

## Protocolos de Seguridad

Un escenario típico consiste de un número de *principales*, tales como individuos, compañías, computadoras, lectores de tarjetas magnéticas, los cuales se comunican usando una variedad de canales (teléfono, correo electrónico, radio ...) o dispositivos físicos (tarjetas bancarias, pasajes, cédulas ...).

Un *protocolo de seguridad* define las reglas que gobiernan estas comunicaciones, diseñadas para que el sistema pueda soportar ataques de carácter malicioso.

Protegerse contra todos los ataques posibles es generalmente muy costoso, por lo cual los protocolos son diseñados bajo ciertas premisas con respecto a los riesgos a los cuales el sistema está expuesto. La evaluación de un protocolo por lo tanto envuelve dos preguntas básicas: ¿Es el modelo de riesgo realista? ¿El protocolo puede controlar ese nivel de riesgo?

## **Ejemplo:** Autenticación de una alarma de carro

### *Versión 1:*

- Un transmisor de radio  $T$  en el llavero transmite un número de serial, que es reconocido por un receptor ( $R$ ) en el carro:

$$T \rightarrow R: N_T$$

### *Problemas:*

1. Un adversario puede captar el número con su propio receptor, y usarlo en otro momento para robar el carro. Conocido como el “método del grabador”.
2. Si el rango de números válidos es pequeño, el adversario puede generar todas las posibilidades en poco tiempo. Las versiones originales eran de 16 bits. Se crearon dispositivos que podían probar 10 combinaciones por segundo. En un estacionamiento con 100 carros, el tiempo promedio para que uno de ellos reaccionara es menos que un minuto.

*Versión 2:*

- Aumentar el número de bits de 16 a 32. Ahora tardaría  $2^{16}$  veces más tiempo tener “suerte”.

*Problema:*

- El método del grabador sigue funcionando.

*Versión 3:*

1. El transmisor envía un número serial y un *bloque de autenticación* que consiste del mismo serial seguido por un número aleatorio, todo encriptado usando una clave que es única para este par transmisor/receptor:

$$T \rightarrow R: S_T, (S_T, N)_{K_T}$$

- El número aleatorio  $N$  es usado una sola vez, por lo cual se le llama un *nonce*.
  - $S_T$  se manda dos veces. ¿Porqué?
2. Para verificar, el receptor lee  $S_T$ , consigue la clave correspondiente  $K_T$ , descifra el resto del mensaje, chequea que el texto descifrado contenga  $S_T$  y finalmente que el nonce  $N$  no haya sido usado antes, lo cual garantiza que no se trata de un ataque tipo grabador.

Un *protocolo* es una serie de pasos, que involucra a dos o mas principales, diseñado para realizar una tarea particular.

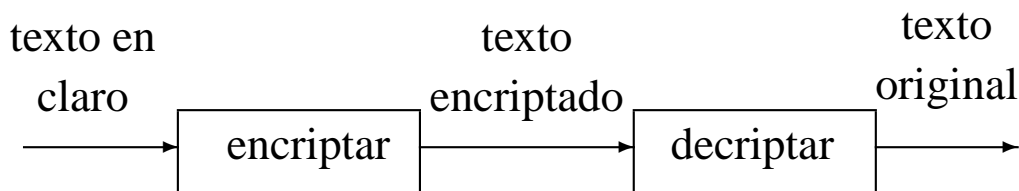
1. Todos los principales deben conocer los pasos del protocolo de antemano.
2. Todos deben estar de acuerdo en seguir el protocolo.
3. El protocolo no admite ambigüedades.
4. El protocolo debe ser completo – define qué hacer en cualquier circunstancia posible.
5. No debe ser posible hacer más (o aprender más) que lo que el protocolo define.

Un *protocolo criptográfico* es un protocolo que usa funciones criptograficas en algunos o todos los pasos.

## Terminología Criptográfica

Un mensaje en su estado original consiste de **texto en claro** (*plaintext, cleartext*). Se disfraza por un proceso de **encriptamiento** (*encryption*), el cual produce **texto encriptado** (*cyphertext*). El proceso inverso es **decriptamiento** (*decryption*). (Tambien **encifrado, decifrado**))

**Criptografía** es la ciencia de la seguridad de mensajes, practicado por **criptógrafos**. La ciencia (o arte) de violar la seguridad de mensajes es **criptanálisis**, practicado por **criptanalistas**. **Criptología** es una rama de las matemáticas que estudia ambos aspectos. Lo practican los **criptólogos**.



$M$  representa el texto en claro (no necesariamente es “texto”),  $C$  el texto encriptado.  $C$  puede ser más largo que  $M$ .

Encriptamiento es una función:  $E(M) = C$ , y decriptamiento es otra:  $D(C) = M$ .

Además, se requiere que:  $D(E(M)) = M$ .

## Algoritmos y Claves

Las funciones  $E$  y  $D$  son implementadas por un **algoritmo criptográfico**, o **cifrado** (*cypher*).

El algoritmo no debe depender por su seguridad, del hecho que sea desconocido por el adversario (esto se llama la *Ley de Kirchoff*). Por lo tanto, todo algoritmo moderno depende de una o más **claves** (*keys*), escogidas al azar de un espacio grande (el *keyspace*).

¿Porqué “al azar”?

¿Porqué “grande”?

Un **criptosistema** consiste de un algoritmo más todos los posibles textos (en claro y encriptados) y claves.

Los algoritmos se dividen en dos clases:

- **Simétricos**, también *de una clave, de clave secreta*.

$$E_K(M) = C$$

$$D_K(C) = M$$

$$D_K(E_K(M)) = M$$

(Ojo: no necesariamente  $E_K(D_K(X)) = X$ )

Hay dos categorías de algoritmos simétricos:

**stream:** Puede aplicarse en forma continua a una entrada semi-infinita.

**block:** Opera sobre bloques de entrada de tamaño fijo.

- **Asimétricos**, también *de dos claves, de clave pública/privada*.

La clave pública es para encriptar, la privada para decriptar.

$$E_{K_1}(M) = C$$

$$D_{K_2}(C) = M$$

$$D_{K_2}(E_{K_1}(M)) = M$$

Los asimétricos siempre son de tipo “block”.

A veces se usan “al revés” como parte de un mecanismo de *firmas digitales*.

Dejaremos la discusión de algoritmos específicos para más adelante.

En la literatura criptográfica se suelen usar algunos personajes para simplificar la explicación de los protocolos:

**Alicia y Bob:** Principales en todos los protocolos.

**Carol y Dave:** Principales en protocolos que envuelven tres o cuatro.

**Eva:** Participante que escucha sin autorización en un canal.

**Manuel:** Atacante activo, modifica mensajes y actúa en el medio del protocolo.

**Arturo:** Arbitro confiable.

**Guillermo:** Guardia, estará protegiendo a Alicia y Bob en algunos protocolos.

**Víctor:** Verificador.

## Sistemas de Desafío y Respuesta

Los sistemas modernos de seguridad y alarmas para carros usan un protocolo con dos pasos de *desafío-respuesta* (*challenge-response*). Cuando la llave es insertada, una unidad de control en el motor envía un *desafío*, el cual consiste de un número aleatorio de  $n$  bits. La llave calcula una respuesta encriptando el desafío:

$$\begin{aligned} E \rightarrow T: & N \\ T \rightarrow E: & \{S_T, N\}_{K_T} \end{aligned}$$

Esto funciona si  $K_T$  es conocida solamente por E y T y si  $N$  es realmente aleatorio. En algunas implementaciones  $N$  era tomada de una secuencia *pseudo-aleatoria*, entonces Eva podía hacer esto:

1. Usar su grabadora para escuchar  $N_i$  cuando Alicia abría su carro.
2. Generar  $N_{i+1}$  y esperar que Alicia estacionara nuevamente.
3. Interrogar el llavero en la cartera de Alicia con  $N_{i+1}$  y copiar la respuesta.
4. Usar la respuesta para contestar el próximo desafío del carro.

## Taxonomía de Protocolos

Los protocolos pueden ser clasificados en base a su modo de funcionamiento en:

*Protocolos arbitrados:* Aquellos en los que es necesaria la participación de un tercero para garantizar la seguridad del sistema. *Ejemplo:* método de compra y venta usando un notario.

Ejemplo: Alicia quiere vender su carro a Bob. Bob quiere pagar con cheque. Ninguno de los dos confía en el otro:

- Se consigue un **árbitro** (Arturo) en que ambos confían (ej. un notario).
- Alicia entrega el título de propiedad a Arturo.
- Bob entrega el cheque a Alicia.
- Alicia deposita el cheque y espera un tiempo determinado. Si el cheque rebota, Arturo devuelve el título a Alicia.
- En caso contrario, Arturo entrega el documento a Bob.

Los protocolos arbitrados pueden ser ineficientes, o incluso vulnerables a ataques contra el árbitro.

*Protocolos adjudicados:* Son protocolos que tienen dos partes, una parte no-arbitrada y una arbitrada. El subprotocolo arbitrado se activa solo en caso de disputa.

Ejemplo: Alicia y Bob quieren suscribir un contrato:

- Negocian los términos
- Alicia firma
- Bob firma
- Si hay una disputa, apelan ante un **juez**, quien resuelve el problema.

*Protocolos auto-reforzados o autoadjudicados:* No se requiere de un árbitro para garantizar el protocolo. Se puede abortar la secuencia de pasos en cualquier momento, dejando sin efecto las acciones tomadas (similar a una transacción atómica). No todas las situaciones son apropiadas para este tipo.

Los **ataques** contra protocolos pueden ser *activos* o *pasivos*. Si el adversario es uno de los participantes, se trata de una **trampa**.

### *Ejemplo: Acordando una Clave de Sesión*

Las *claves de sesión* buscan reducir la oportunidad de ataques, usando una clave nueva cada vez que se abre una nueva sesión de intercambio de mensajes.

Evidentemente la clave no puede ser transmitida en claro sobre el canal.

Entonces, necesitamos un *protocolo inicial* para acordar una nueva clave (aleatoria) para cada sesión.

El protocolo inicial también tiene que ser seguro.

Una posibilidad podría ser usar claves públicas para el protocolo inicial (en general son demasiado ineficientes para la sesión misma).

## **El Hombre en el Medio**

En un intento de usar un protocolo autoadjudicado, Alicia y Bob deciden usar sus claves públicas para el intercambio inicial, pero Manuel está escuchando:

1. Alicia le manda a Bob su clave pública.
2. Manuel intercepta esta clave y envía a Bob su propia clave pública.
3. Bob le envía a Alicia su clave pública, Manuel la intercepta y le envía a Alicia su propia clave pública.
4. Cuando Alicia envía a Bob un mensaje encriptado con su clave pública realmente usa la de Manuel, quien puede descryptar el mensaje y luego encriptarlo usando la clave pública de Bob.
5. Cuando Bob envía un mensaje a Alicia, Manuel puede hacer lo mismo que con Alicia.

Este tipo de ataque se conoce como *Man In The Middle*. El problema es que Bob cree, sin evidencia, que el primer mensaje que recibe proviene de Alicia, y vice versa. Sin soporte directo o indirecto de un tercero, en que ambos confían, esto nunca puede ser seguro.

Resignados a usar un protocolo arbitrado, Alicia y Bob comparten cada uno una clave secreta (permanente) con Arturo, en que ambos confían. Esta vez todo se hace con criptografía simétrica.

1. Alicia llama a Arturo y le solicita un clave de sesión para comunicarse con Bob.
2. Arturo genera una clave de sesión aleatoria y encripta dos copias de esta clave, una usando la clave que comparte con Alicia y la otra usando la clave que comparte con Bob.
3. Arturo le envía ambas copias a Alicia. Alicia desencripta su copia de la clave de sesión.
4. Alicia le envía a Bob su copia de la clave de sesión (encriptada por Arturo).
5. Bob desencripta el mensaje de Arturo y recupera a su vez su clave de sesión.
6. Bob y Alicia usan esta clave de sesión para comunicarse en forma segura.

El protocolo **Needham-Schroeder** es la versión formal del anterior:

1. Alicia manda  $\{A, B, N_A\}$  al servidor Arturo, donde  $N_A$  es un número aleatorio.
2. Arturo genera una clave de sesión aleatoria  $K_{AB}$ , y responde a Alicia con:  $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$ , donde  $K_{AS}$  es una clave secreta compartida entre Alicia y Arturo, y  $K_{BS}$  es otra, compartida entre Bob y Arturo.
3. Alicia decripta y saca  $K_{AB}$ . Confirma que  $N_A$  es el mismo número que había enviado a Arturo. Manda a Bob:  
 $\{K_{AB}, A\}_{K_{BS}}$ .
4. Bob decripta y saca  $K_{AB}$ . Genera otro valor aleatorio  $N_B$  y lo devuelve a Alicia:  $\{N_B\}_{K_{AB}}$ , usando la nueva clave de sesión.
5. Alicia decripta usando  $K_{AB}$ . Calcula  $N_B - 1$  y lo manda encriptado a Bob:  $\{N_B - 1\}_{K_{AB}}$ .
6. Bob decripta y verifica que el número es  $N_B - 1$ .

$N_A$  y  $N_B$  son **nonces**, y existen para evitar **ataques de “replay”** en que Manuel graba mensajes y los reproduce después para tratar de engañar.

Needham-Schroeder tiene una debilidad si Manuel logra obtener acceso a una vieja clave de sesión  $K_{AB}$ :

1. Manuel manda esto a Bob:  $\{K_{AB}, A\}_{K_{BS}}$ .
2. Bob manda a Alicia:  $\{N_B\}_{K_{AB}}$  pero Manuel intercepta.
3. Manuel decripta con  $K_{AB}$  y manda a Bob:  $\{N_B - 1\}_{K_{AB}}$ .
4. Bob “verifica”, pensando que está hablando con Alicia.

Para vencer este ataque, es necesario verificar que  $K_{AB}$  es *fresca*. Una versión modificada de N-S utiliza timestamps y define un período dentro del cual  $K_{AB}$  es válida. Esto requiere relojes sincronizados entre las partes – que no es trivial (los mensajes de sincronización son sujetos a interferencia también).

Algo más serio ocurre si  $K_{AS}$  es robada. Aún en caso que Alicia lo detecta, queda el problema de la *revocación* de todas las claves de sesión generadas usando la clave robada, incluyendo claves generadas por Manuel fingiendo ser Alicia ante Arturo, de las cuales Alicia no tiene ninguna información.

Podemos usar la criptografía pública junto con el árbitro:

1. Alicia pide a Arturo la clave pública de Bob. Arturo la envía sin encriptamiento (es pública ...).
2. Alicia genera una clave de sesión aleatoria, la encripta usando la clave pública de Bob y la envía a Bob.
3. Bob desencripta el mensaje de Alicia usando su clave privada.
4. Posteriormente ambos se comunican usando la clave de sesión aleatoria generada por Alicia.

*(Ojo: Tanto en este caso como en el anterior, los mensajes propios de la sesión se envían usando criptografía simétrica.)*

Manuel puede atacar esto si logra interceptar la comunicación entre Alicia y Arturo. ¿Cómo sabe Alicia que está hablando con Arturo? Dejamos la respuesta para después ...

## Análisis Formal de Protocolos

Es evidente que los protocolos de seguridad son sutiles. Para tratar de ganar más confianza, se ha propuesto usar métodos formales para definirlos, y para probar sus propiedades.

La **Lógica BAN** es uno de los métodos más populares. Fue impulsado principalmente por los aportes de Michael Burrows, Martin Abadi y Roger Needham, quienes desarrollaron un modelo de lógica formal para el análisis del conocimiento y las creencias.

La notación BAN incluye:

- **A cree**  $X$ .  $A$  tiene razones para creer que  $X$  sea cierto.
- **A dijo**  $X$ .  $A$  alguna vez dijo que  $X$  era cierto, posiblemente en el pasado remoto.
- **A controla**  $X$ .  $A$  tiene jurisdicción sobre  $X$ . Es la autoridad sobre  $X$  en que se puede confiar.
- **A ve**  $X$ .  $A$  puede ver un mensaje que contiene  $X$ , y lo puede repetir ( $X$ , no el mensaje).
- **fresco**( $X$ ).  $X$  es fresco, o sea contiene evidencia (quizás un *timestamp*) que indica que fue declarado por el principal correspondiente durante la sesión actual del protocolo.
- $\{X\}_K$ .  $X$  es encriptado con la clave  $K$ .
- $A \xleftrightarrow{K} B$ .  $K$  es una clave compartida entre  $A$  y  $B$ , con la cual pueden comunicarse.

Además, existe un conjunto de *reglas*, por ejemplo (entre otras):

- La *Regla del Significado de Mensajes*: si  $A$  ve un mensaje encriptado con  $K$ , y  $K$  es una buena clave para comunicarse con  $B$ , entonces  $A$  creerá que  $B$  alguna vez emitió el mensaje:

$$\frac{A \text{ cree } A \xleftarrow{K} B, A \text{ ve } \{X\}_K}{A \text{ cree } B \text{ dijo } X}$$

- La *Regla de Verificación de Nonces*: si un principal emitió un mensaje, y dicho mensaje es fresco, entonces el principal todavía cree el mensaje:

$$\frac{A \text{ cree fresco}(X), A \text{ cree } B \text{ dijo } X}{A \text{ cree } B \text{ cree } X}$$

- La *Regla de Jurisdicción*: si un principal cree algo, y es una autoridad sobre ese algo, entonces debemos creerle:

$$\frac{A \text{ cree } B \text{ controla } X, A \text{ cree } B \text{ cree } X}{A \text{ cree } X}$$

Ahora podemos aplicar el formalismo al análisis de un protocolo sencillo. El protocolo COPAC es usado por Visa en algunos sistemas de pago con tarjeta inteligente que deben funcionar sin conexiones al banco. El modelo se parece a un cheque, transferido desde la tarjeta del cliente  $C$  hasta la del comerciante  $M$ :

$$C \rightarrow M: \{C, N_C\}_K$$

$$M \rightarrow C: \{M, N_M, C, N_C\}_K$$

$$C \rightarrow M: \{C, N_C, M, N_M, X\}_K$$

$N_C$  y  $N_M$  son números seriales de transacciones del cliente y comerciante, respectivamente.  $X$  representa el cheque, que contiene su propio código de autenticación (no representado aquí). La clave  $K$  es cableada dentro de todas las tarjetas y no varía.

**Nota:** cada mensaje repite toda la información en el mensaje anterior. Esto sirve para evitar ataques de “cortar-y-pegar” a los mensajes.

Queremos probar, por ejemplo, que el comerciante puede confiar en el cheque, es decir  $M$  **cree**  $X$ .

Esto se podrá deduce de la Regla de Jurisdicción, si resulta que  $M$  **cree**  $C$  **controla**  $X$ , y  $M$  **cree**  $C$  **cree**  $X$ .

La primera parte se cumple porque sólo  $C$  puede generar un mensaje de tipo  $\{C, \dots\}_K$ , debido a la construcción de la tarjeta.

La segunda parte se deduce de la Regla de Verificación de Nonces, siempre que **fresco**( $X$ ) y  $M$  **cree**  $C$  **dijo**  $X$ .

**fresco**( $X$ ) se cumple porque ocurre en el tercer mensaje del protocolo, el cual contiene  $N_M$ , que es fresco. La segunda condición se cumple por la misma razón de la construcción de la tarjeta.