Presentación

Seguridad informática

Alis Yamil, De Rocco Federico y Muchinik Francisco

UBA

29 de noviembre de 2017

Contenido

Introducción

- Log: Es un registro que contiene los eventos ocurridos en un sistema.
  - Esta información puede ser utilizada para detectar la existencia de ataques.
  - Un atacante podría evitarlo modificando estos registros.
  - Auditoría de logs: Método que permite averiguar si los logs fueron alterados.

Protección de logs

- Trusted Computing Base (TCB): Es el componente responsable de realizar el logging. Si se conserva la integridad del mismo, los registros deberían ser seguros. Problema: no siempre están libres de bugs que permitan al atacante obtener permisos.
- Remote logging: Consiste en enviar los registros a otros equipos que los resguarden. Con esta medida el atacante deberá conocer y vulnerar todos estos equipos para poder ocultar el ataque.
- Imprimirlos: Una forma clásica de proteger los logs era imprimirlos de forma continua y ordenada. Problemas:
  - El sistema que se use para la impresión debe darle prioridad a los logs para que la impresora no se vea sobrecargada.
  - Un análisis de actividades sospechosas es mucho más difícil.
  - Se puede comprometer la impresora o las impresiones.
- Write Once Read Multiple (WORM): Son discos ópticos de poco ancho de banda de escritura. Dichos discos se extraen una vez que están llenos. Problema: se puede comprometer la integridad si se sustituyen uno o más discos.

MAC

- Se calcula un MAC para cada log usando un secreto. Esto permite impedir que, en desconocimiento de ese secreto, el atacante pueda modificar el log. Si lo hace, tendría que recalcular el MAC.
  Problemas:
  - En ausencia de un continuo uso de remote logging, este mecanismo no asegura la integridad de los registros viejos. Esto se debe a que el secreto es usado en el sistema que realiza el logging. Si este se compromete, también se obtiene el secreto.
  - El remote logging tampoco es perfecto ya que la seguridad de estos hosts pasa a ser crítica.

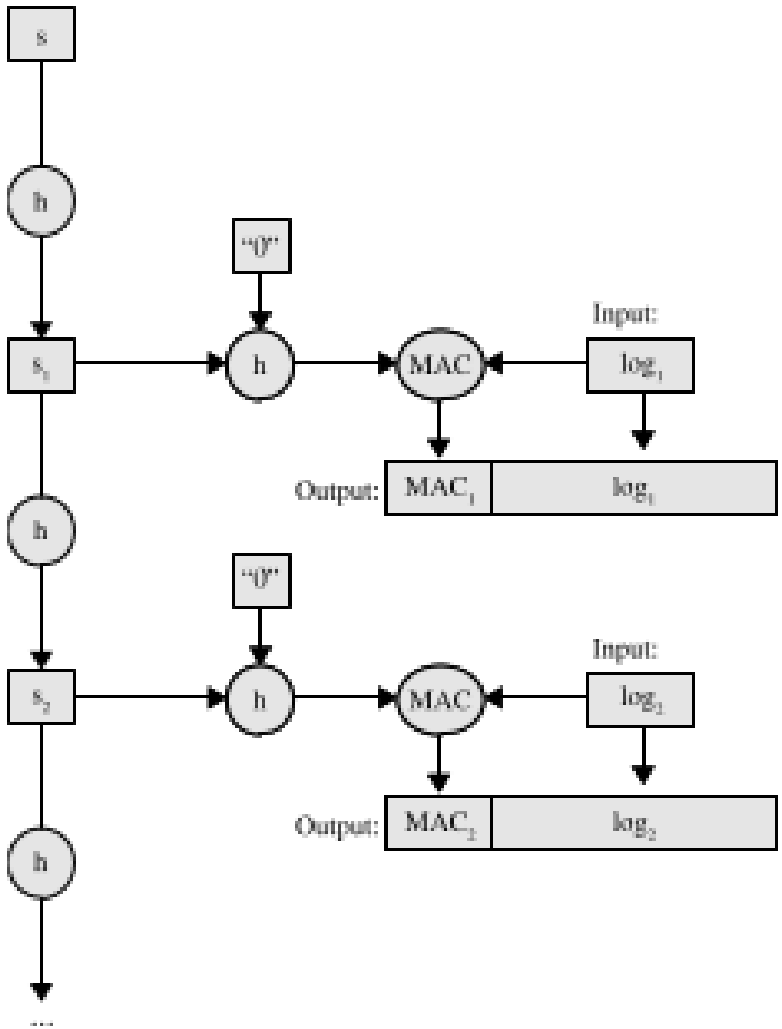
Forward Integrity

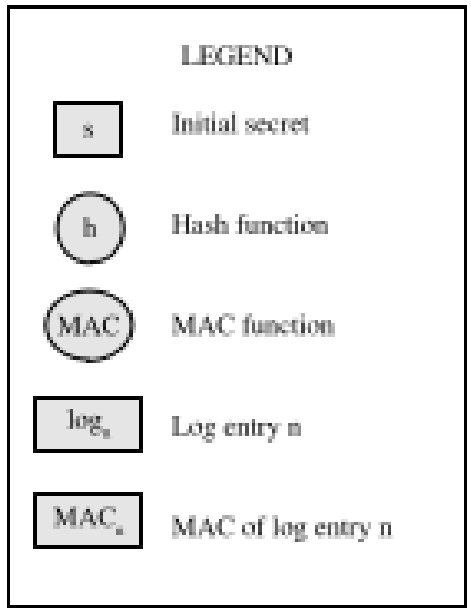
- Básicamente queremos generar un MAC para cada log que no pueda ser modificado sin quedar en evidencia aunque el sistema que genera los logs sea comprometido.
- La idea es que el sistema no repita la clave utilizada en los pasos anteriores. Además, una vez calculado el MAC se debe descartar dicha clave para evitar que se la pueda obtener.
- Una forma de asegurar esta propiedad es generar la clave actual en base a la inmediatamente anterior. Tenemos para el log número i la clave K_i que se obtiene aplicándole una función no reversible a K_{i-1}. Una vez calculada K_i, se borra K_{i-1}. Esto último requiere que el FI sea rápido.
- Si el atacante obtiene control del sistema en el momento que el siguiente log es el número i, obtendría la clave K_i. Con la cual no puede regenerar K_{i-1} ni ninguna de las anteriores.

Forward Integrity

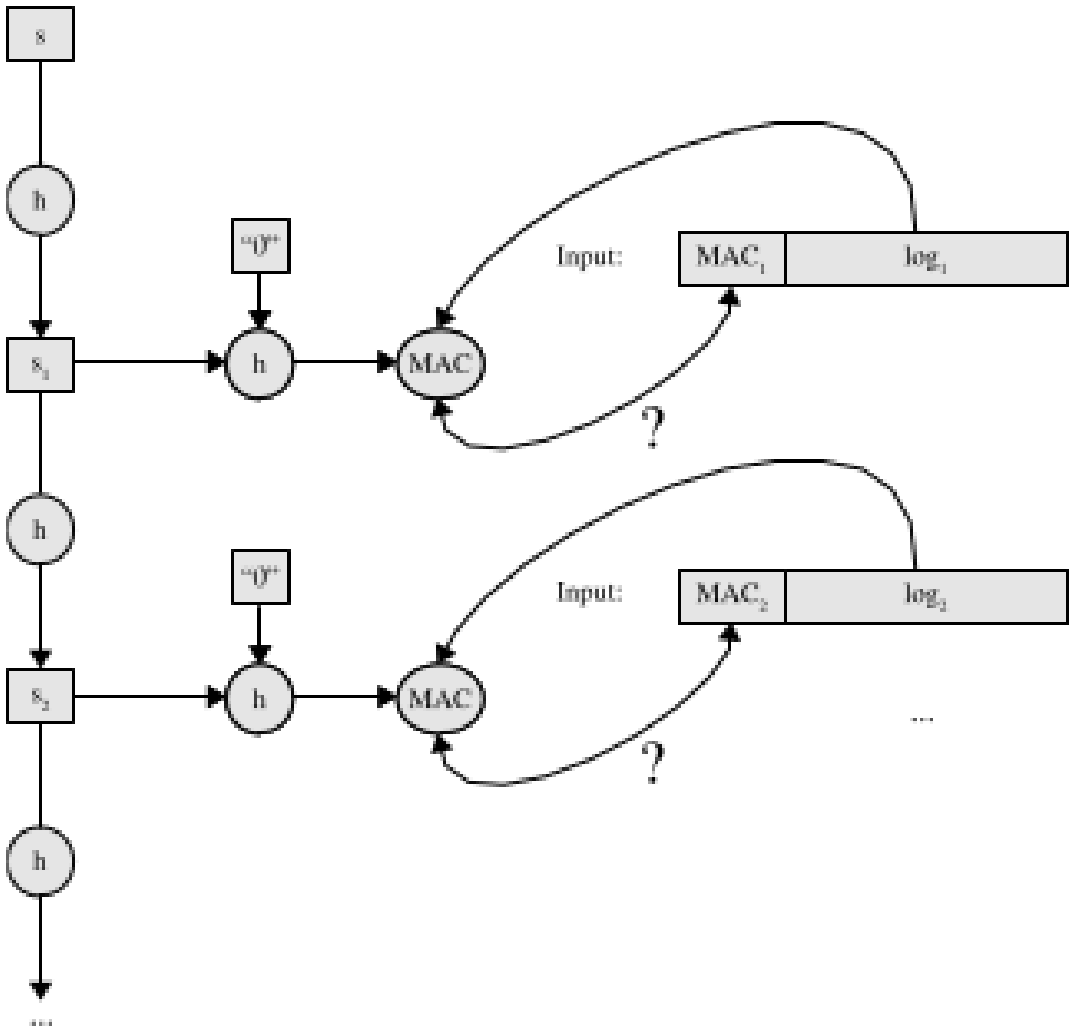
- Este sistema deja implícito que debe existir un secreto inicial y si este se compromete se pierde la seguridad. Por esto el secreto inicial debe estar guardado en un lugar seguro.
- A la hora de verificar el estado de un registro en particular se debe calcular su clave en la cadena de hash. Al log guardado se le aplica la función no reversibles y se calcula el MAC. Si este no coincide con el almacenado, el registro fue modificado.
- Es igualmente válido utilizar una clave aleatoria para cada entrada en el registro. Problema: Se deben guardar todas las claves utilizadas.

Calcular MAC's

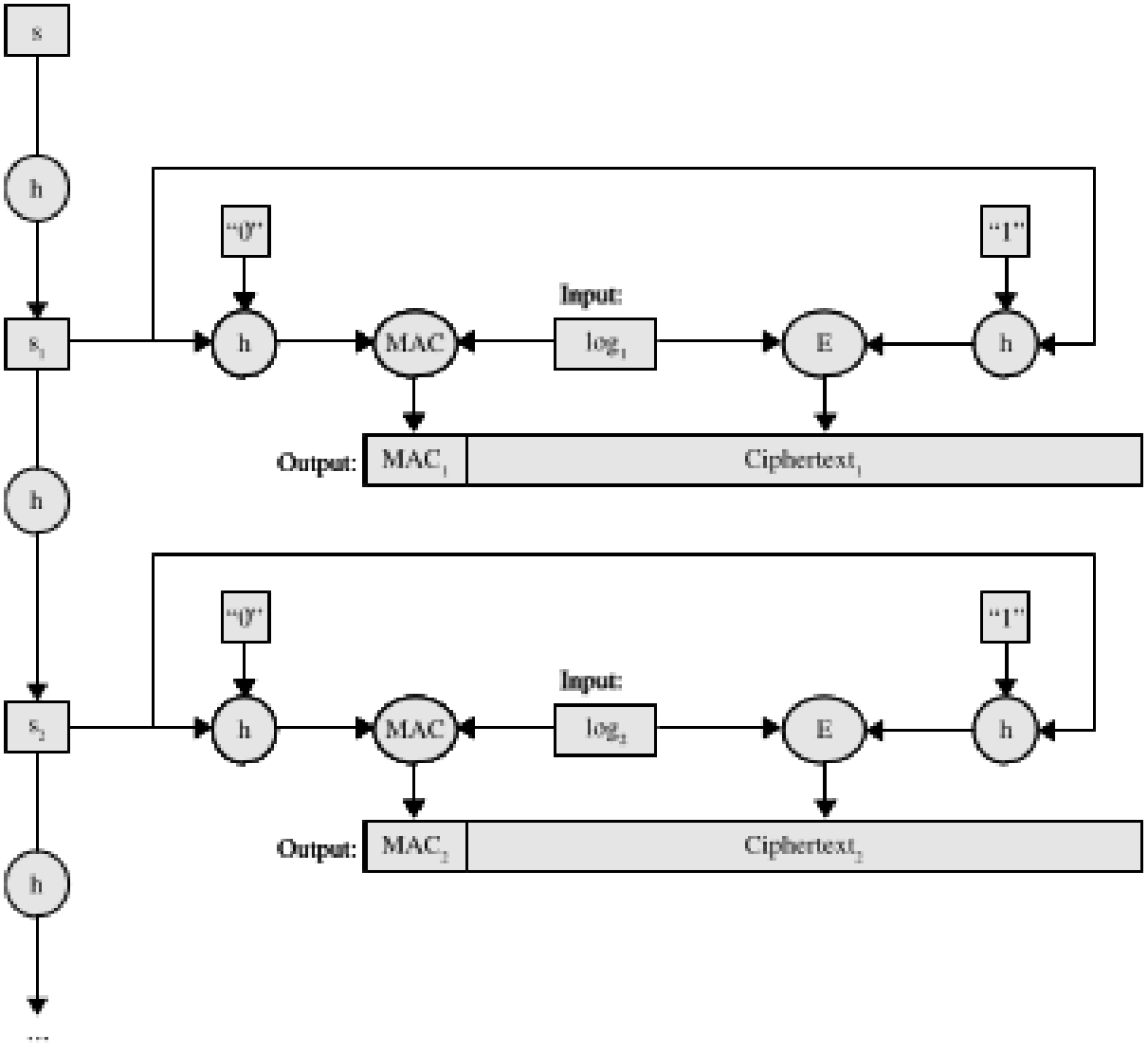




Verificación



Cifrado


s
h
"0"
"1"
Input:
s_1
h
MAC
$\log_1$
E
h
Output:
MAC_1
$Ciphertext_1$
h
"0"
"1"
Input:
s_2
h
MAC
$\log_2$
E
h
Output:
MAC_2
$Ciphertext_2$
h
...

Nuestra implementación

- Utilizamos un secreto inicial, elegido por el administrador del equipo, para iniciar el servidor syslog. Para cada log que se genere, se calcula su MAC y se guarda.
- El propio administrador realizara la verificación de los logs aportando el secreto inicial.
- Los registros de eventos se guardan en un ".log"de solo lectura para un usuario normal.
- Cada verificación realiza el procedimiento descripto para todos los logs en el registro.

Nuestra implementación

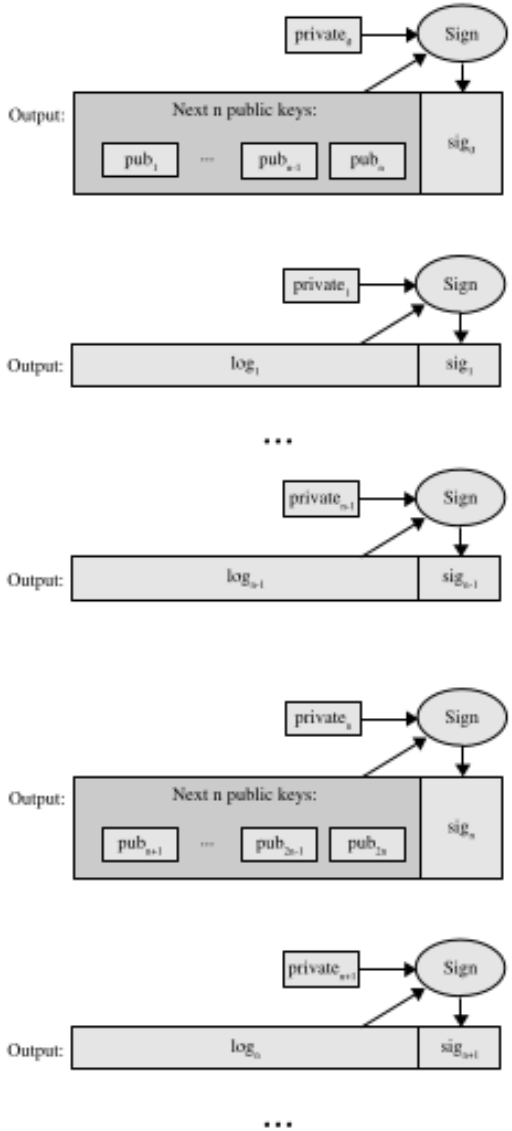
Demo

Alternativas

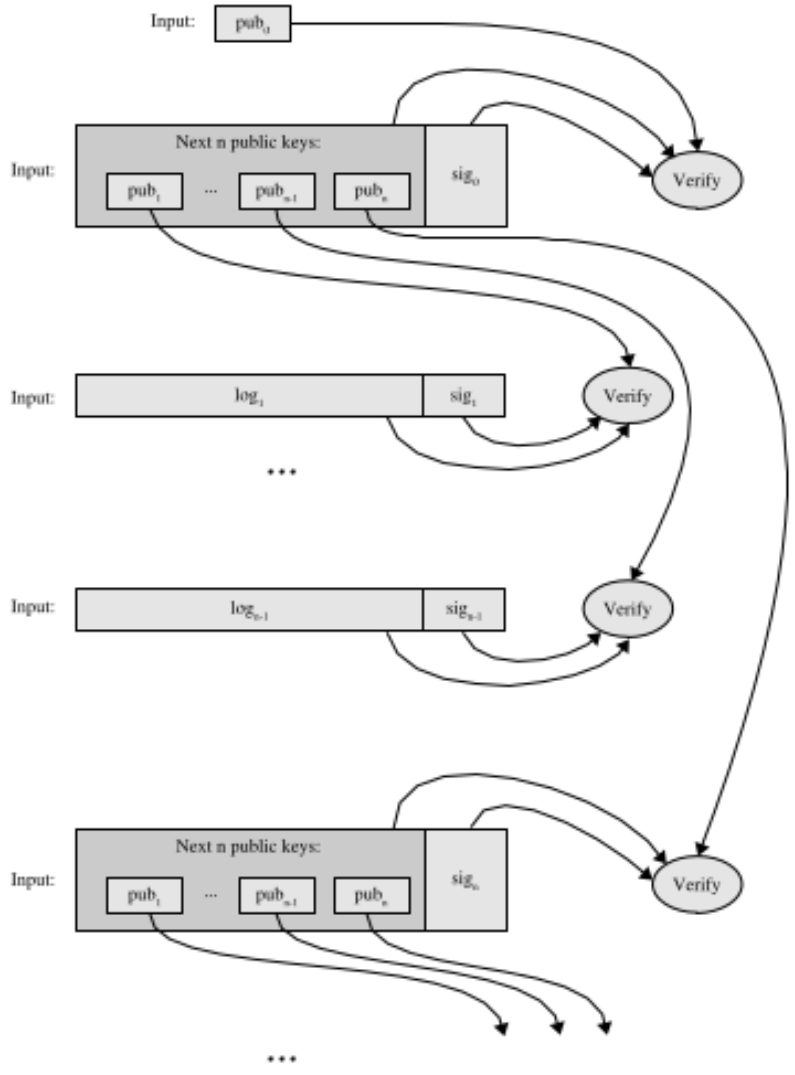
Sistema de clave pública: consiste en generar claves privadas para firmar los logs, y claves publicas para verificarlos.

- Su principal ventaja es que no se necesita las claves privadas para realizar la verificación.

Alternativas



Sistema de clave pública



Sistema de clave pública

- Criptografía de curva elíptica: es una alternativa a RSA para generar pares de claves para criptografía asimétrica. La ventaja es que las claves generadas son cortar, ideales para las entradas del registro.
- Identity-based signature(IBS): permite obtener las claves publicas derivandolas de bits puntuales en un string y en la clave pública de un denerador de claves privadas(PKG), la privada solo puede ser extraida de esa cadena de caracteres y de la clave privada del PKG. Con esto, lo que se necesita para la verificación es la siguiente clave pública del PKG. Con el otro modelo tenía que enumerar las claves públicas y pasarlas todas

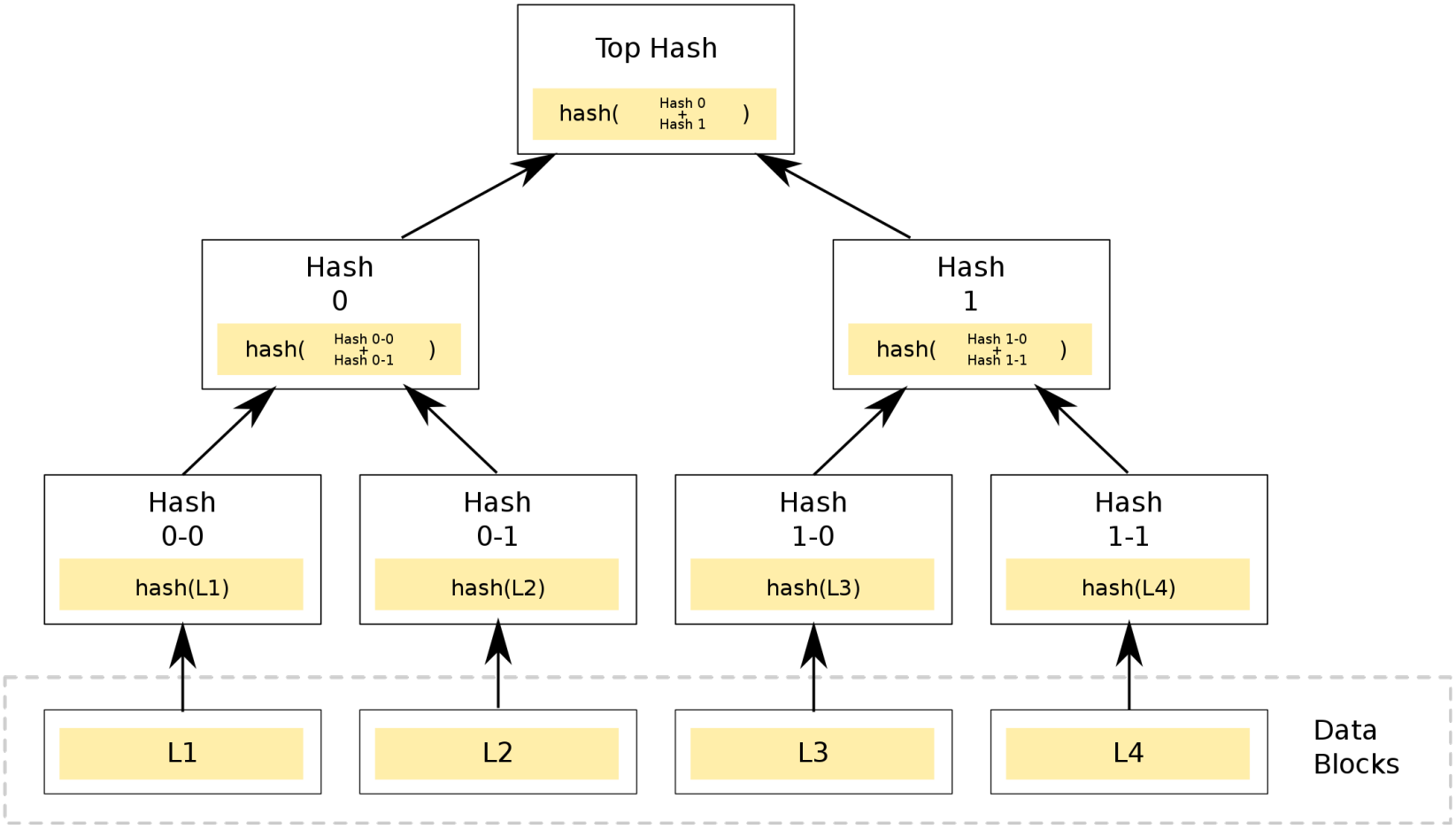
Alternativas en la verificación

- Existen dos alternativas a la hora de validar los logs:
  - Validarlos todos(nuestra opción).
  - Validar solamente los que no fueron validados anteriormente(acumulativamente).

Investigación sobre árboles de Merkle o árboles de hash

- Definición: Es un árbol donde cada nodo contiene el hash de los hashes de sus hijos y las hojas contienen el hash del valor a proteger. En el nodo raíz queda el hash que representa a todo el árbol, es decir que si alguno de los nodos modifica su hash invalidará el hash de los nodos padres.
- Fue patentado por Ralph Merkle en 1979, como método de firmas digitales para robustecer el sistema criptográfico presentado Diffie, Hellman y él dos años antes.

Estructura de Hash Tree



Principales características

- Merkle Audit Paths: Permite verificar si un bloque de datos protegido fue modificado con órden log(n), donde n es la cantidad de bloques de datos totales del árbol. Así sólo requiere calcular 2 * log2(n) hashes, correspondientes al camino desde la raíz al valor a validar junto con los hijos de cada nodo del camino. La misma operación en listas de hash tiene orden n.
- Merkle Consistency Proofs: Dada una lista de los primeros m bloques de datos permite verificar no se hayan borrado o agregado algunos, o modificado el órden entre ellos. Esta es una propiedad útil para mejorar el syslog server.

Algunos usos

- Certificate Transparency: Framework open source para el monitoreo y auditoría de certificados digitales. A partir de un registro público de la emisión de los certificados los clientes pueden verificar su validez.
- Redes p2p
  - Bitcoin: Optimiza el tamaño del blockchain guardando sólo el hash root de las transacciones cada bloque. Permite a los clientes escuchar de forma eficiente la publicación de transacciones asociadas a una billetera en particular.
  - BitTorrent: Valida cada bloque descargado independientemente del archivo completo.
  - Otros: Filesystems como IPFS, Sistemas de control de versiones como Git o Mercurial, bases de datos NoSQL como Cassandra o Dynamo.

Usos: Certificate Transparency

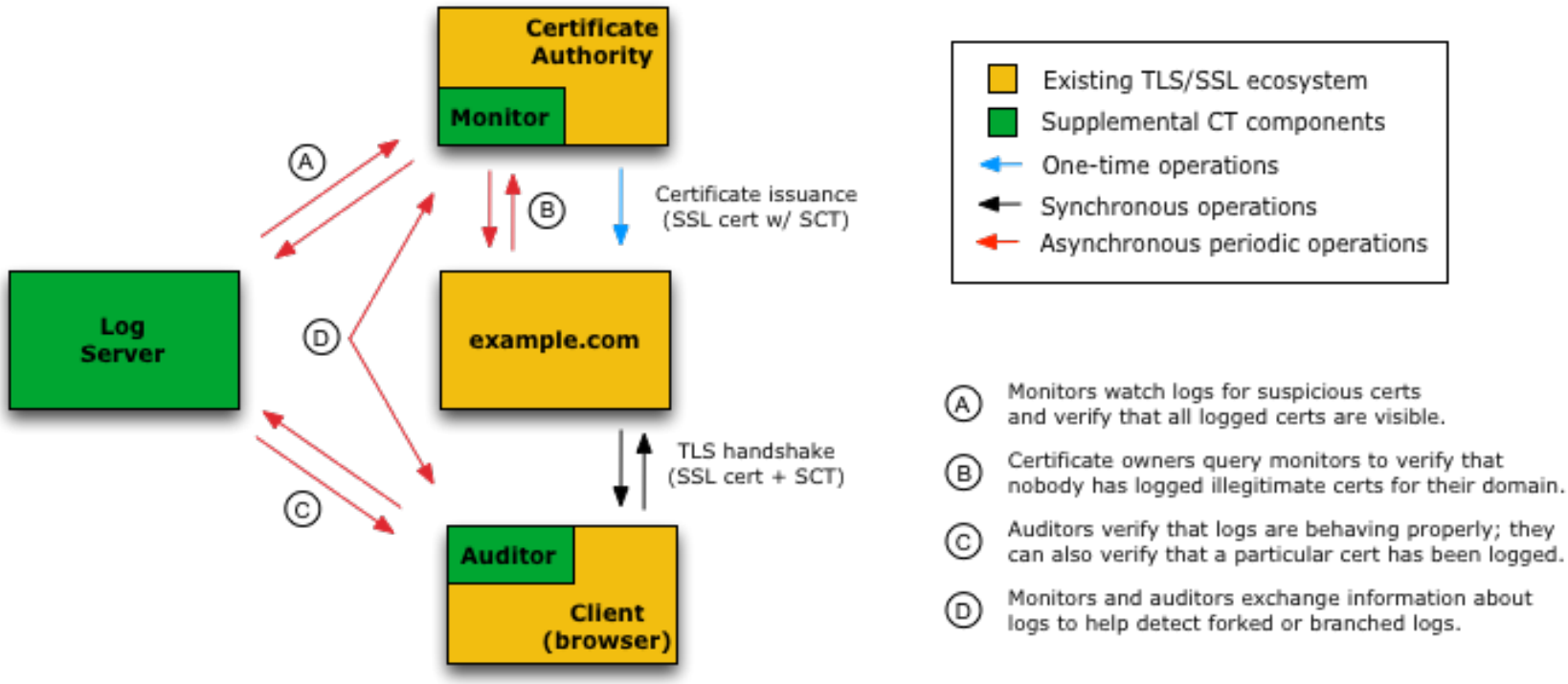


Figura: Arquitectura

Usos: Certificate Transparency

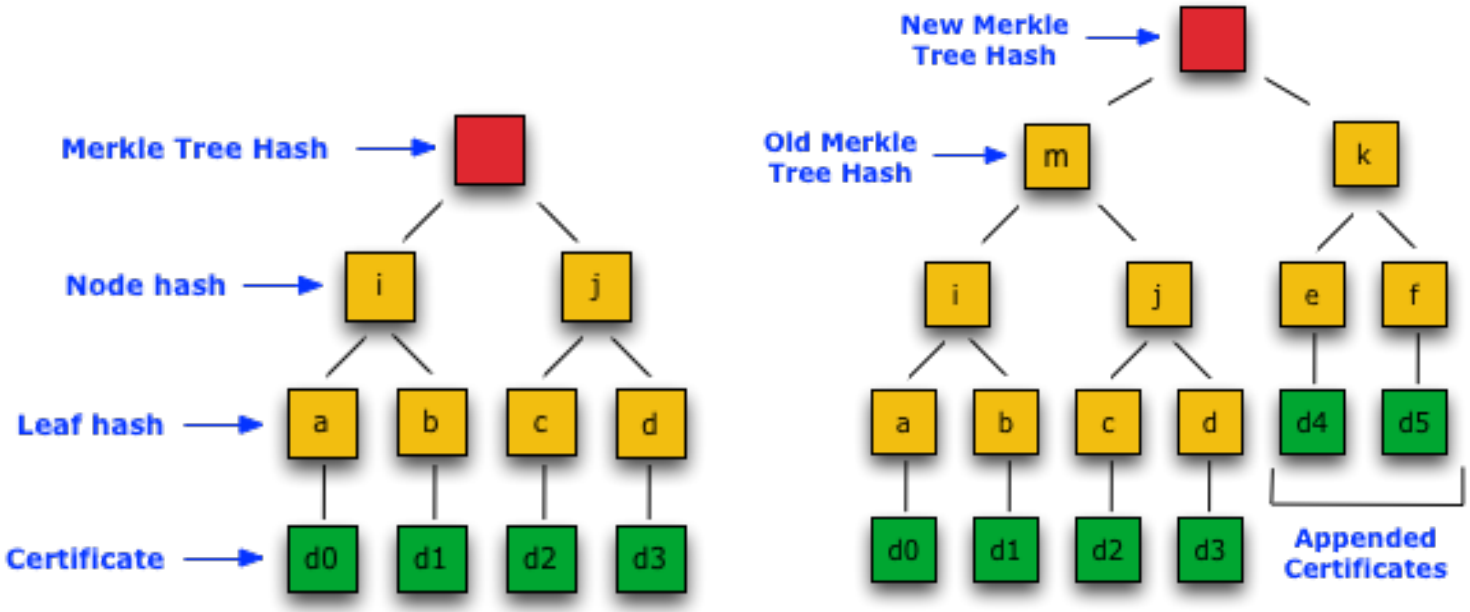
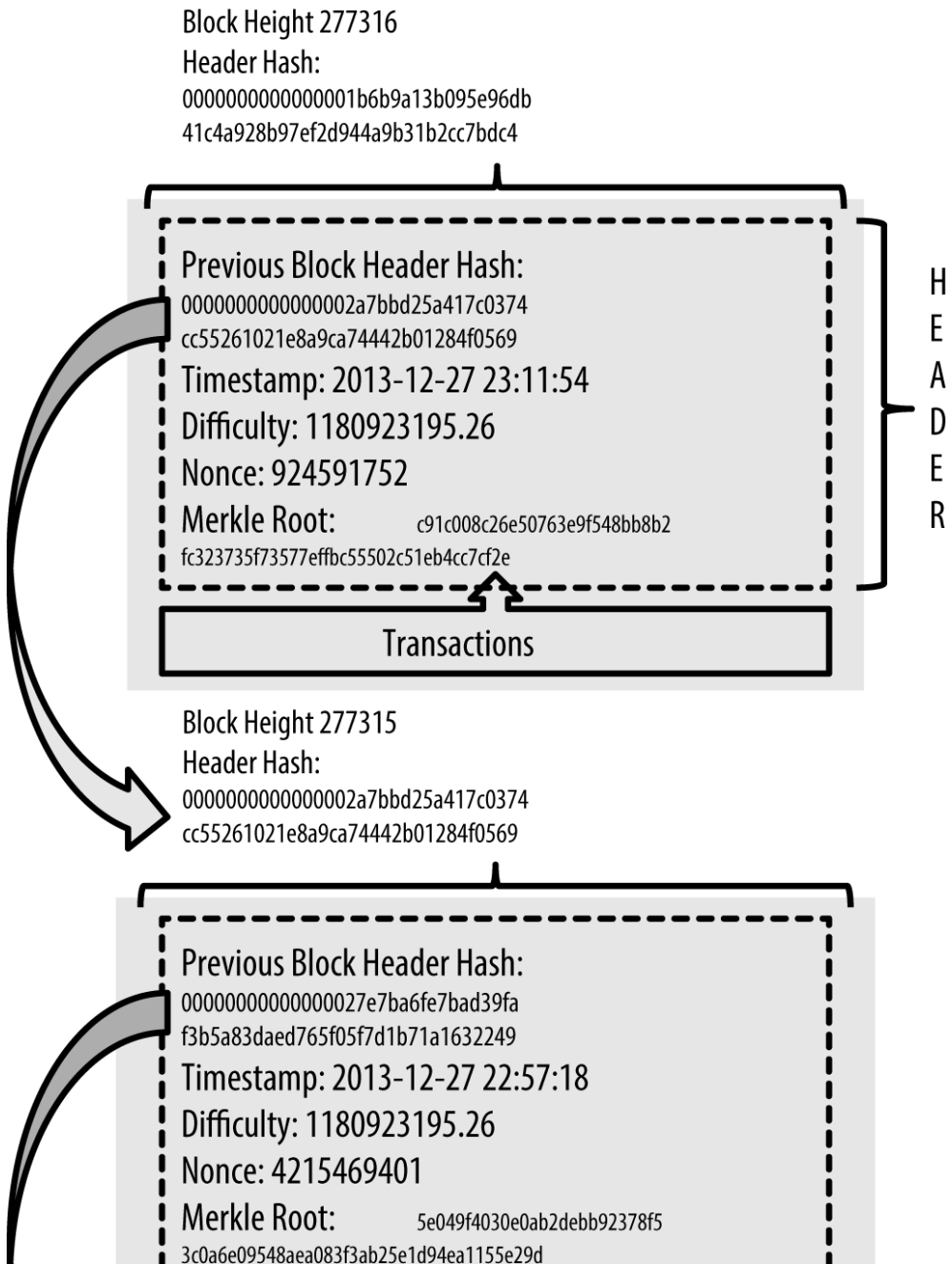


Figura: Construcción incremental del árbol

Usos: Bitcoin



Usos: Bitcoin

Table 7-3. Merkle tree efficiency

Number of transactions	Approx. size of block	Path size (hashes)	Path size (bytes)
16 transactions	4 kilobytes	4 hashes	128 bytes
512 transactions	128 kilobytes	9 hashes	288 bytes
2048 transactions	512 kilobytes	11 hashes	352 bytes
65,535 transactions	16 megabytes	16 hashes	512 bytes

En esta tabla se puede observar la eficiencia del árbol de Merkle según aumenta el camino de Merkle para validar que una transacción es parte de un bloque.
Así un nodo puede bajarse sólo los bloques de los headers (80 bytes por bloque) para esa validación y evita bajarse el blockchain completo que puede de varios gigabytes. Estos nodos, usados sólo para verificación de transacciones se llama simplified payment verification (nodos SPV).

Referencias

- Logcrypt: Forward Security and Public Verification for Secure Audit Logs de Jason E. Holt
- Forward Integrity For Secure Audit Logs de Mihir Bellare y Bennet S. Yee
- Wikipedia. Merkle Tree
- Certificate Transparency RFC 6962
- Mastering Bitcoin. Andreas M. Antonopoulos. Capítulo 7.