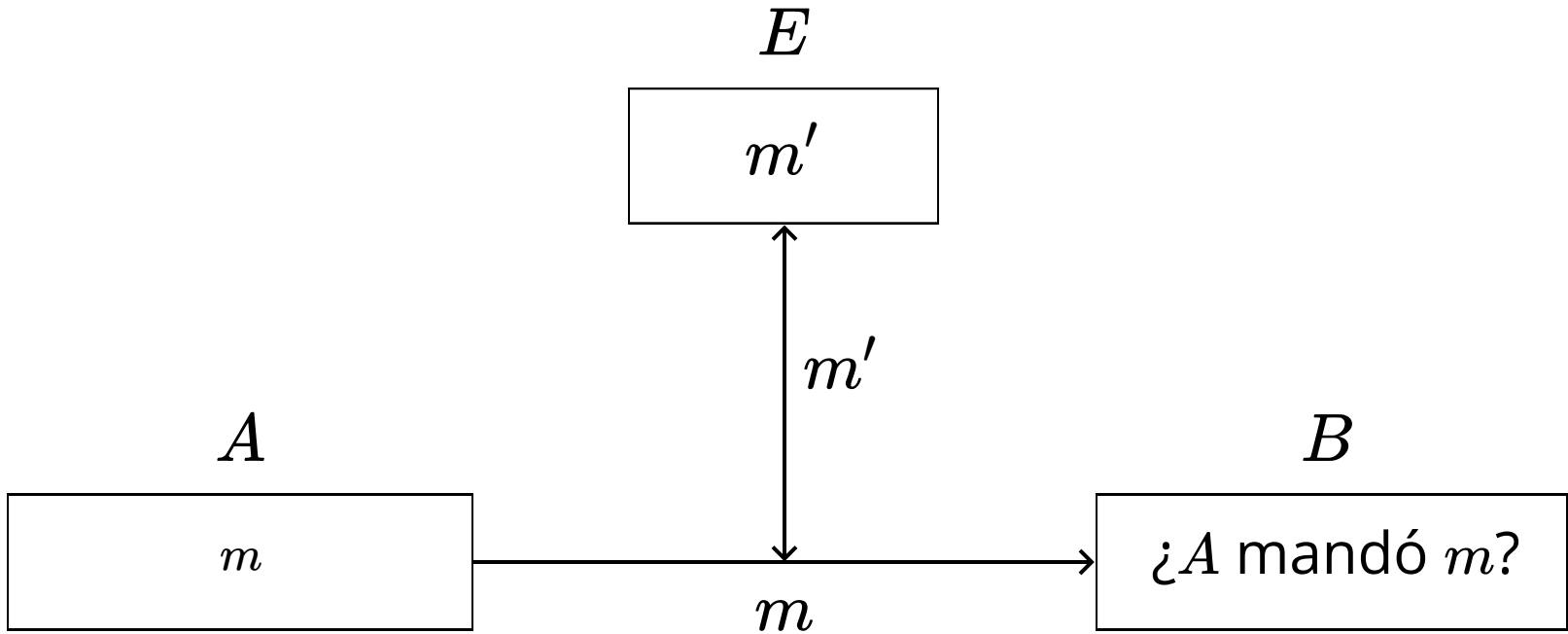


IIC3253

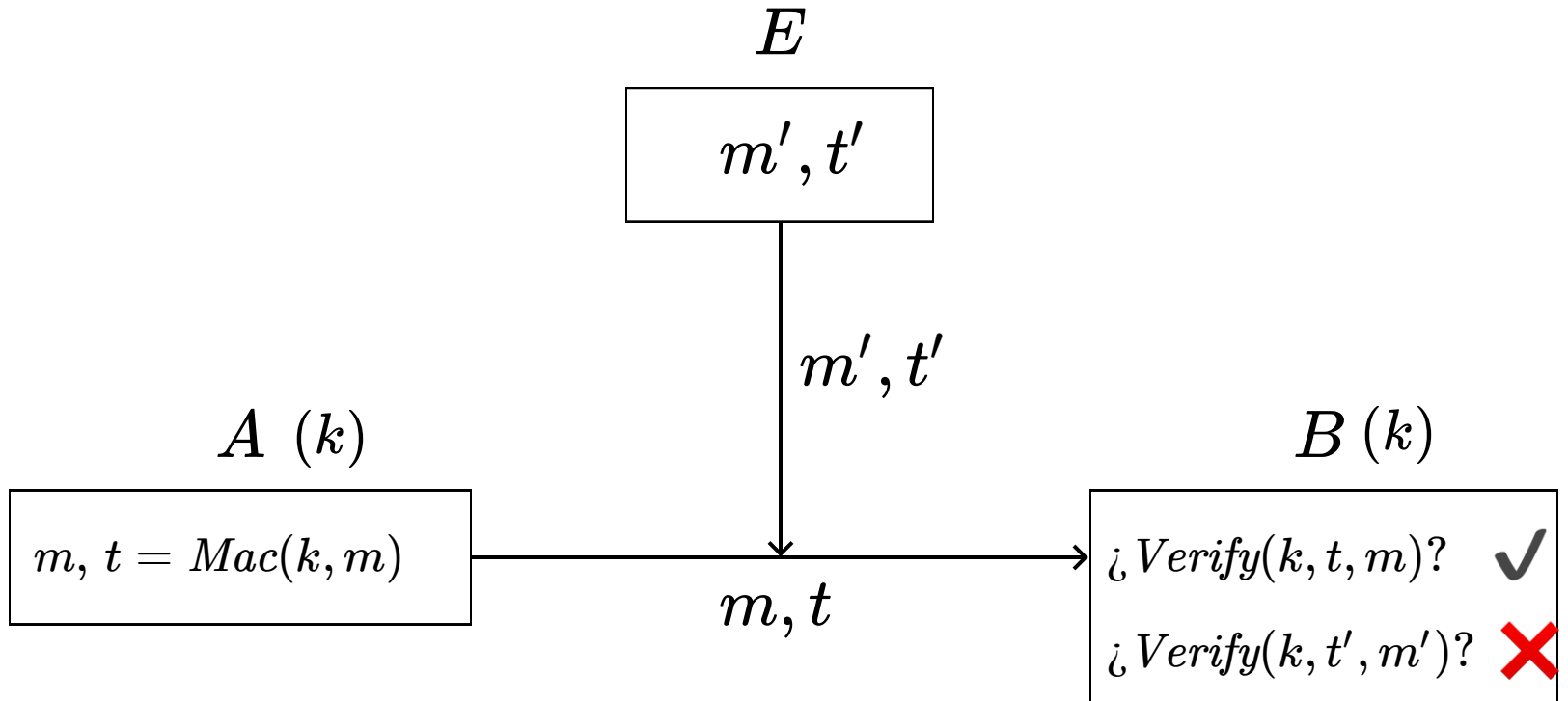
MACs

(≠ Media Access Control, como en MAC address)

Autenticación de mensajes



¿Cómo lo hacemos?



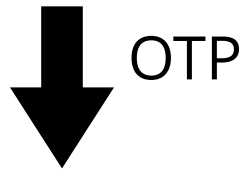
¿Qué relación tiene la encriptación
con la autenticación de mensajes?

¿No basta con encriptar?

"E incluso si tu tienas la llave ni puedes
decriptarlo... ¿cómo se va a ser
capaz de enviar un mensaje encriptado?"

WRONG

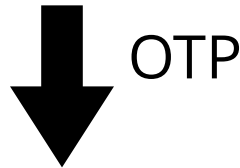
```
1 {  
2   from_account: 239478456,  
3   to_account: 579821324,  
4   amount: 10000,  
5   to_name: "Marcelo Arenas",  
6   to_bank: "BancoMarcelo",  
7   pinPass: 765432  
8 }
```



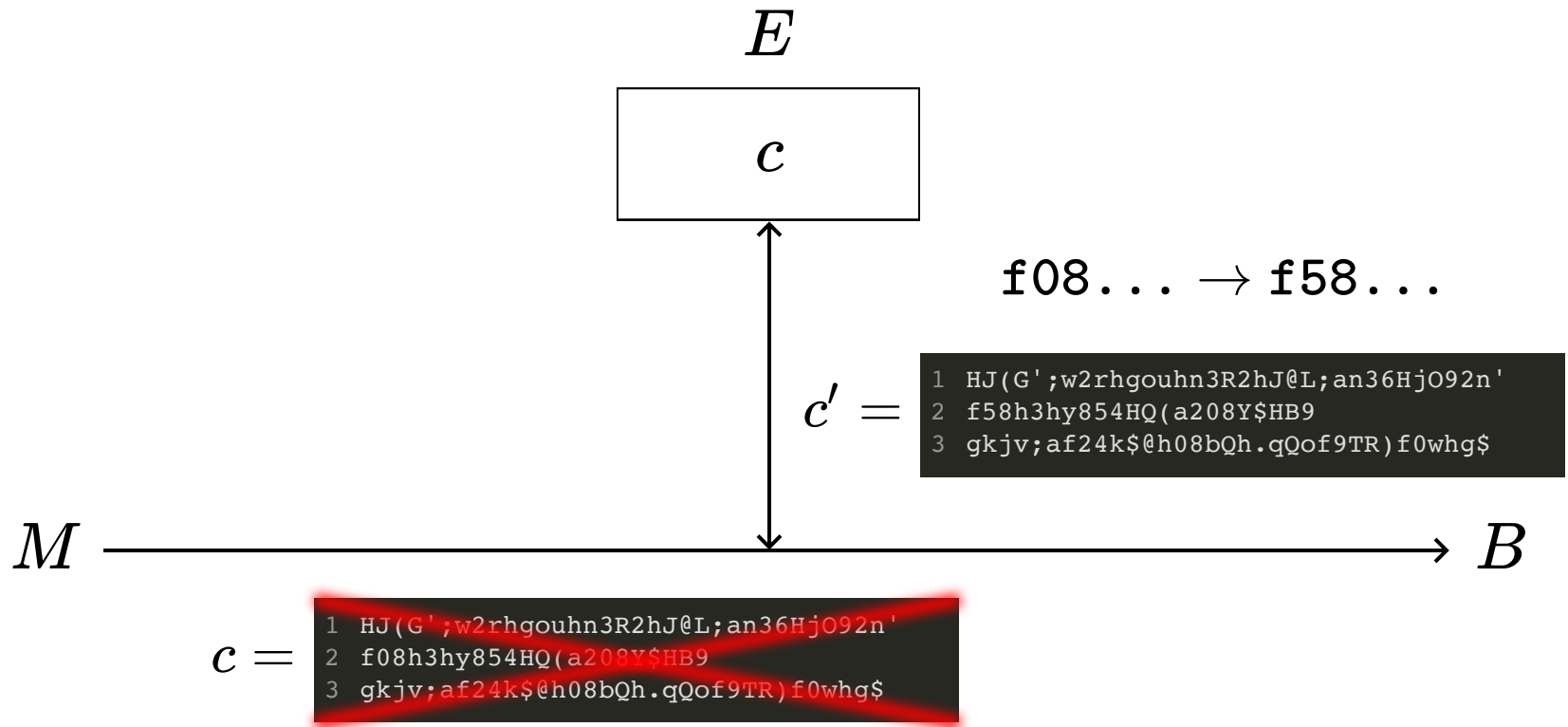
```
1 asdgh09o8hGH3jkG$q325hyghRG  
2 08hyg;no3i5hghwaser  
3 ;'hqr0eihnj5asfdg08hq0[q3rw08gh00qasdfg1kh  
4 4oih,@h9hu0hb
```

El monto está encriptado a partir del caracter número 62...

```
1 {  
2   from_account: 579821324,  
3   to_account: 239478456,  
4   amount: 40000,  
5   to_name: "Martín Ugarte",  
6   to_bank: "BancoMartín",  
7   pinPass: 785613  
8 }
```



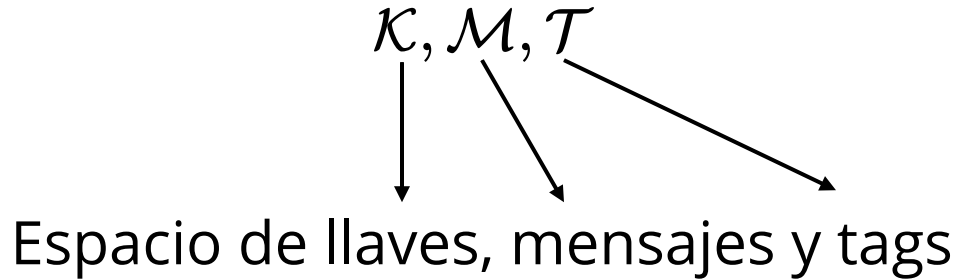
```
1 HJ(G';w2rhgouhn3R2hJ@L;an36HjO92n'  
2 f08h3hy854HQ(a208Y$HB9  
3 gk;v;af24k$@h08bQh.qQof9TR)f0whg$
```

A.K.A. Bit flip attack

A menos que un esquema criptográfico
esté diseñado para ser autenticado,
jamás debemos suponer que lo es.

Formalizando...



Un esquema de autenticación de mensajes es una tupla $(Gen, Mac, Verify)$ tal que:

Gen es un algoritmo aleatorizado tal que, dado 1^n , genera una llave en \mathcal{K} de largo mayor o igual a n

$Mac : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$ es un algoritmo para generar tags

$Verify : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \{0, 1\}$ es un algoritmo para verificar tags

¿Qué esperamos?

Para todo $k \in \mathcal{K}$ y $m \in \mathcal{M}$, se cumple que

$$\text{Verify}(k, \text{Mac}(k, m), m) = 1$$

y cada $m \in \mathcal{M}$ con $m' \neq m$ tenemos

$$\text{Verify}(k, \text{Mac}(k, m'), m) = 0$$

WRONG

?

A jugar...

$(Gen, Mac, Verify)$

Definimos el juego $Forge_{MAC}(n)$:

1. El verificador invoca $Gen(1^n)$ para generar k
2. El adversario envía $m_0 \in \mathcal{M}$
3. El verificador responde $Mac(k, m_0)$
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 había enviado antes

¿Cuándo gana el adversario?

$$Verify(k, t, m) = 1$$

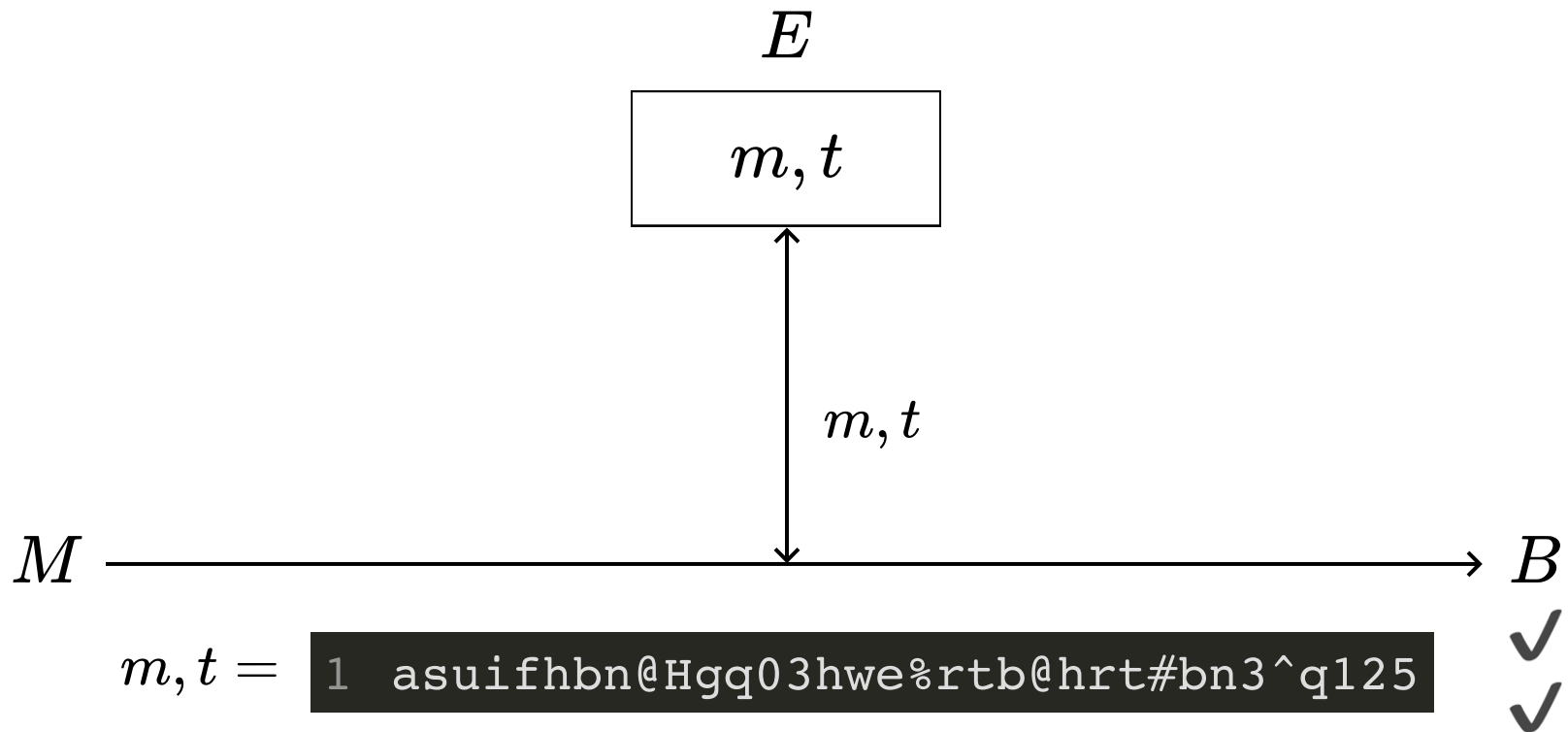
¿Cuándo decimos que MAC es un esquema de autenticación de mensajes **seguro**?

Todo adversario que juega en tiempo polinomial (en n) tiene una probabilidad despreciable (en n) de ganar el juego

En general se usa *seguro* = *unforgeable*

¿Algún ataque más?

```
1 {  
2   from_account: 579821324,  
3   to_account: 239478456,  
4   amount: 40000,  
5   to_name: "Martín Ugarte",  
6   to_bank: "BancoMartín",  
7   pinPass: 785613  
8 }
```

A.K.A. replay attack

Un esquema de autenticación de mensajes **no tiene por qué** mitigar este tipo de ataques. En general esto se maneja a nivel de aplicación.

¿Podemos construir un MAC en base a una función de hash?

$$¿Mac(k, m) = h(k||m)?$$

¿Dado $h(k||m)$, estamos seguros de que nadie puede calcular $h(k||m||m')$?

Ataques de extensión