Tema 4 Sistemas Distribuidos. Seguridad

Los objetos y principales en la seguridad de un SD son:

* **Objeto o recurso:** buzón de correo, sistema de archivo, parte de una web…
* **Principal:** usuario o proceso que tiene permiso para realizar acciones. La identidad del principal es importante.

El enemigo que ataca a la seguridad de nuestro sistema distribuido realizará:

* **Ataques:** en aplicaciones que manejan transacciones comerciales u otra información cuyo secreto o integridad es crucial.
* **Amenazas:** sobre todo a procesos, a los canales de comunicación, denegación de servicio.

Los canales seguros en nuestro sistema distribuido tendrá:

* Propiedades
  + Cada proceso está seguro de la identidad del otro.
  + Los datos son privados y protegidos contra la manipulación.
  + Protección contra repeticiones y reordenación de datos.
* Utiliza criptografía.
  + El secreto se preserva mediante ocultamiento criptográfico.
  + La autenticación basada en la prueba de posesión de secretos.
  + **Ocultamiento criptográfico:** basado en confusión y difusión.
  + **Posesión de secretos:** claves convencionales compartidas. Pares de claves públicas/privadas.

## Amenazas y formas de ataque a nuestra seguridad.

**Escuchar a escondidas:** obteniendo información privada o secreta.

**Enmascararse:** asumiendo la identidad de otro usuario/principal.

**Manipular mensajes:** alterando el contenido de mensajes en tránsito.

**Reenviar:** almacenando mensajes seguros y enviándolos más tarde.

**Negación de servicio:** inundando un canal u otro recurso, negando acceso para los otros.

Algunas de las amenazas que superan los canales seguros son:

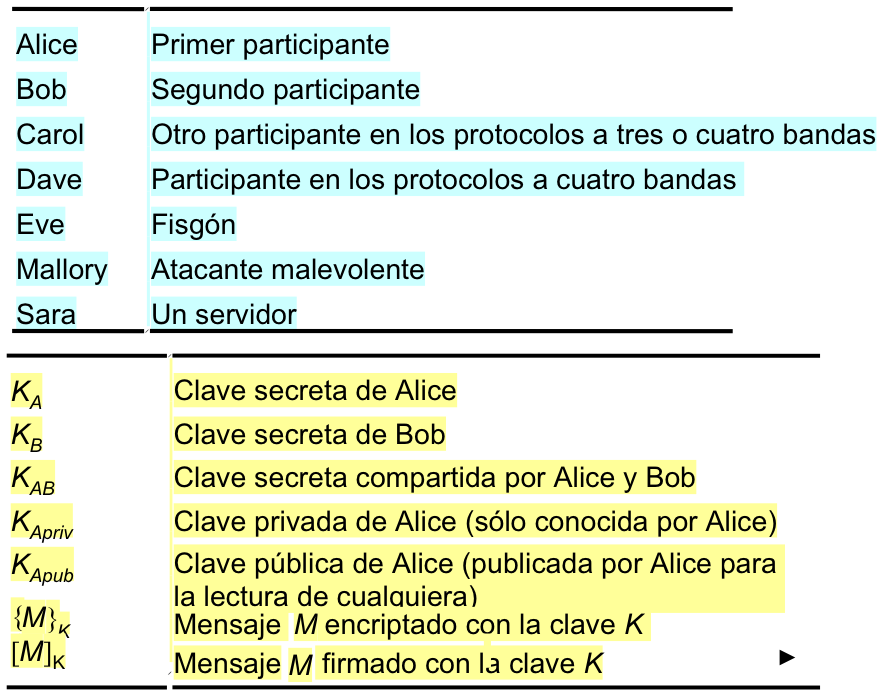
**Ataques de negación de servicio.** El uso excesivo de recursos hasta el grado de impedir su uso a usuarios legítimos. Amazon y Yahoo en Febrero de 2000. PSN en Diciembre de 2014.

**Los caballos de Troya y otros virus.** Los virus sólo pueden entrar en computadoras cuando el código de programa es importado.

Pero los usuarios a menudo requieren programas nuevos.

Las defensas que se presentan a estas amenazas son entre otras: la autenticación de código mediante firmas, la validación de código con comprobación de tipo, seguridad JVM…ANÁLISIS, DISEÑO Y PRUDENCIA.

## NOMENCLATURA EN TÉCNICAS DE SEGURIDAD.



### Escenario 1: secreto con clave compartida.

Alice y Bob comparte una clave secreta Kab

1. Alice usa Kab y acuerda una función de encriptación E(Kab, M) para codificar y enviar una serie de mensajes {Mi}Kab.
2. Bob lee los mensajes encriptados usando la correspondiente función D(Kab, M).

Alice y Bob pueden funcionar con Kab mientras estén seguros que Kab no es conocida.

Algunos problemas que nos encontramos son:

* Distribución de clave. ¿Cómo envía Alice una clave compartida a Bob de forma segura?.
* Caducidad de la comunicación. ¿Cómo sabe Bob que el mensaje no es una copia capturada por Mallory y reenviada más tarde?.

### Escenario 2: autenticación con servidor.

**Un ticket es un mensaje encriptado conteniendo la identidad del principal solicitante y una clave compartida para la sesión.**

Bob es un servidor de ficheros; Sara es un servidor de autenticación. Sara comparte Ka con Alice y Kb con Bob.

1. Alice envía un mensaje no encriptado a Sara identificándose y solicitando un ticket para acceder a Bob.
2. Sara responde a Alice con {{Ticket}Kb, Kab}ka. Consistente en un mensaje codificado según Ka con un ticket (para comunicar con Bob para cada fichero) encriptado según Kb y una nueva clave Kab.
3. Alice usa Ka para desencriptar la respuesta.
4. Alice envía a Bob el ticket, su identidad y una respuesta R para acceder al fichero: {Ticket}Kb, Alice, R.
5. El ticket es realmente {Kab, Alice}Kb. Bob usa Kb para desencriptarlo, chequea la identidad y usa Kab para encriptar las respuestas a Alice.

Este proceso es una simplificación del protocolo Needham and Schroeder (y Kerberos). No es válido para comercio electrónico.

### Escenario 3: autenticación con clave pública.

Bob genera un par de claves pública/privada <KBpub, KBpriv>.

1. Alice obtiene un certificado firmado por una autoridad de confianza que posee la clave pública de Bob, KBpub.
2. Alice crea una clave compartida KAB, la encripta según KBpub un algoritmo de clave pública y envía el resultado a Bob.
3. Bob usa KBpriv para desencriptar KAB.

(Si desean asegurar que el mensaje no ha sido manipulado, Alice puede incluir algún dato aceptado por ambos y Bob chequearlo).

Mallory puede interceptar la solicitud de certificado de clave pública y enviarle su propia clave pública, pudiendo desencriptar el resto de mensajes. La firma digital lo impide.

### Escenario 4: firma digital con resumen seguro.

Alice quiere publicar un documento M de forma que cualquiera pueda verificar su procedencia.

1. Alice calcula un resumen de longitud fija del documento Resumen(M).
2. Alice encripta el resumen con su clave privada, lo adjunta a M y hace el resultado (M, {Resumen(M)}KApriv) público.
3. Bob obtiene el documento firmado, extrae M y computa Resumen(M).
4. Bob usa la clave pública de Alice para desencriptar {Resumen(M)} KApriv y lo compara con el resumen calculado por él. Si coincide, entonces la firma es válida.

La función de resumen debe ser segura frente al “ataque del cumpleaños”.

### Función de resumen seguro h = H(M):

1. Dado M, debe ser fácil calcular h.
2. Dado h, debe ser muy difícil calcular M.
3. Dado M, deber ser difícil encontrar otro M’, tal que H(M) = H(M’).

También llamada función de dispersión de un solo sentido.

### Paradoja del cumpleaños.

La probabilidad de encontrar un par idéntico en un conjunto es mucho mayor que la de encontrar la pareja para un individuo dado. Estos son los elementos:

* A = {al menos dos personas celebran su cumpleaños a la vez}.
* Ac = {no hay dos personas que celebren su cumpleaños a la vez}.
* P(A) = 1 – P(Ac).
* P = Casos favorables / Casos posibles.

El ataque del cumpleaños se produce cuando:

1. Alice prepara dos versiones M y M’ de un contrato para Bob. M favorable y M’ desfavorable.
2. Alice fabrica varias versiones de M y M’ sutilmente diferentes (espacios al final de línea,…). Ella compara los valores de dispersión de todos los M con todos los M’ buscando un par igual.
3. Alice envía el contrato favorable M a Bob, éste lo firma digitalmente usando su clave privada.
4. Cuando lo devuelve, Alice sustituye M por M’, pero manteniendo la firma de Bob sobre M.

Por ejemplo, que para generar colisiones en una función aleatoria perfecta (en funciones hash) de n bits, con una probabilidad del 50% aproximadamente, se requieren solo 2n/2 intentos.

**Certificado:** sentencia firmada por un principal que sirve de credencial y/o autenticación. Un certificado necesita:

* Un formato estándar acordado.
* Acuerdo sobre la forma en que se construyen las cadenas de certificados.
* Fechas de expiración, de forma que pueda ser revocado.

## ALGORITMOS CRIPTOGRÁFICOS

Mensaje: M, clave: K, funciones criptográficas E,D.

* Simétricos (clave secreta).
  + E(K, M) ={M}k D(K, E(K, M)) = M

La misma clave para E y D.

M debe ser difícil de computar si se desconoce K.

La forma usual de ataque es la fuerza bruta. Resistente haciendo K suficientemente grande ~128 bits.

* Asimétricos (clave pública)
  + Claves de encriptación y desencriptación separadas: Ke , Kd.

D(Kd \* E(Ke, M)) = M.

Se basa en el uso de funciones de puerta false, E tiene un alto coste computacional. Las claves son muy grandes > 512 bits.

* Protocolos híbridos (usados en SSL actualmente llamado TLS).
  + Usa criptografía asimétrica para transmitir la clave simétrica que es usada para encriptar la sesión.

## CIFRADORES DE BLOQUE, DE CADENA Y DE FLUJO.

La mayoría de cifradores trabajan sobre bloques de 64 bits. La debilidad de un cifrador de bloques simple es que los patrones repetidos pueden ser detectados.

La conexión debe ser fiable, no se pueden perder bloques.

## ALGORITMOS DE ENCRIPTACIÓN SIMÉTRICA

Todos estos algoritmos realizan operaciones de confusión y de difusión sobre bloques de datos binarios.

* **TEA**: desarrollado en la Universidad de Cambridge en 1994. Muy simple pero efectivo. Triple de veloz que el DES.
* **DES**: US Data Encryption Standard (1977). El original no era demasiado fuerte. Debido al coste computacional se implementó VLSI. En 1997 fue derrotado por fuerza bruta por un consorcio de usuarios de internet. En 1998 una máquina de EFF podía resolver claves DES en 3 días.
* **Triple-DES**: aplica DES tres veces con dos claves distintas.
* **IDEA**: International Data Encryption Algorithm (1990). Parecido al TEA.
* **AES**: US Advanced Encryption Standard (1997).

## ALGORITMOS DE ENCRIPTACIÓN ASIMÉTRICA

Todos ellos dependen del uso de funciones de puerta falsa, que son funciones de un solo sentido con una salida secreta.

* **RSA**: el primer algoritmo práctico (Rivest, Shamir y Adelman 1978) y el más frecuentemente usado. El tamaño de la clave puede variar. Para encriptar según RSA, el texto se divide en bloques de k bits donde 2k < N (el valor numérico de un bloque es siempre menos que N; k entre 512 y 1024).
* **Curvas elípticas:** método reciente, claves más cortas y más veloz.

Los algoritmos asimétricos son ~1000 veces más lentos y no son prácticos para encriptaciones masivas; sin embargo, sus propiedades los hacen idóneos para distribución de claves y para autenticación.

## ALGORITMOS DE RESUMEN SEGURO

Cualquier algoritmo simétrico se puede usar en CBC (cifrador de cadena), donde el último bloque es el resumen H(M).

* MD5: desarrollado por Rivest (1992). Calcula un resumen de 128 bits. Cuatro vueltas con una de cuatro funciones no lineales sobre cada 32 bits de un bloque de 512 bits de texto.
* SHA: (1995) basado en MD4 de Rivest, pero más seguro, produce un resumen de 160 bits.

## PROTOCOLO NEEDHAM-SCHROEDER

En los primeros sistemas distribuidos (1974-84) era difícil proteger los servidores. No había mecanismos de autenticación del origen de las peticiones y la criptografía pública no estaba disponible.

Needham y Schroeder desarrollaron un protocolo de autenticación y distribución de claves para uso en red local:

* Supuso un primer ejemplo del cuidado en el diseño de protocolos de seguridad.
* Introdujeron varias ideas de diseño.

La autenticación de clave secreta sigue el siguiente proceso:

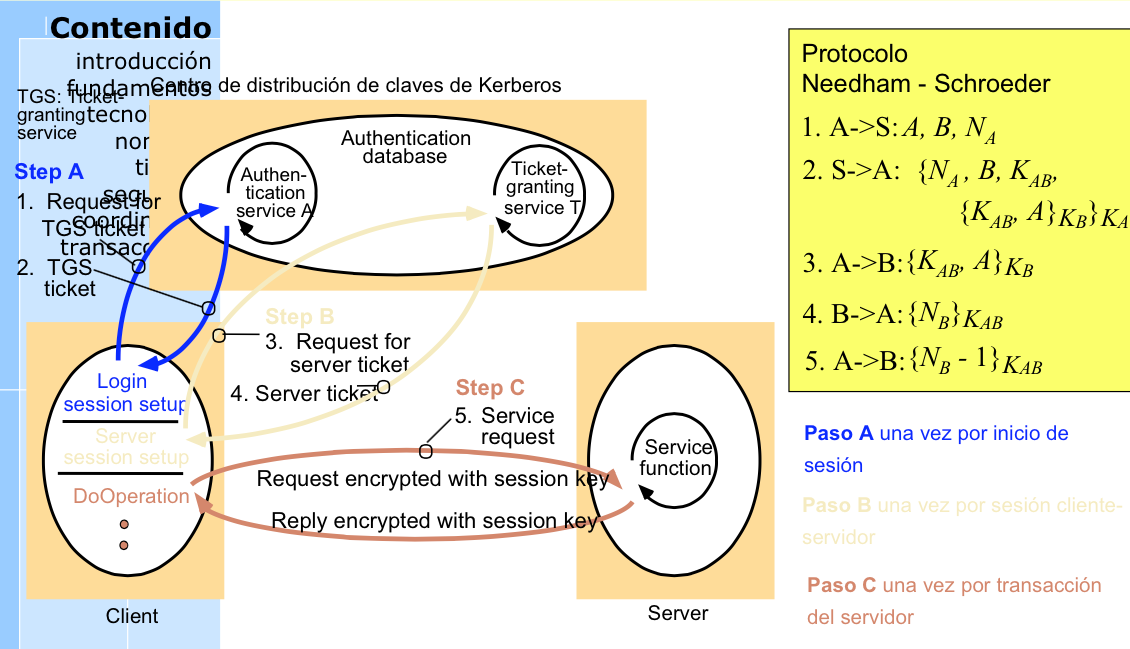
1. A solicita una clave a S para comunicarse con B.
2. S devuelve un mensaje encriptado en la clave secreta de A, con una clave nueva KAB y un “ticket” encriptado en la clave secreta de B. La ocasión NA demuestra que el mensaje fue enviado en respuesta al anterior. A confía en que S envió el mensaje porque sólo S conoce la clave secreta de A.
3. A envía el ticket a B.
4. B desencripta el ticket y utiliza la nueva clave KAB para encriptar otra ocasión NB.
5. A demuestra a B que fue el emisor del mensaje anterior devolviendo una transformación acordada sobre NB.

NA es una ocasión: entero que se añaden a los mensajes para demostrar la frescura de la transacción. Son generador por el proceso emisor cuando se necesita.

## KERBEROS

Comunicación segura con servidores en una red local. Desarrollado para ofrecer seguridad en la red del campus > 5000 usuarios, basado en Needham-Shroeder.

Estandarizado e incluido en muchos sistemas operativos. El servidor Kerberos crea una clave secreta compartida para cada servidor solicitado y la envía encriptada al computador del usuario. El password del usuario es el secreto compartido inicial en Kerberos.



## GUÍAS DE DISEÑO. ASUNCIÓN DEL PEOR CASO.

Las interfaces están desprotegidas, y por tanto un atacante puede enviar un mensaje a cualquier interfaz.

Las redes son inseguras. Para ello hemos de limitar el tiempo y el alcance del secreto, que se usará una sola vez y tendrá caducidad.

Los algoritmos y códigos están disponibles. Lo mejor es publicar los algoritmos criptográficos empleados.

Los atacantes tienen acceso a suficientes recursos. Hay que presuponer que a lo largo de la vida del sistema se desarrollarán computadores mucho más potentes.

Minimizar la base confiable.