

La criptografía en la autenticación de procesos Linux



ITESM-CEM

Dpto. Ciencias Computacionales

Erika Saucedo (esaucedo@campus.cem.itesm.mx)

Roberto Gómez (rogomez@campus.cem.itesm.mx)

<http://webdia.cem.itesm.mx/dia/ac/rogomez>



Agenda

- ◆ Antecedentes
- ◆ Propuesta
- ◆ Criptografía y autenticación
- ◆ Descripción del modelo
- ◆ Conclusiones



Antecedentes

- ◆ Seguridad Informática elemento importante en cualquier organización.
- ◆ Ataques cada vez más sofisticados
- ◆ Riesgo: Información
- ◆ Responsabilidad: Administrador
- ◆ Tarea tediosa y poco confiable.
- ◆ ¿Automatización de sistemas?
 - Degradación del sistema
 - Eliminar la intervención del administrador en lo posible



Seguridad Informática

- ◆ La **seguridad lógica** de un sistema se puede considerar desde tres perspectivas; la confidencialidad, la integridad y la disponibilidad.
- ◆ El conjunto de políticas y mecanismos para mantener la integridad, disponibilidad y confidencialidad de la información de un sistema de cómputo.



Seguridad Informática

- ◆ Un sistema posee la propiedad de:
 - **Confidencialidad** sí, la información manipulada por éste no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados.
 - **Integridad** sí, los datos manipulados por éste no son alterados o destruidos por usuarios, entidades o procesos no autorizados.
 - **Disponibilidad** sí, la información accesible está disponible en el momento que así lo requieran los usuarios, entidades o procesos autorizados.



Propuesta

- ◆ Responsable de llevar a cabo las acciones ante un ataque contra la integridad de los procesos debe ser el Sistema Operativo.
- ◆ Degradación del sistema mínima.



Objetivos de la criptología

- ◆ Mantener la confidencialidad del mensaje
 - la información contenida en el mensaje permanezca secreta
- ◆ Garantizar la autenticidad tanto del mensaje como del par remitente/destinatario
 - el mensaje recibido ha de ser realmente el enviado
 - el remitente y destinatario han de ser realmente quienes dicen ser y no remitentes y/o destinatarios fraudulentos



Criptografía y autenticación

- ◆ Integridad de los datos
 - huellas digitales
- ◆ Autenticación de mensajes
 - firma digital
- ◆ No repudio
 - firma digital



Las funciones de un solo sentido

- ◆ One-way hash function
 - función toma una variable de tamaño variable (cientos o miles de bits) y una salida de tamaño fijo (p.e. 160 bits)
- ◆ Función asegura que, si la información es cambiada (aún en sólo un bit) un valor completamente diferente es producido



Ejemplo función hash: md5

toto@cognac:69>more mensaje

ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 2001
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte;

RGC

toto@cognac:70>hash mensaje

0c60ce6e67d01607e8232bec1336cbf3

toto@cognac:71>



Continuación del ejemplo

toto@cognac:71>more mensaje
ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 2001
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte

RGC

toto@cognac:72>hash mensaje
30a6851f7b8088f45814b9e5b47774da
toto@cognac:73>



Algoritmos

- ◆ Algoritmo MD2
- ◆ Algoritmo MD4
- ◆ Algoritmo MD5
- ◆ SHA-1
- ◆ RIPE MD-160
- ◆ HMAC
- ◆ N-Hash
- ◆ Havalk



La huella digital

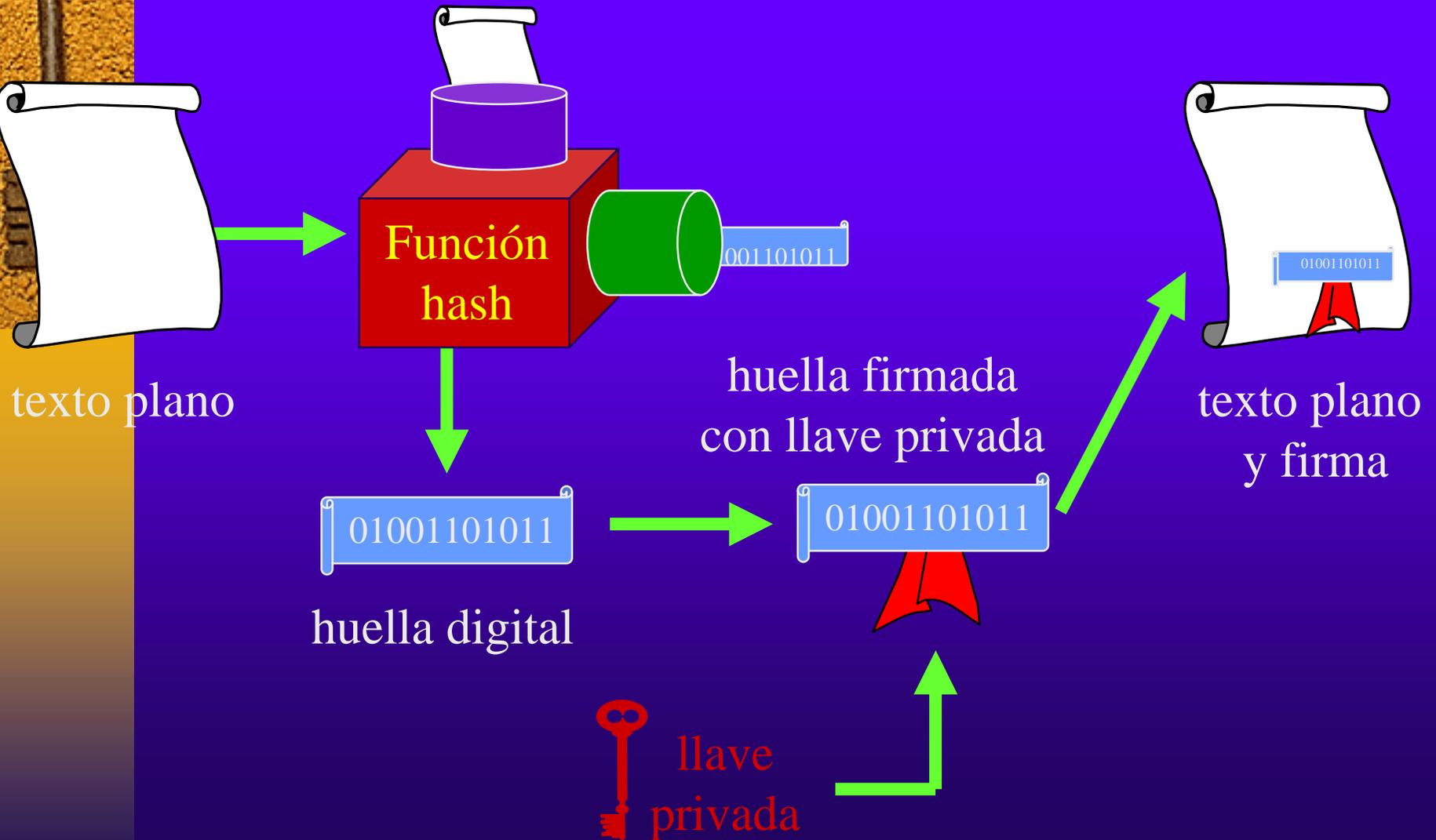
- ◆ La salida producida por una función hash aplicada a un documento, es conocida con el nombre de huella digital de dicho documento
- ◆ Cualquier cambio en el documento produce una huella diferente
- ◆ Huella digital también es conocida como compendio de mensaje (cuando el documento es un mensaje)



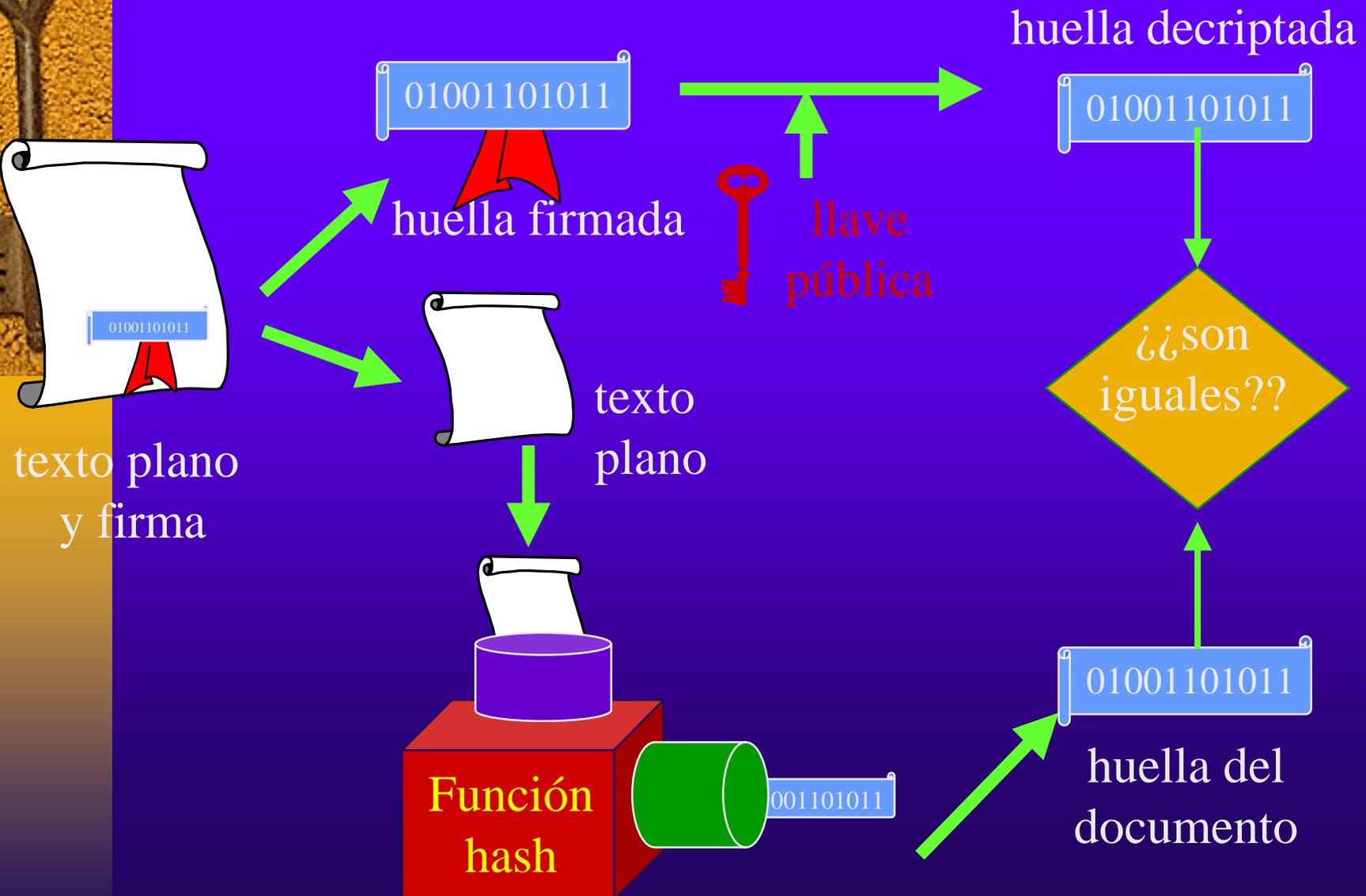
Firmas y huellas digitales

- ◆ Es posible usar la huella y la llave privada para producir una firma
- ◆ Se transmite el documento y la firma juntos
- ◆ Cuando el mensaje es recibido, el receptor utiliza la función hash para recalcular la huella y verificar la firma
- ◆ Es posible encriptar el documento si así se desea

Firma digital segura (envío)



Firma digital segura (recepción)





Seguridad de la firma

- ◆ Seguridad depende de lo seguro de la función hash
- ◆ No existe ninguna forma de tomar la firma de alguien de un documento y ponerla en otro
- ◆ No es posible alterar un mensaje firmado
- ◆ El más simple cambio en el documento firmado se verá en la verificación



Códigos Autenticación Mensaje

- ◆ MAC por sus siglas en inglés
- ◆ También conocido bajo el nombre de DAC (Data Authentication Code)
- ◆ Función de un solo sentido junto con una llave secreta:

$$\text{MAC} = C_K(M)$$

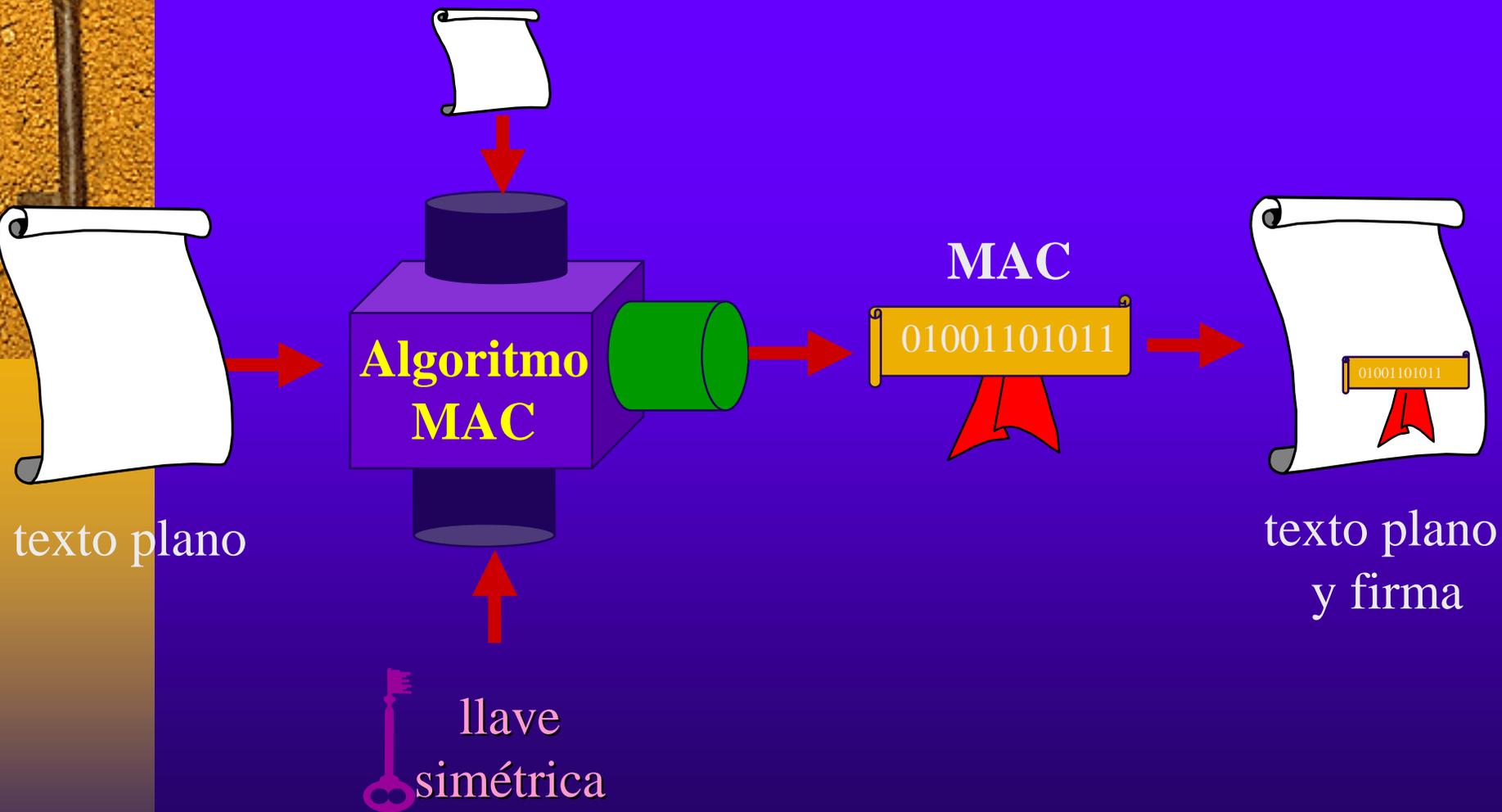
- ◆ M es el mensaje, K es una llave que conoce tanto el emisor como el receptor, y C_K es una función hash basada en K.



Códigos Autenticación Mensaje

- ◆ El principio es el mismo, pero si alguien conoce la llave puede verificar el valor hash.
- ◆ El sitio emisor pega el MAC al mensaje cuando se decide que el mensaje es correcto.
- ◆ El receptor autentica re-calculando el MAC

Firma digital en base a MAC (emisión)



Firma digital en base a MAC (recepción)

texto plano
y firma

texto
plano

Algoritmo
MAC

llave
simétrica

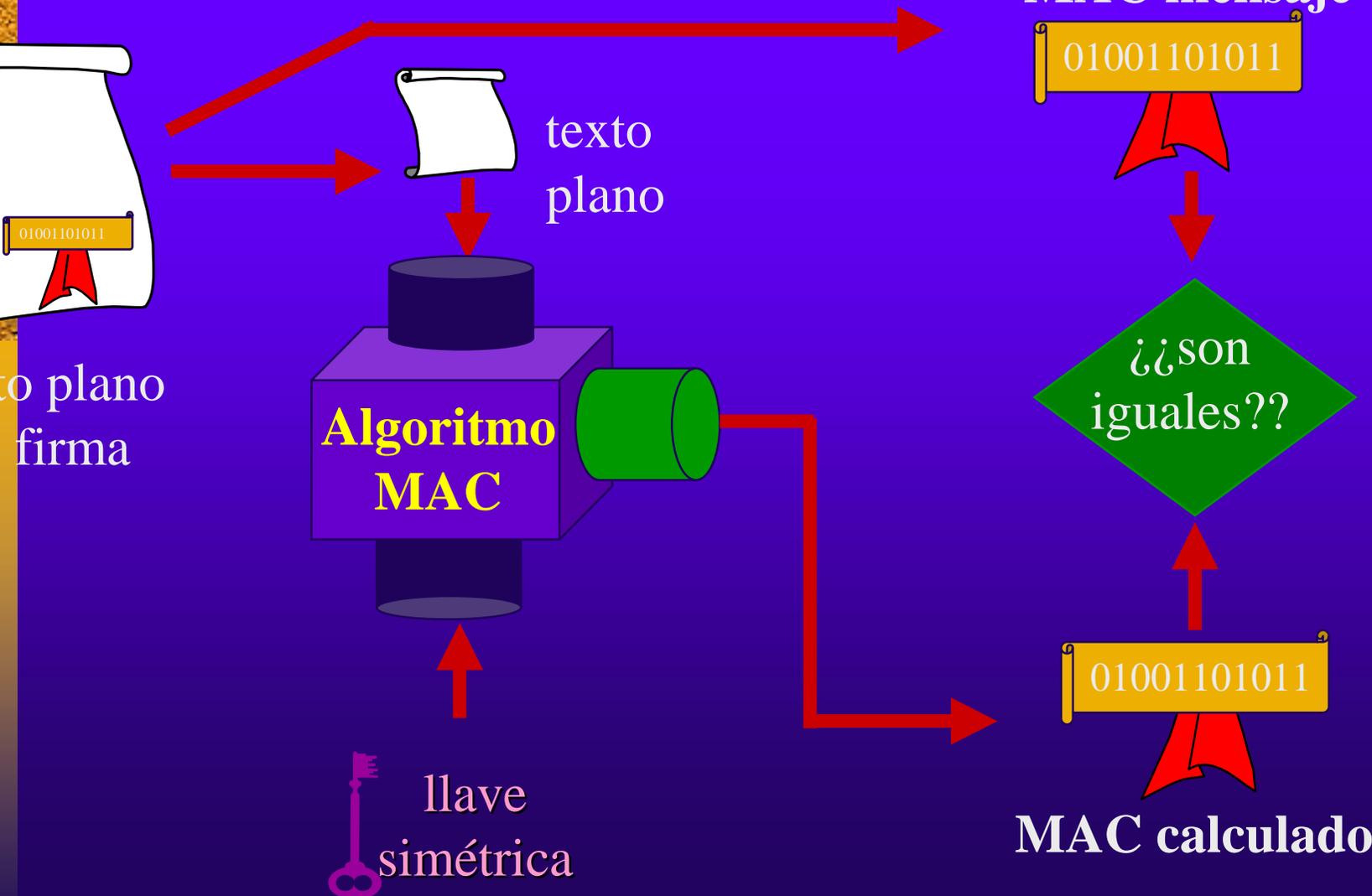
MAC mensaje

01001101011

¿¿son
iguales??

01001101011

MAC calculado





Definición de un proceso

- ◆ Difícil proporcionar una definición proceso.
- ◆ De acuerdo a Deitel
 - programa en ejecución
 - actividad asíncrona
 - “espíritu animado” de un procedimiento
 - la entidad a la que se asignan los procesadores
- ◆ De acuerdo a Tanenbaum
 - un programa en ejecución
 - un programa es un conjunto de instrucciones y datos almacenados en una imagen ejecutable y es una entidad pasiva

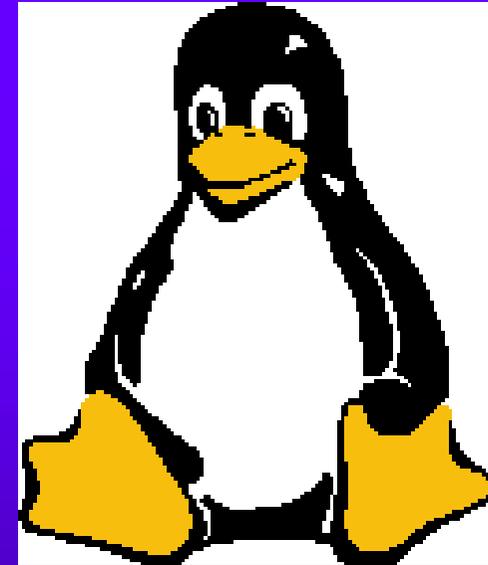
Contexto de un proceso





Linux

- ◆ Sistema Operativo desarrollado por Linux Torvalds
- ◆ Licencia GNU
- ◆ Código fuente disponible
- ◆ Compatible con el estándar POSIX 1003.1
- ◆ Versión del núcleo 2.2.13





Linux vs Unix

- ◆ Linux posee todas las demandas de un sistema tipo Unix
- ◆ Multitasking
- ◆ Acceso multiusuario
- ◆ Multiprocesamiento
- ◆ Independencia de la arquitectura
- ◆ Carga de ejecutables bajo demanda
- ◆ Paginación para el manejo de memoria



Procesos en Linux

- ◆ Desde el punto de vista del núcleo
 - proceso es una entrada en la tabla de procesos
 - es una entrada en `/usr/src/linux/include/linux/sched.h`
- ◆ Cada elemento individual que la conforma es una estructura que se llama `task_struct` esta la encuentran, también definida en `include/linux/sched.h`

Algunos campos de task_struct

```
struct task_struct {
    volatile long state;
    unsigned long flags;
    int sigpending;
    mm_segment_t addr_limit;
    struct exec_domain *exec_domain;
    long need_resched;
    long counter;
    long priority;
    cycles_t avg_slice;
    int lock_depth;
    struct task_struct *next_task, *prev_task;
    struct task_struct *next_run, *prev_run;
    int did_exec:1;
    pid_t pid;
    pid_t pgrp;
    pid_t tty_old_pgrp;
    pid_t session;
    :
    :
}
```





Descripción del modelo

- ◆ Integración de un campos de verificación de integridad a nivel kernel
- ◆ Verificación de integridad
- ◆ Implantación:
 - **Etapa 1.** Añadir un campo de verificación de integridad al código fuente del kernel.
 - **Etapa 2.** Verificar la integridad.
 - **Etapa 3.** definir un protocolo de comunicación de los procesos verificadores.



Campo verificador de integridad

- ◆ Modificación de la estructura *task_struct*
- ◆ Adición de un campo de verificación de integridad donde se guarda la huella digital que definirá a ese proceso.
- ◆ El campo de verificación almacenará el resultado de aplicar MD5 (128 bits) a los campos seleccionados de la estructura *task_struct*
char auth_pro[16]
- ◆ Inicialización del campo.
- ◆ Recompilar el nuevo kernel.



Verificación de integridad

- ◆ Toda la estructura
- ◆ Código ejecutable
- ◆ Código ejecutable y campos de la estructura *task_struct* que permanecen sin cambio



Verificación de toda la estructura

- ◆ Estructura contiene datos que están en constante cambio
- ◆ Necesario actualizar para evitar falsos positivos.
 - trabajo exhaustivo por parte del procesador
 - desempeño sistema afectado



Verificación código ejecutable

- ◆ Mejor opción con respecto a la primera
 - con relación a la eficiencia del sistema
- ◆ Protege ataques modificación códigos
 - virus
- ◆ No protege aquellos ataques que modifican algunos bits de una o más banderas en el estado de un proceso para ganar acceso como root
 - stack/buffer overflow



Verificación código y campos task_struct

- ◆ Verificar la integridad del código ejecutable del proceso
- ◆ Seleccionar algunos campos en la estructura task_struct
 - campos que permanecen sin cambio durante el tiempo de vida del proceso

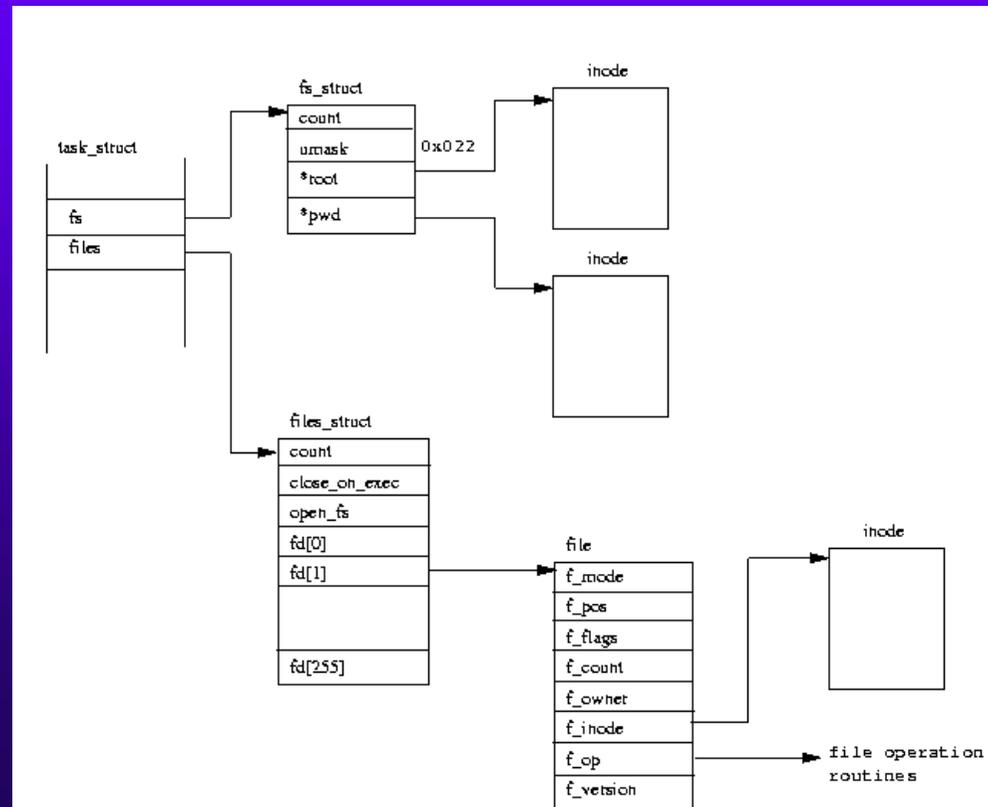


Datos a verificar en `task_struct`

- ◆ Campos que no se modifican durante la vida del proceso
- ◆ Existen campos que en condiciones normales no son modificados
 - pueden modificarse por medio de llamadas al sistema
 - pueda aplicarse a objetivos específicos en caso de que no se quiera que se hagan ciertas llamadas al sistema

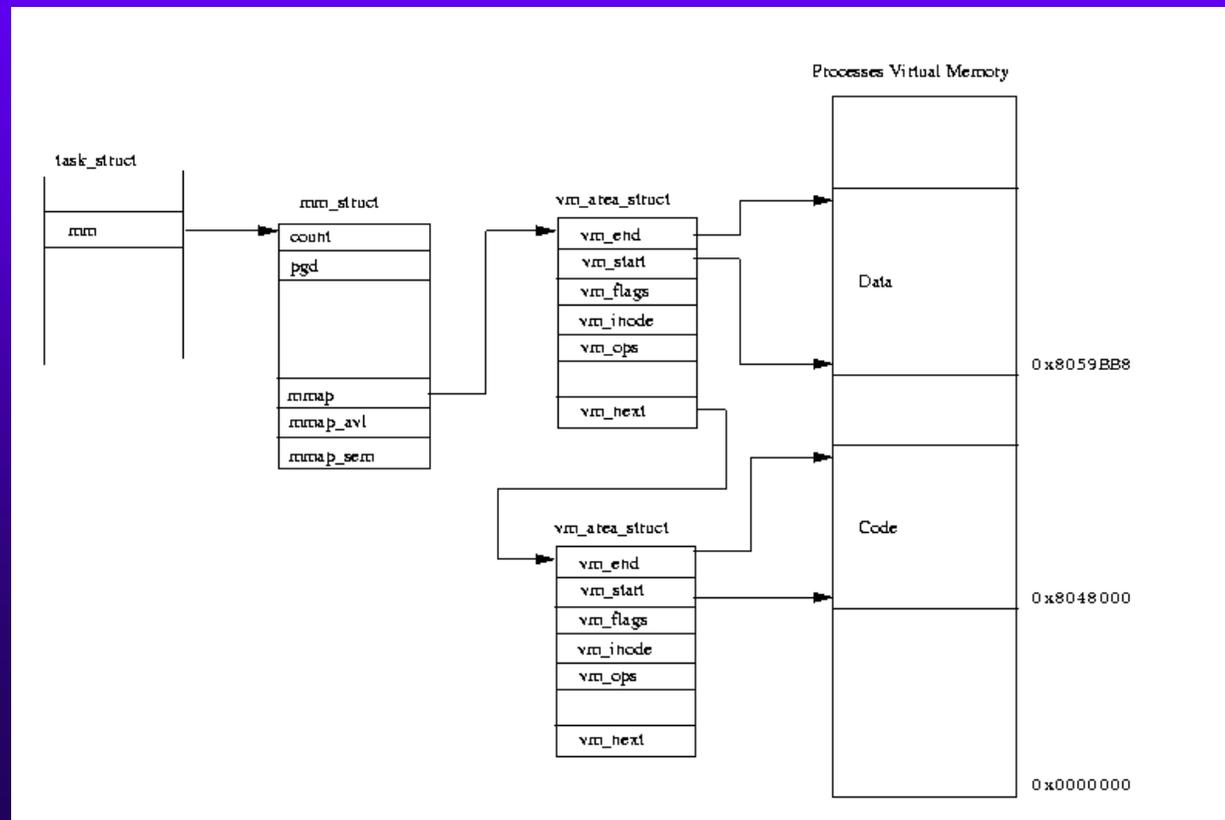
Los campos seleccionados

- ◆ Campos denominados identificadores efectivos
 - `uid`, `egid`
- ◆ El campo de la política de planificación (`policy`)
- ◆ El campo del modo en que se abre un archivo por un proceso (`f_mode`)



Los campos seleccionados

- ◆ Apuntador a la estructura *mm* que define el área de memoria virtual de un proceso, esta área contiene el código ejecutable y el área de datos del proceso





Verificando las huellas

- ◆ Se considera un proceso que esta viajando dentro de la red denominado **proceso protector**
- ◆ Cada vez que este **proceso protector** se encuentra en una máquina se lleva a cabo la verificación de integridad en dos posibles formas.



Opciones verificación

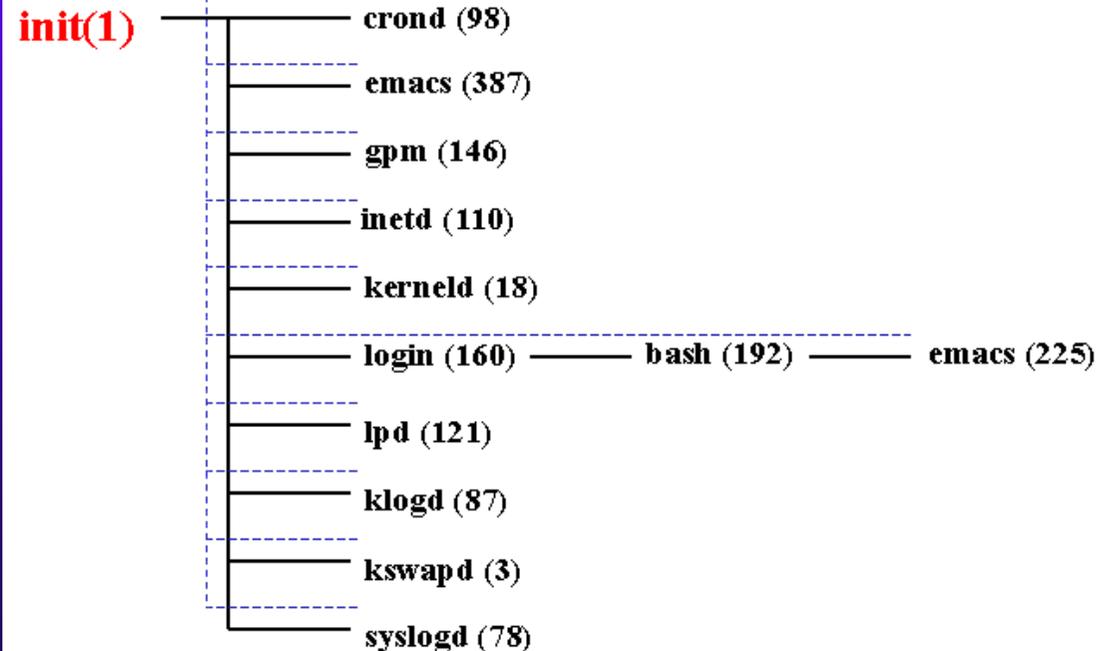
- ◆ La primera opción es que el **proceso protector** genere las huellas correspondientes a todos los procesos que se están ejecutando en una máquina en un tiempo dado.
- ◆ La segunda opción es que el **proceso protector** solamente genere las huellas que corresponde a los procesos que están protegiendo a los procesos normales del sistema; es decir, la protección de las aplicaciones de seguridad.



La primera opción

¿el desempeño?

- ◆ Todos los procesos ejecutándose en el sistema iniciando por el proceso *init*
- ◆ Se lleva a cabo un recorrido de árbol con el proceso *init* como raíz.
- ◆ Se realiza el escaneo de todos los procesos que se están ejecutando en esa máquina en un momento determinado.





La segunda opción

- ◆ La segunda opción implica la generación de huellas para los procesos que están protegiendo los procesos normales del sistema
- ◆ Dejando el trabajo de resguardar la seguridad de los demás procesos a las aplicaciones instaladas para ello.



¿Contra que se compara?

- ◆ Dos opciones:

1. utilizar un archivo que contenga la colección de huellas que a su vez haya sido procesado por medio de una función hash
2. el uso de una base de datos.



Archivo huellas

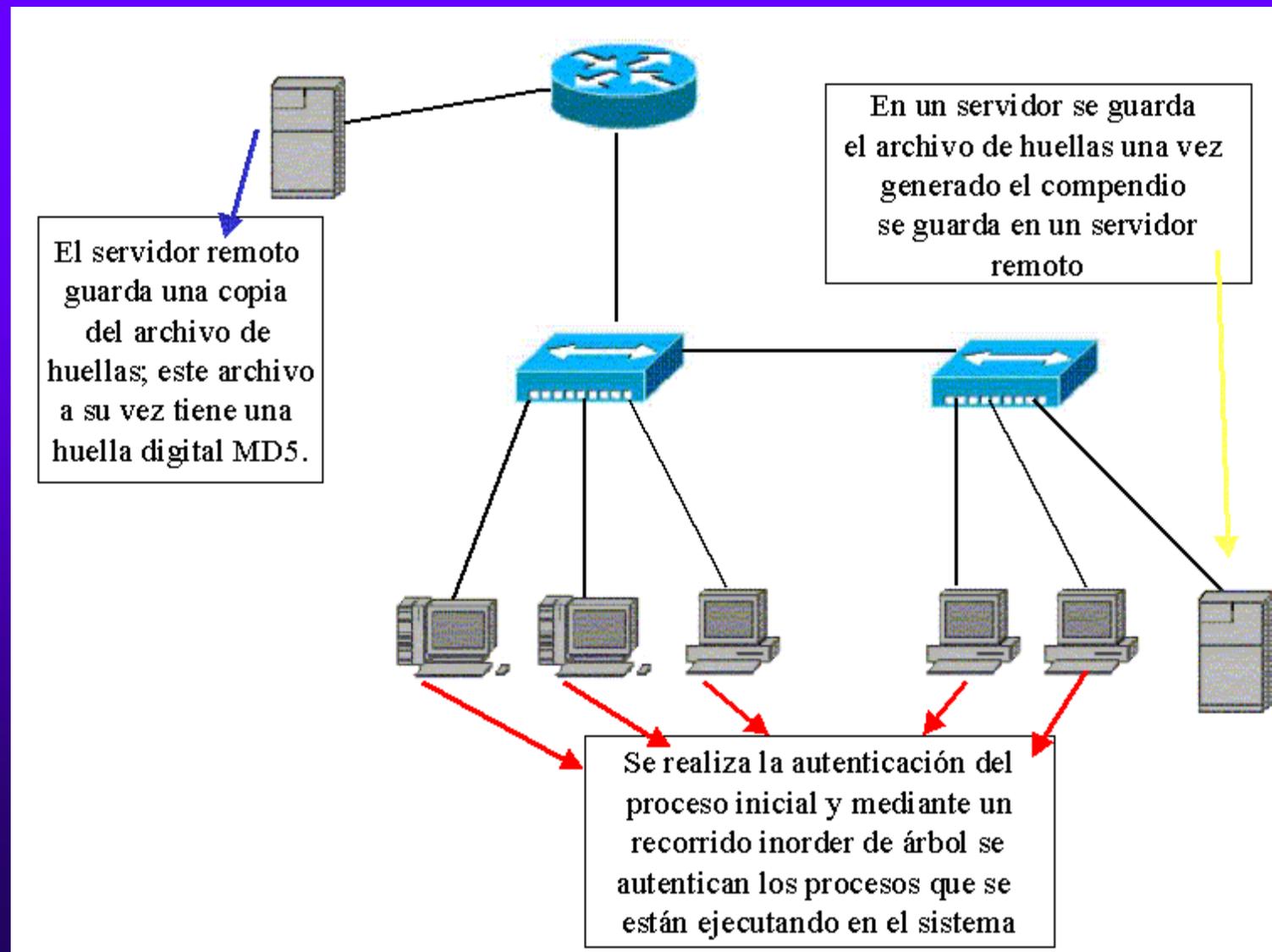
- ◆ Se genera un archivo que contiene todas las huellas
 - a su vez esta firmado digitalmente para así evitar que sea modificados sin autorización
 - se utiliza para esto MD5 una vez mas para garantizar la integridad del archivo de huellas.
- ◆ Este archivo viaja junto con el proceso verificador.



Usando una base de datos

- ◆ Uso de una base de datos como mysql
- ◆ Se comporta de la misma forma que el caso anterior
 - solo que ahora se tiene una base de datos que es la encargada de manipular los datos de huellas.

Esquema base datos





Características de la implantación

- ◆ **Administración de huellas**
 - **Verificación de huellas**
 - **Utilización de un archivo que a su vez será cifrado y almacenado en dos servidores el servidor de archivos y un servidor de respaldo fuera de la subred analizada.**
 - **Utilización de una BD mysql para llevar a cabo esta verificación**



El protocolo de comunicación

1. Se inicia el programa protector en la computadora i elegida al azar.
2. Se lleva a cabo la verificación de integridad de los procesos que se están ejecutando en i.
 - 2.1 Si la integridad de alguno de los procesos ha sido violada se envía una señal de alarma al administrador.
 - 2.2 Si la integridad de los procesos no ha sido violada se continúa con el paso 3.



El protocolo de comunicación

3. Se envía un mensaje a todas las máquinas en la red Linux, se genera una lista de las máquinas que respondieron al mensaje y esta lista se guarda en un archivo.
4. Se elige una nueva máquina sobre la red para llevar a cabo el proceso de verificación de integridad y se envía una señal a dicha máquina para que inicie el proceso protector en esa máquina.



Protocolo de comunicación

5. Se repite el proceso desde el paso 2 de manera infinita hasta que se quiera terminar con la protección de procesos.



Conclusiones

- ◆ La verificación de integridad de procesos móviles puede representar un paso importante hacia la automatización de protección de sistemas.
- ◆ Es importante señalar que la presente propuesta puede ser extendida y aplicada en otros puntos de la Seguridad Informática y las redes.

GRACIAS POR SU
ATENCIÓN!!!!



*La Criptografía en la autenticación
de Procesos Linux*

ITESM-CEM

Dpto. Ciencias Computacionales

Erika Saucedo (esaucedo@campus.cem.itesm.mx)

Roberto Gómez (rogomez@campus.cem.itesm.mx)

<http://webdia.cem.itesm.mx/dia/ac/rogomez>