Criptografía y Seguridad Computacional **2016-01**

Clase 9: 04/05/2016

Fernando Krell *Diego Peña*

1. Firmas Digitales

Criptografía Simétrica (llave privada)	Criptografía Asimétrica (llave pública)
llave: k	llave: $\langle pk, sk \rangle$
$\mathsf{Enc}_k()$ misma llave que $\mathsf{Dec}_k()$ Relación 1-1	Múltiples $\mathsf{Enc}_{pk}()$ para un $\mathsf{Dec}_{sk}()$ Relación N-1

Para la criptografía simétrica hemos estudiado esquemas de autenticación de mensajes. Estos esquemas están compuestos por la tupla de algoritmos $\mathsf{Gen}()$, $\mathsf{Mac}_k()$ y $\mathsf{Vrfy}_k()$. Dado un mensaje m y un tag t, supuestamente generado utilizando $\mathsf{Mac}_k(m)$, aceptamos m si, y sólo si, $\mathsf{Vrfy}_k(m,t) = 1$. Intuitivamente, aceptamos el mensaje si este fue generado por un remitente válido (es decir, que conoce la llave k). ¿Qué paralelo existe para autenticación de mensajes en criptografía asimétrica? Las firmas digitales, donde un usuario firma el mensaje con la llave secreta ($\mathsf{Sign}_{sk}()$), y varios usuarios pueden verificar un mensaje en base a la correspondiente llave pública (utilizando $\mathsf{Vrfy}_{pk}()$). Formalmente, un esquema de firmas digitales Π esta compuesto por 3 algoritmos:

- $\mathsf{Gen}(1^\lambda)$: Algoritmo probabilista generador de llaves. Dado como input el parámetro de seguridad λ, genera llave pública pk y correspondiente llave privada sk.
- $\mathsf{Sign}_{sk}(m)$: Algoritmo (posiblemente probabilista) de firmas digitales. Dada la llave privada sk y mensaje m, genera la firma σ.
- $\mathsf{Vrfy}_{pk}(m, \sigma)$: Algoritmo verificador de firmas. Dada la llave *pública* pk, mensaje m y firma σ, retorna 1 o 0.

Requerimos que para todo par (pk, sk) generable por Gen, y para todo mensaje m, si $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$, entonces $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$. Previo a definir la propiedad de seguridad que queremos, intentaremos construir un esquema de firmas digitales basado en el supuesto RSAy analizaremos informalmente su seguridad.

- $\mathsf{Gen}(1^\lambda)$: $\langle N, e, d\rangle \leftarrow GenRSA(1^\lambda)$. Output $pk = \langle N, e\rangle$, $sk = \langle N, d\rangle$
- $\mathsf{Sign}_{sk}(m)$: Output $\sigma \leftarrow m^d \mod N$.
- $\mathsf{Vrfy}_{pk}(m, \sigma)$: Output 1 si, y sólo si, $\sigma^e = m \mod N$.

Es fácil de ver que el esquema es correcto. Pero, ¿es este seguro?. A primera vista, pareciera que sí lo es, pués el supuestro RSA nos dice que encontrar $\sigma = m^d \mod N$ tal que $\sigma^e = m$ es computacionalmente díficil, y por lo tanto, un adversario no debería poder generar una firma válida. Sin embargo, el supuesto RSA especifica que el input m tiene que ser uniformemente aleatorio. En la práctica, el mensaje puede venir de una distribución arbitraria que depende de la aplicación sobre la cual estemos usando nuestro esquema de firmas. A continuación describiremos una definición de seguridad suficientemente fuerte de manera tal los esquemas que cumplan con tal definición pueden ser usados en variadas aplicaciones. En la práctica, la firma a falsificar es sobre un mensaje elegido por el adversario. Por lo tanto, nuestra definición debe capturar este hecho. Además, es bastante factible que el adversario pueda obtener firmas válidas de otros mensajes (de hecho las firmas en general son de carácter público). Pero más aun, es posible que el adversario tenga cierta influencia sobre que mensajes van a ser firmados. Es bastante difícil intuir que tipo de mensajes el atacante puede influenciar ya que esto depende de la aplicación en cuestión. Para suponer lo menos posible de la aplicación y dar una definición global fuerte, daremos al adversario el poder de elegir no tan solo el mensaje a falsificar su firma, si no también firmas de mensajes a su elección. Es decir, describiremos seguridad frente a ataques de mensajes escogidos (CMA). Requeriremos que el adversario no puede generar una firma válida para un mensahe a su elección, aun teniendo acceso a firmas válidas para mensajes escogidos por el adversario.

Sea $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ un esquemas de firmas digitales. Para definir formalmente la descripción anterior de seguridad, se plantea el experimento $\mathsf{Exp}^{falsFirma}_{\mathcal{A},\Pi}(\lambda)$ ilustrado en la Figura **??**. El experimento comienza generando el par de llaves (pk, sk) utilizando el algoritmo generador de llaves Gen. Para modelar el hecho que el adversario puede obtener firmas válidas para mensajes influenciados por este, el experimento utiliza la llave privada sk para dar al adversario acceso a un oráculo de firmas para ser usado arbitrariamente. El oráculo es implementado de la siguiente manera: El adversario envía un mensaje m_i y el experimento utiliza la llave privada para computar la firma como $\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i)$, retornandole la firma al adversario. El adversario, por supuesto, también tiene a su disposición la llave pública. Decimos que el adversario gana si puede generar una firma válida para un mensaje que no haya sido preguntado al oráculo.

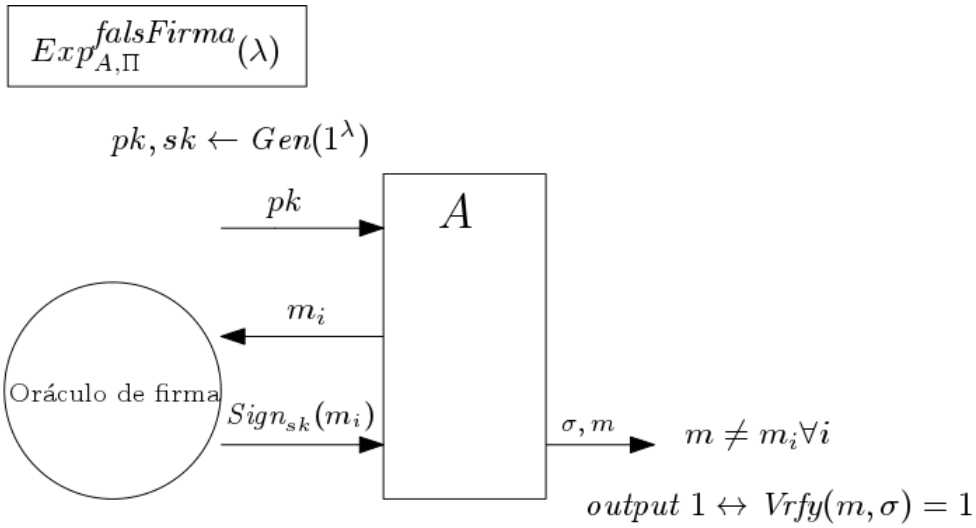

Figura 1: Experimento de falsificación de firmas

Definición 1. *Un esquema de firmas digitales* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *es infalsificable frente a ataques de mensajes escogidos (o CMA-seguro) si para todo algoritmo PPT* $\mathcal{A}$*, existe una función negligible* negl *tal que para todo* λ *suficientemente grande:*

$$\Pr[\mathsf{Exp}^{falsFirma}_{\mathcal{A},\Pi}(\lambda) = 1] \leq negl(\lambda)$$

Para el esquema RSA visto anteriormente, tenemos los siguientes ataques que quiebran su seguridad CMA.

- Ataque 0 (sin utilizar el oráculo): Ataque trivial: Output $m = 1, \sigma = 1$.
- Ataque 1 (sin utilizar el oráculo): Dado un m cualquiera, puede calcular m^e utilizando la llave pública. Por lo tanto m es una firma válida para el mensaje m^e. Ouput $m' = m^e, \sigma = m$
- Ataque 2:

  $$m_1 \rightarrow \sigma_1 \ \textit{vía oráculo}$$
  $$m_2 \rightarrow \sigma_2 \ \textit{vía oráculo}$$

  Output $m = m_1 \times m_2 \mod N, \sigma\sigma_1 \times \sigma_2 \mod M$

Esto es una clara vulnerabilidad. ¿Cómo podemos solucionarlo? Utilizaremos funciones de hash de forma de prevenir ataques algebraicos como los anteriores.

$$\mathsf{Sign}_{sk}(m): \ 1.\hat{m} \leftarrow H(m) 2.\text{Output: } \sigma = \hat{m}^d \mod N \ \textit{(RSA-FDH)}$$

El esquema anterior de llama RSA-FDH (*full-domain hash*). la función H debe cumplir las siguientes propiedades:

- Difícil de invertir
- Resistente a colisiones
- Mapea m a un elemento impredecible. Es decir, es difícil computar m, m_1, m_2 tales que, por ejemplo,
$$H(m) = H(m_1)H(m_2)$$

Teorema 2. *Si H es un oráculo aleatorio y el supuesto RSA se cumple entonces RSA-FDH es seguro.*

Demostración. Sea $\mathcal{A}$ un adversario para RSA-FDH. Construimos un algoritmo $\mathcal{B}$ que quiebra RSA (asumamos primero que $\mathcal{A}$ no hace ninguna llamada al oráculo $\mathsf{Sign}_{sk}(\cdot)$):

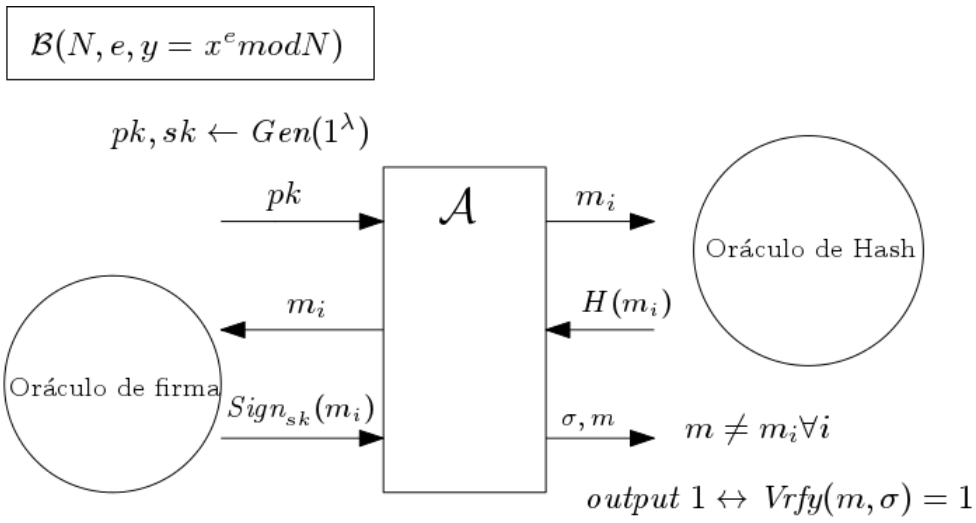

Figura 2: Variación sobre el experimento `falsFirma`. Algoritmo $\mathcal{B}$

Sea q la cantidad de llamdas que $\mathcal{A}$ hace al oráculo H para poder falsificar la firma. Como H es oráculo aleatorio, es fácil argumentar de que $q >= 1$ y que una de estas

llamadas tiene que ser m (output del adversario). Pues de lo contrario, la firma es completamente independiente del mensaje, y por lo tanto $\mathcal{A}$ sólo puede ganar con probabilidad negligible.

La idea general es que utilizaremos $y = x^e$ como el valor $H(m)$, de manera tal de que si el adversario arroja una firma válida, esta corresponde a x, quebrando el supuesto RSA. El problema que tenemos es que no sabemos cuál de estas q llamadas a H $(m_1, m_2, ...m_q)$ corresponde al mensaje m. La solución es adivinar i tal que $m_i = m$ con probabilidad $1/q$.

Para cada llamada $j \neq i$ a H $\mathcal{B}$ retorna un elemento uniformemente aleatorio en $\mathbb{Z}_N^*$ ($y_j \xleftarrow{\$} Z_N^*$ *para* $j \neq i$). Para la i-ésima llamada retornamos y. Esto significa que cuando el adversario arroje la falsificación de la firma $(\mathcal{A} \to \sigma, m)$, entonces

$$H(m) = y, \sigma = y^d$$

Pero $(y^d)^e = y$, por lo tanto σ quiebra RSA ($\sigma = x \mod N$). La probabilidad de adivinar i es

$$\Pr[\textit{adivinar } i] = \frac{1}{q} = \frac{1}{poly(\lambda)}$$

Luego $\Pr[\textit{quebrar RSA}] = \frac{\Pr[\textit{quebrar } \Pi]}{q}$ (tenemos la reducción pues quebrar uno es tan difícil como quebrar el otro).

Hemos asumido que $\mathcal{A}$ no utiliza el oráculo de firmas. ¿Cómo arreglamos la reducción anterior para el caso de que $\mathcal{A}$ ocupase tal oráculo?

Tenemos que simular llamadas al oráculo. Asumimos que $\mathcal{A}$ llama a $H(m_j)$ antes de llamar a $\mathsf{Sign}_k(m_j) \forall j$. Para la j-ésima llamada a $H(m_j)$

$$\sigma_j \leftarrow Z_N^*, y_i = \sigma_i^e \mod N$$

Guardamos la tupla (m_j, y_j, σ_j) y damos como output $y_j \forall j \neq i$. Cuando $j = i$, output y.

Ahora tenemos que responder las preguntas al oráculo, lo cual es fácil pues tenemos todas las consultas guardadas. Para cada llamada a $\mathsf{Sign}_{sk}(m_j)$:

- Si $i \neq j$, output σ_j (ya guardado)
- Si $i = j$ abortamos.

Por lo tanto, hemos reducido la seguridad CMA de RSA-FHD a quebrar el supuesto RSA. □

Firmas de Schnorr

Ahora veremos un nuevo esquema de firmas digitales basado en el problema del logaritmo discreto. No tan solo este nuevo esquema es interesante por utilizar un supuesto distinto, también la demostración utiliza una técnica bastante particular y poderosa no vista antes en clases: la técnica de rebobinar el adversario.

Recordatorio: $DLog(g, g^x)$. Dado la descripción de un grupo $\mathcal{G}$, un generador del grupo g, el orden del grupo q, y un elemento aleatorio y, es computacionalmente difícil de obtener x tal que $g^x = y$.

Sea $H : \{0,1\}^* \to \mathbb{Z}_q$. El esquema de firmas de Schnorr es el siguiente:

- $\mathsf{Gen}(1^\lambda)$: 1) $(\mathbb{G}, g, q) \leftarrow \mathcal{G}$. 2) $x \xleftarrow{\$} \mathbb{Z}_q$. 3) Output $sk = x, pk = g^x$
- $\mathsf{Sign}_k(m)$:1) $k \xleftarrow{\$} Z_q$. 2) $e = H(m, g^k)$. 3)$s = k + ex$ 4) Output $\sigma = \langle s, e \rangle$.
- $\mathsf{Vrfy}_{pk}(m, \sigma)$: 1) $\sigma = \langle s, e \rangle$ 2) $R = g^s \cdot (g^x)^{-e}$ 3). Output $H(m, R) \stackrel{?}{=} e$

Idea: Computar g^k a partir de g^x, s y e. Luego comparo $H(m, g^k)$ con e.

¿Es seguro? Ocupemos la técnica de rebobinar al adversario:

Teorema 3. *Si H es un oráculo aleatorio y DLog se cumple con respecto a $\mathcal{G}$ entonces el esquema de firmas de digitales de Schorr es CMA-seguro.*

Demostración. Describiremos un algoritmo $\mathcal{B}(G, g, g^x, q)$ para el logaritmo discreto utilizando un adversario $\mathcal{A}$ que falsifica firmas en el experimento $\mathsf{Exp}^{falsFirmas}_{\Pi,\mathcal{A}}$.

Idea general: Obtener 2 firmas $\sigma = \langle s, e \rangle$ y $\sigma' = \langle s', e' \rangle$ tales que $g^{s+xe} = g^{s'+xe'}$, y por lo tanto el logaritmo discreto de g^x es $x = (s - s') \cdot (e' - e)^{-1} \mod q$. Para esto ejecutamos el adversario 2 veces, el punto clave es asegurar que las firmas obtenidas tengan la propiedad deseada. Dejaremos este problema para el final de la demostración.

Reducción: Utilizaremos el valor g^x como llave pública para simular el experimento para el adversario $\mathcal{A}$. Al ejecutar $\mathcal{A}$ debemos responder preguntas al oráculo H y al oráculo de firmas *sin saber la llave privada x*. Sea q una cota superior a la cantidad de llamadas realizadas por el oráculo.

- $H(x)$: Si $T[x] = \perp$, $T[x] \xleftarrow{\$} \mathbb{Z}_q$. Retornar $T[x]$ al adversario.
- $\mathsf{Sign}_{sk}(m_i)$: Obtenemos s_i, e_i uniformes en $\mathbb{Z}_q$, y definimos $T(m_i || g^{s_i - xe_i}) = e_i$ (precomputamos el hash y lo guardamos en la tabla T). De esta manera la firma s_i, e_i es una firma válida para el mensaje m_i. Un punto importante a notar es que como s_i y e_i son uniformes e independientes, entonces con alta probabilidad el valor $m_i || g^{s_i - xe_i})$ no fue consultado al oráculo H anteriormente.

El adversario finalmente retorna un mensaje $m \neq m_i \forall i \neq q$, y una firma $\sigma\langle s, e\rangle$ tal que $H(m||g^{s-xe}) = e$ con probabilidad no negligible. Al igual que en el caso de RSA, como H es oráculo aleatorio, se puede deducir que el adversario *tiene* que haber preguntado por el valor de $H(m||g^r)$ con $g^r = g^{s-xe}$ al oráculo H.

El paso clave de la reducción es: Ejecutamos el adversario nuevamente de manera de obtener una nueva firma s', e'. Pero ¿Cómo aseguramos que $g^{s-xe} = g^{s'-xe'}$?. Esto se puede hacer "rebobinando" al adversario. El adversario es sólo un programa que podemos ejecutar paso a paso y guardar su estado en cada momento. Por lo tanto, podemos ejecutar normalmente el adversario la primera vez guardando su estado paso a paso. Cuando el adversario retorna el mensaje m y su firma $\sigma = \langle s, e\rangle$, podemos retroceder el adversario justo hasta el momento en el que consulta al oráculo H con input $H(m||g^r)$ $(= H(m||g^{s-xe}))$, pero en vez de retornar e como la primera vez, retornamos un elemento independiente y uniforme e'. Luego seguimos la ejecución hasta que el adversario retorna la nueva firma $\sigma' = (s', e')$. Esta vez, sabemos que $g^{s'-xe'} = g^r = g^{s-xe}$, y por lo tanto podemos retornar $x = (s - s') \cdot (e - e')^{-1} \mod q$ como el logaritmo discreto de g^x.

□

DSA: Digital Signature Algorithm

El esquemas de firmas DSA descrito a continuación es el estandar recomendado por el NIST. Sea $\mathbb{G}$ un grupo de orden primo q en donde el problema DLog es difícil, y g un generador para $\mathbb{G}$.

- $\mathsf{Gen}(1^\lambda)$ $x \xleftarrow{\$} \mathbb{Z}_q$, Output $pk = g^x$, $sk = x$
- $\mathsf{Sign}_{sk}(m)$:
  1. $k \xleftarrow{\$} \mathbb{Z}_q,\ r = F(g^k)$
  2. $s = k^{-1}(H(m) + rx)$
  3. Output $\langle s, r\rangle$
- $\mathsf{Vrfy}_{pk}(m, \langle s, r\rangle)$: $r \overset{\$}{=} F(g^{H(m)s^{-1}} g^{xs^{-1}r})$

Correctitud:

$$\begin{aligned} H(m)s^{-1} + xs^{-1}r &= H(m) \cdot k \cdot (H(m) + rx)^{-1} + rxk(H(m) + rx)^{-1} \\ &= k(H(m) + rx)^{-1}(H(m) + rx) \\ &= k \end{aligned}$$

Teorema 4. *Si H y F son oráculos aleatorios, entonces DSA es CMA-seguro.*

En la práctica H es SHA-1 y $F(g^k) = g^k \mod q$.