Va sobre SSH-TRANS
- Connection Protocol (SSH-CONNECT)
  - multiplexa el túnel cifrado en varios canales lógicos. Va sobre SSH-USERAUTH



Layer4: SSH-TRANS

Formato del paquete SSH

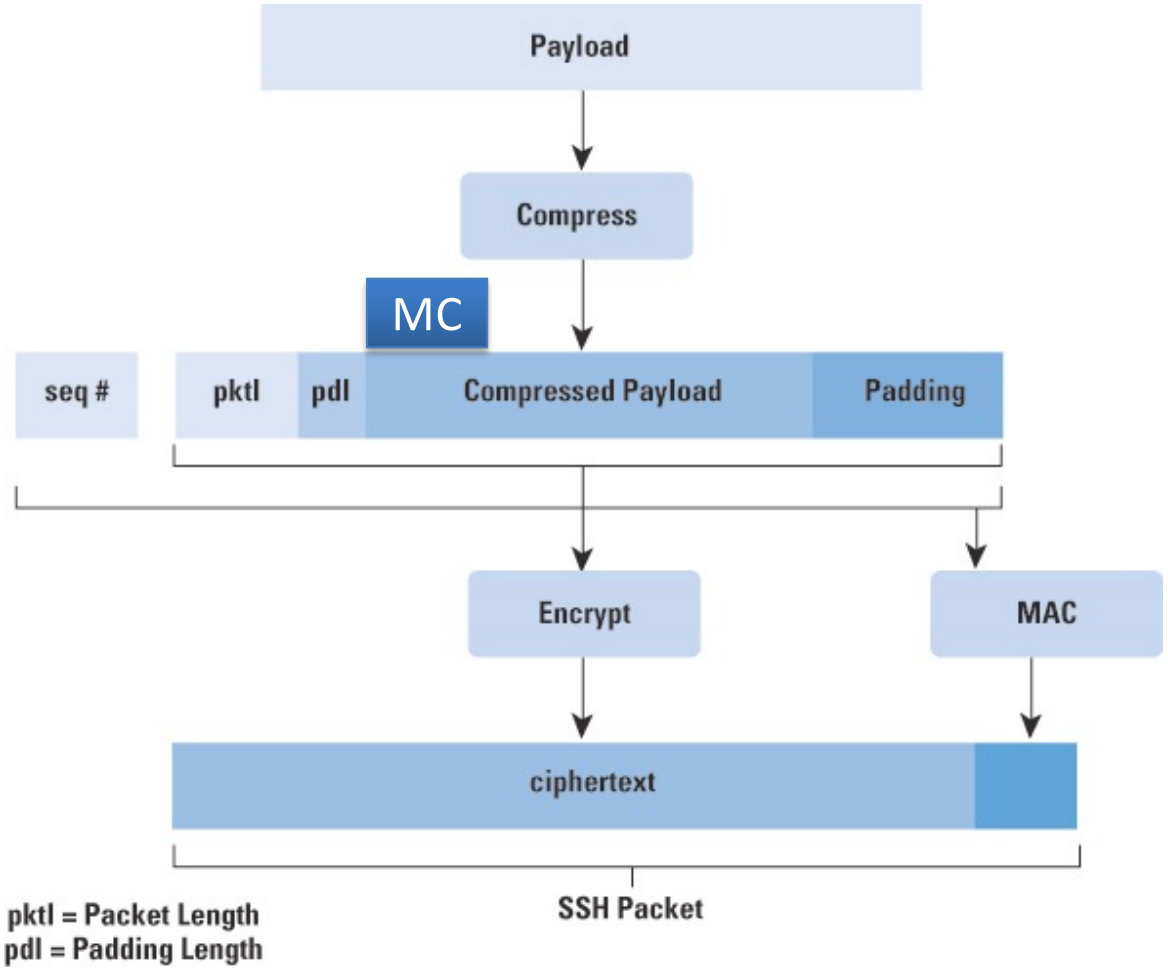
Packet length: Packet length is the length of the packet in bytes, not including the packet length and Message Authentication Code (MAC) fields.

Padding length: Padding length is the length of the random padding field.

Payload: Payload constitutes the useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets this field is compressed.[1]

Random padding: After an encryption algorithm is negotiated, this field is added. It contains random bytes of padding so that that total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher.

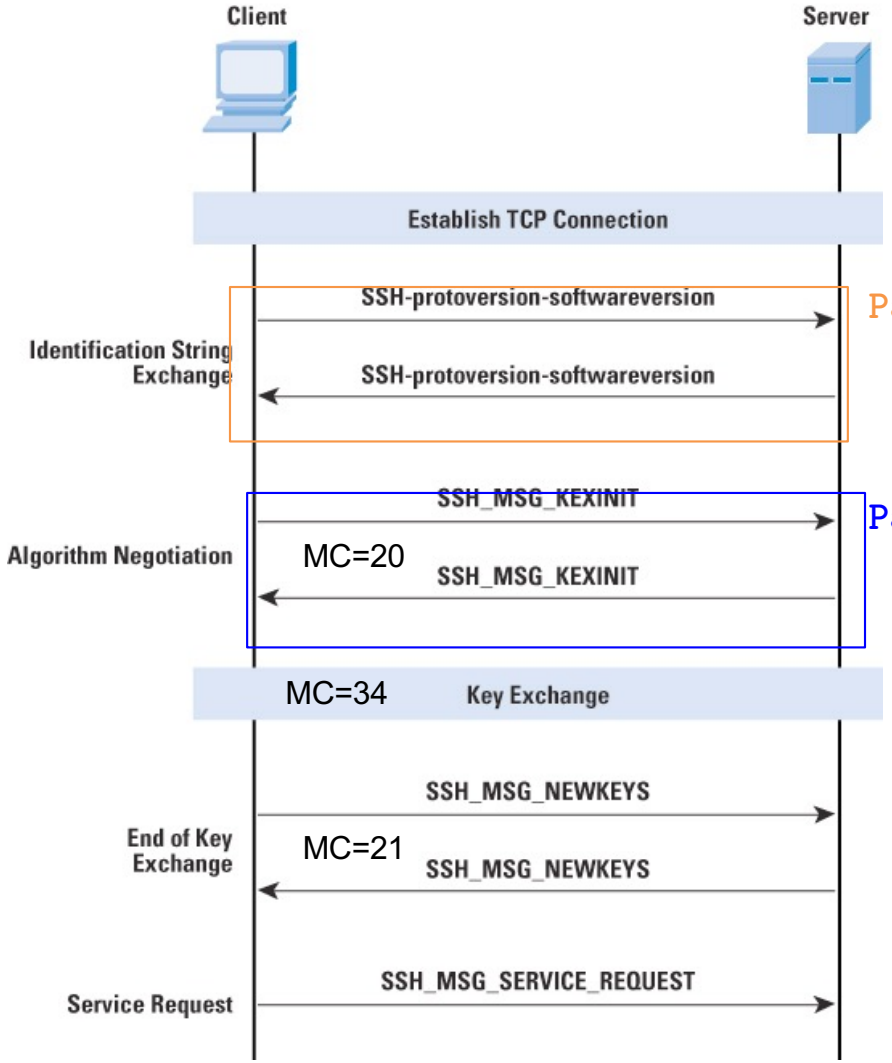
Message Authentication Code (MAC): If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.



MC: Message Code; Si el paquete no es de datos, este código indica el tipo de paquete

Layer4: SSH-TRANS

Intercambio de mensajes



Paso 1: **identification string exchange**

SSH-protoversion-softwareversion SP comments CR LF

(e.g. SSH-2.0-OpenSSH_6.0p1 Debian-4\r\n)

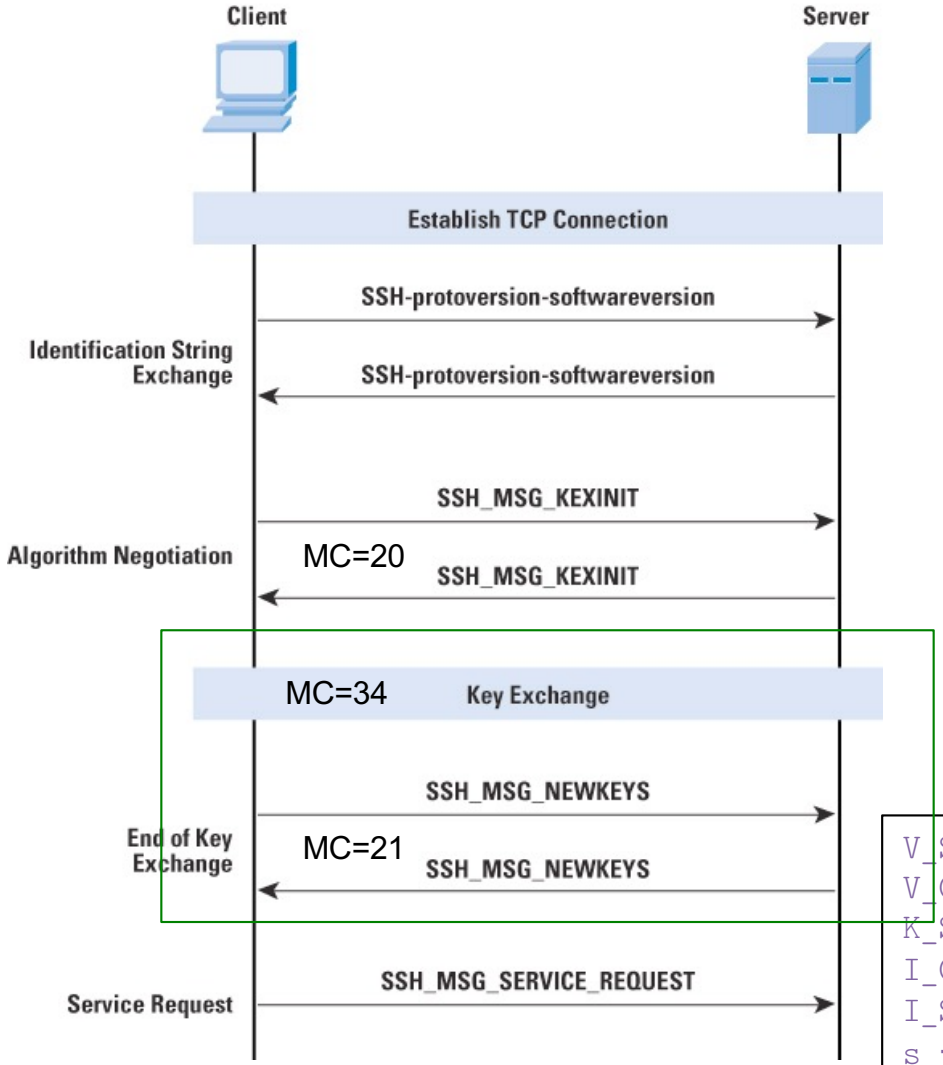
Paso 2: **algorith negociation**

Lista de algortimos (5) por orden de preferencia

1. *key exchange method (e.g. Diffie-Hellman)*
2. *Server Host Key algorithm*
   (e.g. ssh-rsa-cert-v01@openssh.com)
3. *encryption method (e.g. 3des-cbc)*
4. *MAC algorithm (e.g. hmac-sha1)*
5. *compression algorithm (e.g. zlib)*

Layer4: SSH-TRANS

Intercambio de mensajes



Paso 4: key exchange (Diffie-Hellman)

C generates a random number x (1 < x < q) and computes e = g^x mod p. **C** sends e to **S**.

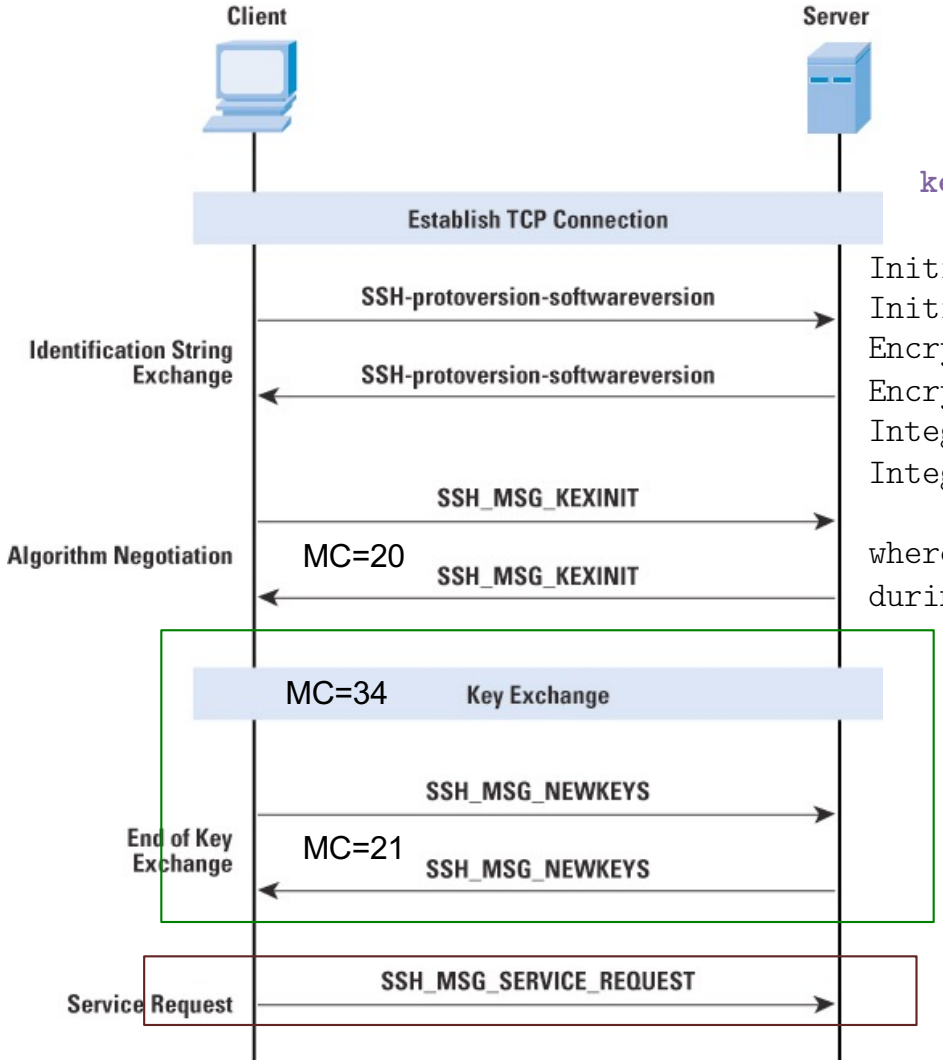
S generates a random number y (0 < y < q) and computes f = g^y mod p. **S** receives e. It computes K = e^y mod p, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K), and signature s on H with its private host key. **S** sends (K_S || f || s) to **C**.

C verifies that K_S really is the host key for **S** (for example, using certificates or a local database). **C** is also allowed to accept the key without verification; however, doing so will render the protocol insecure against active attacks (but may be desirable for practical reasons in the short term in many environments). **C** then computes K = f^x mod p, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K), and verifies the signature s on H.

V_S is the S identification string;
V_C is the C identification string;
K_S is the S public host key;
I_C is the C SSH_MSG_KEXINIT message;
I_S is the S SSH_MSG_KEXINIT
s → signature of H

Layer4: SSH-TRANS

Intercambio de mensajes



Paso 4: key exchange (Diffie-Hellman)

the hash value H serves as a **session identifier** for this connection

keys generated from K

Initial IV client to server: HASH(K || H || "A" || session_id)
Initial IV server to client: HASH(K || H || "B" || session_id)
Encryption key client to server: HASH(K || H || "C" || session_id)
Encryption key server to client: HASH(K || H || "D" || session_id)
Integrity key client to server: HASH(K || H || "E" || session_id)
Integrity key server to client: HASH(K || H || "F" || session_id)

where HASH() is the hash function determined during algorithm negotiation.

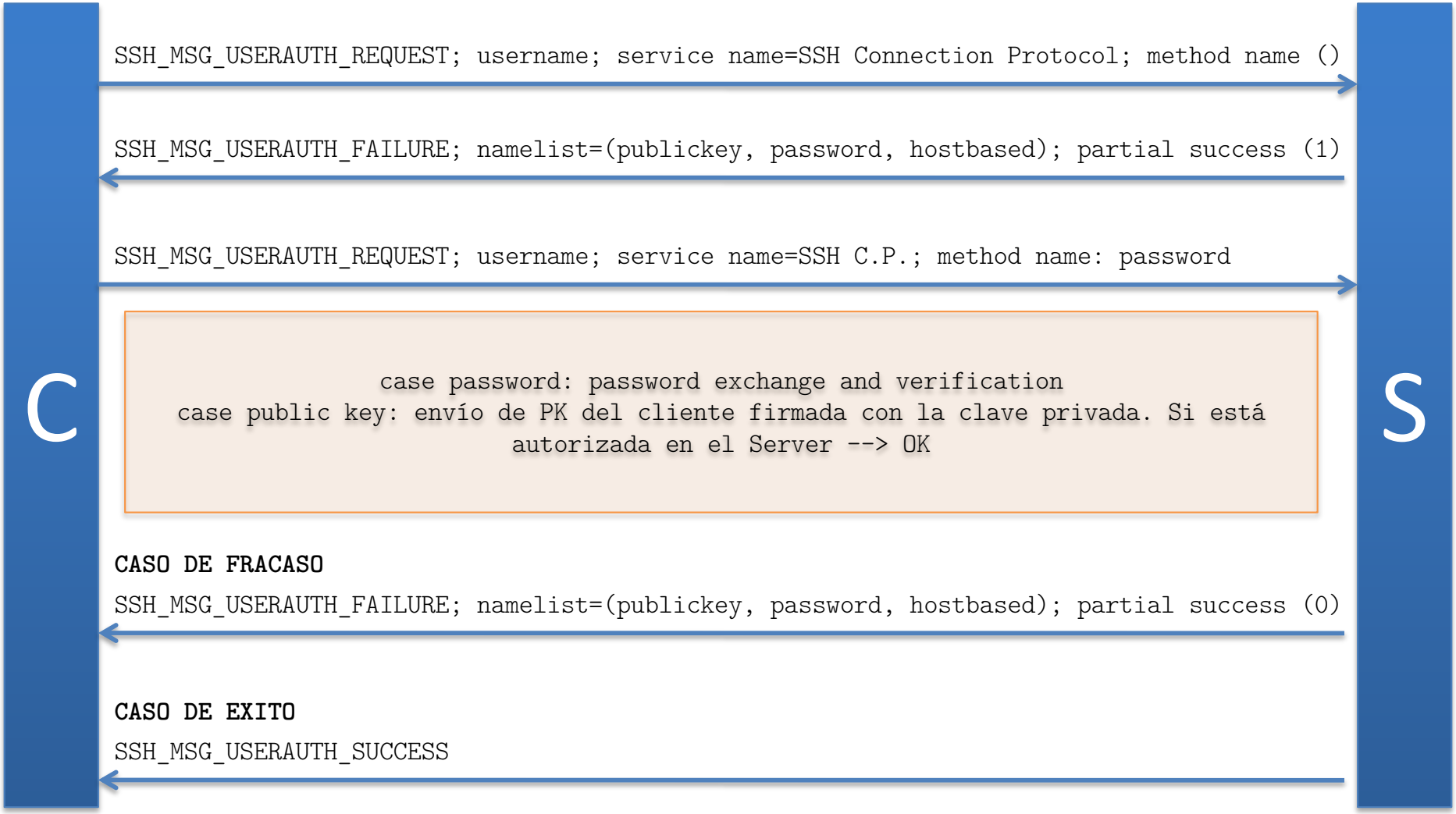
> The end of key exchange is signaled by the exchange of SSH_MSG_NEWKEYS packets. At this point, both sides may start using the keys generated from K

Paso 5: SSH_MSG_SERVICE_REQUEST

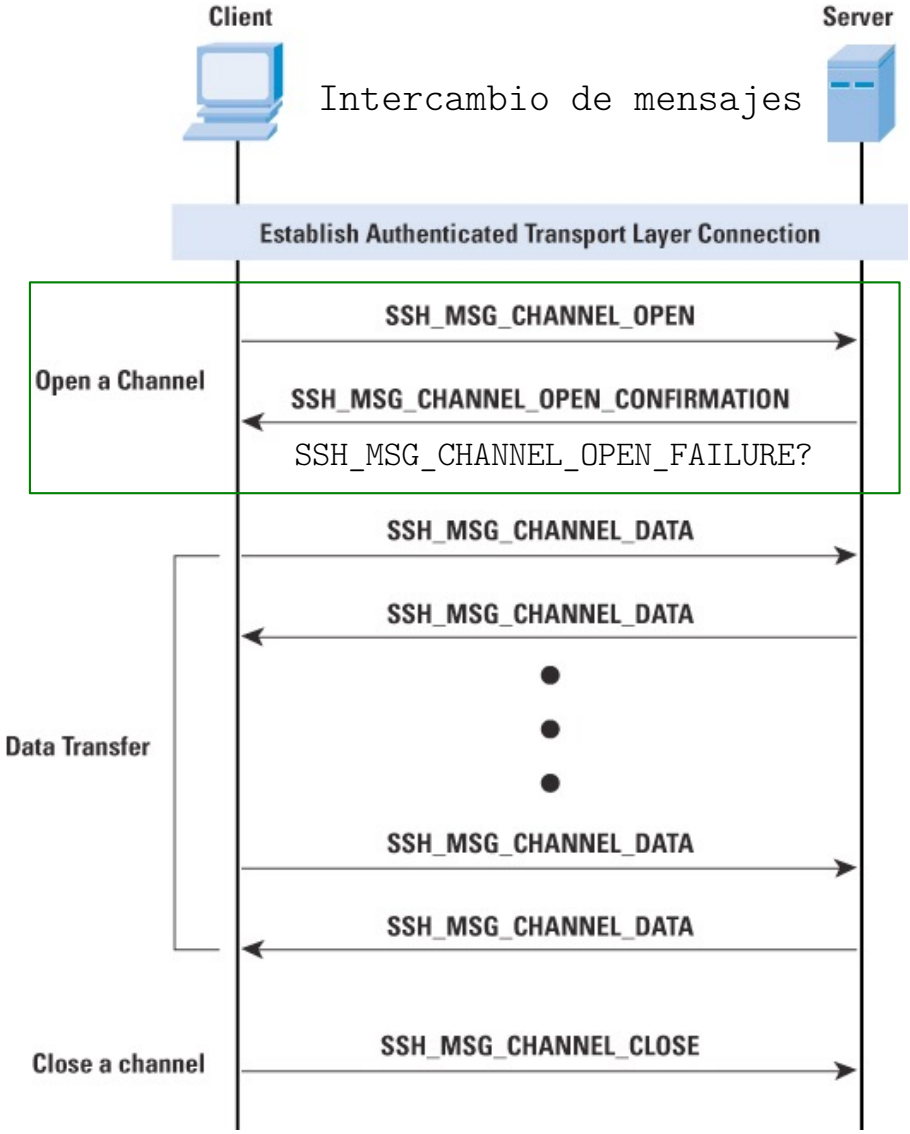
The client sends an SSH_MSG_SERVICE_REQUEST packet to request either the User Authentication or the Connection Protocol. Subsequent to this request, all data is exchanged as the payload of an SSH Transport Layer packet, protected by encryption and MAC.

Layer4: SSH-USERAUTH

Intercambio de mensajes



Layer4: SSH-CONNECT



Comunicación autenticada y segura --> Túnel

SSH-CONNECT utiliza el túnel para multiplexar uno o varios canales lógicos

Canal lógico

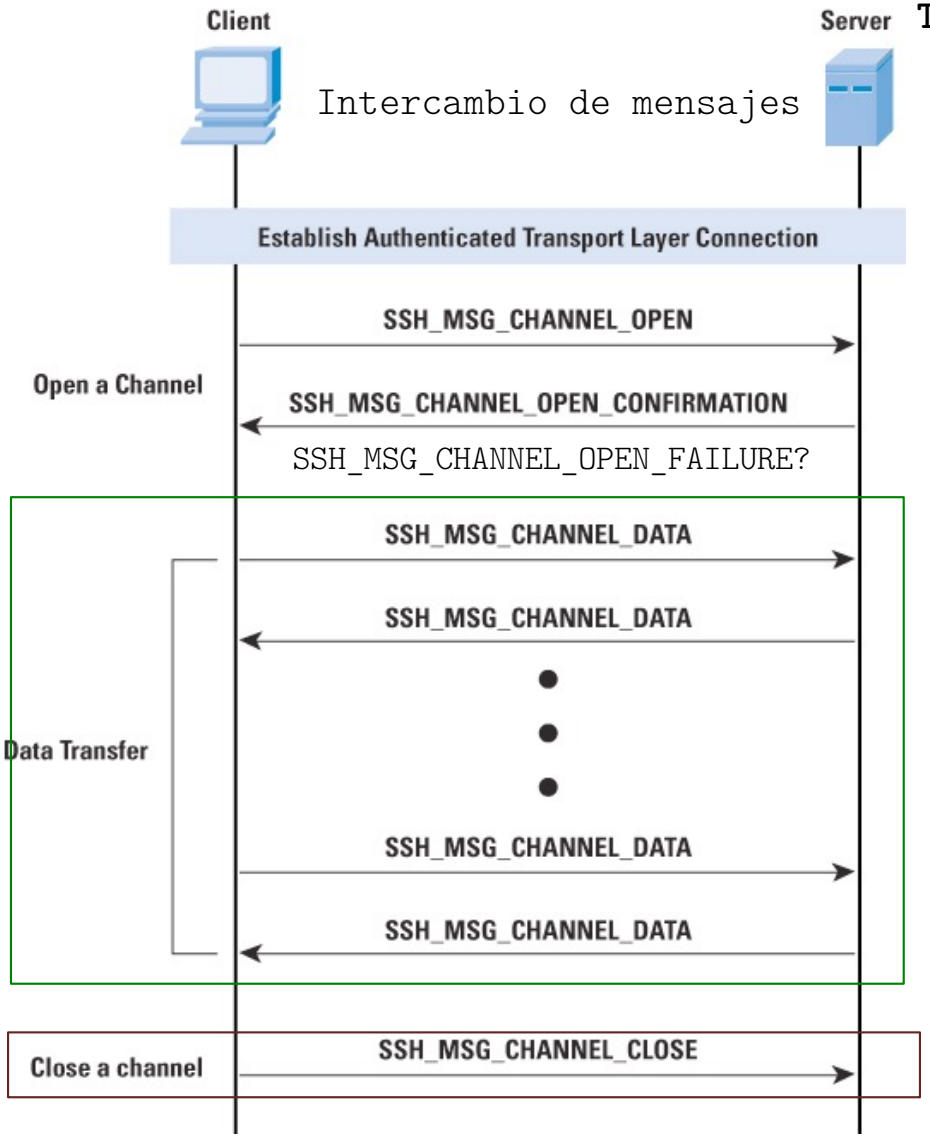
Puede ser abierto por los dos extremos
ID único asociado (no tiene por qué ser el mismo en cliente y servidor)
Tienen control de flujo mediante ventana
Ciclo de vida: Apertura, Datos, Cierre

Apertura del canal (envío de mensaje: campos)

SSH_MSG_CHANNEL_OPEN
channel type: aplicación para este canal
sender channel: ID local del canal
initial window size
maximum packet size
channel type specific data follow

SSH_MSG_CHANNEL_OPEN_CONFIRMATION
sender channel: ID local del canal
reciver channel: ID local del canal
window size
packet size

Layer4: SSH-CONNECT



Transferencia de datos

SSH_MSG_CHANNEL_DATA
reciver channel: ID local del canal
data

SSH_MSG_CHANNEL_DATA
reciver channel: ID local del canal
data

Cierre de conexión

SSH_MSG_CHANNEL_CLOSE
reciver channel: ID local del canal

Layer4: SSH-CONNECT

Tipos de canal

session: Session refers to the remote execution of a program. The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem. When a session channel is opened, subsequent requests are used to start the remote program.

x11: This channel type refers to the X Window System, a computer software system and network protocol that provides a GUI for networked computers. X allows applications to run on a network server but be displayed on a desktop machine.

forwarded-tcpip: This channel type is remote port forwarding[1].

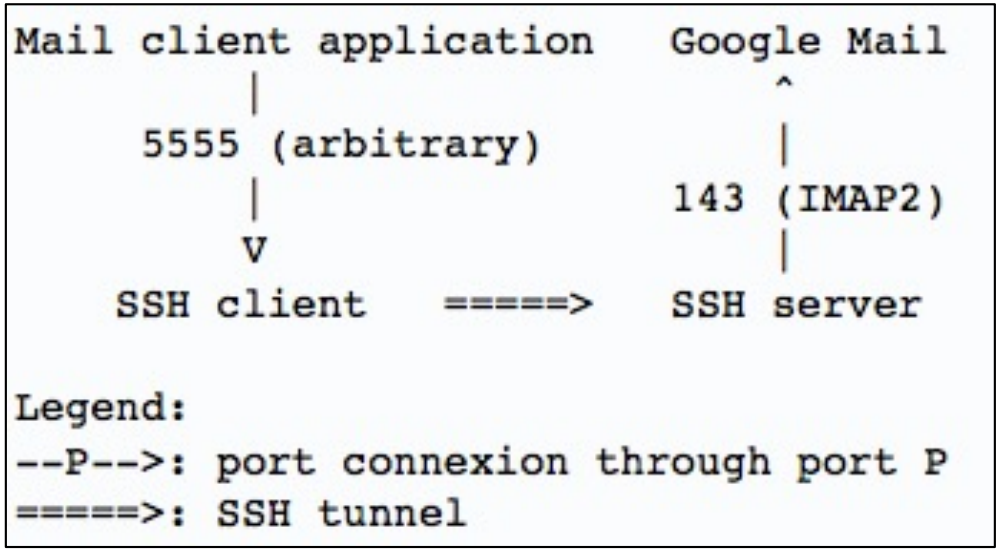
direct-tcpip: This channel type is local port forwarding[1].

[1] *One of the most useful features of SSH is port forwarding. Port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. It is also referred to as SSH tunneling.*

Layer4: SSH-CONNECT

Ejemplo de direct-tcpip

1. *El cliente abre una conexión SSH (SSH-TRANS + SSH-USERAUTH)*
2. *El cliente abre un SSH-CONNECT de tipo direct-tcpip,*
   puerto local 5555 (todo lo que tenga destino ese puerto, al túnel)
   puerto remoto 143 (informa al servidor de que todo lo reenvíe al 143)



OpenSSH

- Generación de pares de claves

```
[elvis@station elvis]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/elvis/.ssh/id_rsa): RETURN
Enter passphrase (empty for no passphrase): RETURN
Enter same passphrase again: RETURN
Your identification has been saved in /home/elvis/.ssh/id_rsa.
Your public key has been saved in /home/elvis/.ssh/id_rsa.pub.
The key fingerprint is:
e0:71:43:df:ed:40:01:0b:44:54:db:c2:80:f2:33:aa elvis@station
```

- Localización del pares de claves (servidor) (~/.ssh directory)

```
[elvis@station elvis]$ cat .ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQClJnymgdK0myP41/DcIyR9aam0DZQJUT20RLfqQb8ptk90jXSL
FrcIR2Ia59W/kJVLo4pqwJDsEJetWdhYiKUVJTANxbV2Pv21OACMlYcM316YLTOm
...
qigTMYAxoBKwPVnpAkEAvHl24SepSlAuSIwgtbluJApOfaDTizIAHh/G8PPFvH1e
p0J+MM7d/qFjg9gpcqZN34LOW8lD7Ab/GTQGl/XsWw==
-----END RSA PRIVATE KEY-----
```

```
[elvis@station elvis]$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEApSZ8poHStJsj+Nfw3CMkfWmptA2UCVE9tES36kG/KbZP
dI10ixa3CEdiGufVv5CVS6OKasCQ7BCXrVnYWIilFSUwDcW1dj79tTgAjJWHDN9emC0zpiHqBGY9dvMo
2XHobNmgQYTRFVv2NBNTA5/Zpt+Ml0+M9+uzlpxl03PcjFk= elvis@station
```

OpenSSH

- Conexión

```
$ ssh ssh-server.example.com
  The authenticity of host 'ssh-server.example.com (12.18.429.21)' can't be established.
  RSA key fingerprint is 98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d.
  Are you sure you want to continue connecting (yes/no)?
```

- Almacenamiento de la clave publica (cliente) (~/.ssh directory)

```
$ tail -1 $HOME/.ssh/known_hosts
ssh-server.example.com,12.18.429.21 ssh-rsa
AAAB3NzaC1yc2EAAAABIwAAAIEA06jFqviLMMJ/GaJNhGx/P6Z7+4aJIfUqcVjTGQasS1daDY
ejcfOAWK0juoD+zS3BsGKKYKPA5Gc5M8v+3NHLbPn1yTpDBgl6UzA0iiMPCbwnOLx61MrBT
k+/qJI9kyDaJf4LEY6Chx4IJP0ZN5NmAlCtXQsca3jwFAF72mqPbF8=
```

OpenSSH

- Conexión por password

```
cafetera-2:~ alesanco$ ssh root@155.210.156.22
root@155.210.156.22's password:
```

- Conexión por public key (almacenadas las autorizadas en el server)

```
server:~ alesanco$ more .ssh/authorized_keys
        ssh-rsa AAAAB3NzaC1yc …
```

Layer3: IPsec

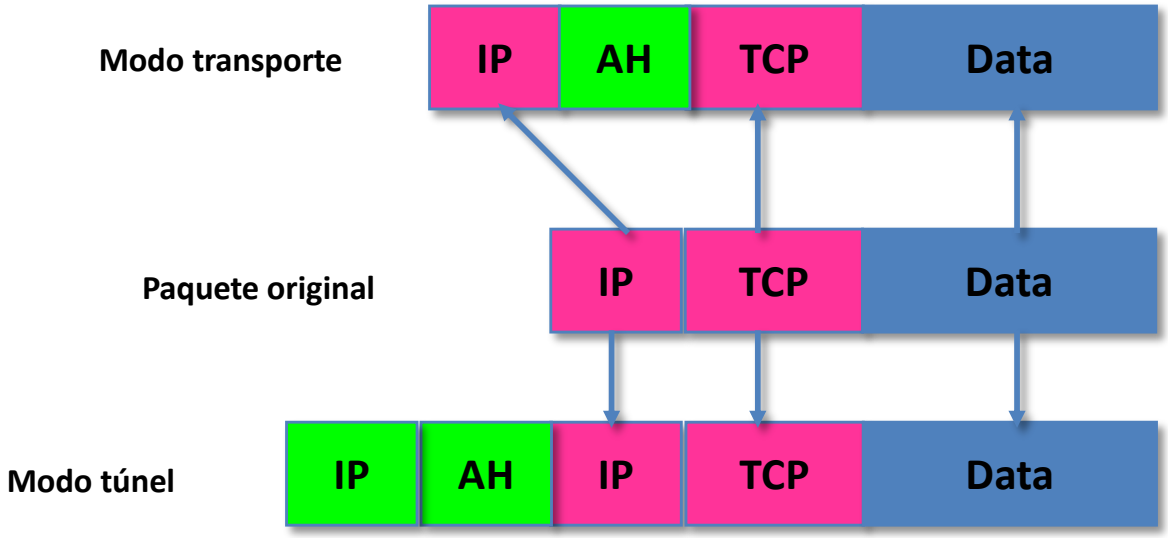
- IPsec proporciona al datagrama IP (con header)
  - Confidencialidad
  - Integridad
  - Autenticación
- Incluye protocolos para establecer una autenticación mutua
- Incluye protocolos para el negociado de las claves durante la sesión

Layer3: IPsec

- Usa dos protocolos diferentes para proporcionar integridad y confidencialidad
  - AH: Authenticated Header
  - ESP: Encapsulating Security Payload
- Puede proteger el datagrama entero o los protocolos de nivel superior (2 modos de funcionamiento)
  - Modo túnel: protege el datagrama entero
  - Modo transporte: sólo protege el payload
- En total, 4 modos de funcionamiento posibles

Layer3: IPsec

- Ejemplo: AH trabajando en modos transporte y túnel



Layer3: IPsec

- ¿Cómo proporciona IPsec seguridad?
  - Proteger integridad: HMAC (Hash Message Authentication Code)
    - MD5 (Message Digest algorithm 5)
    - SHA (Secure Hash Algorithm)
  - Proteger confidencialidad: algoritmos de cifrado simétrico (AESy 3DES)
  - Proteger frente a ataques DoS y replay: uso de números de secuencia
    - Si el paquete no está en secuencia, se descarta

Layer3: IPsec

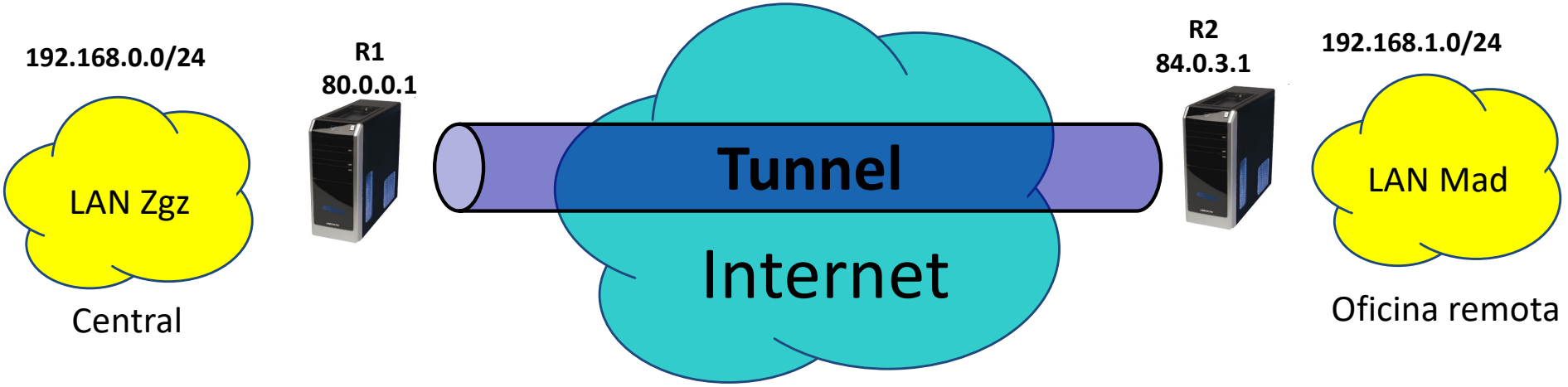
- ¿Dónde guardan los peers todos los datos necesarios para encapsular/desencapsular paquetes? Security Associations (SA)
  - Security Parameter Index (SPI)
  - IP origen y destino de la cabecera Ipsec
  - El protocolo IPSec usado (AH o ESP)
  - El algoritmo de cifrado y la clave secreta
  - El modo Ipsec (tunnel o transport)
  - Tamaño de la ventana
  - Lifetime del SA
- A su vez, las SA se guardan en las SAD (SA database)

Layer3: IPsec

- SA es unidireccional: sólo protege un sentido de la comunicación. Necesitamos 2 SA

- SA define cómo proteger el tráfico. ¿Qué trafico protegemos? Security Policy (SP)
  - IP origen y destino de los paquetes a proteger
  - El protocolo y el puerto a proteger (no todas las implementaciones lo permiten)
  - La SA a utilizar para proteger los paquetes

Layer3: IPsec



- SA de R1
  - IP ori: 80.0.0.1
  - IP dest: 84.0.3.1
  - IPSec tunnel mode con ESP
  - Cifrado con DES
  - SPI:1001

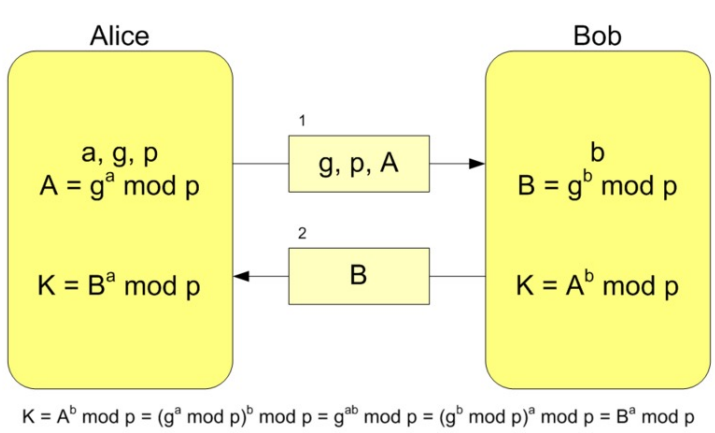
- SP de R1
  - IP ori: 192.168.0.0/24
  - IP dest: 192.168.1.0/24
  - Proteger: all/all (proto and port)
  - SA: 1001

- SA de R2
  - IP ori: 84.0.3.1
  - IP dest: 80.0.0.1
  - IPSec tunnel mode con ESP
  - Cifrado con DES
  - SPI:1002

- SP de R2
  - IP ori: 192.168.1.0/24
  - IP dest: 192.168.0.0/24
  - Proteger: all/all (proto and port)
  - SA: 1002

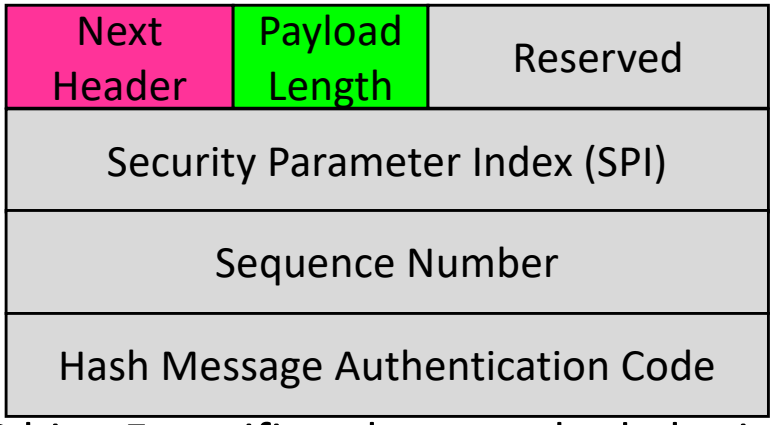
Layer3: IPsec

- Internet Security Association and Key Management Protocol (ISAKMP) + Internet Key Exchange (IKE)
  - Establecer de forma manual las SA: difícil
  - ISAKMP lo hace por nosotros
  - Necesita autenticación de los peers: secreto compartido y certificados digitales
  - IKE emplea Diffie-Hellman para intercambio de claves



Layer3: IPsec

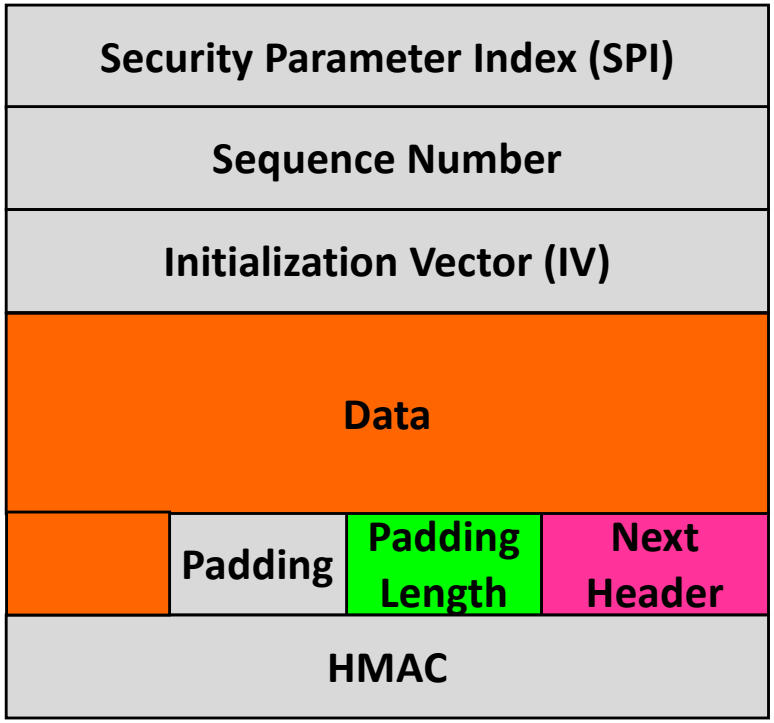
- AH – Integridad y autenticación



- **Next Header**: 8 bits. Especifica el protocolo de la siguiente cabecera (4 IP, 6 TCP)
- **Payload Length**: 8 bits. Longitud del payload
- **Reserved**: 16 bit. Usos futuros
- **Security Parameter Index**: 32 bit. Identifica la SA asociada al paquete
- **Sequence number**: 32 bit.
- **HMAC**: 96 bit. Basado en el payload y las partes inmutables de IP

Layer3: IPsec

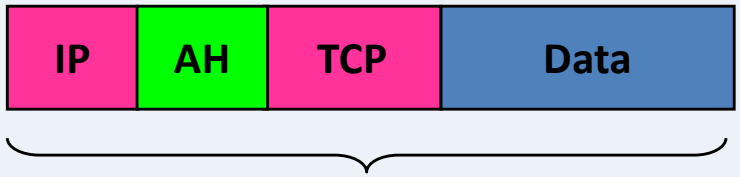
- ESP – Integridad, autenticación y confidencialidad



- Security Parameter Index: 32 bit. Identifica la SA asociada al paquete
- Sequence number: 32 bit.
- Initialization Vector: 16 bit. Para proteger de ataques de frecuencia
- Data: los datos
- Padding: codificación por bloque
- Padding Length
- Next Header: 8 bits. Especifica el protocolo de la siguiente cabecera (4 IP, 6 TCP)
- HMAC: 96 bit. Sólo Payload

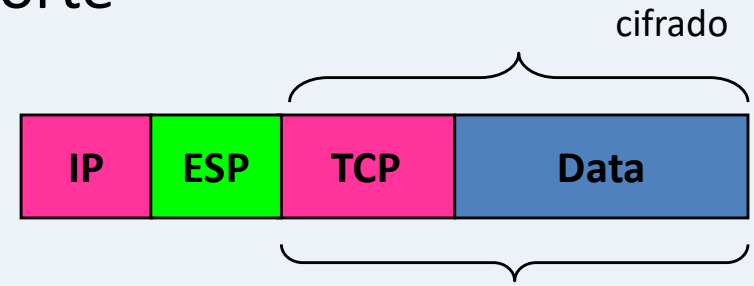
Layer3: IPsec

- PDUs en función del modo y protocolo
  - AH modo transporte



HMAC protege @IP y puertos TCP

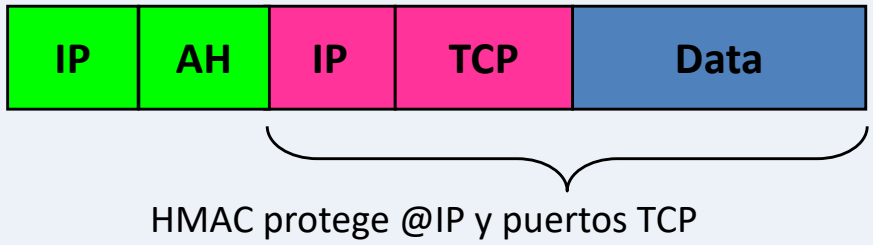
  - ESP modo transporte



HMAC protege payload IP

Layer3: IPsec

- PDUs en función del modo y protocolo
  - AH modo túnel



  - ESP modo túnel