Background: stream ciphers

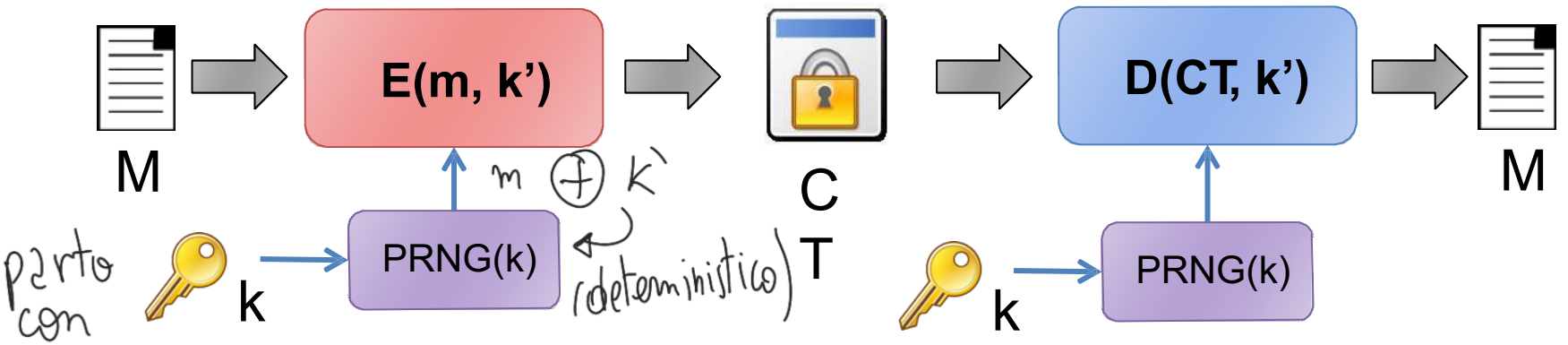
In realtà, che ciphers abbiamo ?

Practical ciphers

➔Two major categories

- ⇨STREAM ciphers
  - ➔Traditional (weak): RC4
  - ➔Modern (strong): Salsa20, ChaCha20
- ⇨BLOCK ciphers
  - ➔Traditional (weak): DES, 3DES
  - ➔Modern (strong): AES
- ⇨Plus BLOCK ciphers used in STREAM mode
  - ➔AES-CTR counter mode

Stream ciphers



➔**Goal: "approximate" a one-time-pad**

Non potendo avere una chiave truly random (deve essere simmetrica), la faccio pseudo random, più "debole".

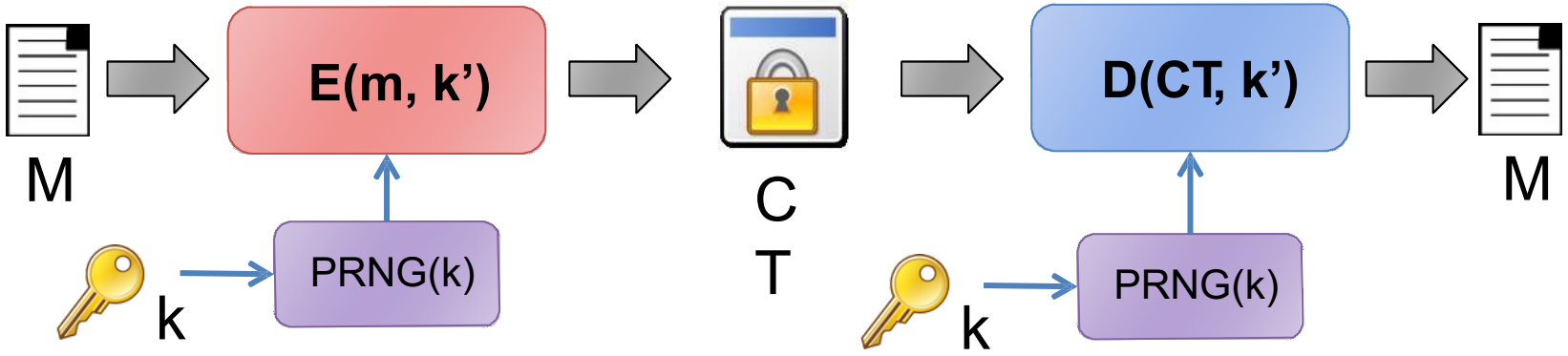
➔**Crucial difference:**

- ⇨One Time Pad → random key
- ⇨Stream cipher → random keystream

➔**DO NOT confuse key K with keystream k'!!!!**

Graphics courtesy of Marco Spaziani Brunella

Stream ciphers (basati su XOR)



➔**ENC and DEC based on XOR (exactly as OTP)**

➔CT = ENC(K, M) = M ⊕ keystream = M ⊕ PRNG(K)

➔M = DEC(K, CT) = CT ⊕ keystream = CT ⊕ PRNG(K)

➔**If substring repeats, ciphertext changes (good!)**

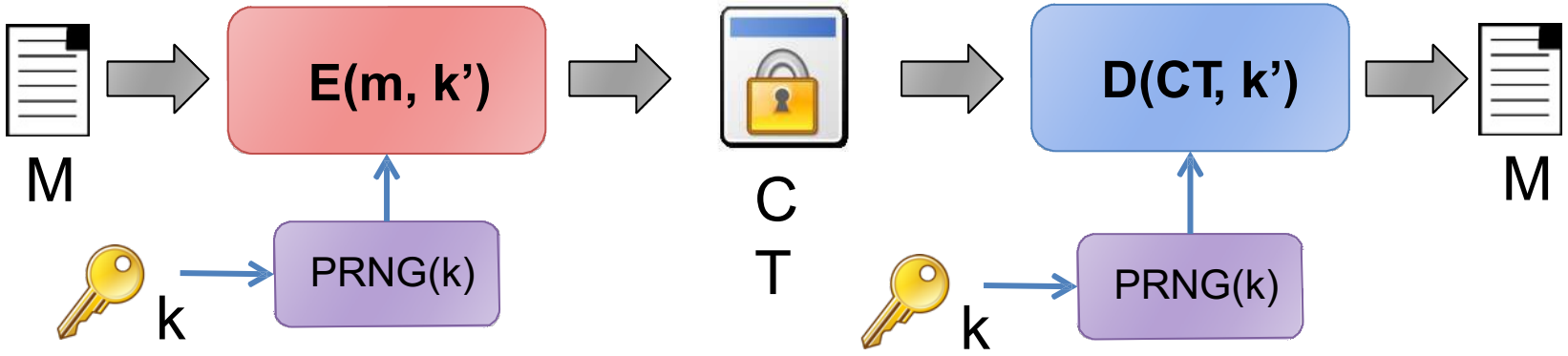
➔

C	I	A	O	C	I	A	O	C	I	A	O	⊕
1	3	4	f	2	5	9	5	8	d	d	1	=
α	β	η	δ	φ	ϕ	τ	κ	σ	δ	λ	λ	

ciaociaociao e' unico msg !

Graphics courtesy of Marco Spaziani Brunella

Stream ciphers



➔ **Still, if message is encrypted twice, ciphertext will repeat!**

⇨ PRNG is deterministic! same input (key K), same output (keystream K')!

➔ **RC4 example:**

⇨ Encrypt "giuseppebianchi" → CT = giuseppebianchi ⊕ RC4(key)

⇨ Key = 12345678790 → CT = 474d4caf78a844afa8b29add814e86

⇨ Key = 12345678791 → CT = d291ee1272acdb9e9e982986dfb4f8 - different key: ☺

⇨ Key = 12345678790 → CT = 474d4caf78a844afa8b29add814e86 - same key: ☹

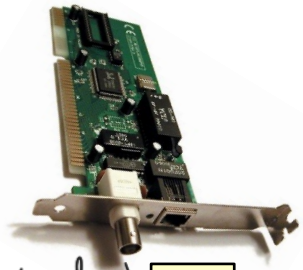
➔ **Problem: with static key, IND-CPA breaks!**

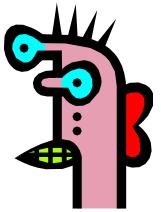
⇨ Question: how to guarantee semantic security?

Graphics courtesy of Marco Spaziani Brunella

Come faccio un cipher che usa stessa chiave che NON si ripete?

Initialization Vectors in stream ciphers





- **Initialization Vector IV → fresh PRNG seed at every new msg**
- **IV (usually) transmitted in clear, along with ciphertext**
- **RX combines it with long term key → reconstruct keystream**

Provable semantic security… but only if IV will NEVER REPEAT!!

(IV può essere RANDOM/SEQUENZIALE qui, ma non ripetibile)
NOT predictable predictable

KPA e CPA, qui non scelgo cosa cifrare, ∄ oracolo.
IMPORTANTE: stesso msg, cifrato 2 volte: 2 output diversi! requisito base!

Main thesis of this (entire!) class:
Good crypto can be badly used

Most breaches exploit poor protocol constructions and/or vulnerabilities

Warm-up example 2
802.11 WEP

Preamble: a MUST-KNOW fact of life…

The incompetence of unskilled people
metacognitive ability to
y are incompetent
→

being ignorant
ture of ignorance
l world examples!
ol, work, working groups,…

Dunning-Kruger Effect

Wish I were here..☺

ho are unskilled in these
ıs conclusions and make
ility to realize it. Across 4
ts of humor, grammar, and
test scores put them in the
linked this miscalibration
from error. Paradoxically,
competence, helped them

'96 - '99 : WiFi include buggatissimo (viene rivisto)

WEP lessons

Wired Equivalent Privacy

USA → RC4 (stream cipher) e good

➔ **Good cipher is far from being enough**

- ⇨ You must make good USAGE of cipher
- ⇨ And you must design GOOD protocols

(sbagliare IV rende tutto vano)

➔ **WEP goals**

- ⇨ Authentication — Ridicolous! Facilitates attack!
- ⇨ Integrity — Inconsistent!
- ⇨ Confidentiality — Broken!

} FLOP!

➔ **WEP: best "real world" example of how to make things incredibly (!) wrong**

- ⇨ And well «besides» its weak cipher (RC4)

WEP cipher: RC4

➔**RC4: a specific PRNG algorithm**

keystream

⇨used to generate the keystream $RC4(K, IV) = 0101101\ldots$

⇨Today weak, but still considered to be strong at that time

⇨**Important WEP lesson:**
WEP confidentiality broken even if RC4 were perfect!
➔RC4 weaknesses "just" make the situation worse…

➔**RC4 encryption:**

⇨ENC(KEY,MSG) = MSG ⊕ RC4(IV, KEY)

⇨Keystream = RC4(IV, KEY)
intuitively consider this as PRNG(IV, Key)

In WIFI pkt perdibili, se lo avevo criptato perdo parte di informazione. Invece di 1 keystream unico per tutti i pkt, ne faccio 1 ∀ pkt+iv (nuovo ∀ pkt). Nell'unico keystream avevo unico IV.

Wep and IV

➔WEP: Per-frame IV (a must in WLAN)

- ⇨Stream cipher must be synchronized
  - ➔BIG problem in lossy channels
- ⇨WEP solution: send IV per each frame
  - ➔Can independently decrypt each frame irrespective of previously received/missed ones

➔WEP IV: transmitted in clear

- ⇨Not a problem in any good stream cipher: even if IV known, this should **not** break security
  - ➔Security should ONLY require secrecy of the key
  - ➔Keystream = PRNG(IV,Key); if IV known key remains unknown

WEP and IV

➔ **If we assume RC4 strong, then WEP secure as long as IV never repeats**

➔ **Indeed, same keystream iff same (Key, IV)**

- ⇨ $\{ks_i\}$ = PRNG(IV,Key)
- ⇨ If IV repeats, no more semantic security
  - → $(M1 \oplus ks_i) \oplus (M2 \oplus ks_i) = M1 \oplus M2$
  - → If one message known, other known

➔ **If IV repeats, the weakness is PRACTICAL!!**

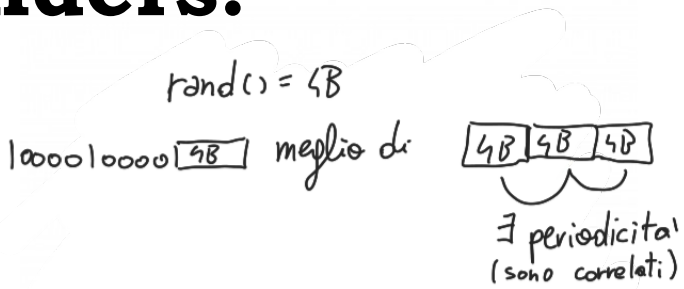
- ⇨ Easy to create a KPA or CPA condition in wifi
- ⇨ Even without KPA, Limited message entropy → helps to infer
- ⇨ "context" information → helps to infer

And since IV is soooooooo important....

➔WEP incredible blunders:

(3 bytes)

1. IV size = 24 bits
   (poco overhead), 2^{24}



2. IV generation:
   left to the implementation
   [never, NEVER, do this in security protocols!!]

bisogna specificare, non lasciare ad altri.

Repeating IV in WEP

(key statica come sempre)

SVOLGIMENTO:
throughput è 7*10^6 [bit/s] =
(7*10^6)/8 [bytes/s] =
875000 [bytes/s]
allora in un minuto moltiplico per 60 secondi, e in un'ora di nuovo per 60, moltiplico quindi per 3600 =
3,15*10^9 [bytes/h]
allora faccio (16777216 frames * 1500) [bytes] diviso 3,15*10^9 [bytes/h], ed ottengo 7.98 ore

➔ **Cyclic generation {1,2,3,…}?**

- ⇨ 24 bit → 2^{24} cycle = 16.777.216 frames
- ⇨ Assume:1500 bytes frames, ~7 mbps (net) wifi throughput
- ⇨ IVs re-cycle after less than 8h!

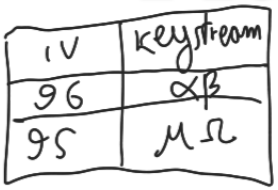
➔ **Random IV generation?**

- ⇨ Birthday paradox!
- ⇨ 50% collision probability after approx 2^{12} = 4000 frames!

➔ **Restart after reboot?**

- ⇨ In many cases, IV re-start from 0! Same IV sequence!!

Practical attacks to small IV space



➔**Passive attack: create dictionary**

IV → RC4(IV,Key)

⇨How to? Use known messages to recovery keystream

(MSG ⊕ RC4(IV,key)) ⊕ MSG → RC4(IV,key) QUA NON SI PARLA DI IV XOR KEY

⇨Known messages? (constant static IV)

→Send an email with large known attachment ☺ CPA

→Use…. AUTHENTICATION PROCEDURE!!

» **See next 4-5 slides!!**

➔**Wait for IV to repeat: game over**

(M' ⊕ RC4(IV,key)) ⊕ RC4(IV,key) = M'

➔**Larger WEP key size (e.g. 128)?**

⇨Irrelevant: IV still 24 bits!! Same attack time!!

non dipende da Wep Key, ma da IV!

contesto: [IV | cipher MSG], IV usato per keystream PRNG. Nel modello CPA, mando MSG noto e lo ricevo cifrato. Invio 'M', ricevo 'C', 'IV' in chiaro → C⊕M=keystream poiché C = M⊕keystream. keystream = RC4(K, IV). Secondo uno studio, ∃ 5% di prob. che ho KS rilasci info sulla chiave K. Non è perfetto!

➔ Cryptoanalytic attack to RC4

➔ Fluhrer, Mantin, Shamir: show weakness

- →Some "weak" IVs leak key information into keystream
  - » With 5% probability, a byte in the keystream is equal to a byte in the key
  - » Look at many packets and see where bias is

➔ Stubblefield, Ioannidis, Rubin use weakness

- →Focus on first keystream byte
- →BUT this is even known for ALL packets
  - » **Remember: 802.11 uses SNAP LLC encapsulation: first byte = 0xAA**
  - » **Known plaintext!** (Ho campi dei pkt noti a priori)
- →Attack linear (!) with key size

User authentication

➔Prove that you are really the one you claim to be

- ⇨Do not confuse:
  - ➔**Identification ➔ show** your «digital identity» (email, ID)
  - ➔**Authentication ➔ prove** your digital identity is controlled by you (e.g. you know a secret password)

➔Actually, not so simple…

- ⇨There might be no requirement to link identifier to a real-life identity
  - ➔Successful authentication = I'm reasonably assured that **the subject accessing the service today is the same as the one which accessed the service in the past**.
- ⇨NIST SP 800-63-3 definition for digital user authentication: the process of establishing confidence in user identities that are presented electronically to an information system

Authentication Means

- **Something you (and only you) know**
  - Password, PIN, secret key, answer to a pre-arranged question, …
- **Something you (and only you) have**
  - Smart card, Physical devices (tokens)
    - Very interesting area: "authentication" based on HW uniqueness → PUF (Physically Unclonable Function)
- **Something you are (static biometrics)**
  - retina, fingerprint, face, …
    - Privacy? And what about DNA (Remember Gattaca?)?
- **Something you do (dynamic biometrics a.k.a. behavioral authentication)**
  - Voice recognition, hand writing, typing characteristics
    - … even Mouse Movements!!

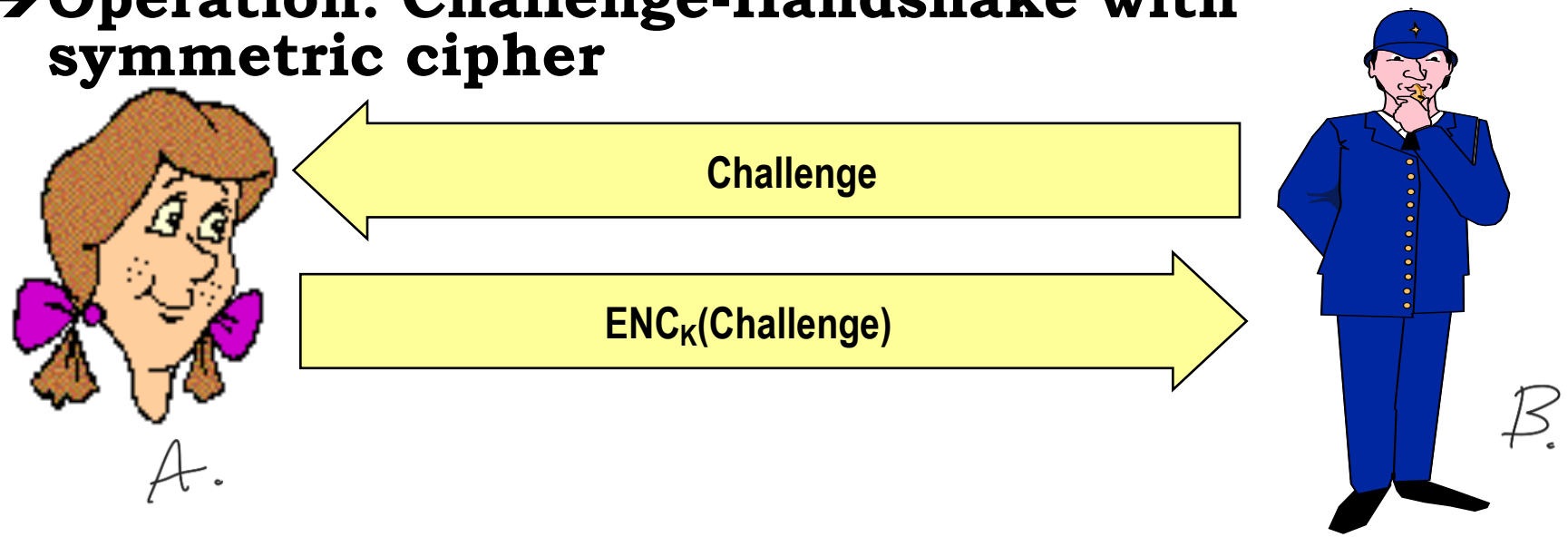
But there are also other specific means harder to classify, e.g. location-based auth

WEP authentication

- **Grant network access only to users who KNOW a pre-shared secret**
  - A DIFFERENT security goal (service) than confidentiality!
- **How to "prove" I know a secret?**
  - By revealing it while accessing the network?
    - Nah…
      - » anyone else would overhear it
      - » And would use it later on (replay attack)
  - Perform some operation which REQUIRES knowledge of the secred
    - **But whose result does NOT reveal it!**

WEP idea:

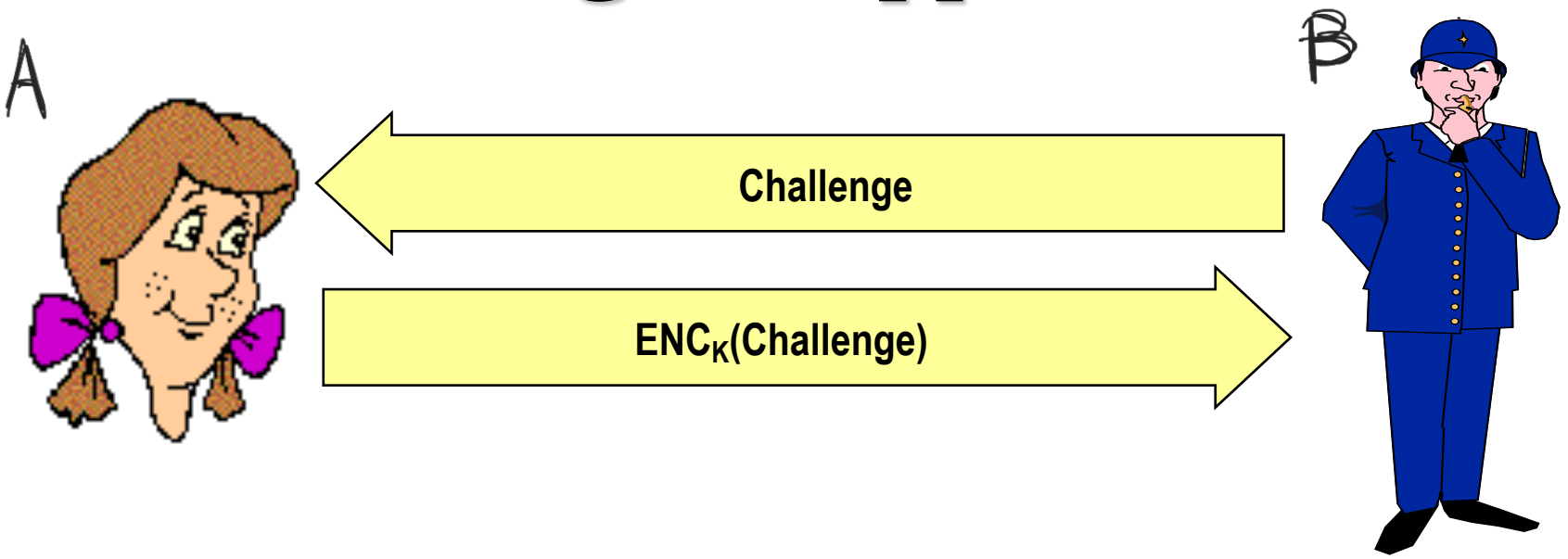
- **Secret: use the SAME of encryption**
  - ⇨ Minimize «secrets» to handle
    - → In principle might even make sense…
      - » But you MUST know how to properly handle this more in a month from now!
- **Operation: Challenge-Handshake with symmetric cipher**



- **Prove knowledge of K by showing ability to encrypt the challenge**

Se B. decripta il msg, ed è uguale a quella originale, allora OK.

It is a good approach?



➔ **Let's check the Menezes book:**

- ⇨ Yes, chapter 10.3.2 mentions this
- ⇨ As long as challenge random and never repeats, I'm OK then

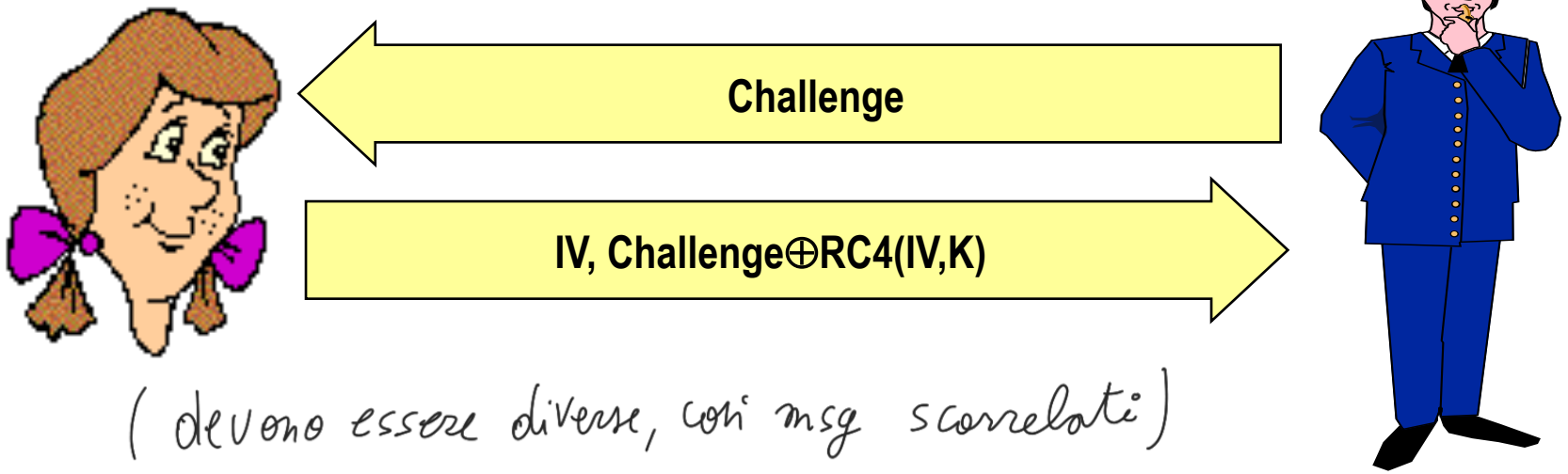
➔ **SURE???**

Menezes dice che è OK, ma bisogna implementarlo bene!

Why WEP "authentication" helps?

con chiavi diverse potrei autenticarmi, ma non avrei ancora la chiave di encryption!

➔**WEP details**



(devono essere diverse, con msg scorrelati)

➔**Same key as frame encryption (!)**

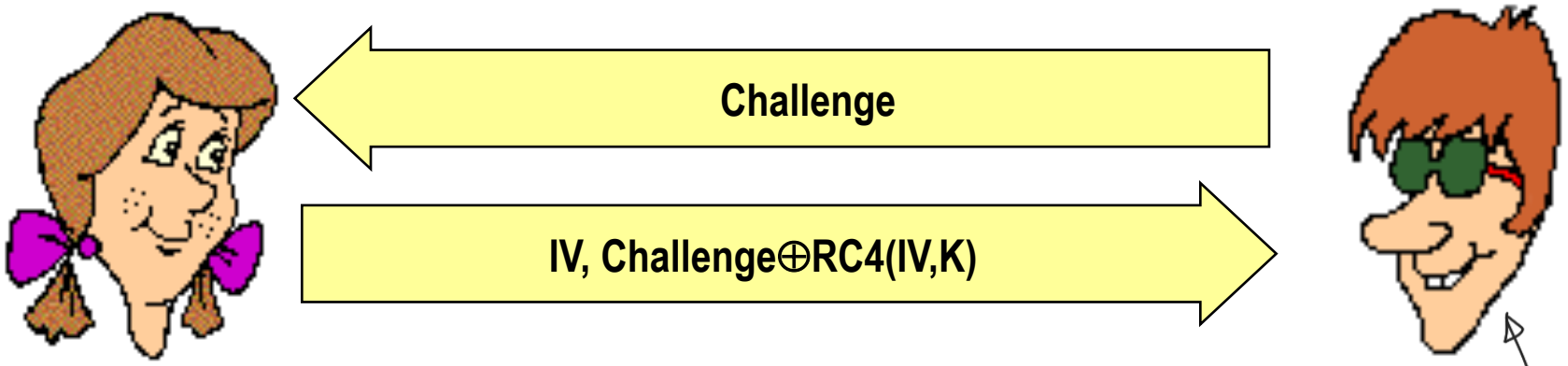
➔**Challenge = plaintext = known**

➔**Isn't this a Known Plaintext Attack??!!!!!**

Why WEP "authentication" helps?

➔Rogue AP: send multiple challenges

⇨Gather multiple [IV, RC4(IV,K)] pairs!!



⇨When enough, go and enjoy decrypting anyone else in the BSS!

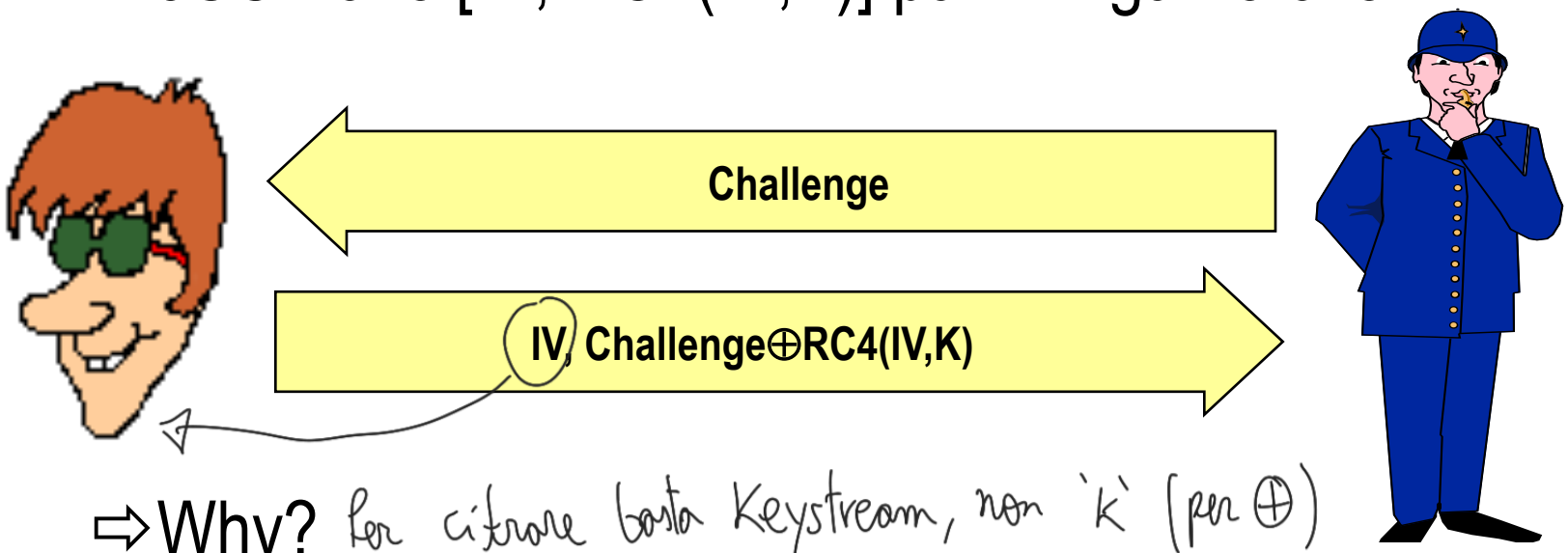
ha visionato tutto il traffico passato, per avere info!

But at least is WEP "authentication" robust??

➔**No, its broken as well!!**
No need to know key to enter network!

Nello challenge il client non deve scegliere nulla.

⇨JUST one [IV, RC4(IV,K)] pair → game over!



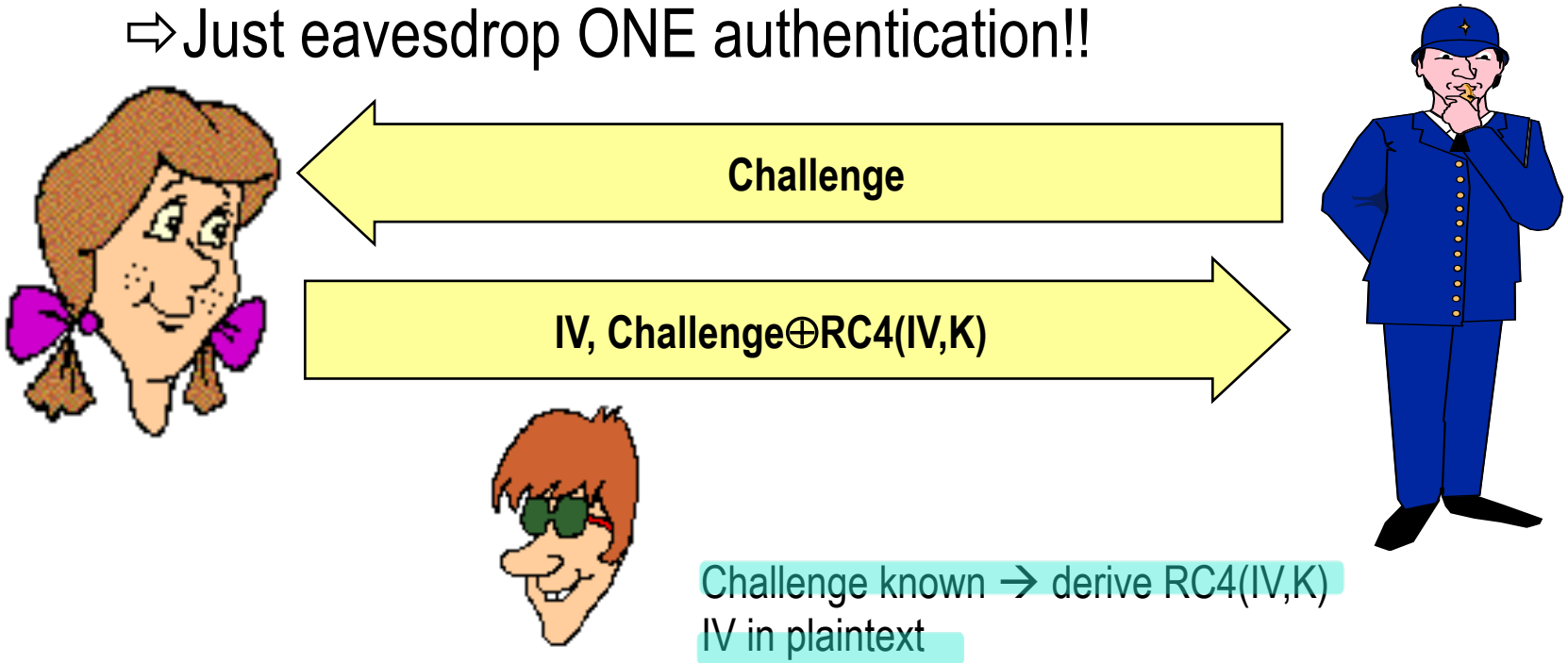
⇨Why? per cifrare basta Keystream, non 'K' (per ⊕)

⇨Because… IV chosen by… responder!!!

Why WEP "authentication" helps?

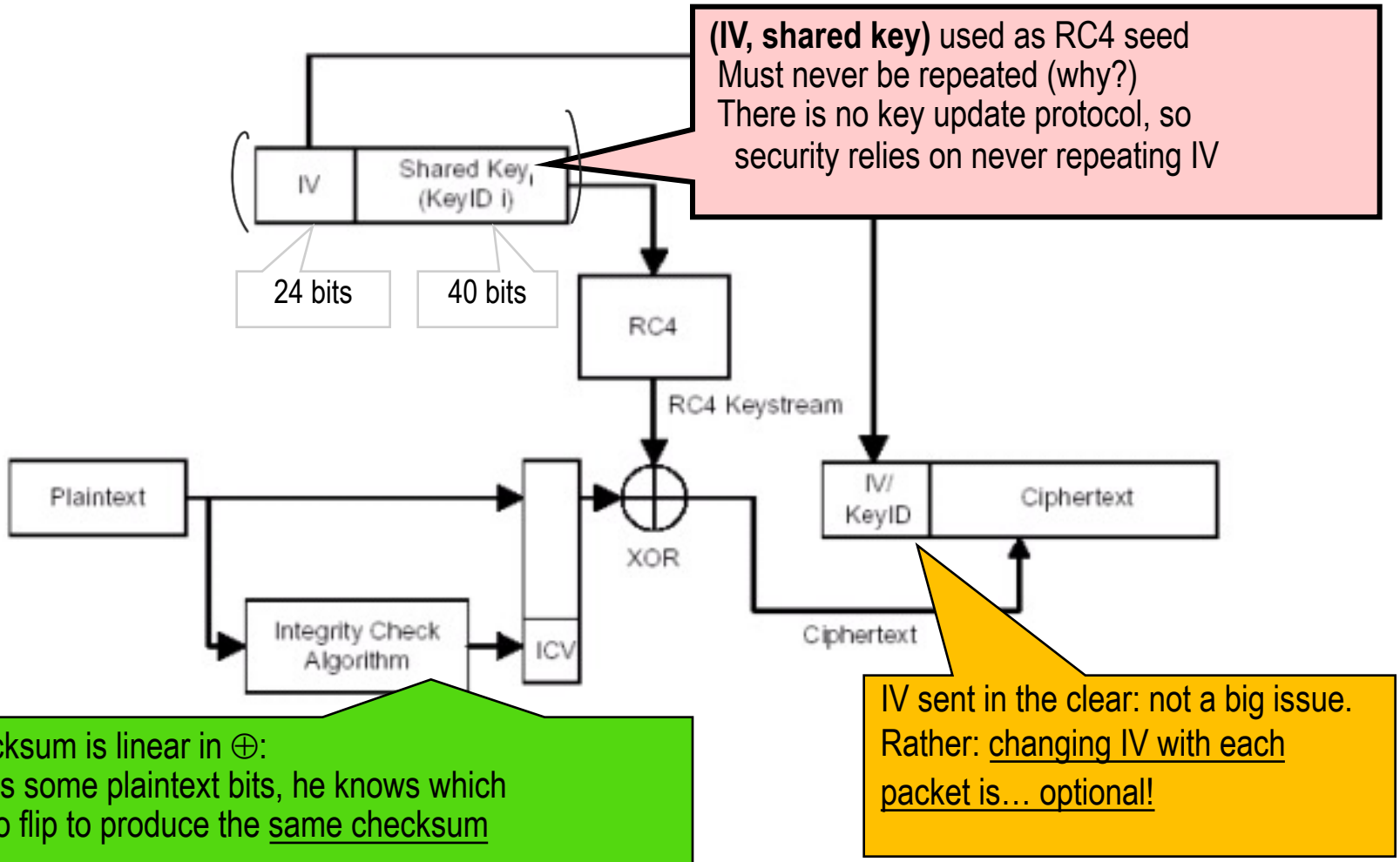
➔**And if no IV available and no active attack possible?**

⇨Just eavesdrop ONE authentication!!



Challenge known → derive RC4(IV,K)
IV in plaintext
One pair ready to be used for YOUR authentication!

WEP – summary (so far)

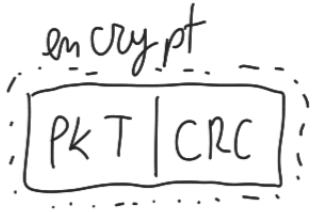


no integrity!

Source: V. Shmatikov

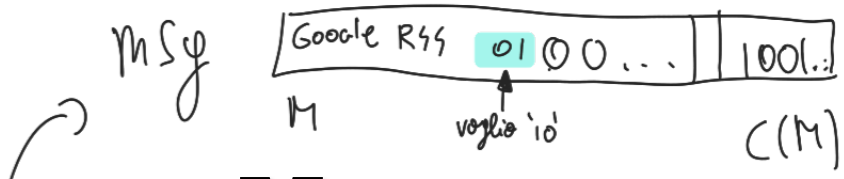
(intermediario NON deve modificare il msg)

What about integrity?



, tutto cifrato, ma CRC e' lineare!

- **"Terrific" idea: use native CRC-32 as integrity check value**
  - External encryption wrapping should prevent (!) attacker to perform valid alterations
- **Problem: CRC-32 is linear!**
  - $c(A) \oplus c(B) = c(A \oplus B)$
  - **Deadly**: use only cryptographically strong algorithms
- **Consequences:**
  - (Meaningful!) Message modification
  - Message injection



Message modification

IV; C = [M|C(M)] ⊕ RC4(IV,K)

δ = 00000 ... 11 .... poiché δ⊕M cambia quei 2 bit, faccio anche CRC(δ)

Chooses δ as long as M

Computes c(δ) ORIGINAL

Computes:

$$C' = C \oplus \{\delta, c(\delta)\} =$$

$$= [RC4(IV,K) \oplus \{M, c(M)\}] \oplus \{\delta, c(\delta)\} =$$

$$= RC4(IV,K) \oplus \{M \oplus \delta, c(M) \oplus c(\delta)\} =$$

$$= RC4(IV,K) \oplus \{M', c(M \oplus \delta)\} = \quad \text{lineare!}$$

$$= RC4(IV,K) \oplus \{M', c(M')\} \quad \text{coerente}$$

M' = M ⊕ δ

C' is an encrypted message with valid CRC32!! Fooled!

NO NEED to know M

δ can be chosen so as to flip specific bits

L'unico alg. integrità + confidenzialità e' AEAD

Authenticated encryption associated data

AES-GCM, AES-CBC, AES-CTR sono alcuni esempi.

Message injection

Obtain one pair IV, RC4(IV,K)

Goal: inject message M so that it is authenticated and encrypted

Steps:
- **Compute c(M)**
- **inject IV, RC4(IV,K) ⊕[M|c(M))**

Take-home messages:

1) integrity check must be NON LINEAR

2) Integrity check must explicitly include a key

external encryption may not provide ANY guarantee

802.11: Aftermath

- **Optimized WEP attacks on many open source**
  - Although just very few WEP protected networks remaining
- **WiFi evolution (802.11i):**
  - WPA: retain hardware compatibility (RC4), but patch with:
    - Longer IV (48 bit)
    - Protection of IV (plaintext differs from RC4 input)
    - Ephemeral key derivation (changing in time)
    - …
- **WPA2: as above but using AES (new hardware)**
- **2017: discovered KRACK attack to WPA2**
  - Note: attack works against a **mathematically proven secure** (!) 4-way handshake… OUCH!
- **2018+ → WPA3 standardized**

NON POTEVANO CAMBIARE TUTTO SUBITO

(reply attack quando mi scollego da un Access Point e mi ricollego ad un altro, stesso dominio)