Review of Block ciphers: Authenticated Encryption

Study reference: Aumasson, chapter 8

Recap: encryption does not guarantee integrity!

nel CPA attack encrypto mio msg all'oracolo e vedo msg nel canale

➔**Confidentiality: semantic security against CPA**

- ⇨Encryption secure against eavesdropping only
- ⇨NOT secure against active attackers (see next 5 slides!!)

➔**Integrity: unforgeability under CCA**

- ⇨But messages remain in plaintext

➔**Authenticated Encryption:**

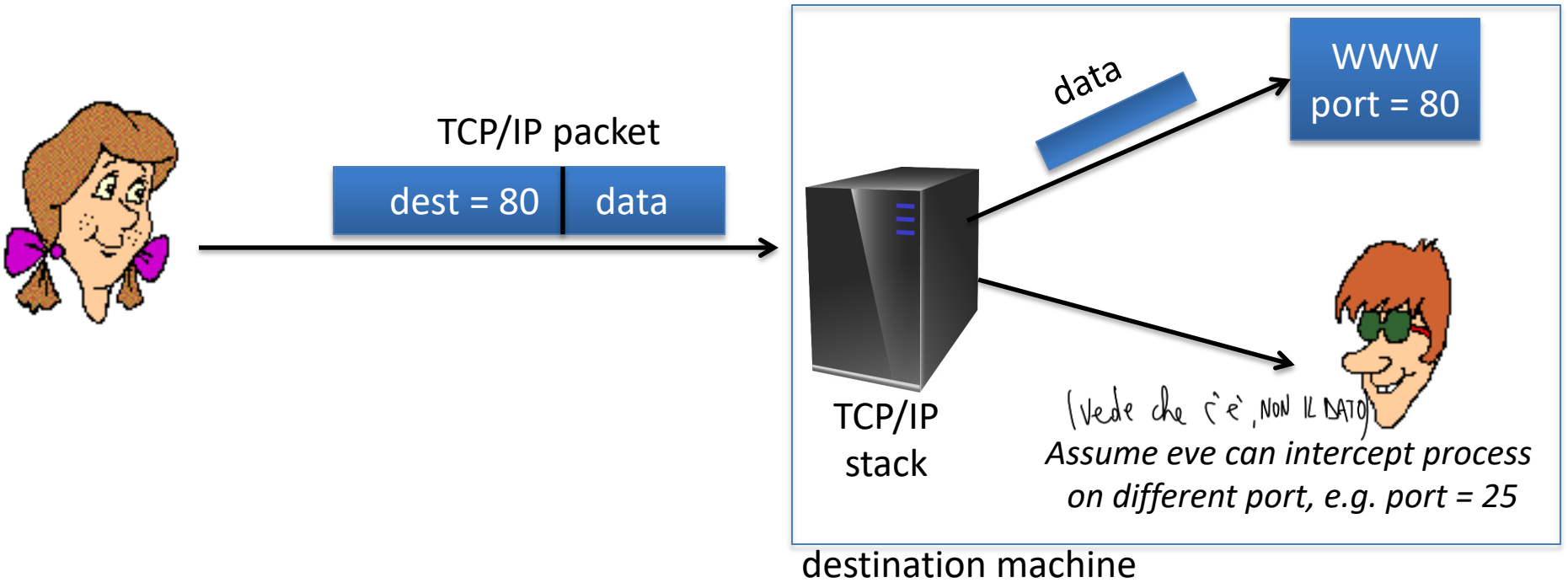
- ⇨Cipher designed to be secure against active attacks (tampering)
- ⇨Ensure both confidentiality and integrity

Indeed: tampering attacks CAN *also* break confidentiality!

➔Two illustrative examples in the next slides

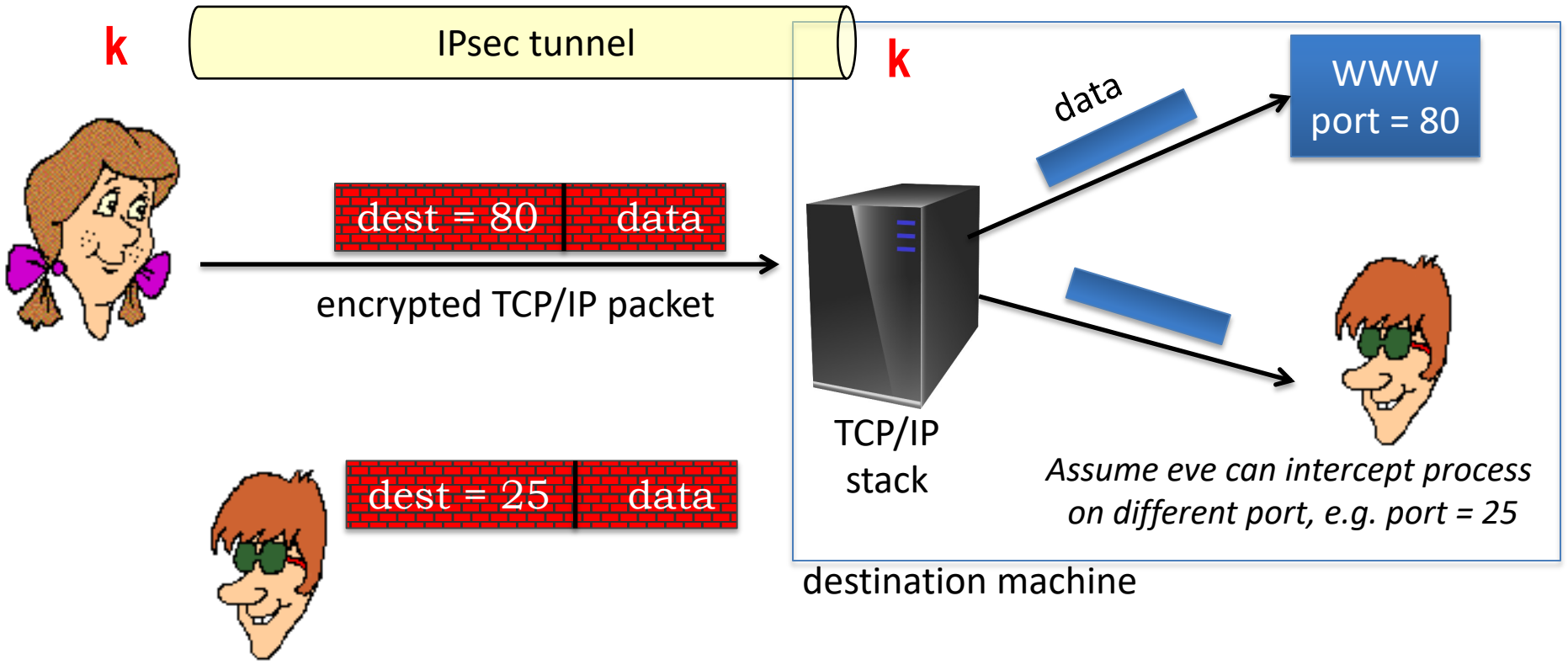
➔(credits to Dan Boneh's class)

Example 1



Souce: Dan Boneh (adapted by me)

Example 1



If attacker could just change port…

Could then read data in clear!!

TRIVIAL, with CBC encryption!!!

AES-CBC
(non XOR diretto)

Souce: Dan Boneh (adapted by me)

Example 1

k

Ipsec tunnel

k

Non protetto, lo cambio, cambierebbe tutto. Voglio ridirezionare il pkt : PRP⁻¹[dest] = port 25

IV' | dest = 80 | data

CBC-encrypted TCP/IP packet

data

WWW port = 80

TCP/IP stack

Assume eve can intercept process on different port, e.g. port = 25

destination machine

IV

dest=80

IV ⊕ dest=80

K→ PRP

dest=80

Test: IV'=??

IV' = IV ⊕ dest=80 ⊕ dest=25

Now the SAME previous ciphertext decrypt to a new plaintext where port=25!!

Souce: Dan Boneh (adapted by me)

I'm not yet convinced!

➔Too strong (?) attacker model!

⇨Attacker has access to the backend

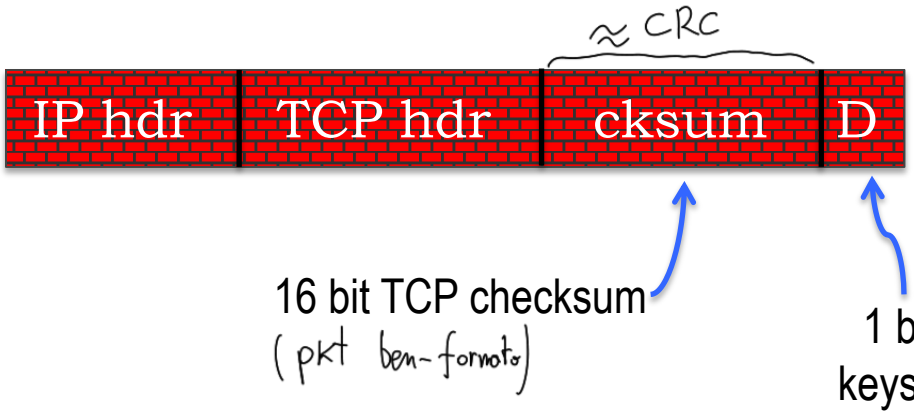
➔And you always use CBC in your examples!

⇨Perhaps CTR is not vulnerable??

See next slide an example of CCA against CTR!

Example 2:
Attacker has only network access;

Remote terminal app: each keystroke encrypted with CTR mode



FACT: TCP/IP stack processes only valid packets

→ TCP will ACK only packets with correct cksum

(let's use this as an oracle!!)

Souce: Dan Boneh (adapted by me)

Example 2:
Attacker has only network access;

≈ stream cipher

Remote terminal app: each keystroke encrypted with CTR mode

IP hdr | TCP hdr | cksum | D

For all t, s: IP hdr | TCP hdr | cksum | D

⊕ t ⊕ s

ACK if valid checksum, nothing otherwise

MANY (!) valid equations: {checksum(hdr, D⊕s) = t ⊕ checksum(hdr, D)}
hdr often known, {cksum, D} unknown → may be solved!

Souce: Dan Boneh (adapted by me)

Take home

CPA security cannot guarantee secrecy under active attacks.

Only use one of two modes:

➔If message needs integrity but no confidentiality:

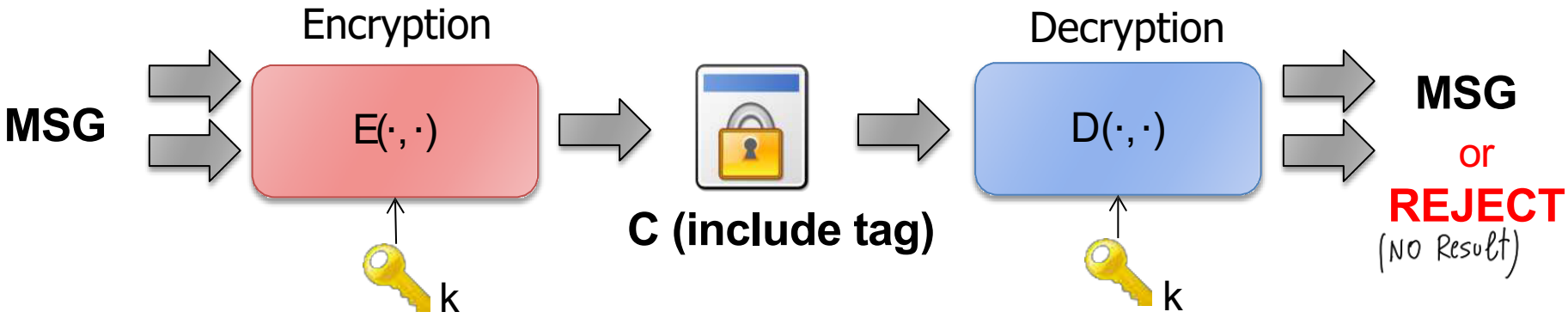
use a MAC

➔If message needs both integrity and confidentiality:

use authenticated encryption modes

Authenticated Encryption

size C_T > size Plaintext



Unlike standard ENC which always returns a msg, AE may output a "REJECT"

AE **fundamentally stronger** than basic cipher:
decrypts ONLY IF authentication tag is valid. CPA non più applicabile.

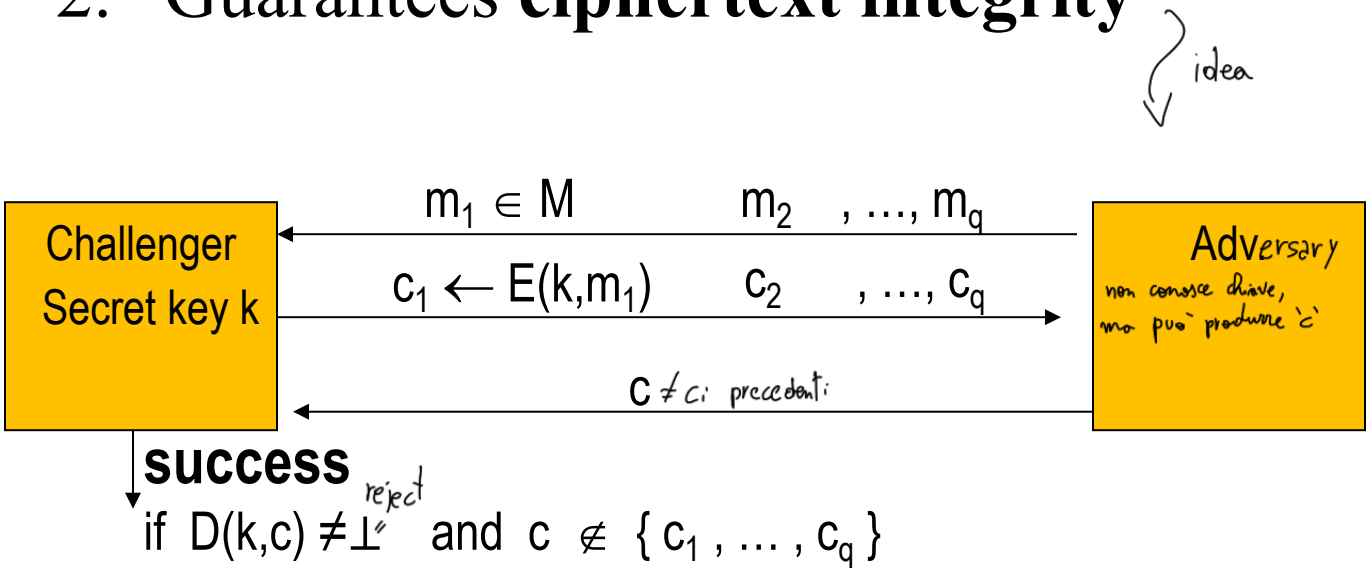
→*Prevents attacker from performing CCA*

Encryption then MAC soddisfa AE.

Security definition

➔AE is secure if:

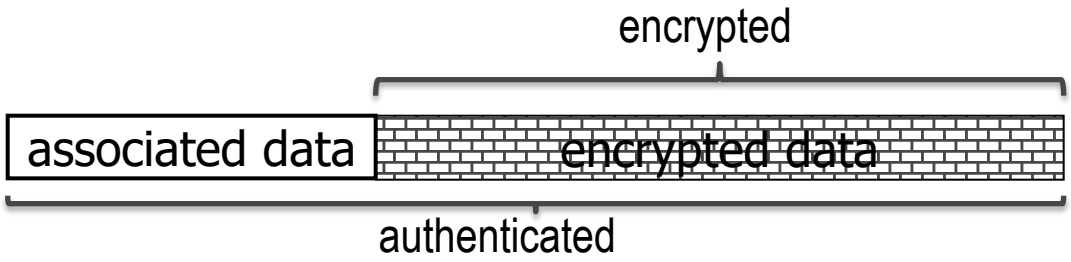
1. is semantically secure under CPA
2. Guarantees **ciphertext integrity**



ciphertext integrity (informally): if probability of success is negligible

(note: success even if forged c decryping to completely random msg!!!)

AEAD: AE with Associated Data



➔**Associated data: often packet hdr**

⇨hdr must usually remain in plaintext

➔**Extreme cases also frequent**

⇨No associated data → CCA-secure encryption

⇨No encrypted data → secure MAC

➔**Now should be clear why modern protocols only use AEAD (e.g. TLSv1.3)**

⇨If no AEAD, then use Encrypt-then-MAC, also CCA-secure

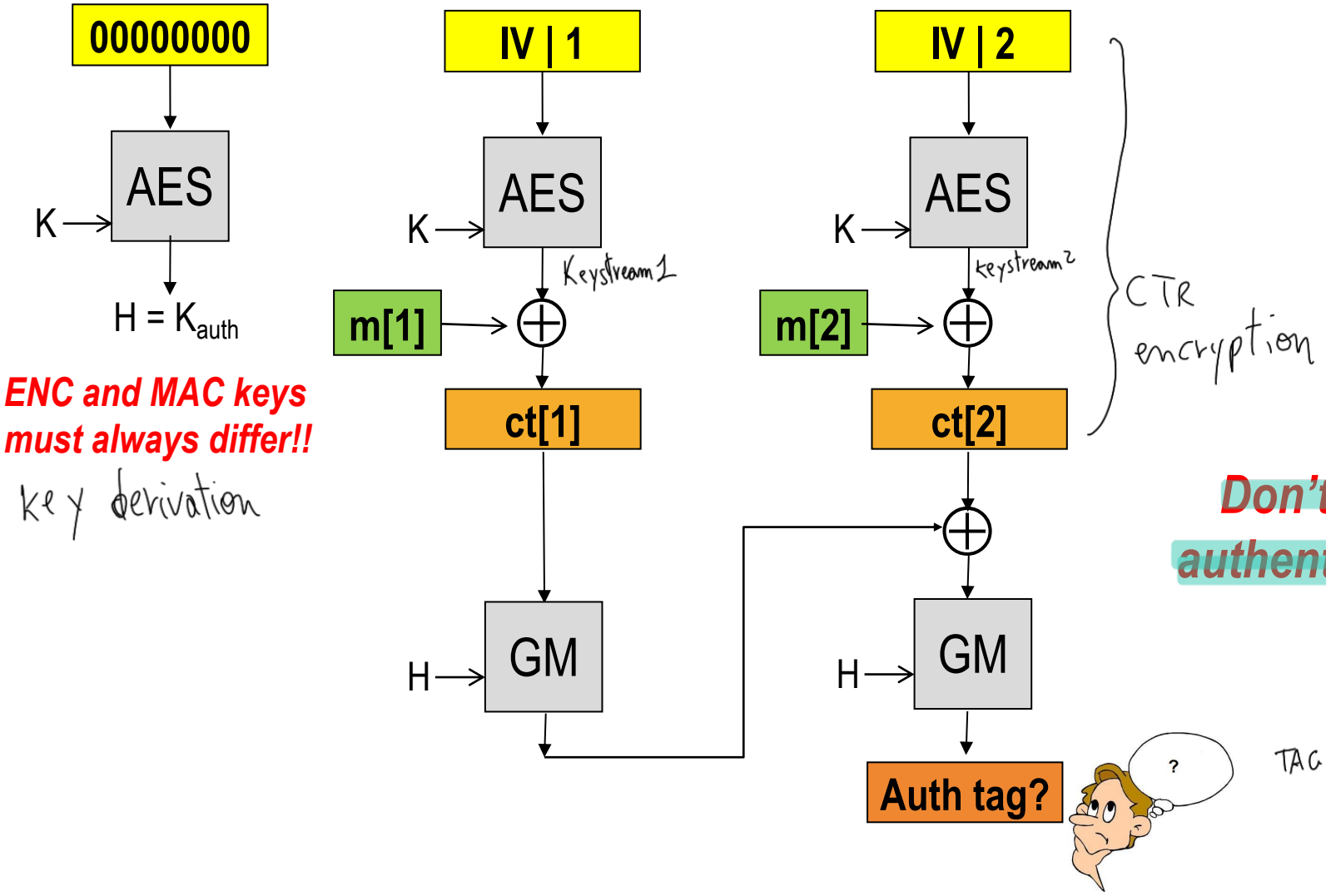
Design choices for AEAD

- **→Structure**
  - ⇨Two-layer = first encrypt and then MAC (e.g. AES-GCM)
  - ⇨One-layer = all in one (e.g. OCB → faster than GCM)
- **→Performance**
  - ⇨First encrypt then MAC may require two pass → slow
  - ⇨«full» parallelizability: hard
    - →authenticity check requires «all» data!
  - ⇨***Streamability (= online cipher): one pass***
- **→Functional requirements**
  - ⇨Where to place Associated Data? Before? After? Any mix?
- **→Misuse resistance**
  - ⇨How robust is AEAD **when IV is reused**?

AES-GCM: Galois Counter Mode

- **First standardized algorithm**
  - NIST SP 800 38D recommendation
  - Used in most security protocols (IPsec, TLS, …)
    - (though not used in WiFi → AES-CCM)
- **Encrypt-then-MAC structure**
  - Encrypt: AES-CTR (this is the CM – Counter Mode - in GCM)
  - MAC: GHASH (this is the G – Galois - in GCM)
    - Not a crypto hash, but a (much faster!) multiplication in GF(2^{128}) with an auth key
      - » more later Come la trasformo in secure auth. code?

Construction @ high level (example: msg in two blocks)



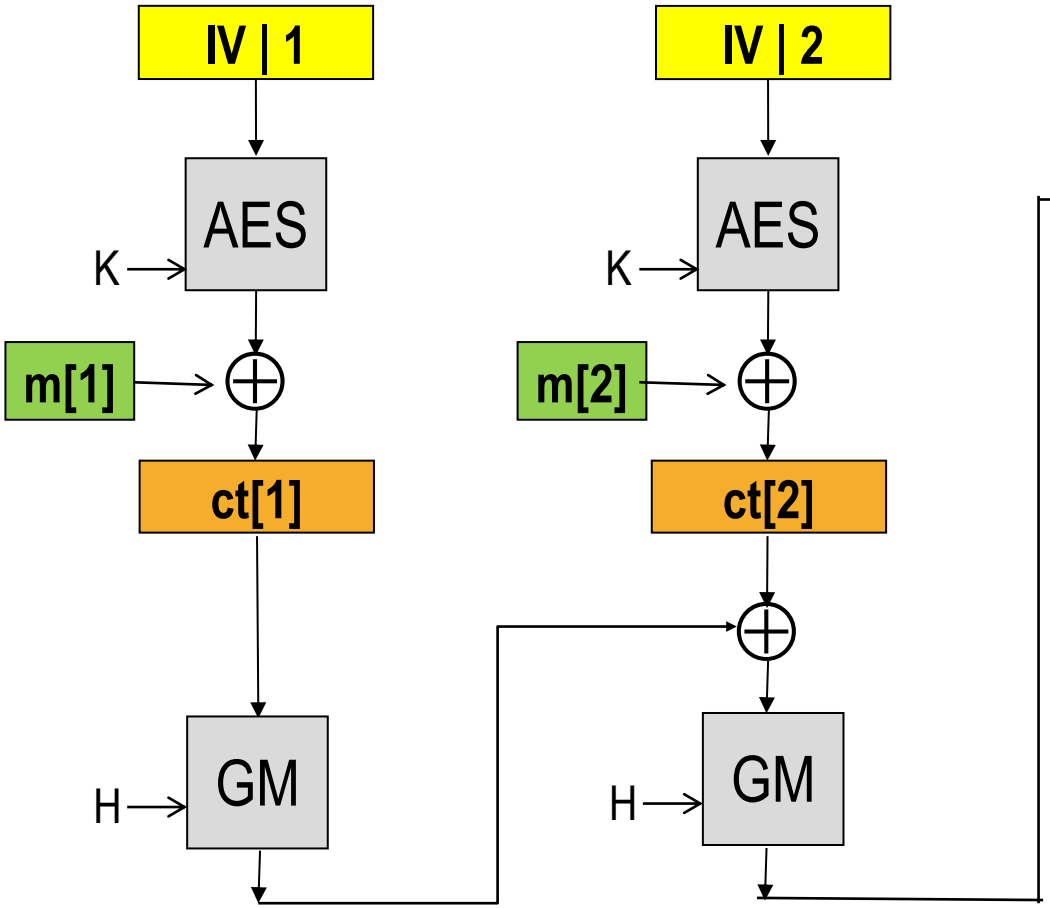
ENC and MAC keys must always differ!!

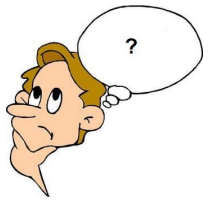
key derivation

Don't forget to authenticate LEN!!

TAG → msg
→ Key = H

Construction @ high level (example: msg in two blocks)

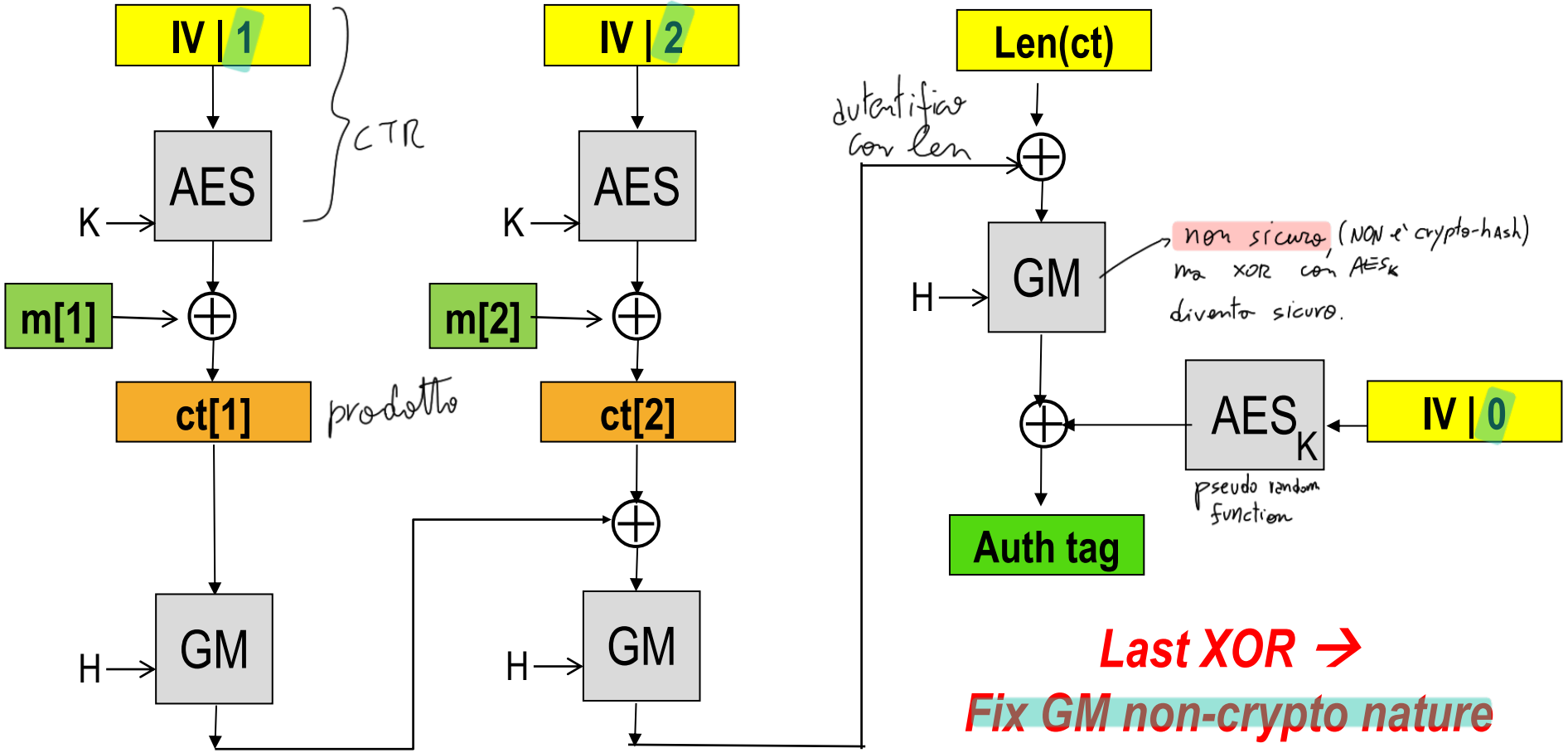




Don't forget to include IV in auth tag!!

(remember previous CCA example against IV)

Construction @ high level (example: msg in two blocks)



Last XOR →

Fix GM non-crypto nature (Wegman-Carter MAC)

(see next slides)

Wegman-Carter MAC

➔**Crypto hash are slow**

➔**Non crypto hash can be a lot faster**

➔**Wegman-Carter: how to MAC using NON-CRYPTO hash functions**

➔**Two building blocks:**

⇨Universal Hash Function , NON CRYPTO, HA ALCUNE PROP. DI ANTICOLLISIONE.

⇨Pseudo Random Function , AES-CTR result.

Universal Hash Function

➔ **A Keyed Hash Function is a FAMILY of functions**

⇨ $H_k(msg) \rightarrow \{0,1,\ldots, m-1\}$ → k = key, se cambia, hash func. cambia.

➔ **The family is «universal» if it is hard for an attacker which does NOT know the key to find a collision $H_k(M_1)=H_k(M_2)$**

⇨ MANY differences with respect to crypto hash:

→ If you know the key, finding a collision CAN be easy

→ The output $H_k(M)$ may NOT be pseudo-random

➔ **Companion property:**

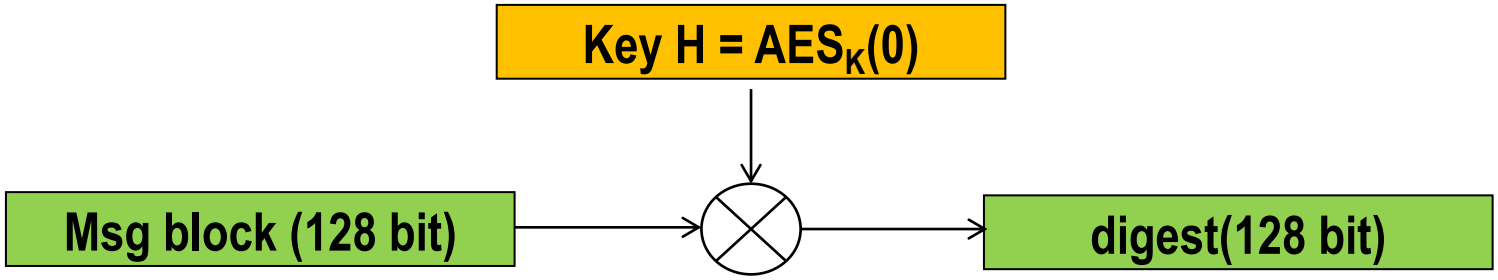
⇨ for any pair x≠y, Prob { $H_x(M) = H_y(M)$ } ≤ 1/m (m → specie possibili digest)

→ Changing key should randomly change digest (the probability that a collision repeats after changing key should be negligible)

→ there should be no pair (M1, M2) that gives the same hash for many different keys.

AES-GCM: GHASH

➔**GHASH is NOT crypto, but it is «just» a universal hash function**



➔**Building block: Multiplication in GF(2^{128})**

⇨a.k.a. polynomial carry-less multiplication

➔CLMUL() instruction available in CPUs

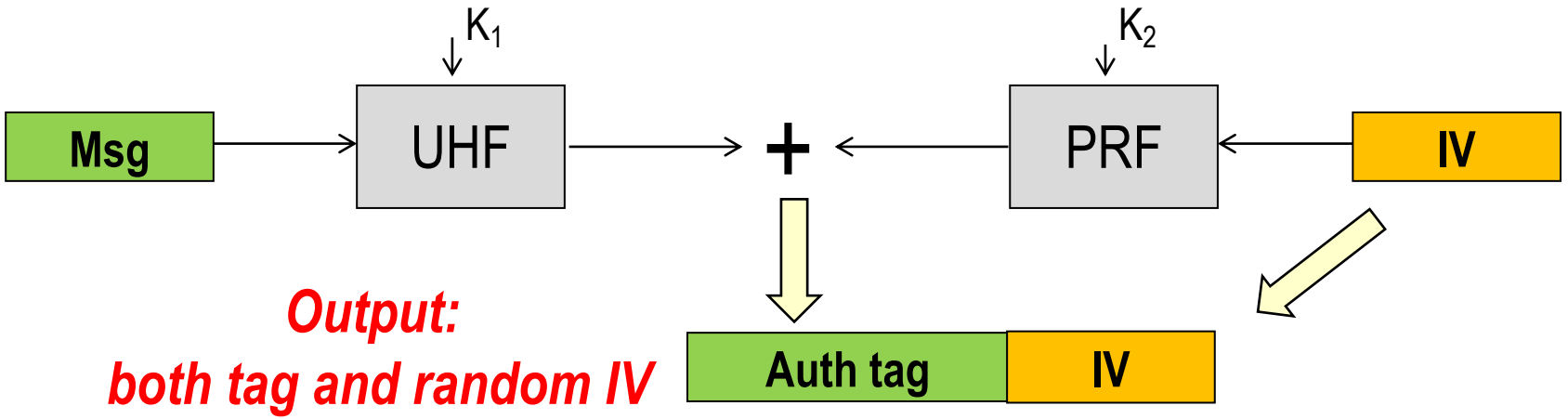
Se supero la size uso il riporto per non eccedere "riducendolo".

es: $x(x^{128} + x^7 + x^2 + x + 1) = p$

$x^{129} = x^8 + x^3 + x^2 + x$ (non uso i "-1")

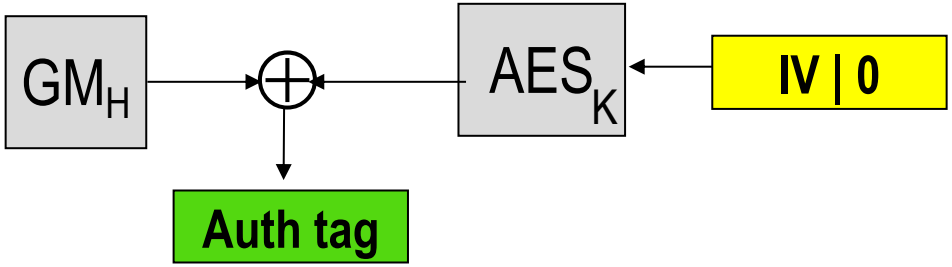
Wegman-Carter construction

➔**A secure MAC can be constructed from an UHF and a PRF as $UHF_{k1}(msg) + PRF_{k2}(rand)$**

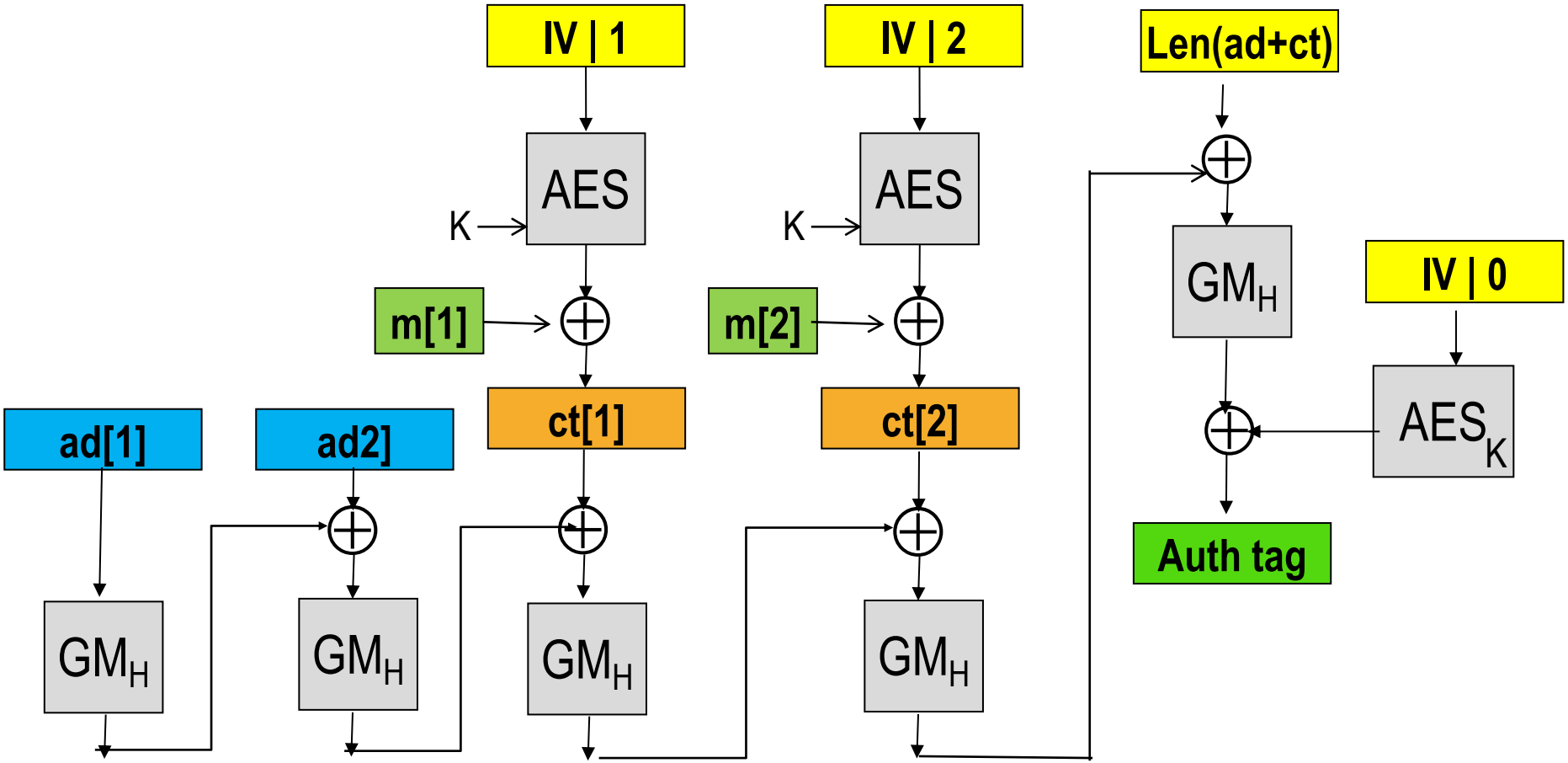


Output:
both tag and random IV
a new tool for you:
randomized MAC!

AES-GCM specific case:



What about Associated Data?



Len adjusted to include also associated data

Come si usa?

GCM critical issue: nonce reuse

➔**AES-GCM tag:**

⇨tag = $GHASH_H(CT) \oplus AES_K(IV,0)$

➔**Nonce reused (same IV):**

⇨tag1 = $GHASH_H(CT1) \oplus AES_K(IV,0)$

⇨tag2 = $GHASH_H(CT2) \oplus AES_K(IV,0)$

⇨tag1 ⊕ tag2 = $GHASH_H(CT1) \oplus GHASH_H(CT2)$

➔**But GHASH is… linear (poly mult)!**

⇨Attacker can now recover auth key H, and forge valid messages!

→(Though at least encryption key K remains unknown…)

➔**There exists some mode robust to nonce reuse?**

⇨Yes, AES-GCM-SIV (Synthetic IV), see RFC 8452, April 2019