Criptografía y Seguridad

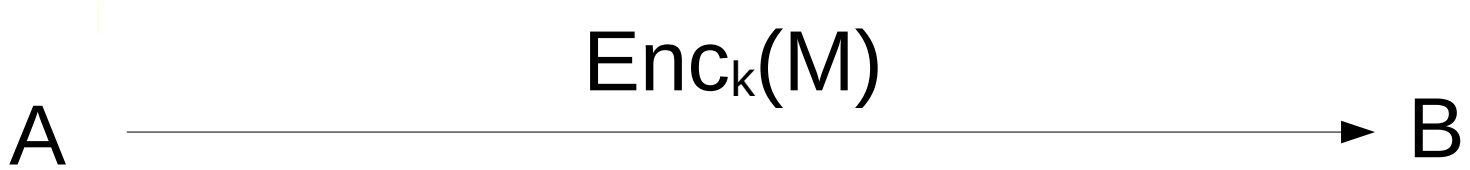
En la teoria es todo perfecto, pero en la practica surgen algunos problemas.
Por ejemplo, como es que yo comparto mi clave publica a todos.

Criptografía: Protocolos

Distribución de claves (repaso)

- Un criptosistema CCA-Secure permite enviar información manteniendo:
  - Confidencialidad
  - Integridad

Definimos seguridad como confidencialidad + integridad



- Pero requiere que ambas partes conozcan una misma clave

Intercambio de claves (repaso)

- Es un protocolo $\Pi(n)$, ejecutado por dos partes:
  - Π: (n) $\rightarrow$ Tran, k_a, k_b
  - No tiene entrada (salvo el parámetro de seguridad)
  - La salida del protocolo es
    - Un conjunto de mensajes intercambiados
    - Una clave k_a conocida solo por una de las partes
    - Una clave k_b conocida solo por la otra parte
- Condición fundamental:
  - $k_a = k_b$

Ataques Activos

- Un atacante puede

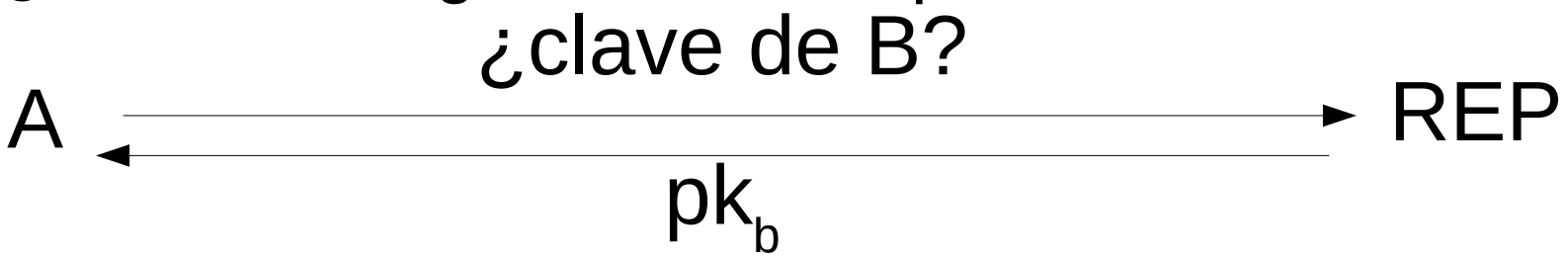
  En seguridad, que un atacante sea activo significa que gana estas 4 capacidades

  Un atacante pasivo solo veia los mensajes. Cuando hablamos de atacantes activos, aparecen los ataques MITM.

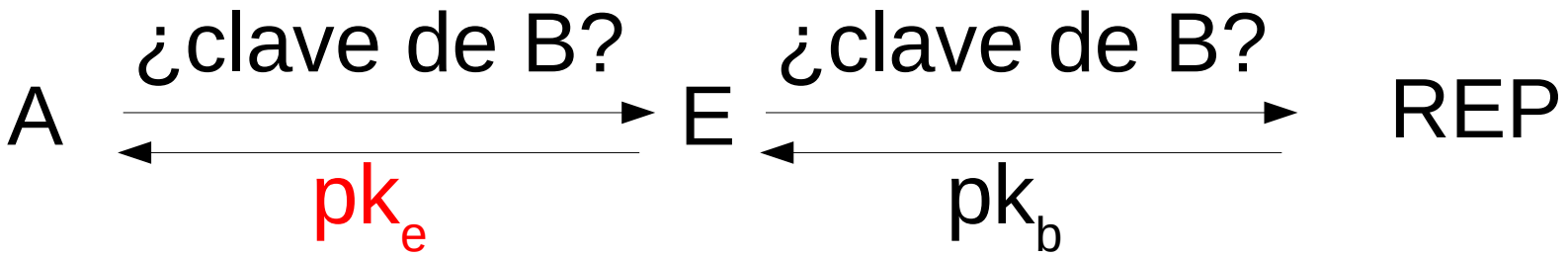
  - Omitir mensajes (A le manda a B 5 mensajes, pero el atacante hace que le lleguen 3)
  - Reescribir contenido de mensajes
  - Reordenar mensajes (los mensajes salen en un orden y llegan en otro orden)
  - Repetir mensajes (el atacante podria repetir un mensaje de una sesion anterior o una actual)
- Los esquema de intercambio vistos hasta ahora no sirven con atacantes activos
  - Ataques Man In The Middle

Problemas

- ¿De donde se obtienen las claves públicas?
  - Se confía implícitamente en la asociación de las claves con la identidad de la personas
- Si se vulnera esa asociación puede realizarse un ataque Man-In-The-Middle:
  - ¿Como consigue A la clave pública de B?



  - Lo que pasa en realidad: (le preguntamos la clave a un servidor)



Man in the middle

- Una vez que A tiene una clave equivocada, intenta comunicarse con B:
  - Lo que A imagina:

$$A \xrightarrow{Enc_{pkb}(M)} B$$

A obtiene una clave publica de B que es incorrecta. Lo que pasa en realidad es que la clave que recibio A es una clave publica que invento el atacante. En el medio, el atacante puede descifrar el mensaje y hacerle los cambios que quiera, y al mismo tiempo cifra el mensaje con la clave publica de B y se lo envia.
Tanto A como B piensan que estan hablando de manera segura.

  - Lo que pasa en realidad:

$$A \xrightarrow{Enc_{pke}(M)} E \xrightarrow{Enc_{pkb}(M)} B$$

- El problema entra en la esfera de administración de claves

Infraestructura de claves (PKI)

- Objetivo: Asociar identidad a las claves
  - Se busca evitar problemas de suplantación de identidad (man in the middle o spoofing)
  - No es aplicable a los criptosistemas simétricos

(porque en los criptosistemas simetricos las claves no se asocian a una entidad)

- Motivo
  - La selección de claves depende de las partes involucradas en la comunicación
  - Usar la clave equivocada significa que no hay garantías de confidencialidad o integridad.

Hay dos soluciones para este problema:
- En infraestructuras chicas como la de una empresa, se pueden hardcodear las claves publicas de todos los hosts
- En infraestructuras grandes como internet, se pueden usar certificados digitales

Certificados

- Son mensajes que contienen al menos:
  - Información de identidad (ej: nombre)
  - Clave pública asociada
  - Fecha de emisión
  - Intervalo de validez
  - Tipo de uso de la clave
    - Firma de mensajes
    - Cifrado de emails
    - Firma de certificados
    - Cifrado de sitios web
- Los certificados están a su vez firmados digitalmente por una autoridad competente

Los certificados digitales asocian a claves publicas con las personas. La idea es que los que emiten estos certificados hacen que sea infalsificable (porque estan firmados digitalmente por alguna autoridad)

La idea es que nosotros recibimos un certificado que esta firmado por una autoridad competente, en la que confiamos ciegamente. Cuando A quiere saber la clave de B, B en vez de devolver su clave publica, devuelve un certificado que contiene su firma (pero que esta firmado por la autoridad competente).

Uso de certificados

- A solicita certificado de B
  - Puede ser provisto por B o cualquier otra entidad
- A puede verificar la identidad del certificado
  - CN=Common Name
  - Emails: CN=<dirección de email>
  - Servidores: CN=<IP> o <hostname>
  - Empresas: CN=Razón social
- A puede obtener la clave pública de B
  - Parte del certificado mismo
  - Incluye información del tipo de clave
    - Ej: RSA-2048, DSA-EC 320, etc

Uso de certificados (2)

- A puede verificar la validez del certificado
  - Comprobando el tipo de uso permitido
  - Comprobando la fecha de vigencia
- A puede verificar la integridad del certificado
  - Validando la firma digital de la autoridad que lo certifica

La validación de una firma digital requiere una clave pública
¿Cómo se obtiene?

Cadenas de firmas

- ¡Las autoridades certificantes tienen a su vez un certificado!
  - Se acostumbra incluir el certificado de la autoridad junto con cada certificado
- ¿Quien valida el certificado de la AC?
  - ¡Otra AC!
- AC Raices
  - Son aquellas autoridades que firman su propio certificado.
  - Son el punto de confianza del sistema

La idea es que yo voy pidiendo el certificado de cada AC hasta llegar a los AC raices. Estos AC raices estan hardcodeados en la computadora (como en los SO o en los navegadores). Esto se puede hacer porque son pocos los AC raices.

Cadenas de firmas (2)

- Concepto de AC Raíces

Nadie esta dispuesto a poner toda la certificacion en un solo organismo (por temas politicos por ejemplo).
Lo que ocurre es que las entidades cerficantes raiz pertenecen a un grupo que se autocertifican entre si y se avalan de forma cruzada.

  - Confiar en una única autoridad certificante
  - Dicha autoridad delega en otras la capacidad de firmar certificados

- Ejemplos:
  - Certificado de A, firmado por CA3 (A<<CA3>>)
    - $C_a = C'_a \| C_{CA3} \| C_{CA2} \| C_{CA1}$
  - Certificado de B, firmado por CA1 (B<<CA1>>)
    - $C_b = C'_b \| C_{CA1}$

Validación entre C.A.

- No existe una única C.A. Raíz
- Si A y B tienen dos C.A. diferentes, ¿como valida A el certificado de B?
  - Opción 1: A confía en la C.A. de B
  - Opción 2: Las C.A. Se certifican entre sí
    - Cada C.A. Emite un certificado de la identidad de la otra
- En la práctica:
  - Existe una lista de C.A.s reconocidas:
    - En el sistema operativo
    - En los navegadores
    - En runtimes como JVM

Certificados X.509

- Estándar de certificados digitales
- Incluye:
  - Versión
  - Número de serie
  - Identificador de algoritmo de firma
  - Nombre del emisor (C.A.)
  - Intervalo de validez
  - Nombre del sujeto (C.N. - common name)
  - Clave pública del sujeto
  - Firma: firma del hash de todo lo anterior
- Forman cadenas de certificados

X.509

```
1.Certificate:
2.   Data:
3.       Version: 3 (0x2)
4.       Serial Number: 1 (0x1)
5.       Signature Algorithm: md5WithRSAEncryption
6.       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
7.               OU=Certification Services Division,
8.               CN=Thawte Server CA/Email=server-certs@thawte.com
9.       Validity
10.          Not Before: Aug  1 00:00:00 1996 GMT
11.          Not After : Dec 31 23:59:59 2020 GMT
12.       Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
13.                OU=Certification Services Division,
14.                CN=Thawte Server CA/Email=server-certs@thawte.com
15.       Subject Public Key Info:
16.          Public Key Algorithm: rsaEncryption
17.          RSA Public Key: (1024 bit)
18.              Modulus (1024 bit):
19.                  00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
20.                  68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
21.                  85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
22.                  6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
23.                  6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
24.                  29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
25.                  6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
26.                  5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
27.                  3a:c2:b5:66:22:12:d6:87:0d
28.              Exponent: 65537 (0x10001)
29.       X509v3 extensions:
30.          X509v3 Basic Constraints: critical
31.              CA:TRUE
32.    Signature Algorithm: md5WithRSAEncryption
33.       07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
34.       a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
35.       3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
36.       4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
37.       8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
38.       e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
39.       b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
40.       70:47
```

X.509

```
1.Certificate:
2.    Da
3.
4.
5.
6.
7.
8.              CN=Thawte Server CA/Email=server-certs@thawte.com
9.        Validity
10.           Not Before: Aug  1 00:00:00 1996 GMT
11.           Not After : Dec 31 23:59:59 2020 GMT
12.
13.
14.
15.
16.
17.
18.             Modulus (1024 bit):
19.                 00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
20.                 68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
21.
22.
23.
24.
25.
26.
27.
28.             Exponent: 65537 (0x10001)
29.       X509v3 extensions:
30.           X509v3 Basic Constraints: critical
31.               CA:TRUE
32.   S
33.
34.
35.
36.
37.
38.
39.       b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
40.       70:47
```

Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com

Autoridad emisora

Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com

Dueno de la clave publica

Validity
Not Before: Aug 1 00:00:00 1996 GMT
Not After : Dec 31 23:59:59 2020 GMT

X509v3 extensions:
X509v3 Basic Constraints: critical
CA:TRUE

La idea es que si se encuentra un certificado critico que no se conoce, no se puede confiar

X.509

```
Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                    68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                    85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                    6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                    6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                    29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                    6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                    5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                    3a:c2:b5:66:22:12:d6:87:0d
                Exponent: 65537 (0x10001)
```

```
5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
```

```
    Signature Algorithm: md5WithRSAEncryption
        07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
        a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
        3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
        4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
        8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
        e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
        b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
        70:47
```

Verificación de certificados X.509

- Obtener la clave pública del emisor
  - En la cadena de certificados o del sistema si es raíz
  - Si es necesario, verificar recursivamente el certificado del emisor
- Verificar integridad del certificado
  - Utilizando el algoritmo especificado y la clave pública del emisor
- Verificar intervalo de validez
  - Debe estar vigente.
  - La autoridad certificante debe estar vigente al comienzo del periodo de vigencia.

La clave de la autoridad certificante debe ser valida al momento de creacion del certificado (porque hay algunos ataques en donde se roban claves viejas de autoridades certificantes)

Verificación de certificados X.509

- Verificar identidad de certificado
  - Comparar el CN con el que espera la aplicación
  - Esto depende del uso del certificado
- Verificar el uso del certificado
  - Validar que el certificado esté autorizado para el uso que se le quiere dar

Dado que hay que validar una cadena larga de certificados a veces (como 4 o 5), podemos notar cierto lag al conectarnos a un sitio por primera vez

Revocación de claves

- Necesidad de invalidar una clave ANTES de su expiración
  - Fue averiguada por un atacante
  - Cambio anticipado (cambio de dueño de clave)
- Problemas
  - No revocar claves que no deben ser revocadas
    - Evitar que se revoquen claves sin autorización
  - Propagar la información para evitar futuras comunicaciones con dicha clave

Listas de revocación

- Listas de certificados revocados
  - Similar a listas de números de tarjeta de crédito robadas
  - Listado actual: certificados vigentes y revocados
  - Listado histórico: certificados expirados y revocados
- X.509: Solo el emisor de un certificado puede revocarlo

Generalmente se suele pedir que se entregue la clave privada para revocar una clave publica (esto suena raro pero total no se va a usar mas la clave publica)

  - Se agrega al CRL de la autoridad certificante global
  - La lista se puede obtener para validar offline o puede ser validada online

Online Certificate Status Protocol

- Permite certificar que una firma esta “activa”
  - Servicio online con API estandarizada
  - Administrados por las ACs
  - El resultado esta firmado digitalmente
    - Es un certificado
    - De corta duración (1 hora a 7 días)
- Puede estar incluido en la cadena de certificados
  - Esto se conoce como OCSP Stappling
  - Hace que el cliente no tenga que consultar una fuente externa
  - Es recomendado para sitios de alto trafico

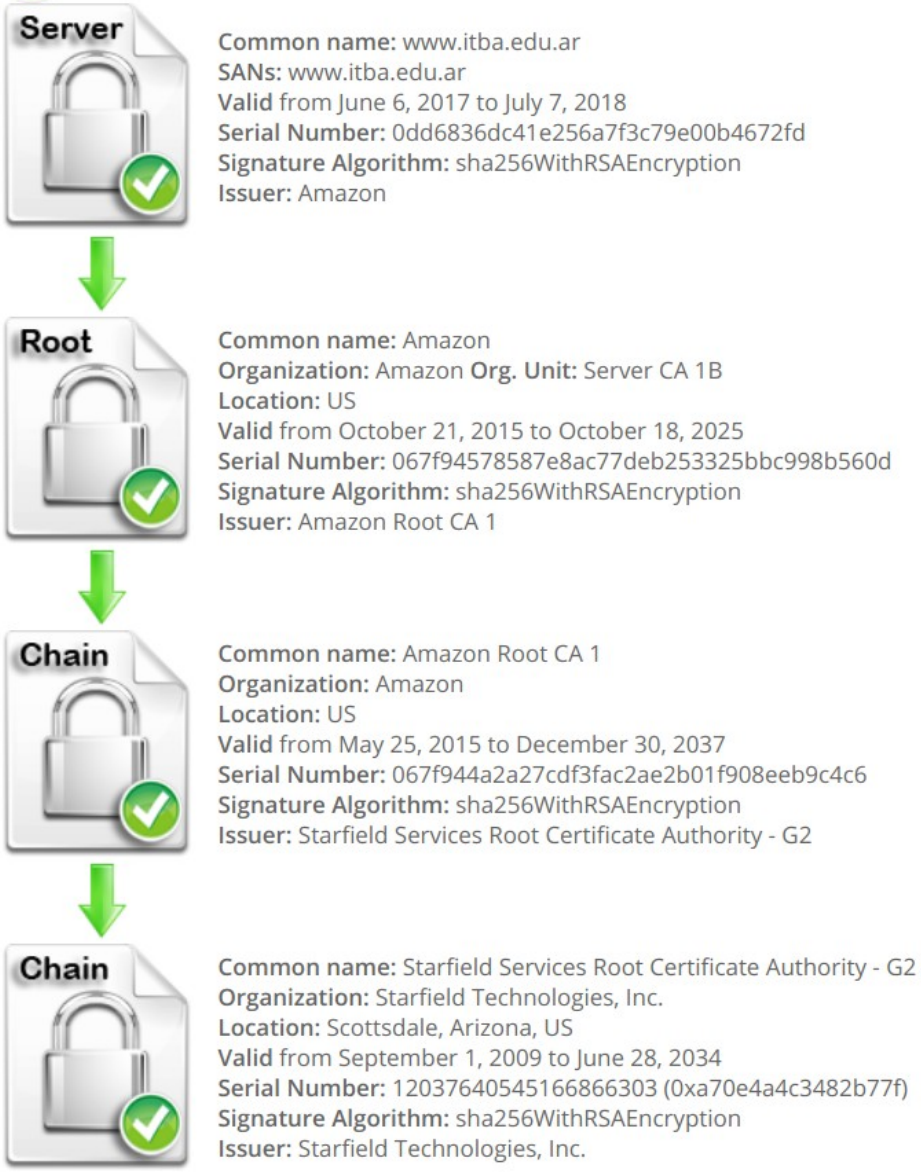
Es una simple consulta online a una API que nos dice si un certificado esta revocado o no. Devuelve un certificado que certifica que tal certificado es valido (y tiene una validez muy corta, un par de horas).
Este tipo de protocolo surge cuando se necesita mas seguridad que la lista de certificados revocados.

Resumen certificados

- Asocian una clave publica con una identidad
- Se pueden validar offline
  - Cuidado con posibles revocaciones
- Son certificados por otra entidad
- Forman cadenas de confianza (cada certificado esta certificado por otra AC)
- Resuelven el tema de integridad de las claves publicas En el dia de hoy, es la mejor manera que tenemos de autorizar la autenticidad de claves publicas

Ejercicio

- Describir todas las validaciones necesarias:



Intercambio simétrico de claves

Equivalente simétrico a DH

Protocolo Needham-Schroeder

- Protocolo de intercambio de claves simétrico
  - Es el protocolo utilizado por Kerberos y Active Directory (con modificaciones)
  - Requiere un servicio centralizado (KDC)
- Genera claves de sesión entre pares
  - Parte de la hipótesis de que cada entidad comparte una clave con el KDC

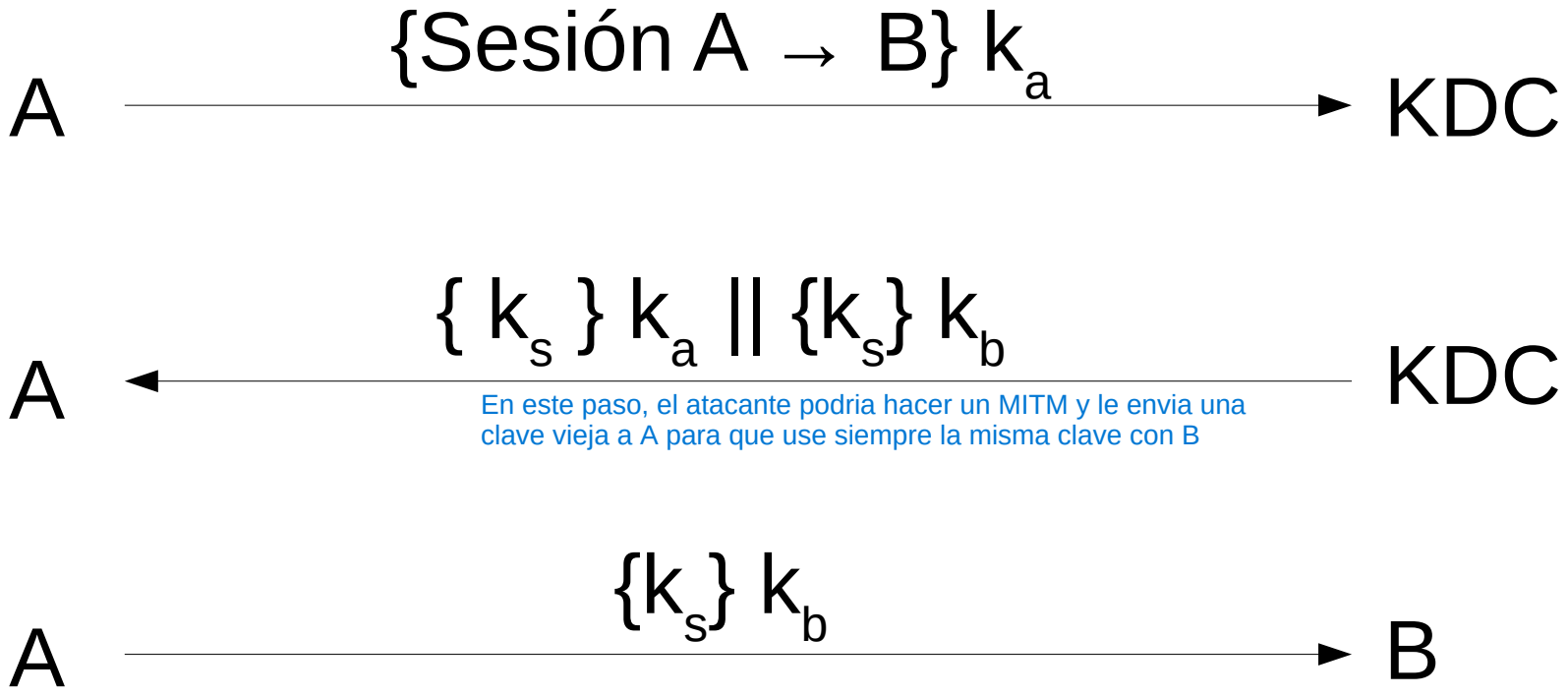
simetrico = unica clave

Servicio central que interviene en la negociacion.
Este protocolo asume que todos los hosts comparten una clave distinta con dicho servicio.

Primera aproximación

Notacion: Lo que esta entre llaves esta cifrado con la clave que continua

Pasos:
1) A le pide al KDC ayuda para establecer una sesion
2) El KDC le contesta una clave de sesion cifrada con la clave de A y la clave de sesion cifrada con la clave de B
3) A le envia la clave de sesion cifrada con la clave de B a B
4) Tanto A como B tienen la clave de sesion



Para simplificar, se utiliza la nomenclatura:
$Enc_k(M) = \{ M \} k$

Problemas

- Repetición (Replay):

  El atacante se hace pasar por A

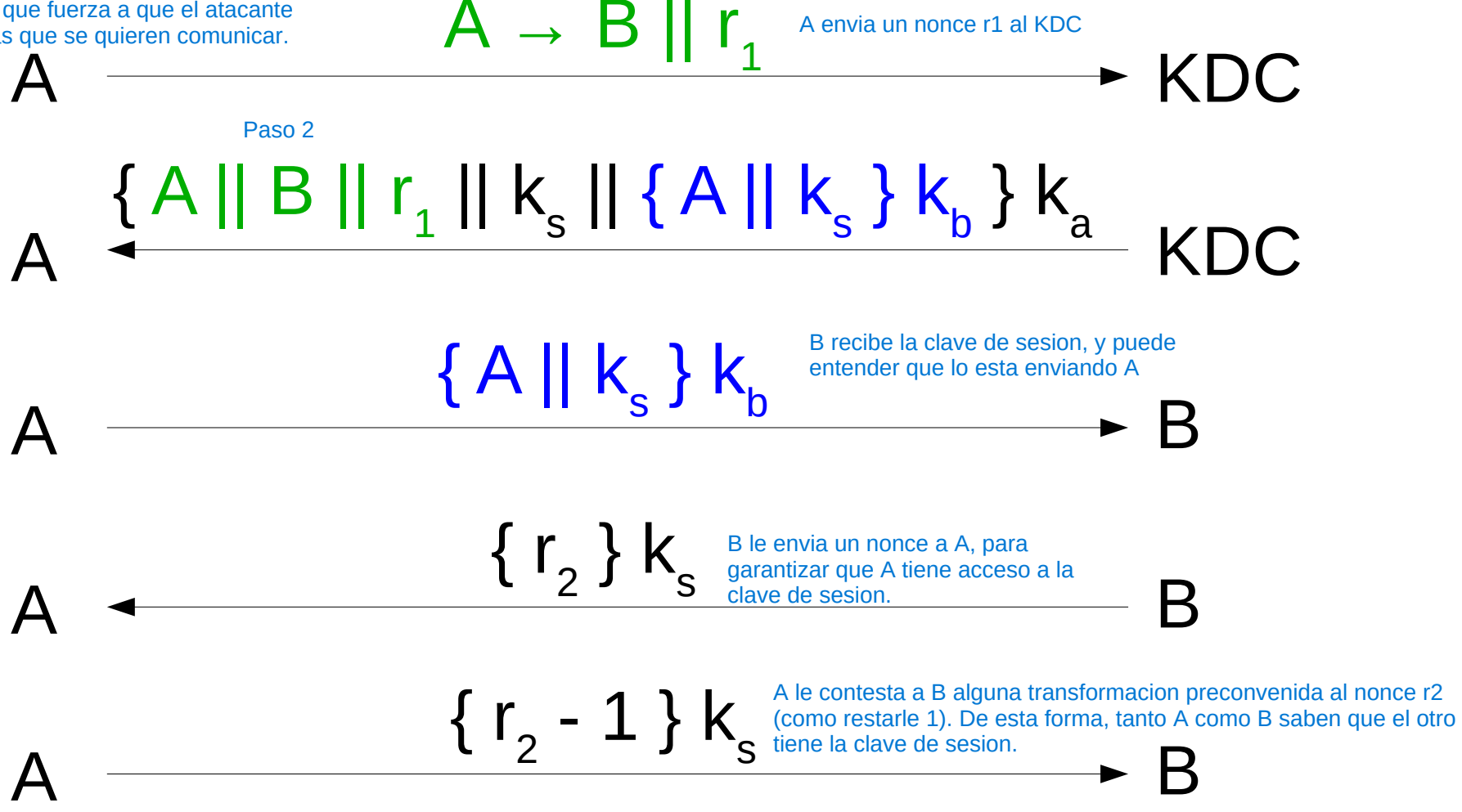
  - Un atacante puede grabar mensajes de A hacia B y luego reenviar $\{k_s\}\ k_b$ y los mensajes siguientes
  - B no tiene forma de saber que no está hablando con A

- Reuso de claves (Key Reuse)
  - Un atacante graba el mensaje de C a A, y cuando A quiere iniciar otra conversación se lo envía
  - A y B estarán utilizando la misma clave de sesión

    El atacante podria forzar a que A y B siempre usen la misma clave.
    Si el atacante llega a obtener esta unica clave, podria descifrar todos los mensajes de la sesion entre A y B.
    Esto no es un problema si A y B usan distintas claves todo el tiempo, ya que conseguir una sola clave es mucho esfuerzo para el atacante, y este solo podria descifrar pocos mensajes.

Segunda aproximación

Paso 2
El KDC incluye el nonce en la respuesta. Esto hace que un atacante no pueda reemplazar la respuesta del KDC por una respuesta vieja (por ende no puede obligar a A a usar siempre la misma clave).
La respuesta tambien incluye que A pidio comunicarse con B, lo que fuerza a que el atacante no cambie las personas que se quieren comunicar.



Explicación

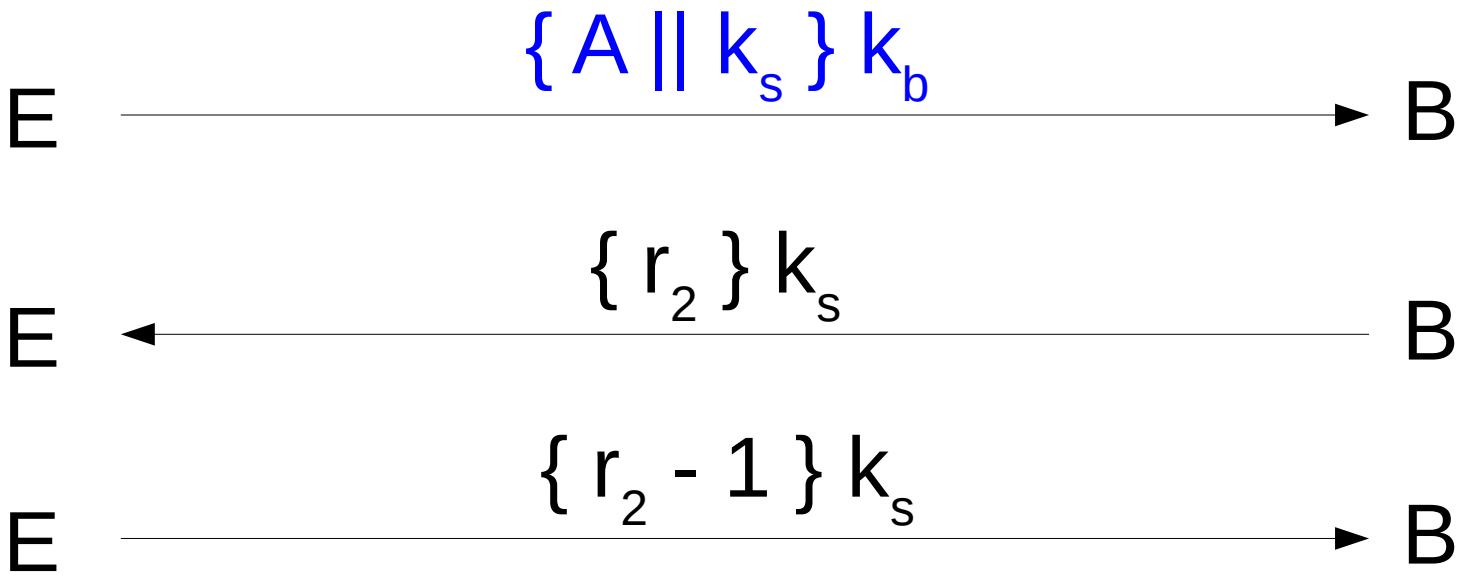
- Segundo mensaje
  - Encriptado con clave compartida A-C (C = KDC)
    - El mensaje proviene de C
  - No es repetición
    - El número r_1 recibido coincide con el enviado
- Cuarto mensaje
  - Solo B puede enviarlo
    - Avisa a A de un intento de comunicación
- Quinto mensaje
  - A confirma la comunicación
  - B sabe que no es una repetición (por r_2)

Problema

Escenario: El atacante (E) obtiene una clave de sesión antigua k_s

Si el atacante consigue una clave de sesion vieja (puede tener años) y la clave generada en el paso 3, entonces E puede impersonar a A.

- Comienza por el tercer paso:



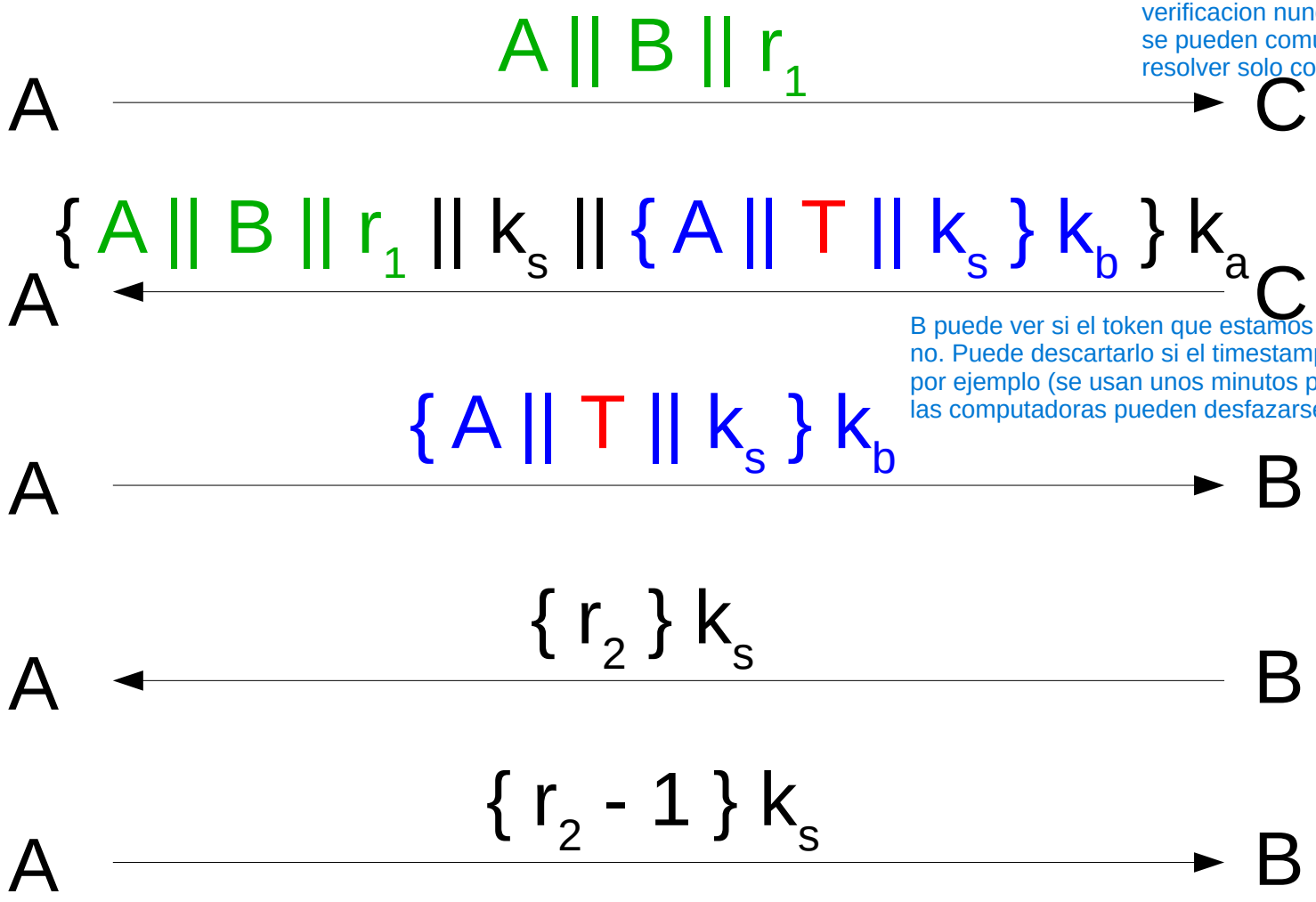
E logra impersonar a A!

Modificación Demming-Sacco

- Arreglo propuesto: Usar timestamps

Si nuestro problema es que un atacante puede recuperar una clave de sesion antigua, entonces incluyo un timestamp entre la comunicacion A-KDC y A-B. Esto resuelve un problema que ninguna primitiva criptografica puede resolver (la reutilizacion de un mensaje valido)

Con este agregado, Kerberos funciona de manera segura. Recordar que Kerberos se usa para que dos partes puedan comunicarse de manera segura. El unico ataque que puede hacer un atacante es denegacion de servicio. El atacante simplemente borra algunos mensajes, por lo que los 5 pasos de verificacion nunca se dan y las partes no se pueden comunicar. Esto no se puede resolver solo con criptografia.



B puede ver si el token que estamos recibiendo es nuevo o no. Puede descartarlo si el timestamp tiene mas de 5 minutos por ejemplo (se usan unos minutos porque en redes cerradas las computadoras pueden desfazarse un poco en tiempo).

Canal seguro

Confidencialidad e integridad sobre un canal inseguro

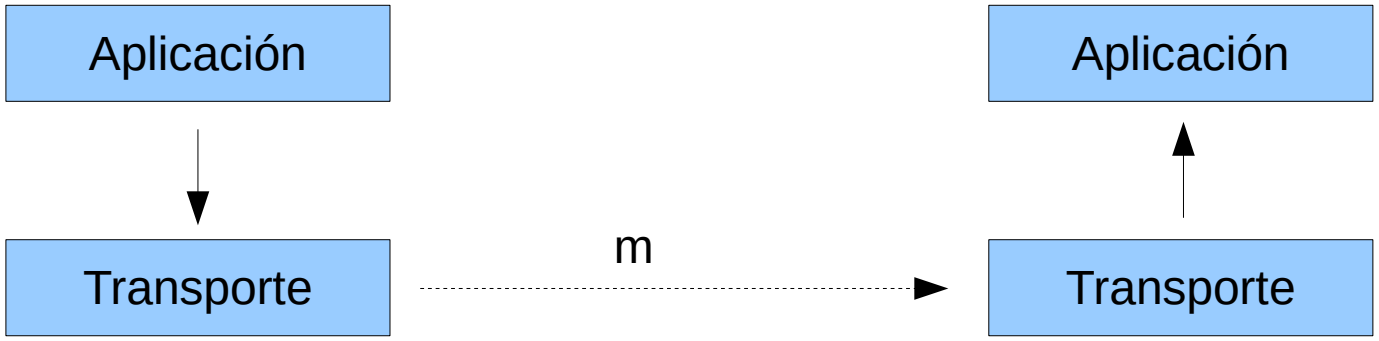
SSL / TLS

- Ofrece Seguridad a nivel transporte
- Utiliza un canal de transporte confiable (TCP?)
- Provee confidencialidad, integridad y autenticación de los participantes (origen y destino)
- SSL – Secure Socket Layer
  - Versión inicial, creada por Netscape
- TLS – Transport Layer Security
  - Evolución de SSL (TLS 1.2 = SSL 3.3)
  - Mejora deficiencias encontradas
  - Estándar actual

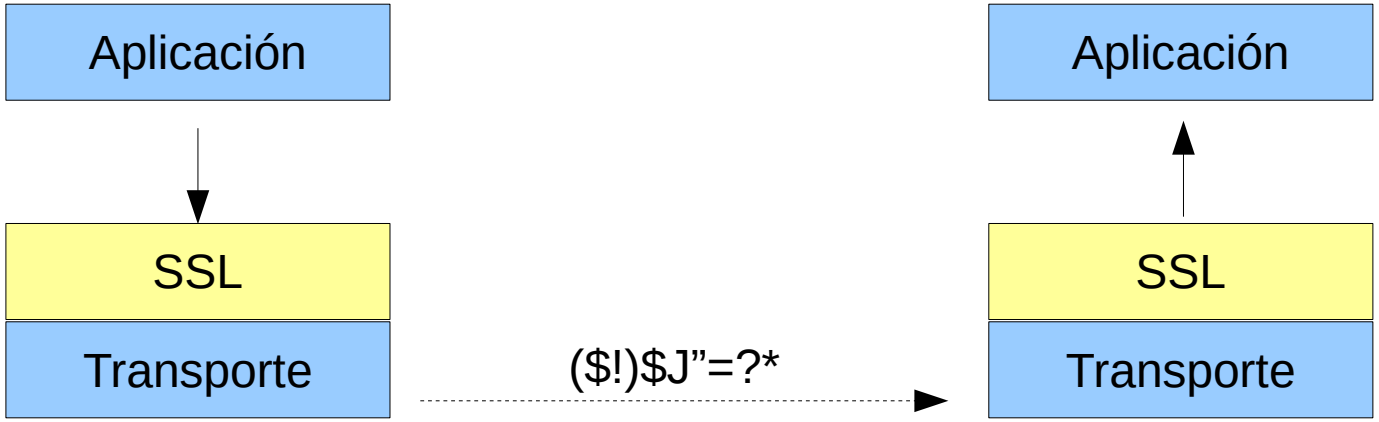
Migracion no compatible hacia atras.
TLS fue un major change.

Concepto

- Conexión normal, insegura

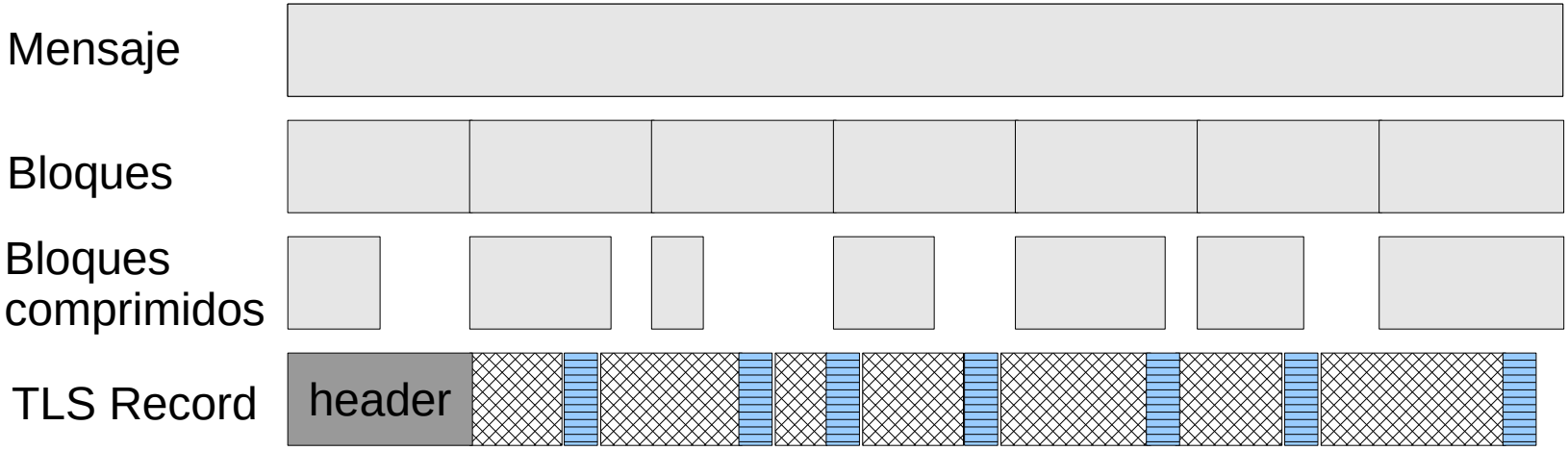


- Conexión con SSL, protegida



TLS Record

- TLS recibe un mensaje a enviar (65kb)
- Lo divide en bloques de no más de 2^{16} bytes
- Comprime cada bloque, y calcula su hash (en realidad es un MAC, no un hash)
- Encripta cada bloque y hash
- Lo envía al servicio inferior (por ejemplo, TCP)



TLS – Tipo de mensajes

- TLS handshake
  - Configuración inicial
  - Autenticación
  - Negociación de material criptográfico
- TLS application data
  - Contiene información del protocolo encapsulado
- TLS alert
  - Informa de eventos y problemas
- TLS Change cipher spec
  - Concluye una renegociación de claves

se indica que se encontro una anomalia (como un MAC que no coincide). generalmente se mata la conexion SSL y se tiene que iniciar una nueva

Como TLS es de uso general, tiene que soportar todo tipo de comunicacion, incluyendo comunicaciones muy largas (algunas que no se cierran nunca).

TLS - Criptografía

- Intercambio de clave:
  - RSA
  - Diffie Hellman, certificado RSA o DSS
  - Diffie Hellman, anonymous
  - ECDH – Diffie Hellman sobre curvas elipticas
  - Kerberos
- Hay variantes obsoletas que utilizan 512 bits
  - Definidas por problemas de exportación de material criptográfico en los EE.UU.

Hay ataques que se llaman downgrade attack, en donde un atacante intenta meterse en el handshake, y modificar el proceso de intercambio de clave, obligando a que las dos partes usen una variantes insegura de 512 bytes.

TLS - Criptografía

- Cifrado simétrico
  - ~~RC4~~
  - ChaCha20 (TLS 1.2)
  - ~~DES-CBC~~
  - Triple DES (3DES-EDE-CBC)
  - AES; AES-CBC, AES-CCM, **AES-GCM**
  - IDEA CBC
  - ARIA: ARIA-CBC, ARIA-CCM, ARIA-GCM
- También hay variantes obsoletas, con claves de 40 bits

TLS esta pensado para transferir cantidades masivas de informacion, entonces se necesita usar cifrados muy rapidos. El problema es que los cifrados asimetricos son uno o dos ordenes de magnitud mas lentos que los simetricos.
Generalmente, se usa cifrado asimetrico para intercambiar una clave, y el resto de la comunicacion se hace por cifrado simetrico.

Aria es un criptosistema que no es publico, y es del departamento de defensa de USA

TLS - Criptografía

- Hash
  - HMAC-MD5
  - HMAC-SHA1
  - HMAC-SHA2 (256/384)
  - AEAD (Authenticated Encryption with Associated Data)
- Autenticación
  - Firmas RSA
  - Firmas DSS

DSS es el mas comun

En resumen, TLS junta
- Criptosistemas simetricos
- Criptosistemas asimetricos
- Hash
- Firmas digitales

Sesión TLS

- Es una asociación entre dos pares
  - Puede soportar múltiples conexiones
- Información:
  - Identificador unico de sesion
  - Certificado X.509v3 del otro extremo (opcional)
  - Método de compresión acordado
  - Método de encriptación y MAC acordado
  - 'Master Secret': clave de sesión compartida de 48 bytes

Sesiones en SSL: Uso transparente por eficiencia.
NO es un protocolo de sesión!!!

Tecnicamente, segun las capas OSI, SSL si es un protocolo de sesion (porque va por arriba de TCP)

Conexión TLS

- Describe como intercambiar datos
- Contiene

A partir de la secuencia aleatoria y la clave de sesion, se generan las claves de escritura y las claves para MAC. La idea es similar a la que vimos en criptosistemas, para que todos los mensajes vayan con clave distinta.

  - Secuencía de bits aleatoria de calidad criptográfica
  - Claves de escritura para ambos lados (diferentes)
  - Claves para MACs (hashes con clave) para ambos lados
  - IVs si fuesen necesarios
  - Número de secuencia para cliente y servidor

SSL esta pensado para que lleguen todos los paquetes. Si no llega un paquete, se emite SSL alert y hay que comenzar de vuelta.

TLS Handshake – parte 1

- **ClientHello**:

El que inicia la comunicacion envia un ClientHello.
Manda un numero aleatorio (para evitar replays), un ID de sesion (que puede ser de una sesion nueva o ya existente), los protocolos de cifrado que soporta (por ejemplo TLS 1.1 para atras), y los algoritmos de compresion que conoce

- **ServerHello**:

El servidor devuelve un ServerHello. Segun las reglas que tenga configurada, elige un ID de sesion, y un cifrado y algoritmo de compresion de los que tenia disponible el cliente.

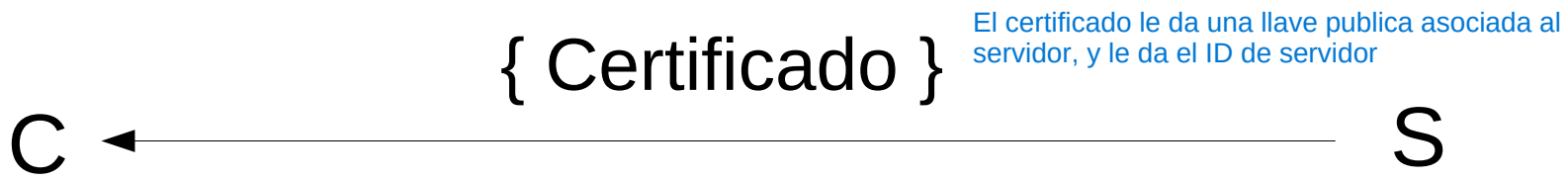
Si el cliente no tiene ciphers o comps que sean compatibles con el servidor, este ultimo emite un SSL alert

Problema: todo esto viaja en texto plano. Necesitamos que el servidor reciba el mensaje original del cliente y viceversa. Generalmente el servidor envia un certificado al cliente para que sepa que se esta conectando con el servidor correcto.

Vc = Version cliente V = min (Version Cliente, Version Servidor)
r1,r2 = nonces (Timestamp y 28 bytes aleatorios)
S_{id} = ID de la sesion. 0 para iniciar una nueva
Ciphers = Lista de stacks criptográficos disponibles
Cipher = Stack criptográfico elegido
Comps = Metodos de compresión disponibles
Comp = Método de compresión elegido

TLS Handshake – parte 2

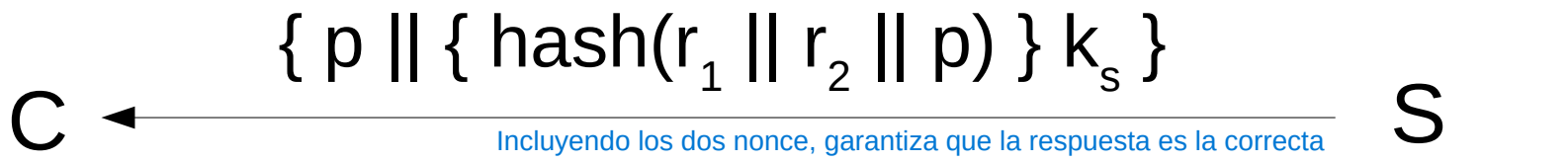
- Certificate (si el servidor es autenticado):



El certificado le da una llave publica asociada al servidor, y le da el ID de servidor

- ServerKeyExchange

Hay mas de un algoritmo de intercambio de claves



Incluyendo los dos nonce, garantiza que la respuesta es la correcta

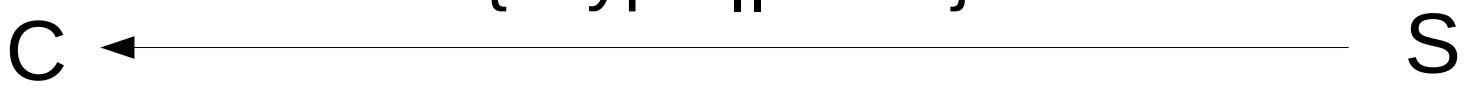
r1,r2 = nonces utilizados durante la parte inicial (evitan replay)
p = paramtros criptográficos (e,n – para RSA, p, g, g^a para DH, r para fortezza)
K_s=Clave privada del servidor correspoondiente a la pública del certificado

Es muy comun que r1, r2 vengan de a pares (en donde estan generadas por cada parte). Hay que tener en cuenta el caso de que el atacante no sea un tercero, sino que sea uno de los dos.

TLS Handshake – parte 2

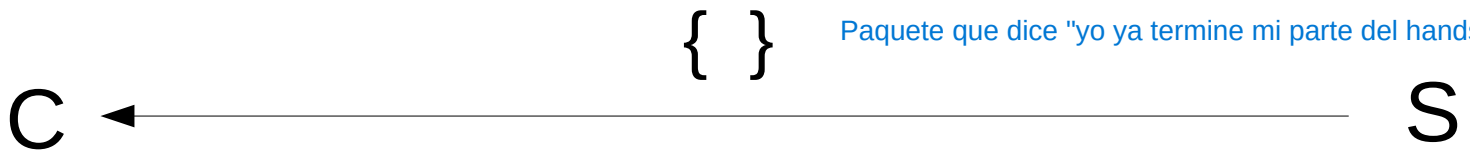
- Certificate request (si el cliente es autenticado):

Como opcional, el servidor le puede pedir un certificado al cliente. En este caso, el servidor le da al cliente los tipos de certificado que conoce (ej formato x509.2), y que entidades certificantes esta dispuesto a aceptar.



- ServerHelloDone

Paquete que dice "yo ya termine mi parte del handshake"



Ctype = tipo de certificados permitidos, desde el punto de vista algoritmico
Cas = lista de autoridades certificantes válidas

TLS Handshake – parte 3

- **Client Certificate (solo si fue solicitado)**

Si al cliente le piden un certificado, lo envia. Generalmente esto aparece en comunicacion servidor a servidor, en donde las dos partes se intercambian certificados (tambien llamado mutual TLS)

- **Client Key Exchange (version RSA)**

El cliente genera un PRE secret (secuencia aleatoria) y la cifra. Tambien se le agrega la version que se esta utilizando (para protegerse de ataques puntuales)

- **Client Key Exchange (version DH)**

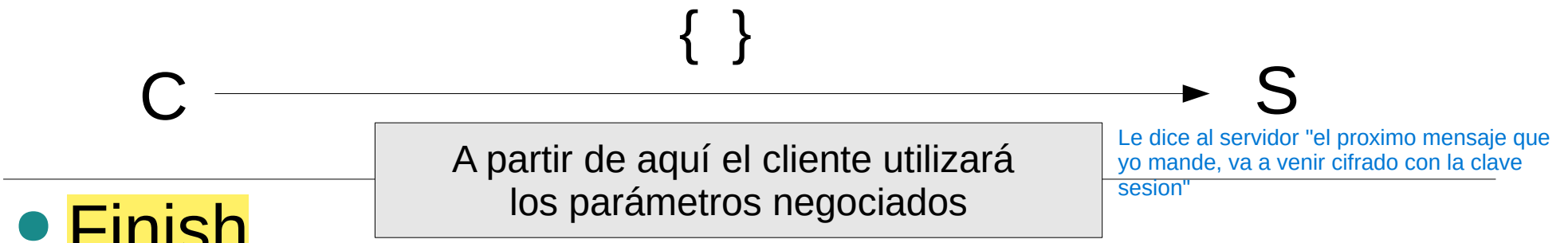
PRE = $g^{ab} \bmod p$

En el server key exchange recibio su numero, ahora va a calcular su parte y enviarla

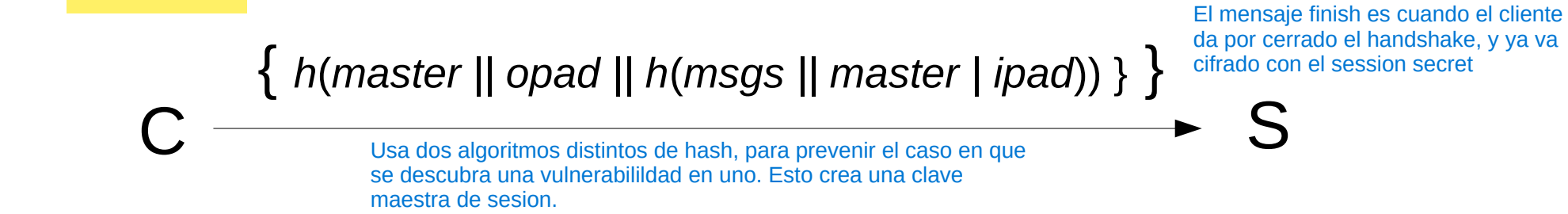
V = Version informada originalmente por el cliente (previene downgrade attacks)
g, p = parametros de DH informados por el servidor

TLS Handshake – parte 4

- Change Cipher Spec



- Finish



$$Master = MD5(pre \,||\, SHA('A' \,||\, pre \,||\, r_1 \,||\, r_2) \,||\, MD5(pre \,||\, SHA('BB' \,||\, pre \,||\, r_1 \,||\, r_2) \,||\, MD5(pre \,||\, SHA('CCC' \,||\, pre \,||\, r_1 \,||\, r_2)$$

Opad = 01011100 (binario, repetido durante todo un bloque – 20 bytes)
Ipad = 00110110 (binario, repetido durante todo un bloque – 20 bytes)
msgs = concatenación de todos los mensajes intercambiados hasta el momento

TLS Handshake – parte 4

- Change Cipher Spec

- Finish

El servidor da por cerrada la sesion, y manda su version del finish ya cifrada con la session key

NOTA: todos estos pasos del handshake se ejecutan ANTES de enviar el primer bit de datos

TLS Change Cipher Spec

- Utilizado durante el handshake
- Puede aparecer en cualquier momento
- Implica una renegociación (cambio) de las claves de sesion utilizadas
- Lo puede solicitar tanto el cliente como el servidor
- No tiene contenido. Es un tipo de mensaje del protocolo

TLS Alert

- Envía eventos fuera de banda
  - Pueden ser de advertencia o fatales
  - Un evento fatal invalida la conexión
- CloseNotify: Evento que indica el fin de una sesión
    - No se enviarán mensajes nuevos
    - Si llega un mensaje nuevo será ignorado

TLS Alert

- Errores fatales
  - Mensaje no esperado,
  - MAC incorrecto,
  - error al descomprimir,
  - error en el handshake,
  - parámetro ilegal
- Advertencias o errores (a elección de quien lo recibe y el contexto):
  - No hay certificado, certificado no válido
  - Certificado no soportado, expirado o revocado

TLS - Panorama

- Estandar de comunicación segura en servicios web (https)
- Depende de infraestructura de PKI
- No es utilizado frecuentemente para validar clientes (es mas servidores que se validan entre si)
- Soportado por todos los navegadores
  - Diferente respuesta frente a alertas
- Problemas de diseño (en los clientes) hicieron que pierda parte de su utilidad
  - Aún así, es la alternativa mayormente adoptada.

Cada sistema operativo implementa su propio stack TCP/TLS.
Por este motivo, hay tecnicas de fingerprinting que permiten adivinar que SO/version de kernel esta utilizando un host

Lectura Recomendada

Capítulo 11
Computer Security Art and Science

Matt Bishop

RFC 5246 – TLS v1.2
http://tools.ietf.org/html/rfc5246

Descripción de vulnerablidad encontrada en la renegociación de claves
http://www.g-sec.lu/practicaltls.pdf