IIC3253

HMAC y KDF

Partimos con AES como ejemplo de cifrado

→Modos de operación para largo arbitrario

→Función de compresión (Davies-Meyer)

→Función de hash (Merkle-Damgård)

→MAC?

HMAC

Hash-based message authentication code

Construyendo un MAC M en
base a una función de hash h

¿Qué pasa si definimos $M(k, m) = h(k||m)$?

Recordemos el juego que definía un buen MAC

1. El verificador genera una llave k
2. El adversario envía $m_0 \in \mathcal{M}$
3. El verificador responde $M(k, m_0)$
4. Los pasos 2 y 3 se repiten tantas veces como quiera el adversario
5. El adversario envía (m, t), siendo m un mensaje que no se había enviado antes

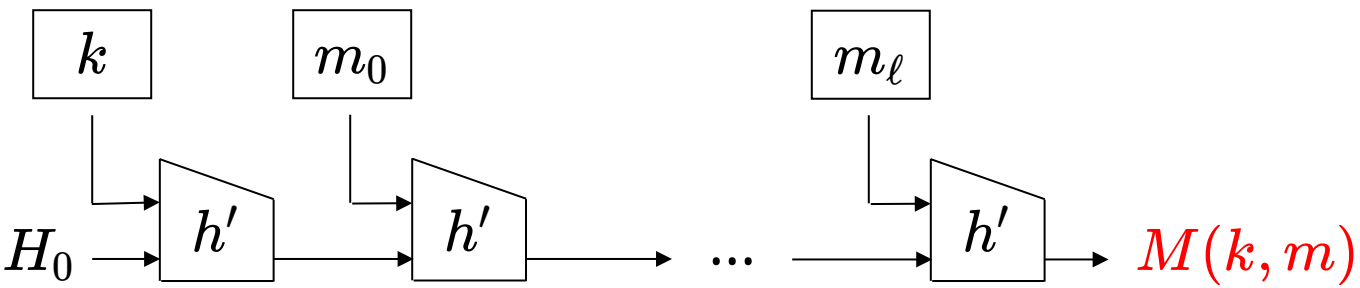
El adversario gana si $Verify(k, t, m) = 1$

¿Qué pasa si definimos $M(k, m) = h(k||m)$?

Pensemos que estamos usando SHA-2

¿Puede el adversario ganar el juego?

Simplificación: el largo de k es un bloque



Donde $k\ m_0\ \cdots\ m_\ell$ es en realidad $pad(k||m)$

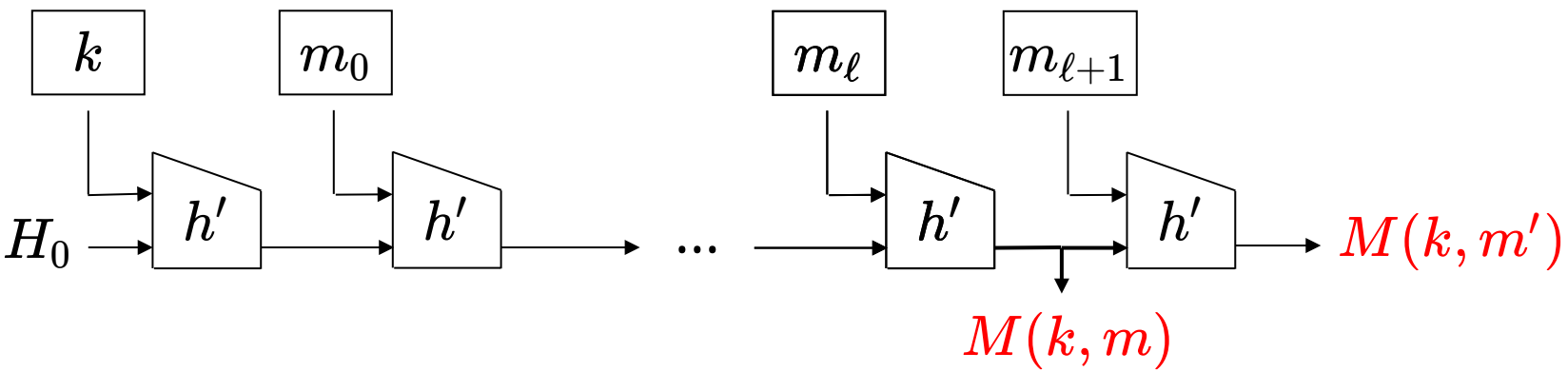
En particular $m \neq\ m_0\ \cdots\ m_\ell$

¿Cómo se ve $pad(pad(k||m))$?

$$pad(pad(k||m)) = k\ m_0\ \cdots\ m_\ell\ m_{\ell+1}$$

$$pad(pad(k||m)) = k\ m_0\ \cdots\ m_\ell\ m_{\ell+1}$$

Por lo tanto el adversario puede calcular



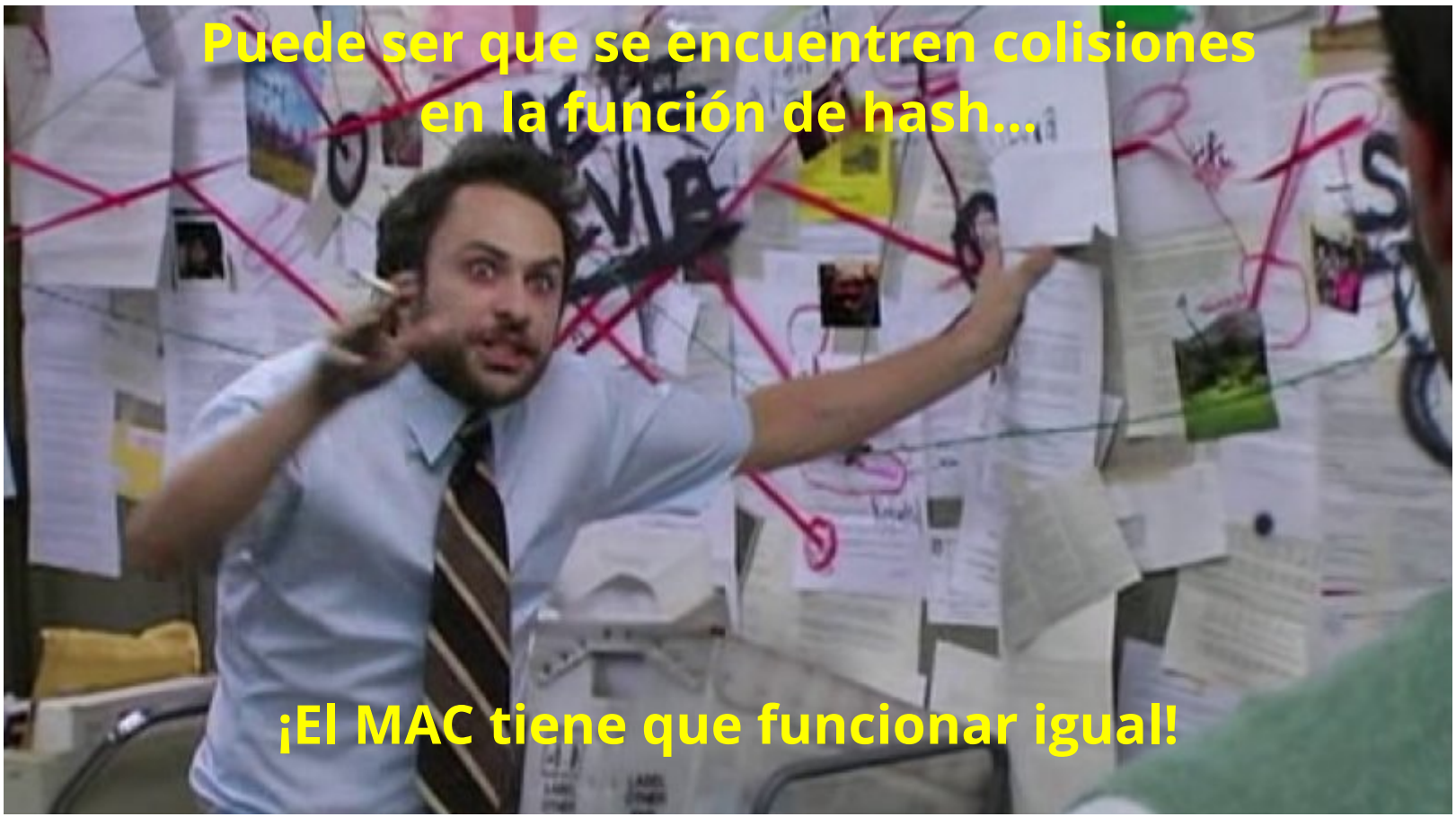
Donde $m' = m_0 \cdots m_\ell$

Length extension attacks

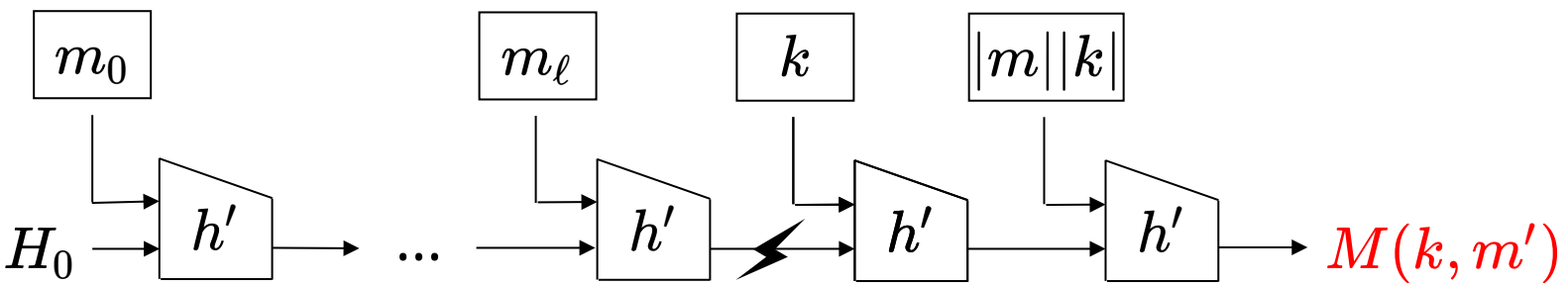
¿Y si tratamos con $M(k, m) = h(m||k)$?

¿Podemos hacer ataques de extensión de largo?

No, pero seamos más paranóicos


Puede ser que se encuentren colisiones
en la función de hash...
¡El MAC tiene que funcionar igual!

$$M(k, m) = h(m||k)$$



Una colisión de dos mensajes del mismo largo implica romper el MAC

¿Qué hacemos entonces?

Podemos tratar varias cosas, pero lo que podemos *demostrar* que es seguro es

$$MAC(k, m) = h(k_1 \mid\mid h(k_2 \mid\mid m))$$

Donde k_1 y k_2 ocupan exactamente un bloque, son distintas, se derivan de forma determinista a partir de k y no se pueden obtener sin k

Si la función de hash es una PRF, MAC es una PRF

El estándar se define de la siguiente forma

$$k' = \begin{cases} h(k) & k \text{ usa más de un bloque} \\ k & \text{e.o.c.} \end{cases}$$

$$k_1 = k' \oplus \texttt{5c} \cdots \texttt{5c} \qquad k_2 = k' \oplus \texttt{36} \cdots \texttt{36}$$

`5c = 01011100`

`36 = 00110110`

$$HMAC(k, m) = h(k_1 \,||\, h(k_2 \,||\, m))$$

No se conocen ataques prácticos para HMAC-MD5

KDF

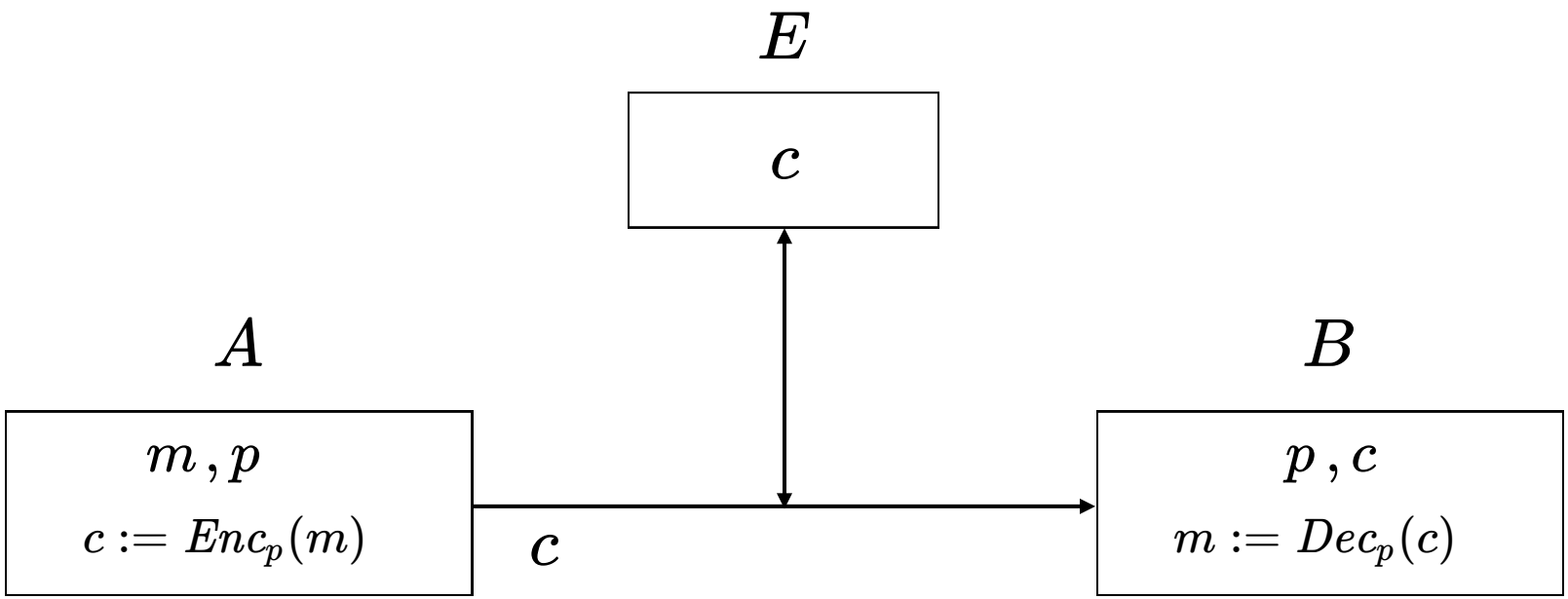
Key-derivation functions

Supongamos que queremos ponernos de acuerdo en una llave para AES-128-CBC y memorizarla

¿Problemas?

Es probable que usemos una mala llave

Cifrado



E podría tener una lista con los 1.000.000 passwords más comunes $\{p_1, \ldots, p_{1000000}\}$

¿Se ve $Dec_{p_i}(c)$ como un mensaje?

¿Y si usamos el SHA256 del password?

$$c := Enc_{h(p)}(m)$$

p_1	$h(p_1)$
p_2	$h(p_2)$
$\vdots$	$\vdots$
$p_{1000000}$	$h(p_{1000000})$

Rainbow tables

¿Está $Dec_{h(p_i)}(c)$ en el formato esperado?

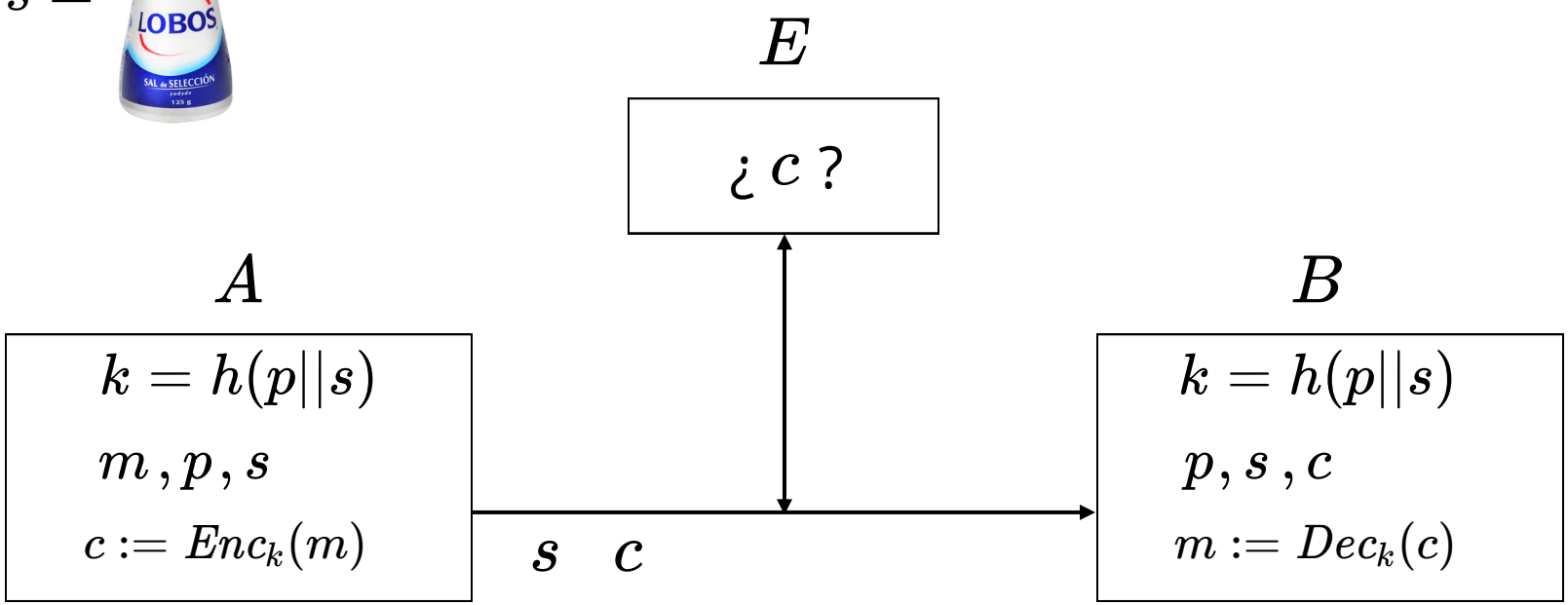
El atacante podría tener esta tabla pre-calculada

¿Cómo podríamos mejorarlo?

Antes de mandar el primer mensaje encriptado, usemos un valor aleatorio

Cifrado

$s =$ 



p_1	$h(p_1 \|\| s)$
p_2	$h(p_2 \|\| s)$
$\vdots$	$\vdots$
$p_{1000000}$	$h(p_{1000000} \|\| s)$

¿Podría el atacante tener esta tabla precalculada?

No conoce s

¿Cómo mejoramos la situación un poco más?

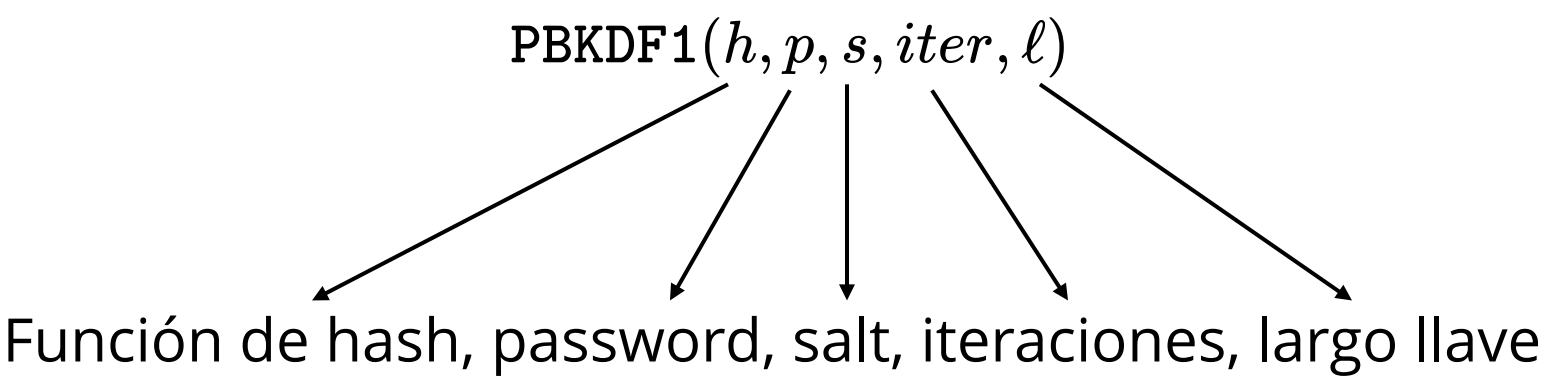
Hagamos que al atacante le
salga caro generar la tabla

En vez de usar h usemos una función que no cueste tanto calcular una vez, pero que sea caro calcular 1.000.000 veces

Una *key derivation function*

PBKDF(1)

Password-based key-derivation function (1)



$\texttt{PBKDF1}(h, p, s, iter, \ell)$

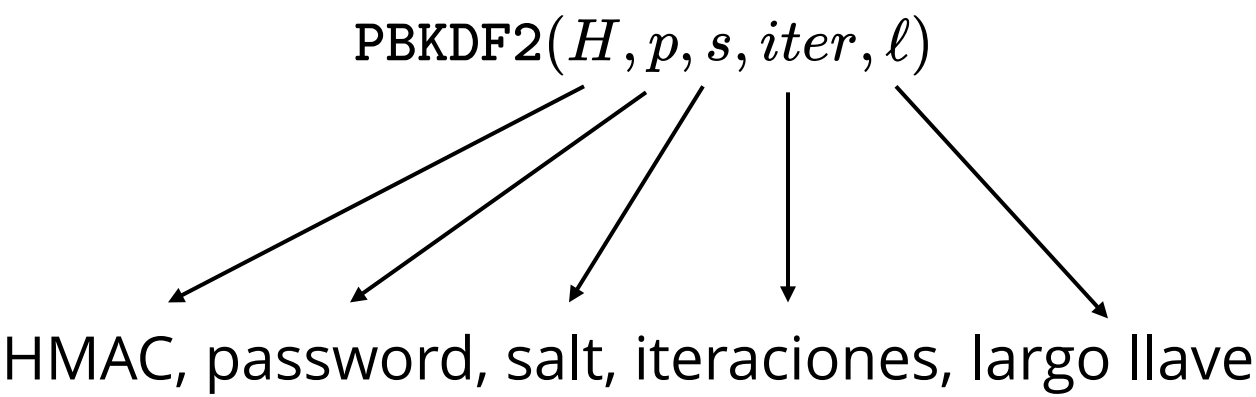
- $U_1 = h(p||s)$
- $U_2 = h(U_1)$
- $\vdots$
- $U_{iter} = h(U_{iter-1})$

El output consiste simplemente en tomar los primeros ℓ bits de U_{iter}

La llave derivada no puede tener largo mayor al output de h

PBKDF2

Password-based key-derivation function 2



$\texttt{PBKDF2}(H, p, s, iter, \ell)$

- $U_1^1 = H(p, s||1)$
- $U_2^1 = H(p, U_1^1)$
- $\vdots$
- $U_{iter}^1 = H(p, U_{iter-1}^1)$

$$T_1 = U_1^1 \oplus U_2^1 \oplus \cdots \oplus U_{iter}^1$$

Si $\ell \leq |T_1|$ entonces el output son los primeros ℓ bits de T_1

De lo contrario calculamos tantos T_i como
sea necesario para obtener al menos ℓ bits

WPA2 usa $\texttt{PBKDF2}(\texttt{HMAC-SHA1}, pass, \texttt{SSID}, 4096, 128)$ para derivar una llave de sesión de 128 bits, que luego es utilizada para encriptar los mensajes entre el cliente y el access point usando AES-128-CCMP

Discusión: ¿qué tan seguro es esto?

Más discusión: ¿Qué tan seguro podría ser esto?


Asymmetric Cryptography
SO HOT RIGHT NOW

To be continued...