Criptografía - Ingeniería en Computación
Curso 2019

Funciones de Hash Segunda Parte

Hashing long messages

En la práctica usamos funciones de hash que toman como entrada pocos bits, en SHA-256 se toman 512 bits de entrada y como salida se devuelven 256 bits. A éstas funciones de hash le aplicamos alguna estrategia para poder aplicarlo a todo el mensaje.

Construcción **Merkle-Damgård** (1990)

Se define la función de hash h: $\mathbb{B}^{\ell} = \{0,1\}^{\ell} \rightarrow \mathbb{B}^{n}$, con $\ell \geq n + 2$.

El mensaje x tiene largo κ, con $\kappa > \ell$.

Idea:

1. $h(\ell\text{-bits})$ [vector de inicialización] dando como resultado n-bits (y)
2. $h(y \,||\, (\ell - n)$ bits de x)
3. *Así hasta calcular el hash para todo x*

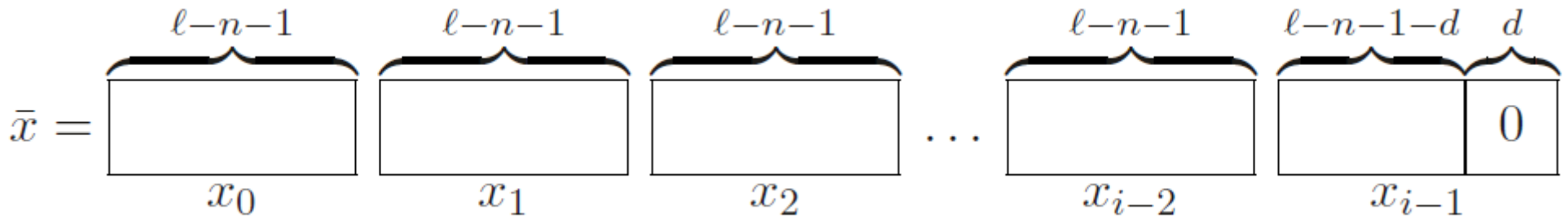
Tendremos $\kappa/(\ell - n)$ pasos

Hashing long messages - **Merkle-Damgård**

Definimos

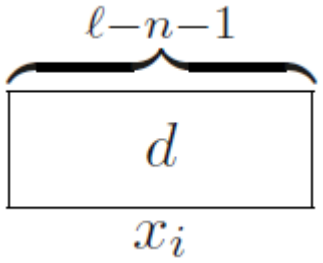
- $i = \lceil \not{k}/(\ell - n - 1) \rceil$, con lo cual x tiene i-bloques de tamaño $(\ell - n - 1)$
- $d = i(\ell - n - 1) - \not{k}$, es el exceso

Le agregaremos a x d-ceros, con lo cual el largo del nuevo mensaje $\bar{x}$ va a ser múltiplo i de $(\ell - n - 1)$



Hashing long messages - **Merkle-Damgård**

Agregaremos por último,
un bloque con la representación binaria de d con tamaño $(\ell - n - 1)$



En resumen, tendremos la siguiente función de hash

$$h^*: \bigcup_{\ell>0} \mathbb{B}^\ell \to \mathbb{B}^n$$

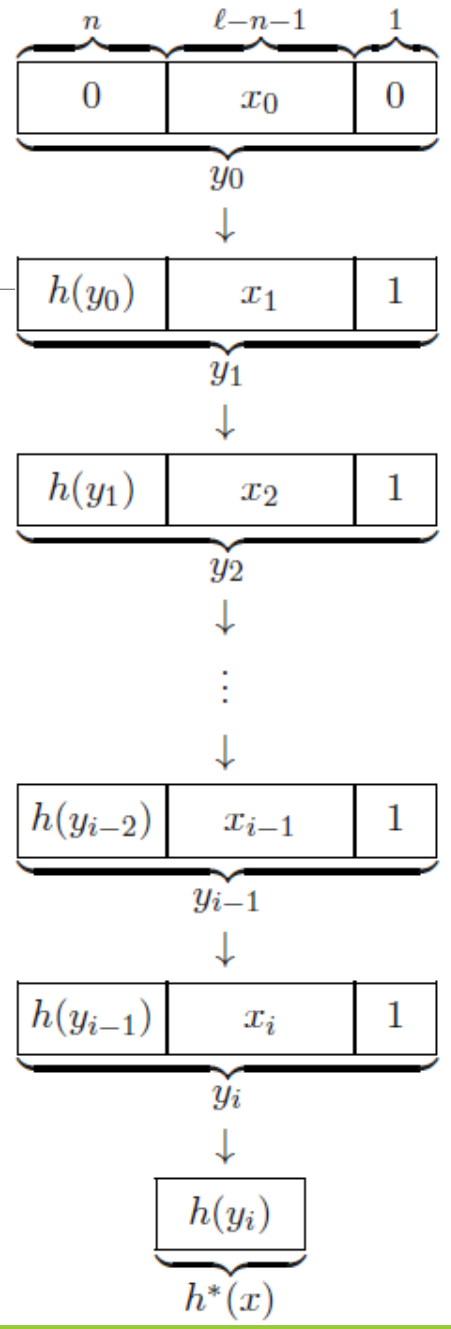
Hashing long messages
Merkle-Damgård

Tendremos que evaluar h exactamente
$1 + \lceil \mathcal{k}/(\ell - n - 1) \rceil$ para calcular h^*

<u>Teorema</u>

Una colisión para h^* produce una colisión para h

Es decir, si h es resistente a colisiones, también lo será h^*



Timestamps

Supongamos que un atacante obtiene la clave privada de un usuario, comprometiendo todos los documentos que éste ha firmado alguna vez!

El uso de timestamps en un esquema de firma digital puede prevenir que esto ocurra, es decir que se comprometan solamente los documentos que fueron firmados después de cierta fecha (la fecha que el atacante obtuvo la clave privada)

Hay muchas formas de utilizar timestamps, una puede ser la definida en el RFC 3161 (https://tools.ietf.org/html/rfc3161), el cual define un protocolo para el uso de timestamps en una infraestructura de clave pública.

Timestamps – RFC 3161
Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)

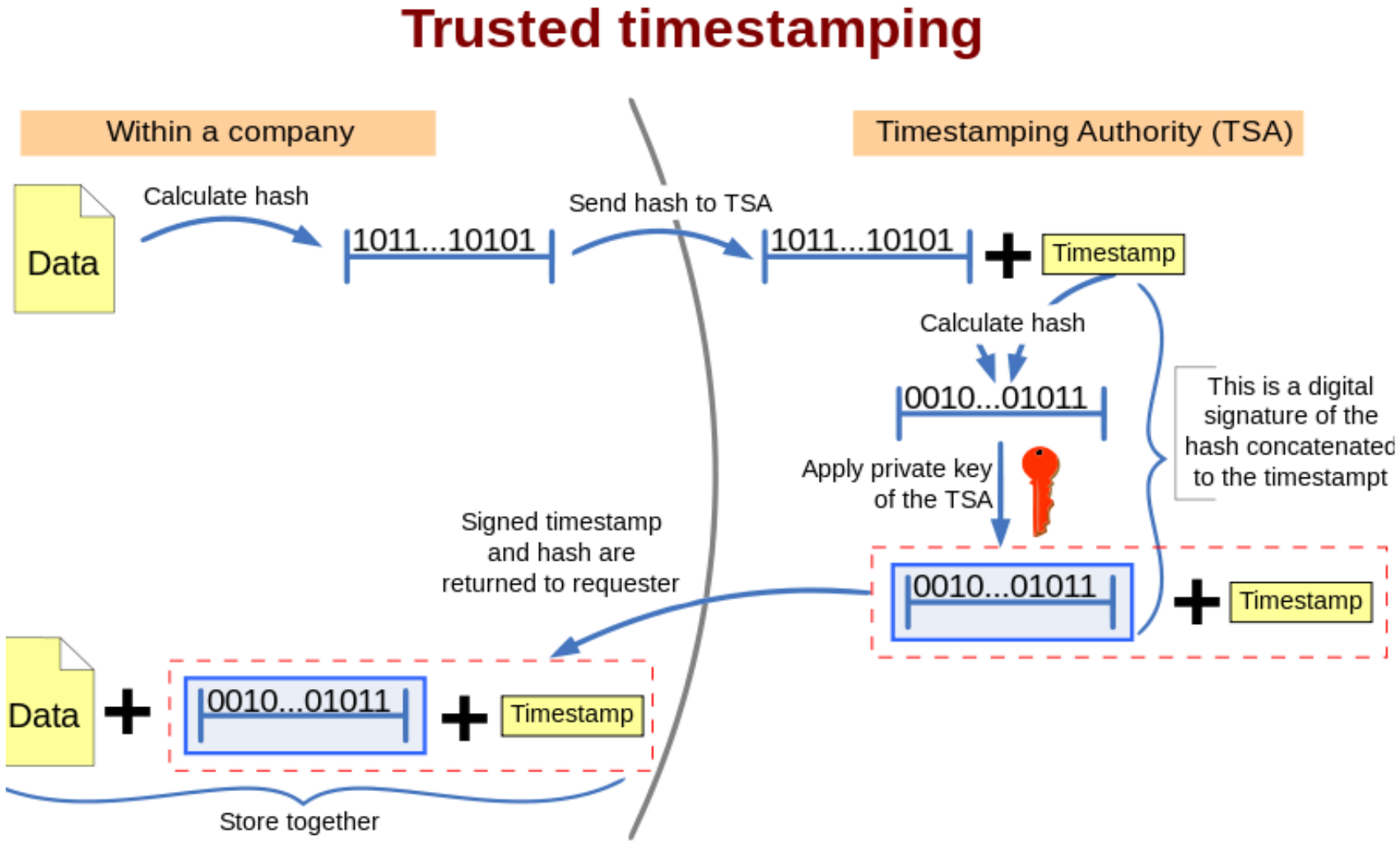
Espera como entrada el hash de un mensaje (documento) para aplicar el timestamp

Produciendo como salida el token correspondiente al timestamp del documento

Básicamente el token de salida es una estructura de datos que consiste en:

- Metadatos (versión, política utilizada)
- La fecha en la cual se realizó el proceso
- Un identificador
- El hash del documento

Timestamps


Trusted timestamping
Within a company
Timestamping Authority (TSA)
Data
Calculate hash
1011...10101
Send hash to TSA
1011...10101
Timestamp
Calculate hash
0010...01011
This is a digital signature of the hash concatenated to the timestampt
Apply private key of the TSA
Signed timestamp and hash are returned to requester
0010...01011
Timestamp
Data
0010...01011
Timestamp
Store together

MD4/MD5/MD6/SHA-3

La familia de funciones de hash MD (message digest) fue de las más utilizadas en cierto momento.

Diseñadas por R. Rivest en 1990. Ya en 1998 se conocen los primeros ataques realizados por Dobbertin, en dónde colisiones pueden ser hayadas en microsegundos para MD4.

Desde 1992 MD5 ha sido utilizado como standard en internet para firmas digitales y certificados SSL.

MD6 surge como propuesta al llamado realizado por el NIST para SHA-3, desarrollado por un equipo del MIT liderado por Ronald Rivest (2007-2012)

- Pueden ver el llamado del NIST en: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html
- Pueden ver más sobre MD6 en: http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/documents/MD6HashFunction.pdf
- Pueden ver más sobre SHA-3 en: http://csrc.nist.gov/groups/ST/hash/sha-3/documents/Keccak-slides-at-NIST.pdf y en http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

Ejemplo de colisión en MD5

- En el año 2005, Xiaoyun Wang y Hongbo Yu de la Universidad Shandong de China describieron un algoritmo que podía encontrar dos diferentes secuencias de 128 bytes con el mismo valor de hash MD5.
- Vamos a ver un ejemplo de colisión creado a partir del algoritmo de Wang y Yu para construir archivos de longitud arbitraria con idéntico valor de hash MD5 y que difieren solo en 128 bytes en algún lugar en el medio.
- Magnus Daum y Stefan Lucks crearon dos archivos postcript con idéntico valor de hash pero con una condición en un if que hace que se muestren dos mensajes muy distintos!

- Hash(carta1.ps) = A25F7F0B29EE0B3968C860738533A4B9
- Hash(carta2,ps) = A25F7F0B29EE0B3968C860738533A4B9

Muchas gracias!