LDACS1310: Introduction to Applied Cryptography

Lecture 4: message authentication

T. Peters, ***F.-X. Standaert***

UCLouvain Crypto Group, fstandae@uclouvain.be

Reminder

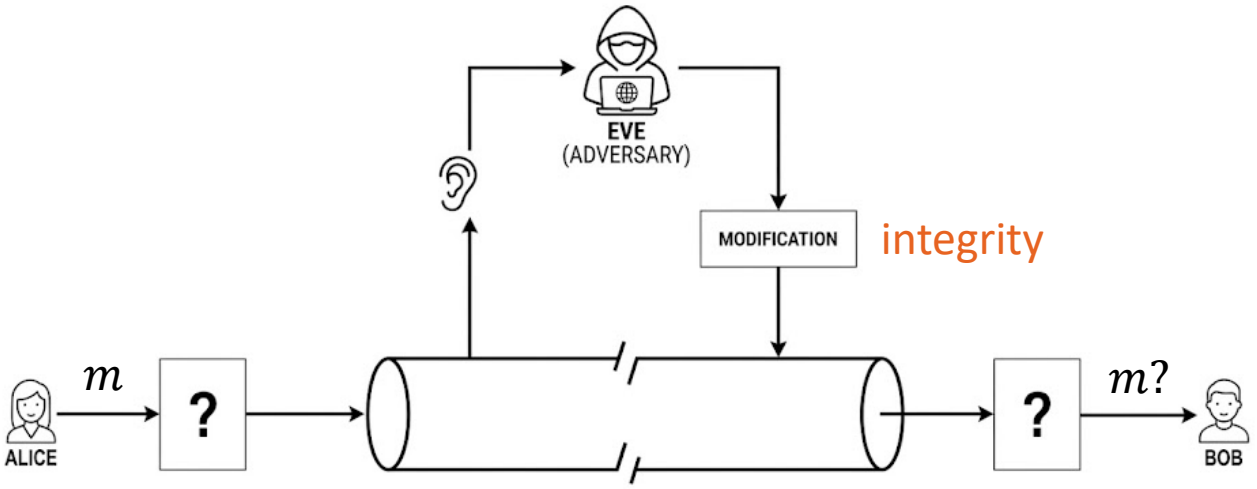


https://perso.uclouvain.be/fstandae/LDACS1310/

And questions from last lecture?

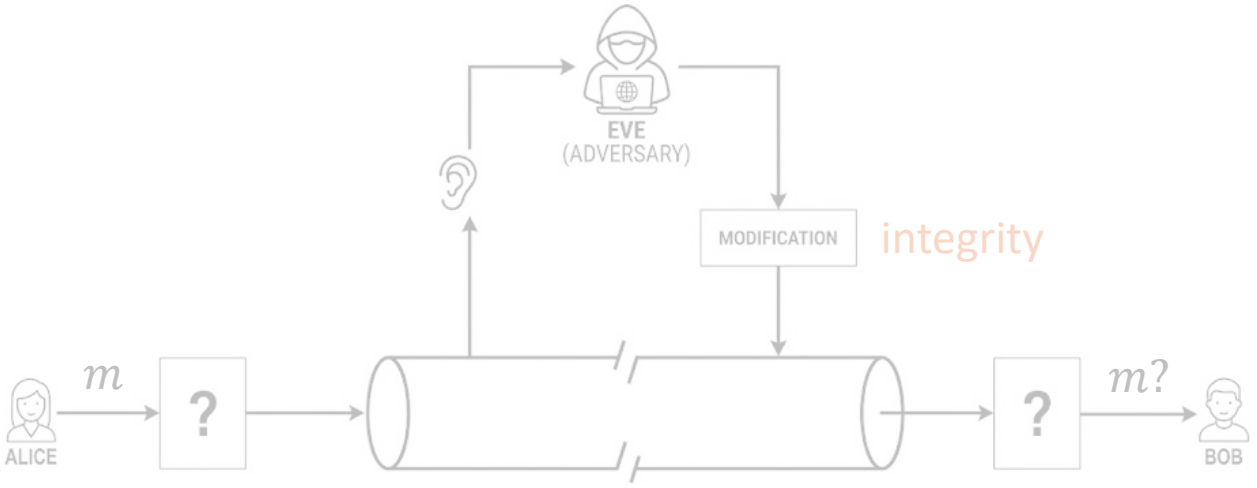
Why and what?

- Integrity is essential to system security (e.g., software updates)



Why and what?

- Integrity is essential to system security (e.g., software updates)



Msg. Auth. Code (MAC)
Key generation: $k \leftarrow \mathrm{Gen}(n)$ Tag generation: $t \leftarrow \mathrm{Mac}_k(m)$ Tag verif.: $\mathrm{Ver}_k(m,t) = \mathsf{T}$ or $\perp$

- MACs are the (symmetric crypto.) answer to this challenge

First attempts

- Question: is $t = m(0) \oplus m(1) \oplus \ldots \oplus m(l)$ a secure MAC?

First attempts

- Question: is $t = m(0) \oplus m(1) \oplus \ldots \oplus m(l)$ a secure MAC?
  - No, the adversary can easily change m without changing t
    (security against malicious adversaries ≠ random faults)
    ⇒ Integrity (like confidentiality) requires a cryptographic key

- Question: is $t \leftarrow \mathrm{Mac}_k(m)$ with a 5-bit t a secure MAC?

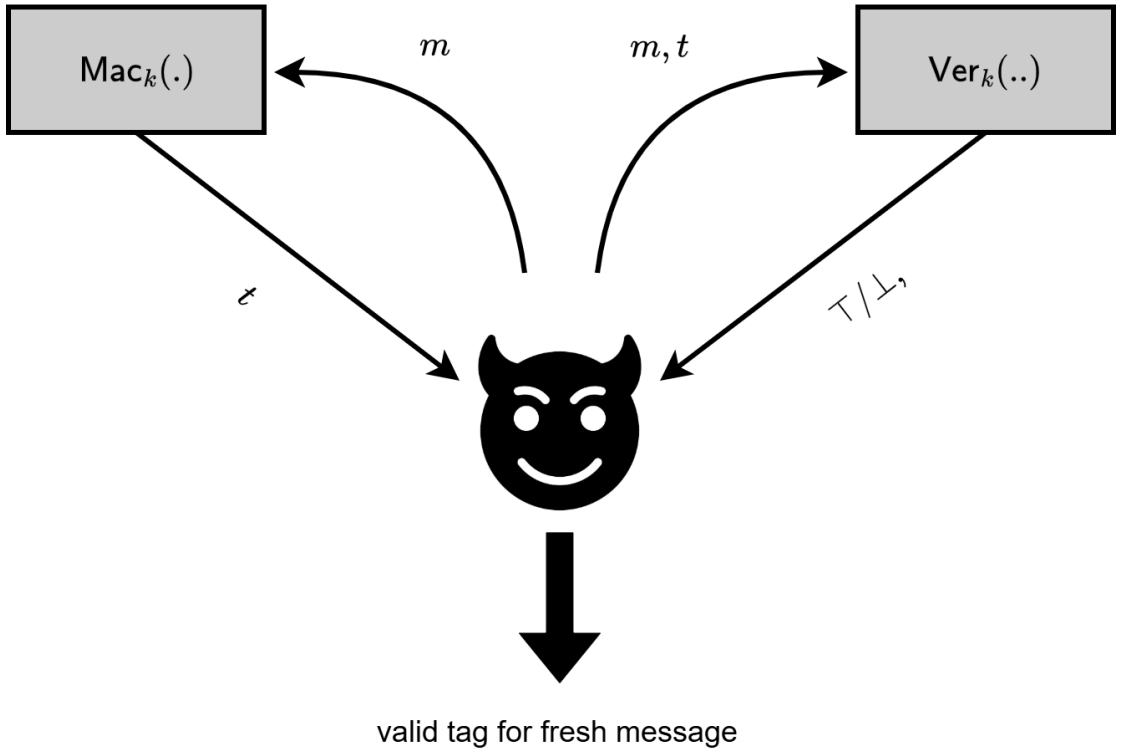
First attempts

- Question: is $t = m(0) \oplus m(1) \oplus \ldots \oplus m(l)$ a secure MAC?
  - No, the adversary can easily change m without changing t
    (security against malicious adversaries ≠ random faults)
    ⇒ Integrity (like confidentiality) requires a cryptographic key

- Question: is $t \leftarrow \mathrm{Mac}_k(m)$ with a 5-bit t a secure MAC?
  - No, the adversary can simply guess the tag for messages
    ⇒ Integrity requires both the key and the tag to be large enough

Security definition (I)

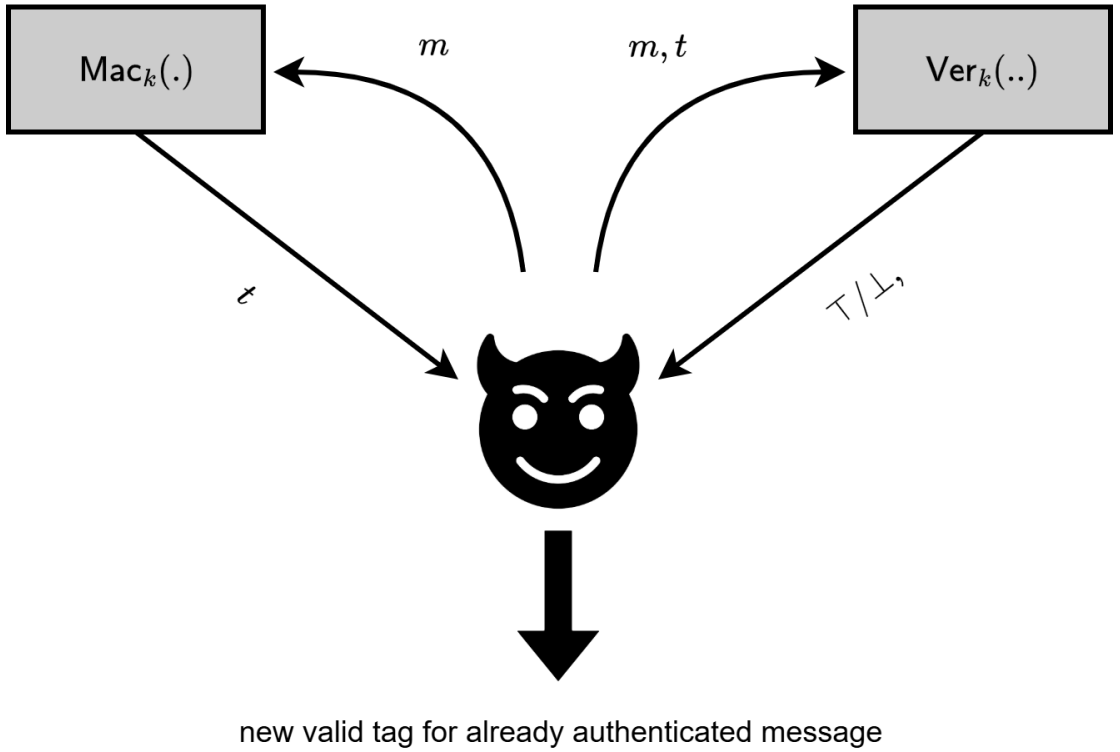
- The MAC unforgeability experiment {…}



A MAC $= (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Ver})$ ensures existential unforgeability against adaptive chosen message attacks if ∀ polytime Adv, we have $\Pr\left(\mathrm{Exp}^{\mathrm{unf}}_{\mathrm{MAC}}(n) = 1\right) \leq \mathrm{negl}(n)$

Security definition (II)

- The MAC strong unforgeability experiment {…}



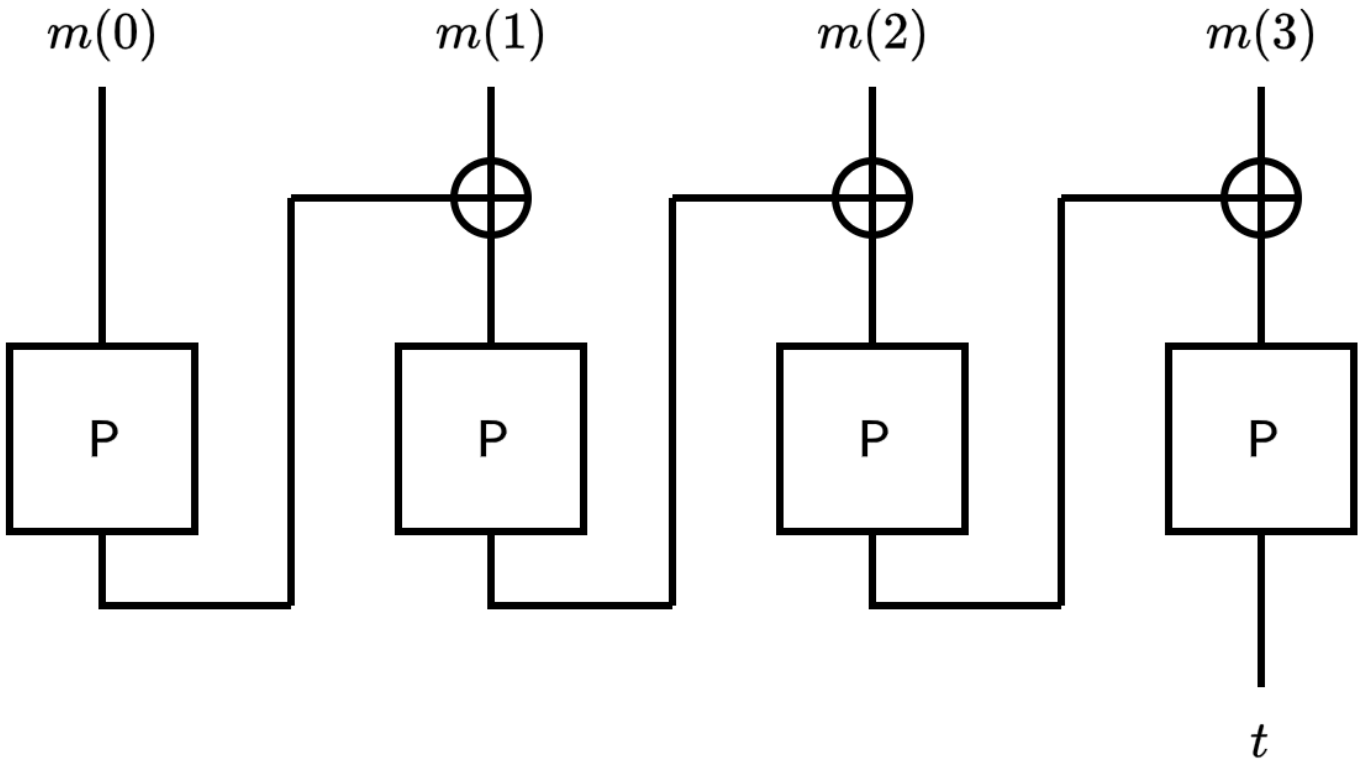
A MAC $= (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Ver})$ ensures strong existential unforgeability against adaptive chosen message attacks if ∀ polytime Adv

$$\Pr\left(\mathrm{Exp}^{\mathrm{suf}}_{\mathrm{MAC}}(n) = 1\right) \leq \mathrm{negl}(n)$$

(*Note: a MAC with canonical verification is a strong MAC*)

Design: CBC-MAC (raw version)

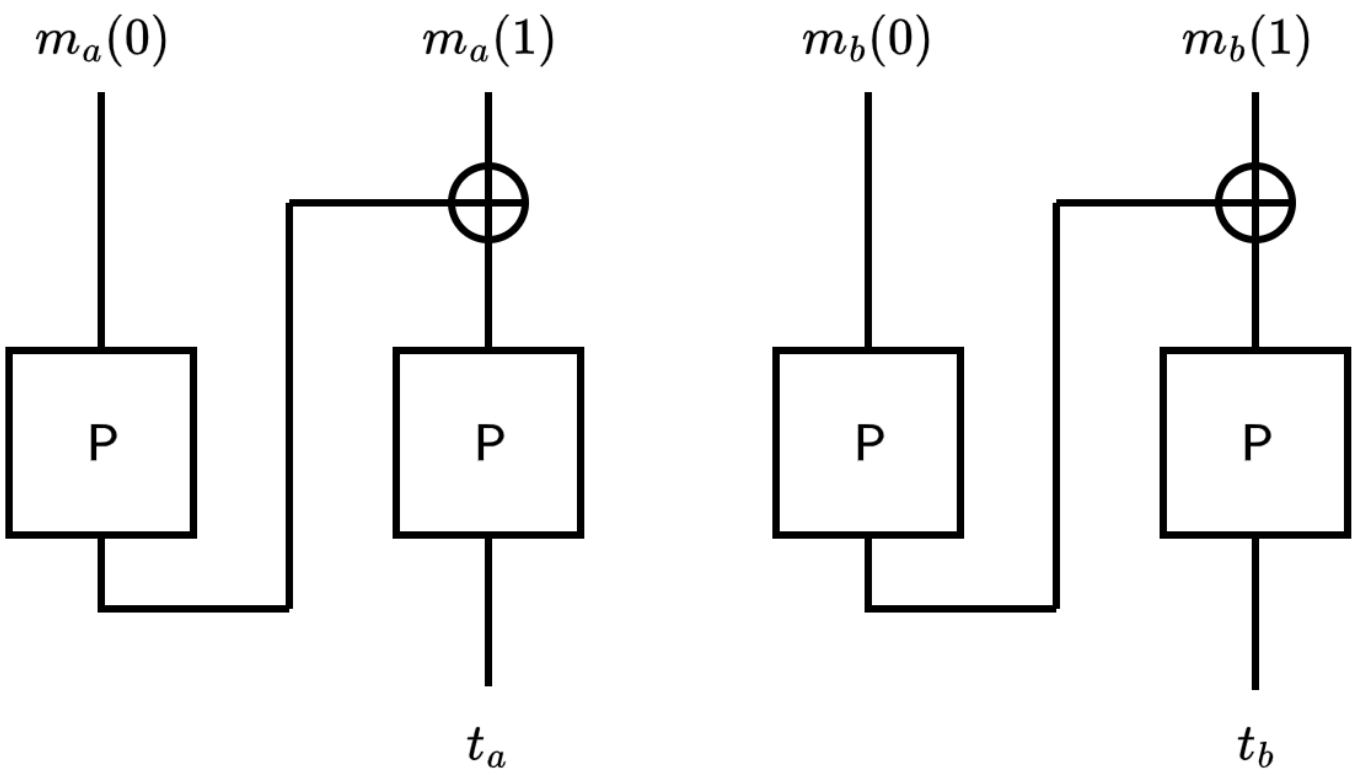
- Similar to CBC-ENC, but deterministic (no IV needed)



Design: CBC-MAC (raw version)

- Similar to CBC-ENC, but deterministic (no IV needed)

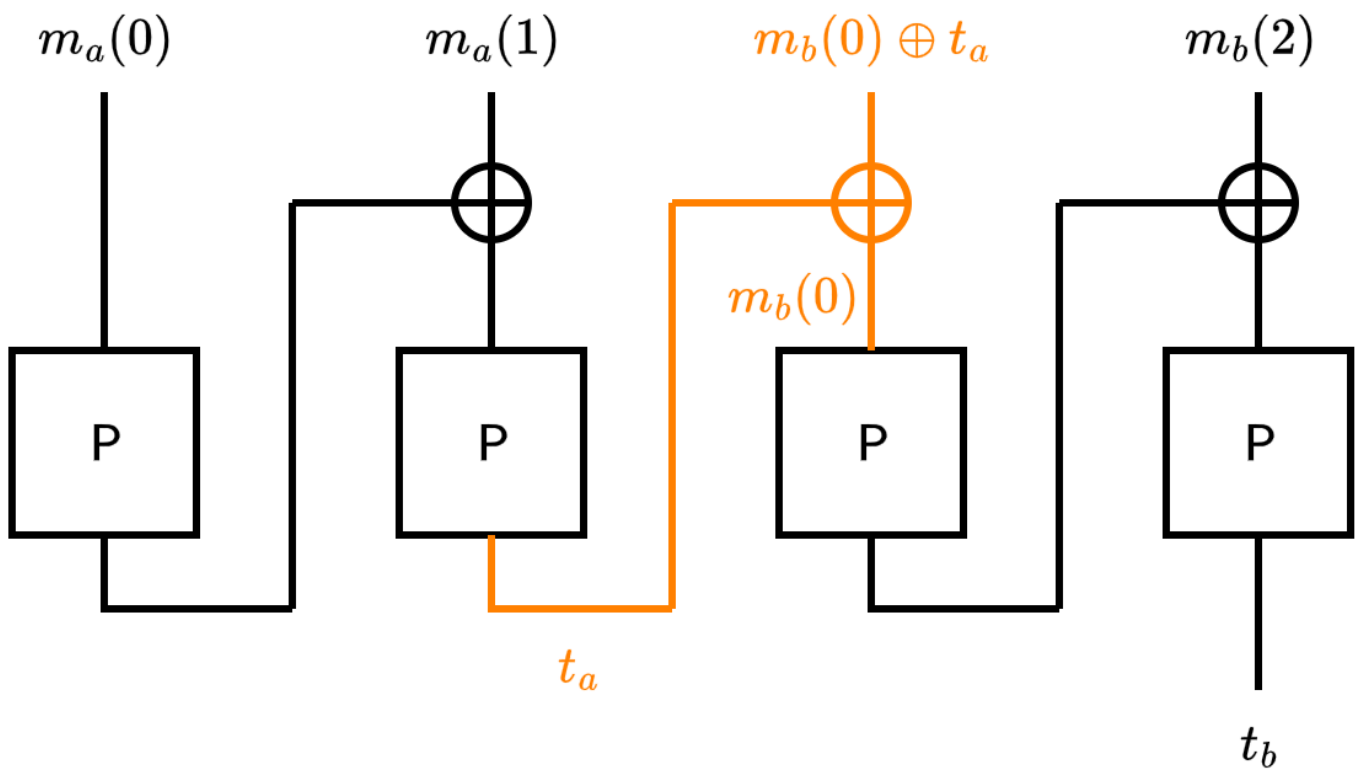
❑ Question: how to forge a MAC for $m_a||m_b$ given t_a and t_b



Design: CBC-MAC (raw version)

- Similar to CBC-ENC, but deterministic (no IV needed)

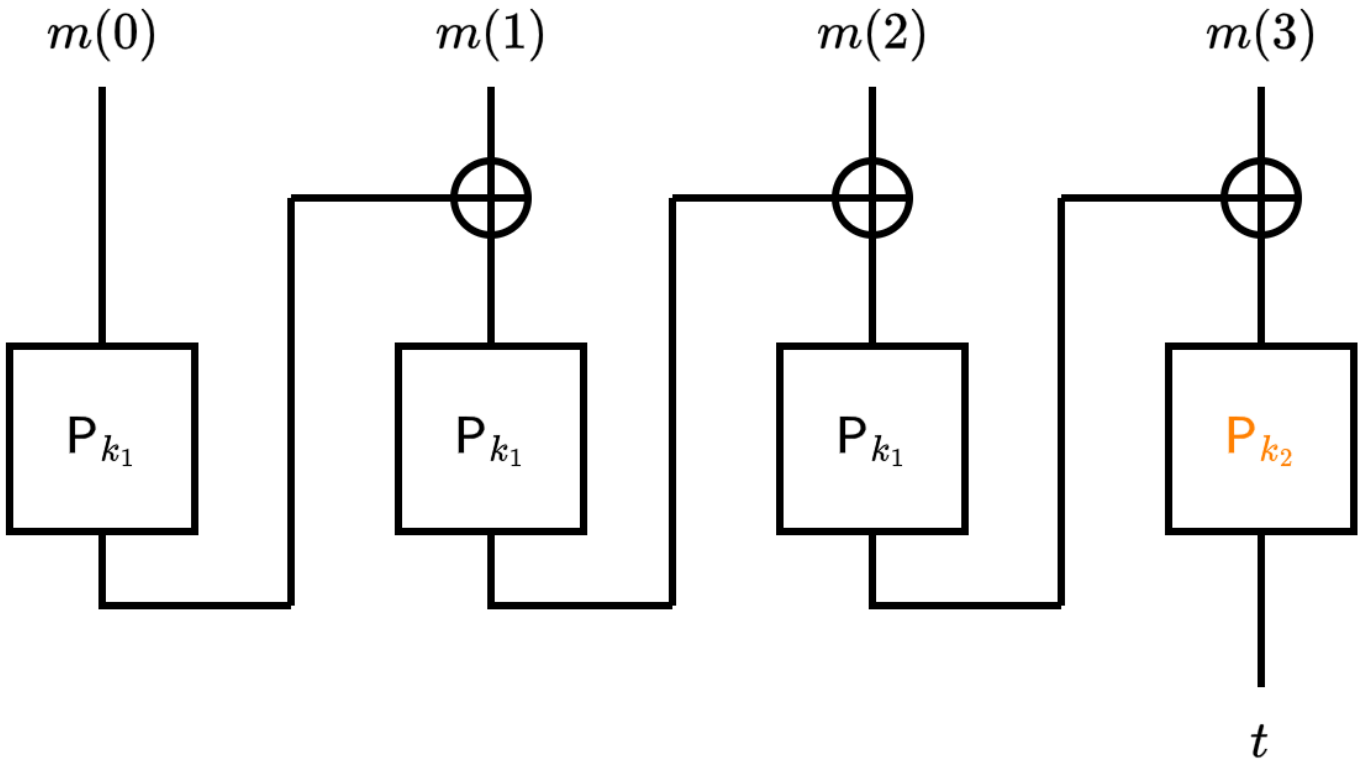
❑ Question: how to forge a MAC for $m_a||m_b$ given t_a and t_b



Raw CBC-MAC susceptible to length extension attacks

Design: CBC-MAC (secure version #1)

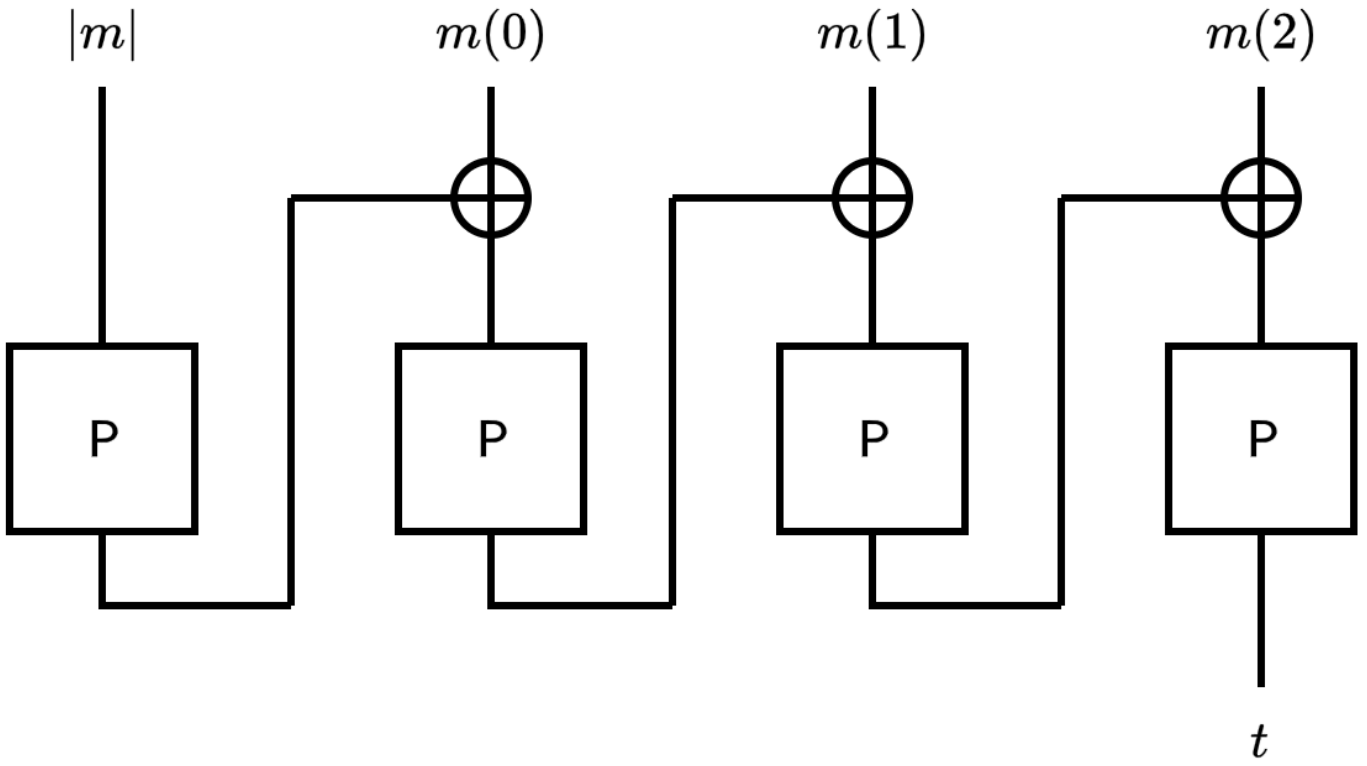
- Similar to CBC-ENC, but deterministic (no IV needed)
- ❑ Question: how to forge a MAC for $m_a||m_b$ given t_a and t_b



Use a different key for the last message bloc (but doubles key size)

Design: CBC-MAC (secure version #2)

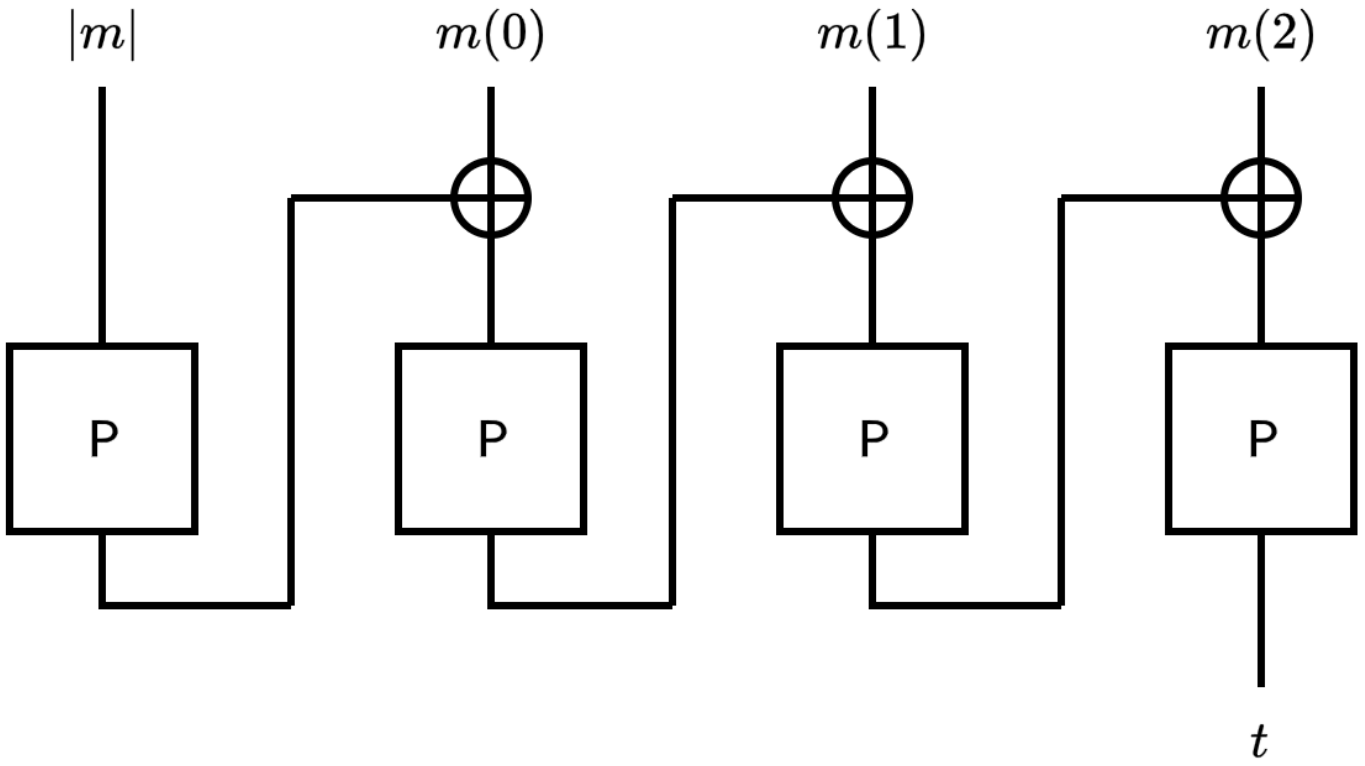
- Similar to CBC-ENC, but deterministic (no IV needed)
- ❑ Question: how to forge a MAC for $m_a||m_b$ given t_a and t_b



Insert the message length as first bloc (but size must be known in advance)

Analysis (reduction)

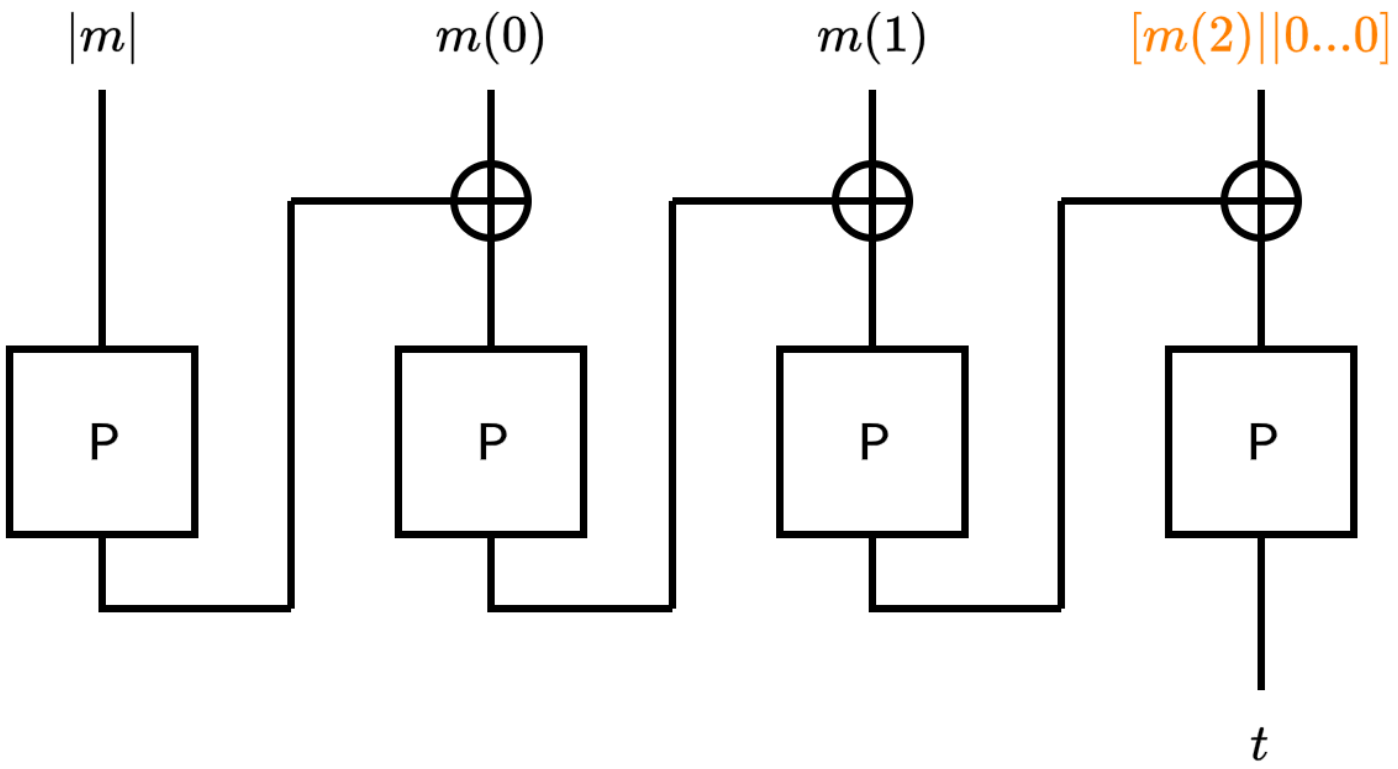
- If P is a $(t, \epsilon_{\mathrm{P}})$ PRP, CBC-MAC is $(t - q \cdot l \cdot \tau_{\mathrm{P}}, \epsilon_{\mathrm{unf}})$-unforgeable with $\epsilon_{\mathrm{unf}} \leq \epsilon_{\mathrm{P}} + \frac{q^2}{2^n}$ and τ_{P} the permutation's execution time



Is the bound tight {…}?

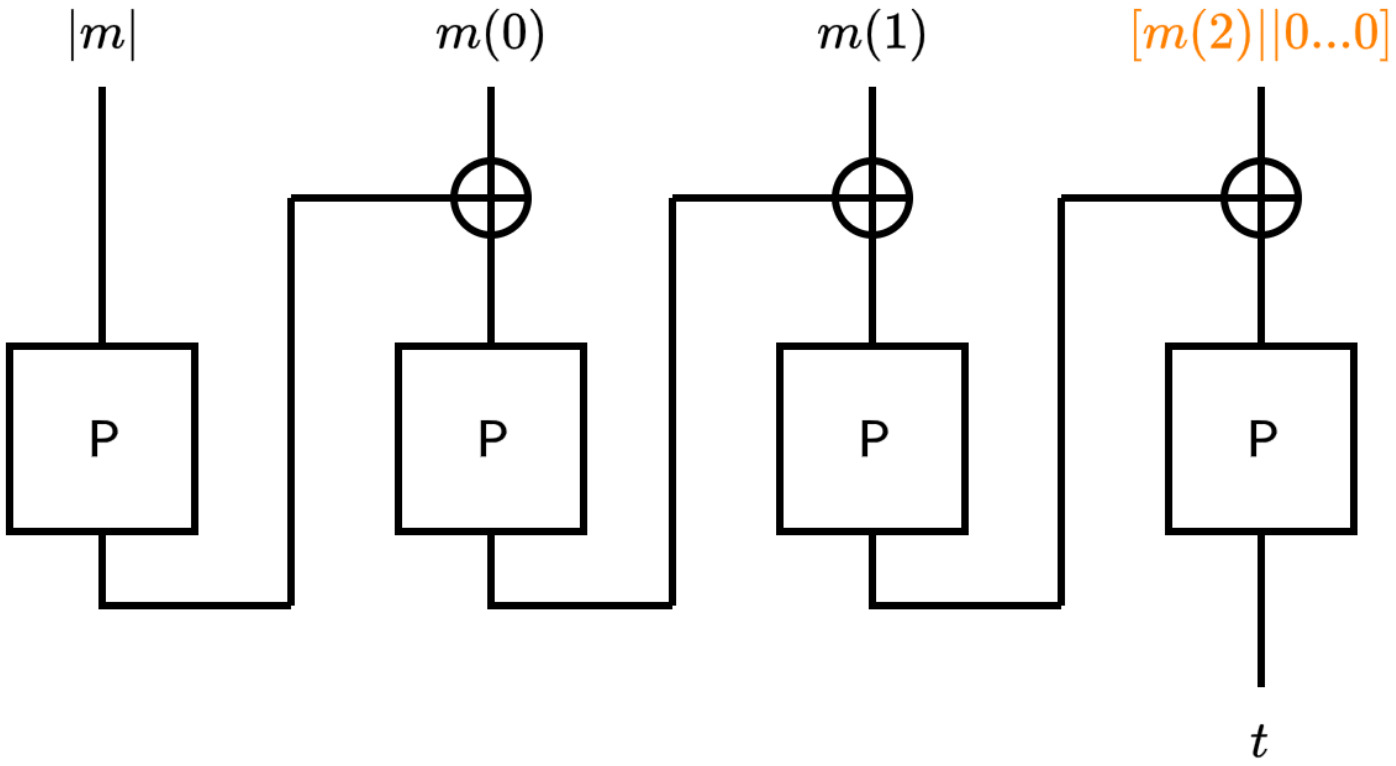
Additional caveats (I)

- Question: is a “zero padding” sufficient?



Additional caveats (I)

- Question: is a "zero padding" sufficient? No: given tag on a message m, an adversary obtains a tag on message $m||0..0$



Padding must be invertible

$$m \neq m'$$

$$\Rightarrow \text{pad}(m) \neq \text{pad}(m')$$

ISO: pad with 10...0 (add a dummy block if needed)

Additional caveats (II)

- Unforgeability does not prevent “replay attacks”, which must be managed at the protocol level. The standard solutions is to include device-ID||counter as first/second message block

❑ Question: what if the adversary checks the tag byte per byte and answers $\perp$ as soon as a single byte is incorrect?

Additional caveats (II)

- Unforgeability does not prevent "replay attacks", which must be managed at the protocol level. The standard solutions is to include device-ID||counter as first/second message block

- ❑ Question: what if the adversary checks the tag byte per byte and answers ⊥ as soon as a single byte is incorrect? Divide-and-conquer timing attacks are possible: try 256 values for the first byte, the correct one should answer ⊥ later, repeat 16 times (*in general, physical side-channels can break many schemes*)

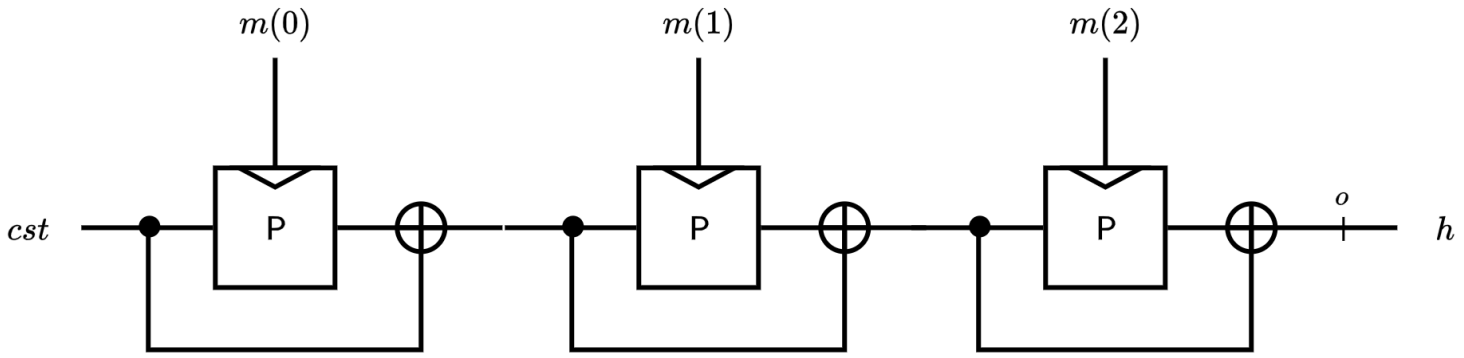
- Secure MACs require careful instantiation/implementation

Other useful building block (I)

- Collision-resistant hash functions $\mathsf{H} : \{0,1\}^* \times \{0,1\}^o$

$$\Pr\left(\mathrm{Adv}^{\mathsf{H}(.)} \rightarrow (m_0, m_1 \text{ s.t. } \mathsf{H}(m_0) = \mathsf{H}(m_1)\right) \leq \mathrm{negl}(n)$$

- Can be constructed from a block cipher

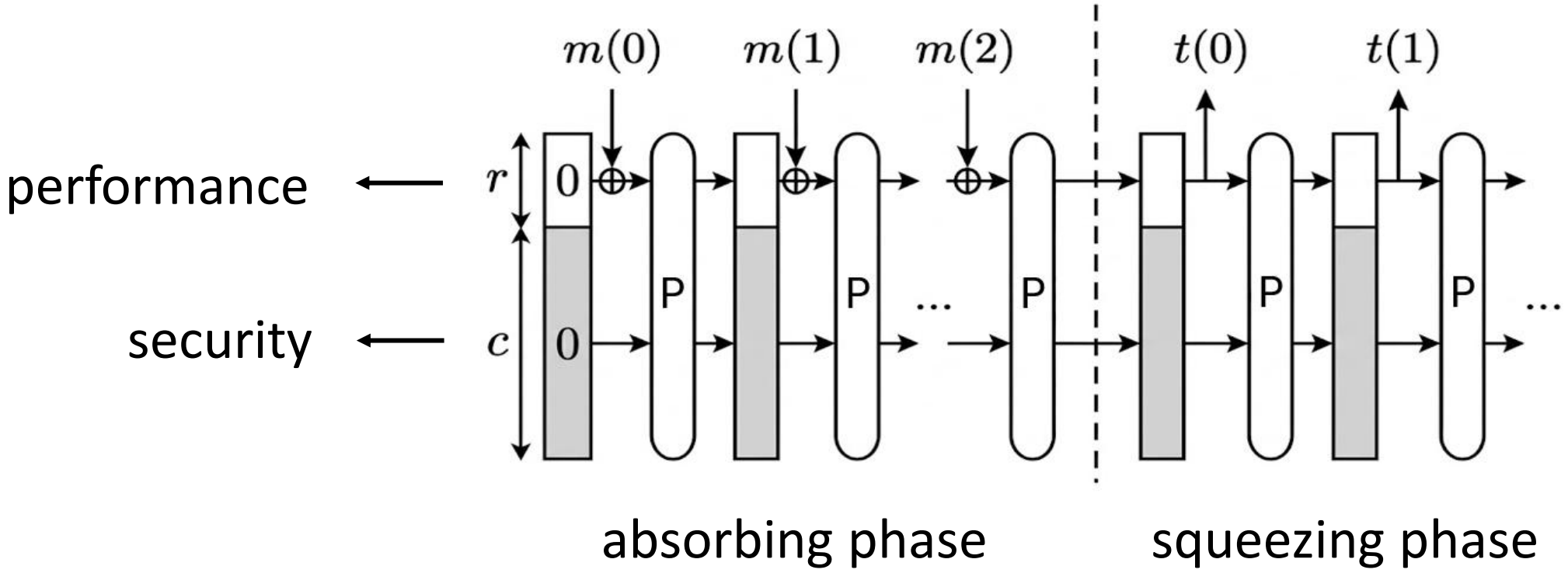


$$\epsilon_{\mathrm{cr}} \approx \frac{q^2}{2^o}$$

- Beware: "key" controlled by the Adv ⇒ PRP security is not enough)
- Ad hoc designs like the SHA3 standardare usually more efficient

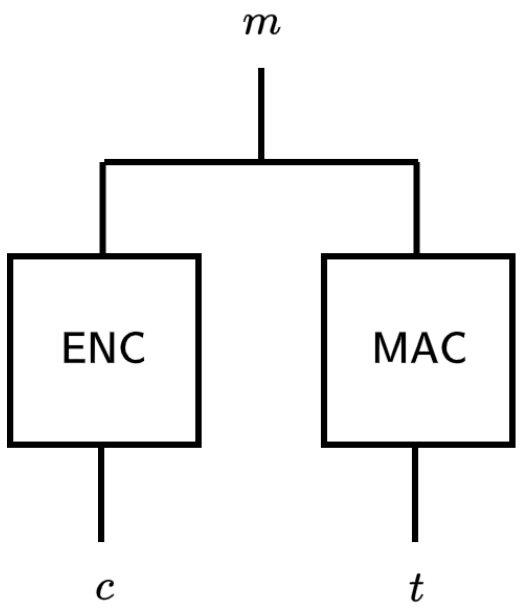
Other useful building blocks (II)

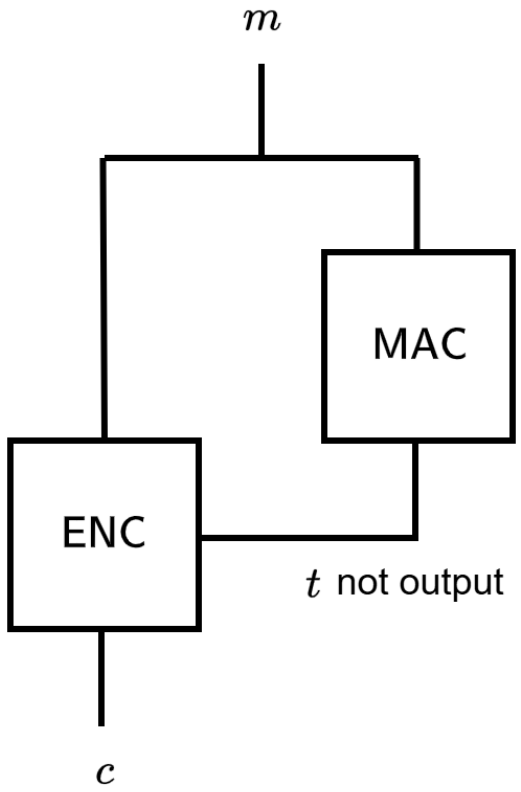
- SHA3 ≈ permutation-based (aka sponge-based) hash function

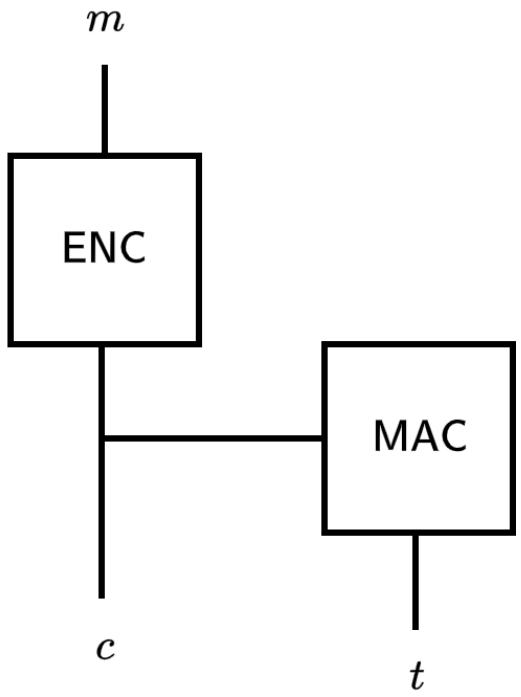


- Standardized as Federal Inf. Process. Std. (NIST FIPS) 202

Composition challenges (again)

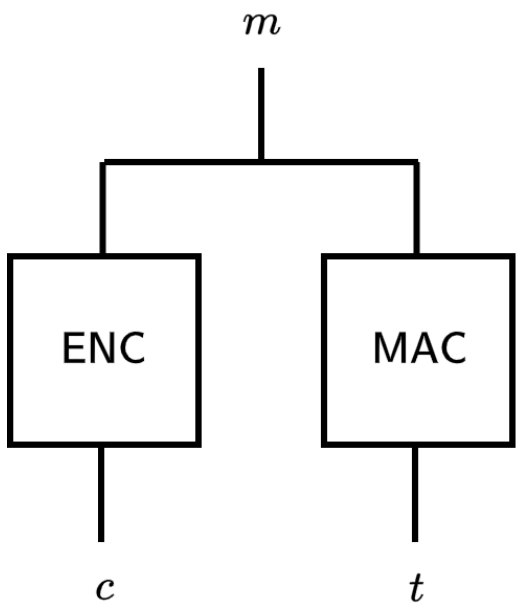


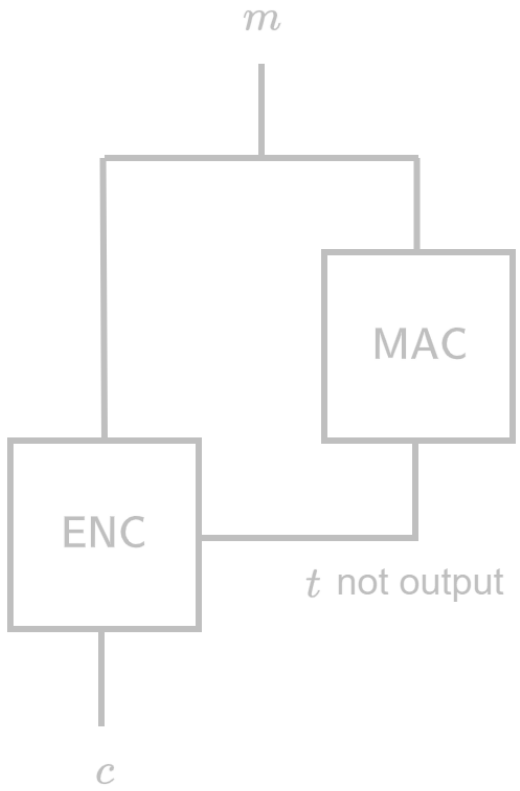


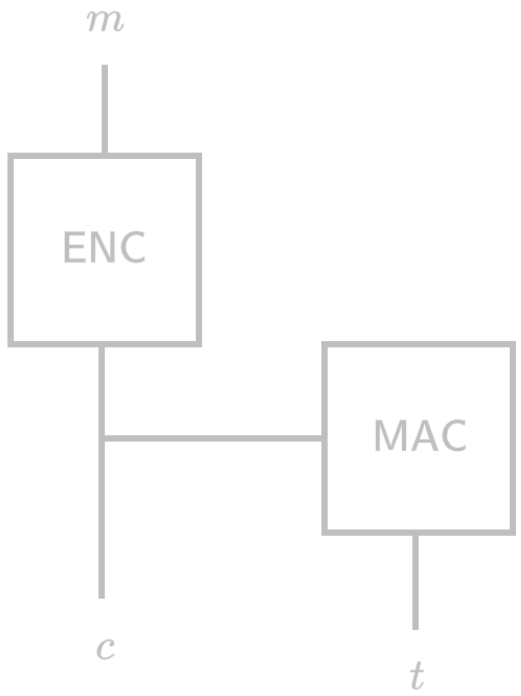


- Question: ENC-and-MAC, MAC-then-ENC or ENC-then-MAC?

Composition challenges (again)

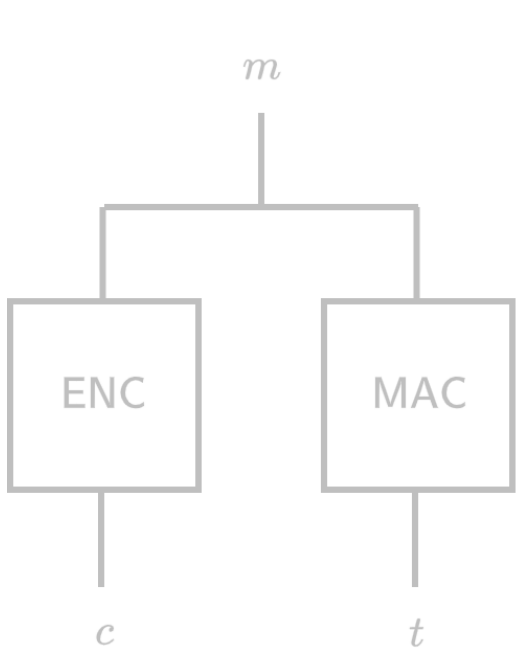


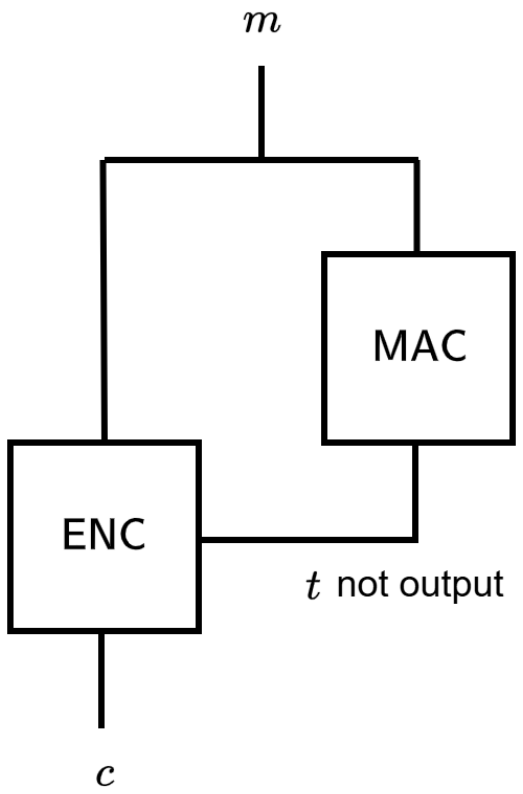


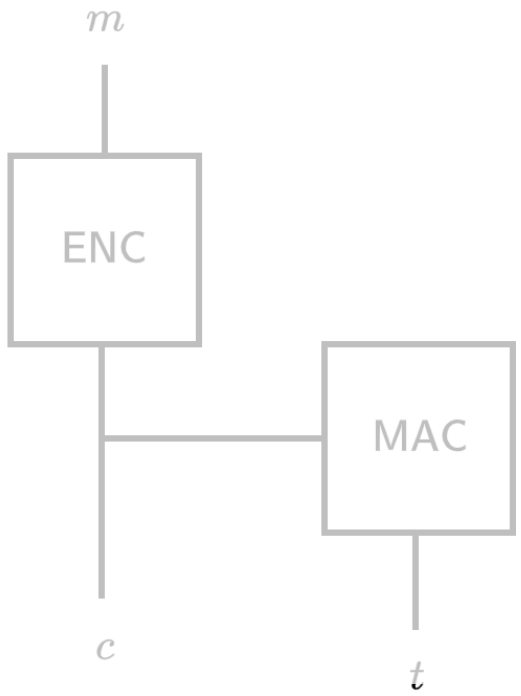


- ENC-and-MAC is not even secure against CPA attacks (nothing prevents the MAC to leak some bits of the message)

Composition challenges (again)

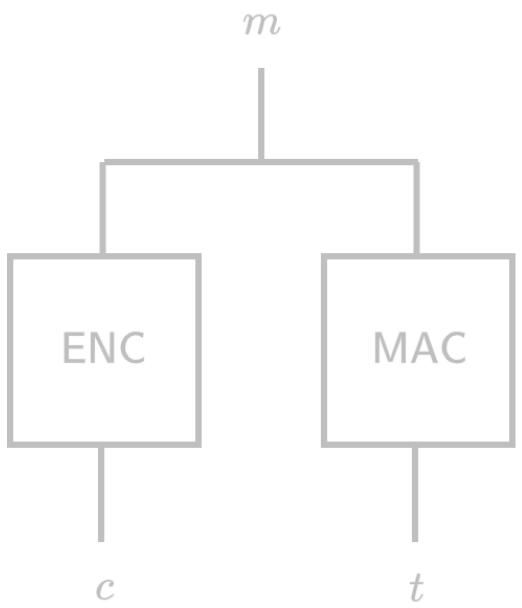


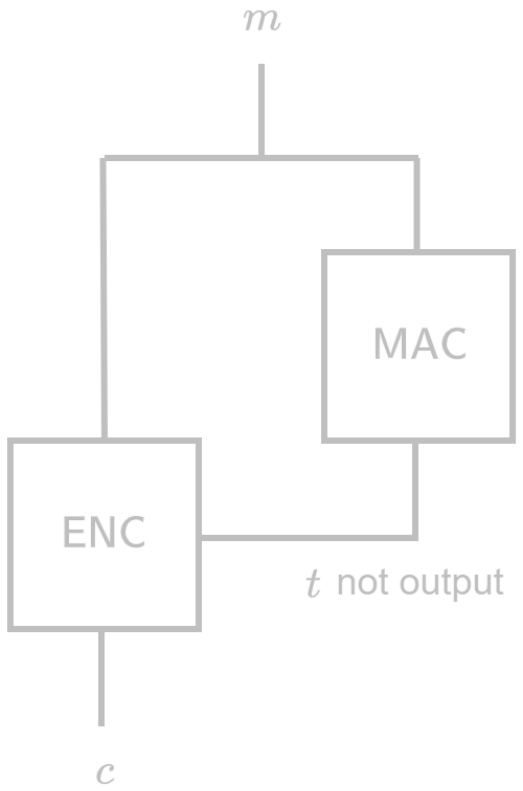


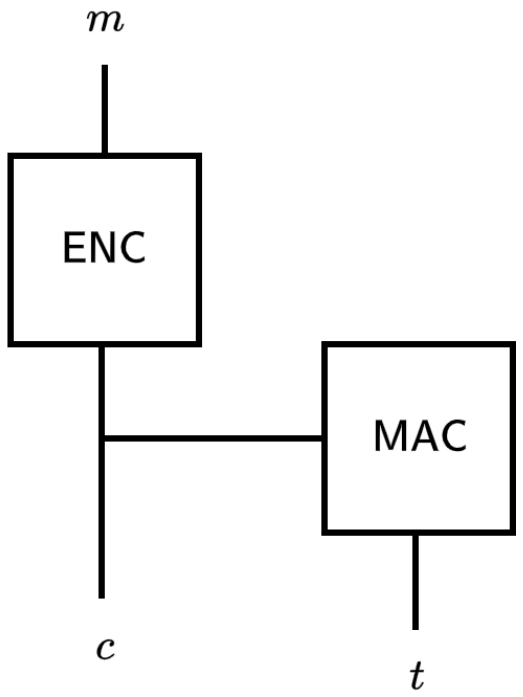


- MAC-then-ENC exposes m in decryption even if t is incorrect
  + padding oracle attacks (if pad & tag errors can be distinguished)

Composition challenges (again)







- ENC-then-MAC is the best (generically): it can ensure CCA security (≈ CPA security enhanced with non-trivial decryption queries)

What is used in practice (examples)

- Encryption only: AES in counter mode (NIST SP 800-38A)
  - More parallelizable than CBC (no block chaining)

What is used in practice (examples)

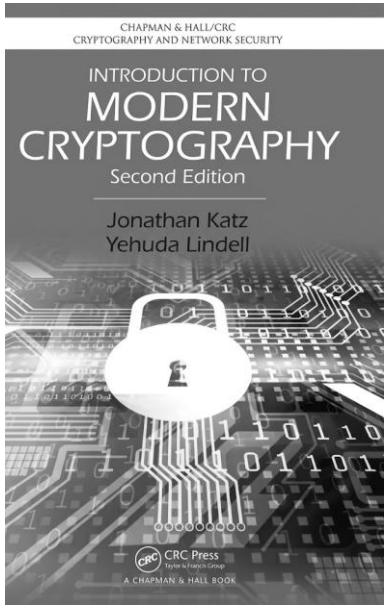
- Encryption only: AES in counter mode (NIST SP 800-38A)
  - More parallelizable than CBC (no block chaining)

- Authentication only: CMAC ≈ secure CBC-MAC (NIST SP 800-38B) or HMAC ≈ a hash-based variant (NIST FIPS 198-1)

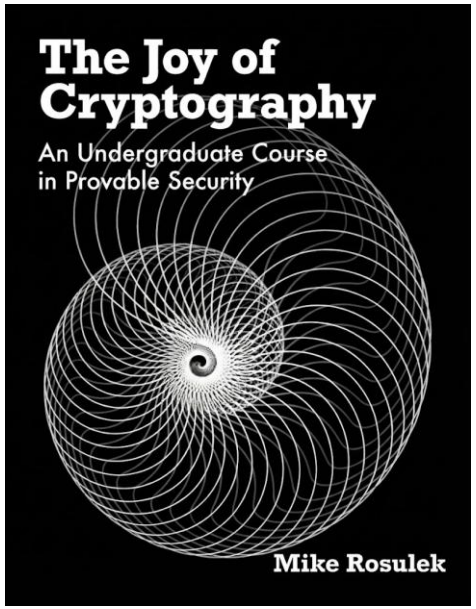
What is used in practice (examples)

- Encryption only: AES in counter mode (NIST SP 800-38A)
  - More parallelizable than CBC (no block chaining)
- Authentication only: CMAC ≈ secure CBC-MAC (NIST SP 800-38B) or HMAC ≈ a hash-based variant (NIST FIPS 198-1)
- Authenticated encryption: AES-GCM (Galois Counter Mode, NIST SP 800-38D) or Ascon (lightweight crypto, NIST SP 800-232)
  - Prevents many caveats https://competitions.cr.yp.to/disasters.html

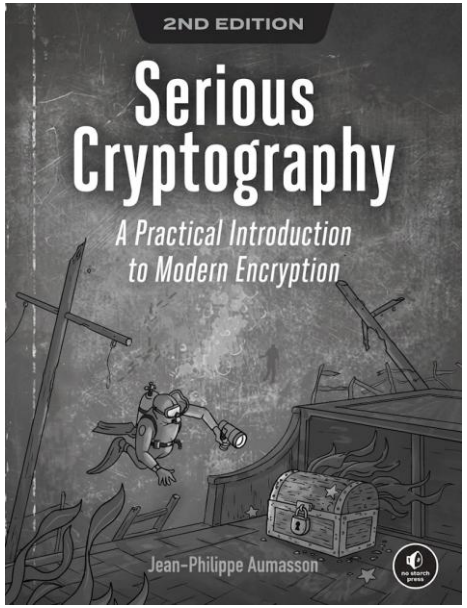
References



(a)



(b)



(c)

This lecture mostly follows Chapter 4 of Reference (a). The security definitions, cryptographic primitives and modes of operation we discuss are detailed there (with a number of additional concepts and examples that we skipped for conciseness purposes). Reference (b) provides an alternative view to these topics. Reference (c) provides discusses these topics in a less technical manner and provides a simple explanation and discussion of padding oracle attacks.

Questions?