

Principios de diseño en seguridad de sistemas

Por: Luis Daniel Benavides

La presentación

- Basada en el capítulo 11 del libro ***Principles of Computer System Design*** por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Recursos en línea

- Segunda parte del libro (incluido cap. 11):
<http://ocw.mit.edu/resources/res-6-004-principles-of-computer-system-design-an-introduction-spring-2009/>
- Libro de Texto del curso: 6.033 de MIT
“Computer Systems Engineering”

Information security. The protection of information and information systems against unauthorized access or modification of information, whether in storage, processing, or transit, and against denial of service to authorized users.

— Information Operations. Joint Chiefs of Staff of the United States Armed Forces, Joint Publication 3-13 (13 February 2006).

Para Hoy

- **Introducción a los sistemas seguros**
- Autenticación de principals
- Autenticación de mensajes
- Confidencialidad de mensajes
- Protocolos de seguridad
- Autorización

Introducción al diseño de sistemas seguros

Seguridad en sistemas

- Un sistema seguro debe garantizar
 - Autenticidad
 - Integridad
 - Autorización

Complete mediation

For every requested action, check authenticity, integrity, and authorization.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Clasificación de Amenazas

- El primer paso en el diseño de seguridad es la identificación de amenazas.
- Estas se pueden clasificar en:
 - Acceso no autorizado a la información
 - Modificación no autorizada a la información
 - Negación de servicio no autorizada

Seguridad es difícil de garantizar

- Arquitecto: Debe bloquear todos los caminos a un estado malo
- Atacante: Debe encontrar un solo camino a una estado malo

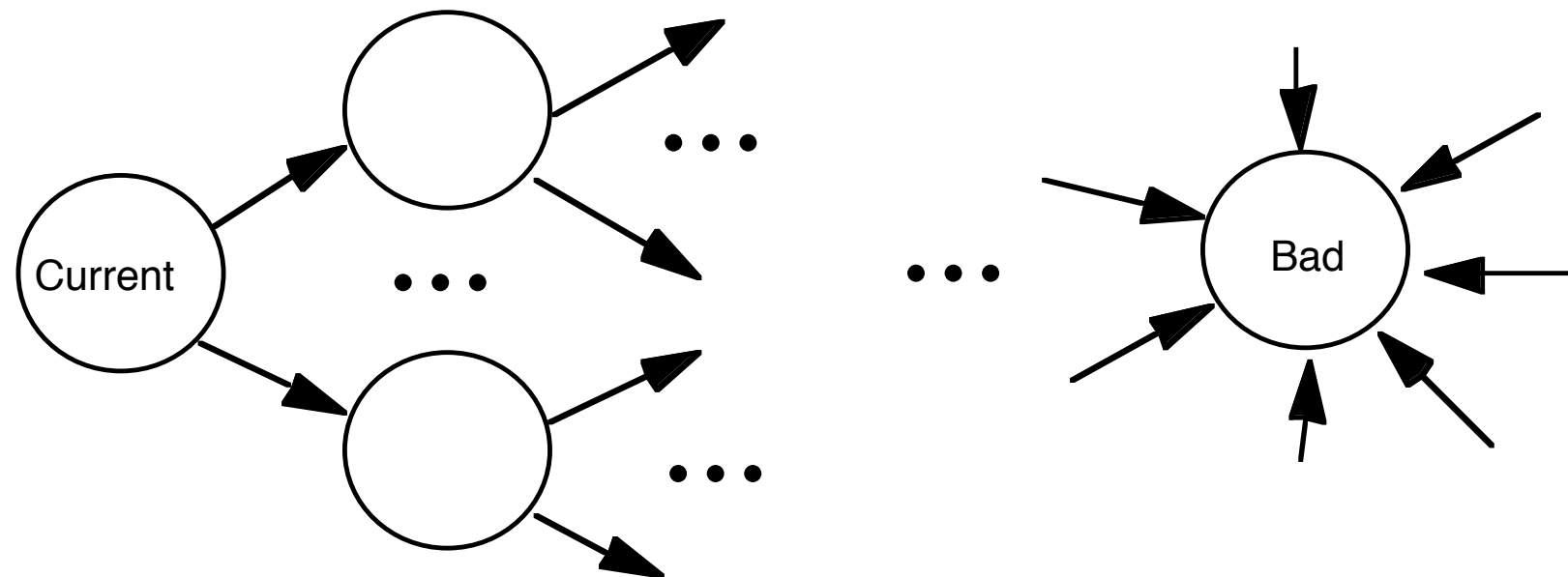


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Diseño paranoico

- Dos principios de diseño
 - Sea explícito. e.g., Todas los supuestos deben ser explícitos y la información de mensajes también.
 - Diseñe para la iteración. Asuma que realizará errores.

Requerimientos del diseño paranoico

- Certifique la seguridad del sistema. Que el diseño corresponda a la política, que la impl. al diseño, y el ejecutable a la impl.
- Mantenga trazas de auditoria
- Diseñe el sistema para obtener feedback

Principios de Diseño

Open design principle

Let anyone comment on the design. You need all the help you can get.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Minimize secrets

Because they probably won't remain secret for long.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Economy of mechanism

The less there is, the more likely you will get it right.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Minimize common mechanism

Shared mechanisms provide unwanted communication paths.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Fail-safe defaults

Most users won't change them, so make sure that defaults do something safe.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Least privilege principle

Don't store lunch in the safe with the jewels.

Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Dificultades por rápido cambio de tecnología

- Tecnología cambia muy rápido
- Es difícil mantener los niveles de seguridad
- Ejemplo:
 - En los 80s solo unos pocos computadores estaban en red
 - En 20 años todo el mundo está conectado
- Aún muchas soluciones Ad-hoc para lidiar con la seguridad de red

Modelo de seguridad

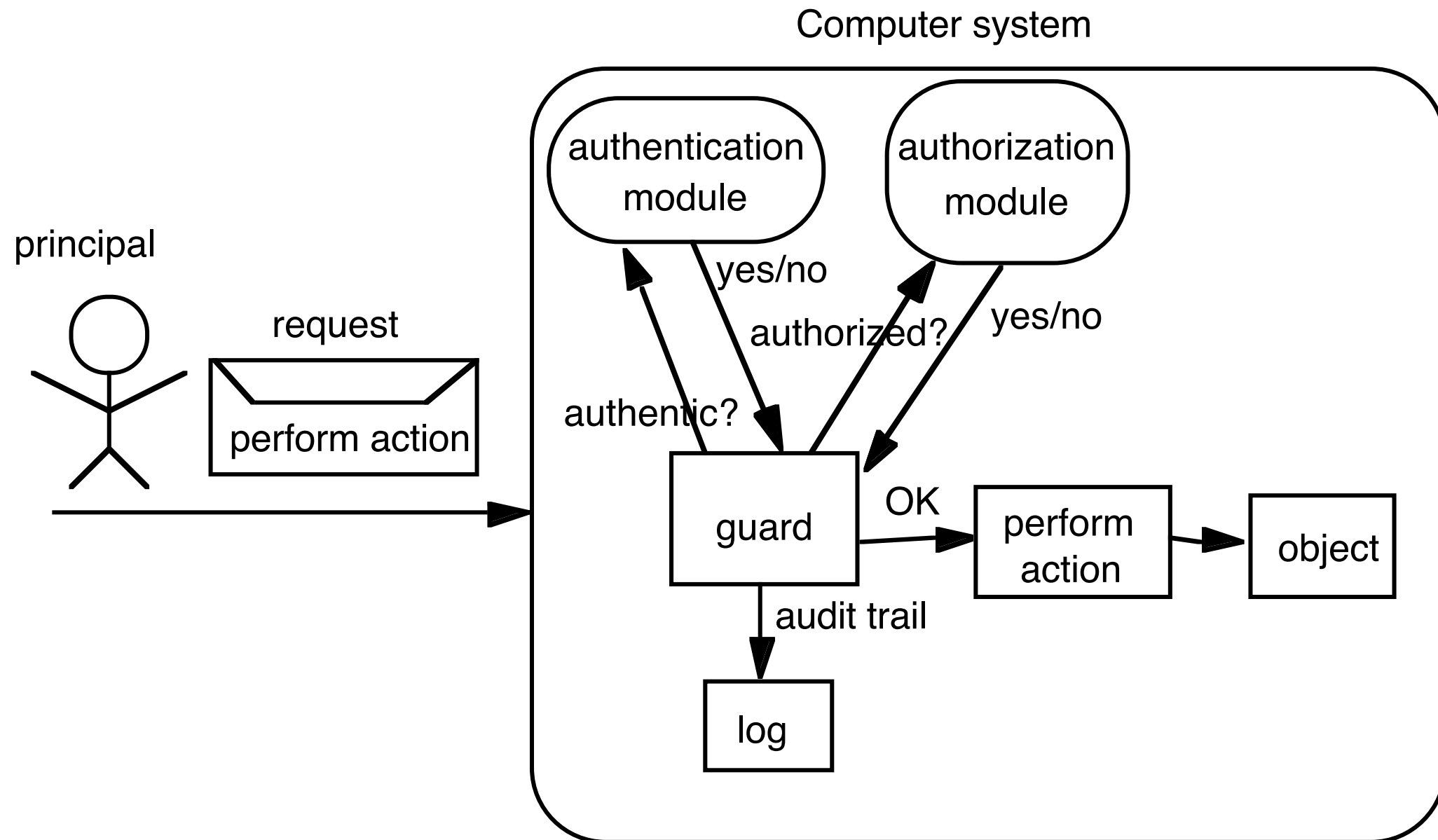


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Trusted Computing Base

- Como atacar la complejidad
 - Divida los módulos en confiables (Trusted Computing Base) y no confiables
 - Defina los requerimientos para el TCB
 - Diseñe un TCB mínimo
 - Implemente el TCB
 - Ejecute el TCB e intente romperlo

Agenda

- Introducción a los sistemas seguros
- **Autenticación de principals**
- Autenticación de mensajes
- Confidencialidad de mensajes
- Protocolos de seguridad
- Autorización

Autenticación de "Principals"

Autenticación

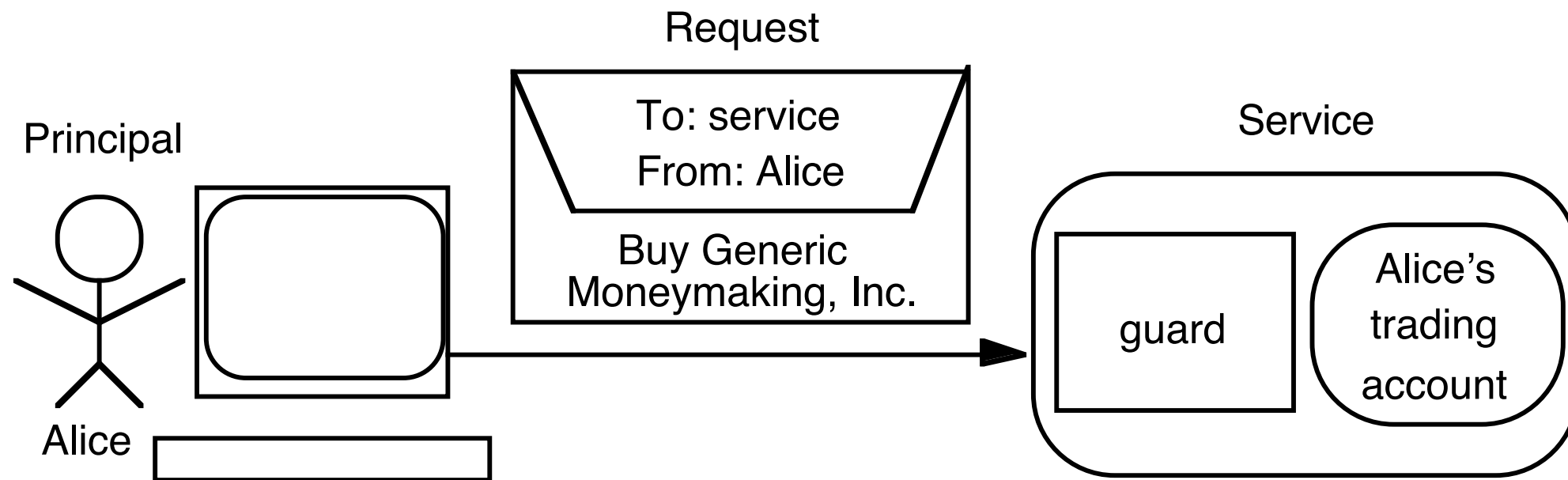


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

- Autenticación: Es Alice realmente quien dice ser?

Sistemas de autenticación

- Sistemas de autenticación
 - **Rendezvous:** crean un identificador de principal y un método para verificar la identidad (e.g., creación de cuenta)
 - **Verificación de identidad**
- El método de verificación puede ser:
 - **Una propiedad física del usuario**
 - **Algo que el usuario tiene**
 - **Algo que el usuario conoce (e.g., pwd)**

Funciones hash criptográfico

- **$hash(M) = V$** donde M es un arreglo de bytes y V es un valor de tamaño fijo.
- Es difícil calcular M conociendo V
- Es difícil encontrar M' tal que

$$hash(M) = hash(M')$$

- La longitud de V es corta pero suficientemente larga para garantizar una probabilidad baja de colisión.
- OJO con la ventana de validez
- SHA-0 (160 bits), SHA-1 (160 bits)

Usando funciones hash para proteger passwords

- El hash de la clave es almacenado
- La clave se envía una sola vez por sesión
- Se genera aleatoriamente una clave mas fuerte para la sesión
- Nunca viaja la clave por la red

Agenda

- Introducción a los sistemas seguros
- Autenticación de principals
- **Autenticación de mensajes**
- Confidencialidad de mensajes
- Protocolos de seguridad
- Autorización

Autenticación de Mensajes

Objetivo

- Integridad de datos
- Autenticidad del origen

Diseños cerrados y criptografía

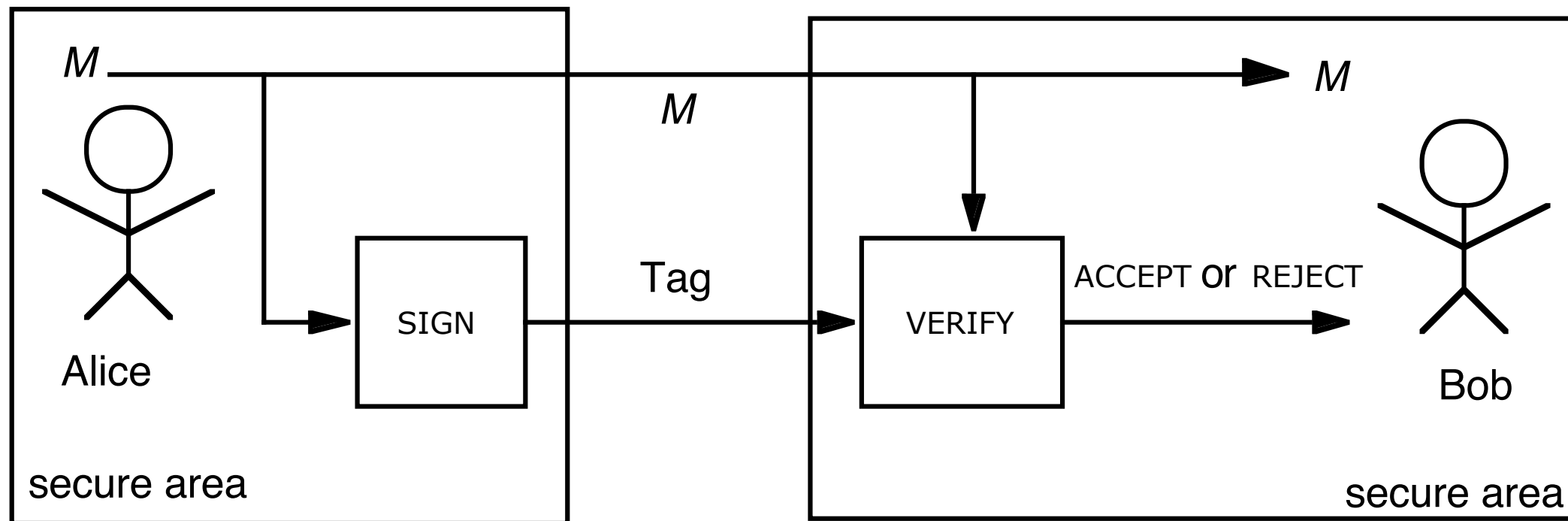


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

- Diseño cerrado, los algoritmos de sign and verify se mantienen secretos

Diseños abiertos y criptografía

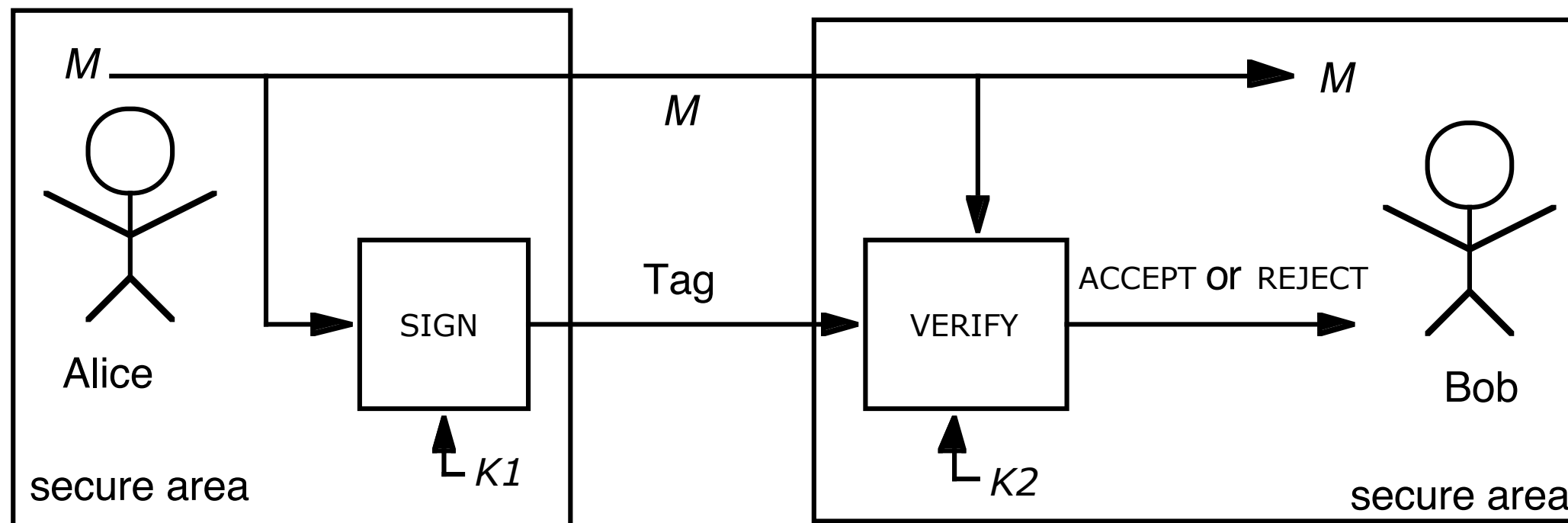


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

- Sign y verify son públicos
- Las llaves son secretas

Modelo de autenticación basado en llaves (key-based)

- Dos aproximaciones:
 - Criptografía de secreto compartido ($k_1 = k_2$)
 - Criptografía de llave pública (K_1 privada, K_2 pública)
- Como Funciona
 - $SIGN(M, K_1) = T$
 - $result = VERIFY(M', T', K_2)$, verdadero si $M' = M$ y $T' = T$
 - K_1 no se puede derivar de K_2

Autenticación de llave pública vs la de secreto compartido

- El Tag generado con algoritmo de llave pública es llamado firma digital
- El de llave pública tiene menos secretos
- Es más fácil perder el secreto en un algoritmo de llave compartida

Distribución de llaves

- Las llaves son públicas
- Pero el mecanismo de distribución debe ser confidencial y autenticado.
- El truco es tener otras llaves ya distribuidas físicamente (e.g., Browsers)
- Generalmente se usa un tercero confiable

Agenda

- Introducción a los sistemas seguros
- Autenticación de principals
- Autenticación de mensajes
- **Confidencialidad de mensajes**
- Protocolos de seguridad
- Autorización

Confidencialidad de mensajes

Confidencialidad de Mensajes Usando Cifrado

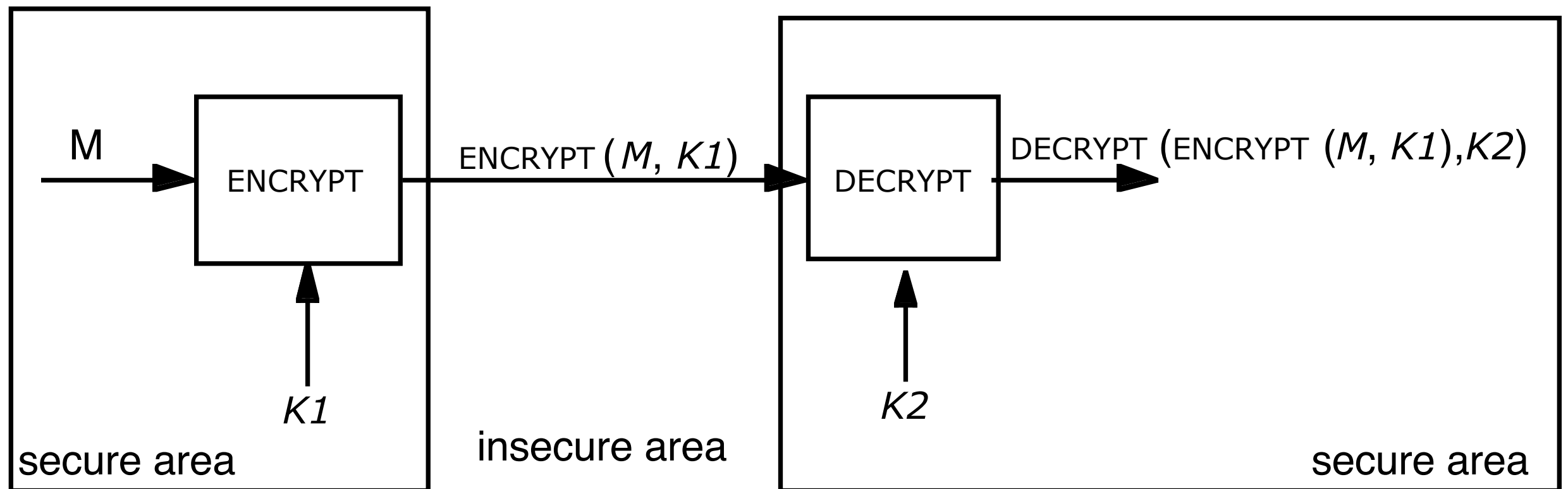


Imagen del libro: *Principles of Computer System Design* por Saltzer y Kaashoek. 2009. Morgan Kaufmann.

Propiedades de los procesos ENCRYPT y DECRYPT

- $C = \text{ENCRYPT}(M, K_1)$
- $M' = \text{DECRYPT}(C, K_2)$
- Debe ser difícil computar K_s , M' o M conociendo ejemplos de partes individuales.
- Algoritmos pueden ser públicos

Asegurando Confidencialidad y Autenticación

- Si quiero confidencialidad, solo cifro el mensajes
- Si quiero autorización, solo firmo el mensaje
- Si quiero autorización y confidencialidad, primero firmo y luego cifro el mensaje
- $\text{SIGN}(\text{ENCRYPT}(M, K_{\text{encrypt}}), K_{\text{sign}}))$
- OJO K_{encrypt} , K_{sign} son diferentes

Se puede usar la cifrado para la autenticación?

- No
- Se considera inseguro
- Son conceptos ortogonales y separados
- No confié en su intuición cuando diseñé sistemas de seguridad.

Agenda

- Introducción a los sistemas seguros
- Autenticación de principals
- Autenticación de mensajes
- Confidencialidad de mensajes
- **Protocolos de seguridad**
- Autorización

Protocolos de seguridad

Ejemplo: Distribución de llaves (Public-key)

- Asumimos que Bob y Alice confían en Charles

1. Alice \Rightarrow Charles: {"Please give me keys for Bob"}

2. Charles \Rightarrow Alice: {"To communicate with Bob, use public key K_{Bpub} ."}_{C_{priv}}

- Charles tiene rol de Certification authority (CA)
- Mensaje 2 se llama certificado

Ejemplo: Distribución de llaves (Shared-key)

- Asumimos que Bob y Alice confían en Charles y tienen llaves ya compartidas
 1. Alice \Rightarrow Charles: {"Please give me keys for Bob"}
 2. Charles \Rightarrow Alice: $\{\{\text{"Use temporary authentication key } A_{kAB}$
and temporary encryption key E_{kAB} to talk to Bob."}\}_{A_{kAC}}^{E_{kAC}}
 3. Charles \Rightarrow Bob: $\{\{\text{"Use the temporary keys } A_{kAB}$ and E_{kAB} to talk to Alice."}\}_{A_{kBC}}^{E_{kBC}}
- Charles tiene rol de Key Distribution Center (KDC)

Un protocolo de intercambio de llaves usando llaves públicas

- Intercambio entre un Cliente y un servidor, ambos anónimos.
- (public-key) Usando certificados y la llave pública crean un par de llaves compartidas de cifrado y de firma
- Las llaves completas no viajan por la red, se generan con algoritmos de generación (e.g., Diffie-Hellman)
- (shared-key) Una vez se garantiza la autenticación y la confidencialidad se usan las claves compartidas para cifrar y firmar.

Agenda

- Introducción a los sistemas seguros
- Autenticación de principals
- Autenticación de mensajes
- Confidencialidad de mensajes
- Protocolos de seguridad
- **Autorización**

Autorización

Operaciones de Autorización

- **Autorización:** Dar permiso a un principal para operar en un objeto.
- **Mediación:** Revisa si un principal tiene permiso de hacer una operación en un objeto
- **Revocación:** remueve un permiso dado anteriormente a un principal

El modelo de guarda simple

- Tíquets: Cada Objeto tiene una tíquet, los usuarios autorizados tienen copia del tíquet
- Listas: Cada objeto tiene una lista de los principales que tienen acceso.
- Agencia: Se puede cambiar entre modelo de tíquets y listas
- A menudo se crean grupos de protección para no crear listas de cada usuario

Ejemplo: El control de acceso en UNIX

- Principal en Unix:
 - Usuarios y Grupos, les asigna un entero de longitud fija (UID o GID)
 - root es UID 0
 - Los servicios corren en nombre de un principal (e.g., www para Apache)

Ejemplo: El control de acceso en UNIX I I

- Access Control List:
 - Todo en Unix son Archivos (Todo)
 - Unix kernel es el guarda
 - Cada archivo tiene una lista de 3 entradas:
 - Owner
 - Owner Group
 - Other
 - Por cada entrada UNIX permite permisos para
 - WRITE, READ, EXECUTE
 - Ejemplo: Archivo “mybin”, UID 18, GID 20, “rwxr-xr--”

Ejemplo: El control de acceso en UNIX I I I

- Autenticando usuarios:
 - Modelo: Ticket; guarda: programa de login
 - Principal: usuario; Token: password
 - Objeto: El sistema Unix
- Control de acceso
 - Modelo: Lista
 - Principal: Usuario, grupo; Token: UID, GID
 - Objeto: Archivos en el sistema