Cryptanalysis
Part IV: Cryptanalysis of Encryption Modes

André Schrottenloher

Inria Rennes
Team CAPSULE

Recap

In symmetric cryptography we have two categories of objects:

- **Primitives**: small, fixed-size objects (block ciphers, compression functions, etc.)
- **Modes**: use the primitives to create true cryptographic functionality: (authenticated) encryption, hashing, etc.

$\Longrightarrow$ **With actual security goals** like confidentiality and authenticity

Modes of operation have **security proofs** which reduce the security to the one of the primitive.

- So we can focus attacks on the primitives
- We should still use the modes with caution (remember Merkle-Dåmgard)

Recap: block ciphers

$$E_K \; : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

- **Family of permutations** of $\{0,1\}^n$ **indexed** by $K \in \{0,1\}^k$
- $k \simeq 80$ to 256 bits, $n \simeq 64$ to 256 bits.

Block cipher security:

- PRP: E_K with random K looks like a random permutation ...
- Strong-PRP: ...even with inverse queries
- Ideal cipher: the whole E looks like a random family of permutations

Recap: IND-CPA and IND-CCA security

A symmetric encryption scheme is:

IND:

An adversary communicating with a **challenger** produces two messages m_0, m_1, learns the **challenge ciphertext** c^*, cannot distinguish if $c^* = \text{Enc}(m_0)$ or $c^* = \text{Enc}(m_1)$.

IND-CPA:

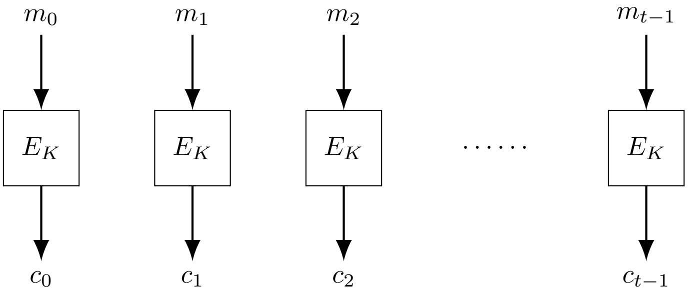
The adversary asks only encryption queries (encryption is randomized, or uses nonces).

IND-CCA:

The adversary asks encryption and **decryption** queries (but cannot decrypt c^*).

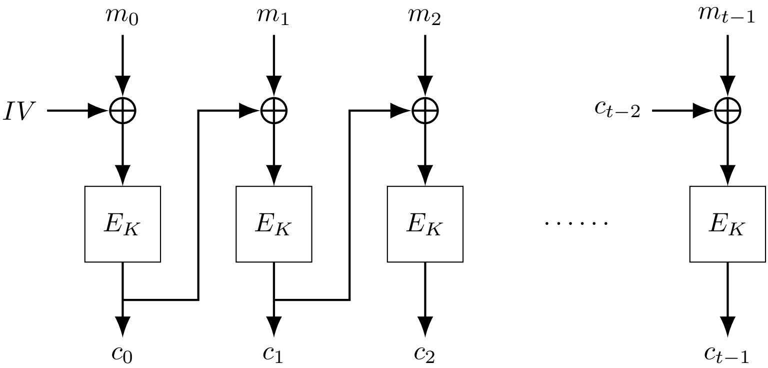
Encryption Modes

ECB (Electronic Codebook)



- Not IND-CPA (and simply a bad idea)

CBC (Cipher Block Chaining)



- IND-CPA security until $\mathcal{O}(2^{n/2})$ blocks queried (and time)
- You should rekey **much before that**

Gaps in proofs

Definition

A proof is **tight** $\iff$ it matches the best known (generic) attack (asymptotically at least).

- When the proof is tight, the problem is solved: we cannot prove more.
- When the proof is not tight, an uncertainty remains.
  (and where there is uncertainty, people can start making mistakes)

CBC attack: birthday bound

Encrypt $2^{n/2}$ times the same block 0, then $2^{n/2}$ times some secret block m. What happens?

CBC attack: birthday bound

Encrypt $2^{n/2}$ times the same block 0, then $2^{n/2}$ times some secret block m. What happens?

$\implies$ we can expect a collision between two ciphertext blocks: one from the first part (c_i), one from the second part (c'_j).

We have:

$$\begin{cases} c_i = 0 \oplus E_K(c_{i-1}) \text{ (encryption of 0 block)} \\ c'_j = m \oplus E_K(c'_{j-1}) \text{ (encryption of } m \text{ block)} \\ c_i = c'_j \end{cases}$$

$\implies E_K(c_{i-1}) \oplus E_K(c'_{j-1}) = m \implies$ get m.

CBC attack: proof limits

CBC is not IND-CCA:

- Encrypt m_b: $c = E_K(m_b \oplus IV), IV$
- Decrypt under $IV \oplus 1$: $E_K^{-1}(E_K(m_b \oplus IV)) \oplus (IV \oplus 1) = m_b \oplus 1$

IND-CPA protects confidentiality, but offers no security against tampering (& no authenticity) (CBC, CTR).

CBC attack: padding oracle

The PKCS7 padding norm fills the last message block with octets $0xi$, where i is the missing number of octets.

- $0x01$ is a valid 1-byte padding
- $0x02$ $0x02$ a 2-byte padding
- $0x03$ $0x03$ $0x03$ a 3-byte padding

Server side decryption (if badly implemented):

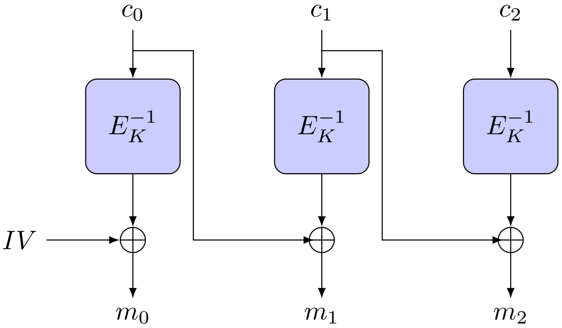
- CBC decrypt
- Check padding, abort if invalid
- Check authentication tag, abort if invalid

$\Longrightarrow$ the timing is different $\Longrightarrow$ **padding oracle** available

Padding attack: attacker submits ciphertext and learns if the last byte of plaintext is a valid pad.

CBC padding attack

Situation: the attacker has ciphertext $c = c_0, c_1, c_2$ and wants (the last byte of) m_1.



- Drop c_2
- Guess last byte of m_1 as g
- Change c_0 to $c_0' = c_0 \oplus (0 \| g \oplus 0x01))$ (last byte changed)
- If last byte $= g$: valid pad, otherwise invalid pad

CBC padding attack

(Note: assumes that the second-to-last byte is not $0x02$, otherwise there is another valid case $\implies$ the attack is a bit more complicated)

Next blocks

use a (0x02 0x02) pad, etc.

Authenticated Encryption and MACs

Recap

A MAC: $\{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$.

- Guarantees integrity
- Security based on unforgeability

Authenticated encryption:

$$\begin{cases} E \ : \{0,1\}^k \times \{0,1\}^* \times \underbrace{N}_{\text{Nonce / IV}} \to \{0,1\}^* \\ D \ : \{0,1\}^k \times \{0,1\}^* \times N \to \{0,1\}^* \cup \{\perp\} \end{cases}$$

$\perp \iff$ ciphertext rejected as **invalid**.

AE security
IND-CPA + **ciphertext integrity**: adversary cannot create a new ciphertext that decrypts correctly.

$$\text{AE security} \implies \text{CCA security}$$

Combiners

Encrypt and MAC

- Insecure: no AE security

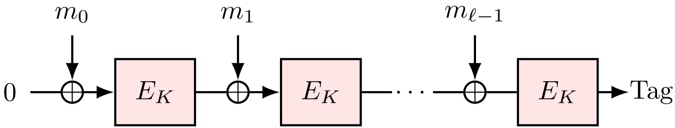
MAC-then-encrypt:

- May be insecure

Encrypt-then-MAC

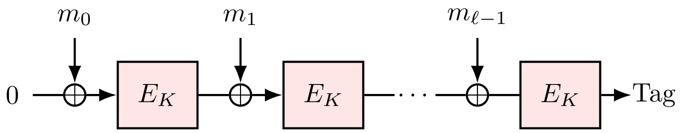
- Always the best choice
- MAC checked first, ciphertext discarded if invalid

CBC-MAC



This version of CBC-MAC is insecure.

CBC-MAC



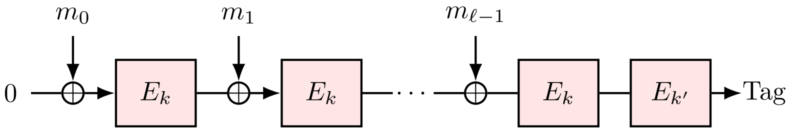
This version of CBC-MAC is insecure.

- MAC a message block m: get $t = E_K(m)$
- MAC $m \| (t \oplus m)$: get
  $t' = E_K(t \oplus m \oplus E_K(m)) = E_K(t \oplus m \oplus t) = E_K(m) = t$

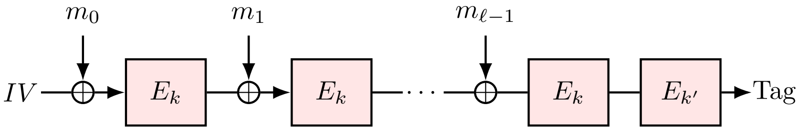
$\implies$ we can obtain a valid MAC without querying the message (breaks unforgeability)

ECBC-MAC

Solution: add another encryption call.



Caution: IV



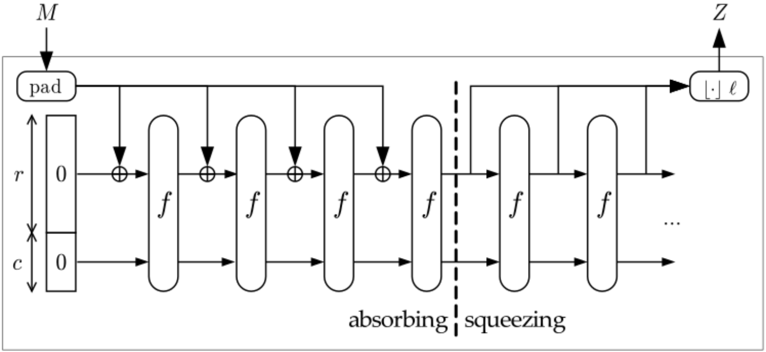
- MAC a message m with IV: get t
- t is a valid tag of $(m \oplus IV \oplus IV')$ for IV'

$\implies$ integrity not guaranteed

Solution: call the block cipher once, **then** XOR the first message block (if you really want to use an IV).

(Duplex) Sponges

The Sponge: hash functions



sponge

- f is a **cryptographic permutation**
- Speed of absorption determined by the **rate** r
- Security determined by the **capacity** c

Attacks (examples)

Collisions

Find two pairs of messages such that the inner part collides: $2^{c/2}$.

Preimages

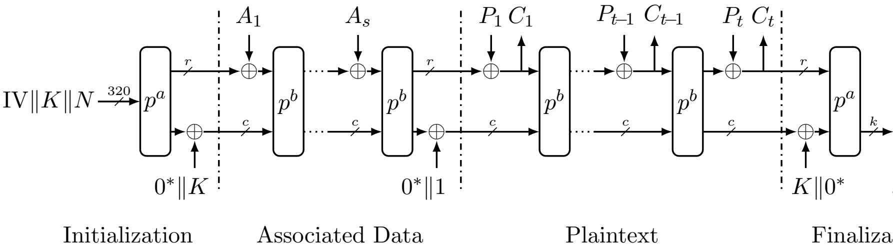
Compute forwards from the initial state and backwards from the output: try to collide on the inner part: $2^{c/2}$.

ASCON-AEAD

- Winner of the NIST lightweight cryptography competition
- Based on a **Duplex Sponge** mode

https://csrc.nist.gov/csrc/media/Presentations/2023/the-ascon-family/images-media/june-21-mendel-the-ascon-family.pdf

ASCON-AEAD



Parameters of ASCON-128

Security	128
Key	128
Rate	64
Capacity	256
Rounds (a,b)	12, 6

Caution

The mode is **nonce-based**: N should not be reused with different messages.

Ascon: decryption